Суперкомпьютерный консорциум университетов России
Российская академия наук

Суперкомпьютерные дни в России 2024

МГУ 270 1755 2025

Труды международной конференции
23–24 сентября 2024 г., Москва

Russian Supercomputing Days 2024

Septemder 23-24
Two Extremely
Parallel Days!

Суперкомпьютерный консорциум университетов России
Российская академия наук

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ

Труды международной конференции

23–24 сентября 2024 г., Москва

МАКС ПРЕСС

МОСКВА – 2024

УДК 519.7
ББК 22.18
С89

https://elibrary.ru/ykwpjq

Под редакцией члена-корреспондента РАН *Вл. В. Воеводина*

Суперкомпьютерные дни в России : Труды международной конференции. 23–24 сентября 2024 г., Москва / Под. ред. Вл. В. Воеводина. – Москва : МАКС Пресс, 2024. – 242 с.

ISBN: 978-5-317-07287-2 e- ISBN: 978-5-317-07283-4
https://doi.org/10.29003/m4296.978-5-317-07283-4

Данный сборник содержит полные статьи на русском языке, короткие статьи и аннотации стендовых докладов, включенных в программу Международной конференции «Суперкомпьютерные дни в России».

УДК 519.7
ББК 22.18

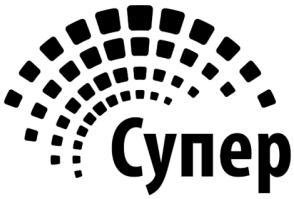
Russian Supercomputing Days : Proceedings of the International Conference. September 23–24, 2024, Moscow, Russia / Ed. by Vl. Voevodin. – Moscow : MAKS Press, 2024. – 242 p.

ISBN: 978-5-317-07287-27 e- ISBN: 978-5-317-07283-4
https://doi.org/10.29003/m4296.978-5-317-07283-4

The book contains full papers in Russian, short papers and poster abstracts included in the agenda of the International Conference “Russian Supercomputing Days”.

Подробную информацию о конференции можно найти в сети Интернет по адресу http://RussianSCDays.org

ISBN: 978-5-317-07287-27
e- ISBN: 978-5-317-07283-4
https://doi.org/10.29003/m4296.978-5-317-07283-4



Суперкомпьютерные дни в России

Полные и короткие статьи

Benchmarking BE-S1000 SoC by Baikal Electronics

M.N. Kaurkin[1,2], G.Y. Khrenov[1], V.V. Gusev[1]

[1]Baikal Electronics JSC
[2]Shirshov Institute of Oceanology, Russian Academy of Sciences

The aim of the article is to evaluate and demonstrate the performance of the BE-S1000 System-on-a-Chip (SoC), which contains 48 ARM Cortex-A75 cores operating up to 2.5 GHz, six DDR4-3200 memory channels and has a total TDP of 120 W. Testing was performed on single-socket and dual-socket systems using the HPL, SPEC CPU 2006 and SPEC CPU 2017 benchmarks. For this purpose, a software environment consisting of ARM-TF and UEFI boot loaders and Linux Debian OS was deployed on the systems based on the BE-S1000 SoC. The test results were compared with the results obtained on other servers based on Intel Xeon Gold 6230 and Kunpeng 920-4826 processors. From their analysis, we can draw conclusions about the competitiveness of the Russian SoC.

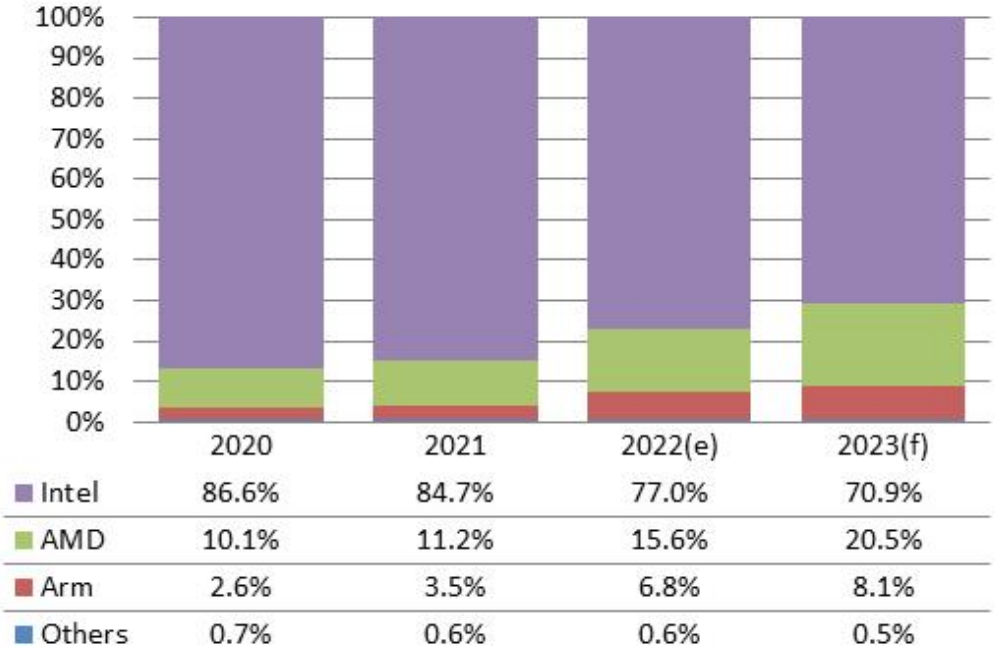
Keywords: parallel computing, Baikal Electronics, HPC, benchmarks, SPEC CPU, HPL.

1. Introduction

In 2020, a system based on Arm processors took first place in the Top500 supercomputer ranking for the first time. This is the *Japanese Fujitsu Fugaku*, whose performance on the High Performance Linpack (HPL) test turned out to be *415.5 petaflops*. Fugaku is powered by Fujitsu's 48-core *A64FX SoC*. At single or lower precision, which is often used for machine learning and artificial intelligence tasks, Fugaku's peak performance is over 1,000 petaflops (1 exaflops). The new system is installed at the RIKEN Center for Computational Science (R-CCS) in Kobe, Japan [1], [2].

Processors with Arm architecture by the end of 2023 occupied about 8.1% of the server market in physical terms, while the share of Intel processors dropped to 70% (see Figure 1). The main problem of processors with Arm architecture remains incompatibility with a significant part of the software infrastructure, but it is gradually being resolved. At the same time, manufacturers are attracted by the good energy efficiency and performance of Arm-based servers. Some data center operators and server brands have already started migrating to Arm-based solutions. *Amazon*, *Microsoft*, *Google*, *HPE* and *Alibaba* have already developed Arm-based products. *Nvidia* is now pushing its GPUs based on Arm architecture [6].

Development of server SoCs has also begun in our country. Developed by Baikal Electronics the BE-S1000 SoC became the first Russian ARM-based server SoC. In addition, it corresponds to modern foreign analogues in terms of performance indicators and a set of high-speed interfaces.



	2020	2021	2022(e)	2023(f)
Intel	86.6%	84.7%	77.0%	70.9%
AMD	10.1%	11.2%	15.6%	20.5%
Arm	2.6%	3.5%	6.8%	8.1%
Others	0.7%	0.6%	0.6%	0.5%

Figure 1: Server shipment share by CPU, 2020-2023. Source: DIGITIMES Research, February 2023.

The goal of this work is to benchmark the BE-S1000 SoC and then compare it with its competitors.

2. Technical Specifications

2.1. Main Features of BE-S1000

BE-S1000 is a SoC for general purpose servers that require high performance, low power consumption, and multiple PCIe Gen 4 interfaces with extensive customization options. It complies with Arm® Server Base System Architecture, IPMI, ACPI, and UEFI specifications. The SoC implements 48 Arm Cortex™-A75 cores operating up to 2.5 GHz. It supports L1, L2, L3, and L4 coherent caches. The BE-S1000 microprocessor implements six independent DDR4-3200 memory channels and a wide range of peripheral interfaces: PCIe Gen 4, 1 Gb Ethernet, USB 2.0, SPI, UART, GPIO, etc. The SoC provides three PCIe Gen 4 CCIX interfaces to build a multi-socket system up to four BE-S1000 microprocessors and with a joint L4 coherent cache. For more detailed information see [5].

2.2. Hardware for Benchmarking

The *Kunpeng 920-4826* and *Intel Xeon 6230* were chosen as processors for comparison. They have similar performance, cost, TDP (thermal design power), and interface set. I also note that the release date differs by a little more than two years, as shown in Table 1. The hardware configurations described in Table 2. Although the BE-S1000 supports DDR4-3200 memory, there is not yet a dual-processor board for it with properly routed memory running at 3200 MHz, so the memory frequency was reduced to 2400 MHz.

Table 1: Technical characteristics of processors BE-S1000, Kunpeng 920-4826, Xeon 6230

Processor	**Baikal BE-S1000**	**Kunpeng 920-4826**	**Intel Xeon Gold 6230**
ARCH	ARM Cortex-A75	ARM-based TaiShan v110	X86_64
Release date	**Q4'21**	**Q1'19**	Q2'19
Lithography, nm	16 nm	7 nm	14 nm
Cores	48	48	20
Threads	48	48	40
Core base Frequency, GHz	2.5	2.6	2.1
CPU GFLOPS perf., PEAK (DP)	480	499	1344
Cache memory	L2: 48x512 KB; L3: 12x2 MB; L4: 32 MB	L2: 48x512 KiB L3: 48 MiB	Intel® Smart Cache 27,5 MB
Max memory capacity	768 GB	1 TB	1 TB
Memory types	DDR4-3200	DDR4-2933	DDR4-2933
Max # of Memory Channels	6	8	6
PCI Express	Gen 4	Gen 4	Gen 3
TDP,W	120	150	125

Table 2: Technical Characteristics of Hardware for Benchmarking.

Board/Server	**DBS-OV**	**MBS-2S**	**TaiShan 200 (Model 2280)**	**Supermicro SYS-6019P-MTR**	**Supermicro 2029TP-HTR**
Microprocessor	BE-S1000		Kunpeng 920-4826	Intel® Xeon® Gold 6230	
Microprocessors #	1	2	2	2	2
Memory	6 x 64 GB DDR4-3200	12 x 16 GB DDR4-2400	16 x 32 GB DDR4-2933	8 x 64 GB DDR4-2933	12 x 32 GB DDR4-2933

2.3. Software for Benchmarking

For benchmarking, we selected *SPEC CPU 2006* [3] and *SPEC CPU 2017* [4] as industry-standardized CPU-intensive benchmark suites for measuring and comparing computing performance, and HPL-2.3 as a benchmark for the *TOP500* list.

All tests were compiled by *gcc-12* [9]. Linear algebra libraries for HPL is Arm Performance Libraries 22.0.2 [10] for Kunpeng and BE-S1000 or Intel oneAPI Math Kernel Library (oneMKL) version 2021.1 [11] for Xeon. Operating System is Ubuntu 22.04 (*ARM64* or *x86_64* respectively).

3. Result Discussion

3.1. High-Performance Linpack

High-Performance Linpack benchmark results are provided in Table 3. Intel Xeon 6230 demonstrates high performance on this test due to AVX (SIMD with 512-bit registers). Their registers are wider then the 128-bit NEON registers from the ARMv8 ISA. However, the Arm Cortex-A75 cores (BE-S1000 cores) are more efficient at floating point arithmetic than Kunpeng's custom cores, so the BE-S1000's result at 2.5 GHz is higher than the Kunpeng 920-4826 at 2.6 GHz.

3.2. SPEC CPU

SPEC CPU 2006 and SPEC CPU 2017 results are presented in Tables 4, 5, 6 and 7. All results presented for BE-S1000, Kunpeng, Xeon were obtained on the *GCC-12* compiler on a base run with a *-Ofast* option. These results are consistent with the official results for the dual-socket *TaiShan 200* server with *Kunpeng 920-7260* processor (64 cores, see [13], [14]). But the official results for Xeon were obtained on the Intel compiler (see the last two rows in the Tables 6 and 7, source [7], [8]). So the Intel Compiler results are 50% higher than the GCC results on Xeon (see fig. 2). Such results may not be reproducible and do not reflect real application performance, since most of the server (non-HPC) applications used are built by GCC or installed from Linux repositories. In addition, the use of proprietary compilers on servers in critical information infrastructure is not reasonable in our difficult times. The reproducibility of SPEC CPU results on Intel processors using a compiler is also regularly questioned [12]. BE-S1000 demonstrates 15%-20% lower performance than the Kunpeng 920-4826 on SPEC CPU 2017 in a single-socket configuration, and in a two-socket configuration the gap becomes larger due to the

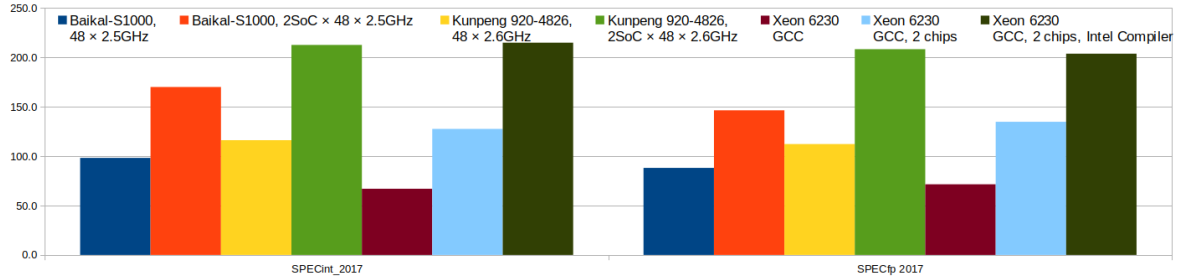


Figure 2: SPEC CPU 2017 rate results on BE-S1000, Kunpeng 920-4826, Xeon 6230

Table 3: High-Performance Linpack benchmark (HPL) results on BE-S1000, Kunpeng 920-4826, Xeon 6230

SPEC CPU 2006 int rate	**BE-S1000**	**BE-S1000, 2 SoC**	**Kunpeng 920-4826**	**Kunpeng 920-4826, 2 SoC**	**Xeon 6230 GCC**	**Xeon 6230 GCC, 2 CPU**
Mem	**6ch×64GB DDR4 3200**	**2×6ch×16GB DDR4 2400**	**8ch×32GB DDR4 2933**	**2×8ch×32GB DDR4 2933**	**4ch×64GB DDR4 2933**	**2×4ch×64GB DDR4 2933**
HPL result, all cores, GFLOPs	**353**	**599**	**302**	**596**	**849**	**1700**
PEAK PERF, All cores, GFLOPs	**480**	**960**	**499**	**998**	**1344**	**2688**

Table 4: SPEC CPU 2006 int rate results on BE-S1000, Kunpeng 920-4826, Xeon 6230

SPEC CPU 2006 int rate	**BE-S1000**	**BE-S1000, 2 SoC**	**Kunpeng 920-4826**	**Kunpeng 920-4826, 2 SoC**	**Xeon 6230 GCC**	**Xeon 6230 GCC, 2 CPU**
Mem	**6ch×64GB DDR4 3200**	**2×6ch×16GB DDR4 2400**	**8ch×32GB DDR4 2933**	**2×8ch×32GB DDR4 2933**	**4ch×64GB DDR4 2933**	**2×4ch×64GB DDR4 2933**
Threads #	**48**	**96**	**48**	**96**	**40**	**80**
Freq, GHz	**2.5**	**2.5**	**2.6**	**2.6**	**2.1**	**2.1**
400. perlbench	912.0	1657.6	1130.0	2077.7	671.9	1166.3
401, bzip2	477.0	803.6	570.0	1123.5	381.7	675.1
403. gcc	437.0	699.6	595.0	1207.6	480.6	916.0
429. mcf	266.0	444.7	321.0	670.4	311.4	604.9
445. gobmk	914.0	1612.4	1050.0	2104.2	551.7	901.7
456. hmmer	1290.0	2083.7	1070.0	3321.6	915.1	1617.5
458. sjeng	839.0	1535.8	1050.0	1953.7	560.3	931.2
462. libquantum	305.0	459.7	515.0	925.0	300.0	602.2
464. h264ref	1530.0	2819.9	1880.0	3702.7	1062.2	1787.8
471. omnetpp	258.0	415.1	277.0	526.6	267.9	532.0
473. astar	328.0	544.9	528.0	1005.1	328.6	596.6
483. xalancbmk	503.0	849.4	530.0	981.9	438.3	843.5
SPECint rate	**562.0**	**948.9**	**686.2**	**1375.6**	**475.5**	**863.0**
SPECint/GHz/core	**4.7**	**4.0**	**5.5**	**5.5**	**5.7**	**5.1**

Table 5: SPEC CPU 2006 fp rate results on BE-S1000, Kunpeng 920-4826, Xeon 6230

SPEC CPU 2006 fp rate	**BE-S1000**	**BE-S1000, 2 SoC**	**Kunpeng 920-4826**	**Kunpeng 920-4826, 2 SoC**	**Xeon 6230 GCC**	**Xeon 6230 GCC, 2 CPU**
Mem	**6ch×64GB DDR4 3200**	**2×6ch×16GB DDR4 2400**	**8ch×32GB DDR4 2933**	**2×8ch×32GB DDR4 2933**	**4ch×64GB DDR4 2933**	**2×4ch×64GB DDR4 2933**
Threads #	**48**	**96**	**48**	**96**	**40**	**80**
Freq, GHz	**2.5**	**2.5**	**2.6**	**2.6**	**2.1**	**2.1**
410.bwaves	376.6	634.0	591.3	1152.5	346.1	722.1
416.gamess	1135.1	2136.9	1188.0	2376.0	586.6	1070.7
433.milc	301.4	505.4	349.9	669.7	219.4	437.6
434.zeusmp	619.7	1059.5	755.8	1511.6	516.9	992.6
435.gromacs	932.7	1715.3	686.1	1372.2	526.3	898.7
436.cactusADM	575.8	1051.4	910.6	1821.2	481.2	919.3
437.leslie3d	239.0	406.5	387.1	774.2	240.1	458.1
444.namd	957.3	1805.9	820.6	1641.1	569.5	927.4
447.dealII	1253.4	2095.3	1325.1	2650.3	827.0	1502.1
450.soplex	244.5	424.2	393.7	787.5	291.6	580.3
453.povray	1364.6	2577.4	1217.6	2435.3	731.1	1345.8
454.calculix	940.4	1738.0	823.6	1647.1	303.4	510.3
459.GemsFDTD	222.5	375.9	357.8	715.7	227.1	452.2
465.tonto	872.3	1548.9	1067.8	2135.7	598.6	1105.6
470.lbm	209.1	366.9	398.5	796.9	222.3	445.5
481.wrf	438.1	738.2	677.1	1354.3	450.0	907.5
482.sphinx3	439.3	753.7	539.2	1078.5	444.8	864.8
SPECfp rate	**542.1**	**954.7**	**668.8**	**1332.2**	**409.7**	**772.2**
SPECfp/GHz/core	**4.5**	**4.0**	**5.4**	**5.3**	**4.9**	**4.6**

Table 6: SPEC CPU 2017 int rate results on BE-S1000, Kunpeng 920-4826, Xeon 6230 (GCC-12 and Intel Compiler)

SPEC CPU 2017 int rate	**BE-S1000**	**BE-S1000, 2 SoC**	**Kunpeng 920-4826**	**Kunpeng 920-4826, 2 SoC**	**Xeon 6230 GCC**	**Xeon 6230 GCC, 2 CPU**	**Xeon 6230 GCC, 2 CPU, Intel Compiler**
Mem	**6ch×64GB DDR4 3200**	**2×6ch×16GB DDR4 2400**	**8ch×32GB DDR4 2933**	**2×8ch×32GB DDR4 2933**	**4ch×64GB DDR4 2933**	**2×4ch×64GB DDR4 2933**	**2×6ch×32GB DDR4 2933**
Threads #	**48**	**96**	**48**	**96**	**40**	**80**	**40**
Freq, GHz	**2.5**	**2.5**	**2.6**	**2.6**	**2.1**	**2.1**	**2.1**
500.perlbench_r	109.3	199.7	140.9	256.5	75.9	141.5	166.4
502.gcc_r	71.8	111.0	100.8	182.0	64.5	130.9	175.9
505.mcf_r	45.2	69.8	68.9	120.2	42.9	86.9	283.6
520.omnetpp_r	40.1	64.4	50.2	80.9	40.6	82.7	142.6
523.xalancbmk_r	68.5	117.4	72.0	132.0	57.0	103.2	241.6
525.x264_r	246.4	454.4	309.9	605.1	151.5	275.9	428.9
531.deepsjeng_r	127.1	233.3	160.7	279.3	70.8	134.8	177.1
541.leela_r	151.5	287.4	162.7	312.2	70.1	135.1	170.4
548.exchange2_r	317.3	570.8	214.8	452.9	95.2	185.7	388.1
557.xz_r	56.9	99.0	72.6	131.6	52.4	89.2	145.7
SPECint_2017	**98.2**	**170.0**	**116.1**	**212.6**	**67.0**	**127.6**	**214.8**
SPECint/ /GHz/core	**0.8**	**0.7**	**0.9**	**0.9**	**0.8**	**0.8**	**2.6**

Table 7: SPEC CPU 2017 fp rate results on BE-S1000, Kunpeng 920-4826, Xeon 6230 (GCC and Intel Compiler)

SPEC CPU 2017 fp rate	**BE-S1000**	**BE-S1000, 2 SoC**	**Kunpeng 920-4826**	**Kunpeng 920-4826, 2 SoC**	**Xeon 6230 GCC**	**Xeon 6230 GCC, 2 CPU**	**Xeon 6230 GCC, 2 CPU, Intel Compiler**
Mem	**6ch×64GB DDR4 3200**	**2×6ch×16GB DDR4 2400**	**8ch×32GB DDR4 2933**	**2×8ch×32GB DDR4 2933**	**4ch×64GB DDR4 2933**	**2×4ch×64GB DDR4 2933**	**2×6ch×32GB DDR4 2933**
Threads #	**48**	**96**	**48**	**96**	**40**	**80**	**40**
Freq, GHz	**2.5**	**2.5**	**2.6**	**2.6**	**2.1**	**2.1**	**2.1**
503.bwaves_r	170.0	269.7	288.2	520.3	161.2	322.8	484.6
507.cactuBSSN_r	90.0	154.5	117.3	220.1	74.0	139.4	163.9
508.namd_r	119.0	222.6	116.3	229.7	82.5	143.9	160.0
510.parest_r	49.2	79.7	64.6	110.9	39.9	80.7	115.2
511.povray_r	171.0	258.5	189.7	401.8	135.9	239.9	231.7
519.lbm_r	17.6	27.3	31.3	60.9	17.2	34.2	116.3
521.wrf_r	77.6	121.2	108.0	197.4	76.0	151.6	211.0
526.blender_r	128.0	229.4	138.3	209.4	90.8	157.6	214.0
527.cam4_r	111.0	169.0	120.0	212.6	92.2	195.6	236.4
538.imagick_r	199.0	369.9	222.3	439.2	102.9	175.5	478.6
544.nab_r	179.0	339.9	150.1	308.0	131.1	217.5	353.4
549.fotonik3d_r	50.5	79.6	82.8	149.4	50.2	100.9	155.5
554.roms_r	36.1	57.6	59.7	109.3	31.9	62.9	94.1
SPECfp 2017	**88.1**	**146.4**	**112.2**	**208.3**	**71.4**	**134.8**	**203.7**
SPECfp/ /GHz/core	**0.7**	**0.6**	**0.9**	**0.8**	**0.9**	**0.8**	**2.4**

insufficient performance of 6-channel memory at 2400 MHz. In some SPEC CPU 2017 sub-tests (*548.exchange2_r*, *508.namd_r*, *544.nab_r*), BE-S1000 performs better than the Kunpeng 920-4826.

4. Conclusions

The SPEC CPU 2006 and 2017 and HPL benchmarks results presented and analyzed in this work show the competitiveness of the Baikal BE-S1000 SoC compared to Intel Xeon Gold 6230 and Kunpeng 920-4826 processors.

References

[1] The 55th edition of the TOP500
`https://top500.org/lists/top500/2020/06/`

[2] S. Kudo, K. Nitadori, T. Ina and T. Imamura, "Implementation and Numerical Techniques for One EFlop/s HPL-AI Benchmark on Fugaku," 2020 IEEE/ACM 11th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), GA, USA, 2020, pp. 69-76,
`https://doi.org/10.1109/ScalA51936.2020.00014`.

[3] The SPEC CPU® 2006 benchmark is SPEC's industry-standardized, CPU-intensive benchmark suite, stressing a system's processor, memory subsystem and compiler.
`https://www.spec.org/cpu2006/`

[4] The SPEC CPU® 2017 benchmark package contains SPEC's next-generation, industry-standardized, CPU intensive suites for measuring and comparing compute intensive performance, stressing a system's processor, memory subsystem and compiler.
`https://www.spec.org/cpu2017/`

[5] Processor Baikal-S (BE-S1000)
`https://www.baikalelectronics.ru/products/4395/`

[6] AMD expected to occupy over 20% of server CPU market and Arm 8% in 2023, according to DIGITIMES Research.
`https://www.digitimes.com/news/a20230217VL209/amd-arm-digitimes-research-intel-meet-the-analyst.html`

[7] SPEC CPU®2017 Integer Rate. Supermicro SuperServer 2029TP-HTR, Intel Xeon Gold 6230
`https://www.spec.org/cpu2017/results/res2019q2/cpu2017-20190318-11224.html`

[8] SPEC CPU®2017 Floating Point Rate. Supermicro SuperServer 2029TP-HTR, Intel Xeon Gold 6230
`https://www.spec.org/cpu2017/results/res2019q2/cpu2017-20190415-11997.html`

[9] GCC 12 Release Series
https://gcc.gnu.org/gcc-12/

[10] Arm Performance Libraries
https://developer.arm.com/Tools%20and%20Software/Arm%20Performance%20Libraries

[11] Intel oneAPI Math Kernel Library (oneMKL) version 2021.1
https://community.intel.com/t5/Intel-oneAPI-Math-Kernel-Library/Intel-oneAPI-Math-Kernel-Library-oneMKL-version-2021-1-is-now/m-p/1235480

[12] Over 2000 SPEC CPU 2017 Results Flagged for Compiler Optimization.
https://www.servethehome.com/over-2000-spec-cpu-2017-results-flagged-for-compiler-optimization-intel/

[13] SPEC CPU®2017 Integer Rate. Huawei TaiShan 200 Server (Model 2280) (2.6 GHz, Huawei Kunpeng 920 7260)
https://www.spec.org/cpu2017/results/res2020q2/cpu2017-20200529-22566.html

[14] SPEC CPU®2017 Floating Point Rate. Huawei TaiShan 200 Server (Model 2280) (2.6 GHz, Huawei Kunpeng 920 7260)
https://www.spec.org/cpu2017/results/res2020q3/cpu2017-20200621-22956.html

Fast Implementation of the Node2Vec

P. Plastova[1], A. Morkovkin[1], A. Sokolov[1,2]

[1]YADRO,
[2]Moscow State University

Node2Vec is a widely used method for learning feature representations for nodes in graphs. However, obtaining embeddings may require quite a lot of time. In this paper, we propose a number of optimizations for the one of the most popular and high performance Node2Vec C++ implementations. Our optimizations accelerate Node2Vec by 2.5-5.1 times for various graphs. We demonstrate the preservation of the accuracy of the optimized algorithm by solving a node classification problem with multiple labels based on the learned embeddings.

Keywords: node2vec, graph embeddings, graph representation learning, software optimization

1. Introduction

Many real-world systems can be described using a graph models. This data organization allows to concentrate on studying the relationships between the objects of the systems. Graphs can be used to display different data: from the interaction of proteins in biology and relationships among users of social networks to financial transactions and citations of scientific publications. The development of machine learning has greatly influenced approaches to data research, including the expansion of graph analysis capabilities. Representation learning is an important aspect of machine learning that allows to extract characteristic patterns from raw data and automatically create feature vectors for this data. The resulting representations can be used to solve applied tasks. Node2Vec[1] is a widely known member of this family of algorithms. Applying machine learning algorithms to vector representations of nodes or edges of a graph, in particular for solving node classification and link prediction problems, is a popular scenario for using Node2Vec embeddings [2, 3, 4]. Often such tasks occur in the context of the high-load systems, for example, recommendation systems that require a large amount of resources. In such conditions, the performance of the software is an important requirement.

There are two original implementations of the Node2Vec: a reference implementation in Python[1] and a high performance implementation in C++[2]. However, obtaining embeddings may require quite a lot of time. In this work, we aim for the speeding up the performance of the reference C++ Node2Vec implementation. We propose fast implementation of the Node2Vec and measure its performance for four graphs from different domains. Our optimized implementation achieves 2.5-5.1 times acceleration relative to the reference version. We also conduct experiments that show that reference and optimized implementations provide embeddings of the same quality.

[1]Code is available at https://github.com/aditya-grover/node2vec
[2]Code is available at https://github.com/snap-stanford/snap/tree/master/examples/node2vec

The rest of the paper is organized as follows. In section 2 we give a brief description of the Node2Vec algorithm, then in section 3 we describe our algorithm optimizations. In section 4, we describe the results of our experiments: we present the obtained acceleration of the Node2Vec, and also compare the accuracy of the algorithm before and after optimizations by solving the problem of multi-label node classification.

2. Node2Vec overview

Node2Vec learns low-dimensional representations for nodes in graph that maximize the likelihood of preserving network neighbourhoods of the nodes. Node2Vec uses a Skip-gram model originally invented as variant of the Word2Vec algorithm [5, 6]. Formerly, Skip-gram model was designed to learn feature representations for words in natural language processing. Node2Vec algorithm extends this model to graphs. The pseudocode for the Node2Vec, is given below (Algorithm 1). Node2Vec consists of two stages (see Figure 1): random walks generation to obtain sequences of nodes and optimization of neighbourhood preserving likelihood objective using Skip-gram model with negative sampling[6].

Algorithm 1: The Node2Vec algorithm

Node2Vec *(Graph G = (V, E, W), Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)*:
- π = PreprocessModifiedWeights(G, p, q)
- $G' = (V, E, \pi)$
- Initialize *walks* to Empty
- **for** *iter* = 1 **to** r **do**
  - **forall** nodes $u \in V$ **do**
    - *walk* = RandomWalk(G', u, l)
    - Append *walk* to *walks*
  - **end**
- **end**
- t = PrepareAliasProbTable(*walks*)
- M_{emb} = SkipGram($k, d,$ *walks*, t)

Output: M_{emb} - trained embeddings table

RandomWalk *(Graph G' = (V, E, π), Start node u, Length l)*:
- Inititalize *walk* to [u]
- **for** *walk_iter* = 1 **to** l **do**
  - $curr$ = *walk*[-1]
  - V_{curr} = GetNeighbours($curr, G'$)
  - s = AliasSample(V_{curr}, π)
  - Append s to *walk*
- **end**

Output: *walk* - generated walk

At the first stage, ordered sequences of nodes of the input graph are generated. Thus, the graph is represented as a text document: the generated node sequences are analogous to the sequences of words in the text. To do this, Node2Vec uses its own node neighbour

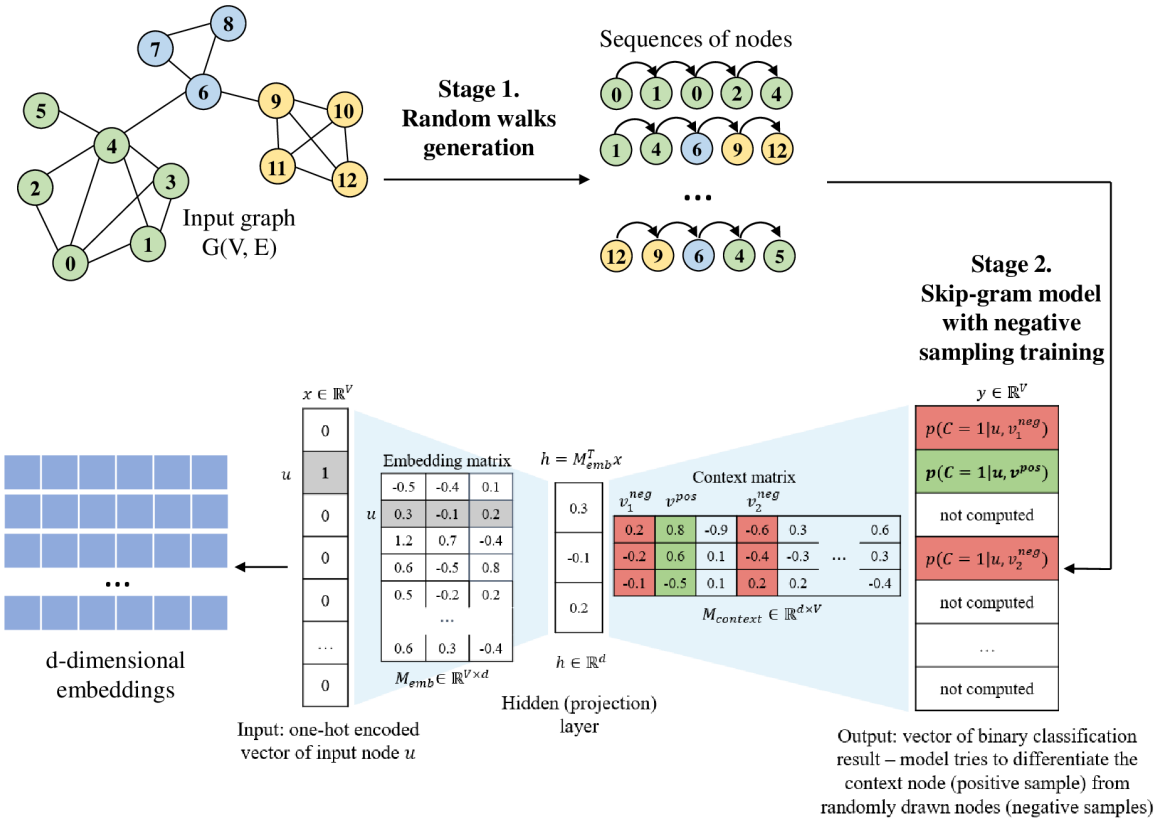


Figure 1: Illustration of the Node2Vec execution

selection strategy - a biased second order random walk model [1] with two parameters p and q. These parameters regulate whether the random walk will be performed locally and pass through nodes near the source node or will be investigated further from the source node. This search procedure performs "interpolation" between Breadth-first Sampling (BFS) and Depth-first Sampling (DFS). In considered implementation algorithm uses p and q parameters to modify the weights of the graph edges for further random walks sampling. As a result of this step, a given number of random walks of fixed length are simulated for each node of the graph.

At the second stage of the algorithm, the Skip-gram model is trained on the generated node sequences. In natural language processing Skip-gram model aims to learn word representations that are good for predicting the contextual words in a document for a given word. During the model training word embeddings are continuously modified to fit training set words distribution. For the Node2Vec algorithm Skip-gram model works similarly, but trains graph node embeddings as parts of sequences instead of word embeddings as parts of sentences.

A sliding window runs through each node sequence obtained in the previous stage (see Figure 2). The node located in the center of the window is considered the target node, and the other nodes located in the window before and after the target are its context. Skip-gram model uses a neural network with one hidden layer, consisting of d neurons, and optimizes weight matrices M_{emb} and $M_{context}$ that contain embeddings for the nodes (Figure 1). The one-hot encoded vector of the target node is sent to the input layer so the hidden-layer outputs h copy and transpose a row of the M_{emb} which associated

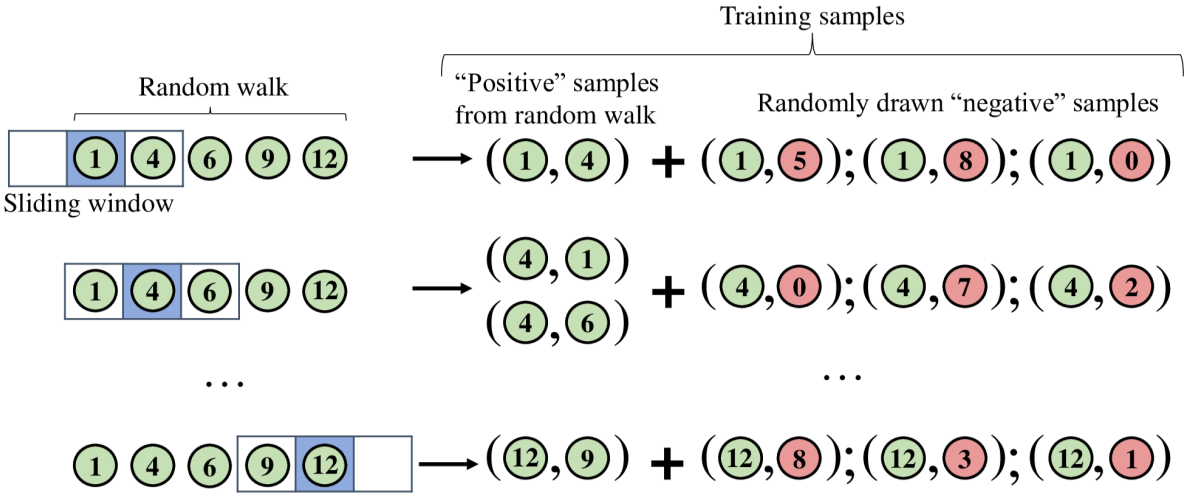


Figure 2: Extracting the positive and negative samples

with current target node. In the original Skip-gram model output layer is a probability distribution of all unique words in the corpus, which is determined using the softmax function, given a target word. The prediction error is calculated and summed across all context words for the target word. Based on this error, the row corresponding to the target word is updated in the weight matrix M_{emb}, and the weight matrix $M_{context}$ is completely updated. Calculating softmax and recalculating the weight matrix $M_{context}$ requires a lot of calculations, since both the number of unique words in the dictionary and the number of nodes in the graph can be very large.

Negative Sampling is used to make Skip-gram model calculations more efficient. With this approach, for each training pair, which consists of a target word and a word from its context and is called "positive" sample, n "negative" samples are randomly drawn from a noise distribution. Then the task of the Skip-Gram model is to distinguish positive pairs from negative ones by solving the problem of binary classification and using sigmoid function instead of softmax. The probability that the selected word is in the context of the target word is calculated $n + 1$ times. As a rule, values of the n are selected in the range of 5-20. As a result, the values in the matrix $M_{context}$ are updated only in $n + 1$ rows. After all the epochs of learning, the resulting nodes feature vectors are contained in the M_{emb} matrix.

3. Optimizations

3.1. Block random numbers generation

We analyzed the Node2Vec computations and determined that a significant amount of time is spent generating random numbers in the process of training the Skip-gram model, including for negative sampling. To overcome this problem, we implemented block random number generation. The pseudocode for original approach to generating random numbers in the Node2Vec is given below (Algorithm 2). The pseudocode for our block random numbers generation approach is also given below (Algorithm 3).

Algorithm 2: Reference Skip-gram training algorithm

```
1  SkipGram (Context size k, Dimensions d, Walks walks, AliasProbTable t):
2      do in parallel:
3      forall walks w ∈ walks do
4          forall nodes u ∈ w do
5              Generate random integer number offset in interval [0, k)
6              forall nodes v ∈ w inside the window of size 2 * (k − offset) around u do
7                  for iter = 1 to NegSamN + 1 do
8                      if iter == 1 then
9                          target_node = u
10                         Train embeddings of nodes from "positive" pair (v, target_node)
11                     else
12                         Generate random floating-point number x in interval [0, 1]
13                         Generate random floating-point number y in interval [0, 1]
14                         target_node = AliasMethod(x, y, t)
15                         Train embeddings of nodes from "negative" pair (v, target_node)
16                     end
17                 end
18             end
19         end
20     end
   Output: Trained embeddings table
```

Algorithm 3: Optimized Skip-gram training algorithm

```
1  SkipGram (Context size k, Dimensions d, Walks walks, AliasProbTable t):
2      Allocate memory RndF* and RndI* for each thread
3      do in parallel:
4      forall walks w ∈ walks do
5          Generate random float numbers and fill the RndF* memory belonging to the
            current thread with it
6          Generate random integer numbers and fill the RndI* memory belonging to the
            current thread with it
7          forall nodes u ∈ w do
8              Get offset value from RndI*
9              forall nodes v ∈ w inside the window of size 2 * (k − offset) around u do
10                 Train embeddings of nodes from "positive" pair (v, u)
11                 for iter = 1 to NegSamN do
12                     Get two floating-point numbers x and y from RndF*
13                     target_node = AliasMethod(x, y, t)
14                     Train embeddings of nodes from "negative" pair (v, target_node)
15                 end
16             end
17         end
18     end
   Output: Trained embeddings table
```

Considered implementation of Negative Sampling utilizes Alias method. This method provides capability to sample from a discrete probability distribution in the constant time. Here Alias method is used for sampling nodes, that will be taken as negative samples parts, corresponding to their frequency in generated random walks. It's worth to mention that Alias method needs a precomputed alias table for sampling. Implementation calculates alias table outside of Skip-gram, and the table is passed to the function as an argument.

The Skip-gram model is trained in multithreaded mode. Each thread performs Skip-gram training based on one random walk at a time. Having preserved the original random numbers generator, we generate and provide the threads with such a block of random integers and floating-point numbers in advance that covers the needs of the stream during training on a single walk.

3.2. Cycles splitting

As part of the negative sampling method, the Skip-gram model receives positive and negative samples for training. In the original Node2Vec implementation, training on all these pairs of nodes is implemented in one cycle (see Algorithm 2). We have divided this cycle into two stages: positive sample is processed first, and then negative samples are processed in a separate cycle (see Algorithm 3). This made it possible to reduce the number of calculations within cycles, including when calculating gradient values. We also carried out in different cycles updating the values of the auxiliary matrix $M_{context}$ and counting the changes for the matrix M_{emb} after calculating the gradient. This separation reduces computational time of the Node2Vec algorithm.

3.3. Alignment to processor cache lines

Many of the processors have hardware caches. Their purpose is to keep data closer to the cores to reduce data access time. Minimal portion of the data, that is possible to be fetched from the memory to the cache, is called cache line. CPU fetches cache lines from the memory and writes them back to the memory during processing the given data.

Original implementation of the algorithm uses method when processed embeddings are stored in continuous memory array side-by-side with each other. Example of the method is shown on the Figure 3. There are shown 3 embeddings stored under this scheme. Some

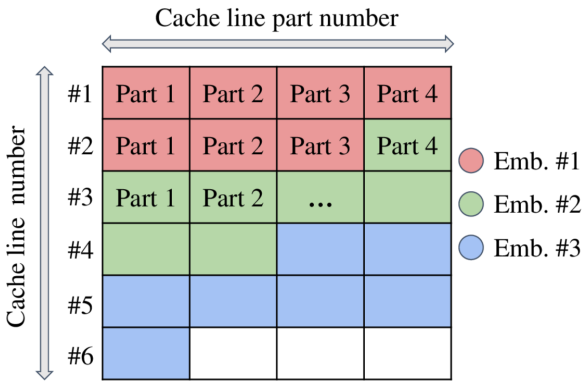


Figure 3: Original embeddings placement on the cache lines

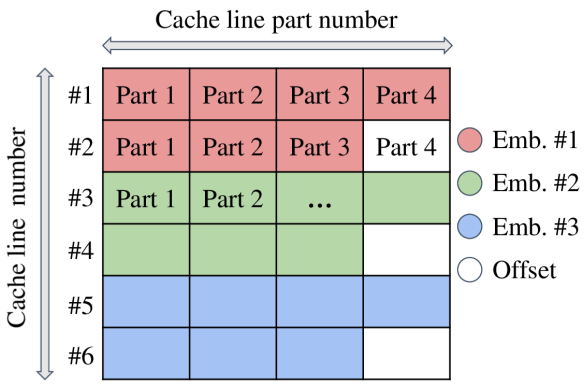


Figure 4: Aligned embeddings placement on the cache lines

of the neighbouring embeddings share cache lines located on the edges with each other. Therefore, as long as considered algorithm works in multithreaded mode, there may be the situation when two different cores of the CPU work on two embeddings which share the same cache line. It potentially brings us to the memory access pattern named as "false sharing". In this case two cores will continuously invalidate the same cache line in caches of each other during simultaneous processing that one. It may be cause of extra loading of the memory subsystem. As a result program requires more time for processing the data. Possible solution for the mentioned problem is to align each embedding in the matrices on cache line size. In other words, we should allocate number of cache lines for each embedding exclusively. Figure 4 shows application of the method to the previously mentioned example. This method eliminates "false sharing" problem by design.

As an implementation of the proposed method, we've created an embedding storage data structure where each embedding occupy own set of continuously stored cache lines in the memory.

4. Results

4.1. Experimental setup

To evaluate the performance of the original and our implementation of Node2Vec and check its accuracy, we use the following datasets:

- Protein-Protein Interactions (**PPI**) [7] - a subgraph of the PPI graph for Homo Sapiens. The subgraph node lazhang2019pronebels are derived from hallmark gene sets and represent biological states.
- **Wikipedia** [8] - a cooccurrence graph of words in the first million bytes of the Wikipedia dump. Node labels represent the Part of-Speech tags.
- **BlogCatalog** [9] - a social blogger graph listed on the BlogCatalog website. Node labels represent the bloggers' interests.
- **DBLP** [10] - an academic citation graph. Nodes represent authors, and labels represent their dominant conferences.

The characteristics of the datasets are presented in the Table 1.

Table 1: Datasets information

Dataset	**Number of nodes**	**Number of edges**	**Number of labels**
PPI	3 890	76 584	50
Wikipedia	4 777	184 812	40
BlogCatalog	10 312	333 983	39
DBLP	51 264	127 968	60

4.2. Experimental results

We measure the total running time of the algorithm, which includes graph reading, random walks generation, and Skip-gram model training. We perform training for a single epoch. We set values of parameters as follows: $d = 128, r = 80, l = 40, k = 10$. Values of the

p and q depend on the dataset. We compare the execution time of the reference C++ implementation and our optimized version. The results of the comparison are presented in the Table 2. The optimizations described above allowed us to achieve 2.5-5.1x acceleration on various datasets.

Table 2: Execution time of the reference and optimized implementations of the Node2Vec

Dataset	**Reference, sec**	**Optimized, sec**	**Ratio**
PPI	76	24	**3,2**
Wikipedia	86	35	**2,5**
BlogCatalog	218	66	**3,3**
DBLP	937	184	**5,1**

4.3. Embeddings evaluation

Node2Vec produces non-deterministic results so node embeddings differ between runs of the algorithm. For this reason, it is impossible to evaluate the quality of the optimized Node2Vec implementation by comparing the resulting embeddings directly with the embeddings obtained using reference implementation. To control the accuracy of the resulting node embeddings, we used a common approach that involves solving the problem of multi-label node classification [1, 11]. Each of the datasets is provided together with labels for its graph nodes. Labels determine classes that each graph node belongs to. As part of this task, it is assumed that each graph node has one or more labels from a finite set of labels. A certain proportion of nodes is provided for training a classifier to predict labels by graph node embedding. The remaining nodes are used for testing trained classifier. The selection of the nodes into these groups occurs randomly. We conduct training and testing 10 times, and then report the average value of Micro-F1 and Macro-F1. The resulting metric values are presented in Table 3 and Figure 5 and indicate that our optimizations didn't affect the accuracy of the Node2Vec algorithm.

Table 3: Micro-F1 and Macro-F1 scores for multi-label node classification

Dataset	**Training ratio**	**Micro-F1**					**Macro-F1**				
		0.1	**0.3**	**0.5**	**0.7**	**0.9**	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
PPI	Reference	15,8	19,4	20,7	21,7	22,3	11,5	15,7	17,3	18,2	18,0
	Optimized	16,5	19,7	21,0	21,7	22,1	12,5	16,3	17,7	18,5	18,3
Wikipedia	Reference	45,4	48,9	50,6	51,8	51,7	7,6	9,4	10,3	10,6	11,0
	Optimized	46,0	49,1	50,4	51,0	51,8	8,0	9,5	10,1	10,5	10,8
BlogCatalog	Reference	34,2	36,9	37,8	38,4	38,8	16,3	19,9	21,2	22,1	22,6
	Optimized	34,4	36,7	37,8	38,4	38,9	16,7	20,1	21,5	22,3	22,5
DBLP	Reference	58,6	59,9	60,1	60,3	60,4	5,7	5,9	5,9	6,0	6,0
	Optimized	59,2	60,0	60,3	60,4	60,4	5,9	6,1	6,1	6,1	6,1

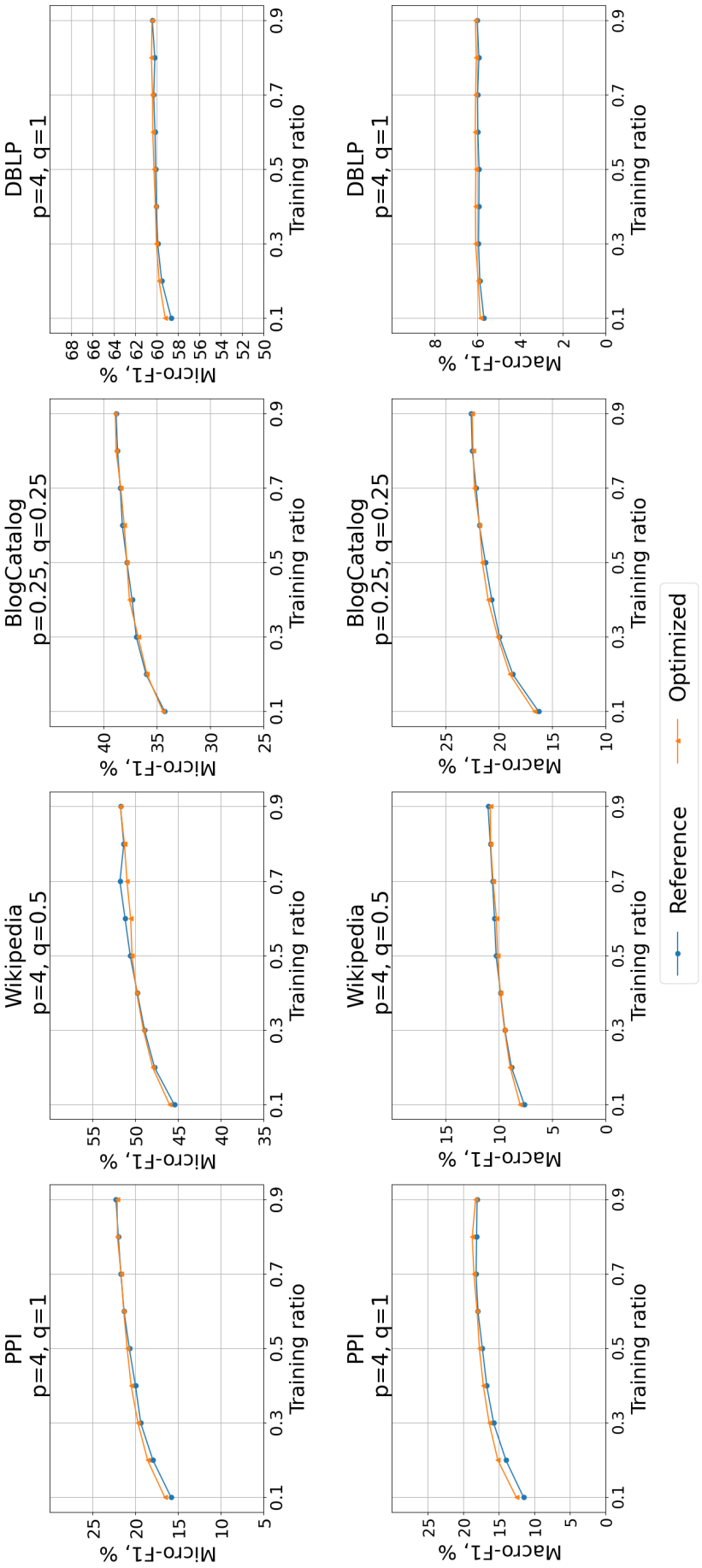


Figure 5: Evaluation of Micro-F1 and Macro-F1 scores on varying the proportion of labeled data used for training. The optimized implementation of the Node2Vec provides the node embeddings of the same quality as the reference implementation.

4.4. Comparison with other implementations

To compare our optimized implementation of the Node2Vec with other solutions, we measured the performance of three other open source implementations: eliorc/node2vec[1], shenweichen/GraphEmbedding[2], and VHRanger/nodevectors[3]. Algorithms parameters for the test were aligned with each other: $d = 128, r = 80, l = 40, k = 10$. The results of the comparison are presented in the Table 4.

Dataset	**Implementation**				
	VHRanger/ nodevectors	**eliorc/ node2vec**	**shenweichen/ GraphEmbedding**	**SNAP**	**Ours**
PPI	21	76	42	76	24
Wikipedia	49	377	140	86	35
BlogCatalog	139	1366	363	218	66
DBLP	132	360	348	937	184

Table 4: Execution time of different Node2Vec implementations

These results show that our implementation is one of the fastest implementations of Node2Vec algorithm. Our optimized implementation is faster than eliorc/node2vec and shenweichen/GraphEmbedding on all tested datasets. Geomean speedup relative to VHRanger/nodevectors on conducted experiments is 1.17.

5. Conclusions

In this paper, we propose an optimized implementation of the Node2Vec, a widely used graph representation learning algorithm. Our optimized implementation achieves acceleration up to 5.1 times compared to the reference Node2Vec C++ implementation while maintaining the quality of the generated node embeddings. As a future work, it seems possible to use the solutions found to optimize other representation learning algorithms.

References

[1] Grover A., Leskovec J. node2vec: Scalable feature learning for networks //Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. – 2016. – P. 855-864.

[2] Zhou H. et al. Internet financial fraud detection based on a distributed big data approach with node2vec //IEEE Access. – 2021. – Vol. 9. – P. 43378-43386.

[3] Kim M., Baek S.H., Song M. Relation extraction for biological pathway construction using node2vec //BMC bioinformatics. – 2018. – Vol. 19. – P. 75-84.

[1]https://github.com/eliorc/node2vec
[2]https://github.com/shenweichen/GraphEmbedding
[3]https://github.com/VHRanger/nodevectors

[4] Rosenthal G. et al. Mapping higher-order relations between brain structure and function with embedded vector representations of connectomes //Nature communications. – 2018. – Vol. 9. – No. 1. – P. 2178.

[5] Mikolov T. et al. Efficient estimation of word representations in vector space //arXiv preprint arXiv:1301.3781. – 2013.

[6] Mikolov T. et al. Distributed representations of words and phrases and their compositionality //Advances in neural information processing systems. – 2013. – Vol. 26.

[7] Breitkreutz B.J. et al. The BioGRID interaction database: 2008 update //Nucleic acids research. – 2007. – Vol. 36. – No. suppl_1. – P. D637-D640.

[8] Mahoney M. Large text compression benchmark. – 2011.

[9] Reza Z., Huan L. Social computing data repository. – 2009.

[10] Tang J. et al. Arnetminer: extraction and mining of academic social networks //Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. – 2008. – P. 990-998.

[11] Zhang J. et al. Prone: Fast and scalable network representation learning //IJCAI. – 2019. – Vol. 19. – P. 4278-4284.

[12] Walker A.J. New fast method for generating discrete random numbers with arbitrary frequency distributions // Electronics Letters. – 1974. – 10 (8): 127

On an algorithm for decomposing multi-block structured meshes for calculating dynamic wave processes in complex structures on supercomputers with distributed memory

I. Agrelov, N. Khokhlov, V. Stetsyu, S. Agibalov

Moscow Institute of Physics and Technology, Dolgoprudny City, Russia

Development of the oil and gas industry is one of the priority areas of the Russian Federation. Large hydrocarbons reserves are located in the Arctic region, which is difficult to explore. This paper is devoted to numerical calculation of dynamic impact propagation on an oil platform using parallel computing methods. To solve this problem, we used a grid-characterization method. The large amount of computation requires the use of parallel computing techniques such as MPI. We have built a grid model based on a real platform and developed an algorithm for decomposition of the computational domain to reduce the message time between MPI processes and increase acceleration. A series of test calculations were performed to show the capabilities of the algorithms. Examples of calculations and application of the developed method of decomposition are given. Decomposition and parallelization algorithms are currently under study. The conducted tests have shown the possibility of using our model for real calculations.

Keywords: Parallel calculation, Numerical modeling, Wave propagation, Multiple grids modeling, Parallel algorithm

1. Introduction

Numerical modeling of propagation of dynamic wave perturbations in solids is used to solve a wide range of problems. Such problems include seismic exploration, seismic stability, computational geophysics, and dynamic strength problems. The oil and gas sector of the industrial complex of the Russian Federation forms a significant share of the country's economy. The most important reserve of hydrocarbons are the fields of the North and the Arctic. The development of resources of the oil and gas complex increases the economic potential for the growth of modern infrastructure. Russia is actively developing the Arctic territories, including new oil and gas fields. According to open sources, about 13% of the world's oil reserves and up to 30% of the world's gas reserves are located in the Arctic zone. Despite the great prospects, the development of the Arctic shelf is severely hampered by the specifics of the location and natural environment. This creates risk factors that are absent in temperate latitudes, which requires new technologies and advances in this direction. One such direction is the development and research in the field of oil platforms.

Oil platforms are quite complex structures located on the Arctic shelf. One of the important factors is their resistance to dynamic loads. These disturbances can be caused by various factors, both anthropogenic and natural. Dynamic disturbances arise from seismic loads, collisions with various kinds of objects (ice, ship, etc.), explosive impacts (man-made and terrorist). This paper considers the issue of wave disturbance propagation in the structure of an oil platform.

A frequently used method for solving such problems is the grid-characteristic method [1, 2]. Due to the long computation time, parallel computing techniques, such as MPI, are required [3, 4]. The purpose of this paper is to create a grid decomposition method to reduce the communication time between processes and improve acceleration.

2. Equations

We denote the displacement vector at a point in space given by the radius vector $\vec{x}$ as $\vec{u}(\vec{x}, t)$. We also introduce a deformation tensor, $\varepsilon(\vec{x}, t)$, whose components are calculated from the displacement vector by the following rule

$$\varepsilon_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}), i, j \in \{1, 2, 3\} \tag{1}$$

The law of motion of each point of the medium can be written using Newton's second law in the following form

$$\rho \frac{\partial^2 u_i}{\partial t^2} - \sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_i} - f_i = 0, i \in \{1, 2, 3\} \tag{2}$$

where $f(\vec{x}, t) = (f_1, f_2, f_3)$ is a density of the field of forces acting on the medium, σ_{ij} is the stress tensor and $\rho = \rho(\vec{x}, t)$ is the density distribution of the medium material at each point.

Under the assumption of small deformations, the stress and small displacement tensors are related by Hooke's law

$$\sigma_{ij} = \sum_{k=1}^{3} \sum_{l=1}^{3} C_{ijkl} \varepsilon_{ij}, i, j \in \{1, 2, 3\} \tag{3}$$

The fourth rank tensor C_{ijkl} is a tensor of elastic constants that defines the relationship between strain and stress tensors. The fourth rank tensor has 81 components, but due to its symmetry, as well as the symmetry of the stress and strain tensors, it is described only by the 21 independent constants. Then the tensor C_{ijkl} can be written as a tensor of the second rank

$$C_{\alpha\beta} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{12} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{13} & C_{23} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{14} & C_{24} & C_{34} & C_{44} & C_{45} & C_{46} \\ C_{15} & C_{25} & C_{35} & C_{45} & C_{55} & C_{56} \\ C_{16} & C_{26} & C_{36} & C_{46} & C_{56} & C_{66} \end{bmatrix} \tag{4}$$

In the case of an isotropic linear-elastic medium, the equality (3) is greatly simplified. The number of independent variables is reduced to 2, and the elastic constant tensor (4) can be written as

$$C_{\alpha\beta} = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{5}$$

where λ and μ are the so-called elastic Lamé parameters.

Hooke's law (3) for this case can be written as

$$\sigma_{ij} = \lambda\delta_{ij}\sum_{k=1}^{3}\varepsilon_{kk} + 2\mu\varepsilon_{ij}, i, j \in \{1, 2, 3\} \tag{6}$$

where δ_{ij} is the Kronecker symbol.

Let us introduce the velocity vector $\vec{v}$ as the time derivative of the displacement vector $\vec{u}$. Then we can write the system of equations as follows:

$$\rho\frac{\partial\vec{v}}{\partial t} = (\nabla\cdot\sigma)^T + \vec{f}, \quad \frac{\partial\vec{\sigma}}{\partial t} = \lambda(\nabla\cdot\vec{v})I + \mu(\nabla\otimes\vec{v} + (\nabla\otimes\vec{v})^T) \tag{7}$$

3. Grid-characteristic method

Let $\mathrm{q} = [v_1, v_2, v_3, \sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{13}, \sigma_{23}]^T$. Then system 1.7 can be written in the form

$$\frac{\partial\mathbf{q}}{\partial t} - \mathbf{A_1}\frac{\partial}{\partial x_1}\mathbf{q} - \mathbf{A_2}\frac{\partial}{\partial x_2}\mathbf{q} - \mathbf{A_3}\frac{\partial}{\partial x_3}\mathbf{q} = 0 \tag{8}$$

We perform splitting by spatial coordinates, i.e., we consider separately 3 systems of the form

$$\frac{\partial\mathbf{q}}{\partial t} = \mathbf{A_i}\frac{\partial}{\partial x_i}\mathbf{q} \tag{9}$$

Since the system is hyperbolic, all matrices $\mathbf{A_i}$ have a complete set of eigenvalues and linearly independent eigenvectors, so diagonalization can be performed:

$$\mathbf{A_i} = \mathbf{\Omega_i^{-1}\Lambda\Omega_i} \tag{10}$$

where Ω_i consists of columns that are eigenvectors of A_i and Λ is a diagonal matrix consisting of eigenvalues.

Then we pass to the new variables ω_i by multiplying q by the matrix of eigenvectors Ω

$$\mathbf{w_i} = \mathbf{\Omega_i q} \tag{11}$$

Then the system will be written as:

$$\frac{\partial}{\partial t}\mathbf{w_i} + \mathbf{\Lambda_i w_i} = 0 \tag{12}$$

Since the matrix Λ is diagonal, this system consists of independent equations, each of which is a transport equation. In numerical simulations, finite-difference schemes are used to solve them.

4. Decomposition method

One of the goals of this work was to create an algorithm to reduce the communication time and increase the granularity of the decomposition. This algorithm is based on previously published papers [5, 6].

Let the model contain N rectilinear grids and M curvilinear grids, n_i is the number of nodes in the i-th rectilinear grid, m_j is the number of nodes in the j-th curvilinear grid and P is the number of MPI processes. The calculation of curvilinear grids takes more time, so we introduce an additional coefficient a. The algorithm consists of the following steps:

1) The number of nodes for each mpi process is calculated
$$M_{opt} = \frac{\sum_{i=1}^{N} n_i + a \cdot \sum_{j=1}^{M} m_j}{P}$$
2) The number of processes allocated to grids of size greater than M_{opt} is defined as the quotient of their size and M_{opt} rounded down.
$$P_i = \left[\frac{N_i}{M_{opt}}\right]$$
3) Large grids are divided between the processes selected on them by a simple geometric decomposition method.
4) The remaining grids are distributed among the processes using a greedy algorithm. The grids are organized into groups, initially there is only one grid in each group. Then at each step two smallest groups are merged into one. This is done until the number of groups is equal to the number of remaining processes.

Figure 1 shows an example of decomposition of the computational domain, the number of processors used in the example is 32, the color indicates the rank of the process. On the left is a standard decomposition in which each grid is distributed among all MPI processes. On the right is an example of decomposition using the developed algorithm. As can be seen from the figures, the decomposition algorithm allows to increase the granularity of the partitioning of the computational grids.

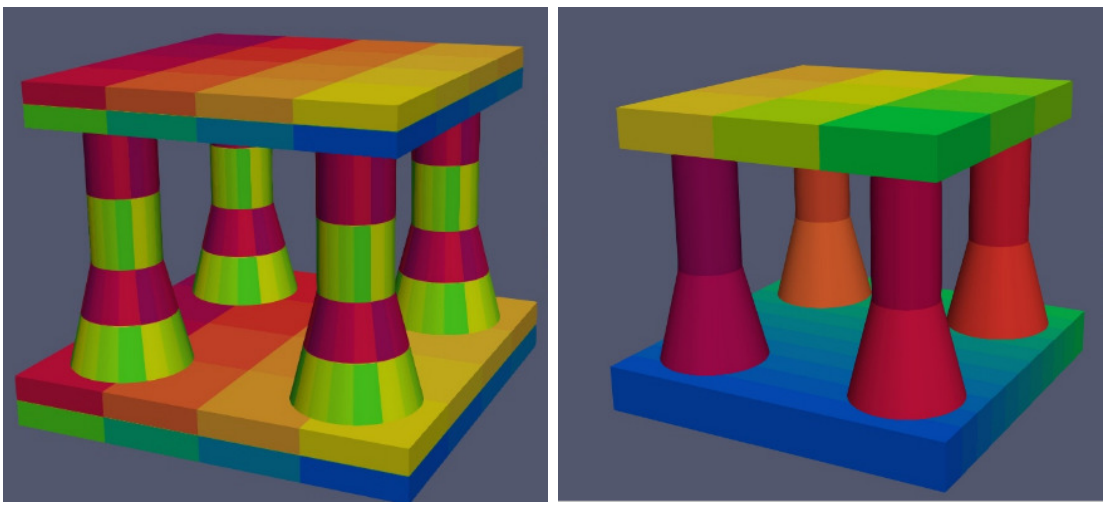


Figure 1: Decomposition of the computational domain. Color indicates the rank of the process.

5. Grid construction

The lower platform has a geometric shape largely similar to a rectangular parallelepiped. Due to the heterogeneity of the upper platform with the infrastructure and the lack of sufficient information about the dimensions of its parts, it was decided to approximate it by a rectangular parallelepiped as well.

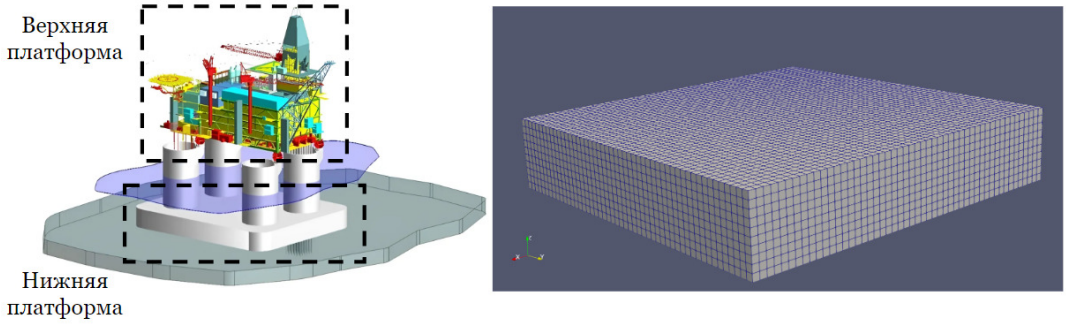


Figure 2: Construction of lower and upper platforms grids

In order to construct a rectangular grid to model the cylindrical supports, a block grid approach was used. The study area was divided into 5 parts. The central part is the same rectangular parallelepiped that was used to model the platforms of the structure (Figure 2). The remaining 4 parts are shells, one side of which coincides node to node with the central part, and the other side is a 90 degree arc of a circle (Figure 3).

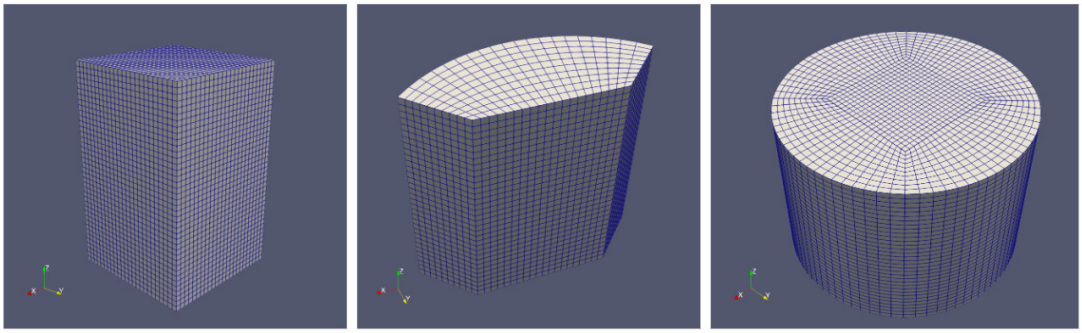


Figure 3: Construction of cylindrical supports grids

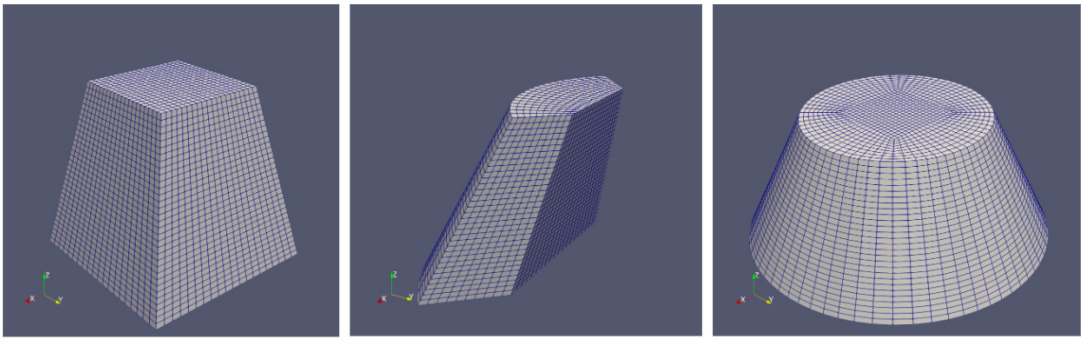


Figure 4: Construction of supports bases grids

It was decided to do the same with the column bases expanding in diameter. The central part is a truncated quadrilateral pyramid, and the geometry of the shells and the appearance of the resulting truncated cone is shown in Figure 4. The upper and lower parts of the columns at the junction of the grids also coincide node to node at the same radii.

6. Results

6.1. Calculation of disturbance propagation

To verify the program operation a series of calculations were performed on two-dimensional and three-dimensional grids. The figure 5 shows a slice of the column composed of the five grids discussed above. Also a two-dimensional calculation of the perturbation evolution is shown. The lateral face has reflection boundary conditions, which is clearly visible in the initial steps.

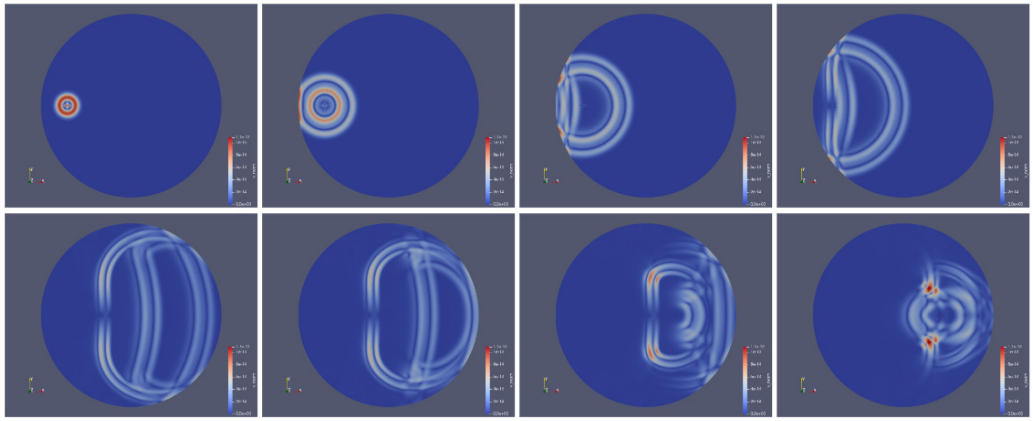


Figure 5: Example of calculations

Figure 6 shows a view of a column consisting of a cylinder and a truncated cone with coincident radii in the contact plane. The results of the calculation of the lateral impact on the column in section are shown below. The lateral surface of the column has reflecting boundary conditions, and the top and bottom bases are non-reflecting.

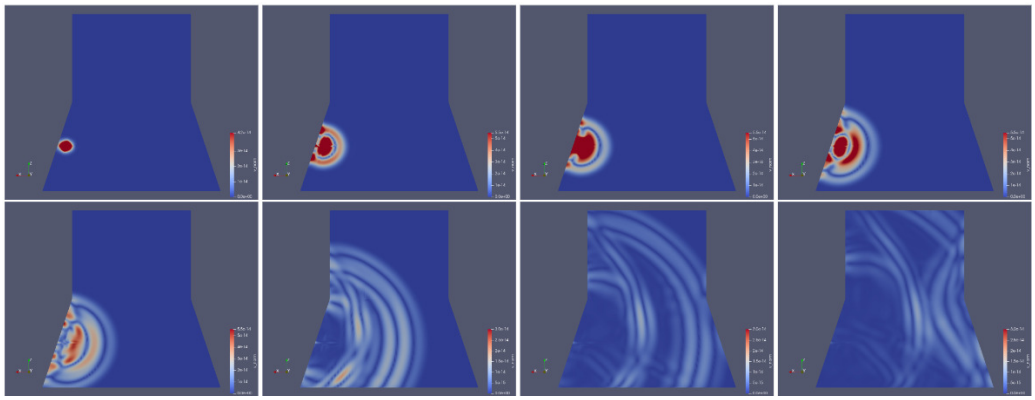


Figure 6: Example of calculations

Figure 7 shows the calculation of disturbance propagation in the oil platform structure.

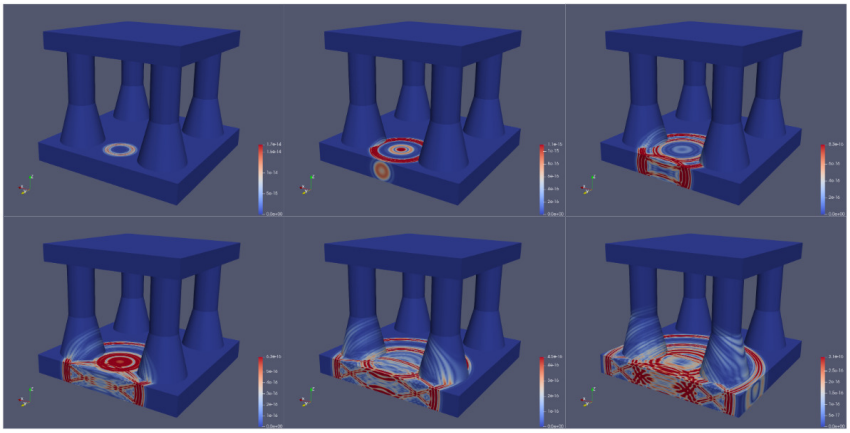


Figure 7: Example of calculations

6.2. Acceleration

To verify the effectiveness of the decomposition method, acceleration calculations were performed. The results are shown in Table 1 and Figure 8. As can be seen the use of the decomposition method does not give a strong gain in acceleration.

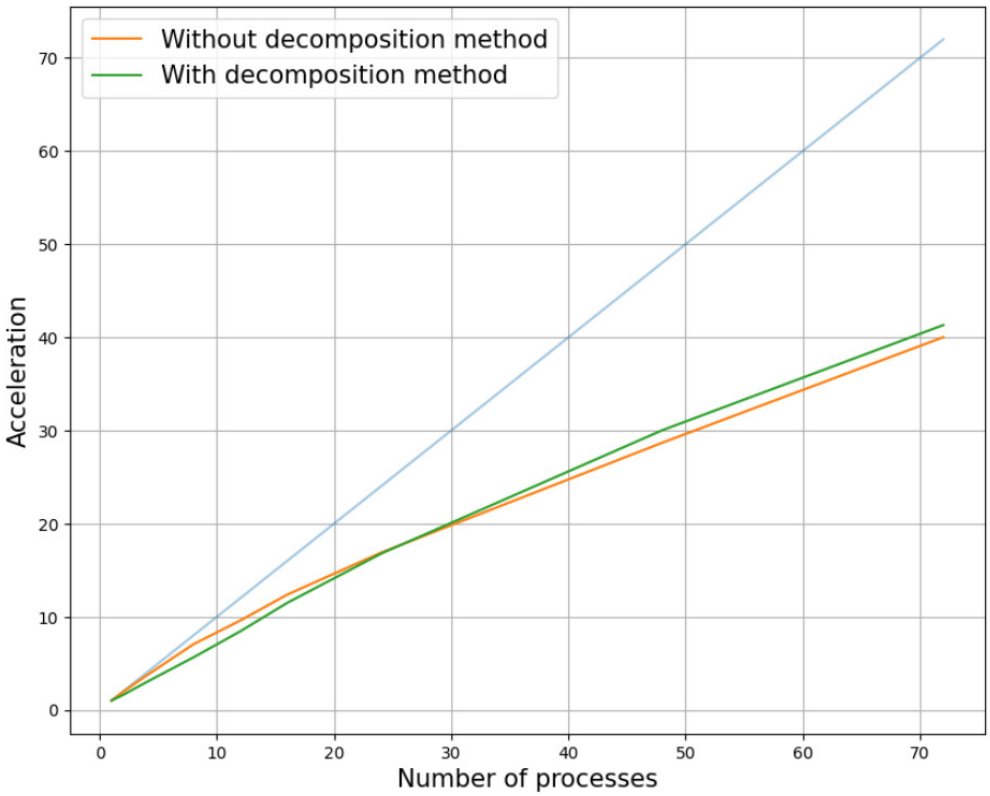


Figure 8: Graph of acceleration dependence on the number of processes

Table 1: Comparison of acceleration with and without decomposition algorithm

Number of processes	Acceleration without decomposition method	Acceleration with decomposition method
2	1.94	1.61
4	3.69	2.99
8	7.05	5.64
12	9.60	8.44
16	12.37	11.49
24	16.88	16.77
48	28.68	30.02
72	40.02	41.32

7. Conclusion

In this paper, a grid model of an oil platform was constructed and a decomposition method was developed to increase the granularity. During the test calculations no defects were detected, which makes this grid construction approach applicable to the problem. Tests have shown that the use of the developed method of grid decomposition does not give a big gain in acceleration in comparison with the default decomposition. The next step is to analyze the obtained results and improve the decomposition method.

Acknowledgements. The research was supported by the Russian Science Foundation grant no. 20-71-10028, `https://rscf.ru/project/20-71-10028/`. This work has been carried out using computing resources of the federal collective usage center Complex for Simulation and Data Processing for Mega-science Facilities at NRC "Kurchatov Institute", `http://ckp.nrcki.ru/`.

References

[1] Alena V Favorskaya et al. "Modelling the wave phenomena in acoustic and elastic media with sharp variations of physical properties using the grid-characteristic method". In: *Geophysical Prospecting* 66.8 (2018), pp. 1485–1502.

[2] Igor B Petrov, Alena V Favorskaya, and Nikolay I Khokhlov. "Grid-characteristic method on embedded hierarchical grids and its application in the study of seismic waves". In: *Computational Mathematics and Mathematical Physics* 57 (2017), pp. 1771–1777.

[3] Andrey M Ivanov, Nikolay Igorevich Khokhlov, et al. "Parallel implementation of the grid-characteristic method in the case of explicit contact boundaries". In: *Computer research and modeling* 10.5 (2018), pp. 667–678.

[4] Andrey M Ivanov and Nikolay I Khokhlov. "Efficient inter-process communication in parallel implementation of grid-characteristic method". In: *Smart Modeling for*

Engineering Systems: Proceedings of the Conference 50 Years of the Development of Grid-Characteristic Method. Springer. 2019, pp. 91–102.

[5] Vladislav Fofanov, and Nikolay Khokhlov. "Optimization of load balancing algorithms in parallel modeling of objects using a large number of grids". In: *Supercomputing: 6th Russian Supercomputing Days, RuSCDays 2020, Moscow, Russia, September 21–22, 2020, Revised Selected Papers 6*. Springer. 2020, pp. 63–73.

[6] Alena Favorskaya et al. "Parallel computations by the grid-characteristic method on Chimera computational grids in 3D problems of railway non-destructive testing". In: *Russian Supercomputing Days*. Springer. 2022, pp. 199–213.

Анализ производительности прямого решателя СЛАУ с разреженной матрицей на мини-кластере с процессорами архитектуры RISC-V[1]

А.Ю. Пирова, И.Б. Мееров

Нижегородский государственный университет им. Н.И. Лобачевского

Затраты времени и памяти при решении систем линейных алгебраических уравнений (СЛАУ) с разреженной матрицей существенно зависят от числа ненулевых элементов, возникающих в процессе факторизации матрицы системы. В работе рассматривается разработанный авторами программный комплекс DMORSy, решающий задачу нахождения перестановки, минимизирующей число ненулевых элементов фактора разреженной матрицы, а также его аналог – библиотека ParMETIS. В работе изучается влияние перестановок, полученных DMORSy и ParMETIS, на полное время решения СЛАУ в широко распространенном открытом решателе MUMPS. В отличие от аналогичных работ, анализ проводится на мини-кластере, построенном на базе устройств Lichee Pi 4A новой открытой архитектуры RISC-V.

Ключевые слова: прямое решение СЛАУ, переупорядочение разреженной матрицы, RISC-V.

1. Введение

Прямые методы решения разреженных систем алгебраических линейных уравнений (СЛАУ) широко применяются для численного моделирования физических процессов. Успешное применение таких методов требует использования важной предварительной процедуры – переупорядочения строк и столбцов исходной матрицы для сокращения числа новых ненулевых элементов, возникающих в процессе факторизации. Этот эффект характерен для разреженных матриц и называется заполнением. Задача минимизации заполнения является NP-трудной и решается эвристическими методами, основанными на методе минимальной степени и методе вложенных сечений. Наибольшее распространение для кластерных систем получил программный комплекс ParMETIS [3], который имеет большое число приложений и интегрируется с большинством известных решателей СЛАУ. В ННГУ авторами был разработан аналог – библиотека DMORSy, в которой реализована гибридная схема параллелизма, использующая технологии MPI+OpenMP. Библиотека DMORSy показывает результаты, близкие к ParMETIS по заполнению факторов матриц и времени работы [1]. Выводы о конкурентоспособности DMORSy подтверждались и другими исследователями [5].

[1]Работа поддержана Минобрнауки РФ, госзадание FSWR-2023-0034

В данной работе анализируется производительность переупорядочивателей ParMETIS и DMORSy и влияние результатов их работы на полное время решения СЛАУ, что важно для практического использования. Сравнение выполняется на первом публично доступном мини-кластере, построенном на базе устройств Lichee Pi 4A с четырехъядерными процессорами. В качестве решателя был выбран пакет MUMPS [4] – один из наиболее известных открытых прямых решателей, широко используемый в академических разработках, ориентированный на кластерные системы.

Насколько известно авторам, подобный анализ впервые проводится при использовании устройств новой открытой архитектуры RISC-V. Ранее авторами было выполнено аналогичное исследование для DMORSy, ParMETIS и MUMPS на процессорах стандартной архитектуры x86 [1]. Сейчас представляет интерес принципиальная возможность использования указанных пакетов на устройствах RISC-V, их производительность и воспроизводимость на RISC-V выводов, полученных ранее для x86.

Работа построена следующим образом. В п. 2 приведен обзор смежных работ по тестированию ПО на процессорах архитектуры RISC-V, в п. 3 описана стандартная схема прямого решения разреженных СЛАУ, в п. 4 описаны основные результаты статьи – сравнение работы библиотек ParMETIS и DMORSy по отдельности и в составе решателя СЛАУ. В п. 5 сформулированы выводы и направления дальнейшей работы.

2. Связанные работы

Архитектура RISC-V была представлена в Университете Калифорнии в Беркли в 2010 году, и стремительно завоевывает популярность как в академическом сообществе, так и в индустрии. Открытость, расширяемость, отсутствие патентных отчислений, простота и лаконичность системы команд являются отличительными особенностями данной архитектуры, во многом обеспечивающими ее быстрое распространение. На конец 2023 года на рынке пока еще не представлено высокопроизводительных устройств, однако их появление ожидается в 2024 году. Следует также отметить быстрое развитие соответствующего системного программного обеспечения [10]. Так, текущие версии компиляторов для RISC-V уже поддерживают глубокую оптимизацию кода, использование многопоточности, векторизацию (пусть и с рядом оговорок). Конечно, нельзя утверждать, что уровень развития устройств и системного ПО полностью сопоставим с x86. Компиляторы пока не достигли таких результатов оптимизации кода, как их аналоги для x86, наблюдается недостаток мощных средств профилирования и отладки, не хватает оптимизированных математических библиотек и пакетов численного моделирования. Однако развитие идет очень быстро. Отметим, например, что пакеты для тренировки и вывода нейросетевых моделей уже вполне успешно используют процессоры RISC-V [12]. Большое внимание уделяется оптимизации базовых алгоритмов с плотными и разреженными матрицами, поскольку они являются ключевыми вычислительно-затратными операциями во многих пакетах для моделирования. Например, в работе [6] сравнивается производительность различных реализаций BLAS, в [7]–[9] рассматриваются возможности аппаратной оптимизации устройств RISC-V для повышения производительности вычисления операции SpMV (произведение разреженной матрицы на плотный вектор). Однако вопрос об эффективной реализации операций BLAS и SparseBLAS для новой архитектуры пока остается открытым. Пример запуска большого пакета для численного

моделирования на RISC-V описан в [10]. Характерно также развитие академических инициатив, в частности, появление серии воркшопов по тематике высокопроизводительных вычислений на RISC-V на крупнейших международных конференциях, таких как ISC High Performance, EuroPar, PPAM, «Суперкомпьютерные дни в России». Учитывая заявленные планы выпуска высокопроизводительных процессоров, вопросы производительности пакетов численного моделирования на RISC-V-устройствах, уже сейчас вызывают интерес.

3. Процедура прямого решения разреженной СЛАУ

Пусть дана система линейных уравнений $Ax = b$, где A – симметричная действительная положительно определенная разреженная матрица, b – плотный вектор, x – вектор неизвестных. В решателе MUMPS для решения такой СЛАУ используется прямой метод в виде нахождения разложения

$$A = LDL^T \tag{1}$$

где L – нижнетреугольная матрица с единичной диагональю, называемая *фактором*, D – диагональная или блочно-диагональная матрица с блоками размера 1×1 и 2×2. Как правило, решение исходной СЛАУ выполняется в несколько этапов:

1. *Фаза анализа.* На этом этапе выполняют переупорядочение исходной матрицы A с целью уменьшения числа ненулевых элементов в факторе L. Это эквивалентно нахождению матрицы перестановки P и переходу к решению системы (2):

$$(PAP^T)(Px) = Pb \tag{2}$$

От полученной перестановки зависят не только затраты памяти и времени на выполнение наиболее трудоемкой численной фазы решения, но и потенциал ее параллелизма. Также, при необходимости, на этом этапе выполняется масштабирование значений элементов матрицы (*scaling*) и нахождение перестановки строк, которая перемещает большие по модулю элементы к диагонали. Затем выполняется *символьная факторизация*, включающая оценку объема памяти для хранения фактора L, определение расположения ненулевых элементов в L, построение вспомогательных структур данных.

В результате преобразований переходят к решению системы (3):

$$A_{pre}x_{pre} = b_{pre} \tag{3}$$

2. *Численная фаза.* На этом вычислительно трудоемком этапе выполняется численное разложение матрицы, получение разложения (1) для матрицы A_{pre}.

3. *Обратный ход* – решение треугольных систем.

$$LDy = b_{pre}, \quad L^T x_{pre} = y. \tag{4}$$

4. Постобработка решения. Анализ вычислительных ошибок.

Как правило, в зависимости от плотности матрицы, фаза анализа занимает около 10–20% общего времени решения системы, обратный ход – около 2–7%, остальное время занимает численная фаза решения.

Для выполнения переупорядочения в DMORSy и ParMETIS используется параллельный многоуровневый метод вложенных сечений. Подробно параллельный алго-

ритм, реализованный в DMORSy, описан в работах [1, 2]. Оба пакета, ParMETIS и DMORSy, ориентированы на кластерные системы, однако отличаются схемами распараллеливания. В частности, в пакетах по-разному выполняется распределение данных между процессами и выполняется обработка подзадач. ParMETIS не в полной мере поддерживает параллелизм на общей памяти: при работе внутри одного вычислительного узла можно использовать только несколько MPI-процессов. Такая схема распараллеливания часто работает приемлемо, а иногда не уступает схеме, основанной на комбинации технологий MPI+OpenMP. Однако в некоторых случаях запуск многих MPI-процессов на одном узле может приводить к увеличению накладных расходов из-за недостаточно эффективной схемы работы с данными. Авторы ParMETIS разработали пакет mt-METIS для работы на общей памяти, однако он не позволяет задействовать более одного узла. Пакет DMORSy позволяет гибко варьировать число используемых MPI-процессов и OpenMP-потоков, позволяя выбрать наиболее подходящий режим работы. Отметим, что в решателе MUMPS также реализован гибридный MPI+OpenMP параллелизм, в первую очередь, за счет рационального использования многопоточного BLAS.

4. Вычислительные эксперименты

4.1. Методика тестирования

Целью проведения вычислительных экспериментов было получить ответы на следующие вопросы:

- Возможно ли запустить библиотеки без существенной оптимизации и переработки кода?
- Какие размеры матриц приемлемы для прямого решения СЛАУ на одном и нескольких узлах?
- Какую масштабируемость можно получить для решателя и переупорядочивателей?
- Справедливы ли выводы, которые делались раньше на основе экспериментов на процессорах архитектуры x86?

Вычислительные эксперименты проводились на мини-кластере, состоящим из устройств Lichee Pi 4A. Конфигурация мини-кластера позволила исследовать гибридный MPI+OpenMP параллелизм, поскольку представляла собой четыре 4-ядерных вычислительных узла, соединенных гигабитной сетью. Такие условия приближены к будущим серверным конфигурациям. Решатель MUMPS содержит интегрированную версию ParMETIS, а также позволяет подавать на вход перестановку строк и столбцов, полученную другими средствами.

На первом этапе проведения экспериментов анализировалась производительность переупорядочивателей DMORSy и ParMETIS при работе на 1, 2 и 4 вычислительных узлах. На втором этапе переупорядочиватели использовались в решателе MUMPS для решения СЛАУ с симметричной матрицей. При этом ParMETIS вызывался через параметры MUMPS, а перестановки DMORSy передавались как пользовательские. Оба варианта вызова переупорядочивателя являются допустимыми. Отметим, что способ вызова переупорядочивателя влияет на параллелизм фазы анализа: в случае пользовательской перестановки фаза анализа в MUMPS выполняется после-

довательно, однако это не оказывает существенного влияния на общее время работы решателя. Во второй группе экспериментов результаты сравнивались по трем критериям: заполнение фактора матрицы, время получения перестановки, общее время решения системы.

В вычислениях использовалась система следующей конфигурации: процессоры CPU TH1520 (4 RISCV узла C910), частота 1.848 Ghz, память 8 GB, операционная система Debian 12. Использовались компиляторы gcc v. 13.2.0, gfortran; решатель MUMPS v. 5.6.2, библиотеки OpenBLAS v. 0.3.26 и ParMETIS v. 4.0.3 (версия, совместимая с MUMPS). Для тестирования были выбраны симметричные матрицы из коллекции SuiteSparse (https://sparse.tamu.edu/) порядком от 0.5 млн до 1.5 млн строк и заполнением порядка 1×10^{-6}-1×10^{-5} ненулевых элементов. В данной работе переупорядочиватели и решатель запускались без модификации кода, собирались скриптами, предоставленными разработчиками.

4.2. Сравнение работы переупорядочивателей

Сравним результаты работы переупорядочивателей ParMETIS и DMORSy. На первом этапе переупорядочиватели запускались отдельно от решателя, их параметры были зафиксированы по умолчанию. Был проведен эксперимент по подбору оптимального соотношения числа процессов и потоков для работы переупорядочивателей (рис. 1, рис. 2).

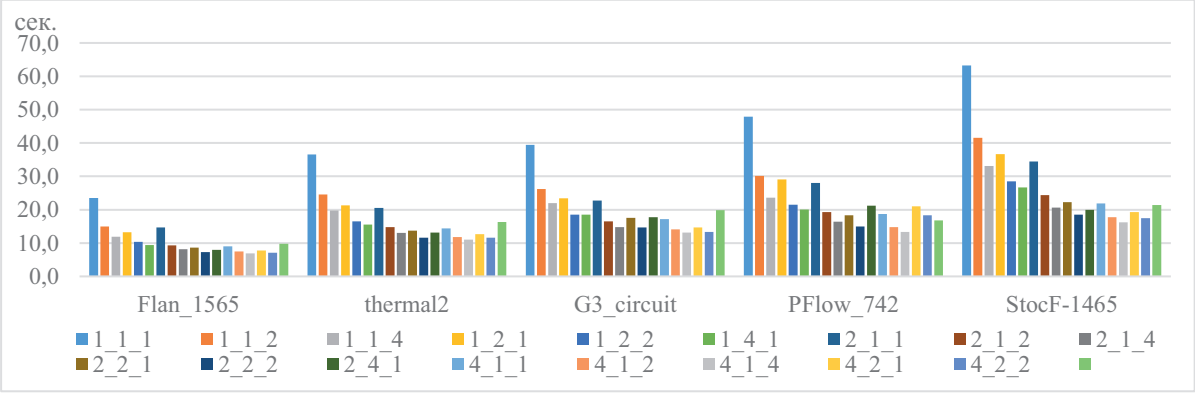


Рис. 1: Время работы DMORSy с параметрами по умолчанию при разном соотношении числа процессов и потоков, в секундах. Конфигурации обозначены X_Y_Z, где X – число узлов, Y – число процессов на узел, Z – число потоков на процесс.

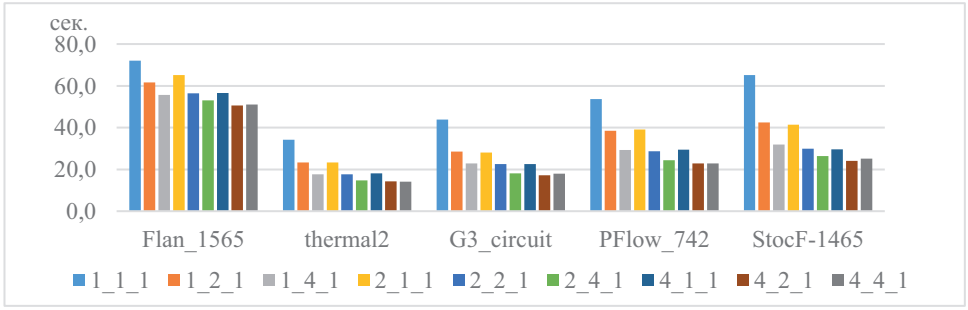


Рис. 2: Время работы ParMETIS при разном соотношении числа процессов и потоков, в секундах. Конфигурации обозначены X_Y_Z, где X – число узлов, Y – число процессов на узел, Z – число потоков на процесс. В связи с особенностями реализации ParMETIS Z=1.

Для DMORSy при работе на 2 и 4 узлах наилучшие по времени результаты были получены при соотношении 1 процесс по 4 потока на узел или 2 процесса по 2 потока на узел. Среднее ускорение составило 2,8 раз, наилучшее ускорение – 3,9 раз. Для ParMETIS лучшие результаты были получены при работе 4 процессов на узле, среднее ускорение составило 1,8 раз.

Далее при запусках DMORSy использовались параметры переупорядочения, дающие лучшее заполнение за приемлемое время работы. Сравнение проводилось в конфигурации 1 процесс по 4 потока на каждом узле мини-кластера, для дальнейшего использования в решателе. В таблице 1 показаны время работы и заполнение фактора DMORSy и ParMETIS, использовавшиеся в решателе.

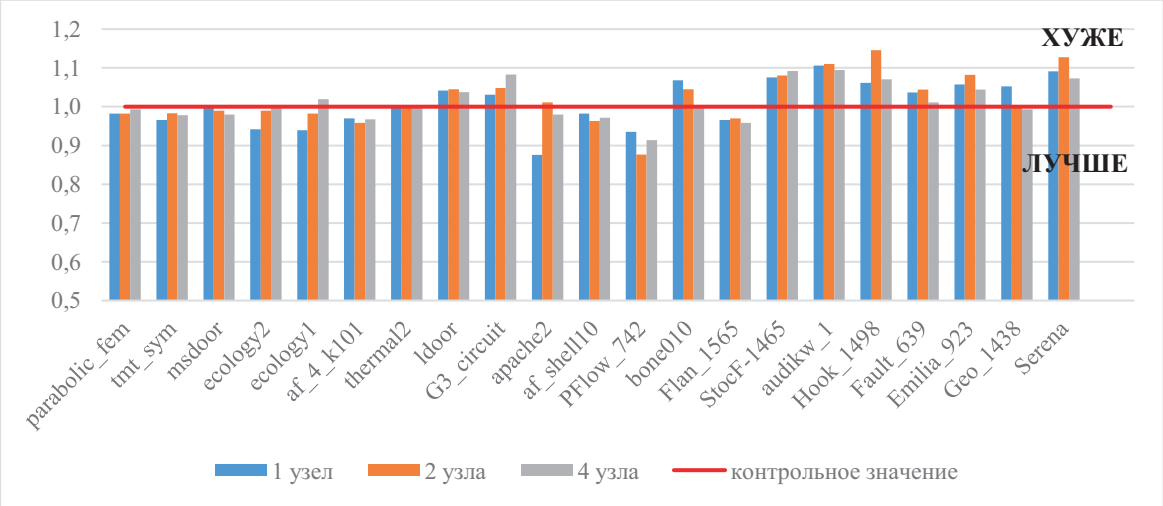


Рис. 3: Заполнение фактора, полученное DMORSy, относительно ParMETIS, разы. Значения выше контрольной линии – DMORSy уступает ParMETIS, ниже – DMORSy опережает ParMETIS.

Для устранения неоднозначности, заполнение указано по оценке MUMPS, что несколько больше, чем при точном подсчете числа ненулевых элементов, ранее применявшемся в наших работах, а также в работах других исследователей.

На рис. 3 показано отношение заполнения фактора в результате применения перестановок DMORSy и ParMETIS. Из графиков видно, что на половине тестовых матриц DMORSy незначительно опережает ParMETIS по заполнению фактора (в среднем, на 3%). На других матрицах DMORSy уступает ParMETIS, в среднем, на 6%. DMORSy дает лучшие перестановки на более разреженных матрицах.

Сравним время работы переупорядочивателей (таблица 1, рис. 4). Из результатов экспериментов видно, что DMORSy работает значительно быстрее ParMETIS, наибольшая разница наблюдается при работе на двух и четырех вычислительных узлах, где DMORSy работает быстрее в 3,7 и 2,4 раза соответственно. Отметим, что ParMETIS практически не масштабируется на процессорах архитектуры RISC-V, на половине тестовых матриц ParMETIS замедляется на двух узлах и лишь на 2/3 тестовых матриц ускоряется в среднем в 1,7 раз. Среднее ускорение DMORSy составляет 1,6 раз. Отметим, что ускорение DMORSy с параметрами переупорядочения по умолчанию было выше, в среднем составляло 2,6 раз, на 8 матрицах – более 3 раз.

Таблица 1: Результаты работы ParMETIS и DMORSy.
Обозначения: fnz – заполнение фактора (число ненулевых элементов), t – время в секундах.

	ParMETIS						**DMORSy**					
	1 узел		**2 узла**		**4 узла**		**1 узел**		**2 узла**		**4 узла**	
матрица	**fnz**	**t**	**fnz**	**t**	**fnz**	**t**	**fnz**	**t**	**fnz**	**t**	**fnz**	**t**
parabolic_fem	3,2E+07	9,86	3,3E+07	9,77	3,2E+07	8,20	3,2E+07	10,09	3,2E+07	7,93	3,2E+07	10,75
tmt_sym	3,9E+07	12,92	3,9E+07	9,69	3,9E+07	8,05	3,8E+07	12,77	3,9E+07	8,26	3,8E+07	6,86
msdoor	5,9E+07	1,86	5,9E+07	7,94	6,0E+07	7,13	5,9E+07	1,15	5,8E+07	1,25	5,9E+07	1,50
ecology2	5,0E+07	15,42	4,8E+07	12,13	4,9E+07	8,82	4,7E+07	16,02	4,7E+07	9,89	4,9E+07	8,25
ecology1	4,8E+07	15,43	4,9E+07	12,23	4,9E+07	8,94	4,6E+07	14,45	4,8E+07	9,72	5,0E+07	8,25
af_4_k101	1,0E+08	2,42	1,0E+08	7,05	1,0E+08	5,88	1,0E+08	1,51	1,0E+08	1,40	1,0E+08	1,75
thermal2	6,7E+07	24,71	6,7E+07	18,02	6,7E+07	12,79	6,7E+07	25,88	6,7E+07	16,67	6,7E+07	13,20
ldoor	1,6E+08	5,15	1,6E+08	17,48	1,6E+08	14,28	1,6E+08	3,11	1,7E+08	2,44	1,6E+08	4,36
G3_circuit	1,2E+08	35,18	1,2E+08	25,49	1,2E+08	16,87	1,2E+08	25,35	1,3E+08	16,69	1,3E+08	14,36
apache2	1,5E+08	16,46	1,4E+08	12,17	1,6E+08	8,57	1,3E+08	16,74	1,5E+08	8,85	1,5E+08	7,01
af_shell10	3,7E+08	9,34	3,7E+08	22,99	3,7E+08	16,66	3,6E+08	5,47	3,6E+08	4,06	3,6E+08	4,05
PFlow_742	5,2E+08	39,13	5,3E+08	33,38	5,3E+08	20,55	4,9E+08	28,60	4,6E+08	20,22	4,9E+08	15,81
bone010	1,1E+09	22,47	1,1E+09	34,28	1,1E+09	24,05	1,2E+09	10,37	1,2E+09	7,74	1,1E+09	9,02
Flan_1565	1,6E+09	37,93	1,6E+09	55,78	1,6E+09	38,05	1,5E+09	14,63	1,5E+09	9,61	1,5E+09	7,72
StocF-1465	1,1E+09	49,32	1,1E+09	35,21	1,1E+09	20,97	1,2E+09	41,53	1,2E+09	22,19	1,2E+09	19,48
audikw_1	1,3E+09	28,84	1,3E+09	47,29	1,3E+09	31,37	1,4E+09	9,93	1,4E+09	6,88	1,4E+09	8,87
Hook_1498	1,6E+09	39,16	1,6E+09	44,50	1,6E+09	29,03	1,7E+09	14,43	1,8E+09	12,16	1,7E+09	14,79
Fault_639	1,1E+09	12,88	1,2E+09	18,52	1,2E+09	12,73	1,1E+09	4,91	1,2E+09	3,74	1,2E+09	3,91
Emilia_923	1,7E+09	18,54	1,7E+09	24,58	1,7E+09	17,89	1,8E+09	8,06	1,9E+09	5,29	1,8E+09	6,62
Geo_1438	2,5E+09	30,99	2,6E+09	38,67	2,6E+09	26,93	2,6E+09	10,81	2,6E+09	7,76	2,5E+09	7,83
Serena	2,8E+09	35,49	2,8E+09	42,98	2,8E+09	29,91	3,0E+09	12,47	3,2E+09	8,03	3,0E+09	9,90

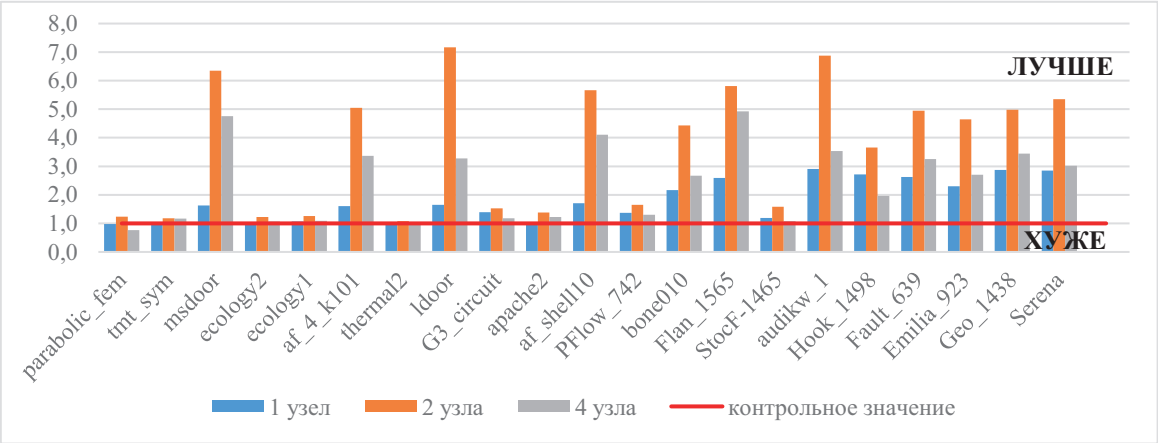


Рис. 4: Время работы DMORSy относительно ParMETIS, разы. Значения выше контрольной линии – DMORSy опережает ParMETIS, ниже – DMORSy уступает ParMETIS.

4.3. Сравнение времени работы решателя

Сравним время работы и масштабируемость решателя MUMPS с перестановками ParMETIS и DMORSy. На рис. 5 отображено общее время работы MUMPS с переупорядочивателем для наибольших матриц из тестового набора. На рис. 6 показано отношение времени работы MUMPS с DMORSy ко времени работы MUMPS с ParMETIS, для всех тестовых матриц.

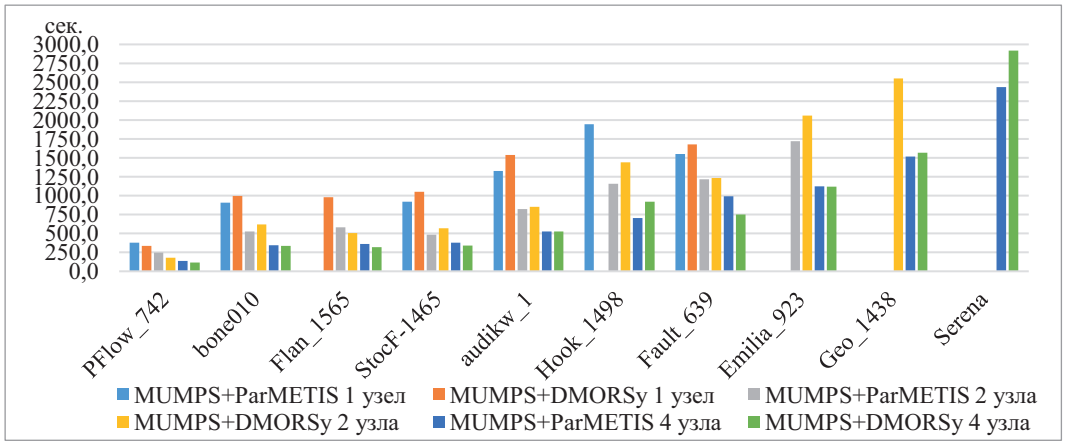


Рис. 5: Время работы MUMPS на больших матрицах из тестового набора (в секундах).

Из результатов экспериментов можно сделать следующие выводы:

- Объема памяти одного вычислительного узла достаточно для получения перестановки на матрицах порядка до 1,5 млн. вершин и заполнения до $\sim 5,5 \times 10^{-5}$. На трех матрицах (Emilia_923, Geo_1438, Serena) не удалось решить СЛАУ на одном вычислительном узле из-за нехватки памяти.
- На маленьких матрицах из тестового набора DMORSy дает перестановки, близкие по заполнению фактора к ParMETIS, *за меньшее время работы*. Для большинства из этих матриц при работе на 1м узле время решения СЛАУ с ParMETIS и DMORSy близко, а с увеличением числа узлов MUMPS с DMORSy работает быстрее: на двух узлах – в среднем, на 12%, на четырех узлах – в среднем, на 19%.

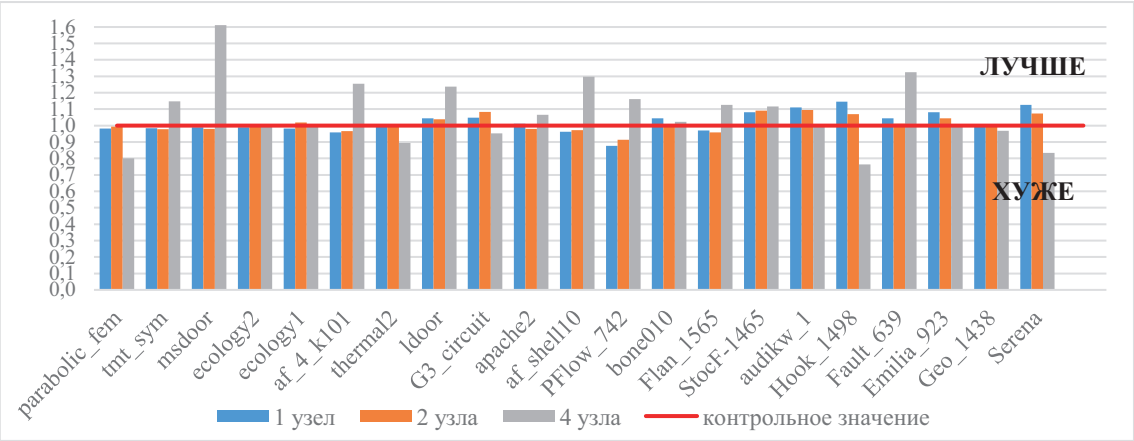


Рис. 6: Опережение по времени работы MUMPS с DMORSy относительно MUMPS с ParMETIS, разы. Значения выше контрольного значения – результаты с использованием DMORSy лучше, чем с использованием ParMETIS, ниже – хуже, чем с использованием ParMETIS.

- На самых больших матрицах из тестового набора DMORSy дает худшие по заполнению фактора перестановки, чем ParMETIS, и, как следствие, приводит к замедлению MUMPS при работе на одном и двух вычислительных узлах. При этом при работе на 4х вычислительных узлах общее время решения СЛАУ с DMORSy для большинства матриц лучше, чем с ParMETIS, в том числе на больших матрицах.
- На тех матрицах, где DMORSy дает худшее заполнение фактора и время получения перестановки значительно меньше времени численной факторизации, проигрыш в заполнении фактора влечет за собой проигрыш в общем времени решения системы. Эти результаты соотносятся с ранее проведенными экспериментами для процессоров архитектуры x86.
- Масштабируемость общего времени решения СЛАУ с DMORSy лучше, чем с ParMETIS. Для больших матриц ускорение MUMPS с DMORSy на 4х узлах составило в среднем 2,8 раз, для маленьких – 1,7 раз. Для MUMPS с ParMETIS среднее ускорение составляет 2,4 и 1,6 раз соответственно. При появлении более производительных процессоров архитектуры RISC-V, а также основанных на этих процессорах кластеров, результат может измениться в связи с существенным ускорением вычислений на одном узле. Однако при этом появится возможность решать СЛАУ большего размера, и использование кластера по-прежнему будет оправданным.
- Отметим, что в сравнении с результатами решения СЛАУ на процессорах архитектуры x86 [1], на RISC-V наблюдается другое соотношение между временем работы этапов решения СЛАУ. Так, на большинстве рассмотренных маленьких матриц при работе на одном узле архитектуры RISC-V время получения перестановки сравнимо со временем численной факторизации, а при работе на четырех вычислительных узлах – превышает его. Для x86 на этих матрицах характерно соотношение 1:2 между временем работы переупорядочивателя и численной факторизации. Напротив, на матрицах порядка более 1 млн. строк и более плотных, время работы переупорядочивателя на процессорах архитектуры RISC-V составляет 1–7% от времени численной факторизации. Для процессоров архитектуры x86 на этих матрицах переупорядочение занимало около

20% времени от численной факторизации при работе на одном узле и около 10% – при работе на двух 18-ядерных узлах. Таким образом, время нахождения перестановки на более разреженных матрицах и матрицах маленького порядка может оказывать значительное влияние на общее время решения СЛАУ на процессорах RISC-V.

5. Заключение

Интерес к устройствам RISC-V и программированию для RISC-V наблюдается во всем мире, начиная от академических организаций и заканчивая лидерами индустрии микропроцессоров. В настоящее время в России и за рубежом ведутся работы по оптимизации различных алгоритмов для процессоров архитектуры RISC-V, в том числе, классических алгоритмов плотной и разреженной алгебры. Поскольку существующие процессоры архитектуры RISC-V пока уступают по производительности процессорам x86 и GPU, то портирование таких вычислительно трудоемких алгоритмов, как библиотек для решения СЛАУ и алгоритмов переупорядочения разреженных матриц, исследовано в меньшей степени. Так, в открытой печати нет работ, описывающих работу переупорядочивателя или решателя СЛАУ на процессоре RISC-V. Тем не менее, учитывая перспективы появления высокопроизводительных процессоров архитектуры RISC-V в 2024-2025 гг., а также перспективы появления отечественной элементной базы, основанной на архитектуре RISC-V, уже сейчас представляет интерес исследование возможностей по портированию на RISC-V научного и инженерного ПО.

В работе была экспериментально исследована работа переупорядочивателей ParMETIS, DMORSy и прямого решателя разреженных матриц MUMPS на процессорах архитектуры RISC-V. Исследовано влияние полученных перестановок на время работы и масштабируемость решателя. Показано, что использование DMORSy позволяет сократить общее время решения СЛАУ при работе на нескольких вычислительных узлах и улучшить масштабируемость решателя для большинства тестовых матриц. В дальнейшем планируется исследовать возможности оптимизации переупорядочивателя DMORSy для новой архитектуры, а также возможности сокращения времени работы решателя за счет использования реализации базовых алгоритмов линейной алгебры (BLAS), оптимизированных для процессоров архитектуры RISC-V.

Список литературы

[1] Пирова А.Ю. Гибридный MPI + OpenMP алгоритм переупорядочения симметричных разреженных матриц и его применение к решению СЛАУ // Проблемы информатики. 2022. №1 (54). С. 28-41.

[2] Пирова А.Ю., Мееров И.Б., Козинов Е.А. Программный комплекс DMORSy для переупорядочения разреженных матриц на кластерных системах. // Международная конференция «Суперкомпьютерные дни в России»: Труды конференции (Москва, 24-25 сентября 2018). – М: изд-во МГУ. – 2018. – С. 749-757.

[3] Karypis G., Kumar V. Parallel Multilevel k-way Partitioning Scheme for Irregular Graphs // Supercomputing'96: Procs. of the 1996 ACM/IEEE Conference on

Supercomputing. IEEE. 1996. P. 35-35.
https://doi.org/10.1145/369028.369103

[4] Amestoy P.R. et al. A fully asynchronous multifrontal solver using distributed dynamic scheduling // SIAM J. on Matrix Analysis and Applications. 2001. Vol. 23, No. 1. P. 15–41.
https://doi.org/10.1137/S0895479899358194

[5] Berlizov D.M. et al. Comparative analysis of popular open packages of matrix reordering for the Cholesky decomposition // Параллельные вычислительные технологии (ПаВТ'2023). 2023. С. 28–33.

[6] Fibich C. et al. Evaluation of Open-Source Linear Algebra Libraries targeting ARM and RISC-V Architectures //2020 15th Conference on Computer Science and Information Systems (FedCSIS). IEEE. 2020. P. 663–672.
https://doi.org/10.15439/2020F145

[7] Rodrigues A., Sousa L., Ilic A. Performance Modelling-Driven Optimization of RISC-V Hardware for Efficient SpMV // International Conference on High Performance Computing. Cham: Springer Nature Switzerland. 2023. P. 486–499.
https://doi.org/10.1007/978-3-031-40843-4_36

[8] Vasireddy P., Kavi K., Mehta G. Sparse-t: Hardware accelerator thread for unstructured sparse data processing // Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design. 2022. P. 1–8.
https://doi.org/10.1145/3508352.3549441

[9] Scheffler P. et al. Indirection stream semantic register architecture for efficient sparse-dense linear algebra // 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE. 2021. P. 1787–1792.
https://doi.org/10.23919/DATE51398.2021.9474230

[10] Suárez D., Almeida F., Blanco V. Comprehensive analysis of energy efficiency and performance of ARM and RISC-V SoCs // The Journal of Supercomputing. 2024. P. 1–19.
https://doi.org/10.1007/s11227-024-05946-9

[11] Mezger B.W. et al. A survey of the RISC-V architecture software support // IEEE Access. 2022. Vol. 10. P. 51394-51411.
https://doi.org/10.1109/ACCESS.2022.3174125

[12] Mukhin I. et al. Benchmarking Deep Learning Inference on RISC-V CPUs (submitted to the Russian Supercomputing Days conference). 2024.

Исследование алгоритмов размещения задач на кластере с учетом топологий задачи и системы

Д.С. Жданов, В.В. Корхов

Санкт-Петербургский государственный университет

В данном исследовании рассматриваются различные алгоритмы размещения задач на кластере, учитывающие топологию кластера и взаимодействие задач. С помощью фреймворка SimGrid были созданы модели вычислительной системы. На этих моделях проводились эксперименты для оценки эффективности ряда алгоритмов размещения задач. Результаты показали, что использование алгоритмов размещения задач, которые учитывают как топологию системы, так и задач, значительно ускоряют выполнение задач. Работа проведена в рамках учебно-исследовательского проекта на факультете Прикладной математики-процессов управления СПбГУ, успешно защищена в качестве выпускной квалификационной работы бакалавра, а ее результат в виде инструмента для моделирования и исследования работы алгоритмов планирования задач может быть использован в дальнейшем учебном процессе.

Ключевые слова: Высокопроизводительные системы, HPC, топологии, алгоритмы, размещение задач

1. Введение

Важной частью образовательного процесса является проведение студентами научной работы и учебно-исследовательских проектов. Особый интерес в области высокопроизводительных вычислений, на наш взгляд, представляют работы, направленные на анализ особенностей сочетания архитектуры приложений и архитектуры вычислительных систем, на которых выполняются приложения. С точки зрения учебного процесса в целом ценным может являться результат подобной работы в виде готового инструмента, который может быть использован другими студентами для изучения особенностей построения и функционирования высокопроизводительных вычислительных систем. В данной статье представлено исследование, проведенное в рамках выпускной квалификационной работы бакалавра, результатом которого стал инструмент для исследования алгоритмов планирования задач на вычислительном кластере с учетом топологии.

Вычислительные кластеры уже много лет используются для проведения высокопроизводительных вычислений. Кластером называют множество вычислительных узлов, объединенных одной выскопроизводительной сетью. Узлы могут быть соединены между собой различными способами, используются различные топологии для организации взаимодействия. Эти топологии называются топологией системы или топологией кластера.

Задачи, которые запускаются на кластере, часто используют ресурсы нескольких вычислительных узлов. Поэтому такие задачи можно разбить на подзадачи. Описание того, как подзадачи обмениваются информацией, назовем топологией задачи.

К сожалению, при большом количестве вычислительных узлов нельзя соединить их все между собой, что позволило бы каждому узлу иметь прямую связь с любым другим узлом. На практике используются топологии системы, предлагающие ограниченный набор связей каждому из узлов, поэтому возникает необходимость грамотного размещения вычислительных задач с учетом топологии системы. Важно, чтобы подзадачи размещались на в непосредственной близости или на небольшом расстоянии в топологии кластера, чтобы минимизировать нагрузку на коммуникационную сеть и, следовательно, время на обмен данными. Особо остро эта проблема возникает, если на кластере параллельно выполняется несколько задач и нет возможности задействовать любые ресурсы.

В данном исследовании использованы наиболее популярные топологии кластера и задач, а также множество алгоритмов для отображения подзадач на вычислительные узлы. Некоторые алгоритмы размещали подзадачи без учета топологии, другие основывались только на топологии кластера, оставшиеся учитывали обе топологии.

В рамках работы был проведен ряд исследований, в которых была изучена эффективность данных алгоритмов в ситуациях, когда задача запускается на пустом кластере и в ситуациях, когда некоторые узлы уже заняты другими задачами. Для проведения экспериментов работа кластера и выполнения задач была промоделирована с использованием фреймворка SimGrid [1], исходный код разработанного инструмента для моделирования и проведения экспериментов доступен по ссылке [2].

2. Обзор литературы

В данной работе рассматривается в качестве одного из примеров топологий кластера «толстое дерево» [3]. В статье [4] представлены модификации этой топологии в виде «тонких» и «стройных» деревьев. Поскольку по итогам данной статьи такие топологии стоят дешевле, чем «толстое дерево», а падение производительности минимально, в этой статье также рассматриваются «тонкие деревья».

Топологии тора, которые также представлены в данной статье, рассматриваются в статье [5]. В статье [6] представлен подход к размещению с использованием кривой Гилберта [7], а также ряд алгоритмов выбора узлов, основная идея которых легла в нашу реализацию. Основой для построения кривой Гилберта послужила статья [8].

Используемые алгоритмы выбора узлов, на которых будут размещаться подзадачи, идейно совпадают с теми, которые используются в менеджере рабочей нагрузки SLURM [9].

Основой для размещения задач на кластере с учетом топологии задачи послужила статья [11]. В ней авторы рассматривают иерархический алгоритм отображения задачи на вычислительные узлы кластера. Реализация подобного метода в виде плагина для SLURM была представлена в статье [12]. В этой же статье была продемонстрирована практическая эффективность такого подхода.

В нашей реализации для этапа алгоритма, на котором происходит разбиение графа на несколько партиций, использовались, после небольших правок, результаты API программного пакета METIS [13], который использует многоуровневый алгоритм разбиения графа [14].

3. Топологии кластера

Взаимосвязь компьютеров в вычислительной среде или топологию кластера рассматривают как граф, в котором вершины представляют вычислительные узлы. Наличие ребра между вершинами равносильно наличию прямого соединения между компьютерами.

В данной работе рассматриваются два типа топологии кластера: сетчатые и иерархические.

3.1. Сетчатые топологии

В рамках данной топологии каждое ребро решетки параллельно одной из осей координат и связывает два смежных узла вдоль этой оси. Данные топологии можно рассматривать как в виде плоских решеток, так и в форме гиперкуба с размерностью 3 или более.

В наших экспериментах выбрана частая модификация решетчатых топологий, а конкретнее, топологии двумерного и трехмерного тора. Данные топологии получаются путем соединения краевых узлов сетчатых топологий с соответствующей размерностью.

3.2. Иерархические топологии

Данный вид топологий имеет большое количество реализаций, однако мы используем только топологии толстого и тонкого дерева. Данные топологии отлично подходят для организации вычислений на большом количестве вычислительных узлов.

В рамках данных топологий вычислительные узлы находятся на листьях дерева и связываются друг с другом посредством сети коммутаторов, которые и образуют дерево. Обозначим за n количество уровней коммутаторов в дереве, за k примем количество связей, ведущих к коммутаторам на уровне ниже, а за k’ — количество связей, ведущих к коммутаторам на уровне выше. Тогда отношение k/k’ называется коэффициентом сужения. Обычно деревья обозначаются как k:k’-арные n-деревья.

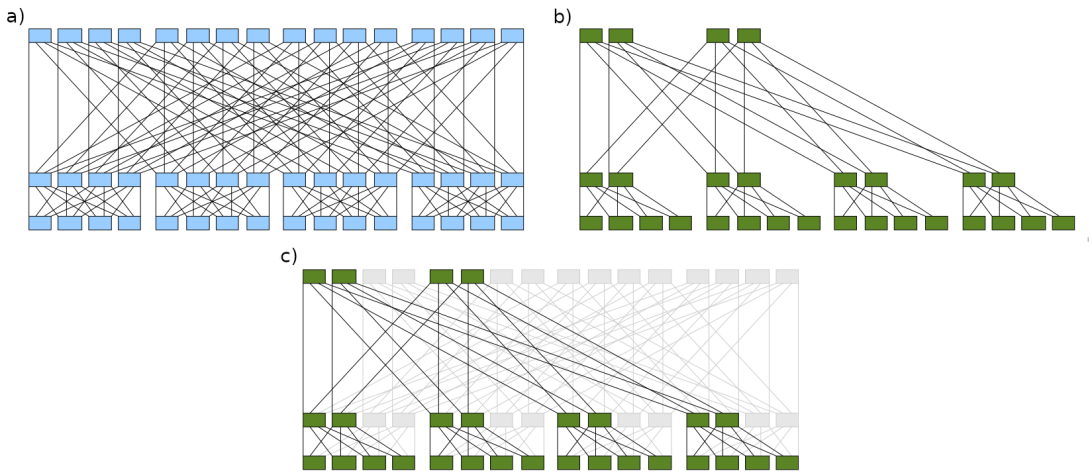


Рис. 1: Толстые и тонкие деревья: a) 4-арное 3-fat-tree или 4,3-fat-tree b) 4:2,3-thin-tree c) сравнение fat-tree и thin-tree

В топологии fat-tree коэффициент сужения всегда равен 1:1. Данный факт обеспечивает большое количество альтернативных путей для передачи данных между удаленными друг от друга вычислительными узлами и высокую пропускную способность.

Топология thin-tree представляет из себя урезанную версию толстого дерева. В нём количество нисходящих связей не совпадает с восходящими. При этом не ставится условия, чтобы эти два значения были кратными, однако k должно быть больше k', чтобы дерево было сужающимся к корню (рисунок 1).

4. Топологии задачи

Под топологией задачи понимается взвешенный граф, вершинами которого являются подзадачи. Наличие ребра между двумя вершинами сигнализирует о том, что эти две задачи будут обмениваться информацией друг с другом во время выполнения. Вес такого ребра характеризует интенсивность общения данных подзадач.

Простым примером одной из используемых топологий задачи является Master-Worker. Смысл данной топологии заключается в том, что имеется несколько подзадач, которые получают от основной подзадачи какие-то данные, обрабатывают их и возвращают обратно результат. При этом такие подзадачи никак не взаимодействуют друг с другом.

Если попробовать графически отобразить такую топологию, то получим картину, совпадающую с топологией звезды. Аналогичным образом можно провести аналогии с другими известными топологиями. В рамках данной работы используются топологии задач, основанных на топологиях звезды, плоской решетки, куба и двоичного дерева.

5. Алгоритмы размещения

5.1. Простая (Simple) и Случайная (Random) стратегия

Данные алгоритмы являются тривиальными и не учитывают ни топологию кластера, ни топологию задачи.

Простой алгоритм предполагает, что можно некоторым образом пронумеровать вычислительные узлы кластера. В рамках данной стратегии последовательно перебираем все узлы и присваиваем подзадачи первым свободным узлам.

Смысл случайного алгоритма заключается в том, что произвольным образом выбираются свободные вычислительные узлы и назначаются их подзадачам.

5.2. Оптимальная (Optimal) стратегия

Под данным классом алгоритмов подразумеваются стратегии размещения, при которых в процессе размещения учитывается топология кластера. Поскольку сетчатые и иерархические топологии кластера сильно отличаются друг от друга по структуре, то для этих классов топологий будут использоваться различные алгоритмы размещения.

Оптимальное размещение в сетчатых топологиях. Основная идея при размещении задач в сетчатых топологиях заключается в использовании кривой Гилберта. Смысл данной идеи заключается в том, чтобы представить многомерное пространство в виде прямой, которая обеспечивает хорошую плотность заполнения пространства.

При наличии такого представления топологии можно получить все свободные отрезки на данной прямой. Под свободным отрезком понимаются незанятые подряд идущие вычислительные узлы. После этого, начиная с каждого отрезка, предпринимаются попытки разместить задачу. Если отрезок не может вместить все подзадачи, то заполняется следующий. В конечном итоге подзадачи заполняются с того отрезка, для которого расстояние между первой и последней подзадачей минимально.

Оптимальное размещение в иерархических топологиях. Для топологий толстого и тонкого дерева необходимо как можно больше локализовать подзадачи и уменьшить нагрузку на верхние уровни коммутаторов.

Для этого перебираются уровни коммутаторов с целью найти самый низкий уровень, на котором имеется коммутатор, имеющий достаточное количество свободных вычислительных узлов для размещения запрошенных ресурсов.

Для увеличения кучности подзадач после нахождения необходимого коммутатора сортируются его нижележащие коммутаторы по убыванию свободных узлов и повторяется данная операция для всех дочерних коммутаторов.

Поскольку коммутаторы высоких уровней связаны с большим количеством вычислительных узлов, то, используя такую предварительную обработку, с большой вероятностью сокращается количество коммутаторов на верхних уровнях, что сильно уменьшает их использование в целом.

5.3. Продвинутая (Advanced) стратегия

Данная стратегия учитывает топологии кластера и задачи. Она состоит из двух частей. На первом этапе выбираются узлы, на которых будут размещаться подзадачи, а на втором соотносятся сами подзадачи с выбранными узлами.

Выбор узлов и построение дерева ближайших соседей. Для выбора узлов в зависимости от топологии, на которой запущена задача, применяется одна из оптимальных стратегий. После чего строится дерево ближайших соседей. Вычислительные узлы расположены на листьях данного дерева, а нелистовые узлы представляют из себя группу смежных узлов (рисунок 2).

В случае с кластером, имеющим топологию толстого или тонкого дерева, дерево ближайших соседей строится наивным образом. На листьях дерева расположены вычислительные узлы, а другие узлы дерева ближайших соседей представляют из себя коммутаторы, через которые связываются вычислительные узлы, начиная с подходящего коммутатора, найденного оптимальной стратегией.

В случае сетчатых топологий вычислительные узлы представлены в виде списка, отсортированного по номеру соответствующего узла на кривой Гилберта. Потом этот список делится на две части, и создается новая вершина, которая их объединяет, и рекурсивно проделываем эту операцию для каждой половины списка.

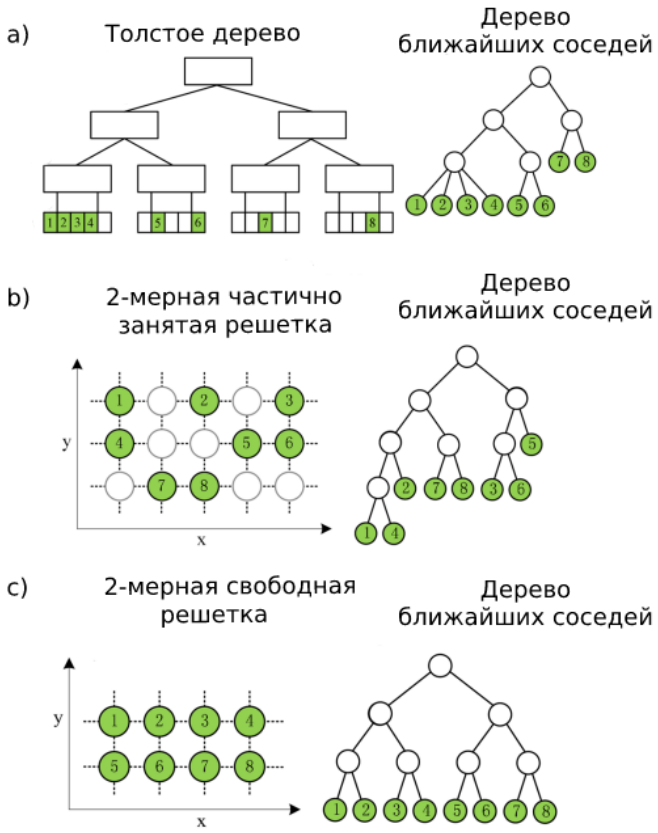


Рис. 2: Представление узлов в виде дерева ближайших соседей

Отображение подзадач на вычислительные узлы. После построения дерева ближайших соседей начинается процесс рекурсивного отображения подзадач текущей задачи на выбранные вычислительные узлы. Для этого граф связи подзадач разбивается на такое количество подграфов, сколько поддеревьев у выбранного корня дерева ближайших соседей. Разделение графа на подграфы происходит таким образом, чтобы число вершин в соответствующих поддеревьях и подграфах было одинаково, и связь элементов из разных подграфов была минимальной. Для выполнения данной операции используется API программного пакета METIS, который реализует алгоритм многоуровневого разбиения графа.

6. Симуляция

6.1. Программная реализация.

Программная реализация для проведения экспериментов состоит из двух основных частей.

Во-первых, был создан функционал для моделирования кластера и обмена сообщениями между различными узлами с использованием фреймворка SimGrid.

Во-вторых, были реализованы описанные ранее алгоритмы размещения задач.

В итоге программное обеспечение включает в себя несколько конфигурационных файлов: один для описания моделируемых кластеров, а другой — для определения параметров запуска задач.

6.2. Параметры системы.

Мы моделировали 4 различные топологии кластеров:

1) Трехмерный тор с размерностью 16 по каждой оси,
2) Двумерный тор с размерностью 64 по каждой оси,
3) Толстое дерево с 6 уровнями коммутаторов, 4 восходящими и нисходящими связями,
4) Тонкое дерево с 6 уровнями, 2 восходящими и 4 нисходящими связями.

Каждый кластер имеет в своем распоряжении 4096 вычислительных узлов с вычислительной мощностью 1 GFlops. Вычислительные узлы и коммутаторы в тонком и толстом дереве соединены связями, имеющими пропускную способность в 125 Mb/s и задержку в 50 us.

6.3. Параметры задач в экспериментах.

Были проведены два типа экспериментов с различным объемом передаваемых сообщений. В частности, рассматривался вариант с минимальным объемом сообщений в 1Б и задачи с объемом сообщений в 1 Мбайт. В первом случае фактически пренебрегается время на передачу сообщений и изучается влияние задержки при отправке сообщений между узлами. Такой подход пусть и не описывает реальный случай использования кластера, но позволяет сравнить стратегии без неизбежных констант. Во втором случае анализируются результаты в обстановке более приближенной к реальности, и, возможно, выявятся какие-либо изменения в поведении различных алгоритмов.

Каждый эксперимент запускается на всех имеющихся топологиях кластера для каждого типа рассматриваемой задачи. В качестве топологий кластера, как было сказано ранее, рассматриваются двумерный тор (torus2d), трехмерный тор (torus3d), тонкое дерево (thinTree) и толстое дерево (fatTree). В качестве топологий задач используются топологии звезды (STAR), плоской решетки (GRID), трехмерного куба (CUBE) и двоичного дерева (TREE).

В первом эксперименте запускаются задачи на пустом кластере и изучается зависимость времени от количества запрашиваемых задачей узлов. В рамках этого эксперимента планируется оценить, насколько хорошо стратегии показывают себя в идеальных ситуациях.

Во втором эксперименте запускается контрольная задача, запрашивающая 256 вычислительных узлов на частично заполненном кластере, и изучается зависимость времени выполнения контрольной задачи от процента заполненности кластера. В рамках данного эксперимента рассчитывается оценить работу алгоритмов в условиях ограниченности выбора узлов.

Для моделирования частично заполненного кластера перед запуском контрольной задачи загружаются на кластер задачи, запрашивающие 32 узла. Причем некоторые из них помечаются как мнимые, и узлы, занятые данными задачи, освобождаются перед размещением контрольной задачи.

Также важным моментом является то, что для задач, которые имитируют нагруженность кластера, используются различные стратегии размещения задач, в том числе и случайная, для создания максимально приближенного к реальности состояния кластера.

7. Результаты экспериментов

Результаты первого эксперимента, в котором запускаются задачи на пустом кластере, представлены на Рисунке 3. Поскольку графики для различных топологий кластеров имеют одинаковую структуру, мы приводим пример только для кластера с топологией трехмерного тора.

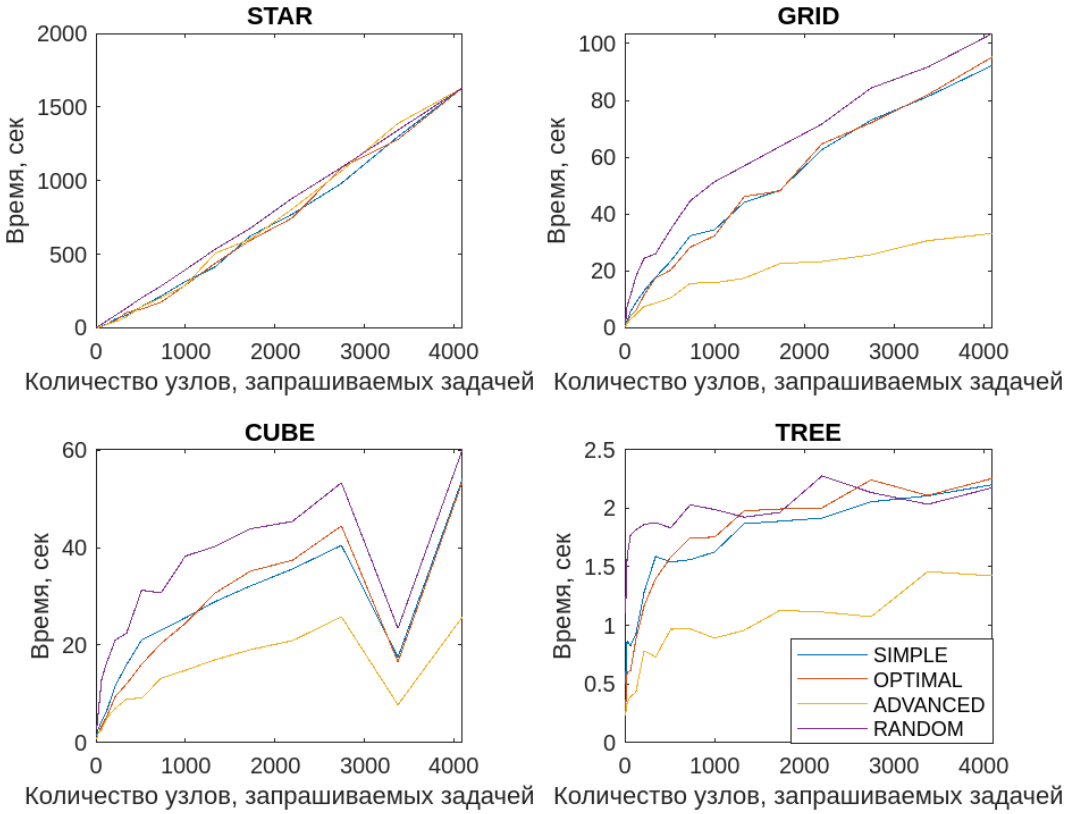


Рис. 3: Результаты первого эксперимента

Из графиков видно, что задачи с различными топологиями имеют различное поведение при увеличении числа запрашиваемых узлов. Так, задачи с топологией звезды растут практически линейно, в то время как рост времени выполнения задач, имеющих топологию двоичного дерева, очень велик на небольшом количестве узлов, но позже изменяется незначительно.

Относительно стратегий размещения можно сказать, что на пустом кластере продвинутая стратегия размещения, учитывающая топологию задачи, показывает значительно лучшие результаты. Исключением являются задачи с топологией звезды, для которых выбор стратегии практически не влияет на результат.

Для того, чтобы более формально сравнить стратегии размещения, были рассмотрены средние отношения различных политик размещения к продвинутой политике размещения. Из таблицы 1 можно увидеть, что продвинутая стратегия размещения дает большее ускорение для кластеров с топологией тора, нежели в иерархических топологиях.

В данном эксперименте оценивается время работы при запуске задачи на пустом кластере. Однако бывают ситуации, когда на кластере уже запущены некоторые задачи, и планировщик сталкивается с ограничениями в выборе узлов.

Таблица 1: Средние отношения времени работы алгоритмов к продвинутой стратегии в первом эксперименте

		Объем сообщений в 1 байт			Объем сообщений в 1 Мбайт		
		Simple	Optimal	Random	Simple	Optimal	Random
torus2d	STAR	1.423	1.004	3.302	1.109	0.984	1.59
	GRID	3.341	2.633	6.002	1.748	1.501	2.191
	CUBE	2.298	1.792	5.314	1.5735	1.284	1.963
	TREE	2.567	1.891	4.731	2.681	1.79	1.654
torus3d	STAR	1.182	1.012	2.005	1.033	0.989	1.21
	GRID	2.148	1.986	3.334	1.268	1.206	1.391
	CUBE	1.764	1.546	2.871	1.211	1.141	1.328
	TREE	1.868	1.738	2.661	1.887	1.678	1.363
fatTree	STAR	0.974	0.974	1.472	0.99	0.99	1.131
	GRID	1.76	1.206	2.623	1.169	1.031	1.293
	CUBE	1.406	1.016	2.161	1.098	0.982	1.233
	TREE	1.26	1.206	1.927	1.532	1.393	1.325
thinTree	STAR	0.974	0.974	1.472	0.99	0.99	1.131
	GRID	1.756	1.212	2.641	1.242	1.011	1.359
	CUBE	1.426	1.02	2.169	1.252	0.974	1.359
	TREE	1.235	1.21	1.878	3.877	4.482	2.216

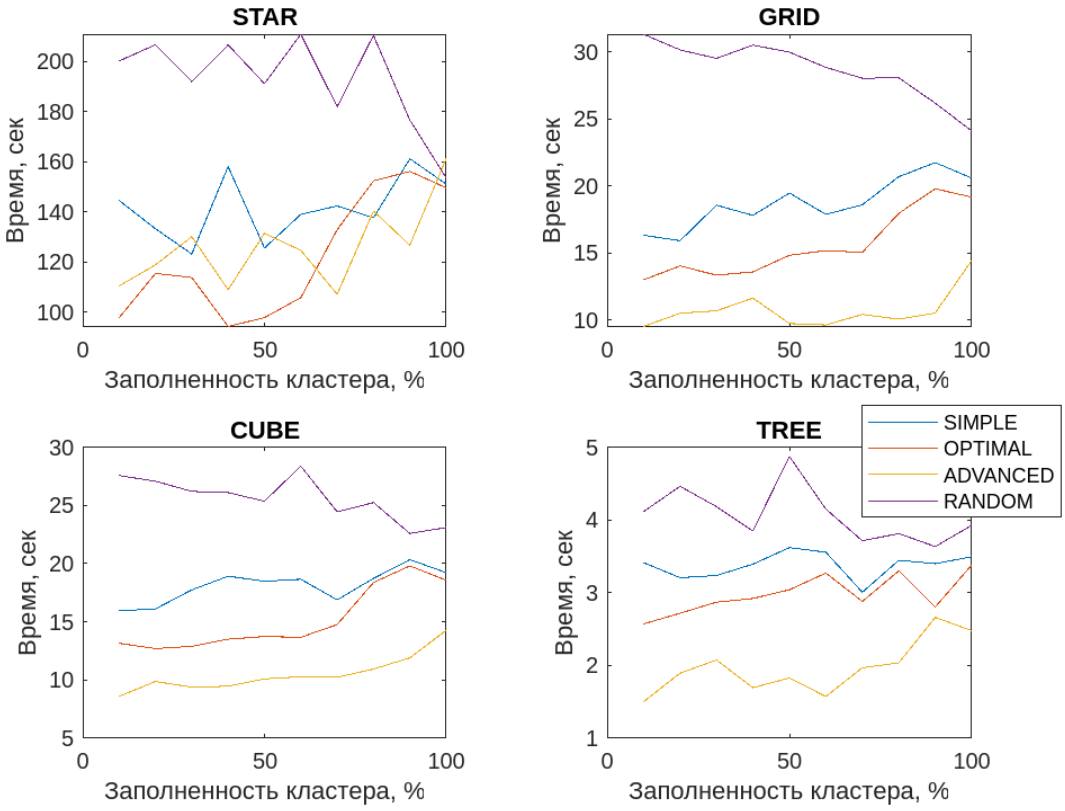


Рис. 4: Результаты второго эксперимента

Таблица 2: Средние отношения времени работы алгоритмов к продвинутой стратегии во втором эксперименте

		Объем сообщений в 1 байт			Объем сообщений в 1 Мбайт		
		Simple	Optimal	Random	Simple	Optimal	Random
torus2d	STAR	1.411	1.013	2.252	1.245	1.016	1.163
	GRID	2.422	1.484	4.375	1.656	1.378	2.199
	CUBE	2.161	1.31	3.819	1.512	1.122	2.199
	TREE	2.486	1.907	3.602	2.869	1.98	1.664
torus3d	STAR	1.142	0.968	1.562	1.069	1.032	1.206
	GRID	1.771	1.467	2.730	1.239	1.156	1.334
	CUBE	1.744	1.44	2.499	1.199	1.09	1.277
	TREE	1.768	1.55	2.14	1.923	1.755	1.402
fatTree	STAR	0.921	0.848	1.153	0.966	0.929	1.048
	GRID	1.54	1.088	2.081	1.162	1.017	1.246
	CUBE	1.365	1.025	1.921	1.110	0.991	1.197
	TREE	1.391	1.290	1.72	1.442	1.384	1.23
thinTree	STAR	0.912	0.848	1.153	0.969	0.929	1.048
	GRID	1.521	1.089	2.083	1.137	1.033	1.295
	CUBE	1.37	1.023	1.922	1.176	0.981	1.22
	TREE	1.42	1.306	1.714	2.291	2.862	1.47

Для изучения работы стратегий размещений при частично заполненном кластере проведен второй эксперимент. Его результаты представлены на рисунке 4. График представлен только для одной топологии трехмерного тора, поскольку графики для других кластеров в целом похожи.

На графиках можно наблюдать тенденцию уменьшения разницы во времени работы между различными политиками размещения при увеличении процента заполненности кластера. Это связано с тем, что у планировщика уменьшается выборка узлов, на которых он может разместить задачу. Так, при 10% заполненности кластера наблюдается разница во времени выполнения при размещении на конкретных оставшихся вычислительных узлах, когда время выполнения зависит только от того, как подзадачи были распределены на конкретных узлах.

При сравнении времени работы различных стратегий размещений можно отметить, что для задач с топологией звезды выбор стратегии размещения уже играет какую-то роль. Причем на разных процентах заполненности кластера лидируют различные стратегии размещения. В других же случаях наблюдается ситуация, что размещение с учетом топологии задачи всегда показывает наилучший результат. В таблице 2 представлены средние отношения различных политик размещения к продвинутой стратегии во втором эксперменте.

При сравнении результатов экспериментов при объеме сообщений в 1 байт и 1 Мбайт, можно отметить, что в большинстве случаев ускорение несколько падает. Это связано в первую очередь с тем, что существует определенная доля константы времени, приходящаяся на непосредственно передачу сообщения. однако в некоторых случаях, как, например, в топологии тонкого дерева при задаче, имеющей гра-

фик коммуникаций в виде двоичного дерева, наоборот, наблюдается существенный прирост ускорения. Стоит отметить, что сохраняется наблюдение из первого эксперимента о том, что продвинутая стратегия размещения дает большее ускорение на кластерах с топологией тора. При этом в иерархических топологиях оптимальная стратегия очень близка по результатам к продвинутой.

8. Заключение

По результатам исследования можно сделать вывод, что использование алгоритмов, которые учитывают топологии кластера и задачи, могут существенно сократить время, затрачиваемое на передачу сообщения, и, как следствие, увеличить скорость выполнения задачи. Однако, как показывает ситуация с топологией звезды, не для всех топологий задач можно получить прирост производительности. По этой причине для некоторых топологий может понадобиться использование иных алгоритмов. Для продолжения исследований, в частности в рамках учебно-исследовательских проектов, можно воспользоваться созданным в рамках данного проекта инструментом, доступным по ссылке [2].

Список литературы

[1] SimGrid.
`https://simgrid.org/AccessedonMay1,2024`

[2] Исходный код проекта.
`https://github.com/Zhdanov-Dmitrii/Study_Algorithms_For_Placement_Tasks`
Accessed on May 1, 2024.

[3] Leiserson, Charles E. Fat-trees: universal networks for hardware-efficient supercomputing. IEEE Transactions on Computers. 34 (10): 892–901, 1985.
`https://doi.org/10.1109/TC.1985.6312192`

[4] Navaridas, J., Miguel-Alonso, J., Ridruejo, F. J., & Denzel, W. Reducing complexity in tree-like computer interconnection networks. Parallel Computing, 36(2-3), 71-85, 2010.
`https://doi.org/10.1016/j.parco.2009.12.004`

[5] N.R. AdigaM. A.B., others. Blue Gene/L torus interconnection network // IBM Journal of Research and Development, 2005. — P. 265–276.

[6] Leung V.J. A.E.B.e.a. Processor allocation on Cplant: achieving general processor locality using one-dimensional allocation strategies. — Proc. 4th IEEE International Conference on Cluster Computing, 2002. — P. 296–304.

[7] D. Hilbert.Über die stetige abbildung einer linie auf ein flachenstück. Math.Ann., 38:459–460, 1 891.

[8] Skilling, J. (2004, April). Programming the Hilbert curve. In AIP Conference Proceedings (Vol. 707, No. 1, pp. 381-387). American Institute of Physics.

[9] SLURM. Topology Guide. URL: https://slurm.schedmd.com/topology.html Accessed on May 1, 2024

[10] Javier N. Jose M. F.J.W.D. Reducing complexity in tree-like computer interconnection networks // Parallel Computing. — 2010. — P. 71–85.

[11] Wu J X.X., Z L. Hierarchical task mapping for parallel applications on supercomputers // The Journal of Supercomputing. — 2015. — V. 71, no. 5. — P. 1776–1802.

[12] Georgiou Y Jeannot E M.G.V.A. Topology-aware job mapping // The International Journal of High Performance Computing Applications. — 2018. — V. 32, no. 1. — P. 14–27.

[13] Metis.
`https://github.com/KarypisLab/METIS`
Accessed on May 1, 2024.

[14] Karypis G K.V. Multilevel k-way partitioning scheme for irregular graphs. J Parallel Distrib Comput // J Parallel Distrib Comput 48(1). — 1998. — P. 96–129.

Исследование масштабируемости параллельной реализации алгоритма AlFaMove для линейного программирования на кластерной вычислительной системе[1]

Н.А. Ольховский, Л.Б. Соколинский

Южно-Уральский государственный университет
(национальный исследовательский университет)

Работа посвящена параллельной реализации нового алгоритма линейного программирования, получившего название AlFaMove. Алгоритм строит на поверхности допустимого многогранника оптимальный целевой путь от произвольной граничной точки до точки, являющейся решением задачи линейного программирования. Оптимальность пути заключается в том, что при поиске точки максимума целевой функции выбирается направление движения по грани многогранника, соответствующее максимальному увеличению значения целевой функции. Для вычисления направления движения используется итерационный алгоритм проекционного типа. Выполнена параллельная реализация алгоритма AlFaMove. Приведены результаты вычислительных экспериментов на кластерной вычислительной системе, демонстрирующие высокую масштабируемость предложенной реализации.

Ключевые слова: линейное программирование, AlFaMove, алгоритм движения по граням, параллельная реализация, кластерная вычислительная система, исследование масштабируемости.

1. Введение

Эпоха больших данных и индустрия 4.0 породили задачи линейного программирования (ЛП) сверхбольших размерностей, включающих в себя миллионы переменных и миллионы ограничений [1, 2, 3, 4]. Во многих случаях объектом линейного программирования являются задачи, связанные с оптимизацией нестационарных процессов [5]. В нестационарных задачах ЛП целевая функция и/или ограничения изменяются в течение вычислительного процесса. Также среди этого класса задач встречаются приложения, в которых необходимо выполнять оптимизацию в режиме реального времени. Для решения таких задач необходимы масштабируемые методы и параллельные алгоритмы линейного программирования.

Один из стандартных подходов к решению нестационарных задач оптимизации состоит в том, чтобы рассматривать каждое изменение как появление новой задачи оптимизации, которую необходимо решать с нуля [5]. Однако такой подход часто непрактичен, поскольку решение проблемы с нуля без повторного использования ин-

[1]Исследование выполнено при финансовой поддержке РНФ (проект № 23-21-00356).

формации из прошлого может занять слишком много времени. Таким образом, желательно иметь алгоритм оптимизации, способный непрерывно адаптировать решение к изменяющейся среде, повторно используя информацию, полученную в прошлом. Этот подход применим для процессов реального времени, если алгоритм достаточно быстро отслеживает траекторию движения оптимальной точки. В случае больших задач ЛП последнее требует разработки масштабируемых методов и параллельных алгоритмов ЛП.

До настоящего времени наиболее популярными методами решения больших задач ЛП являются симплекс-метод [6] и метод внутренних точек [7]. Эти методы способны решать задачи с десятками тысяч переменных и ограничений. Однако масштабируемость параллельных алгоритмов, основанных на симплекс-методе, в общем случае ограничивается 16–32 процессорными узлами [8]. Что касается алгоритмов внутренних точек, они не поддаются эффективному распараллеливанию в общем случае. Это ограничивает применение указанных методов для решения сверхбольших нестационарных задач ЛП в режиме реального времени. В соответствие с этим задача разработки масштабируемых методов и эффективных параллельных алгоритмов ЛП для кластерных вычислительных систем остается актуальной.

В недавней работе [9] было дано теоретическое описание нового метода ЛП — метода поверхностного движения, строящего на поверхности допустимого многогранника оптимальный целевой путь к решению задачи ЛП. Под оптимальным целевым путем понимается путь по поверхности допустимого многогранника в направлении наибольшего увеличения значений целевой функции. Однако предложенный в этой статье алгоритм 1 на шаге 15 требует нахождения на границе гипердиска точки с максимальным значением целевой функции. При этом не приводится численный алгоритм, позволяющий выполнить этот шаг. В этой статье мы приводим и исследуем алгоритм AlFaMove, устраняющий допущенный пробел.

Статья организована следующим образом. Раздел 2 содержит теоретический базис, необходимый для описания метода поверхностного движения и его численной реализации. Раздел 3 посвящен описанию операции псевдопроекции, позволяющий найти вектор движения по оптимальному целевому пути для линейного многообразия, получаемого в результате пересечением гиперплоскостей. В разделе 4 дается формализованное описание алгоритма AlFaMove, представляющего собой численную реализацию метода поверхностного движения. Раздел 5 посвящен описанию параллельной версии алгоритма AlFaMove. В разделе 6 представлены информация о программной реализации алгоритма AlFaMove и результаты экспериментов на кластерной вычислительной системе по исследованию его масштабируемости. В заключении суммируются полученные результаты и намечаются направления дальнейших исследований.

2. Теоретический базис

Данный раздел содержит необходимый теоретических базис, используемый для описания алгоритма движения по граням AlFaMove. Рассмотрим задачу ЛП в следующем виде:

$$\bar{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ \langle \boldsymbol{c}, \boldsymbol{x} \rangle \,|\, A\boldsymbol{x} \leqslant \boldsymbol{b} \right\}, \tag{1}$$

где $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$, $m > 1$, $\boldsymbol{c} \neq \boldsymbol{0}$. Здесь $\langle \cdot, \cdot \rangle$ обозначает скалярное произведение двух векторов. Мы предполагаем, что ограничение $\boldsymbol{x} \geqslant \boldsymbol{0}$ также включено в

матричное неравенство $A\boldsymbol{x} \leqslant \boldsymbol{b}$ в форме

$$-\boldsymbol{x} \leqslant \boldsymbol{0}.$$

Линейная целевая функция задачи (1) имеет вид

$$f(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle .$$

Вектор $\boldsymbol{c}$ в данном случае является градиентом целевой функции $f(\boldsymbol{x})$.

Пусть $\boldsymbol{a}_i \in \mathbb{R}^n$ обозначает вектор, представляющий i-тую строку матрицы A. Мы предполагаем, что $\boldsymbol{a}_i \neq \boldsymbol{0}$ для всех $i \in \{1, \ldots, m\}$. Обозначим через $\hat{H}_i$ замкнутое полупространство, определяемое неравенством $\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \leqslant b_i$, а через H_i — ограничивающую его гиперплоскость:

$$\hat{H}_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \leqslant b_i\}; \tag{2}$$

$$H_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle = b_i\}. \tag{3}$$

Определим допустимый многогранник

$$M = \bigcap_{i=1}^{m} \hat{H}_i, \tag{4}$$

представляющий множество допустимых точек задачи ЛП (1). Заметим, что M в этом случае будет замкнутым выпуклым множеством. Мы будем предполагать, что множество M является ограниченным и $M \neq \emptyset$, то есть задача ЛП (1) имеет решение.

Дадим определение рецессивного полупространства [10].

Определение 1. Полупространство $\hat{H}_i$ называется рецессивным, если

$$\forall \boldsymbol{x} \in H_i, \forall \lambda > 0 : \boldsymbol{x} + \lambda \boldsymbol{c} \notin \hat{H}_i. \tag{5}$$

Геометрический смысл этого определения состоит в том, что луч, исходящий в направлении вектора $\boldsymbol{c}$ из любой точки гиперплоскости, ограничивающей рецессивное полупространство, не имеет общих точек с этим полупространством, за исключением начальной. Известно [10], что следующее условие является необходимым и достаточным для того, чтобы полупространство $\hat{H}_i$ было рецессивным:

$$\langle \boldsymbol{a}_i, \boldsymbol{c} \rangle > 0.$$

Определим

$$\mathcal{I} = \{i \in \{1, \ldots, m\} \, | \langle \boldsymbol{a}_i, \boldsymbol{c} \rangle > 0\}, \tag{6}$$

то есть $\mathcal{I}$ представляет множество индексов, для которых полупространство $\hat{H}_i$ является рецессивным. Поскольку допустимый многогранник M представляет собой ограниченное множество, имеем

$$\mathcal{I} \neq \emptyset.$$

Положим

$$\hat{M} = \bigcap_{i \in \mathcal{I}} \hat{H}_i. \tag{7}$$

Очевидно, что $\hat{M}$ является выпуклым, замкнутым, неограниченным многогранником. Будем называть его рецессивным. Из (4) и (6) следует

$$M \subset \hat{M}.$$

Обозначим через $\Gamma(M)$ множество граничных точек допустимого многогранника M, а через $\Gamma(\hat{M})$ — множество граничных точек рецессивного многогранника $\hat{M}$[1]. Согласно утверждению 3 в [10] имеем

$$\bar{\boldsymbol{x}} \in \Gamma(\hat{M}),$$

то есть решение задачи ЛП (1) лежит на границе рецессивного многогранника $\hat{M}$.

Следуя [11], дадим определение ортогональной проекции на гиперплоскость.

Определение 2. Пусть в пространстве $\mathbb{R}^n$ имеется гиперплоскость

$$H = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}, \boldsymbol{x} \rangle = b\}.$$

Ортогональной проекцией точки $\boldsymbol{v} \in \mathbb{R}^n$ на гиперплоскость H называется отображение $\boldsymbol{\pi}_H$, определяемое формулой

$$\boldsymbol{\pi}_H(\boldsymbol{v}) = \boldsymbol{v} - \frac{\langle \boldsymbol{a}, \boldsymbol{v} \rangle - b}{\|\boldsymbol{a}\|^2} \boldsymbol{a}. \tag{8}$$

Следующее утверждение дает способ вычисления оптимального целевого пути на гиперплоскости.

Предложение 1. *Пусть в пространстве $\mathbb{R}^n$ задана гиперплоскость H с нормалью $\boldsymbol{a} \in \mathbb{R}^n$, проходящая через точку $\boldsymbol{u} \in \mathbb{R}^n$:*

$$H = \{\boldsymbol{x} \in \mathbb{R}^n \,|\langle \boldsymbol{a}, \boldsymbol{x} \rangle = \langle \boldsymbol{a}, \boldsymbol{u} \rangle\}. \tag{9}$$

Пусть задана линейная целевая функция $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ с градиентом $\boldsymbol{c} \in \mathbb{R}^n$:

$$f(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle. \tag{10}$$

Пусть векторы $\boldsymbol{a}$ и $\boldsymbol{c}$ линейно независимы (не коллинеарны, и среди них нет нулевого вектора). Положим

$$\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{c}. \tag{11}$$

Построим ортогональную проекцию $\boldsymbol{\pi}_H(\boldsymbol{v})$ точки $\boldsymbol{v}$ на гиперплоскость H:

$$\boldsymbol{w} = \boldsymbol{\pi}_H(\boldsymbol{v}). \tag{12}$$

Тогда вектор $\boldsymbol{d} = \boldsymbol{w} - \boldsymbol{u}$ однозначно задает направление максимального увеличения целевой функции $f(\boldsymbol{x})$, определяемой формулой (10).

Доказательство. Предположим противное, то есть существует точка $\tilde{\boldsymbol{w}} \in H$ такая, что

$$\langle \boldsymbol{c}, \tilde{\boldsymbol{w}} \rangle \geqslant \langle \boldsymbol{c}, \boldsymbol{w} \rangle, \tag{13}$$

$\|\tilde{\boldsymbol{w}} - \boldsymbol{u}\| = \|\boldsymbol{w} - \boldsymbol{u}\|$ и $\tilde{\boldsymbol{w}} \neq \boldsymbol{w}$ (см. рис. 1). Здесь и далее $\|\cdot\|$ обозначает евклидову норму.

Вычислим $\langle \boldsymbol{c}, \boldsymbol{w} \rangle$. В соответствии с определением 2 ортогональная проекция $\boldsymbol{\pi}_H(\boldsymbol{v})$ точки $\boldsymbol{v}$ на гиперплоскость H, задаваемую формулой (9), вычисляется следующим образом:

$$\boldsymbol{w} = \boldsymbol{v} - \frac{\langle \boldsymbol{a}, \boldsymbol{v} - \boldsymbol{u} \rangle}{\|\boldsymbol{a}\|^2} \boldsymbol{a}.$$

[1]Под граничной точкой множества $\hat{M} \subset \mathbb{R}^n$ понимается точка в $\mathbb{R}^n$, для которой любая открытая ее окрестность в $\mathbb{R}^n$ имеет непустое пересечение как с множеством $\hat{M}$, так и с его дополнением.

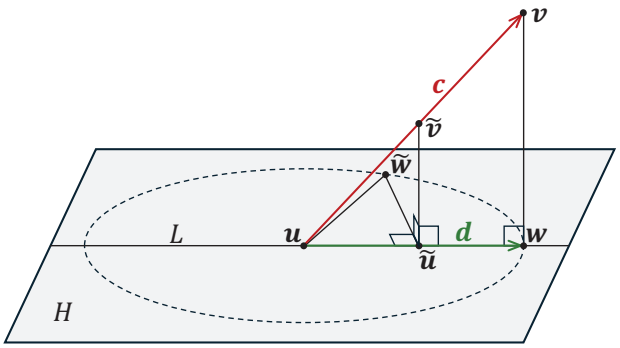


Рис. 1: Иллюстрация к доказательству предложения 1. Пунктиром обозначена гиперокружность с радиусом $\|\boldsymbol{w}-\boldsymbol{u}\|$.

Подставив вместо $\boldsymbol{v}$ правую часть формулы (11), получим

$$\boldsymbol{w} = \boldsymbol{u} + \boldsymbol{c} - \frac{\langle \boldsymbol{a}, \boldsymbol{c} \rangle}{\|\boldsymbol{a}\|^2}\boldsymbol{a}. \tag{14}$$

Используя (14), находим

$$\langle \boldsymbol{c}, \boldsymbol{w} \rangle = \langle \boldsymbol{c}, \boldsymbol{u} \rangle + \|\boldsymbol{c}\|^2 - \frac{\langle \boldsymbol{a}, \boldsymbol{c} \rangle^2}{\|\boldsymbol{a}\|^2}. \tag{15}$$

Поскольку $\boldsymbol{a}$ и $\boldsymbol{c}$ линейно независимы, в соответствии с неравенством Коши — Буняковского имеем

$$\langle \boldsymbol{a}, \boldsymbol{c} \rangle^2 < \|\boldsymbol{a}\|^2 \cdot \|\boldsymbol{c}\|^2.$$

Отсюда следует

$$\|\boldsymbol{c}\|^2 - \frac{\langle \boldsymbol{a}, \boldsymbol{c} \rangle^2}{\|\boldsymbol{a}\|^2} > 0. \tag{16}$$

Теперь вычислим $\langle \boldsymbol{c}, \tilde{\boldsymbol{w}} \rangle$. Определим $\tilde{\boldsymbol{u}} = \boldsymbol{\pi}_{\boldsymbol{L}}(\tilde{\boldsymbol{w}})$ — ортогональная проекция точки $\tilde{\boldsymbol{w}}$ на прямую L, проходящую через точки $\boldsymbol{u}$ и $\boldsymbol{w}$. По построению существует число $\delta \in \mathbb{R}$, удовлетворяющее условию

$$-1 \leqslant \delta < 1, \tag{17}$$

такое, что

$$\tilde{\boldsymbol{u}} = \boldsymbol{u} + \delta(\boldsymbol{w} - \boldsymbol{u}).$$

Положим

$$\tilde{\boldsymbol{v}} = \boldsymbol{u} + \delta(\boldsymbol{v} - \boldsymbol{u}). \tag{18}$$

Тогда точка $\tilde{\boldsymbol{u}}$ является ортогональной проекцией точки $\tilde{\boldsymbol{v}}$ на гиперплоскость H, определяемую формулой (9), и может быть вычислена следующим образом:

$$\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{v}} - \frac{\langle \boldsymbol{a}, \tilde{\boldsymbol{v}} - \boldsymbol{u} \rangle}{\|\boldsymbol{a}\|^2}\boldsymbol{a}.$$

Подставив вместо $\tilde{\boldsymbol{v}}$ правую часть формулы (18), отсюда получим

$$\tilde{\boldsymbol{u}} = \boldsymbol{u} + \delta\left(\boldsymbol{v} - \boldsymbol{u} - \frac{\langle \boldsymbol{a}, \boldsymbol{v} - \boldsymbol{u} \rangle}{\|\boldsymbol{a}\|^2}\boldsymbol{a}\right).$$

Используя (11), произведем замену $\boldsymbol{v} - \boldsymbol{u} = \boldsymbol{c}$:

$$\tilde{\boldsymbol{u}} = \boldsymbol{u} + \delta \left(\boldsymbol{c} - \frac{\langle \boldsymbol{a}, \boldsymbol{c} \rangle}{\|\boldsymbol{a}\|^2} \boldsymbol{a} \right). \tag{19}$$

Очевидно

$$\tilde{\boldsymbol{w}} = (\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{u}}) + \tilde{\boldsymbol{u}}. \tag{20}$$

Заменим второе слагаемое в (20) на правую часть формулы (19):

$$\tilde{\boldsymbol{w}} = (\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{u}}) + \boldsymbol{u} + \delta \left(\boldsymbol{c} - \frac{\langle \boldsymbol{a}, \boldsymbol{c} \rangle}{\|\boldsymbol{a}\|^2} \boldsymbol{a} \right).$$

Отсюда получаем

$$\langle \boldsymbol{c}, \tilde{\boldsymbol{w}} \rangle = \langle \boldsymbol{c}, \tilde{\boldsymbol{w}} - \tilde{\boldsymbol{u}} \rangle + \langle \boldsymbol{c}, \boldsymbol{u} \rangle + \delta \left(\|\boldsymbol{c}\|^2 - \frac{\langle \boldsymbol{a}, \boldsymbol{c} \rangle^2}{\|\boldsymbol{a}\|^2} \right).$$

По построению вектор $\tilde{\boldsymbol{w}} - \tilde{\boldsymbol{u}}$ ортогонален вектору $\boldsymbol{v} - \boldsymbol{u} = \boldsymbol{c}$. Поэтому $\langle \boldsymbol{c}, \tilde{\boldsymbol{w}} - \tilde{\boldsymbol{u}} \rangle = 0$. Таким образом, последняя формула преобразуется к виду

$$\langle \boldsymbol{c}, \tilde{\boldsymbol{w}} \rangle = \langle \boldsymbol{c}, \boldsymbol{u} \rangle + \delta \left(\|\boldsymbol{c}\|^2 - \frac{\langle \boldsymbol{a}, \boldsymbol{c} \rangle^2}{\|\boldsymbol{a}\|^2} \right). \tag{21}$$

Сопоставляя (15) и (21), с учетом (16) и (17) получаем

$$\langle \boldsymbol{c}, \tilde{\boldsymbol{w}} \rangle < \langle \boldsymbol{c}, \boldsymbol{w} \rangle,$$

что противоречит (13). □

Возвращаясь к задаче ЛП (1), можно сказать следующее. Пусть $\boldsymbol{u} \in M \cap \Gamma(\hat{M})$ и существует единственная рецессивная гиперплоскость $H_{i'}$ ($i' \in \mathcal{I}$) такая, что $\boldsymbol{u} \in H_{i'}$. В этом случае вектор $\boldsymbol{d}$, определяющий направление оптимального целевого пути из точки $\boldsymbol{u}$, в соответствии с утверждением 1 вычисляется по формуле

$$\boldsymbol{d} = \boldsymbol{c} - \frac{\langle \boldsymbol{a}_{i'}, \boldsymbol{u} + \boldsymbol{c} \rangle - b_{i'}}{\|\boldsymbol{a}_{i'}\|^2} \boldsymbol{a}_{i'}. \tag{22}$$

В следующем разделе мы рассмотрим общий случай, когда через точку $\boldsymbol{u}$ проходит несколько гиперплоскостей.

3. Операция псевдопроектирования на линейное многообразие

Пусть $\mathcal{J} \subseteq \{1, \ldots, m\}$, $\mathcal{J} \neq \emptyset$, и $\bigcap\limits_{i \in \mathcal{J}} H_i \neq \emptyset$. В этом случае множество индексов $\mathcal{J}$ определяет в пространстве $\mathbb{R}^n$ линейное многообразие

$$L = \bigcap_{i \in \mathcal{J}} H_i. \tag{23}$$

Обозначим k_L — размерность линейного многообразия L. При $k_L < n-1$ многообразие L не является гиперплоскостью, и для определения вектора движения $\boldsymbol{d}$ по этому многообразию в направлении максимального увеличения целевой функции нельзя применить формулу (22), так как такое линейное многообразие невозможно задать одним линейным уравнением в пространстве $\mathbb{R}^n$. Однако мы можем найти указанный вектор $\boldsymbol{d}$ с помощью операции псевдопроектирования [10]. Определим проекционное отображение $\boldsymbol{\varphi}(\cdot)$:

$$\boldsymbol{\varphi}(\boldsymbol{x}) = \frac{1}{|\mathcal{J}|}\sum_{i\in\mathcal{J}}\boldsymbol{\pi}_{H_i}(\boldsymbol{x}). \tag{24}$$

Известно [12], что отображение $\boldsymbol{\varphi}(\boldsymbol{x})$ является непрерывным L-фейеровским отображением, и последовательность точек

$$\left\{\boldsymbol{x}_k = \boldsymbol{\varphi}^k\left(\boldsymbol{x}_0\right)\right\}_{k=1}^{\infty}, \tag{25}$$

порождаемая этим отображением, начиная с произвольной точки $\boldsymbol{x}_0 \in \mathbb{R}^n$, сходится к точке, принадлежащей L:

$$\boldsymbol{x}_k \to \tilde{\boldsymbol{x}} \in L.$$

Теперь мы готовы дать определение псевдопроекции на линейное многообразие, образуемое пересечением гиперплоскостей.

Определение 3. Пусть $\mathcal{J} \subseteq \{1,\ldots,m\}$, $\mathcal{J} \neq \emptyset$, $\bigcap_{i\in\mathcal{J}} H_i \neq \emptyset$, $\boldsymbol{\varphi}(\cdot)$ — проекционное отображение, определяемое формулой (24). Псевдопроекцией $\boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{x})$ точки $\boldsymbol{x} \in \mathbb{R}^n$ на линейное многообразие $L = \bigcap_{i\in\mathcal{J}} H_i$ называется предельная точка последовательности (25):

$$\lim_{k\to\infty}\left\|\boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{x}) - \boldsymbol{\varphi}^k(\boldsymbol{x})\right\| = 0.$$

Algorithm 1: Вычисление псевдопроекции $\boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{x})$

Require: $H_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle = b_i\}$; $\mathcal{J} \subseteq \{1,\ldots,m\}$; $\mathcal{J} \neq \emptyset$; $\bigcap\limits_{i\in\mathcal{J}} H_i \neq \emptyset$

```
function ρ_J(x)
    k := 0
    x_0 := x
    repeat
        Σ := 0
        for i ∈ J do
            Σ := Σ + (⟨a_i, x_k⟩ − b_i) a_i / ‖a_i‖^2
        end
        x_(k+1) := x_k − Σ/|J|
        ξ_max := 0                                   ▷ Максимальная невязка
        for i ∈ J do
            ξ_i := ‖⟨a_i, x_(k+1)⟩ − b_i‖
            if ξ_i > ξ_max then
                ξ_max := ξ_i
            end
        end
        k := k + 1
    until ξ_max < ε_ξ                               ▷ Малый параметр ε_ξ ;> 0
    return x_k
end
```

Процедура приближенного вычисления псевдопроекции на линейное многообразие представлена в виде алгоритма 1. Дадим краткий комментарий по шагам этого алгоритма. На шаге 2 счетчик итераций k устанавливается в значение ноль. Шаг 3 задает начальное приближение $\boldsymbol{x}_0$. Шаг 4 открывает итерационный цикл вычисления псевдопроекции. Шаги 5–8 для текущего приближения $\boldsymbol{x}_k$ вычисляют сумму из правой части формулы (24). Шаг 9 находит следующее приближение $\boldsymbol{x}_{k+1}$. Шаги 10–16 определяют максимальную невязку ξ для нового приближения относительно всех гиперплоскостей H_i, участвующих в вычислении. На шаге 17 счетчик итераций увеличивается на единицу. Шаг 18 проверяет условие выхода из итерационного цикла. Шаг 19 возвращает в качестве результата полученное приближение $\boldsymbol{x}_k$.

4. Алгоритм движения по граням AlFaMove

В этом разделе мы опишем новый алгоритм AlFaMove (Along Faces Movement), представляющий собой численную реализацию метода поверхностного движения [9]. Алгоритм AlFaMove строит на поверхности допустимого многогранника путь из произвольной граничной точки $\boldsymbol{u}^{(0)} \in M \cap \Gamma(\hat{M})$ до точки $\bar{\boldsymbol{x}}$, являющейся решением задачи ЛП (1). Перемещение по граням допустимого многогранника происходит в направлении наибольшего увеличения значения целевой функции. Путь, построенный в результате такого движения, называется оптимальным целевым путем.

В основе алгоритма AlFaMove лежит процедура $\boldsymbol{D}(\boldsymbol{u})$ вычисления вектора движения $\bar{\boldsymbol{d}}$ по грани допустимого многогранника M из точки $\boldsymbol{u}$ в направлении максимального увеличения значения целевой функции. Процедура $\boldsymbol{D}(\boldsymbol{u})$ представлена в виде алгоритма 2. Схематично работа этого алгоритма изображена на рис. 2.

Дадим краткий комментарий по шагам алгоритма 2. Шаги 2–7 строят множество $\mathcal{U}$, включающее в себя индексы всех гиперплоскостей H_i, проходящих через точку $\boldsymbol{u}$. На шаге 8 вектору направления движения $\bar{\boldsymbol{d}}$ в качестве начального значения присваивается нулевой вектор. На шаге 9 значению целевой функции f присваивается бесконечно малое число. На шаге 10 строится единичный вектор $\boldsymbol{e_c}$, сонаправленный с вектором $\boldsymbol{c}$. На шаге 11 строится точка $\boldsymbol{v}$ путем прибавления вектора $\delta\boldsymbol{e_c}$ к вектору $\boldsymbol{u}$ (см. рис. 2). Здесь δ — «большой» положительный параметр: чем больше δ,

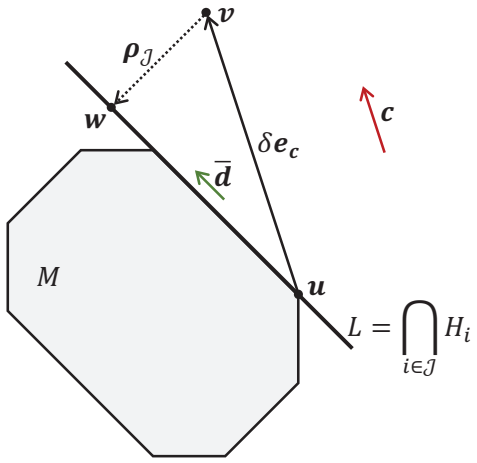


Рис. 2: Вычисление вектора направления движения $\bar{\boldsymbol{d}}$ по грани линейного многообразия L.

Algorithm 2: Вычисление вектора движения $\bar{\boldsymbol{d}} = \boldsymbol{D}(\boldsymbol{u})$

Require: $H_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle = b_i\}$; $\boldsymbol{u} \in \Gamma(M)$

1 **function** $\boldsymbol{D}(\boldsymbol{u})$
2 $\quad \mathcal{U} := \emptyset$ ▷ $\mathcal{U}$ — множество индексов гиперплоскостей H_i, проходящих через $\boldsymbol{u}$
3 $\quad$ **for** $i = 1 \dots m$ **do**
4 $\quad\quad$ **if** $\langle \boldsymbol{a}_i, \boldsymbol{u} \rangle = b_i$ **then**
5 $\quad\quad\quad \mathcal{U} := \mathcal{U} \cup \{i\}$
6 $\quad\quad$ **end**
7 $\quad$ **end**
8 $\quad \bar{\boldsymbol{d}} := \boldsymbol{0}$
9 $\quad f := -\infty$ ▷ f — значение целевой функции $f(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle$
10 $\quad \boldsymbol{e_c} := \boldsymbol{c} / \|\boldsymbol{c}\|$
11 $\quad \boldsymbol{v} := \boldsymbol{u} + \delta \boldsymbol{e_c}$ ▷ Большой параметр $\delta > 0$
12 $\quad$ **for** $\mathcal{J} \in \mathcal{P}(\mathcal{U}) \backslash \emptyset$ **do** ▷ $\mathcal{P}(\mathcal{U})$ — множество всех подмножеств множества $\mathcal{U}$
13 $\quad\quad \boldsymbol{w} := \boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{v})$
14 $\quad\quad \boldsymbol{d} := \boldsymbol{w} - \boldsymbol{u}$
15 $\quad\quad \boldsymbol{e_d} := \boldsymbol{d} / \|\boldsymbol{d}\|$
16 $\quad\quad$ **if** $(\boldsymbol{u} + \tau \boldsymbol{e_d}) \in M$ **then** ▷ Малый параметр $\tau > 0$
17 $\quad\quad\quad$ **if** $\langle \boldsymbol{c}, \boldsymbol{u} + \boldsymbol{e_d} \rangle > f$ **then**
18 $\quad\quad\quad\quad f := \langle \boldsymbol{c}, \boldsymbol{u} + \boldsymbol{e_d} \rangle$
19 $\quad\quad\quad\quad \bar{\boldsymbol{d}} := \boldsymbol{e_d}$
20 $\quad\quad\quad$ **end**
21 $\quad\quad$ **end**
22 $\quad$ **end**
23 $\quad$ **return** $\bar{\boldsymbol{d}}$
24 **end**

тем точнее будет вычислен вектор направления движения $\bar{\boldsymbol{d}}$. Однако при увеличении параметра δ будет увеличиваться и время вычисления псевдопроекции $\boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{v})$ на шаге 13. Цикл **for** на шаге 12 перебирает все возможные комбинации индексов гиперплоскостей, проходящих через точку $\boldsymbol{u}$. Каждой такой комбинации $\mathcal{J}$ соответствует линейное многообразие $L = \bigcap_{i \in \mathcal{J}} H_i$, которое также проходит через точку $\boldsymbol{u}$. Шаг 13 вычисляет точку $\boldsymbol{w}$, выполняя псевдопроекцию точки $\boldsymbol{v}$ на линейное многообразие, соответствующее комбинации $\mathcal{J}$. На шаге 14 вычисляется возможный вектор движения $\boldsymbol{d}$ по текущему линейному многообразию в направлении максимального увеличения значений целевой функции. Шаг 15 вычисляет единичный вектор $\boldsymbol{e_d}$, сонаправленный с вектором $\boldsymbol{d}$. Шаг 16 проверяет, что малое движение из точки $\boldsymbol{u}$ в направлении $\boldsymbol{e_d}$ не выходит за границы допустимого многогранника. Шаг 17 проверяет, что значение целевой функции в точке $(\boldsymbol{u} + \boldsymbol{e_d})$ превышает максимальное значение, полученное на предыдущих итерациях цикла **for** (шаг 12). В этом случае это значение запоминается как максимальное (шаг 18), а найденное направление присваивается вектору $\bar{\boldsymbol{d}}$ (шаг 19). После того, как проверены все возможные комбинации, вектор $\bar{\boldsymbol{d}}$ возвращается в качестве результата (шаг 19). Если ни одна комбинация не прошла проверку на шагах 16–17, то в качестве результата будет возвращен нулевой вектор. Это означает, что любое движение из точки $\boldsymbol{u}$ по поверхности допустимого многогранника локально не приводит к увеличению значения целевой функции.

Algorithm 3: Алгоритм движения по граням AlFaMove

Require: $\hat{H}_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \leqslant b_i\}$; $M = \bigcap\limits_{i=1}^{m} \hat{H}_i$; $\hat{M} = \bigcap\limits_{i \in \mathcal{I}} \hat{H}_i$; $i \in \mathcal{I} \Leftrightarrow \langle \boldsymbol{a}_i, \boldsymbol{c} \rangle > 0$

1 **input** $\boldsymbol{u}_0$
2 **assert** $\boldsymbol{u}_0 \in M \cap \Gamma(\hat{M})$
3 $\boldsymbol{d}_0 := \boldsymbol{D}(\boldsymbol{u}_0)$
4 **assert** $\boldsymbol{d}_0 \neq \boldsymbol{0}$
5 $k := 0$ **repeat**
6 $\quad \boldsymbol{u}_{k+1} := \boldsymbol{u}_k + \mu(\boldsymbol{u}_k, \boldsymbol{d}_k, \epsilon_\mu) \cdot \boldsymbol{d}_k$ ▷ Малый параметр $\epsilon_\mu > 0$
7 $\quad \boldsymbol{d}_{k+1} := \boldsymbol{D}(\boldsymbol{u}_{k+1})$
8 $\quad k := k + 1$
9 **until** $\boldsymbol{d}_k = \boldsymbol{0}$;
10 **output** $\boldsymbol{u}_k$ ▷ Решение задачи ЛП (1)
11 **stop**

Теперь мы готовы описать алгоритм движения по граням AlFaMove, решающий задачу ЛП (1). За основу мы берем алгоритм 1 из работы [9]. Реализация алгоритма AlFaMove на псевдокоде представлена в виде алгоритма 3. Прокомментируем шаги этого алгоритма. На шаге 1 вводится начальное приближение $\boldsymbol{u}^{(0)}$. Это может быть произвольная граничная точка рецессивного многогранника $\hat{M}$, удовлетворяющая условию

$$\boldsymbol{u}_0 \in M \cap \Gamma(\hat{M}).$$

Это условие проверяется на шаге 2. Для получения подходящего начального приближения может применяться алгоритм, реализующий стадию Quest апекс-метода [10]. Шаг 3 вычисляет начальный вектор движения $\boldsymbol{d}_0$. Для этого используется функция $\boldsymbol{D}(\cdot)$, реализованная в алгоритме 2. Функция $\boldsymbol{D}(\cdot)$ выдает вектор единичной длины, либо нулевой вектор. Предполагается, что $\boldsymbol{d}_0 \neq \boldsymbol{0}$[1]. Это условие проверяется на шаге 4. На шаге 5 счетчик итераций k устанавливается в значение ноль. Шаг 5 открывает цикл **repeat**, выполняющий движение по граням допустимого многогранника до тех пор, пока очередной вектор движения $\boldsymbol{d}_k$ не окажется нулевым вектором. Это означает, что последнее найденное приближение $\boldsymbol{u}_k$ является решением задачи ЛП (1). Шаг 7 в теле цикла вычисляет следующее приближение $\boldsymbol{u}_{k+1}$ путем прибавления к вектору $\boldsymbol{u}_k$ единичного вектора $\boldsymbol{d}_k$, умноженного на положительное число, вычисляемое функцией $\mu(\boldsymbol{u}_k, \boldsymbol{d}_k, \epsilon_\mu)$, удовлетворяющей следующим двум условиям:

$$\boldsymbol{u}_k + \mu(\boldsymbol{u}_k, \boldsymbol{d}_k, \epsilon_\mu) \cdot \boldsymbol{d}_k \in M;$$
$$\boldsymbol{u}_k + (\mu(\boldsymbol{u}_k, \boldsymbol{d}_k, \epsilon_\mu) + \epsilon_\mu) \cdot \boldsymbol{d}_k \notin M.$$

Здесь ϵ_μ — малый положительный параметр алгоритма. На практике эта функция может быть легко и эффективно реализована с помощью метода дихотомии. Шаг 8 вычисляет вектор движения $\boldsymbol{d}_{k+1}$ для вновь найденного приближения $\boldsymbol{u}_{k+1}$. Шаг 9 увеличивает счетчик итераций k на единицу. Если последний вектор движения равен нулевому вектору, то на шаге 10 происходит выход из цикла **repeat**/**until**, после чего на шаге 11 выводятся координаты точки $\boldsymbol{u}_k$ в качестве решения задачи ЛП (1). Шаг 12 завершает работу алгоритма AlFaMove.

[1]Равенство вектора $\boldsymbol{d}_0$ нулевому вектору означает, что точка $\boldsymbol{u}_0$ принадлежит гиперплоскости, ортогональной целевому вектору $\boldsymbol{c}$.

5. Параллельная версия алгоритма AlFaMove

Наиболее ресурсоемкой операцией, используемой в алгоритме AlFaMove (алгоритм 3), является операция вычисления вектора движения, выполняемая в цикле на шаге 8. При решении больших задач ЛП она занимает более 90% процессорного времени. Это объясняется тем, что функция вычисления вектора движения $\boldsymbol{D}(\cdot)$, реализованная в виде алгоритма 2, использует в цикле на шаге 13 операцию псевдопроектирования, заключающуюся в последовательном применении отображения $\varphi(\cdot)$, задаваемого формулой (24), к исходной точке (см. алгоритм 1, реализующий вычисление псевдопроекции). Известно, что в случае больших задач ЛП проекционный метод может потребовать значительных временных затрат [13]. Кроме того, следует отметить, что алгоритм 2 в цикле на шаге 12 перебирает все непустые подмножества множества $\mathcal{U}$, включающего в себя индексы гиперплоскостей, проходящих через точку $\boldsymbol{u}$. Если, например, через точку проходит 30 гиперплоскостей, то у нас

Algorithm 4: Параллельная версия алгоритма AlFaMove

мастер		***l*-тый рабочий** $(l = 0, \ldots, L-1)$	
1	**input** $n, m, A, \boldsymbol{b}, \boldsymbol{u}_0$	1	**input** $n, m, A, \boldsymbol{b}, \boldsymbol{c}$
2	$k := 0$	2	
3	**repeat**	3	**repeat**
4	**Bcast** $\boldsymbol{u}_k$	4	**RecvFromMaster** $\boldsymbol{u}_k$
5		5	$\mathcal{U} := [\,]$
6		6	**for** $i = 1 \ldots m$ **do**
7		7	**if** $\langle \boldsymbol{a}_i, \boldsymbol{u}_k \rangle = b_i$ **then**
8		8	$\mathcal{U} := \mathcal{U} +\!\!+ [i]$
9		9	**end**
10		10	**end**
11		11	$K := 2^{\lvert\mathcal{U}\rvert} - 1$
12		12	$L :=$ NumberOfWorkers
13		13	$\mathcal{L}_{map(l)} := [lK/L, \ldots, (l+1)K/L - 1]$
14		14	$\mathcal{L}_{reduce(l)} := \mathit{Map}(\mathrm{F}_{\boldsymbol{u}_k}, \mathcal{L}_{map(l)})$
15		15	$(\boldsymbol{d}_l, f_l) := \mathit{Reduce}(\oplus, \mathcal{L}_{reduce(l)})$
16	**Gather** $\mathcal{L}_{reduce}$	16	**SendToMaster** $(\boldsymbol{d}_l, f_l)$
17	$(\boldsymbol{d}_k, f_k) := \mathit{Reduce}(\oplus, \mathcal{L}_{reduce})$	17	
18	**if** $\boldsymbol{d}_k = \boldsymbol{0}$ **then**	18	
19	$\mathit{exit} :=$ **true**	19	
20	**else**	20	
21	$\boldsymbol{u}_{k+1} := \boldsymbol{u}_k + \mu(\boldsymbol{u}_k, \boldsymbol{d}_k, \epsilon_\mu) \cdot \boldsymbol{d}_k$	21	
22	$k := k + 1$	22	
23	$\mathit{exit} :=$ **false**	23	
24	**end**	24	
25	**Bcast** exit	25	**RecvFromMaster** exit
26	**until** exit;	26	**until** exit;
27	**output** $\boldsymbol{u}_k, f_k$	27	
28	**stop**	28	**stop**

получится $2^{30}-1 = 1\,073\,741\,823$ непустых подмножества. Перебор такого количества подмножеств потребует суперкомпьютерных мощностей. Потому мы разработали параллельную версию алгоритма AlFaMove, представленную в виде алгоритма 4.

Параллельный алгоритм 4 построен на основе модели параллельных вычислений BSF [14], ориентированной на кластерные вычислительные системы. Модель BSF использует схему распараллеливания «мастер–рабочие» и требует представление алгоритма в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*.

В качестве второго параметра функции высшего порядка *Map* в алгоритме 4 используется список $\mathcal{L}_{map} = \left[1, \ldots, 2^K\right]$, содержащий порядковые номера всех подмножеств множества $\mathcal{U}$, за исключением пустого. Здесь $K = 2^{|\mathcal{U}|}$. В качестве первого параметра фигурирует параметризованная функция

$$\mathrm{F}_{\boldsymbol{u}} : \left\{1, \ldots, 2^K\right\} \to \mathbb{R}^n \times \mathbb{R},$$

определенная следующим образом:

$$\begin{aligned}
&\mathrm{F}_{\boldsymbol{u}}(j) = (\boldsymbol{d}_j, f_j);\\
&\boldsymbol{d}_j = \begin{cases} \boldsymbol{w}-\boldsymbol{u}, & \text{если } (\boldsymbol{u}+\tau\boldsymbol{d}/\|\boldsymbol{d}\|) \in M \wedge \langle \boldsymbol{c}, \boldsymbol{w}\rangle > \langle \boldsymbol{c}, \boldsymbol{u}\rangle;\\ \boldsymbol{0}, & \text{если } (\boldsymbol{u}+\tau\boldsymbol{d}/\|\boldsymbol{d}\|) \notin M \vee \langle \boldsymbol{c}, \boldsymbol{w}\rangle \leqslant \langle \boldsymbol{c}, \boldsymbol{u}\rangle; \end{cases}\\
&f_j = \begin{cases} \langle \boldsymbol{c}, \boldsymbol{w}\rangle, & \text{если } (\boldsymbol{u}+\tau\boldsymbol{d}/\|\boldsymbol{d}\|) \in M \wedge \langle \boldsymbol{c}, \boldsymbol{w}\rangle > \langle \boldsymbol{c}, \boldsymbol{u}\rangle;\\ -\infty, & \text{если } (\boldsymbol{u}+\tau\boldsymbol{d}/\|\boldsymbol{d}\|) \notin M \vee \langle \boldsymbol{c}, \boldsymbol{w}\rangle \leqslant \langle \boldsymbol{c}, \boldsymbol{u}\rangle, \end{cases}
\end{aligned} \tag{26}$$

где

$$\boldsymbol{w} = \boldsymbol{\rho}_{\sigma(j)}(\boldsymbol{u} + \delta\boldsymbol{c}/\|\boldsymbol{c}\|). \tag{27}$$

Очевидно, что семантика функции $\mathrm{F}_{\boldsymbol{u}}(\cdot)$ однозначно определяется алгоритмом 2. Функция $\sigma(j)$, используемая в формуле (27), преобразует натуральное число j в двоичное представление, состоящее из K разрядов. Каждому разряду, содержащему единицу, сопоставляется соответствующая гиперплоскость из списка $\mathcal{U}$. Таким образом получается подмножество гиперплоскостей. Например, пусть $\mathcal{U} = [H_2, H_4, H_7, H_9]$ и $j = 5$. В этом случае $K = 2^4 - 1$. Функция $\sigma(\cdot)$ преобразует число 5 в двоичную запись из 4-х разрядов 0101 и возвращает в качестве результата подмножество гиперплоскостей $J = \{H_2, H_7\}$. Таким образом, функция высшего порядка $Map\,(\mathrm{F}_{\boldsymbol{u}}, \mathcal{L}_{map})$ преобразует список номеров подмножеств $\mathcal{L}_{map}$ в список пар $(\boldsymbol{d}_j, f_j)$:

$$Map\,(\mathrm{F}_{\boldsymbol{u}}, \mathcal{L}_{map}) = [\mathrm{F}_{\boldsymbol{u}}(1), \ldots, \mathrm{F}_{\boldsymbol{u}}\,(K)] = [(\boldsymbol{d}_1, f_1), \ldots, (\boldsymbol{d}_K, f_K)]\,.$$

Здесь $\boldsymbol{d}_j$ является вектором движения по соответствующей грани допустимого многогранника M, а f_j — значение целевой функции, которое при этом достигается.

Обозначим $\mathcal{L}_{reduce} = [(\boldsymbol{d}_1, f_1), \ldots, (\boldsymbol{d}_K, f_K)]$. Определим бинарную ассоциативную операцию

$$\oplus : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n \times \mathbb{R},$$

являющуюся первым параметром функции высшего порядка *Reduce*:

$$(\boldsymbol{d}', f') \oplus (\boldsymbol{d}'', f'') = \begin{cases} (\boldsymbol{d}', f')\,, & \text{если } f' \geqslant f'';\\ (\boldsymbol{d}'', f'')\,, & \text{если } f' < f''. \end{cases} \tag{28}$$

Функция высшего порядка $Reduce\,(\oplus, \mathcal{L}_{reduce})$ редуцирует список $\mathcal{L}_{reduce}$ к одной паре путем последовательного применения операции $\oplus$ ко всем элементам списка:

$$Reduce\,(\oplus, \mathcal{L}_{reduce}) = (\boldsymbol{d}_1, f_1) \oplus \ldots \oplus (\boldsymbol{d}_K, f_K) = (\boldsymbol{d}_{j'}, f_{j'}), \quad \text{где} \quad j' = \arg\max_{1 \leqslant j \leqslant K} f_j.$$

Параллельная работа алгоритма 4 организована по схеме «мастер–рабочие» и включает в себя $L+1$ процесс: один процесс–мастер и L процессов–рабочих. Процесс–мастер осуществляет общее управление вычислениями, распределяет работу между процессами–рабочими, получает от них результаты и формирует итоговый результат. Для простоты будем предполагать, что количество подмножеств K кратно количеству рабочих L. На шаге 1 мастер вводит исходные данные задачи ЛП и начальную точку $\boldsymbol{u}_0$. На шаге 2 мастер устанавливает счетчик итераций k в значение ноль. Шаги 3–26 реализуют основной цикл **repeat/until**, вычисляющий решение задачи ЛП (1). На шаге 4 мастер рассылает текущее приближение $\boldsymbol{u}_k$ всем рабочим. На шаге 16 он получает от рабочих частичные результаты, которые на шаге 17 редуцируются в пару $(\boldsymbol{d}_k, f_k)$. Если на шаге 18 выполняется условие $\boldsymbol{d}_k = \boldsymbol{0}$, то решение найдено (мы предполагаем $\boldsymbol{d}_0 \neq \boldsymbol{0}$). В этом случае на шаге 19 мастер присваивает логической переменной *exit* значение **true**. Если $\boldsymbol{d}_k \neq \boldsymbol{0}$, то мастер на шаге 21 вычисляет следующее приближение $\boldsymbol{u}_{k+1}$, увеличивает на шаге 22 счетчик итераций k на единицу и на шаге 23 присваивает логической переменной *exit* значение **false**. На шаге 25 мастер рассылает всем рабочим значение логической переменной *exit*. Если логическая переменная *exit* принимает значение «истина», цикл **repeat–until** завершается на шаге 26. На шаге 27 мастер выводит в качестве результата последнее приближение $\boldsymbol{u}_k$ и оптимальное значение целевой функции f_k. Шаг 28 завершает работу процесса–мастера.

Все рабочие выполняют один и тот же код, но над различными данными. На шаге 1 l-тый рабочий вводит исходные данные задачи ЛП. Цикл **repeat/until** рабочего (шаги 3-26) соответствует циклу **repeat/until** мастера. На шаге 4 рабочий получает от мастера текущее приближение $\boldsymbol{u}_k$. Затем он формирует подсписок $\mathcal{L}_{map(l)}$ своих порядковых номеров подмножеств для последующей обработки (шаги 5–13). Для удобства программирования нумерация подмножеств начинается с нуля. Подсписки различных рабочих не пересекаются, и их объединение дает полный список порядковых номеров всех рассматриваемых подмножеств:

$$\mathcal{L}_{map} = \mathcal{L}_{map(0)} +\!\!+ \ldots +\!\!+ \mathcal{L}_{map(L-1)}. \tag{29}$$

Символ $+\!\!+$ здесь обозначает операцию конкатенации списков. На шаге 14 рабочий вызывает функцию высшего порядка Map, которая, в свою очередь, применяет параметризованную функцию $\mathrm{F}_{\boldsymbol{u}_k}$, определенную формулами (26), ко всем элементам подсписка $\mathcal{L}_{map(l)}$, формируя на выходе подсписок пар $\mathcal{L}_{reduce(l)}$. Этот подсписок на шаге 15 редуцируется рабочим в единственную пару $(\boldsymbol{d}_l, f_l)$ с помощью функции высшего порядка *Reduce*, которая последовательно применяет бинарную операцию $\oplus$, определенную по формуле (28), ко всем элементам подсписка $\mathcal{L}_{reduce(l)}$. Результат пересылается мастеру на шаге 16. На шаге 25 рабочий получает от мастера значение логической переменной *exit*. Если эта переменная принимает значение **true**, то рабочий процесс завершается. В противном случае продолжает выполняться цикл **repeat–until**. Операторы обмена **Bcast**, **Gather**, **RecvFromMaster** и **SendToMaster** обеспечивают неявную синхронизацию работы процесса–мастера и процессов–рабочих.

6. Вычислительные эксперименты

Мы реализовали параллельную версию алгоритма AlFaMove на языке C++ с использованием программного BSF-каркаса [15], базирующегося на модели параллельных вычислений BSF [14]. BSF-каркас инкапсулирует все аспекты, связанные с распараллеливанием программы на основе библиотеки MPI. Исходные коды параллельной реализации алгоритма AlFaMove решения задач ЛП свободно доступны в репозитории GitHub по адресу `https://github.com/leonid-sokolinsky/BSF-Surface-movement-method`. С помощью этой программы мы исследовали масштабируемость алгоритма AlFaMove. Масштабные вычислительные эксперименты проводились на вычислительном кластере «Торнадо ЮУрГУ» [16], характеристики которого представлены в табл. 1. В качестве тестов мы использовали искусственные задачи, полученные с помощью генератора случайных задач ЛП FRaGenLP [17]. Все сгенерированные задачи доступны на GitHub по адресу `https://github.com/leonid-sokolinsky/Random-LP-Problems`. Верификация решений, выдаваемых алгоритмом AlFaMove, осуществлялась программой VaLiPro [18]. Была выполнена серия вычислительных экспериментов, в которой для задач ЛП различной размерности исследовалось ускорение и затраченное время в зависимости от количества используемых процессорных узлов кластера. Результаты этих экспериментов представлены на рис. 3. В данном контексте ускорение $\alpha(L)$ определялось как отношение времени $T(1)$

Таблица 1: Характеристики кластера «Торнадо ЮУрГУ»

Параметр	Значение
Количество процессорных узлов	480
Процессоры	Intel Xeon X5680 (6 cores, 3.33 GHz)
Число процессоров на узел	2
Память на узел	24 GB DDR3
Соединительная сеть	InfiniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS

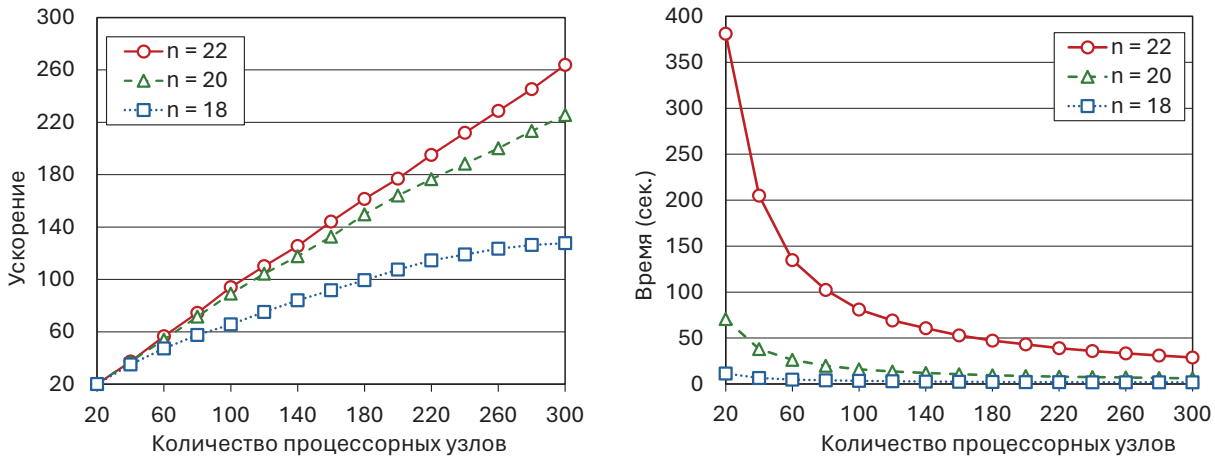


Рис. 3: Ускорение и временные затраты алгоритма AlFaMove.

решения задачи на конфигурации с узлом-мастером и единственным узлом-рабочим ко времени $T(L)$ решения той же задачи на конфигурации с узлом-мастером и L узлами-рабочими:

$$\alpha(L) = \frac{T(1)}{T(L)}. \tag{30}$$

Вычисления проводились для следующих размерностей: 18, 20 и 22. Число ограничений соответственно составило 37, 41 и 45. Для размерности $n = 22$ длина каждого из списков $\mathcal{L}_{map}$ и $\mathcal{L}_{reduce}$ составила 4 194 304 элементов. Следует отметить, что вычисления проводились с двойной точностью, при которой число с плавающей точкой занимает в оперативной памяти 64 бита.

Эксперименты показали, что алгоритм AlFaMove хорошо масштабируется уже на относительно небольших размерностях. Задачи размерностей $n = 20$ и $n = 22$ показали на 300 процессорных узлах ускорение, близкое к линейному (см. левый график). Для задачи размерности $n = 22$ эффективность распараллеливания на 300 процессорных узлах составила 88%. При $n = 20$ и $n = 18$ эффективность оказалась равной 75% и 43% соответственно. В тоже время правый график показывает, что с ростом размерности наблюдается экспоненциальный рост времени вычислений.

Мы не смогли протестировать алгоритм AlFaMove на задачах из репозитория Netlib-LP [19], так как во всех этих задачах количество гиперплоскостей, проходящих через начальную точку $\boldsymbol{u}_0$, превышало число 30, что соответствует количеству возможных комбинаций, равному 1 073 741 824. Это — максимальный размер массива, допускаемый используемым компилятором. В целях преодоления «проклятия размерности» мы планируем в ближайшее время разработать и реализовать стохастический вариант алгоритма AlFaMove, который не будет требовать создания больших массивов.

Заключение

В статье предложен новый масштабируемый алгоритм линейного программирования, получивший название «AlFaMove». Алгоритм является численной реализацией метода поверхностного движения, ранее предложенного авторами. Ключевой особенностью этого метода является построение оптимального пути на поверхности допустимого многогранника от начальной точки к решению задачи линейного программирования. Под оптимальным путем понимается кратчайший путь по поверхности допустимой области в направлении максимального увеличения значений целевой функции. Практическая значимость предложенного метода состоит в том, что он открывает возможность применения искусственных нейронных сетей прямого распространения для решения нестационарных многомерных задач линейного программирования в режиме реального времени.

Теоретической основой алгоритма AlFaMove является операция построения псевдопроекции на линейные многообразия, формирующие грани допустимого многогранника. Псевдопроекция реализуется на основе фейеровского процесса и является обобщением понятий ортогональной проекции на линейное многообразие и метрической проекции на выпуклое множество. В случае грани, образуемой гиперплоскостью псевдопроекция сводится к ортогональной проекции. Доказано, что путь по гиперплоскости, построенный с помощью градиента целевой функции и ортогональной проекции, является оптимальным. Приведен алгоритм проекционного типа для

построения псевдопроекции на линейные многообразия меньших размерностей, образуемые пересечением гиперплоскостей. Представлено формализованное описание алгоритма AlFaMove, строящего оптимальный путь на поверхности допустимого многогранника. В основе алгоритма AlFaMove лежит процедура вычисления вектора движения по грани допустимого многогранника из точки текущего приближения в направлении максимального увеличения значения целевой функции. Дано формализованное описание этой процедуры.

Алгоритмы проекционного типа характеризуются низкой скоростью сходимости, зависящей от углов между гиперплоскостями, образующими линейное многообразие. Также отмечено, что при вычислении вектора движения возникает переборная задача комбинаторного типа, имеющая высокую пространственную и временную сложность. Представлена параллельная версия алгоритма AlFaMove, ориентированная на кластерные вычислительные системы. Параллельная версия реализована на языке C++ с использованием программного BSF-каркаса, основанного на модели параллельных вычислений BSF. Проведены эксперименты по исследованию масштабируемости алгоритма AlFaMove на кластерной вычислительной системе. Вычислительные эксперименты показали, что задача линейного программирования с 24 переменными и 49 ограничениями демонстрирует линейное ускорение близкое к оптимальному на 320 процессорных узлах кластера. Задачи большей размерности приводили к ошибке компилятора, связанной с превышением максимального допустимого размера массивов.

В качестве направлений дальнейших исследований выделим следующие. Мы планируем разработать новый, более эффективный метод построения пути на поверхности допустимого многогранника, приводящего к решению задачи линейного программирования. Основная идея состоит в сокращении перебираемых комбинаций гиперплоскостей при определении направления движения. Это может быть достигнуто, если мы ограничимся перемещениями только по ребрам многогранника (линейным многообразиям размерности один). Проблема пространственной сложности может быть решена путем перехода к стохастическим методам выбора направления движения.

Список литературы

[1] Optimization in Large Scale Problems: Industry 4.0 and Society 5.0 Applications / ed. by M. Fathi, M. Khakifirooz, P.M. Pardalos. Cham, Switzerland: Springer, 2019. https://doi.org/10.1007/978-3-030-28565-4

[2] Solving Large-Scale Production Scheduling and Planning in the Process Industries. Cham, Switzerland: Springer, 2019. https://doi.org/10.1007/978-3-030-01183-3

[3] Schlenkrich M., Parragh S.N. Solving large scale industrial production scheduling problems with complex constraints: an overview of the state-of-the-art // 4th International Conference on Industry 4.0 and Smart Manufacturing. Procedia Computer Science. Vol. 217 / ed. by F. Longo, M. Affenzeller, A. Padovano, W. Shen. Elsevier, 2023, P. 1028–1037. https://doi.org/10.1016/J.PROCS.2022.12.301,

[4] Соколинская И.М., Соколинский Л.Б. О решении задачи линейного программирования в эпоху больших данных // Параллельные вычислительные технологии (ПаВТ'2017). Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2017. С. 471–484.
http://omega.sp.susu.ru/pavt2017/short/014.pdf

[5] Branke J. Optimization in Dynamic Environments // Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation, vol. 3. Boston, MA: Springer, 2002. P. 13–29.
https://doi.org/10.1007/978-1-4615-0911-0_2

[6] Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.

[7] Зоркальцев В.И. Мокрый И.В. Алгоритмы внутренних точек в линейной оптимизации // Сибирский журнал индустриальной математики. 2018. Т. 21, 1 (73), С. 11–20.
https://doi.org/10.17377/sibjim.2018.21.102

[8] Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method // Algorithms, Probability, Networks, and Games. Lecture Notes in Computer Science, vol. 9295 / ed. by C. Zaroliagis, G. Pantziou, S. Kontogiannis. Cham: Springer, 2015. P. 281–307.
https://doi.org/10.1007/978-3-319-24024-4_17

[9] Ольховский Н.А., Соколинский Л.Б. О новом методе линейного программирования / Вычислительные методы и программирование. 2023. Т. 24, № 4. С. 408–429.
https://doi.org/10.26089/NumMet.v24r428

[10] Соколинский Л.Б., Соколинская И.М. О новой версии апекс-метода для решения задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 5–46.
https://doi.org/10.14529/cmse230201

[11] Мальцев А.И. Основы линейной алгебры. Москва: Наука, 1970. 402 с.

[12] Васин В.В., Ерёмин И.И. Операторы и итерационные процессы фейеровского типа. Теория и приложения. Екатеринбург: УрО РАН, 2005. 211 с.

[13] Gould N.I. How good are projection methods for convex feasibility problems? // Computational Optimization and Applications. 2008. Vol. 40, no. 1. P. 1–12.
https://doi.org/10.1007/S10589-007-9073-5

[14] Sokolinsky L.B. BSF: A parallel computation model for scalability estimation of iterative numerical algorithms on cluster computing systems // Journal of Parallel and Distributed Computing. 2021. Vol. 149. P. 193–206.
https://doi.org/10.1016/j.jpdc.2020.12.009

[15] Sokolinsky L.B. BSF-skeleton: A Template for Parallelization of Iterative Numerical Algorithms on Cluster Computing Systems // MethodsX. 2021. Vol. 8. Art. 101437.
https://doi.org/10.1016/j.mex.2021.101437

[16] Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC Resources of South Ural State University // Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2022. P. 43–55.
https://doi.org/10.1007/978-3-031-11623-0_4

[17] Соколинский Л.Б., Соколинская И.М. О генерации случайных задач линейного программирования на кластерных вычислительных системах // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 38–52.
https://doi.org/10.14529/cmse210103

[18] Соколинский Л.Б., Соколинская И.М. О валидации решений задач линейного программирования на кластерных вычислительных системах // Вычислительные методы и программирование. 2021. Т. 22, № 4. С. 252–261.
https://doi.org/10.26089/NUMMET.V22R416

[19] Gay D.M. Electronic mail distribution of linear programming test problems // Mathematical Programming Society COAL Bulletin. 1985. Vol. 13. P. 10–12.

Исследование перспективности использования нейронных сетей для поиска новых высокомолекулярных высокоэнергетических веществ[1]

В.М. Волохов[1], Е.С. Амосова[1], В.В. Парахин[1,2], Д.Б. Лемперт[1], И.И. Акостелов[1,3], В.В. Воеводин[4]

[1]Федеральный исследовательский центр проблем химической физики и медицинской химии РАН, Черноголовка
[2]Институт органической химии им. Н.Д. Зелинского РАН, Москва
[3]Московский государственный университет им. М.В. Ломоносова
[4]Научно-исследовательский вычислительный центр Московского государственного университета им. М.В. Ломоносова, Москва

В статье на базе нескольких нейронных сетей и баз данных, содержащих информацию о структуре и физико-химических свойствах большого числа (более 130 000) молекул, проведено обучение нейронных сетей и дан анализ перспективности их использования. Предсказательные модели, получившиеся после обучения выбранных архитектур нейронных сетей на наборах данных, были протестированы на четырех тестовых группах молекул. Проведено сравнение предсказательных моделей по точности и обобщаемости.

Ключевые слова: высокоэнергетические материалы, нейронные сети, предсказательное моделирование

1. Введение

В настоящее время в развитых странах активно ведутся работы по созданию новых высокоэнергетических веществ перспективных для применения в различных областях науки и техники. В последние два десятилетия активно развиваются современные методы их компьютерного моделирования. Один из таких современных подходов к созданию новых высокоэнергетических веществ – это компьютерный дизайн молекул новых веществ, как правило, ещё не синтезированных, но по различным соображениям перспективных для применения в различных областях. Фундаментальной проблемой, стоящей перед наукой о материалах, является построение модели «структура-свойство», которая позволила бы по заранее заданным свойствам вещества определять его структуру. Казалось бы, квантовая механика вообще и квантовая

[1]Работа выполнена по темам государственных заданий, №№ госрегистрации 124020100045-5 и 124013100856-9. Расчеты ресурсоемкими методами осуществлялись за счет гранта Российского научного фонда (проект № 23-71-00005).

химия в частности позволяют с высокой точностью определить структуру молекулы вещества (хотя бы в газовой фазе) и рассчитать некоторые его свойства, например, энтальпию образования. Созданы огромные базы данных [1, 2, 3, 4, 5, 6, 7], в которых собраны рассчитанные квантово-химическими методами структурные характеристики молекул (количество атомов в молекуле, их координаты, межатомные расстояния и углы) и различные физико-химические свойства (геометрии минимальной энергии, гармонические частоты, дипольные моменты, поляризуемость, энергии атомизации, энтальпии атомизации и свободные энергии). Однако выявление явных и неявных связей между структурой молекул и их свойствами удаётся осуществить только для небольших групп структурно подобных молекул [8, 9, 10, 11, 12].

В этой связи чрезвычайно перспективными представляются новые активно развивающиеся технологии искусственного интеллекта, нейронных сетей, глубокого обучения и хемоинформатики.

Исследование вещества на основе компьютерных и информационных технологий, предваряющее его синтез, позволяет существенно экономить время и средства и дает надёжные средства контроля процесса синтеза (например, по инфракрасным (ИК) спектрам). Для получения наиболее достоверных результатов оценки перспективности того или иного соединения в энергетических составах различного назначения необходимо иметь как можно более точные величины энтальпии образования (ΔH_f^o) компонентов. Наиболее точные величины ΔH_f^o дают (наряду с экспериментальными значениями) квантово-химические расчёты на основе ab initio подходов. Современные методы квантово-химических расчётов позволяют получить усредненное отклонение результатов расчётов от экспериментальных данных 0.83 ккал/моль (для высокоэнтальпийных веществ отклонение от эксперимента менее 1%). Кроме того, и, самое важное: квантово-химические методы позволяют осуществлять компьютерный дизайн молекул перспективных, ещё не синтезированных соединений, с высокой точностью рассчитывать их физико-химические (в частности, термохимические) параметры, и детально исследовать их зависимость от структуры молекулы.

В настоящее время в открытых источниках накоплен обширный массив данных по энтальпиям образования веществ и различным структурным свойствам. Это позволяет применять методы компьютерной обработки информации, включая методы хемоинформатики и нейронных сетей, для исследования связей «структура-свойство». Применение нейронных сетей в перспективе может уменьшить требования к вычислительным ресурсам по сравнению с квантово-химическими методами в сотни раз, что может значительно ускорить разработку новых высокоэнергетических веществ.

Целью данной работы стало исследование перспективности использования нейронных сетей для поиска новых высокоэнергетических веществ и предсказания их физико-химических свойств.

2. Методология

2.1. Нейронные сети

Для предсказания свойств молекул и химических взаимодействий, учитывая их трехмерную структуру, разработано много глубоких нейронных сетей, которые имеют как много общего, так и некоторые отличия, связанные с особенностями исследуемых молекул.

Их основные характеристики:

1) Создают эффективные векторные представления молекул, учитывая их трехмерную геометрию и химическую структуру. Эти представления позволяют улавливать сложные химические особенности молекул и их взаимодействий.
2) Способны моделировать сложные химические взаимодействия между атомами и функциональными группами в молекулах. Это включает в себя взаимодействия водородных связей, взаимодействия Ван-дер-Ваальса, электростатические взаимодействия и другие.
3) Учитывают трехмерную структуру молекул, включая расстояния и углы между атомами, а также их пространственное расположение. Это позволяет более точно моделировать химические свойства и взаимодействия.
4) Обучаются на основе квантово-химических данных, что обеспечивает высокую точность предсказаний химических свойств и взаимодействий.
5) Могут быть использованы для предсказания различных химических свойств молекул, анализа молекулярных взаимодействий, дизайна новых химических соединений и других приложений в химическом моделировании.

В целом, нейронные сети представляют собой мощный инструмент для моделирования и анализа химических веществ, обеспечивая высокую точность предсказаний и учитывая их трёхмерную структуру и взаимодействия.

В данной работе из всех представленных в литературе архитектур были выбраны для обучения четыре наиболее точных и современных: Allegro [13] (2023), DimeNet++ [14] (2020), PaiNN [15] (2021), TorchMD-NET ET [16] (2022).

2.2. Базы данных

Для обучения и валидации нейронных сетей используются различные наборы данных, широко представленные в литературе. В данной работе для обучения был выбран набор данных QM9 [1], содержащий информацию о геометрических, энергетических, электронных и термодинамических свойствах молекул, входящих в состав набора GDB-17 [17]. QM9 широко используется в машинном обучении и компьютерном моделировании для разработки и оценки методов предсказания химических свойств молекул.

База данных QM9 содержит информацию о 134 тысячах органических молекул в газовой фазе, содержащих до 9 тяжёлых атомов (углерод, кислород, азот, фтор). Для каждой молекулы предоставляется информация о её геометрии, числе атомов, связей, углов и длин связей. База данных содержит разнообразные химические свойства молекул, такие как энергии основного состояния, энергии возбуждения, энтальпии образования, теплоёмкости, электронные структуры и др. Все данные в базе получены из квантово-химических расчётов с использованием методов первых принципов (ab initio), таких как метод Хартри-Фока и методы функционала плотности. База данных QM9 является популярным выбором для обучения и тестирования алгоритмов машинного обучения, направленных на предсказание химических свойств молекул и разработку новых методов в области химического моделирования. Её широкое использование связано с высоким качеством данных и разнообразием информации о химических свойствах органических молекул в газовой фазе.

База данных GDB-17 - это набор химических структур, содержащий все возможные устойчивые молекулы, состоящие из 17 атомов углерода, водорода, азота, кис-

лорода и фтора, которые могут быть сгенерированы с использованием ограничений на валентность и структурную целостность. База данных GDB-17 содержит огромное количество уникальных молекул: на момент создания базы данных в неё вошли миллиарды молекул, но точное число молекул в GDB-17 может изменяться в зависимости от критериев отбора и обновлений базы данных. Каждая молекула в базе представлена в виде графа, где атомы представляют узлы, а химические связи - рёбра. Эта форма представления позволяет компактно описать структуру молекулы и её химические свойства. Молекулы генерируются с помощью алгоритмов, которые учитывают правила валентности и структурную целостность. Это позволяет создавать только устойчивые и реалистичные молекулы, что делает базу данных полезной для различных химических исследований. База данных GDB-17 используется в различных областях науки, включая химию, машинное обучение и компьютерное моделирование. Её разнообразие молекулярных структур делает её ценным инструментом для исследования химических свойств и разработки новых материалов и лекарственных препаратов.

3. Обсуждение результатов

На основе каждой архитектуры и набора данных QM9 были обучены модели для предсказания энергии атомизации из геометрии молекул. Хотя нейросетевые модели и позволяют предсказать широкий набор молекулярных характеристик, энергия атомизации является наиболее важной для предсказания, что связано с повышенной точностью предсказания именно нормализованных величин. Также, из энергии атомизации и энергии составляющих молекулу атомов можно алгебраически рассчитать электронную энергию молекулы. Среднее абсолютное отклонение предсказаний моделей от расчетных данных (квантово-химические расчеты высокого уровня) при валидации на QM9 у всех моделей составило не более 0,2 ккал/моль для всех архитектур.

Все использованные модели были применены для предсказания энергии атомизации группы веществ с азокси-группой, исследованных нами ранее [8]. Было обнаружено, что точность предсказания энергии таких веществ сильно варьируется в процессе обучения модели и не достигает удовлетворительных величин. К концу обучения ошибка предсказания в среднем лишь увеличивалась и составляла 24 ккал/моль для модели на основе архитектуры PaiNN. Предсказательные модели были протестированы на группе молекул моносахаридов, имеющих более простое строение. Точность предсказания энергии моносахаридов варьировалась слабо и достигала 6 ккал/моль для модели на основе архитектуры PaiNN [18].

Для объяснения полученных результатов был проанализирован состав набора данных QM9, основанного на наборе молекул GDB-17. QM9 является достаточно плохим представлением пространства химических веществ из-за того, что молекулы в GDB-17 сгенерированы программным образом и при этом не включают множества важных структурных фрагментов и функциональных групп: неароматических двойных связей, алифатических нитрогрупп, азидов, алифатических галогенов, аминалей и других.

Проведен поиск альтернативных наборов данных в литературе, по итогам которого были отобраны наборы данных PC9 [2], OD9 [3] (и два его подмножества), содержащие данные по молекулам, включающим до 9 тяжёлых атомов CNOF, и

Alchemy [4], содержащий данные по молекулам, включающим до 14 тяжёлых атомов CNOF. Благодаря использованию одного и того же базиса при квантово-химических вычислениях характеристик в QM9 и Alchemy, был дополнительно создан набор данных AlQM9 - объединение QM9 и Alchemy. Также рассматривались наборы данных ANI-1[5] и SPICE [6], но для обучения предсказанию молекулярных характеристик данные наборы не подошли, так как содержат большое количество конформаций относительно малого числа молекул. Данные проекта PubChemQC [7], наоборот, содержат слишком большое число молекул, что сделало бы обучение с использованием данного набора слишком продолжительным.

Расчётные квантово-химические характеристики в вышеуказанных наборах данных были получены при вычислениях в различных базисах, поэтому была проведена оценка корректности сравнения результатов обучения в различных базисах. Так как квантово-химические вычисления в более полных базисах требуют большего количества вычислительных ресурсов, можно предположить, что характеристики, полученные в более полных базисных наборах, будут предсказываться с меньшей точностью – вычислительная сложность нейросетевых моделей фиксируется на стадии проектирования архитектуры и не изменяется. Наиболее подходящим для выполнения этой оценки был набор молекул QM9 с характеристиками, рассчитанными методом G4MP2 [19]. Хоть G4MP2 является комплексным методом, а не просто более полным базисом, более подходящих наборов данных в литературе на данный момент не опубликовано. По итогам обучения моделей TorchMD-NET ET и Dimenet++ был сделан вывод, что изменение значений молекулярных характеристик при переходе от QM9 к QM9-G4MP2 не оказало существенного эффекта на точность предсказаний использованной архитектуры. Можно заключить, что точность модели выбранной архитектуры не уменьшается в зависимости от того, какие характеристики (полученные методом B3LYP или G4MP2) предсказываются этой моделью.

В наборах данных PC9 и OD9 отсутствовали расчётные данные по энергии нулевой точки, поэтому, чтобы получить единую характеристику для сравнения результатов обучения на всех наборах, данные QM9 и Alchemy были дополнительно обработаны: из энергии атомизации была вычтена энергия нулевой точки.

Стандартный подход к оценке точности обучаемых моделей, описанный в литературе, предполагает проведение оценки вычисленных и предсказанных характеристик на том же наборе данных, на котором проходило обучение модели. В нашей работе оценка точности обучаемых моделей была проведена путём сравнения предсказаний и вычисленных характеристик на четырёх группах молекул, соответствующих различным распространенным классам органических веществ. Предложенный способ позволяет лучше оценить переносимость модели при предсказании ранее неизвестных молекул. Например, точность модели PaiNN при обучении и валидации на QM9 составляет 0.169 ккал/моль, но при этом на тестовой группе моносахаридов эта модель демонстрирует среднюю ошибку 6 ккал/моль. Точность модели при обучении и валидации на PC9 составляет примерно 0.286 ккал/моль, а средняя ошибка на группе моносахаридов - 0.5 ккал/моль [18]. Несмотря на то что обучение на QM9 позволяет лучше предсказывать энергии молекул из QM9, чем обучение на PC9 позволяет предсказывать энергии из PC9, на ранее неизвестных молекулах точность может различаться в обратную сторону.

Четыре тестовые группы молекул были сформированы на основе наших предыдущих исследований следующим образом: в первую группу вошли 8 ранее исследо-

ванных молекул с азокси-группой, во вторую - 6 распространённых и применяемых на практике взрывчатых веществ, в третью - 7 распространённых нейромодуляторов и нейротрансмиттеров, и в четвёртую - 6 моносахаридов. Такой выбор обусловлен перспективами практического применения и предполагаемой градацией сложности предсказания групп, от первой с наибольшим разнообразием и наличием нескольких редких функциональных групп, до последней с наиболее распространенными функциональными группами и простым строением. Для всех молекул были рассчитаны молекулярные характеристики с помощью квантово-химического программного комплекса NWChem в трёх базисах, соответствующих различным используемым наборам данных: B3LYP/6-31G(2df,p) для QM9 и Alchemy, B3LYP/6-31G(d) для PC9 и B3LYP/3-21G для OD9, и рассчитаны энергии атомизации без учета энергии нулевой точки для сравнения с предсказанными характеристиками.

Точность в каждой группе оценивалась по среднему абсолютному отклонению предсказанных значений от рассчитанных значений. Общая точность оценивалась как среднее абсолютное от точности для каждой из тестовых групп, в качестве метрики обобщаемости использовалось стандартное отклонение точности для каждой из тестовых групп. Из изученных моделей наиболее точной и обобщаемой является архитектура TorchMD-NET ET, обученная на PC9. Её среднее абсолютное отклонение от расчётных данных - 0.8 ккал/моль, а стандартное отклонение составляет лишь 0.2 ккал/моль. Среди остальных моделей лучше всего себя показывают архитектуры, обученные на PC9. Хуже всего все рассмотренные нейросетевые модели проявили себя при предсказании характеристик для тестовой группы соединений с азокси-группой, состоящей из молекул, содержащих до 28 тяжелых атомов CNO. Средняя абсолютная ошибка доходила до 190 ккал/моль.

Работа выполнена с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова (проект 2312). Нейросетевые и квантово-химические расчёты проводились на суперкомпьютере “Ломоносов-2” в разделе *volta2*, оснащённом процессорами Intel Xeon Gold 6240 (18 ядер, 2.60 ГГц, 1497.6 Гфлопс/с) и графическими ускорителями Nvidia Tesla V100 (900-2G500-0010-000, 1246 МГц, 7 Тфлопс/с).

4. Заключение

Было проведено исследование перспективности использования нейронных сетей для поиска новых высокоэнергетических веществ и предсказания их физико-химических свойств. Для исследования были отобраны четыре наиболее точных и современных нейросети: Allegro (2023), DimeNet++ (2020), PaiNN (2021), TorchMD-NET ET (2022). На основе каждой архитектуры и наборов данных QM9, PC9, OD9 (и его двух подмножества), Alchemy, а также объединения QM9 и Alchemy (AlQM9) были обучены модели для предсказания энергии атомизации из геометрии молекул. Все 28 обученных моделей были применены для предсказания энергии атомизации четырёх тестовых групп веществ, ранжированных по структурной сложности. Из изученных моделей наиболее точной и обобщаемой оказалась архитектура TorchMD-NET ET, обученная на PC9. Её среднее абсолютное отклонение от расчетных данных составило 0.8 ккал/моль. Среди остальных моделей лучше всего себя показывают архитектуры, обученные также на PC9, что может объясняться высоким разнообразием химически значимых структурных фрагментов в этом наборе. Все рассмотренные

нейросетевые модели демонстрируют наименьшую точность при тестировании на группе веществ, содержащих до 28 тяжелых атомов.

Эти исследования проводились в рамках проекта РНФ № 23-71-0005. Основной целью проекта является исследование большого числа высокомолекулярных соединений, содержащих около и более 30 атомов первого и второго периода таблицы Менделеева. Все расчёты проводились с использованием высокоточных квантово-химических методов. Расчёты с использованием нейронных сетей для таких высокомолекулярных соединений не позволяют с достаточной степенью надёжности получать требуемый уровень точности.

Список литературы

[1] Ramakrishnan, R., Dral, P., Rupp, M. et al. Quantum chemistry structures and properties of 134 kilo molecules // Sci Data. 2014. Vol. 1. 140022.
https://doi.org/10.1038/sdata.2014.22

[2] Glavatskikh, M., Leguy, J., Hunault, G. et al. Dataset's chemical diversity limits the generalizability of machine learning predictions // J Cheminform. 2019. Vol. 11. 69.
https://doi.org/10.1186/s13321-019-0391-2

[3] Leguy, J., Glavatskikh, M., Cauchy, T. et al. Scalable estimator of the diversity for de novo molecular generation resulting in a more robust QM dataset (OD9) and a more efficient molecular optimization // J Cheminform. 2021. Vol. 13. 76.
https://doi.org/10.1186/s13321-021-00554-8

[4] Chen, G., Chen, P., Hsieh, C.-Y. et al. Alchemy: A Quantum Chemistry Dataset for Benchmarking AI Models. 2019.
https://doi.org/10.48550/arXiv.1906.09427

[5] Smith, J., Isayev, O., and Roitberg, A. ANI-1, A data set of 20 million calculated off-equilibrium conformations for organic molecules // Sci Data. 2017. Vol. 4. 170193.
https://doi.org/10.1038/sdata.2017.193

[6] Eastman, P., Behara, P.K., Dotson, D.L. et al. SPICE, A Dataset of Drug-like Molecules and Peptides for Training Machine Learning Potentials // Sci Data. 2023. Vol. 10. 11.
https://doi.org/10.1038/s41597-022-01882-6

[7] Nakata, M., Shimazaki, T. PubChemQC Project: A Large-Scale First-Principles Electronic Structure Database for Data-Driven Chemistry // J. Chem. Inf. Model. 2017. Vol. 57. No. 6. P. 1300–1308.
https://doi.org/10.1021/acs.jcim.7b00083

[8] Volokhov, V., Akostelov, I., Parakhin, V., et al. Quantum-Chemical Simulation of High-Energy Azoxy Compounds // In: Sokolinsky, L., Zymbler, M. (eds) Parallel Computational Technologies. PCT 2023. Communications in Computer and Information Science. Springer, Cham. 2023. Vol. 1868. P. 231–243.
https://doi.org/10.1007/978-3-031-38864-4_16

[9] Volokhov, V., Amosova, E., Parakhin, V., et al. Quantum-Chemical Study of Some Tris(pyrrolo)benzenes and Tris(pyrrolo)-1,3,5-triazines // In: Voevodin, V., Sobolev, S., Yakobovskiy, M., Shagaliev, R. (eds) Supercomputing. RuSCDays 2023. Lecture Notes in Computer Science. Springer, Cham. 2023. Vol. 14388. P. 177–189. https://doi.org/10.1007/978-3-031-49432-1_14

[10] Volokhov, V.M., Amosova, E.S., Volokhov, A.V., et al. Quantum-chemical calculations of physicochemical properties of high enthalpy 1,2,3,4- and 1,2,4,5-tetrazines annelated with polynitroderivatives of pyrrole and pyrazole. Comparison of different calculation methods // Comp. Theor. Chem. 2022. Vol. 1209. 113608. https://doi.org/10.1016/j.comptc.2022.113608

[11] Volokhov, V.M., Parakhin, V.V., Amosova, E.S., et al. Quantum-Chemical Calculations of the Enthalpy of Formation for 5/6/5 Tricyclic Tetrazine Derivatives Annelated with Nitrotriazoles // Russ. J. Phys. Chem. B. 2024. Vol. 18. No. 1. P. 28–36. https://doi.org/10.1134/S1990793124010196

[12] Volokhov, V.M., Parakhin, V.V., Amosova, E.S., et al. Quantum-Chemical Study of Gas-Phase 5/6/5 Tricyclic Tetrazine Derivatives // Supercomput. Front. Innov. 2023. Vol. 10. No. 3. P. 61–72. https://doi.org/10.14529/jsfi230306

[13] Musaelian, A., Batzner, S., Johansson, A. et al. Learning local equivariant representations for large-scale atomistic dynamics // Nat. Commun. 2023. Vol. 14. 579. https://doi.org/10.1038/s41467-023-36329-y

[14] Gasteiger, J., Giri, S., Margraf, J.T., et al. Fast and Uncertainty-Aware Directional Message Passing for Non-Equilibrium Molecules // Machine Learning for Molecules Workshop at NeurIPS. 2020. https://doi.org/10.48550/arXiv.2011.14115

[15] Schütt, K., Unke, O., and Gastegger, M. Equivariant message passing for the prediction of tensorial properties and molecular spectra // Proceedings of the 38th International Conference on Machine Learning, in Proceedings of Machine Learning Research. 2021. Vol. 139. P. 9377–9388. https://proceedings.mlr.press/v139/schutt21a.html (accessed: 23.08.2023).

[16] Thölke, P., De Fabritiis, G. TorchMD-NET: Equivariant Transformers for Neural Network based Molecular Potentials // Proceedings of the 10th International Conference on Learning Representations. 2022. https://doi.org/10.48550/arXiv.2202.02541

[17] Ruddigkeit, L., van Deursen, R., Blum, L.C., et al. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17 // J. Chem. Inf. Model. 2012. Vol. 52. No. 11. P. 2864–2875. https://doi.org/10.1021/ci300415d

[18] Волохов, В.М., Акостелов, И.И., Амосова, Е.С., и др. Обобщаемость нейросетевых моделей для предсказания энергии атомизации молекул // Суперкомпьютерные дни в России : Труды международной конференции. 25–26 сентября 2023 г., Москва / Под. ред. Вл. В. Воеводина. – Москва : МАКС Пресс, 2023. – 224 с.

[19] Kim, H., Park, J.Y., and Choi, S. Energy refinement and analysis of structures in the QM9 database via a highly accurate quantum chemical method // Sci Data. 2019. Vol. 6. 109.
https://doi.org/10.1038/s41597-019-0121-7

Исследование свойств признакового пространства для повышения эффективности обучения по нескольким примерам

В.Д. Кучеров, Д.Ю. Буряк

Московский Государственный Университет им. М.В. Ломоносова

Направление обучения по нескольким примерам активно развивается, однако, в случае использования на конечных, маломощных устройствах, данная задача остается крайне сложной. Один из основных подходов для преодоления данной проблемы - построение признакового пространства с использованием глубокого обучения, с последующим использованием классификатора, основанного на прототипах новых классов. Данная статья посвящена исследованию методов оценки эффективности признакового пространства с целью его оптимального использования в контексте обучения по нескольким примерам. В работе проанализированы существующие метрики оценки качества признакового пространства и рассмотрена их применимость в качестве функций потерь.

Ключевые слова: Обучение по нескольким примерам, признаковое пространство, кластеризация, распознавание ключевых слов, эмбеддинг, прототипическая сеть, машинное обучение, нейронные сети

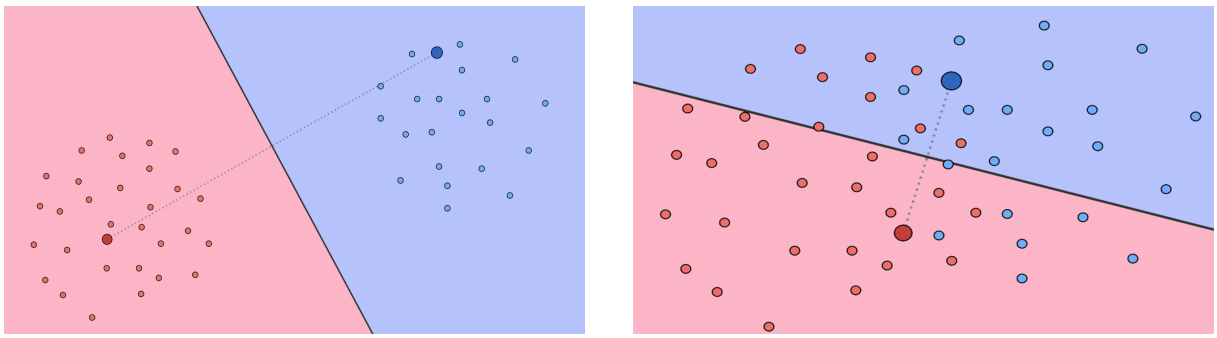
1. Введение

Методы глубокого машинного обучения позволяют решать широкий класс задач, однако одной из основных проблем остается сбор большого объема данных для эффективного обучения моделей. Хотя в открытом доступе находится множество больших, качественных, размеченных наборов данных, они могут не подходить для конкретных приложений, требующих уникальные данные. Среди таких приложений можно выделить следующие:

- Распознавание ключевых слов (Keyword spotting) [2]. Данная задача возникает, например, в системах умного дома или голосовых ассистентах. Добавление новых ключевых слов (голосовых команд) не должно занимать много времени и быть простым, поэтому можно рассчитывать лишь на небольшое количество примеров реплик от пользователя.
- Верификация по биометрии [3]. Распознавание отпечатков пальцев, лица или голоса владельца. Аналогично задаче распознавания ключевых слов возникает проблема малого количества экземпляров для обучения.
- Поиск лекарств, основанный на подборе молекул [4]. Сложность и дороговизна реального синтеза молекул не позволяет решать данную задачу перебором. На основе кандидатов - реально созданных и изученных молекул - требуется подобрать более безопасные и активные аналоги.

Задача обучения по нескольким примерам становится все более актуальной с активным внедрением технологий машинного обучения в повседневную жизнь. Возникает необходимость в разработке эффективных стратегий обучения, способных извлекать максимальную пользу из ограниченного объема данных.

Данная проблема способствовала появлению такого направления как мета обучение (metalearning). Мета обучение основывается на извлечении пользы из предварительных знаний: модель сначала обучается решать вспомогательную задачу, похожую на целевую, на большом датасете (например для задачи классификации изображений в качестве такой вспомогательной задачи можно выбрать ImageNet). Проведение такой подготовки позволяет модели намного быстрее адаптироваться к целевой задаче, что дает возможность использовать намного меньше тренировочных данных. Тем не менее, в экстремальных случаях, когда в наличии имеется лишь единицы примеров новых классов, мета обучение не может дать желаемых результатов.



a *b*

Рис. 1: Примеры непересекающихся (a) и пересекающихся (b) кластеров.

Исследование таких экстремальных случаев породило направление обучения по нескольким примерам (Few Shot Learning — FSL). Одним из предлагаемых решений является построение промежуточного признакового пространства с лучшими качествами кластеризации. Если кластера классов в новом пространстве будут хорошо разделимыми, это позволит легко строить качественные разделяющие гиперплоскости независимо от конкретных тренировочных дынных на этапе обучения по нескольким примерам рис. 1. Идея данного подхода состоит в обучении энкодера, который будет переводить входные данные в промежуточное пространство, в котором данный энкодер обучается на большом наборе данных классическими методами. Несмотря на наличие лишь ограниченного набора данных для обучения целевой задачи, очень вероятно, что существует большой датасет той же структуры (текст, изображения, аудио, ...), но, например, с другими классами. После этого полученный энкодер используется для преобразования входных данных целевой задачи в новое пространство. Качество данного пространства напрямую влияет на точность последующего решения задачи классификации. Возникает проблема оценки качества такого пространства в процессе обучения энкодера.

В рамках данной работы проводится исследование методов построения и анализа признакового пространства для последующего решения задачи обучения по нескольким примерам.

2. Постановка задачи

Выбирается некоторая задача обучения по нескольким примерам. В качестве тренировочных примеров доступен очень ограниченный набор $D_{FSL} = \{L_{FSL}, M_{FSL}\}$, где L_{FSL} — набор меток классов, M_{FSL} — количество примеров на один класс, $|L_{FSL}|$ — количество различных меток классов. Характерное число различных классов — 10, характерное количество примеров на класс — от 1 до 5.

Для данной задачи ищется вспомогательный датасет большого объема, который содержит данные той же природы, что и задача FSL (изображения, аудио, текст, ...), но другие классы и включают намного больше примеров на класс:

$$D_{ENCODER} = \{L_{ENCODER}, M_{ENCODER}\}$$

$$|L_{FSL}| \ll |L_{ENCODER}|, M_{FSL} \ll M_{ENCODER}, L_{ENCODER} \cap L_{FSL} = \emptyset$$

Требуется построить такой энкодер $E : F \to F_E$, который переводит задачу из исходного пространства F в новое пространство признаков F_E. Энкодер обучается исключительно на наборе данных $D_{ENCODER}$. Тогда исходная FSL задача из пространства $F^{FSL} \subset F$ переводится в новое пространство $F_E^{FSL} \subset F_E$ при помощи энкодера E:

$$E(F^{FSL}) : F^{FSL} \to F_E^{FSL}$$

После этого классификатор C работает уже с элементами из пространства F_E^{FSL}:

$$C : F_E^{FSL} \to L_{FSL}$$

Таким образом для $x_i \in F_E^{FSL}$ предсказание строится следующим образом:

$$C(E(x_i)) = l_k, l_k \in L_{FSL}$$

Качество классификации в значительной мере зависит от разделимости классов в пространстве F_E. Данные качества предлагается измерять с помощью метрик кластеризации. Целью данной работы является выявление зависимостей между значениями метрик и точностью классификации последующей задачи FSL в пространстве F_E^{FSL}:

$$Accuracy = \frac{N_T}{N}$$

где N_T — количество правильно классифицированных примеров, N — количество примеров.

3. Обзор существующих решений

Существует множество направлений оптимизации построения признакового пространства: изменение архитектуры энкодера, изменение процесса обучения, выбор различных функций потерь. Рассмотрим методы, основанные на базовых знаниях [1].

3.1. Многозадачное обучение

Для целевой задачи обучение по нескольким примерам выбираются одна или несколько схожих задач с большим набором тренировочных данных. Одновременно несколько одинаковых по архитектуре моделей обучаются на вспомогательных и на целевых задачах. Существует несколько различных способов передавать информацию между этими моделями. Например, выделяют несколько первых слоев, которые будут иметь одинаковые параметры для всех моделей. Эти первые несколько слоев могут обновляться только на вспомогательных задачах, а в целевой задаче они заморожены и в процессе обучения обновляются только последующие слои.

3.2. Обучение с внешней памятью

Малый размер целевого тренировочного набора данных позволяет хранить его полностью в памяти, а во время классификации обращаться к нему. Так, строится хранилище эмбеддингов тренировочного набора данных. Для построения такого хранилища используется предобученный энкодер. Во время классификации для тестового примера также строится эмбеддинг, который сравнивается со всеми эмбеддингами из внешнего хранилища. В качестве функции схожести зачастую используют косинусное сходство или предобученные нейронные сети.

3.3. Эмбеддинги

Идея данного подхода состоит в переводе задачи из исходного пространство в меньшее по размерности пространство признаков (эмбеддингов). При этом, в пространстве эмбеддингов примеры одно класса расположены близко друг к другу, а примеры разных классов далеко друг от друга. Это позволяет решать задачу классификации в новом пространстве, например, с помощью прототипических сетей [9]. Рассмотрим как достигается разделимость классов в пространстве эмбеддингов с помощью специальных функций потерь contrstive-loss.

3.4. Функции потерь

3.4.1. Contrastive loss

Contrastive loss [5] для каждой пары объектов (x_i, x_j) минимизирует расстояние эмбеддингов, если они из одного класса, и максимизирует, если они из разных классов:

$$\mathcal{L}_{cont}(x_i, x_j, \theta) = 1[y_i = y_j]\|f_\theta(x_i) - f_\theta(x_j)\|_2^2 + 1[y_i \neq y_j]\max(0, \epsilon - \|f_\theta(x_i) - f_\theta(x_j)\|_2)^2$$

Где ϵ это гиперпараметр, определяющий минимальное расстояние между объектами разных классов.

3.4.2. Triplet loss

Triplet Loss [6] развивает идею Contrastive loss, но расчет идет сразу по трем объектам: базовый пример x, положительный пример (того же класса, что и базовый) x^+ и один негативный пример (другого класса) x^-. Triplet loss одновременно увеличивает

расстояние между базовым и негативным объектами и уменьшает расстояние между базовым и положительным.

$$\mathcal{L}_{triplet}(x, x^+, x^-) = \sum_{x \in \mathcal{X}} \max(0, \|f(x) - f(x^+)\|_2^2 - \|f(x) - f(x^-)\|_2^2 + \epsilon)$$

3.4.3. Lifted Structured loss

Lifted Structured loss [7] работает по тому же принципу, что и Triplet loss, однако он использует все пары объектов внутри одного тренировочного батча:

Пусть $D_{ij} = |f(x_i) - f(x_j)|_2$, $\mathcal{P}$ — набор положительных примеров, $\mathcal{N}$ — набор отрицательных примеров, тогда:

$$\mathcal{L}_{struct} = \frac{1}{2|\mathcal{P}|} \sum_{(i,j) \in \mathcal{P}} \max(0, \mathcal{L}_{struct}^{(ij)})^2$$

$$\mathcal{L}_{struct}^{(ij)} = D_{ij} + \max\Big(\max_{(i,k) \in \mathcal{N}} \epsilon - D_{ik}, \max_{(j,l) \in \mathcal{N}} \epsilon - D_{jl}\Big)$$

Правая часть в $\mathcal{L}_{struct}^{(ij)}$ призвана искать сложные негативные примеры (наиближайшие отрицательные примеры к базовому). Однако это функция ступенчатая, что может вызвать проблемы во время обучения, поэтому используют её сглаженную версию:

$$\mathcal{L}_{struct}^{(ij)} = D_{ij} + \log\left(\sum_{(i,k) \in \mathcal{N}} \exp(\epsilon - D_{ik}) + \sum_{(j,l) \in \mathcal{N}} \exp(\epsilon - D_{jl})\right)$$

3.4.4. N-pair loss

N-pair loss [8] — это обобщение triplet loss, использующее не один, а несколько $(N-1)$ негативных примеров. Таким образом, выбирается базовый пример, один положительный и $(N-1)$ негативных примеров.

$$\mathcal{L}_{\text{N-pair}}(x, x^+, \{x_i^-\}_{i=1}^{N-1}) = \log\left(1 + \sum_{i=1}^{N-1} \exp(f(x)^\top f(x_i^-) - f(x)^\top f(x^+))\right) =$$

$$= -\log \frac{\exp(f(x)^\top f(x^+))}{\exp(f(x)^\top f(x^+)) + \sum_{i=1}^{N-1} \exp(f(x)^\top f(x_i^-))}$$

3.5. Метрики

Для определения качества кластеризации данных вводятся следующие метрики.

- **Отношение внутриклассовой к межклассовой дисперсии** (intra-class to inter-class variance ratio):

$$R_{FC} = \frac{\sigma_{within}^2}{\sigma_{between}^2} = \frac{\frac{\sum_{i,j} \|\phi_{i,j} - \mu_i\|_2^2}{N}}{\frac{\sum_i \|\mu_i - \mu\|_2^2}{C}} = \frac{C}{N} \frac{\sum_{i,j} \|\phi_{i,j} - \mu_i\|_2^2}{\sum_i \|\mu_i - \mu\|_2^2}$$

где C — количество классов, N — количество экземпляров класса, $\phi_{i,j}$ — эмбеддинг j-ого объекта в i-ом классе, μ_i — среднее эмбеддингов i-ого класса, μ - среднее эмбеддингов всех классов. Чем меньше значение полученного выражения, тем лучше кластеризация.

- **Ограничение разброса гиперплоскостей**: Пусть x_1, x_2 - объекта одного класса, y_1, y_2 - объекты другого класса, $f_\theta(x)$ - эмбеддинг объекта x, тогда выражение $f_\theta(x_1) - f_\theta(y_1)$ показывает направление разделяющей гиперплоскости с наибольшим зазором. Следующее выражение показывает насколько разные гиперплоскости можно построить, выбирая различные пары объектов из двух классов:

$$R_{HV}(f_\theta(x_1), f_\theta(x_2), f_\theta(y_1), f_\theta(y_2)) = \frac{\|(f_\theta(x_1) - f_\theta(y_1)) - (f_\theta(x_2) - f_\theta(y_2))\|_2}{\|(f_\theta(x_1) - f_\theta(y_1)\|_2 + \|f_\theta(x_2) - f_\theta(y_2)\|_2}$$

  Как и в первом случае чем меньше значение R_{HV}, тем лучше кластеризация.
- **Silhouete score.** Silhouette score определяется как средний Silhouette Coefficient для всех точек. Silhouette Coefficient:

$$Silhouette\_score = \frac{1}{n}\sum_{i=1}^{n}\frac{b_i - a_i}{\max(a_i, b_i)}$$

  где n — количество точек, b_i — среднее расстояние от i-ой точки до точек ближайшего класса, a_i — среднее расстояние от i-ой точки до точек ее класса. Silhouette score определен на промежутке $[-1, 1]$. 1 означает наилучшую кластеризацию, значения около 0 означает пересекающиеся кластера.
- **Davies-Bouldin Index.** Индекс показывает "схожесть" кластеров, под "схожестью" понимается сравнение размеров кластеров с расстоянием между кластерами.

$$DB = \frac{1}{n}\sum_{i=1}^{n}\sum_{j\neq i}^{n}\max\left(\frac{s_i + s_j}{d_{ij}}\right)$$

  где n — количество точек, s_i — среднее расстояние от точек i-го класса до центроида класса, d_{ij} — расстояние между центроидами классов i и j. Минимальное значение — 0, означает наилучшую кластеризацию. Чем меньше значение DB, тем лучше кластеризация.

3.6. Прототипическая сеть

Рассмотрим набор из N размеченных объектов $S = (x_1, y_1), \ldots, (x_N, y_N)$, где $x_i \in R^D$ — D-мерный вектор признаков объекта (полученный, например, с помощью эмбеддинг функции (энкодера) $f_\phi : R^M \to R^D$), $y_i \in 1, \ldots, K$ — соответствующая метка класса. S_k — поднабор набора S, где $y_i = k$.

Прототипическая сеть[9] для каждого класса k из $1, \ldots, K$ вычисляет его прототип c_k — среднее признаковых векторов:

$$c_k = \frac{1}{|S_k|}\sum_{(x_i, y_i)\in S_k} x_i$$

Для классификации требуется функция расстояния $d : R^D \times R^D \to [0, +\infty)$, тогда прототипическая сеть для запроса x выдает распределение вероятностей по классам:

$$p(y = k|x) = \frac{\exp(-d(x, c_k))}{\sum_i \exp(-d(x, c_i))}$$

4. Принятые решения

- Для проведения экспериментов выбрана задача распознавания ключевых слов (KWS). Данная задача актуальна в системах умного дома и голосовых помощниках, например, для распознавания фразы активации.
- В качестве энкодера решено использовать сверточную нейронную сеть DS-CNN[10]. Данный тип сетей считается стандартным для выбранной задачи и особенно часто применяется, если конечное устройство маломощное. Характеристики энкодера: 6 сверточных слоев, размерность выходного вектора признаков - 64 (размерность итогового признакового пространства), общее число обучаемых параметров — 32000.
- На вход энкодера подается мел-спектрограмма. Данный вид спектрограмм ориентирован на слуховые возможности человека: чувствительность уменьшается логарифмически с увеличением частоты. Мел-спектрограмма сохраняет больше информации в низких частотах и меньше в высоких.
- Энкодер обучается с использованием следующих функций потерь: Cross Entropy Loss, Triplet Loss, Lifted Structured Loss, N-pair Loss. Данные функции потерь улучшают качества кластеризации итогового пространства. Как показали предварительные эксперименты изменение гиперпараметра ϵ (margin) незначительно влияет на итоговое качество получаемого пространства: с увеличением параметра растет расстояние между кластерами, но растут и сами кластера. Однако само наличие данного параметра с положительным значением существенно улучшает качество итогового пространства. Во всех функциях потерь значением данного гиперпараметра выбрана 1.
- Предлагается также использовать в качестве функции потерь метрику Silhouette Score с небольшими модификациями, по аналогии с функциями потерь contrastive-loss:

  $$Silhouette\_loss(\{x_i\}_{i=1}^N) = \frac{1}{N}\sum_{i=1}^{N}\max\left(0, \frac{b_i - a_i}{\max(a_i, b_i)} + \epsilon\right)$$

  где N — количество точек, b_i — среднее расстояние от i-ой точки до точек ближайшего класса, a_i — среднее расстояние от i-ой точки до точек ее класса, ϵ — гиперпараметры.
- В качестве метрик оценки качества признакового пространства выбраны: FC, HV, Silhouette Score, Davies-Bouldin Index. Данные метрики отражают качества кластеризации признаковых пространств, которые являются наиболее важными для выбранного классификатора — прототипической сети.
- Выбор прототипической сети в качестве классификатора обусловлен ее низкими требованиями к вычислительным ресурсам, что важно при использовании в конечных маломощных устройствах, и высокой эффективностью в задаче обучения по нескольким примерам.

5. Постановка эксперимента

Для проведения экспериментов выбрана задача распознавания ключевых слов (keyword spotting) на наборах данных Google Speech commands V2[11] и MLCommons Multilingual Spoken Words Dataset (дальше используются аббревиатуры SC и MSWC соответственно). Датасеты представляют собой размеченный набор аудиофайлов, длиной не более одной секунды. Каждый файл соответствует единственному слову-классу.

Для обучения энкодера выделено четыре выборки из данных датасетов:

- SC_25CS_1250SPC - подвыборка SC из 25 слов-классов по 1250 примеров на класс. Слова английские.
- MSWC_EN_100CS_500SPC - подвыборка MSWC из 100 слов-классов по 500 примеров на класс. Слова английские.
- MSWC_EN_500CS_100SPC - подвыборка MSWC из 500 слов-классов по 100 примеров на класс. Слова английские.
- MSWC_RU_50CS_500SPC - подвыборка MSWC из 50 слов-классов по 500 примеров на класс. Слова русские.

Для каждой из перечисленных выборок также есть, соответствующая ей, валидационная выборка, на которой вычисляются метрики кластеризации. Энкодеры обучаются 200 эпох. Всего 5 функций потерь x 4 датасета = 20 обученных энкодеров.

На протяжении обучения на каждой эпохе вычисляются выбранные метрики кластеризации. Из 200 эпох по каждой метрике выбирается наилучшая, таким образом, для каждого энкодера выбирается до 5 различных эпох: по числу метрик + loss.

После окончания обучения веса энкодера заморожены и больше не меняются. Для каждого энкодера решается задача обучения по нескольким примерам для следующих списков слов:

- Speech Commands 10 слов-классов: ['on', 'off', 'up', 'down', 'right', 'left', 'yes', 'no', 'go', 'stop'].
- MSWC EN 10-слов классов: ['open', 'close', 'wait', 'save', 'resume', 'exit', 'accept', 'search', 'start', 'listen'].
- MSWC RU 10-слов классов: ['вперед', 'назад', 'принять', 'пожалуйста', 'спасибо', 'хорошо', 'всегда', 'внимание', 'продолжать', 'больше'].

Для каждого списка классификатор строится в двух вариантах: 1-shot и 5-shot (1 пример на класс и 5 примеров на класс соответственно в тренировочном наборе). Поскольку качество классификации зависит от конкретных примеров в тренировочном наборе, то точность усредняется по 10 попыткам: всего 3 набора классов x 2 варианта количества тренировочных примеров x 10 конкретных наборов примеров = 60 FSL-экспериментов для каждого построенного энкодера. После обучения классификаторов оценивается их точность на тестовых выборках, соответствующих классам FSL. Тестовые выборки сбалансированы по количеству примеров на класс и содержат соответственно:

- 396 примеров на класс для Speech Commands
- 50 примеров на класс для MSWC EN
- 28 примеров на класс для MSWC RU

Такой необычный разброс по количеству примеров на класс для различных экспериментов FSL обусловлен максимальным доступным количеством примеров на класс из соответствующих исходных тестовых выборок Speech Commands и MSWC для выбранных слов-классов.

Таким образом, будут проведены эксперименты наиболее приближенные к реальным: тренировочный датасет энкодера отличается от тренировочного датачета классификатора (MSWC и Speech Commands). Особый интерес представляют кросс-языковые эксперименты: энкодер обучен на одном языке, а классификатор на другом. Такие эксперименты позволяют оценить устойчивость построенных признаковых пространств к различным сценариям использования.

6. Результаты

Эксперименты показали, что наиболее выраженная корреляция наблюдается между точностью FSL и метриками Silhouette Score и Davies-Bouldin Index, про метрики FC и HV такое сказать нельзя табл. 1. Пример высокой корреляции изображен на рис. 2, а низкой — на рис. 3. При этом корреляция сохраняется как в обычных, так и в кросс-языковых экспериментах табл. 1.

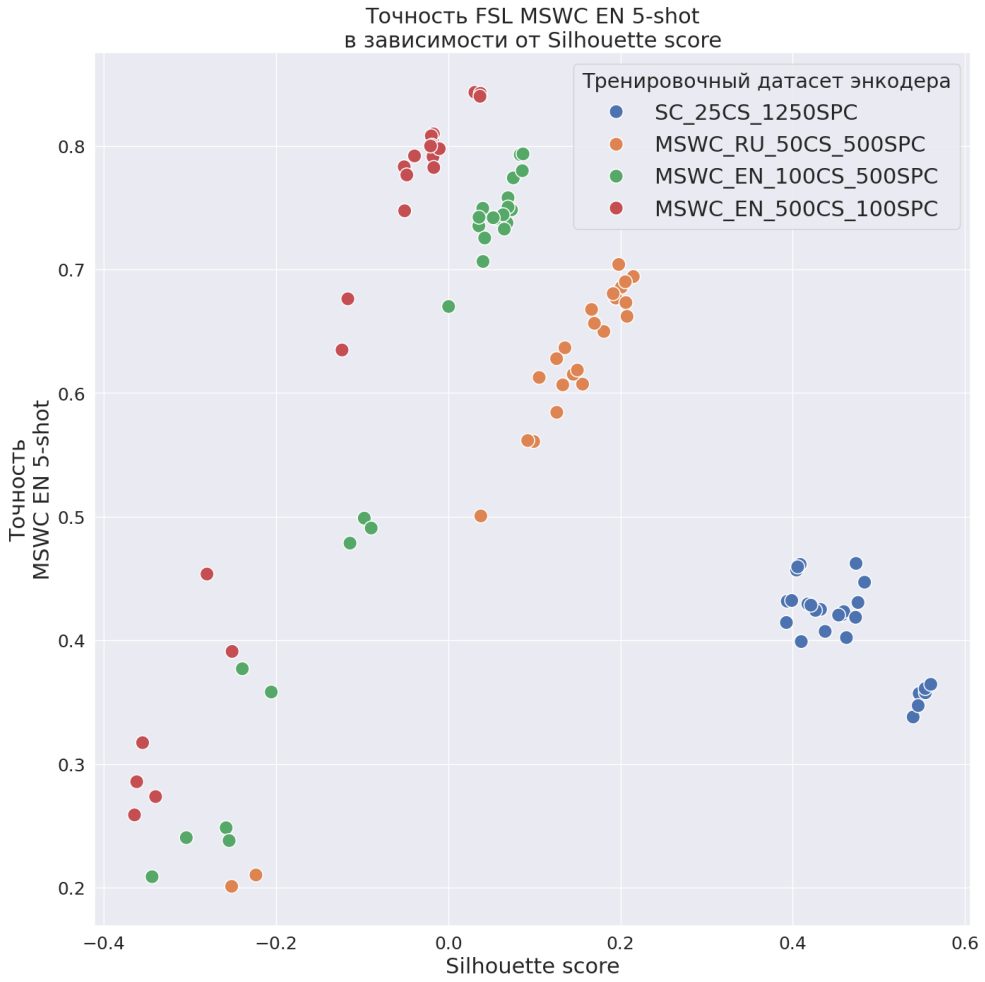


Рис. 2: Высокая корреляция Пирсона между метрикой Silhouette Score и точностью FSL эксперимента MSWC EN 5-shot.

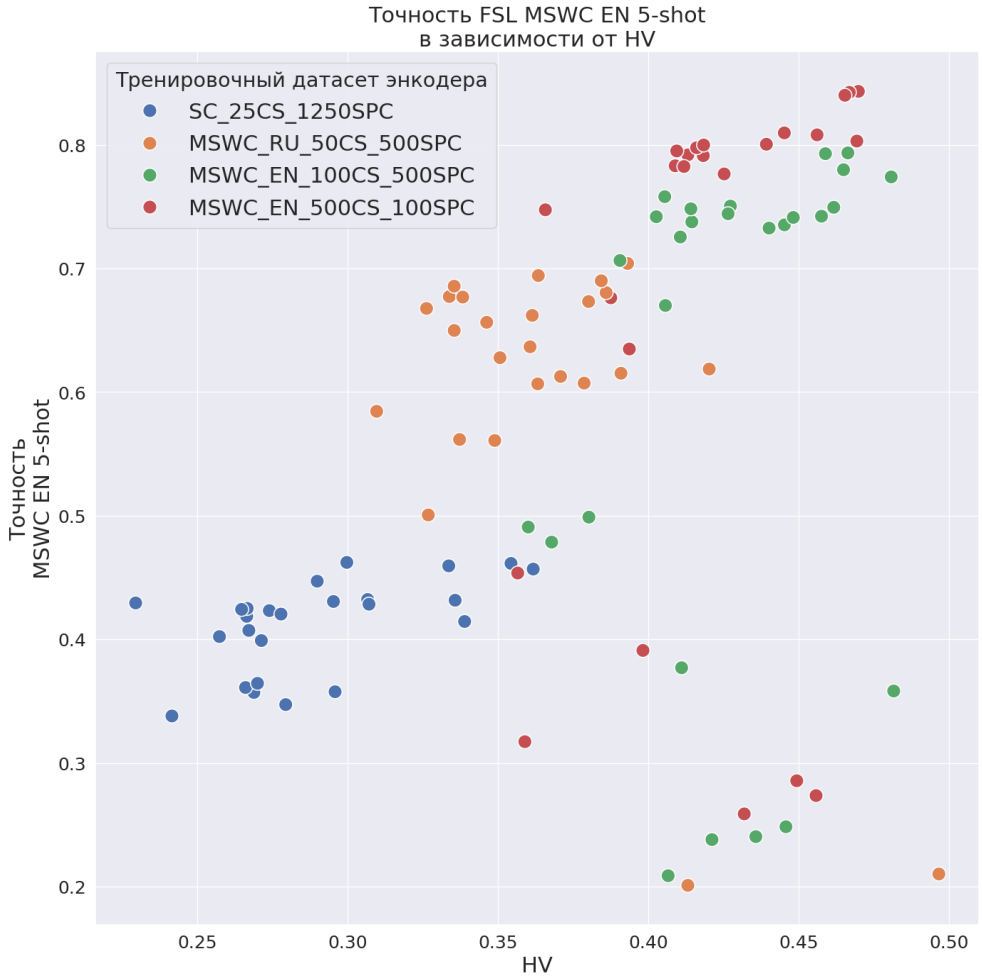


Рис. 3: Низкая корреляция Пирсона между метрикой HV и точностью FSL эксперимента MSWC EN 5-shot.

Таблица 1: Среднее значение корреляции Пирсона между значением метрики и точностью классификатора. Эксперименты разделены по типам по столбцам: 'все' — усреднение по всем экспериментам, 'кросс-языковые' — энкодер и классификатор обучены на разных языках, 'ru-en' — энкодер обучен на русском, классификатор на английском и так далее

Метрика	все	кросс-языковые	один язык	ru-en	en-ru	ru-ru	en-en
FC	0.485	0.449	0.510	0.183	**0.627**	0.095	0.579
HV	0.048	-0.054	0.121	-0.551	0.277	-0.625	0.245
Silhouette Score	**0.655**	**0.647**	**0.661**	**0.976**	0.428	0.967	**0.610**
Davies-Bouldin Index	-0.608	-0.610	-0.607	-0.948	-0.384	**-0.973**	-0.545

При этом абсолютное значение метрик не играет особой роли, поскольку измерения метрик для разных энкодеров проводились на разных выборках. Таким образом, в среднем лучше выбирать эпоху по метрике Silhouette Score табл. 2. Также Silhouette Score обладает еще одним приятным свойством — дифференцируемостью, что позволяет использовать ее в качестве функции потерь. Silhouette Margin Loss показала свою эффективность по сравнению с другими рассматриваемыми в этой работе функциями потерь, превосходя их в среднем более чем на 3% табл. 3

Таблица 2: Среднее значение точности классификатора (в процентах) в зависимости от выбранного метрикой энкодера. Эксперименты разделены по типам по столбцам: 'все' — усреднение по всем экспериментам, 'кросс-языковые' — энкодер и классификатор обучены на разных языках, 'ru-en' — энкодер обучен на русском, классификатор на английском и так далее

Метрика	все	кросс-языковые	один язык	ru-en	en-ru	ru-ru	en-en
FC	35,8	34,0	37,1	35,3	33,0	49,1	35,1
HV	52,1	51,0	52,9	47,5	53,3	65,5	50,8
Silhouette Score	**64,0**	**61,0**	**66,1**	**52,6**	**66,6**	**69,4**	**65,5**
Davies-Bouldin Index	63,5	60,5	65,7	52,0	66,1	68,8	65,2

Таблица 3: Среднее значение точности классификатора (в процентах) в зависимости от функции потерь, с помощью которой обучался энкодер. Эксперименты разделены по типам по столбцам: 'все' — усреднение по всем экспериментам, 'кросс-языковые' — энкодер и классификатор обучены на разных языках, 'ru-en' — энкодер обучен на русском, классификатор на английском и так далее

Функция потерь	все	кросс-языковые	один язык	ru-en	en-ru	ru-ru	en-en
Cross Entropy	62,8	58,4	67,2	49,6	67,2	68,6	65,8
Lifted Structured	64,1	60,2	67,9	52,5	68,0	70,1	65,7
N-pair	64,2	60,8	67,5	55,3	66,4	70,5	64,6
Silhoeutte Margin	**67,4**	**63,4**	**71,3**	**56,6**	**70,1**	**73,2**	**69,4**
Triplet	63,1	59,1	67,1	52,2	66,0	69,2	64,9

7. Выводы

В результате работы были реализованы различные методы построения и оценки признаковых пространств. В ходе экспериментального сравнения данных методов было выявлено, что среди метрик оценки пространств наилучшее значение корреляции с точностью последующего решения FSL являются Silhouette Score и Davies-Bouldin Index, коэффициент корреляции Пирсона для которых соответственно составляет 0.655 и -0.608. Дифференцируемость метрики Silhouette Score позволила составить на ее основе функцию потерь Silhouette Margin Loss, которая превзошла функции потерь contrastive-loss в среднем более чем на 3%. Обобщение полученных результатов на различные задачи обучения по нескольким примерам является предметом для будущего исследования.

Список литературы

[1] Yaqing Wang and Quanming Yao and James Kwok and Lionel M. Ni Generalizing from a Few Examples: A Survey on Few-Shot Learning, 2020. `https://arxiv.org/abs/1904.05046`

[2] Dongjune Lee and Minchan Kim and Sung Hwan Mun and Min Hyun Han and Nam Soo Kim. Fully Unsupervised Training of Few-shot Keyword Spotting, 2022. `https://arxiv.org/abs/2210.02732`

[3] Saad Bin Ahmed and Umaid M. Zaffar and Marium Aslam and Muhammad Imran Malik. Few-Shot Learning for Biometric Verification, 2023
https://arxiv.org/abs/2211.06761

[4] Han Altae-Tran and Bharath Ramsundar and Aneesh S. Pappu and Vijay Pande. Low Data Drug Discovery with One-shot Learning, 2016.
https://arxiv.org/abs/1611.03199

[5] Chopra, S. and Hadsell, R. and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification, 2005. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)

[6] Florian Schroff and Dmitry Kalenichenko and James Philbin. FaceNet: A unified embedding for face recognition and clustering, 2015.
https://doi.org/10.1109\%2Fcvpr.2015.7298682

[7] Hyun Oh Song and Yu Xiang and Stefanie Jegelka and Silvio Savarese. Deep Metric Learning via Lifted Structured Feature Embedding, 2015.
https://arxiv.org/abs/1511.06452

[8] Sohn, Kihyuk. Improved Deep Metric Learning with Multi-class N-pair Loss Objective, 2016.
https://proceedings.neurips.cc/paper_files/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf

[9] Jake Snell and Kevin Swersky and Richard S. Zemel, Prototypical Networks for Few-shot Learning, 2017.
https://arxiv.org/abs/1703.05175

[10] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions, 2017.
https://arxiv.org/abs/1610.02357

[11] Pete Warden, Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition, 2018.
https://arxiv.org/abs/1804.03209

М.В.Келдыш — идеолог космических исследований: информационно-математический аспект. К 300-летию РАН и 270-летию МГУ им. М.В. Ломоносова

Т.А. Сушкевич

Институт прикладной математики им. М.В. Келдыша РАН

В 2023 году в условиях тектонических геополитических сдвигов глобального порядка в мире и в России и новых вызовов вплоть до угроз третьей мировой "ядерной" войны вышел Указ Президента от 31.03.2023 № 229 "Об утверждении Концепции внешней политики". Этот Указ фактически является актуализацией приоритетных направлений, целей, задач государства для обеспечения суверенитета и безопасности страны в новых условиях. В переломные моменты рождаются новые большие вызовы для отечественной науки и развития технологического уклада. Ныне это постиндустриальная концепция: "космическая" и "цифровая" цивилизации, истоки которых связаны с именем М.В. Келдыша – Главного Теоретика космонавтики, Главного Математика, Президента АН СССР, который отвечал за науку, космос, математику, ЭВМ, расчеты.

Ключевые слова: Мстислав Всеволодович Келдыш, Эпоха Келдыша, математика, ЭВМ, расчеты, ракетно-ядерный щит, космические исследования, информационно-математический аспект, Академия наук, МГУ имени М.В.Ломоносова

1. Введение

В год 300-летнего юбилея Академии наук (АН) накануне 270-летия Московского государственного университета имени М.В. Ломоносова (МГУ) публикация посвящается Мстиславу Всеволодовичу Келдышу (10.02.1911-24.06.1978) – уникальной личности в истории мировой цивилизации и государства российского, светило советской науки, который вошел в список "Великие умы России": первый в этом списке М.В. Ломоносов – это 18 век, а М.В. Келдыш в списке второй – это Ломоносов 20-го века [1]. М.В. Келдыш превзошел своих УЧИТЕЛЕЙ – единственный математик Трижды Герой Социалистического Труда СССР [2]! С именем М.В. Келдыша связывают "золотой век" отечественной науки – "Эпоха Келдыша" [3] – непревзойденный организатор и руководитель науки [3, 4]. В.А. Садовничий на Ученом совете МГУ по случаю 110-летия М.В. Келдыша сказал: таких сейчас нет! Уникальный опыт М.В. Келдыша не исчезает бесследно! Успехи и достижения М.В. Келдыша – это заслуги наполовину Академии наук и МГУ.

На Конференции в МГУ, посвященной 110-летию М.В.Келдыша [5], своим видением роли и значения М.В. Келдыша для отечественной науки поделился президент РАН академик А.М. Сергеев. Сергеев подчеркнул, что работа М.В.Келдыша всегда была на острие главных научных проблем, которые решала страна в то время – в области авиации, атомных исследований, космонавтики. Это было такое время, когда в докомпьютерную эпоху, почти только с одной логарифмической линейкой ученые стояли во главе ключевых государственных проектов. Но именно при Келдыше во главе АН СССР были заложены основы компьютерной науки. А.М. Сергеев подчеркнул, что годы президентства М.В. Келдыша – это золотые, пиковые годы Академии, время безусловного признания авторитета советских ученых со стороны государства и общества. Академик А.М. Сергеев напомнил о работе М.В. Келдыша в межведомственном совете по космическим исследованиям, основанном в 1959 году на базе Академии наук. По его оценке, он всегда был советом № 1 по своему значению. Работы М.В. Келдыша по космическим исследованиям, осуществленные при нем первые международные проекты в этой области, российско-американское партнерство в космосе – это наказ выдающегося ученого и для современности, «так как мир в космосе означает и мир на земле». А.М. Сергеев напомнил, что М.В. Келдыш – это один из тех ученых, жизнь и наследие которого по праву делят поровну Академия наук и Московский университет.

Нет сомнений, почему в "Эпоху Келдыша" отечественная наука потрясала результативностью научной мысли весь мир. М.В. Келдыш мыслил стратегически и масштабно, не боялся работать с равными себе учеными умами – его окружали серьезные личности – талантливые, увлеченные люди, единомышленники и оппоненты разных возрастов и поколений. Он же всегда был их признанным лидером, компетентным и беспристрастным арбитром. "Эпоха Келдыша" и открытие "космической эры" – наглядная иллюстрация, когда "отечественная" наука стала "мировой": научные достижения русских ученых, полученные исключительно в пределах своего государства, не только поднялись до уровня "мировой" науки, но и превзошли достижения всей "мировой" науки за всю историю земной цивилизации!

М.В. Келдыш фактически реализовал предсказание "скромного преподавателя математики и основоположника современной космонавтики" Константина Эдуардовича Циолковского (17.09.1857-19.09.1935) [6]: Математика – могучее орудие ума. И предсказывал, что именно математики своими расчетами помогут вывести корабли в космос. К.Э. Циолковский умер в Калуге 19 сентября 1935 года. За шесть дней до своей смерти он писал: «До революции моя мечта не могла осуществиться. Лишь Октябрь принёс признание трудам самоучки: лишь советская власть и партия Ленина–Сталина оказали мне действенную помощь. Я почувствовал любовь народных масс, и это давало мне силы продолжать работу, уже будучи больным... Все свои труды по авиации, ракетоплаванию и межпланетным сообщениям передаю партии большевиков и советской власти – подлинным руководителям прогресса человеческой культуры. Уверен, что они успешно закончат мои труды».

На Главном здании МГУ у входа, что напротив памятника М.В. Ломоносова, всего одна мемориальная доска и она посвящена М.В. Келдышу. А на Высотном здании, что у метро "Красные ворота", где М.В. Келдыш жил с 1952 до 1975 гг., и во дворе сохранились помещения и мемориальные доски первой ГИРД, но так и нет мемориальной доски М.В. Келдыша. Ни в одном школьном учебнике нет имени этого русского гения, который в середине 20-го века открывал "цифровую" и "косми-

ческую" цивилизации – реальность 21-го века. Выросли поколения, которые не знают этого Ломоносова 20-го века, а другие поколения уже не помнят. В 2021 году единственную конференцию, посвященную 110-летию со дня рождения М.В. Келдыша, организовал ректор В.А. Садовничий в МГУ [5]. Однако, долг автора, единственной ученой – ученицы великих М.В. Келдыша и А.Н. Тихонова, пока жива сохранять их наследие и память, напоминая в своих публикациях и докладах о великой советской цивилизации и её достижениях и Героях. В любом случае важно понимать, что "абсолютно новых" решений в научно-техническом прогрессе и инженерно-конструкторских изобретениях практически не бывает. Мы стоим "на плечах гигантов", порой не замечая этого... и наша обязанность с благодарностью помнить этих "гигантов"!

Автора интересует роль М.В. Келдыша, который стоял у истоков и отвечал за математику, ЭВМ, расчеты с 1946 и с 1951 гг., в условиях "холодной войны" и "гонки – конкуренции между СССР и США" по ключевым областям научно-технической революции в середине 20 века: развитие авиации, покорение атома, покорение космоса, разработка ЭВМ, старт "IT-технологий".

2. Исторические задачи России: суверенитет, безопасность, отечественная наука

Выступая 17 декабря 2023 года на XXI съезде партии "Единая Россия", глава государства Владимир Путин заявил, что перед Россией стоят исторические задачи, масштаб которых требует объединения усилий: "Вместе со всем народом России отстаивать суверенитет, свободу, безопасность России, всё то, что дорого нам, – нашу историю, культуру, ценности и традиции. Укреплять экономику, социальную сферу, научно-технологический потенциал в интересах людей, благополучия миллионов российских семей" [7]. Послание Федеральному Собранию 29 февраля 2024 года [8] – "это прежде всего взгляд в будущее. ... речь идёт не только о наших ближайших планах, но и о стратегических задачах, о тех вопросах, решение которых считаю принципиально важным для уверенного, долгосрочного развития страны".

"Дестабилизирующие наращивание и модернизация наступательных военных потенциалов, разрушение системы договоров в сфере контроля над вооружениями подрывают стратегическую стабильность. Использование военной силы в нарушение международного права, **освоение космического и информационного пространства в качестве новых сфер военных действий**, стирание грани между военными и невоенными средствами межгосударственного противоборства, обострение в ряде регионов застарелых вооруженных конфликтов увеличивают угрозу всеобщей безопасности, усиливают риски столкновений между крупными государствами, в том числе с участием ядерных держав, повышают вероятность эскалации таких конфликтов и их перерастания в локальную, региональную или глобальную войну" [16].

В 2023 году в условиях тектонических геополитических сдвигов глобального порядка в мире и в России и новых геополитических вызовов вплоть до угроз третьей мировой "ядерной" войны вышел Указ Президента Российской Федерации от 31.03.2023 № 229 "Об утверждении Концепции внешней политики Российской Феде-

рации" [11]. Указ фактически является актуализацией приоритетных направлений, целей и задач внешнеполитической деятельности в новых условиях обеспечения суверенитета страны и вызовом для отечественной науки и развития постиндустриального технологического уклада – "космической цивилизации", "цифровой экономики", "цифровой реальности", "цифровой цивилизации", "информационное общество" и т.п. [17].

На государственном уровне приняты ключевые директивные документы:

- Указ Президента Российской Федерации от 15.03.2021 № 143 "О мерах по повышению эффективности государственной научно-технической политики" [9];
- Указ Президента Российской Федерации от 25.04.2022 № 231 "Об объявлении в Российской Федерации Десятилетия науки и технологий (2022-2031)" [10];
- Распоряжение Правительства от 20.05.2023 № 1315-р "Концепция технологического развития до 2030 года" [12];
- Указ Президента Российской Федерации от 26.10.2023 № 812 "Об утверждении Климатической доктрины Российской Федерации" [14];
- Указ Президента Российской Федерации от 10.10.2019 г. № 490 " О развитии искусственного интеллекта в Российской Федерации" (в редакции Указа Президента Российской Федерации от 15.02.2024 № 124) [15];
- Указ Президента Российской Федерации от 28.02.2024 № 145 "О Стратегии научно-технологического развития Российской Федерации" [16].

Принимаются меры по наращиванию космической группировки (сейчас РФ занимает пятое место, а во времена СССР была лидером) [13].

2022-2024 гг. – начало эпохальной исторической деформации и переустройства всей международной системы мироустройства и геополитики, сложившихся после окончания Второй Мировой войны и создания Организации объединенных наций (ООН). Это новые вызовы для науки, но нечто подобное в начале 20-го века говорил выдающийся ученый и блестящий популяризатор науки академик Владимир Иванович Вернадский (12.03.1863-06.01.1945). В интересном докладе [18] на тему "Мысли о современном значении истории знаний", прочитанном на первом заседании Комиссии по истории знаний 14.XI.26, он высказал много умных и полезных мыслей: "Переживаемое нами время является удивительным временем в истории человечества. Сходного с ним приходится искать в далеких столетиях прошлого. Это время интенсивной перестройки нашего научного миросозерцания, глубокого изменения картины мира. Представление об окружающем, с которым человечество Запада вступило в XX век, несмотря на все успехи естествознания, математики, исторических наук, техники, которыми так ярко может характеризоваться XIX столетие, по существу являлось результатом постепенного и неуклонного развития принципов и построений новой эпохи, подготовлявшейся в XVI и ясно вылившейся в XVII столетии, когда окончательно сказались в научной работе еще более ранние достижения Коперника и путь, проложенный Колумбом, новая математика, новая философия, коренная ломка идей о строении и положении в мире человека. . . ". Важно помнить о преемственности в науке: "История науки является в такие моменты орудием достижения нового", говоря о переломных моментах или острых проблемах в истории государств.

Жизнь и масштабная научная, общественная, организационная, государственная деятельность М.В. Келдыша являются образцом и историческим примером служения и защиты Родины в сложнейшие критические периоды во время Великой Отечественной Войны, когда спасал авиацию от катастроф с помощью математики, и

после войны, когда вместе с И.В.Курчатовым и С.П.Королевым (“Три К”) возглавил “Атомный” и “Космический” проекты и сыграл ключевую роль в создании “Ракетно-ядерного щита” как Главный Математик страны. При активном участии М.В. Келдыша “математика стала производительной силой”!

3. Московский университет и математика у истоков покорения космоса

70 лет назад, 14 февраля 1954 года, в кабинете (н. Мемориальный Кабинет-музей академика М.В. Келдыша при Президиуме РАН в историческом здании Института прикладной математики им. М.В. Келдыша РАН) директора ОПМ МИАН СССР (с 1953 до 1966 года секретный п/я 2287) М.В. Келдыша состоялось ПЕРВОЕ совещание [19, 20], на котором впервые “обсуждались примерные сроки и технические вопросы запуска первого искусственного спутника Земли (ИСЗ), научные проблемы, которые предполагалось решить с помощью аппаратуры на искусственных спутниках. Кроме состава научных экспериментов обсуждался также вопрос об ориентации спутника: запускать ли неориентированный спутник или следует разрабатывать достаточно долго функционирующую систему ориентации” [19]. На это совещание были приглашены С.П. Королев, П.Л. Капица, Л.И. Седов, С.Э. Хайкин, И.А. Кибель, М.К. Тихонравов, А.Ю. Ишлинский, С.Н. Вернов, Г.Ю. Максимов, И.М. Яцунский. и другие. Это были те ученые, конструкторы и инженеры – ведущие советские профессионалы и специалисты, кто будет непосредственно связан с созданием космической техники, и те, кто мог высказать предложения по космическим научным исследованиям, которые нужно было бы проводить со спутников [19].

В совещании участвовали ученики М.В. Келдыша – выпускники механико-математического факультета МГУ [21], которые как механики и математики – теоретики прокладывали самые первые маршруты космических кораблей. Это кандидаты физико-математических наук – будущие академики математики-механики Дмитрий Евгеньевич Охоцимский (26.02.1921–18.12.2005) – Герой Социалистического Труда СССР за полет Гагарина (1961) и Тимур Магометович Энеев (23.09.1924–08.09.2019) – оба Лауреаты Ленинской премии за первый ИСЗ (1957). Это аспиранты – будущие профессора Всеволод Александрович Егоров (12.12.1930–2001) – Лауреат Ленинской премии (1962) за баллистическое обеспечение полёта станции “Луна-3”, получившей первые фото обратной стороны Луны, известный своими уникальными работами по динамике космических полётов, первый исследователь траекторий перелетов “Земля — Луна”, и Василий Андреевич Сарычев (08.01.1931–01.12.2022) – Лауреат Государственных премии (1970, 1996), в Московском физико-техническом институте с 1973 г. будучи профессором читал лекции по механике космического полёта и системам ориентации спутников и космических аппаратов.

Математик М.В. Келдыш и однокурсник механик Аркадий Александрович Космодемьянский (07.03.1909–08.12.1988), будущий профессор мех-мата МГУ (1939–1975), Лауреат Государственной премии СССР (1953), Генерал-майор инженерно-технической службы (1970), Президент Научного студенческого общества МГУ (1944–1951) [21], окончили физико-механический факультет МГУ в 1931 году. В студенческие годы оба находились под влиянием Александра Ивановича Некрасова (09.12.1883–21.05.1957) [21–23], ученика профессора Николая Егоровича Жуковского

(17.01.1847–17.03.1921) [21–23], который окончил физико-математический факультет Московского университета в 1906 году, с 1918 года профессор и в 1930–1957 гг. заведующий разными кафедрами по механике и аэродинамике, член-корреспондент с 1932 года, в один день с М.В. Келдышем 30.11.1946 избран академиком в Отделении технических наук АН СССР, специальность "механика, гидродинамика", Лауреат Государственной премии СССР (1952). В 1918–1921 гг. А.И. Некрасов был ректором Иваново-Вознесенского политехнического института, где временно после эвакуации из Риги работал отец М.В. Келдыша – профессор Всеволод Михайлович Келдыш (25.06.1878–19.11.1965), генерал-майор инженерно-технической службы (1942), "отец русского железобетона". В.М. Келдыш как авторитетнейший русский ученый принял активное участие в организации Академии архитектуры СССР и был избран действительным членом и вице-президентом академии. В 1956 году академия была преобразована в Академию строительства и архитектуры СССР (АСиА). В.М. Келдыш был избран ее действительным членом.

В 1923 году А.И. Некрасов был введён в Коллегию – руководящий орган Центрального аэро-гидродинамического института (ЦАГИ), в 1930-1938 гг. заместитель начальника ЦАГИ. В 1931 году М.В. Келдыш после окончания МГУ "с отличием" по специальности "чистая математика" по личной инициативе при поддержке А.И. Некрасова, которого знал с детства и трудами которого интересовался, был принят на работу в ЦАГИ. Это было второе судьбоносное решение на пути к космосу. А первое решение было связано с интересом юного Мстислава к инженерной работе отца, потому он и закончил среднюю школу со строительным уклоном. Это были первые шаги формирования уникального талантливого математика, который по мнению директора Математического института им. В.А. Стеклова АН СССР (МИАН) академика Ивана Матвеевича Виноградова (14.09.1891-20.03.1983) "в любом приложении математики способен разобраться лучше всякого" [4, с. 179].

В 1934 году, когда АН СССР переехала из Ленинграда в Москву и вышли Указы об аспирантуре, М.В. Келдыш поступил в аспирантуру (перешедшую в докторантуру) МИАН. В 1935 году ему присуждена ученая степень кандидата физико-математических наук (без защиты диссертации), в 1936 году присуждена ученая степень кандидата технических наук (без защиты диссертации) и в возрасте 25 лет присвоено звание профессора по специальности "аэродинамика"; в 1938 году в возрасте 27 лет присуждена ученая степень доктора физико-математических наук, тема диссертации "О представлении рядами полиномов функций комплексного переменного и гармонических функций" [4, с. 19]. Параллельно основным местом работы М.В. Келдыша оставался ЦАГИ (1931-1946), а в МИАН он повышал свой математический уровень и багаж знаний по математике. Это было третье судьбоносное решение на пути к космосу. Так что в научной тематике диссертаций и в полученных фундаментальных математических результатах нашли отражение прикладные задачи, которыми М.В. Келдыш занимался в Теоретическом отделе ЦАГИ под руководством Сергея Алексеевича Чаплыгина (05.04.1869-08.10.1942), специалиста в области теоретической механики, гидро- и аэромеханики [21-23]. С.А. Чаплыгин выпускник МГУ: окончил физико-математический факультет Московского университета в 1890 году; магистр (1898); доктор прикладной математики (1903). За выдающиеся достижения С.А. Чаплыгин 06.12.1924 избран член-корреспондентом по специальности "математика", а 12.01.1929 – академиком по специальности "аэро- и гидродинамика" Отделения физико-математических наук.

У М.В. Келдыша по жизни и в начале творческого пути было четыре УЧИТЕЛЯ. Первым и главным УЧИТЕЛЕМ и непререкаемым авторитетом был отец В.М. Келдыш. Выпускник физико-математического факультета МГУ 1922 года "чистый математик" Михаил Алексеевич Лаврентьев (19.11.1900-15.10.1980) был вторым УЧИТЕЛЕМ – это были годы студенчества в МГУ и аспирантуры в МИАН, который в один день 30.11.1946 с М.В. Келдышем был избран академиком АН СССР в Отделении физико-математических наук по специальности "математика" и звание Героя Социалистического Труда СССР ему присвоили 29 апреля 1967 года "за выдающиеся заслуги в развитии науки и организацию Сибирского отделения АН СССР".

Третьим УЧИТЕЛЕМ был С.А. Чаплыгин, который принял Октябрьскую революцию и сознательно посвятил свою дальнейшую жизнь развитию отечественной науки, – Герой Социалистического Труда, звание присвоено Указом Верховного Совета СССР от 1 февраля 1941 года "за выдающиеся научные достижения в области аэродинамики, открывшие широкие возможности для повышения скоростей боевых самолетом" и в день 50-летнего юбилея научной деятельности – он стал первым Героем Социалистического Труда среди советских ученых. Удивительная судьба А.С. Чаплыгина: экстраординарный профессор (1904), ординарный профессор кафедры механики теоретической и практической механики (1909–1911, 1917–1924) физико-математического факультета; действительный член Института математики и механики при физико-математическом факультете (1922–1924); покинул Московский университет в 1911 г.; Президент Московского механического общества при МГУ (1936–1942); директор Московских высших женских курсов (1905–1918); ректор 2-го МГУ (1918–1919); начальник Теоретического отдела ЦАГИ, где в 1931-1941 гг. работал М.В. Келдыш. В 1918 году открывается ЦАГИ и его возглавляет Н.Е. Жуковский – отец русской авиации. После кончины в 1921 году Н.Е. Жуковского С.А. Чаплыгин становится главным научным руководителем и председателем Коллегии (1921-1930), директором ЦАГИ (1928–1931).

Четвертым УЧИТЕЛЕМ был выпускник 1914 года физико-математического факультета Петербургского университета И.М. Виноградов – Дважды Герой Социалистического Труда: первое звание присвоено 10 июня 1945 года "за выдающиеся научные достижения в области математики, ... и за многолетнюю плодотворную работу по подготовке кадров математиков" и 13 сентября 1971 года "за выдающиеся заслуги в развитии и организации советской математической науки и в связи с 80-летием". Не случайно такие заслуги отмечены высоким званием, поскольку после переезда в Москву МИАН стал главным базовым академическом институтом, который подготавливал сильнейших в мире молодых математиков на механико-математическом факультете МГУ, организованного в 1933 году после разделения физико-математического факультета и выделения физического факультета.

28 октября 1947 г. в докладе на Юбилейной сессии Отделения физико-математических наук Академии Наук СССР, посвященной 30-летию Великой Октябрьской социалистической революции, академик М.А. Лаврентьев подвел итоги советской математики [24]: "За прошедшие 30 лет советская математика проделала огромный путь. Советская математика сейчас охватывает все основные направления современной математики. Во многих разделах Советский Союз занял первое место в мировой математике... Если до революции и в течение многих лет после революции высшим арбитром ценности результата, значимости того или иного направления считалось мнение иностранных ученых, то теперь этим арбитром являемся мы сами... Если по

основным разделам математики ... мы можем рапортовать: мы догнали, а во многих разделах и перегнали зарубежную математику, то в отношении машинной математики нам нужно еще много усилий, чтобы решить эту задачу. Вычислительная ячейка, созданная в 1935 г. в Математическом институте им. В.А. Стеклова, начинает выполнять, особенно за последние годы, крупные вычисления. Эта ячейка за 12 лет из двух комнат распространилась на целый этаж и занимает сейчас больше половины всей площади Математического института... Мне хочется высказать пожелание, чтобы решение ОФМН о создании специального Института, вынесенное более двух лет назад, нашло скорейшее и полное разрешение".

После Великой Отечественной войны на мех-мате МГУ получили блестящее математическое образование УЧЕНИКИ М.В. Келдыша, которые вместе с УЧИТЕЛЕМ открывали космическую эру человечества. Помогал однокурсник А.А. Космодемьянский, который читал лекции "Механика тел переменной массы", "Динамика космического полета" и проводил семинары по теоретической механике, ракетодинамике, истории механики. А.А. Космодемьянский проявил себя как организатор науки в ракетно-космической отрасли и один из первых научных кадров, воспитанных советской властью, который внёс большой вклад в прикладную аэродинамику и развивал научные основы механики космического полета. Вот его мнение: "Думаю, что для некоторых известных в наши дни ученых интерес к определенным проблемам современной механики зародился в результате работы в научных кружках и семинарах механико-математического факультета МГУ. Я могу назвать, например, следующих товарищей: Д.Е. Охоцимский, Т.М. Энеев, В.А. Егоров, В.В. Белецкий, В.А. Сарычев..." От семинара А.А. Космодемьянского в 50-ые годы отпочковался кружок по космонавтике, основанный В.А. Егоровым и Т.М. Энеевым. Этот кружок перерос вскоре в семинар, который с разными со-руководителями полвека возглавлял В.А. Егоров. В этих кружках и на семинарах зарождались фундаментальные основы теории и методов расчета космических полетов, а их участники стали ПИОНЕРАМИ открытия космической эры и покорения космического пространства [25-27]. Сильной стороной этих молодых исследователей-механиков ИПМ АН СССР является высочайший математический уровень!

Как выражался ученый секретарь и основатель Музея М.В. Келдыша Николай Николаевич Ченцов (19.02.1930-05.07.1992), М.В. Келдыш с 1939 года стал "всемерно засекреченным" [28], а после полета Ю.А. Гагарина 12 апреля 1961 года и избрания 19 мая 1961 года Президентом АН СССР М.В. Келдыш стал "отвечать за науку" и был "всемирно известным" представителем советских достижений и триумфа СССР. Наконец-то, М.В. Келдыш получил возможность под прикрытием Президента Академии наук выступать открыто и проводить пресс-конференции после каждого космического полета и запуска космических кораблей, что успел зафиксировать в истории космонавтики [25-27] его референт Геннадий Александрович Скуридин (01.03.1927-13.01.1991). Г.А. Скуридин работал в ИПМ АН СССР – "Институте Келдыша", с 1962 года – ученый секретарь, с 1966 года – заместитель М.В. Келдыша – председателя МНТС по КИ при АН СССР, с 15 мая 1965 года, когда согласно Постановлению Совета Министров СССР № 392-147 был создан Институт космических исследований АН СССР [29], – и.о. директора, затем заместитель директора Института (до 1981); Лауреат Ленинской премии (1960).

М.В. Келдыш с 1941 года был начальником отдела динамической прочности в ЦАГИ им. Н.Е. Жуковского, где работал с 1931 года после окончания МГУ. В ЦАГИ

М.В. Келдыш занимается не только актуальными теоретическими исследованиями в области математики, но и решает практические задачи самолетостроения. Основополагающие результаты по разработке авиационных конструкций были дважды отмечены Сталинскими премиями (1942 и 1946 годы) и орденом Трудового Красного Знамени (1943 год). В 1944 году по-совместительству в Математическом институте им. В.А. Стеклова АН СССР создал отдел механики. После окончания МГУ в этот отдел поступили на работу "ученики" М.В. Келдыша. 30.11.1946 М.В. Келдыша в возрасте 35 лет, как Л. Эйлера и А.Н. Колмогорова, избирают действительным членом Академии наук СССР в Отделении технических наук по специальности "математика, механика" и он становится Лидером по "прикладной математике". Через два дня по решению лично И.В. Сталина 02.12.1946 М.В. Келдыша назначили начальником Реактивного НИИ-1 МАП (Министерства авиационной промышленности), в 1950-1961 гг. научный руководитель этого учреждения, в котором занимались проблемами ракетостроения, ракетного двигателестроения и космической энергетики. Одновременно М.В. Келдыша как математика привлекают к решению атомной проблемы. С декабря 1946 года основная его научная, организационная и государственная деятельность связана с ракетной техникой и космосом. В 1946 году М.В. Келдыш поручил Д.Е. Охоцимскому в составе отдела механики организовать небольшую группу учёных (С.С. Камынин, Т.М. Энеев, В.А. Егоров, В.А. Сарычев), занимавшихся динамикой космического полета и ракетами – как баллистическими, так и крылатыми. Это был первый ключевой этап на пути к покорению космоса. Эти специалисты занимались ракетно-космическими задачами, расчетами траекторий, небесной механикой, ориентацией, навигацией, стабилизацией космических аппаратов и космических кораблей, а также коррекцией орбит, запуском и посадкой... А содержательную часть космических исследований обеспечивали физики и геофизики...

4. Путь М.В. Келдыша от ученика средней школы со строительным уклоном до Главного Теоретика космонавтики и идеолога космических исследований и Главного Математика, отвечающего за науку, расчеты, ЭВМ, "ракетно-ядерный щит": "В СССР решение масштабных исследовательских и инженерных задач обеспечивалось за счет концентрации ресурсов в системе Академии наук СССР и отраслевых институтах, директивного планирования научных исследований и разработок, осуществляемого Государственным комитетом СССР по науке и технике и Госпланом СССР"

Особого внимания заслуживает Мстислав Всеволодович Келдыш– уникальная ЛИЧНОСТЬ [4] в истории цивилизации и культуры российского государства, который является Ломоносовым 20-го века [1] и именем которого названа "Эпоха Келдыша" [3], которая продолжается и в 21-м веке. Путь Мстислава Келдыша в науке был стре-

мителен. В двадцать лет он окончил математическое отделение МГУ, в двадцать семь – уже доктор наук, в тридцать пять – академик. Талантливых математиков в СССР хватало, но вот вклад М.В. Келдыша в ракетно-космические исследования и в организацию фундаментальной науки трудно переоценить.

М.В. Келдыш после окончания математического отделения физико-механического факультета МГУ по специальности "чистая математика" в 1931 году принят на работу в Теоретический отдел ЦАГИ, где работал под руководством С.А. Чаплыгина до 1946 года и получил две сталинские премии (1942, 1946). 30.11.1946 по рекомендации И.М. Виноградова М.В. Келдыш избран академиком в Отделении технических наук по специальности "математика, механика" и становится лидером по "прикладной математике". 02.12.1946 впервые математик назначается начальником технического Реактивного НИИ (НИИ-1 МАП), с 1950 по 1961 гг. научный руководитель НИИ-1 МАП. Это были ПЕРВЫЕ шаги на пути к покорению космоса.

Следующим после 1946 года ключевым в судьбе М.В.Келдыша стал 1951 год, когда вышло Постановление Совмина "О работах по РДС-6Т" № 1552-774сс/оп от 09 мая 1951 г. за подписью И.В. Сталина, в котором М.В. Келдыша фактически назначили "Главным Математиком СССР". Обратите внимание на состав математической секции: какие ещё совсем молодые, но уже ВЕЛИКИЕ математики, которые решали судьбу мира на планете! Все заслуженно стали Героями Социалистического Труда. Наступает расцвет "Эпохи Келдыша", когда "царица всех наук" МАТЕМАТИКА, расчеты и ЭВМ на высшем уровне официально на уровне Совета Министров СССР и лично И.В. Сталина – главы руководства СССР признаны приоритетными – это было начало становления в СССР "цифровой цивилизации":

«1. Разрешить Академии наук СССР (т. Несмеянову):

а) увеличить общий штат Математического института АН СССР на 73 человека сверх установленного лимита АН СССР на 1951 г. . . . ;

в) организовать в Математическом институте АН СССР отдел прикладной математики со штатом 30 человек в составе двух секторов.

4. Обязать Первое главное управление при Совете Министров СССР (тт. Ванникова, Завенягина) организовать в составе Научно-технического совета математическую секцию (секцию № 7) для научного руководства по разработке конструкций, быстродействующих вычислительных машин, а также методов их эксплуатации в составе:

академик Келдыш М.В.	—	председатель секции
академик Петровский И.Г.	—	член секции
академик Соболев С.Л.	—	- " -
член-корреспондент АН СССР Боголюбов Н.Н.	—	- " -
член-корреспондент АН СССР Тихонов А.Н.	—	- " -
академик Лаврентьев М.А.	—	член секции (по вопросам вычислительных машин)
член-корреспондент АН СССР Лебедев С.А.	—	- " -
инженер Базилевский Ю.Я.	—	- " -
инженер Лесечко М.А.	—	- " -

(Прим. автора: Базилевский Ю.Я. и Лесечко М.А. из СКБ-245 – разработчики первой серийной промышленной ЭВМ "Стрела".)

Возложить на секцию № 7 рассмотрение планов научно-исследовательских, экспериментальных и проектных работ, а также проектов математических машин и планов работ организаций, выполняющих расчетные работы по тематике Первого главного управления при Совете Министров СССР.»

М.А. Лаврентьев и С.Л. Соболев в 1957 году переехали в Новосибирск, где создавали первый в СССР наукоград, а Н.Н. Боголюбов углубился в теоретическую и математическую физику, так что ключевыми руководителями и организаторами в науке и образовании фактически были ТРОЕ: М.В. Келдыш, А.Н. Тихонов и И.Г. Петровский, назначенный в 1951 году ректором МГУ. Это подтверждает и референт ректора МГУ Е.Б. Козельцева, которая с 1948 до 1980 гг. была куратором МГУ от КГБ.

С 1951 г. М.В. Келдыш как Главный математик – председатель “математической секции” НТС отвечал за математику, расчеты и ЭВМ в стратегических “Трех проектах” на основе “новых технологий” (прикладная математика, расчеты, ЭВМ).

В 1953 г. было принято историческое стратегическое решение о создании первого в мире Института прикладной математики АН СССР: Распоряжение СМ СССР № 6111-рс об образовании Отделения прикладной математики Математического института АН СССР, г. Москва. Кремль, 18.04.1953. Сов. Секретно (рассекречено).

По рекомендации ЛИЧНО М.В. Келдыша и Президиума АН СССР было создано секретное ОПМ МИАН СССР, чтобы не дробить и сохранить Математический институт им. В.А. Стеклова! В 2024 году 90-летний юбилей Математического института им. В.А. Стеклова АН.

Рождение МИАН: в 1921 году создан Физико-математический институт РАН на основе

- Математического кабинета (организован В.А. Стекловым в 1919 г.),
- Физической лаборатории (организована Б.Б. Голицыным в 1912 г.) и
- Постоянной Центральной Сейсмической комиссии, образованной в 1897 г.

Директорами Физико-математического института РАН были: В.А. Стеклов (1921-1926), А.Ф. Иоффе (1926-1928), А.Н. Крылов (1928-1932) и И.М. Виноградов (1932-1934).

В 1934 году решением Общего собрания АН СССР Физико-математический институт был разделен на Институт математики АН СССР и Институт физики АН СССР. Институт математики получил официальное наименование – Математический институт им. В.А. Стеклова АН СССР (МИАН). Директором МИАН был назначен И.М. Виноградов (1934-1941, 1944-1983). В годы войны 1941-1944 И.М. Виноградов читал лекции в Казанском университете – действовало централизованное решение об обязательной эвакуации из Москвы членов АН СССР старше 50 лет. “За выдающиеся научные достижения в области математики, разработку мощных аналитических методов теории чисел и за многолетнюю плодотворную работу по подготовке кадров математиков”, указом Президиума Верховного Совета СССР от 10 июня 1945 года Виноградову Ивану Матвеевичу присвоено звание Героя Социалистического Труда с вручением ордена Ленина и золотой медали “Серп и Молот”, а второго Героя получил 13.09.1971.

В 2024 году 160-летний юбилей со дня рождения математика академика Стеклова Владимира Андреевича (09.01.1864-30.05.1926), вице-президента АН (31.05.1919-30.05.1926), который много сделал для спасения Академии наук после революций 1917 года. Членство в АН:

Ступени членства	Дата избрания	Специальность	Отделение
член-корреспондент	07.12.1902	по разряду математических наук	Физико-математическое Отделение
Адъюнкт	06.11.1910	прикладная математика	Физико-математическое Отделение
экстраординарный академик	03.03.1912		
ординарный академик	01.07.1912		

Зам директора МИАН М.В. Келдыш был в 1945-1947 и 1949-1953 гг. В 1956 году М.В. Келдыш назначен Председателем Комиссии по подготовке и осуществлению запуска ПЕРВОГО искусственного спутника Земли [31]. В 1959-1960 гг. для координации работ в области космонавтики был образован Межведомственный научно-технический совет по космическим исследованиям при Академии наук СССР (МНТС по КИ), председателем которого был М.В. Келдыш в статусе министра и генерала (1959-1978) [32-41].

За четыре года с 1957 по 1961 СССР совершил поражающий воображение землян прорыв в истории ЦИВИЛИЗАЦИИ: СССР открыл "космическую эру", покорил и освоил Космос!

Искусственные спутники Земли: 04.10.1957 СССР запустил ПЕРВЫЙ в истории планеты "Простой" ИСЗ и открыл КОСМИЧЕСКУЮ ЭРУ ЧЕЛОВЕЧЕСТВА – эпохальный прорыв; следующие ИСЗ были научно-исследовательскими – требовались знания о космической среде, электромагнитных излучениях, магнитосфере и ионосфере: 03.11.1957 – второй ИСЗ посвящен 40-летию Октябрьской революции; 15.05.1958 – третий ИСЗ; 04.02.1961 – четвертый спутник.

Космические аппараты для исследования Луны: 02.01.1959 "Луна-1" – пролет над Луной; 12.09.1959 "Луна-2" – на поверхности Луны установили вымпел с гербом СССР в Море Ясности; 04.10.1959 "Луна-3" – Фотографирование обратной стороны Луны.

Корабли-спутники – подготовка к полету человека: 15.05.1960 – манекен; 19.08.1960 – Белка и Стрелка; 01.12.1960, 09.03.1961, 25.03.1961 – живые организмы.

12 апреля 1961 года ВПЕРВЫЕ человек на космическом корабле пролетел над Землей и это был советский гражданин Юрий Гагарин – СССР совершил очередной эпохальный прорыв ЦИВИЛИЗАЦИИ и был впереди планеты всей! Запуск первого ИСЗ посвящен 100-летию К.Э. Циолковского и 50-летию С.П. Королева, а полёт Гагарина – это подарок к 50-летию М.В. Келдыша!

19 мая 1961 года после полета Ю.А. Гагарина в возрасте 50 лет Дважды Героя Социалистического Труда М.В. Келдыша избрали Президентом Академии наук СССР! М.В. Келдыш – самый молодой Президент АН и ПЕРВЫЙ МАТЕМАТИК – Президент АН (1961-1975)! Это ЛУЧШИЙ Президент за всю историю Академии наук! В советское время без ученых и Академии наук стратегические решения не принимались! М.В. Келдыш отвечал за науку в государстве!

Стало очевидно: кто владеет космосом, тот правит миром! США не могли смириться со своим поражением и началась "гонка" с угрозой "ядерной войны". США и СССР уже создали "ядерные бомбы" и баллистические ракеты дальнего действия как средства доставки ... "Лунная речь" президента Джона Ф. Кеннеди, Хьюстон, Техас, 12 сентября 1962 г. [42]:

"...мы стремимся покорить космос просто потому, что он есть. Вселенная, Солнечная система, Луна – это неизведанный мир, который дарит нам надежду на новые знания и благополучие. Мы отправляемся в далекий путь, и да хранит нас Господь, ведь это опаснейшее и величайшее путешествие в истории человечества. "

За этими миролюбивыми словами, обращенными к населению США, скрывалась истинная новая политика начала "тонки в космосе"! В октябре 1962 года произошел Карибский кризис.

Советский ответ возглавил академик М.В. Келдыш как Президент АН СССР, Председатель МНТС по КИ при АН СССН в статусе министра и генерала, Главный Теоретик космонавтики, Главный Математик страны (1946, 1951), который разберется в любой технической проблеме.

Почему МЫ – ПЕРВЫЕ покорили космос и открыли космическую эру – основу цивилизации с постиндустриальным информационным укладом своим умом без привлечения зарубежных технологий и спецов? Пора извлекать полезные уроки для реализации Стратегии научно-технологического развития... Ответ, с одной стороны, прост: талантливый народ и КАДРЫ РЕШИЛИ ВСЁ! Но этого недостаточно... А с другой стороны, важнейшие факторы: мобилизационная экономика; управление и команды из одного центра; проведение индустриализации; создание инфраструктуры; денег на науку не жалели! наукой руководила АКАДЕМИЯ НАУК СССР – реальный ШТАБ научных фундаментальных и прикладных исследований и форпост СССР в мире!

О значении Мстислава Всеволодовича для всей советской космонавтики красноречиво говорят факты. Так, первое совещание об искусственном спутнике Земли состоялось 14 февраля 1954 г. в кабинете Келдыша. В том же году М.В. Келдыш вместе с Сергеем Павловичем Королёвым (12.01.1907-14.01.1966) и Михаилом Клавдиевичем Тихонравовым (16.07.1900-04.03.1974) представили в Правительство письмо с предложением о создании ИСЗ. А запуск первого ИСЗ в 1957 году Н.С. Хрущёв разрешил только под личные гарантии М.В. Келдыша. Он совмещал большой талант учёного с большим государственным весом как руководителя науки. Не только первые советские искусственные спутники Земли, но и полёты автоматических станций к Луне и к планетам солнечной системы, первый луноход и главное – первый космический полёт человека – во все эти проекты внесли непосредственный личный вклад М.В. Келдыш и "Институт Келдыша".

М.В. Келдыш выступил одним из инициаторов широкого развертывания в нашей стране работ по изучению и освоению космоса и как Главный Теоретик космонавтики возглавил решающий участок в их проведении. Когда же полёты в космос стали реальностью, М.В. Келдыш вместе с С.П. Королёвым сначала в 1955 году, а потом в 1962 году составили программу космических исследований, которая на десятилетия предопределила развитие советской науки. Это и исследование Луны, планет солнечной системы, развитие космической техники, разработка принципиально новой научной аппаратуры и т.д. Не говоря уже о возможностях, которые предоставляли космические полёты для военной разведки, исследования Земли и народного хозяйства.

Эти и другие его заслуги получили признание. М.В. Келдыш стоял во главе АН СССР 14 лет – с 19 мая 1961 по 19 мая 1975 года – и за это время вывел отечественную науку в мировые лидеры. Авторитет Мстислава Всеволодовича в научном сообществе был непререкаемым. "Надо пойти посоветоваться к Мудрому", – говорили многие,

зная о феноменальной способности учёного разобраться в самых сложных и, казалось бы, не поддающихся решению проблемах. Такого другого теоретика и практика да ещё и математика больше нет! За 10 лет успехов М.В. Келдыша–Президента в год его 60-летия присвоили Третьего Героя Социалистического Труда (1971).

5. Информационно-математический аспект космических исследований

Г.Г. Малинецкий [17]: "Техносферу часто называют второй природой, сравнивая ее с биосферой. С неменьшим основанием формирующуюся компьютерную реальность можно назвать третьей природой." Фундаментальные основы такой триады заложены в СССР под руководством М.В. Келдыша! Ключевым фактором и главным движителем становления и развития компьютеризации, "машинной" математики, "цифровизации" и информационно-математического направления явился глобальный проект создания "ракетно-ядерного щита", который стимулировал открытие космической эры и покорение космоса. В 1955 году как государственный деятель и Главный Математик М.В. Келдыш определил две главные задачи, которые явились триггером создания ракетно-космической отрасли, – это разведка и наблюдение Земли из космоса для научных и прикладных задач. В СССР фактически существовало две научно-технические базы: одна, засекреченная, военно-промышленного, а другая открытая – народно-хозяйственного комплекса. М.В. Келдыш с 1939 года имел высший уровень секретности, но после избрания в Президенты появилась возможность открытых публичных выступлений и международного сотрудничества.

Космические полёты и ядерные исследования потребовали сложнейших расчётов и новых методов расчётов. С именем М.В. Келдыша и как автора важных работ, и как руководителя большого коллектива учёных связано становление современной вычислительной и прикладной математики и создание первых в СССР электронных вычислительных машин (ЭВМ). Первый сигнал М.В. Келдыш получил в 1947 году лично от И.В. Сталина, который сообщил, что некто фон Нейман в США разрабатывает "большие математические счетные машины" и поручил заняться этой проблемой. В 1947 году в Успехах математических наук была опубликована первая статья о разработках ЭВМ в США [43]. В 1948 году, который принято считать годом основания "информатики" в нашей стране и 4 декабря отмечают День информатики, были основаны "секретный" СКБ-245 и Институт точной механики и вычислительной техники АН СССР (ИТиВТ). В "Институте Келдыша" в 1953 году организован первый в СССР отдел программирования, заведующий отдела Алексей Андреевич Ляпунов (07.10.1911-23.06.1973) – член-корреспондент Академии наук СССР, доктор физико-математических наук, профессор, основоположник советской кибернетики и программирования. Вскоре А.А. Ляпунов создает первый в СССР отдел кибернетики, а заведующим первого в СССР отдела автоматизации программирования становится профессор Михаил Романович Шура-Бура (21.10.1918-14.12.2008) – первый программист в СССР, который в 1953 году перешел в ОПМ МИАН из ИТиВТ, где работал с его основания. Под руководством и личном участии в разработке системы команд М.В. Келдыша создана первая в СССР серийная промышленная ЭВМ "Стрела". А.А. Ляпунов и М.Р. Шура-Бура готовили первых программистов и алгоритмистов на мех-мате МГУ, при котором был создан первый в СССР вузовский

вычислительный центр: при содействии М.В. Келдыша, А.Н. Тихонова и ректора И.Г. Петровского в 1955 году была установлена ЭВМ "Стрела". В 2024 году три юбилея отечественных ЭВМ:

- 1954 год – 70 лет назад в "Институте Келдыша" ввели в действие первую отечественную промышленную ЭВМ "Стрела", на которой осуществляли расчеты для запуска первого спутника и полета первого космонавта Ю.А. Гагарина, а также для "Атомного проекта".
- 1964 год – 60 лет назад ЭВМ "Весна" введена в действие в августе, впервые введен многопрограммный поток задач, Т.А. Сушкевич и Ю.М. Баяковским построены первые в СССР компьютерные графики и анимационный фильм по космическим исследованиям.
- 1964 год – 60 лет назад разработана ЭВМ "БЭСМ-6", в 1966 году ПЕРВЫЙ экземпляр поставили и ввели в действие в ОПМ МИАН СССР – академическом "Институте Келдыша".

В 2024 году следует вспомнить важные юбилейные даты: август 1964 года – 60 лет назад ввели в строй ЭВМ "Весна" с мультипрограммной Операционной системой (разработка "Института Келдыша" под руководством М.Р. Шура-Бура и В.С. Штаркмана), в "математической" сдаче-приемке которой приняла Т.А. Сушкевич, которая разработала первую большую программу для научных космических исследований и математического моделирования прохождения ракет и спутников через радиационные пояса в ионосфере и возмущений для космической связи [44, 45]. Радиационные пояса были открыты во время полетов первого и второго спутников в 1957-1958 гг. [46]. Вместе с Ю.М. Баяковским ВПЕРВЫЕ в СССР на компьютере были построены графики и создана анимация на основе расчетных данных Т.А. Сушкевич. В 1964 году была разработана первая БЭСМ-6, которую установили в "Институте Келдыша" в 1966 году и ввели в строй в 1967 году. Эта ЭВМ – легенда СССР – находится в Музее науки Лондона!

Связующая цепь времен выдающихся российских достижений в математике для космоса:

- 1136 – "Трактат о числах" – первый научный труд, посвященный изучению чисел.
- 1703 – "Арифметика" Магницкого – первый учебник по математике.
- Леонард Эйлер (26.10.1945-07.09.1783) – становление наук: математик, механик, физик.
- Михаил Васильевич Ломоносов (19.11.1711-15.04.1765), первый русский крупный учёный естествоиспытатель, физик, химик, математик, астроном...
- Николай Иванович Лобачевский (01.12.1792-24.02.1856) – математик первооткрыватель неевклидовой геометрии, ректор Казанского университета (1827-1845).
- Михаил Васильевич Остроградский (24.09.1801-01.01.1862) – первый ординарный академик по "прикладной математике" с 21.12.1831, по "чистой математике" с 15.06.1855.
- Пафнутий Львович Чебышёв (16.05.1821-26.11.1894) – математик и механик, основоположник петербургской математической школы, академик по "прикладной математике"; окончил физико-математическое отделение философского факультета МГУ (1837-1846).

- Крылов Алексей Николаевич (15.08.1863-26.10.1945) – Первые лекции о приближенных вычислениях прочитаны в 1906 году и изданы в 1911 году, математик, физик, первый академик по "математической физике" 02.04.1916!
- Можайский Александр Федорович (21.03.1825-01.04.1890) – в Академии не состоял, основоположник самолетостроения, построил первый управляемый человеком самолет.
- Жуковский Николай Егорович – первый аэрогидродинамик, создатель аэродинамики как науки, "отец русской авиации".
- Чаплыгин Сергей Алексеевич – соратник Н.Е. Жуковского – УЧИТЕЛЬ М.В. Келдыша.

6. Заключение

«*Мстислав Всеволодович – это уникальное явление. Такого не было и не будет... Что поражало при общении с ним – это впечатление, что имеешь дело с ядерным реактором, который внешне интеллигентен, но главное в нём – это внутреннее существо. Это непрерывное горение, необычайный внутренний накал, огромное количество внутренней энергии – впечатление чего-то скрытого, могучего в этом человеке...*» Академик О.Г. Газенко.

«*Он вызывал чувство удивления своей неутомимостью в работе, чувство удивления масштабом своей деятельности и, я должен сказать, вызывал чувство большой человеческой симпатии... Он необычайно ответственно относился к своей деятельности, необычайно самокритично, иногда даже до самоистязания.*» Академик М.А. Марков.

«*Он обладал свойством эксперта, свойством задать тот единственный вопрос, который проясняет задачу. Он видел глубоко и далеко вперёд. Он не мог не работать.*» Академик Б.В. Раушенбах.

«*Мстислав Всеволодович – рыцарь науки. Всё – для науки и ради науки. Я не знаю такого второго человека. Я абсолютно уверен, что за науку он фактически положил жизнь*». Академик Г.К. Скрябин.

«*В современной истории таких людей, как Мстислав Всеволодович, было очень немного. И его вклад в науку, в развитие жизни настолько велик, что и обсуждать его не имеет смысла. Всё, к чему прикасался Мстислав Всеволодович, освещалось ещё и значительностью его личности. Я думаю, что ещё долго будет помниться его образ. Это один из самых выдающихся людей, которых мы когда-либо знали...*» Академик Ю.А. Осипьян.

Приведу слова из выступления академика Б.Е. Чертока, Героя Социалистического Труда за полет Ю.А. Гагарина, на Торжеством заседании РАН в октябре 2007 года по случаю 50-летия запуска ПЕРВОГО искусственного спутника Земли: "*М.В.Келдыш был истинным лидером нашей науки. Будучи президентом Академии, он вышел далеко за пределы тех прав и возможностей, которые формально государство отвело науке. Он поднимал науку, образованность и тем самым величие страны. Именно такие люди должны руководить страной.*"

В 2024 году отмечается 40-летие со дня кончины Дмитрия Федоровича Устинова (30.10.1908-20.12.1984): с 1946 года и до конца жизни "генерал науки" М.В. Келдыш тесно сотрудничал с "маршалом-легендой обороны" – это был успех создания "Ракетно-ядерного щита"!

Список литературы

[1] Мстислав Келдыш // «Великие умы России» под ред. В.С.Губарева. Выпуск 2. М.: Издательский дом «Комсомольская правда», 2016. 96 с.
https://avidreaders.ru/book/mstislav-keldysh.html

[2] Губарев В.С. Три звезды Героя: знания и страсти. Несколько страниц из жизни великого ученого нашей Родины М.В. Келдыша // Земля и Вселенная. 2021. № 1. С. 86-100 (начало).
https://www.elibrary.ru/item.asp?id=44870432;
Земля и Вселенная. 2021. № 2. С. 79-92 (окончание).
https://www.elibrary.ru/item.asp?id=44873285

[3] Марчук Г.И., Алдошин С.М., Григорьев А.И., Козлов В.В. Эпоха М.В. Келдыша: выводы и уроки. 17 февраля 2011 г.
http://www.ras.ru/news/shownews.aspx?id=6531c71e-d91f-44a2-bd7e-812a1405cffc

[4] Келдыш М.В. Творческий портрет по воспоминаниям современников. М.: Наука, 2001. 399 с. 2-е изд. 2002.
http://elib.biblioatom.ru/text/keldysh-tvorcheskiy-portret_2002/go,0/

[5] Конференция «математика, механика и прикладные исследования, посвященная 110-летию со дня рождения Мстислава Всеволодовича Келдыша.
https://event.msu.ru/keldysh

[6] Константин Циолковский: скромный учитель математики и основоположник современной космонавтики / Президентская библиотека.
https://www.prlib.ru/news/688644?ysclid=lv0oksqwmx593130199

[7] Выступление В.В.Путина на XXI съезде Всероссийской политической партии “Единая Россия” 17 декабря 2023 года.
http://kremlin.ru/events/president/news/73013

[8] Послание Президента Федеральному собранию, Москва, 29.02.2024
http://kremlin.ru/events/president/news/73585

[9] Указ Президента Российской Федерации от 15.03.2021 № 143 “О мерах по повышению эффективности государственной научно-технической политики”
http://publication.pravo.gov.ru/Document/View/0001202103150017?ysclid=lu6pcpydf7452076251

[10] Указ Президента Российской Федерации от 25.04.2022 № 231 “Об объявлении в Российской Федерации Десятилетия науки и технологий (2022-2031)”
http://publication.pravo.gov.ru/document/0001202204250022

[11] Указ Президента Российской Федерации от 31.03.2023 № 229 “Об утверждении Концепции внешней политики Российской Федерации”.
http://publication.pravo.gov.ru/Document/View/0001202303310007

[12] Распоряжение Правительства от 20.05.2023 № 1315-р “Концепция технологического развития до 2030 года”
http://publication.pravo.gov.ru/document/0001202305250050

[13] Совещание по вопросам развития космической отрасли, 26.10.2023, Московская область, Королёв, Ракетно-космическая корпорация “Энергия”.
http://kremlin.ru/events/president/news/72606

[14] Указ Президента Российской Федерации от 26.10.2023 № 812 “Об утверждении Климатической доктрины Российской Федерации”
http://publication.pravo.gov.ru/document/0001202310260009?index=1

[15] Указ Президента Российской Федерации от 10.10.2019 г. № 490 “ О развитии искусственного интеллекта в Российской Федерации” (в редакции Указа Президента Российской Федерации от 15.02.2024 № 124)
http://pravo.gov.ru/proxy/ips/?docbody=&firstDoc=1&lastDoc=1&nd=102608394

[16] Указ Президента Российской Федерации от 28.02.2024 № 145 “О Стратегии научно-технологического развития Российской Федерации”
http://actual.pravo.gov.ru/text.html#pnum=0001202402280003

[17] Малинецкий Г.Г. Развитие компьютерного пространства как фактор стратегической стабильности России. М.: ИПМ им. М.В. Келдыша, 2024. 104 с. https://doi.org/10.20948/mono-2024-malin
https://keldysh.ru/e-biblio/malin/

[18] Вернадский В.И. (1863-1945). Труды по всеобщей истории науки / Общ. ред. и вступ. ст. С.Р. Микулинского; Предисл. А.Л. Яншина; АН СССР. Комис. по разраб. науч. наследия акад. В.И. Вернадского и др. 2-е изд. М.: Наука, 1988. 334 с.
https://reallib.org/reader?file=1515935;
https://search.rsl.ru/ru/record/01001398228

[19] Прикладная небесная механика и управление движением. Сборник статей, посвященный 90 летию со дня рождения Д.Е. Охоцимского / Составители: Т.М. Энеев, М.Ю. Овчинников, А.Р. Голиков. М.: ИПМ им. М.В. Келдыша, 2010. 368 с.
https://keldysh.ru/memory/okhotsimsky/index.htm

[20] Т.А. Сушкевич, С.А. Стрелков, С.В. Максакова. 60 лет от первого совещания по ИСЗ до современных систем дистанционного зондирования и мониторинга Земли из космоса: информационно-математический аспект (история и перспективы) // Оптика атмосферы и океана. 2014. Т. 27, № 7. С. 573-580.
https://ao.iao.ru/ru/content/vol.27-2014/iss.07/2

[21] Летопись Московского университета.
http://letopis.msu.ru/

[22] Информационная система «Архивы Российской академии наук». Персональный состав РАН.
https://isaran.ru/?q=welcome;https://www.isaran.ru/?q=ru/persostav

[23] Российская академия наук. Персональный состав. В 4-х кн. М.: Наука, 2009. (1724-1917; 1918-1973; 1974-1999; 2000-2009). Действительные члены. Члены-корреспонденты. Почетные члены. Иностранные члены. (К 275-летию Академии наук) (при поддержке РФФИ)

[24] Лаврентьев М.А. Пути развития советской математики // Изв. АН СССР. Сер. матем. 1948. Т. 12, вып. 4. С. 411–416. Доклад, прочитанный 28 октября 1947 г. на Юбилейной сессии ОФМН Академии Наук СССР, посвященной 30-летию Великой Октябрьской социалистической революции.
http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=im&paperid=3088&option_lang=rus

[25] Освоение космического пространства в СССР. Официальные сообщения ТАСС и материалы центральной печати 1957-1967 гг. // ИКИ АН СССР. Отв. ред. Скуридин Г.А. М.: Издательство «Наука», 1971. 555 с.
https://epizodsspace.airbase.ru/bibl/osvoen-kosm-pr-sssr/1957-1967/01.html

[26] Освоение космического пространства в СССР. Официальные сообщения ТАСС и материалы центральной печати октябрь 1967-1970 гг. // ИКИ АН СССР. Отв. ред. Петров Г.И. М.: Издательство «Наука», 1971. 360 с.
https://epizodsspace.airbase.ru/bibl/osvoen-kosm-pr-sssr/1968-1970/01.html

[27] Освоение космического пространства в СССР. По материалам центральной печати 1971 г. // ИКИ АН СССР. Отв. ред. Нариманов Г.С. М.: Издательство «Наука», 1973. 302 с.
https://epizodsspace.airbase.ru/bibl/osvoen-kosm-pr-sssr/1971/01.html

[28] Ченцов Н.Н. Всемирно известный, всемерно засекреченный // Наука и жизнь. 1991. № 2. С. 102-107.
https://web.archive.org/web/20131127005054
https://publ.lib.ru/ARCHIVES/N/''Nauka_i_jizn'''_(jurnal)/Nauka_i_jizn',1991,N02.\%5bdjv-fax\%5d.zip

[29] История ИКИ РАН. От идеи Объединенного Института космических исследований до пятидесятилетия.
https://iki.cosmos.ru/popular/iki-history

[30] Советская космическая инициатива в государственных документах. 1946-1964 гг. / Под ред. Ю.М. Батурина. М.: Издательство «РТСофт», 2008. 416 с.
https://search.rsl.ru/ru/record/01004133883?ysclid=lpnsmtx4hm35013959;
http://www.coldwar.ru/arms_race/iniciativa/

[31] Постановление Совета Министров СССР № 149-88с «О создании объекта "Д"». План разработки и изготовления объекта «Д», проведения научно-исследовательских работ и эскизной проработки по объекту «Д» от 30 января 1956 г.

Совершенно секретно. Особая папка (рассекречено)
https://rvsn.info/library/docs/doc_1_1072.html

[32] Постановление ЦК КПСС и Совета Министров СССР "О развитии исследований по космическому пространству" от 10 декабря 1959 г. № 1388-618. Совершенно секретно особой важности (рассекречено)
https://www.kosmonavtika.com/bibliographie/documents/1388-618.pdf;
http://sovet.cosmos.ru/sites/default/files/history/2_9.pdf

[33] Постановление Совета Министров СССР «Об утверждении Положения о Междуведомственном научно-техническом совете по космическим исследованиям при Академии наук СССР» от 24 сентября 1960 г. № 1026-421 секретно (рассекречено).
http://www.coldwar.ru/arms_race/iniciativa/ob-utverzhdenii-polozheniya.php

[34] Постановление Совета Министров СССР «О комиссии по пускам космической ракеты объекта «М» № 999-414 9 сентября 1960 г. СОВ.СЕКРЕТНО (рассекречено)
http://www.coldwar.ru/arms_race/iniciativa/o-komissii-po-puskam.php

[35] Батурин Ю.М. Академия наук и космос. К 50-летию полета Ю.А.Гагарина.
https://arran.ru/data/collections/col8_.pdf?ysclid=lptkvvht75345858949

[36] Ракетные войска стратегического назначения. Справочник. Библиотека РВСН. Документы. Исторические документы: РВСН и ракетостроение (1945-1967).
https://rvsn.info/library_main.html
https://rvsn.info/index.html

[37] Задача особой государственной важности. Из истории создания ракетно-ядерного оружия и ракетных войск стратегического назначения (1945-1959 гг.): сб. док. / Сост. Ивкин В.И., Сухина Г.А. М.: Российская политическая энциклопедия (РОССПЭН), 2010.

[38] Сушкевич Т.А. Главный Теоретик М.В. Келдыш и Главный Конструктор космонавтики С.П. Королев – покорители космоса // Современные проблемы дистанционного зондирования Земли из космоса. 2011. Т. 8, № 1. С. 9–25.
http://jr.rse.cosmos.ru/article.aspx?id=819

[39] Сушкевич Т.А. М.В.Келдыш – организатор международного сотрудничества в космосе и первой советско-американской Программы "Союз-Аполлон" (ЭПАС) // Современные проблемы дистанционного зондирования Земли из космоса. 2011. Т. 8, № 4. С. 9-22.
http://jr.rse.cosmos.ru/article.aspx?id=930

[40] Черток Б.Е. Ракеты и люди (в 4-х книгах). 2-е издание. М.: Машиностроение, 1999.
https://bookscafe.net/author/chertok_boris-17704.html

[41] Каманин Н.П. Скрытый космос (рукописи “Космические дневники генерала Каманина” в 4-х книгах). М.: Инфортекст-ИФ, 1995–1997.
https://bookscafe.net/author/kamanin_nikolay-9440.html

[42] “Лунная речь” президента Джона Ф. Кеннеди, Хьюстон, Техас, 12 сентября 1962 г. // Электронный журнал «Все о космосе»
https://aboutspacejornal.net/2016/09/12/

[43] Быховский М.Л. Новые американские счетно-аналитические машины // УМН. 1947. Т. 2. № 2. С. 231–234.
https://www.mathnet.ru/links/b5266bfba93d8beda3ab091952bad35b/rm6951.pdf

[44] Масленников М.В., Сигов Ю.С., Сушкевич Т.А. Численное решение задачи о стационарном обтекании тела разреженной плазмой // Тезисы докладов. Четвертое совещание по магнитной гидродинамике. 22-27 июня 1964 г. Рига: Изд. АН Латв.ССР, 1964.

[45] Масленников М.В., Сигов Ю.С. Дискретная модель вещества в задаче об обтекании тез разреженной плазмой // Докл. АН СССР. 1964. Т. 159. № 5. С. 1013–1016.
https://www.mathnet.ru/links/7f1b79b5833dfb629db8c4abe351d33c/dan30654.pdf

[46] Кузнецов В Д “Космические исследования Института земного магнетизма, ионосферы и распространения радиоволн им. Н.В. Пушкова РАН”// УФН. 2010. Т. 180. № 5. С. 554–560.
https://ufn.ru/ru/articles/2010/5/l/

Модификация схемы метода анализа иерархии для использования качественной информации о предпочтениях критериев

Д.Е. Шапошников, А.А. Кулёва

Нижегородский государственный университет
имени Н.И. Лобачевского

При разработке нового курса «Принятие решений на основе данных» рассматриваются системы поддержки принятия решений и решаемые ими задачи. В качестве одной из них рассматривается проблема многокритериальной оценки при анализе больших данных для некоторой совокупности объектов. Предполагается, что для формирования итоговой оценки необходимо учитывать индивидуальные предпочтения лица, принимающего решение, или эксперта. В качестве основы такого учёта используется метод анализа иерархий (МАИ), схема которого модифицируется использованием качественной информации и оптимизационными задачами вычисления коэффициентов предпочтительности по принципу гарантированного результата. Приведены способы решения оптимизационных задач и примеры использования.

Ключевые слова: метод анализа иерархий, многокритериальная оценка, принцип гарантированного результата, качественная информация о предпочтительности

1. Введение

Курс «Принятие решений на основе данных» соответствует современному направлению развития интерактивных систем, основанных на обработке данных и предполагающих решение проблем оценки и выбора рациональных решений в условиях слабоструктурированных, неполных данных различного объёма, а также данных, имеющих форму экспертных суждений. Студенты получают компетенции в основах аналитики данных экономических систем (как традиционной экономической статистики, так и современными подходами извлечения, хранения и обработки данных), компетенции в области теоретических и практических методов формирования рациональных решений на основе данных, а также принципов построения человеко-машинных систем анализа и принятия решений, предполагающих использования неформальных знаний экспертов.

Структура курса предполагает возможности изучения как студентами, имеющими серьёзную базовую подготовку в области математики и информационных технологий, так и студентами, не имеющими такой подготовки (гуманитарных специальностей). Для последних предусматривается наличие отдельных «выравнивающих» тем.

Основными разделами курса являются:

- теоретические основы принятия решений (кратко о теории систем и моделирования, многокритериальный анализ и оценка альтернатив, решение задач оптимизации);
- теоретические и практические основы систем хранения данных (реляционные базы данных, другие типы баз данных, современные подходы и алгоритмы поиска и предварительной обработки данных);
- сбор данных (вопросы сбора и хранения данных - классификация типов данных, современное состояние и подходы сбора и предварительной обработки, анализ достоверности и очистка данных, группировка данных, средние величины, выборочный сбор данных);
- статистический анализ и визуализация;
- данные как основа принятия решений (некоторые модели проблем принятия решений, принятие решений по модификации сети передачи данных, применение программных средств поиска рациональных решений).

Одной из главных проблем, рассматриваемых в курсе «Принятие решений на основе данных» является проблема, возникающая в системах поддержки принятия решений (СППР) – оценка и последующий (основанный на оценке) выбор вариантов (объектов), являющихся лучшими с точки зрения лица, принимающего решение (ЛПР). В социально-экономических, образовательных и других системах (в частности, например, при оценке кандидатов на замещение вакансий, в системах оценки компетенций и личностных качеств людей) ЛПР должно всесторонне проанализировать имеющиеся варианты с точки зрения его собственной системы приоритетов и предпочтений, так как несёт ответственность за полученную оценку и последующий выбор рационального варианта, под которым понимается существование понятных объяснимых причин, приведших к такой оценке. Такие задачи являются задачами многокритериального выбора (многокритериальной оценки).

Одной из проблем анализа больших данных является необходимость формирования итоговой оценки по всем параметрам (критериям) или по их части для каждого объекта совокупности. Например, в социологических исследованиях для каждого объекта (индивидуума) необходимо сформировать несколько итоговых оценок (не более десяти) из большого количества (сто и более) индивидуальных параметров на основе логического объединения этих параметров в группы.

Также, наряду с количественной проблемой размерности, существует и качественная: разные критерии (параметры) могут иметь разную предпочтительность как на индивидуальном уровне, так и на уровне групп.

В итоге лицо, принимающее решение (ЛПР), должно, во-первых, сформировать множество вариантов решения для оценки и выбора, во-вторых, сформулировать цели принятия решения и, в-третьих, осуществить оценку и выбор либо одного, либо множества вариантов, наилучших (рациональных) с его точки зрения.

Одним из наиболее распространённых интерактивных многокритериальных (многопараметрических) методов оценки вариантов является метод анализа иерархий, разработанный Томасом Саати [1, 2]. Данный метод хорошо известен и широко распространён. Среди его положительных качеств можно отметить достаточную логическую простоту, возможность разделения исходной проблемы на логические подпроблемы, которые могут анализировать отдельные эксперты, компетентные в данном

направлении. Однако на всех этапах от ЛПР (или эксперта) требуется ввод достаточно большого набора точных численных оценок предпочтительности в процессе использования метода парных сравнений. Если ЛПР затруднится в назначении точной оценки, тогда метод прекращает свою работу. Одним из подходов, частично решающих данную проблему, является применение аппарата нечётких оценок [3], но это полностью не решает проблему. В частности, не решена проблема ситуации, при которой ЛПР или эксперт вообще не может численно сравнить по предпочтительности некоторое подмножество частных критериев между собой. Предлагаемый подход решает данную проблему.

2. Схема метода анализа иерархий и её модификация

Метод анализа иерархий (МАИ, Analytic Hierarchy Process) – широко известный математический инструмент системного подхода к сложным проблемам принятия решений. МАИ представляет собой интерактивный процесс численной оценки вариантов (альтернатив), который позволяет ЛПР выразить в численной форме предпочтения и оценки в соответствии со знаниями и пониманием проблемы.

2.1. МАИ как метод формирования оценок альтернатив

МАИ позволяет структурировать сложную проблему оценки альтернатив в виде иерархии целей, затем получить численные оценки альтернатив на основе системы сравнений. Каждый узел иерархии представляет один из аспектов решаемой задачи в соответствии с представлениями ЛПР. При этом для построения отдельных частей иерархии (поддеревьев) могут быть привлечены разные эксперты в соответствии с их компетенциями.

После построения иерархии выполняется процесс назначения приоритетов (численных оценок подцелей) для каждого родительского узла иерархии. Приоритет представляет собой численную оценку относительной предпочтительности каждого элемента иерархии. В классическом варианте МАИ предполагает формирование точных численных оценок приоритетов при помощи метода парных сравнений. Математический аппарат, предложенный в МАИ, обеспечивает анализ вводимых ЛПР или экспертом численных оценок на непротиворечивость и формирование на их основе итоговых оценок альтернатив.

Особенность метода заключается в необходимости ввода именно точных численных оценок приоритетов – точных значений элементов матрицы попарных сравнений. Однако на практике назначение точных значений может вызвать затруднения, и в этом случае ЛПР (эксперт) может указать качественные предпочтения в вербальной форме: «данная подцель является более предпочтительной, чем другая подцель». Это влечёт наличие неопределённости в численных значениях приоритетов и требует модификации процесса формирования итоговых оценок альтернатив.

На первом этапе решения многокритериальной задачи оценки ЛПР (эксперту) необходимо построить иерархическую структуру системы в виде дерева. На рисунке 1 приведён пример трёхуровневой иерархии критериев.

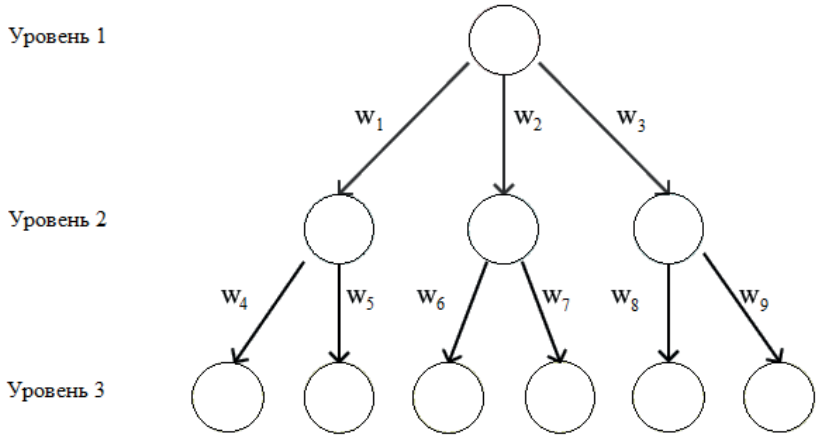


Рис. 1: Пример иерархии критериев

Здесь неотрицательные коэффициенты w_i, $i = 1, \ldots, n$, отражают относительную предпочтительность порождённых параметров для каждого родительского узла иерархии критериев. В дальнейшем будем рассматривать их как вектор коэффициентов предпочтительности, значения которого определяются его областью допустимых значений: $w = (w_1, \ldots, w_n) \in D_w$.

Ведём следующие обозначения:

- e_j – вершина дерева иерархии, $j = 1, \ldots, N$;
- N_j – количество порождённых вершин для узла j дерева иерархии, $\sum_{j=1}^{N} N_j = N$;
- $E_j = \{e_{j1}, \ldots, e_{jm_j}\}$ – множество вершин, являющихся порождёнными для узла j дерева иерархии, $j = 1, \ldots, N$;
- $L = \{l_1, \ldots, L_K\}$ – множество номеров листовых вершин дерева иерархии;
- y_l^s – вектор оценок варианта s, $s = 1, \ldots, S$, по всем листовым вершинам $l \in L$;
- P_l – путь (множество вершин) до каждой листовой вершины l_k из вершины начала иерархии.

Можно указать область допустимых значений вектора коэффициентов предпочтительности (допустимых значений вектора приоритетов):

$$D_w = \left\{ w \in R^n \mid w_i \geq w_0 > 0,\ i = 1, \ldots, n;\ \sum\nolimits_{i \in E_j}^{N_j} w_i = 1,\ j = 1, \ldots, N \right\}. \qquad (1)$$

Область D_w определяется из основного соотношения МАИ: для каждой родительской вершины сумма приоритетов порождённых поддеревьев частных критериев должна быть равна единице. Величина w_0 определяет нижнюю границу каждого приоритета и может определяться ЛПР произвольно.

Величина w_0 задаёт нижнюю границу для всех коэффициентов относительной предпочтительности. Значение коэффициента предпочтительности не может быть нулевым по смыслу метода анализа иерархий. И поскольку в дальнейшем данные коэффициенты будут определяться автоматически, то такое ограничение необходимо.

Если ЛПР может назначить точные значения коэффициентов предпочтительности, то можно считать исходную задачу оценки вариантов решённой.

2.2. Учёт качественной информации о предпочтениях ЛПР

В случае трудности или невозможности со стороны ЛПР назначить точные значения коэффициентов предпочтительности порождённых сегментов, предлагаемый метод позволяет учесть качественную информацию об их относительной предпочтительности.

В ряде работ [7–9] формулируется и обосновывается принцип, в основе которого лежит предположение о том, что вектор весовых коэффициентов w для каждого оцениваемого варианта может быть индивидуальный. Это следует из естественного предположения о том, что некоторый критерий для данного варианта является определяющим, и не является таковым для другого варианта.

Данный принцип позволяет предположить, что коэффициенты w являются неконтролируемыми факторами и могут быть определены по принципам «максимального риска» и «максимальной осторожности» [7] для каждого варианта.

При использовании принципа «максимальной осторожности» коэффициенты w^s для каждой альтернативы s, $s = 1, \ldots, S$, определяются как

$$w^s = \arg\min_{w \in D_w} Q(w, y^s) = \sum_{l \in L} \left(y_l^s \prod_{i \in P_l} w_i \right). \tag{2}$$

При использовании принципа «максимального риска» коэффициенты w^s для каждой альтернативы s, $s = 1, \ldots, S$, определяются из соотношения:

$$w^s = \arg\max_{w \in D_w} Q(w, y^s) = \sum_{l \in L} \left(y_l^s \prod_{i \in P_l} w_i \right. \tag{3}$$

Оптимизируемая функция $Q(w, y^s)$ будет строиться для каждого оцениваемого варианта s. Предполагается, что ЛПР может сформулировать качественные соотношения v_r предпочтений среди порождённых узлов каждого узла j иерархии в форме «критерий i_1 более предпочтителен, чем критерий i_2», что позволяет ввести соотношения:

$$v_r = \{e_{i_1} > e_{i_2}\} \Longleftrightarrow w_{k_1} \geq g_r w_{k_2}, \quad r = 1, \ldots, R; \tag{4}$$

где $e_{j_1}, e_{j_2} \in E_j$; w_{k_1}, w_{k_2} – соответствующие им весовые коэффициенты; $g_r \geq 1$ – индивидуальный коэффициент предпочтительности, задаваемый ЛПР.

Учитывая соотношения (4) область допустимых значений коэффициентов предпочтительности можно сформулировать следующим образом:

$$D_w = \left\{ w \in R^n \;\middle|\; \begin{array}{ll} w_i \geq w_0 > 0, & i = 1, \ldots, n; \\ \sum_{i \in E_j}^{N_j} w_i = 1, & j = 1, \ldots, N; \\ w_{k_1} \geq g_r w_{k_2}, & r = 1, \ldots, R \end{array} \right\} \tag{5}$$

Необходимо подчеркнуть, что качественная информация (4) может быть неполной, то есть ЛПР произвёл сравнение внутри одного узла j иерархии не всех $C_{N_j}^2$ частных критериев между собой.

Таким образом, область D_w, построенная как (5), позволяет учесть как требования классического метода анализа иерархий, так и индивидуальные предпочтения ЛПР внутри каждого узла иерархии. Рассмотрим подобное отношение на простом

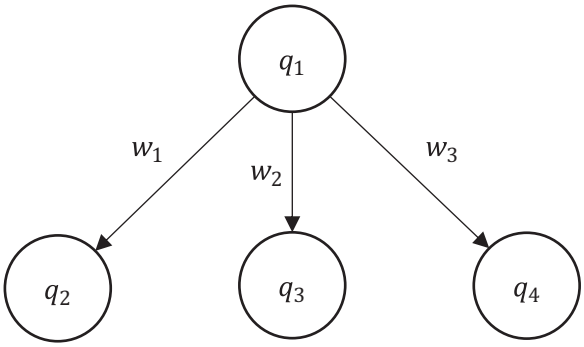


Рис. 2: Построение бинарного отношения на примере узла с тремя потомками

иллюстративном примере узла с тремя потомками, изображённого на рисунке 2. В данном случае иерархия критериев представляет собой один уровень и требуется определить итоговые оценки двух вариантов, оценки которых по каждому из листовых частных критериев заданы и показаны.

Пусть ЛПР задал величину $w_0 = 0.01$ и отношения между критериями заданы следующим образом: $v_1 = \{e_2 > e_3\}$; $v_2 = \{e_2 > e_4\}$. Для каждого из приведённых соотношений заданы коэффициенты предпочтительности: $g_1 = 1.5$; $g_2 = 2.0$.

В этом случае область допустимых значений коэффициентов приоритетов будет иметь вид:

$$D_w = \left\{ w \in R^n \;\middle|\; \begin{array}{l} w_i \geq 0.01,\ i = 1, 2, 3; \\ w_1 + w_2 + w_3 = 1; \\ w_1 \geq 1.5 w_2,\ w_1 \geq 2.0 w_3 \end{array} \right\} \tag{6}$$

2.3. Использование методов оптимизации для численного решения задачи

Изначальная задача многокритериальной оценки была сведена к парным задачам (2) и (3) оптимизации функции $Q(w, y^s)$ для каждого из оцениваемых вариантов при области допустимых значений D_w. Такая задача может быть решена классическими методами, поскольку является задачей выпуклого программирования, например, с помощью метода множителей Лагранжа.

Также задачи (2) и (3) являются задачами геометрического программирования [4, 5], которое рассматривает задачи нелинейного программирования специального типа. Подход геометрического программирования позволяет найти решение исходной задачи с помощью соответствующей двойственной задачи.

Напомним, что одночленным позиномом называется функция $u(x_1, x_2, \ldots, x_n)$ положительных переменных $x_1, x_2, \ldots, x_n$, определённая как

$$u(x_1, x_2, \ldots, x_n) = c x_1^{a_1} \ldots x_n^{a_n}. \tag{7}$$

В свою очередь, позиномом называется сумма конечного числа одночленных позиномов:

$$g(x_1, x_2, \ldots, x_n) = \sum_{k=1}^{m} c_k x_1^{a_1 k} \ldots x_n^{a_n k}. \tag{8}$$

Задача геометрического программирования: найти $\min g_0(x_1, x_2, \ldots, x_n)$ при ограничениях $x_1 > 0$, $x_2 > 0$, $\ldots$, $x_n > 0$,

$$g_i(x_1, x_2, \ldots, x_n) \leq b_i, \quad i = 1, \ldots, p.$$

Рассматриваемая оптимизационные задачи (2) и (3) могут быть решены аналитически, так как представляют собой задачу геометрического программирования.

2.4. Аналитическое решение задачи вычисления коэффициентов

Выпуклая функция непрерывная и ограниченная сверху на многограннике, достигает своего глобального максимума в одной из крайних точек. Известно, что задача максимизации выпуклой функции на ограниченном множестве в общем случае многоэкстремальна, для отыскания ее глобального максимума достаточно сравнить значения функции в крайних точках.

3. Пример решения задачи оценки вариантов

Рассмотрим действие предложенного алгоритма на примере.

Пусть составлено дерево иерархии, представленное на рисунке 3, и ЛПР смогло задать некоторые качественные оценки относительной предпочтительности для частных критериев оптимальности.

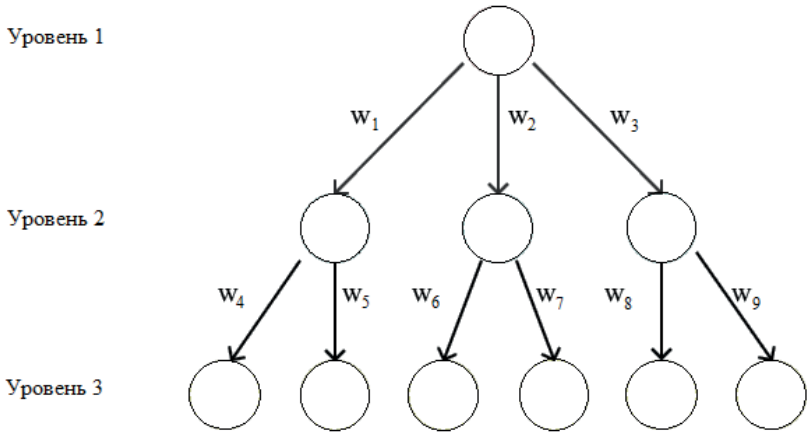


Рис. 3: Пример иерархии критериев

Пусть вербальное описание предпочтительности ЧКО задаётся в виде системы неравенств:

$$\begin{cases} w_1 \geq w_2, \\ w_3 \geq w_4, \\ w_6 \geq 3w_5. \end{cases} \tag{9}$$

Также ограничения на сумму коэффициентов задают следующие равенства:

$$\begin{cases} w_1 + w_2 = 1, \\ w_3 + w_4 = 1, \\ w_5 + w_6 = 1. \end{cases} \tag{10}$$

Перепишем уравнения и неравенства в стандартной форме:

$$\begin{cases} g_1(w) = w_2 - w_1 \leq 0, \\ g_2(w) = w_4 - w_3 \leq 0, \\ g_3(w) = 3w_5 - w_6 \leq 0; \end{cases} \tag{11}$$

$$\begin{cases} h_1(w) = w_1 + w_2 - 1, \\ h_2(w) = w_3 + w_4 - 1, \\ h_3(w) = w_5 + w_6 - 1. \end{cases} \tag{12}$$

Установим нижнюю допустимую границу значения коэффициентов $w_0 = 0,01$.

Пусть в задаче есть три альтернативы, которые предстоит оценить модифицированным методом анализа иерархий. Для каждого варианта задан вектор оценок для узлов иерархии критериев, являющихся листовыми. Обозначим эти вектора a,b,c для первого, второго и третьего вариантов соответственно. Вектора имеют размерность 4, и их элементы соотносятся с коэффициентами w_2, w_4, w_5, w_6 как с коэффициентами при листовых узлах. Для упрощения записи обозначим элементы вектора следующим образом и запишем их значения:

$$\begin{gathered} a = (a_2, a_4, a_5, a_6) = (0,1; 0,1; 0,1; 0,7); \\ b = (b_2, b_4, b_5, b_6) = (0,2; 0,2; 0,5; 0,1); \\ c = (c_2, c_4, c_5, c_6) = (0,25; 0,25; 0,25; 0,25). \end{gathered}$$

Теперь составим для каждой альтернативы оптимизируемую функцию и найдём её максимальное и минимальное значение при помощи метода ОПГ.

$$\begin{gathered} Q_1(w) = w_1 \cdot (w_4 \cdot a_4 + w_3 \cdot (w_5 \cdot a_5 + w_6 \cdot a_4)) + w_2 \cdot a_2 \\ Q_2(w) = w_1 \cdot (w_4 \cdot b_4 + w_3 \cdot (w_5 \cdot b_5 + w_6 \cdot b_4)) + w_2 \cdot b_2 \\ Q_3(w) = w_1 \cdot (w_4 \cdot c_4 + w_3 \cdot (w_5 \cdot c_5 + w_6 \cdot c_4)) + w_2 \cdot c_2 \end{gathered}$$

Результаты вычислений представлены в таблицах 1-4.

Теперь полученные интервальные оценки ЛПР может использовать при принятии окончательного решения. По данным таблицы 4 можно сказать, что второй вариант является худшим по сравнению с остальными. Максимальная и минимальная оценка третьего варианта совпадают, интервальная оценка в данном случае является константой. Самую большую максимальную оценку имеет первый вариант, но он проигрывает третьему варианту по минимальной оценке. Поэтому нельзя с точностью сказать, какой из этих двух вариантов является более оптимальным. Выбор между альтернативами остаётся за ЛПР.

Таблица 1: Значение переменных при минимальной и максимальной оценках для первой альтернативы

Переменные	Коэффициент a_i	Значение переменной min	Значение переменной max
w_0	-	0,01	0,01
w_1	-	0,5	0,99
w_2	0,1	0,5	0,01
w_3	-	0,5	0,99
w_4	0,1	0,5	0,01
w_5	0,1	0,25	0,01
w_6	0,7	0,75	0,99

Таблица 2: Значение переменных при минимальной и максимальной оценках для второй альтернативы

Переменные	Коэффициент b_i	Значение переменной min	Значение переменной max
w_0	-	0,01	0,01
w_1	-	0,5	0,5
w_2	0,2	0,5	0,5
w_3	-	0,99	0,99
w_4	0,2	0,01	0,01
w_5	0,5	0,01	0,25
w_6	0,1	0,99	0,75

Таблица 3: Значение переменных при минимальной и максимальной оценках для третьей альтернативы

Переменные	Коэффициент b_i	Значение переменной min	Значение переменной max
w_0	-	0,01	0,01
w_1	-	0,99	0,5
w_2	0,25	0,01	0,5
w_3	-	0,5	0,99
w_4	0,25	0,5	0,01
w_5	0,25	0,25	0,01
w_6	0,25	0,75	0,99

Таблица 4: Значение интервальных оценок для всех альтернатив (округление до сотых)

Функция	q_k^-	q_k^+
$Q_1(w)$	0,21	0,68
$Q_2(w)$	0,15	0,2
$Q_3(w)$	0,25	0,25

4. Заключение

Рассмотренный метод, представляющий собой модификацию схемы анализа иерархий, используется в качестве одного из подходов к оценке вариантов аналитической совокупности в системах поддержки принятия решений, рассматриваемых при изучении курса «Принятие решений на основе данных». Данный метод используется в практических и лабораторных работах и позволяет студентам получить комплексные компетенции по отдельным разделам курса «Системный анализ» (анализ и построение модели многокритериальной оценки объектов с формулированием и использованием индивидуальных предпочтений, выраженных в качественной форме), курса «Инженерия больших данных» (технические вопросы сбора, очистки и предварительной обработки больших данных), курса «Аналитика экономических данных» (построение и использование аналитических моделей) и курса «Методы оптимизации» (определение численных значений коэффициентов и параметров).

Список литературы

[1] Саати Т. Принятие решений. Метод анализа иерархий. М.: Радио и связь, 1989. — 316 с.

[2] Саати Т., Кернс К. Аналитическое планирование. Организация систем. М.: Радио и связь, 1991. — 224 с.

[3] Артамонов В.С., Лабинский А.Ю., Уткин О.В. Модификация нечёткого метода анализа иерархий // Научно-аналитический журнал «Вестник Санкт-Петербургского университета Государственной противопожарной службы МЧС России». 2016. №4.
`https://cyberleninka.ru/article/n/modifikatsiya-nechetkogo-metoda-analiza-ierarhiy`
(дата обращения: 01.05.2024).

[4] Бекишев Г.А., Кратко М.И. Элементарное введение в геометрическое программирование. Москва «Наука», главная редакция физико-математической литературы, 1980. – 144 с.

[5] Даффин Р., Питерсон Э., Зенер К. Геометрическое программирование. Издательство «Мир», Москва 1972. – 311 с.

[6] Арутюнов А. В. Лекции по выпуклому и многозначному анализу. М.: ФИЗМАТЛИТ, 2004. – 184 с.

[7] Многокритериальный выбор с учётом индивидуальных предпочтений. Д.И. Батищев, Д.Е. Шапошников. Нижний Новгород, 1994. – 98 с.

[8] Шапошников Д. Е. Применение принципа гарантированного результата для учёта качественной информации о предпочтениях при комплексной оценке качества функционирования телекоммуникационных сетей // Инженерный вестник Дона, № 4, 2014. – 24 с. / Электронный журнал:
`ivdon.ru/ru/magazine/archive/n4y2014/2574`

[9] Shaposhnikov, D., Makarova, J. (2021). Multicriteria Problem of Wireless System Design Using Tricriteria Approach with Qualitative Information on the Decision Maker's Preference. In: Balandin, D., Barkalov, K., Gergel, V., Meyerov, I. (eds) Mathematical Modeling and Supercomputer Technologies. MMST 2020. Communications in Computer and Information Science, vol 1413. Springer, Cham. https://doi.org/10.1007/978-3-030-78759-2_24

Оптимизация расчетов биохимических процессов во внутренних водоемах суши на графических ускорителях

Е.М. Гащук[1,2,3], Р.А. Ахтамьянов[3,5,6], В.А. Ломов[3,4,5], А.В. Дебольский[3,4], Д.С. Гладских[3,7], Е.В. Мортиков[3,2]

[1]Московский государственный университет им. М.В.Ломоносова, Механико-математический факультет
[2]Институт вычислительной математики им. Г.И. Марчука РАН, Москва
[3]Московский государственный университет им. М.В. Ломоносова, Научно-исследовательский вычислительный центр, Москва
[4]Институт физики атмосферы им. А.М. Обухова РАН, Москва
[5]Московский государственный университет им. М.В.Ломоносова, Географический факультет
[6]Гидрометцентр РФ, Москва
[7]Институт прикладной физики им. А.В. Гапонова-Грехова РАН, Нижний Новгород

В работе получены оценки эффективности вычислений на GPU относительно CPU для модели термогидродинамики и биохимии внутренних водоемов суши, а также для отдельных программных блоков ее компонент. Получено, что в экспериментах с грубым разрешением CPU ядро и узел оказались эффективнее одного GPU, причем основное время расчетов на GPU занимают MPI обмены и блок биохимии. По этой причине также рассмотренны результаты применения техник оптимизации вычислений и возможность ускорения за счет использования половинной точности для блока биохимии на GPU. Так как большую по времени долю расчетов занимают обмены, в работе также описаны способы ускорения выполнения MPI обменов данными, которые хранятся на графических ускорителях.

1. Введение

Озера и водохранилища являются важнейшими элементами многих природных ландшафтов, определяя их уникальные особенности, а процессы, протекающие в подобных водных объектах представляют интерес в рамках глобальных задач метеорологии и климатологии, таких, как изучение климата Земли и оценки его изменений. Так, в работе [14] предложена трехмерная модель биогеохимических процессов внутреннего водоема, объединенная с моделью термогидродинамики и позволяющая получить полные трехмерные поля гидрологических и биохимических характеристик озер и водохранилищ. Представление моделей в виде единого программного

комплекса имеет ряд преимуществ с существующими в мире аналогами: так, например, в работе [11] для оценок выбросов метана используется комплекс из трех моделей, не объединенных в совместную систему. В качестве физической модели [11] используется одномерная, и серьезной проблемой является количественная оценка вертикальной турбулентной диффузии, что приводит к неточностям в определении пространственной и временной изменчивости концентрации метана и оценках его выброса в атмосферу. В статье [3] предложена узкоспециализированная трехмерная модель, предназначенная для изучения выбросов углекислого газа из субтропического водоема. Численная модель, используемая в настоящей работы, является достаточно универсальной как с точки зрения описываемых веществ, так и с точки зрения исследуемых объектов.

Модели озер и водохранилищ позволяют оценить ряд климатически значимых характеристик, включая эмиссию парниковых газов, в связи с чем их необходимо включать в модели Земной системы. При этом важно отметить, что трехмерное моделирование является единственным способом получить полное распределение термических и биохимических характеристик, а также может использоваться как инструмент для калибровки и уточнения моделей меньшей пространственной детализации, используемых в МЗС. Использование модели [14] позволяет получить распределение метана (газа, важного с точки зрения влияния на климат ввиду высокого парникового потенциала) по поверхности водоема (см. рис. 1), что может быть использовано при подготовке измерительных кампаний на водоемах.

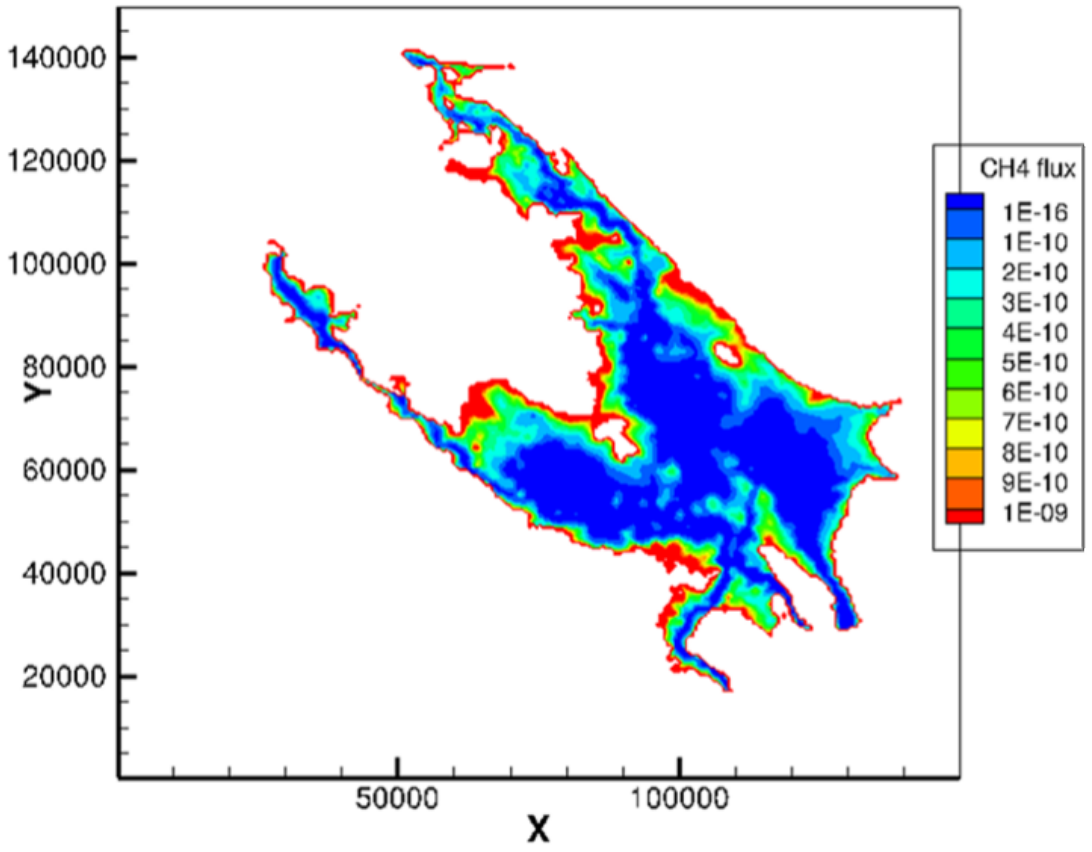


Рис. 1: Мгновенное распределение потока метана (в моль/(м^{2}с)) по поверхности Рыбинского водохранилища.

На Земле насчитывается около 5 миллионов озер, и проведение подобных численных экспериментов для такого количества объектов является вычислительно сложной задачей, в связи с чем необходимо использование суперкомпьютеров. Для того, чтобы ускорить проведение численных экспериментов, расширить возможности моделирования и снизить энергопотребление HPC-систем (High-performance computing), необходимо повышать производительность программного кода за счет написания эффективных численных моделей и алгоритмов. Можно разрабатывать новые численные методы и алгоритмы с меньшей сложностью, которые используют меньше памяти и требуют меньшего количества обращений к памяти. Однако суперкомпьютерные системы непрерывно развиваются, поэтому другой возможный подход – использование архитектуры суперкомпьютеров, в частности массивно-параллельных архитектур, таких как, например, графические ускорители (GPU – Graphics Processing Unit). Пиковая производительность новейших архитектур GPU в несколько раз превышает производительность современных центральных процессоров (CPU – Central Processing Unit), и что более важно, тенденция такова, что со временем разрыв в производительности становится все больше [10, 12].

Для многих задач численные модели могут быть во много раз ускорены за счет использования GPU. В работе [1] показано, что реализация на базе GPU обеспечивает более чем 150-кратное ускорение по сравнению с CPU в LES (Large Eddy Simulation) модели переноса и рассеивания загрязняющих веществ. В статье [13] представлена новая версия модели океана POM (Princeton Ocean Model) с поддержкой вычислений на GPU. Эта версия демонстрирует производительность на сервере с четырьмя GPU, сравнимую с кластером из 408 ядер CPU, что соответствует почти семикратному снижению энергопотребления.

Кроме того, повышение скорости вычислений можно добиться за счет понижения точности вычислений вплоть до половинной (FP16), которая поддерживается на аппаратном уровне на современных архитектурах GPU. Возможность ускорения вычислений за счет использования арифметики с пониженной точностью представляет интерес для реализации моделей прогноза погоды и климата. В работе [2] показано, что нет необходимости использовать двойную точность в задачах декадного и ансамблевого прогнозов. При этом половинная точность оказывается достаточной для расчета всех компонент декадного прогноза, за исключением средних глобальных характеристик. В статье [9] также установлено, что использование двойной точности для всех компонент модели климата, скорее всего, избыточно: одинарной точности более чем достаточно, а расчеты в половинной точности допустимы для отдельных компонент климатической модели. Значительная часть вычислений в модели деятельного слоя суши [4] может быть выполнена с использованием смешанной точности: простая модель теплопереноса в почве и модель деятельного слоя суши допускают такой подход при условии, что переменные текущего состояния хранятся и обновляются в повышенной точности.

Основной целью данной работы является разработка эффективных алгоритмов численного моделирования биохимических процессов во внутренних водоемах суши на графических ускорителях.

2. Численная реализация

В трехмерной численной модели используется осредненная по Рейнольдсу система уравнений термо-гидродинамики в приближении Буссинеска и гидростатики [15, 14].

Уравнения переноса, диффузии и реакций веществ имеют вид:

$$\frac{\partial C}{\partial t} + \frac{\partial u_i C}{\partial x_i} = (\delta_{i1} + \delta_{i2}) \frac{\partial}{\partial x_i}(\lambda_b + \chi_b) \frac{\partial C}{\partial x_i} + \delta_{i3} \frac{\partial}{\partial x_i}(K_b + \chi_b) \frac{\partial C}{\partial x_i} + R_C,$$

где C – концентрация рассматриваемого вещества, K_b, χ_b коэффициенты турбулентной и молекулярной диффузии, λ_b – коэффициент горизонтальной вязкости, R_C – описание того, сколько молекул вещества C было добавлено или извлечено из раствора в результате реакции.

Для интегрирования по времени уравнений переноса биохимических примесей используется явный метод (за исключением вертикальной диффузии, для которой используется полунеявный метод Кранка–Николсона) и схема расщепления для разнесения расчета тенденций, связанных с реакциями веществ, и динамических тенденций. Реализация адвективного переноса примесей основана на численных схемах, рассмотренных в статье [5].

3. Постановка эксперимента

Для тестирования реализации модели биохимии на GPU (для реализации на GPU было использовано расширение языка программирования C CUDA API), использовалась следующая конфигурация численного эксперимента. Рассматривалось небольшое озеро Куйваярви (Финляндия), имеющее размеры 100×1000 метров. Отметим, что данная постановка позволяет верифицировать результаты моделирования также и по имеющимся многолетним измерениям [6, 8]. На дне задан фиксированный поток метана: $3 \cdot 10^{-7}$ моль/(м^{2}с). Поток кислорода на донной поверхности рассчитывается из логарифмического слоя, где полагается, что концентрация кислорода в донных отложениях равна 0 моль м$^{-3}$. Потоки метана и кислорода на водной поверхности рассчитывается из газообмена с атмосферой. В начальных условиях для растворенных газов и температуры заданы вертикальные распределения согласно данным натурных измерений [6, 8]. Данные метеорологического форсинга (температура и влажность воздуха, атмосферное давление, потоки коротковолновой и длинноволновой радиации, компоненты скорости ветра), характеризующие изменения условий на верхней границе в течение всего времени расчета, выбирались также согласно наблюдениям в работах [6, 8]. Для описания газообмена использована так называемая “модель обновления поверхности” [7], учитывающая турбулентные процессы, которые влияют на перемешивание около раздела воздух-вода.

4. Численные результаты

На рис. 2 показаны численные результаты распределения хлорофилл-а (**слева**) и O_2 (**справа**) по глубине в зависимости от времени интегрирования.

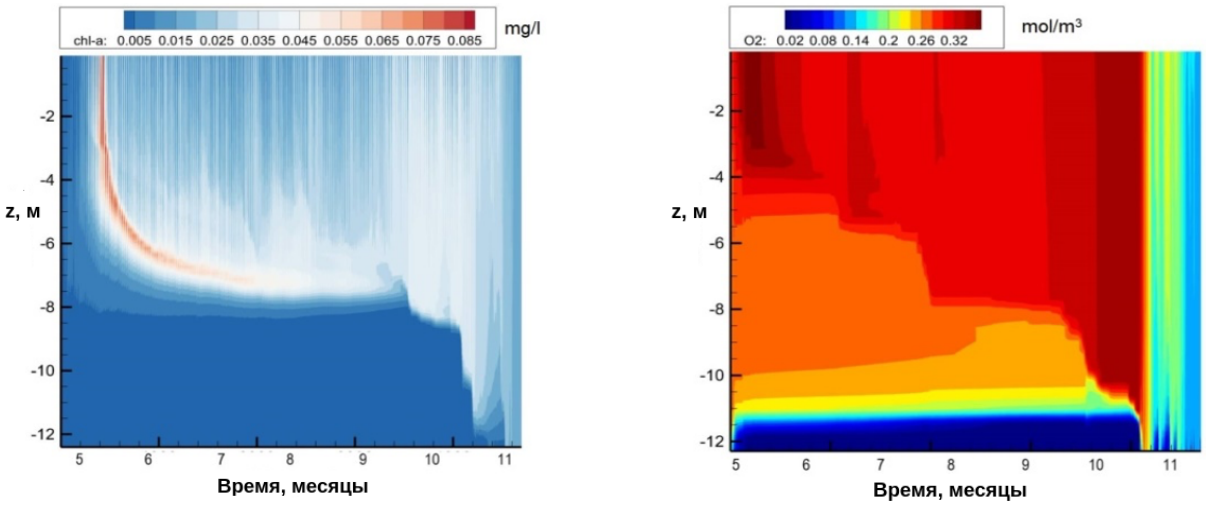


Рис. 2

5. Оценка эффективности проведения вычислений

На рисунке 3 показаны результаты сравнения времени выполнения базовой реализации переноса на одном ядре CPU Intel Haswell-EP E5-2697v3 и на GPU V100 для озера Куйваярви в зависимости от размера расчетной области (количество вертикальных уровней во всех тестах равняется 64). Во всех тестах, за исключением расчетной сетки с количеством ячеек равным $8 \times 32 \times 64$, выполнение расчетов модели на GPU оказалось эффективнее выполнения на CPU (Рис. 3 **(a)**). Ускорение реализации на GPU растет до размера расчетной области $128 \times 512 \times 64$, и максимальное достигнутое ускорение ≈ 16 раз на сетке данного размера. Касательно отдельных блоков модели (Рис. 3 **(b)**), ускорение возрастает с размером сетки для подмодулей, реа-

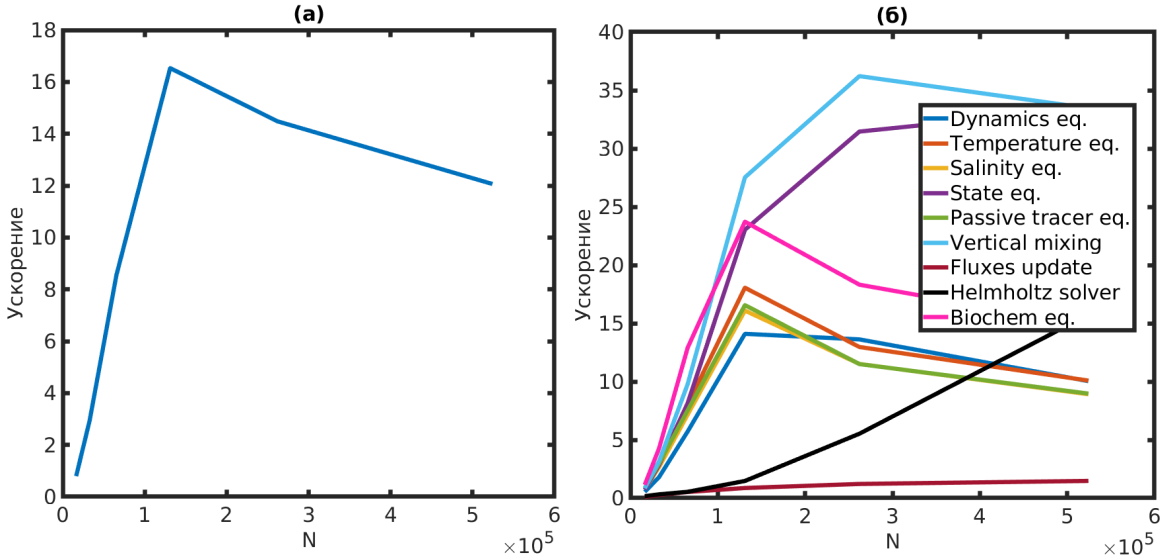


Рис. 3: Ускорение расчетов на GPU реализации относительно CPU для всей модели **(a)** и отдельных блоков модели **(b)**: блоки интегрирования по времени уравнений динамики, термодинамики, уравнения переноса солености, состояния, переноса примесей, блок вертикального перемешивания, блок расчета потоков из атмосферы и донной поверхности, решения уравнения Гельмгольца, блок биохимии.

лизующих интегрирование по времени уравнений динамики, решение разностного уравнения Гельмгольца, блока вертикального перемешивания, уравнения состояния, расчета потоков из атмосферы и донной поверхности , в то время как ускорение для остальных растет до размера сетки $32 \times 128 \times 64$.

6. Заключение

В результате проделанной работы реализована модель термо-гидродинамики и биохимии внутренних водоемов суши на графических ускорителях, и проведено сравнение ее вычислительной эффективности с CPU реализацией.

Было получено, что для 1 MPI-процесса во всех проведенных тестах, за исключением теста на грубой сетке, расчеты модели на GPU проводились быстрее, чем на CPU. Максимальное достигнутое ускорение – ≈ 16 раз.

Целью дальнейшей работы является выявление наиболее вычислительно сложных компонент модели и разработка вычислительно-эффективных алгоритмов при расчетах на графических ускорителях. Будет исследована возможность ускорения алгоритмов численного решения уравнения переноса на GPU за счет понижения точности вплоть до половинной. Также планируется внедрение технологий IPC (Inter-Process Communication) и NCCL (NVIDIA Collective Communication Library) обмена данными на GPU напрямую между графическими ускорителями без использования оперативной памяти CPU.

Программный код модели доступен на GitLab сервере НИВЦ МГУ по запросу `http://tesla.parallel.ru`.

7. Благодарности

Работа выполнена при поддержке гранта РНФ № 21-71-30003. Тесты производительности проводились с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова.

Список литературы

[1] Bieringer Paul E., Piña Aaron J., Lorenzetti David M., Jonker Harmen J.J., Sohn Michael D., Annunzio Andrew J., Fry Richard N. A Graphics Processing Unit (GPU) Approach to Large Eddy Simulation (LES) for Transport and Contaminant Dispersion // Atmosphere. 2021. 12, 7.

[2] Chantry Matthew, Thornes Tobias, Palmer Tim, Düben Peter. Scale-Selective Precision for Weather and Climate Forecasting // Monthly Weather Review. 2019. 147, 2. 645–655.

[3] Chen Zhonghan, Huang Ping, Zhang Zhou. Interaction between carbon dioxide emissions and eutrophication in a drinking water reservoir: A three-dimensional ecological modeling approach // Science of the total environment. 2019. 663. 369–379.

[4] Dawson A, Düben Peter D., MacLeod David A., et al. Reliable low precision simulations in land surface models // Climate Dynamics. 2018. 33, 4. pages 2657–2666.

[5] Gaschuk E.M., Ezhkova A.A., Onoprienko V.A., Debolskiy A.V., Mortikov E.V. Passive Tracer Transport in Ocean Modeling: Implementation on GPUs, Efficiency and Optimizations // Lobachevskii Journal of Mathematics. VIII 2023. 44, 8. 3040–3058.

[6] Heiskanen Jouni J., Mammarella Ivan, Ojala Anne, Stepanenko Victor, Erkkilä Kukka-Maaria, Miettinen Heli, Sandström Heidi, Eugster Werner, Leppäranta Matti, Järvinen Heikki, Vesala Timo, Nordbo Annika. Effects of water clarity on lake stratification and lake-atmosphere heat exchange // Journal of Geophysical Research: Atmospheres. 2015. 120, 15. 7412–7428.

[7] MacIntyre S., Jonsson A., Jansson M., Aberg J., Turney D.E., Miller S.D. Buoyancy flux, turbulence, and the gas transfer coefficient in a stratified lake // Geophysical Research Letters. 2010. 37, 24.

[8] Mammarella I., Nordbo A., Rannik Ü., Haapanala S., Levula J., Laakso H., Ojala A., Peltola O., Heiskanen J., Pumpanen J., Vesala T. Carbon dioxide and energy fluxes over a small boreal lake in Southern Finland // Journal of Geophysical Research: Biogeosciences. 2015. 120, 7. 1296–1314.

[9] Paxton E. Adam, Chantry Matthew, Klöwer Milan, Saffin Leo, Palmer Tim. Climate Modeling in Low Precision: Effects of Both Deterministic and Stochastic Rounding // Journal of Climate. 2022. 35, 4. 1215–1229.

[10] Pi Puig Martín, De Giusti Laura, Naiouf Marcelo. Are GPUs Non-Green Computing Devices? // Journal of Computer Science and Technology. 10 2018. 18. e17.

[11] Schmid Martin, Ostrovsky Ilia, McGinnis Daniel F. Role of gas ebullition in the methane budget of a deep subtropical lake: What can we learn from process-based modeling? // Limnology and Oceanography. 2017. 62, 6. 2674–2698.

[12] Sun Yifan, Agostini Nicolas Bohm, Dong Shi, Kaeli David R. Summarizing CPU and GPU Design Trends with Product Data // ArXiv. 2019. abs/1911.11313.

[13] Xu S., Huang X., Oey L.-Y., Xu F., Fu H., Zhang Y., Yang G. POM.gpu-v1.0: a GPU-based Princeton Ocean Model // Geoscientific Model Development. 2015. 8, 9. 2815–2827.

[14] Гладских Д.С. Исследование термогидродинамических и биогеохимических процессов во внутреннем водоеме на основе модифицированных моделей турбулентного переноса. 2022.

[15] Гладских Д.С. Степаненко В.М., Мортиков Е.В. О влиянии горизонтальных размеров внутренних водоемов на толщину верхнего перемешанного слоя // Водные ресурсы. 2021. 48, 2. 155–163.

Параллельные алгоритмы решения задачи негильотинного размещения на основе точной модели[1]

А.А. Андрианова

Казанский (Приволжский) федеральный университет

В работе предлагаются приемы построения параллельных алгоритмов для решения задачи негильотинного размещения набора прямоугольников на полуполосе на основе точной модели, которая представляется задачей частично булевого линейного программирования большой размерности.

Ключевые слова: негильотинное размещение набора прямоугольников, задача частично булевого линейного программирования, параллельные алгоритмы

1. Введение

Задачи негильотинного размещения набора прямоугольников на полуполосе является одной из задач двумерной упаковки и представляет интерес в промышленности, логистике и других областях принятия решений [1]. В зависимости от условий, которые накладываются на исходные данные задачи, в этом классе можно выделить задачи упаковки на лист заданного размера, на полуполосу (ограничение ставится только по одному размеру), задачи с фиксированной ориентацией прямоугольников или с возможностью их поворота, с гильотинным размещением набора прямоугольников, требующих проводить разрезы материалы от края до края, и с негильотинным размещением, определяющим более сложную конфигурацию разрезов [2].

Практически все формулировки задач двумерной упаковки являются NP-трудными задачами, поэтому основной акцент в исследованиях базировался на применении метаэвристических подходов построения алгоритмов решения задачи. Некоторые из этих подходов базируются на идеях генетических алгоритмов, алгоритмов случайного поиска, другие имитируют многошаговый процесс принятия решений человеком при размещении каждого прямоугольника последовательно на целевом материале. Точным алгоритмам же в силу их сложности не уделяется много внимания, так как существующие точные модели в основном будут связаны с высокими вычислительными затратами и будут малоприменимыми на практике.

Известны как линейные, так и нелинейные (и более того, невыпукные) точные модели для решения задачи негильотинного размещения набора прямоугольников на полуполосе [3]. В работе внимание уделено одной из точных моделей, в которой задача компактного размещения набора прямоугольных деталей на полуполосе представляется задачей частично булевого линейного программирования большой

[1]Работа выполнена за счет средств Программы стратегического академического лидерства Казанского (Приволжского) федерального университета (“ПРИОРИТЕТ-2030”)

размерности [4]. Существуют точные методы решения такой задачи, однако с увеличением количества деталей вычислительная сложность ее решения растет экспоненциально.

В данной работе предлагаются подходы, которые можно назвать комбинированными, использующими эвристические приемы сведения к задачам меньшей размерности, трудоемкость решения которых с помощью точной модели будет существенно ниже исходной. Одной из ключевых особенностей данных приемов является возможность распараллеливания, что позволит в ряде случаев получить дополнительный эффект в уменьшении вычислительной сложности получения решения.

2. Точная модель компактного размещения набора прямоугольников на полуполосе

В [4] была представлена следующая точная модель размещения набора прямоугольников на полуполосе заданной ширины.

Пусть даны набор n прямоугольников с размерами $a_i \times b_i$ $(a_i \geq b_i)$, $i = 1, \ldots, n$, и полуполоса с шириной B $(B \geq \max_{i=1,\ldots,n} b_i)$. Требуется определить минимальную длину листа, необходимого и достаточного для размещения данного набора прямоугольников, и план данного размещения, т.е. координаты всех прямоугольников при данном размещении. Задачи, в которых в качестве основной цели будет использоваться минимизация площади, занимаемой набором прямоугольников, называют задачами компактного размещения. Рассматриваемая задача также принадлежит к этому классу.

2.1. Модель частично булевого линейного программирования

Пусть полуполоса размещается на координатной плоскости таким образом, что левый нижний угол полуполосы имеет координаты (0,0). Будем рассматривать варианты размещения прямоугольника с ориентацией либо вдоль листа (сторона прямоугольника длиной a_i расположена параллельно оси OX), либо с ориентацией поперек листа (сторона прямоугольника длиной a_i расположена параллельно оси OY).

Обозначим через (x_i, y_i) координаты левого нижнего угла прямоугольника с номером i, а через z_i – ее ориентацию на полуполосе – $z_i = 0$, если i-ый прямоугольник ориентирован вдоль листа, $z_i = 1$ в противном случае.

Для формирования условий попарного непересечения набора прямоугольников вводятся дополнительные булевые переменные для каждой пары прямоугольников:

1) s_{ij} – переменная, определяющая предшествование i-ого прямоугольника j-ому $(s_{ij} = 0)$ или предшествование j-ого прямоугольника i-ому $(s_{ij} = 1)$;

2) t_{ij} - переменная, определяющая по горизонтали $(t_{ij} = 0)$ или по вертикали $(t_{ij} = 1)$ производится расхождение местоположения прямоугольников.

В силу прямой связи значений переменных при смене порядке индексов ($s_{ij} = 1 - s_{ji}$, $t_{ij} = t_{ji}$) при рассмотрении пар прямоугольников рассматриваются только индексы, удовлетворяющие условиям $i = 1, \ldots, n-1$, $j = i+1, \ldots, n$.

Задача линейного частично булевого программирования для размещения набора прямоугольников на полуполосе будет иметь следующий вид:

$$A \to \min$$
$$x_i + (b_i - a_i)z_i \le A - a_i, \quad i = 1, \dots, n,$$
$$y_i + (a_i - b_i)z_i \le B - b_i, \quad i = 1, \dots, n,$$
$$-x_i + x_j - (b_i - a_i)z_i + \bar{A}t_{ij} + \bar{A}s_{ij} \ge a_i, \quad i = 1, \dots, n-1,\ j = i+1, \dots, n,$$
$$x_i - x_j - (b_j - a_j)z_j + \bar{A}t_{ij} - \bar{A}s_{ij} \ge a_j - \bar{A}, \quad i = 1, \dots, n-1,\ j = i+1, \dots, n,$$
$$-y_i + y_j - (a_i - b_i)z_i - \bar{B}t_{ij} + \bar{B}s_{ij} \ge b_i - \bar{B}, \quad i = 1, \dots, n-1,\ j = i+1, \dots, n,$$
$$y_i - y_j - (a_j - b_j)z_j - \bar{B}t_{ij} - \bar{B}s_{ij} \ge b_i - 2\bar{B}, \quad i = 1, \dots, n-1,\ j = i+1, \dots, n,$$
$$x_i \ge 0, \quad i = 1, \dots, n,$$
$$y_i \ge 0, \quad i = 1, \dots, n,$$
$$z_i \in \{0, 1\}, \quad i = 1, \dots, n,$$
$$s_{ij} \in \{0, 1\}, \quad i = 1, \dots, n-1,\ j = i+1, \dots, n,$$
$$t_{ij} \in 0, 1, \quad i = 1, \dots, n-1,\ j = i+1, \dots, n.$$

В модели константы ds выбраны так, чтобы $\bar{B} \ge B$, $\bar{A}$ – величина, не меньшая оптимальной длины полуполосы, достаточной для размещения заданного набора прямоугольников, например, в можно взять для расчетов $\bar{B} = B$, $\bar{A} = \sum_{i=1}^{n} a_i$.

Таким образом, количество переменных в модели равно $n^2 + 2n + 1$, где n^2 переменных являются булевыми. Количество ограничений, исключающих ограничения на знак переменных и их булевый тип, $2n^2$. Например, при относительно небольшом количестве прямоугольников $n = 10$ количество переменных задачи будет равно 121, а количество ограничений 200, что является уже достаточно большой задачей. Наличие 100 булевых переменных делает решение задачи сложнее, так как для точного решения задач булевого программирования требуется организация алгоритмов полного перебора.

2.2. Алгоритм решения точной модели булевого программирования

Для решения задачи булевого программирования можно использовать метод Лэнд и Дойг [5] – универсальный метод решения задач частично целочисленного линейного программирования с двусторонними ограничениями на переменные. Задача частично булевого линейного программирования является частным случаем таких задач.

Алгоритм метода Лэнд и Дойг для поставленной модели можно сформулировать обобщенно следующим образом. Будем обозначать через X – вектор всех переменных задачи.

Подготовительный этап. Будем формировать множество отложенных задач $S = \varnothing$. Определим начальное значение рекорда целевой функции $F^* = \bar{A}$. Формулируем задачу линейного программирования, заменяя ограничения булевости в задаче на ограничения типа $0 \le z_i \le 1$, $0 \le s_{ij} \le 1$, $0 \le t_{ij} \le 1$. Помещаем сформулированную задачу в множество S.

Шаг 1. Если $S = \varnothing$, останов. Последний рекорд будет являться оптимальным решением задачи.

Шаг 2. Из множества отложенных задач S извлечем очередную задачу, назовем ее текущей.

Шаг 3. Если текущая задача не имеет решений (область допустимых решений пуста), переходим к шагу 1.

Шаг 4. Вычислим X^* - оптимальное решение текущей задачи. Если значение целевой функции в оптимальном решении не меньше рекордного, переходим к шагу 1.

Шаг 5. Если все соответствующие переменные X^* удовлетворят ограничениям булевости исходной задачи, то примем X^* в качестве очередного рекорда (происходит обновление рекорда) и осуществляем переход к шагу 1.

Шаг 6. Выберем индекс переменной, которая должна быть в исходной задаче булевой, j^* такой, что $X^*_{J^*} \notin \{0, 1\}$. В множество отложенных задач S поместим две новые задачи, полученные из текущей фиксацией переменной в оба допустимых значения: первая задача отличается от текущей фиксацией $X^*_{J^*} = 0$, вторая - $X^*_{J^*} = 1$. Осуществляем переход к шагу 1.

Данный алгоритм укладывается в схему метода ветвей и границ. Строится виртуальное дерево задач. Задача линейного программирования, ослабляющая задачу частично булевого линейного программирования, ставится для каждого узла этого дерева. Шаг 6 интерпретируется как ветвление узла, полученное путем фиксации выбранной переменной. Таким образом, ветвление, если оно будет осуществляться, будет создавать два родительных поддерева с непересекающимися множествами решений – поддерево с фиксированным значением переменной ветвления в 0 и поддерево с фиксированным значением переменной ветвления в 1.

Оптимальное значение длины полуполосы для задачи, решаемой в узле в ситуации ослабленной задачи, может интерпретироваться как нижняя оценка оптимального значения для всего поддерева, корнем которого является узел, соответствующий этой задаче. Поэтому можно использовать это значение как оценку неперспективности всего поддерева, если оно окажется больше текущего найденного рекордного значения и, соответственно, отказа от ветвления данного узла.

Принцип ветвления гарантирует обеспечение полного перебора всех вариантов фиксации булевых переменных по необходимости. Учет нижней оценки узла и его сравнение с рекордным позволяет сократить полный перебор и не рассматривать поддеревья, на которых гарантировано не может быть получено оптимума. Также ветвление не требуется, если оптимальное решение ослабленной задачи без ограничения на булевость переменных даст в ответе значения 0 или 1 во всех булевых переменных. Такое решение можно сравнивать с рекордом на шаге 5 и, возможно, обновлять рекорд, если получено лучшее значение длины полуполосы.

Как и любой алгоритм метода ветвей и границ, метод решения задачи компактного негильотинного размещения прямоугольников на полуполосе может иметь реализации, использующие параллельные вычисления, что позволит если не уменьшить вычислительную сложность алгоритма, то уменьшить время его работы.

3. Параллельные алгоритмы решения задачи негильотинного размещения набора прямоугольников

3.1. Параллельный алгоритм на основе нескольких потоков решения задач узлов дерева

Первый из алгоритмов, использующих параллельные вычисления, базируется на использовании естественного для всех алгоритмов метода ветвей и границ свойства, заключающегося в том, что разные поддеревья одного и того же узла могут быть обработаны независимо и параллельно.

Параллельная модификация метода Лэнд и Дойг может быть сформулирована следующим образом.

Подготовительный этап. Будем формировать множество отложенных задач $S = \varnothing$. Определим начальное значение рекорда целевой функции $F^* = \bar{A}$. Формулируем задачу линейного программирования, заменяя ограничения булевости в задаче на ограничения типа $0 \leq z_i \leq 1$, $0 \leq s_{ij} \leq 1$, $0 \leq t_{ij} \leq 1$. Помещаем сформулированную задачу в множество S. Определяем число m – количество рабочих потоков, которые будут осуществлять вычисления. Определим количество занятых потоков $k = 0$.

Каждый поток действует по следующей последовательности шагов.

Шаг 1. Если $S = \varnothing$ и $k = 0$, останов потока.

Шаг 2. Если $S = \varnothing$ и $k \neq 0$, поток переходит в режим ожидания на некоторый случайный интервал времени t (в пределах нескольких миллисекунд). Переход к шагу 1.

Шаг 3. Если $S \neq \varnothing$, из множества отложенных задач S извлечем очередную задачу, назовем ее текущей, устанавливаем флаг занятости потока – увеличиваем количество занятых потоков $k = k + 1$.

Шаг 4. Если текущая задача не имеет решений (область допустимых решений пуста), снимаем влаг занятости потока – уменьшаем количество занятых потоков $k = k - 1$, переходим к шагу 1.

Шаг 5. Вычислим X^* - оптимальное решение текущей задачи. Если значение целевой функции в оптимальном решении не меньше рекордного, снимаем флаг занятости потока – уменьшаем количество занятых потоков $k = k - 1$, переходим к шагу 1.

Шаг 6. Если все соответствующие переменные X^* удовлетворят ограничениям булевости исходной задачи, то примем X^* в качестве очередного рекорда (происходит обновление рекорда), снимаем флаг занятости потока – уменьшаем количество занятых потоков $k = k - 1$ и осуществляем переход к шагу 1.

Шаг 7. Выберем индекс переменной, которая должна быть в исходной задаче булевой, j^* такой, что $X^*_{J^*} \notin \{0, 1\}$. В множество отложенных задач S поместим две новые задачи, полученные из текущей фиксацией переменной в оба допустимых значения: первая задача отличается от текущей фиксацией $X^*_{j^*} = 0$, вторая - $X^*_{j^*} = 1$. Уменьшаем количество занятых потоков $k = k - 1$. Осуществляем переход к шагу 1.

Алгоритм заканчивает работу, если завершили работу все рабочие потоки. Следует обратить внимание, что условие шагов 1 и 2 гарантируют окончание работы

любого рабочего потока. Действительно, если выполняются условия шага 1, то это значит, что нет больше задач на выполнение, причем так как количество занятых потоков равно 0, то такие задачи уже точно не появятся. Поэтому поток может остановить свою работу для решения данной задачи.

Если выполняются условия шага 2, то, так как есть еще занятые потоки, новые задачи могут быть ими поставлены и тогда список отложенных задач расширится, это может обнаружиться через некоторый период ожидания. Поэтому поток на шаге 2 уходит в режим ожидания новых задач.

Данный алгоритм предполагает совместное использование отдельных переменных - списка отложенных задач, текущего рекорда и количества занятых потоков. При программной реализации алгоритма возможно применение любой технологии блокирования указанных переменных при их изменении и чтении каждым рабочим потоком.

3.2. Параллельный алгоритм на основе нескольких потоков и фиксации сразу нескольких переменных в узле ветвления

Главной проблемой высокой трудоемкости решения точной модели, рассмотренной в разделе 2.1, является большое количество булевых переменных и то, что в узле виртуального дерева фиксируется лишь одна из них, оставляя на ранних этапах задачу линейного программирования сложной по количеству переменных. Одна из идей улучшения показателей эффективности базируется на нахождении подходов, которые бы позволили фиксировать в узле сразу несколько булевых переменных и, таким образом, значительно сокращать глубину дерева и количество переменных в задачах линейного программирования в узлах поддеревьев.

Одна из идей фиксации сразу нескольких переменных сразу заключается в использовании природы переменных $\{s_{ij}\}$ – переменных, которые определяют порядок следования прямоугольников. Эти переменные можно зафиксировать путем генерации перестановки прямоугольников. Очевидно, что имея перестановку $\pi = (i_1, i_2, \ldots, i_n)$, можно однозначно определить значения переменных $\{s_{ij}\}$. Таким образом, будут зафиксированы чуть менее половины булевых переменных сразу. Остальные переменные должны определяться на основе построения виртуального дерева по методу Лэнд и Лойг.

Параллельный алгоритм, реализующий данную идею, выделяет отдельный поток для генерации перестановок по любому известному алгоритму. Этот поток сохраняет любую перестановку в множество сгенерированных перестановок P. Поток должен сигнализировать о том, что он находится в работе. Поэтому он также учитывается как занятый поток, генерирующий задачи. Алгоритм действия остальных рабочих потоков по сравнению с алгоритмом раздела 3.2 изменен в части определения момента останова.

Подготовительный этап. Будем формировать множество сгенерированных перестановок $P = \varnothing$, и множество отложенных задач $S = \varnothing$. Определим начальное значение рекорда целевой функции $F^* = \bar{A}$. Определим количество занятых потоков $k = 1$, учтя таким образом запуск потока генерации перестановок. После генерации последней перестановки поток сигнализирует о завершении своей работы, уменьшив количество занятых потоков ($k = k - 1$).

Каждый рабочий поток решения задачи линейного программирования действует по следующей последовательности шагов.

Шаг 1. Если $P = \varnothing$, $S = \varnothing$ и $k = 0$, останов потока.

Шаг 2. Если $P = \varnothing$, $S = \varnothing$ и $k \neq \varnothing$, поток переходит в режим ожидания на некоторый случайный интервал времени t (в пределах нескольких миллисекунд). Переход к шагу 1.

Шаг 3. Если $P \neq \varnothing$, устанавливаем флаг занятости потока – увеличиваем количество занятых потоков ($k = k + 1$), извлекаем перестановку $\pi \in P$, фиксируем по ней набор переменных $\{s_{ij}\}$, формулируем задачу линейного программирования, заменяя ограничения булевости в точной модели на ограничения типа $0 \leq z_i \leq 1$, $0 \leq t_{ij} \leq 1$. Помещаем сформулированную задачу в множество S. Снимаем флаг занятости потока ($k = k - 1$). Переходим к шагу 1.

Шаг 4. Если $S \neq \varnothing$, из множества отложенных задач S извлечем очередную задачу, назовем ее текущей, устанавливаем флаг занятости потока – увеличиваем количество занятых потоков $k = k + 1$.

Шаг 5. Если текущая задача не имеет решений (область допустимых решений пуста), снимаем влаг занятости потока – уменьшаем количество занятых потоков $k = k - 1$, переходим к шагу 1.

Шаг 6. Вычислим X^* - оптимальное решение текущей задачи. Если значение целевой функции в оптимальном решении не меньше рекордного, снимаем флаг занятости потока – уменьшаем количество занятых потоков $k = k - 1$, переходим к шагу 1.

Шаг 7. Если все соответствующие переменные X^* удовлетворят ограничениям булевости исходной задачи, то примем X^* в качестве очередного рекорда (происходит обновление рекорда), снимаем флаг занятости потока – уменьшаем количество занятых потоков $k = k - 1$ и осуществляем переход к шагу 1.

Шаг 8. Выберем индекс переменной, которая должна быть в исходной задаче булевой, j^* такой, что $X^*_{J^*} \notin \{0, 1\}$. В множество отложенных задач S поместим две новые задачи, полученные из текущей фиксацией переменной в оба допустимых значения: первая задача отличается от текущей фиксацией $X^*_{J^*} = 0$, вторая – $X^*_{J^*} = 1$. Уменьшаем количество занятых потоков $k = k - 1$. Осуществляем переход к шагу 1.

В данном алгоритме рабочий поток занимается двумя типами задач – шаг 3 алгоритма запускает процесс решения методом Лэнд и Дойг для зафиксированного набора переменных $\{s_{ij}\}$, шаги 4–8 предназначены для решения задач фиксации переменных из наборов $\{t_{ij}\}$, $\{z_i\}$.

4. Сравнительный анализ последовательного и параллельных алгоритмов решения задачи негильотинного размещения прямоугольников

Эксперименты по решению задачи компактного размещения набора прямоугольников на полуполосе согласно модели, описанной в разделе 2.1, по алгоритму Лэнд и Дойг, описанному в разделе 2.2, проводились на ряде сгенерированных модельных задачах для обеспечения сравнения с работой других алгоритмов. Основным критерием сравнения различных вариантов алгоритма стало количество задач линейного

программирования, решенных в узлах виртуального дерева перебора. Так как размер одной задачи линейного программирования будет сильно увеличиваться с увеличением количества прямоугольников в наборе, в качестве критерия сравнения также будет использоваться время, затраченное на решение всей задачи размещения. Для эксперимента была использована среда Google Colaboratory.

4.1. Последовательный алгоритм метода ветвей и границ

Показатели, полученные на нескольких модельных задачах разного размера для последовательной реализации алгоритма Лэнд и Дойг представлены в таблице 1. В первом столбце таблицы указывается количество прямоугольников конкретной модельной задачи, общее количество задач, которые были решены в узлах дерева для этой задачи, представлено во втором столбце, и далее в третьем столбце указано время решения задачи.

Таблица 1: Показатели решения задачи последовательным алгоритмом

Номер задачи	Количество прямоугольников	Количество задач в узлах дерева	Время решения задачи (секунды)
1	3	73	0,112
2	4	971	1,552
3	4	253	0,347
4	5	7 853	13,231
5	5	33 049	54,312
6	5	70 565	114,448
7	6	260 403	501,035
8	6	534 719	1015,216
9	6	3 487 209	6413,042
10	7	203 663	672,034
11	7	2 056 290	5912,954

В данной таблице наглядно видно, что даже для небольших по размеру задач будет строиться большое дерево, требующее решения нескольких миллионов задач линейного программирования. Также можно наблюдать, что для задач одной и той же размерности наблюдается большая разница в трудоемкости вычислений, в частности, за счет более эффективно сработавших нижних оценок поддеревьев. Например, для задач с шестью прямоугольниками наблюдается разница в трудоемкости решения примерно в 13 раз. Для показателей, которые были получены для задач большей размерности, например, $n = 10$, разброс количества решенных задач в узлах дерева от 751 до 8 258 729 задач, т.е. видно, что трудоемкость задач, обладающих одними и теми же размерными характеристиками, сильно отличается, что связано с тем, в какой части дерева решений будет получено решение задачи.

Так, в ходе эксперимента, которых проводился в [4] и его продолжения на применении последовательного алгоритма были определены распределения остаточной трудоемкости решения задачи (доли узлов дерева от общего количества просмотрен-

ных узлов, которые просматриваются уже после получения оптимального решения), представленные в таблице 2.

Таблица 2: Доли задач с определенной остаточной трудоемкостью

Диапазон остаточной трудоемкости (проценты)	Доля задач в эксперименте (проценты)
$< 20\%$	$\sim 15\%$
$[20\%; 50\%)$	$\sim 13\%$
$[50\%; 80\%)$	$\sim 22\%$
$[80\%; 100\%]$	$\sim 50\%$

Как видно из таблицы, даже после получения оптимального решения дерево продолжает строится и приходится решать еще достаточно много вспомогательных задач. Это, в том числе, говорит о том, что оптимальное решение может находиться в любом поддереве, не только в левом, и, в том числе, требует просмотра большой доли узлов дерева. Этот факт говорит о возможной эффективности процедуры распараллеливания алгоритма.

Еще один вывод, который можно сделать из указанных данных эксперимента, заключается в том, что время решения задач согласно точной модели для количества прямоугольников, менее 6, можно считать приемлемым и достаточно быстрым. Так, для задач с $n = 5$, в которых присутствуют 25 булевых переменных время задачи было в пределах одной-двух минут, в то время как при $n = 6$ время решения задач варьировалось от 10 минут до 2 часов. Далее наблюдается также экспоненциальный рост времени решения задачи до получения оптимального решения.

4.2. Анализ применения параллельных версий алгоритма метода ветвей и границ

Применение параллельной версии алгоритма метода ветвей и границ, построенного на основе метода Лэнд и Дойг, описанного в разделе 3.1, разнилось в зависимости от принципов выбора очередной задачи из множества отложенных задач S. Алгоритм предполагает как случайный выбор задачи из множества, так и организацию этой коллекции по принципам стека или очереди, которые организуют обход узлов дерева по принципу обхода «в глубину» или «в ширину».

Эксперимент при решении модельных задач, показатели которых указаны в таблице 1, показал в среднем уменьшение времени работы алгоритма по сравнению с последовательной версией алгоритма Лэнд и Дойг в среднем на 23% при количестве рабочих потоков $m = 4$ при случайном выборе задачи из множества отложенных задач. Тем не менее, показатель количества задач, решенных в узлах виртуального дерева, построенного в ходе работы алгоритма, значительно не уменьшался за исключением небольшого количества случаев, а в ряде случаев даже увеличилось. Это связано с тем, что в большинстве своем рекорд получается в начальных 20% трудоемкости просмотра дерева решений, что чаще всего означает расположение в левом поддереве корневого узла дерева. Это объясняется тем, что за счет параллельной реализации решение задач правого поддерева начинается тогда, когда рекорд еще недостаточно хороший. Это не дает возможности сработать нижней оценке узла для

признания поддерева бесперспективным для получения оптимума и приходится решать достаточно много задач, которые при последовательном варианте алгоритма не решались вовсе.

Тем не менее, эффективность решения отдельных классов задач, требующих небольшой доли просмотра дерева (оптимальное решение находится в левом поддереве и узлы правого поддерева блокируются за счет хорошей нижней оценки узлов) или, напротив, просмотра преобладающей доли узлов дерева из-за малого количества срабатывания нижней границы узла для определения его бесперспективности, удалось повысить за счет организации множества отложенных задач по принципу стека для первого типа задач (в среднем, время решения таких задач уменьшилось на 46%), и за счет организации множества отложенных задач по принципу очереди для второго типа задач (в среднем, наблюдалось уменьшение времени решения задач на 67%). В параллельной версии алгоритма, рассмотренной в разделе 3.2, изменена процедура фиксации большинства булевых переменных. Процедура фиксации переменных $\{s_{ij}\}$ стала заметно проще по сравнению с последовательным решением задач линейного программирования большой размерности, поэтому уже первые эксперименты показывают эффективность такого подхода, значительно уменьшая время решения задачи (в большинстве модельных задач более 42% экономии времени). Более того, так как количество булевых переменных при таком подходе уменьшилось почти в 2 раза, уменьшается и глубина дерева решений и размерность вспомогательной задачи, что на отдельных задачах большей размерности (тесты проводились на примерах до $n = 20$) удалось получить решение за приемлемое время (до 3 часов).

Тем не менее, этот подход уже нельзя считать точным. Дело в том, что множество перестановок не определяет все варианты наборов значений переменных $\{s_{ij}\}$. Например, может быть допустимым такая ситуация: $s_{12} = 0$, $s_{23} = 0$, $s_{13} = 1$. Это связано с тем, что непересечение первого и второго прямоугольника, и первого и третьего может быть учтено по различным координатам. Поэтому транзитивность отношений между прямоугольниками на основе генерации перестановки оставляет такие ситуации без рассмотрения. Тем не менее, полученные решения по модельным задачам совпадали с точными, и такой подход показал, что алгоритм можно рассматривать как одну из перспективных эвристик.

В целом, алгоритм с одновременной фиксацией нескольких переменных можно модифицировать применением других принципов выбора и фиксации переменных. Например, можно разбивать множество переменных на два подмножества – прямоугольники, разделяемые по горизонтали, и прямоугольники, которые разбиваются по вертикали, тем самым фиксировать набор переменных $\{t_{ij}\}$. Для каждого такого множества можно генерировать свою перестановку и, тем самым, для построения дерева остается только зафиксировать набор переменных $\{z_i\}$, количество которых совпадает с количеством прямоугольников в наборе, тем самым не будет сильно значительным и строящееся дерево решения будет строиться и решаться очень быстро.

Фактически, второй параллельный алгоритм можно обобщить и сформулировать как обобщенный многоуровневый алгоритм, где на каждом уровне могут определяться значения булевых переменных по разным принципам, а в конечном счете при получении задачи частично булевого программирования в размерах, для которых наблюдается приемлемое время решения точной модели, будет применять метод Лэнд и

Дойг. Поскольку уровни такого алгоритма могут рассматриваться как независимые задачи, поставляющие данные для следующего уровня, процедура дальнейшего распараллеливания за счет решения потоками разных типов задач на разных уровнях представляется перспективной для развития.

Список литературы

[1] Мухачева Э.А. Рациональный раскрой промышленных материалов. Применение в АСУ. Новосибирск: Наука, 1987. 274 с.

[2] Oliveira, Ó., Gamboa, D. and Silva, E. An introduction to the two-dimensional rectangular cutting and packing problem // Intl. Trans. in Op. Res. 2023. No. 30 P. 3238-3266.
https://doi.org/10.1111/itor.13236

[3] Manuel Iori, Vinícius L. de Lima, Silvano Martello, Flávio K. Miyazawa, Michele Monaci, Exact solution techniques for two-dimensional cutting and packing // European Journal of Operational Research. 2021. Vol. 289, No. 2, P. 399-415.
https://doi.org/10.1016/j.ejor.2020.06.050

[4] Андрианова А.А., Мухтарова Т.М., Фазылов В.Р. Модели негильотинного размещения набора прямоугольных деталей на листе и полуполосе // Учен. зап. Казан. ун-та. Сер. Физ.матем. науки. 2013. Т. 155, кн. 2. С. 5–18.

[5] Land A.H., Doig A.G. An automatic method of solving discrete programming problems // Econometrica. 1960. Vol. 28, No. 3. P. 497-520.

Параллельный алгоритм развертывания фазы для обработки данных дистанционного зондирования Земли

Е.Н. Акимова[1,2], В.Е. Мисилов[1,2], А.В. Сосновский[2]

[1]Институт математики и механики им. Н.Н. Красовского УрО РАН,
[2]Уральский федеральный университет

Современные радиолокационные системы дистанционного зондирования Земли из космоса имеют сверхвысокое пространственное разрешение и широкую полосу захвата, что приводит к необходимости обрабатывать интерферометрические изображения из сотен миллионов элементов. Обработка таких изображений может достигать нескольких суток. Разработка параллельных алгоритмов и их реализация на многопроцессорных вычислительных системах является актуальной задачей. В работе представлено построение и исследование параллельного алгоритма интерферометрической развертки фазы в задачах радиолокационного дистанционного зондирования Земли. Для развертывания фазы используется эффективный метод выравнивания встречного вихревого поля, обладающий квазилинейной вычислительной сложностью. Для реализации метода предложен параллельный алгоритм для многопроцессорных распределенных систем с многоядерными центральными процессорами и разработана программа с использованием векторизации и технологии OpenMP. На вычислительной системе с двумя 18-ядерными процессорами достигнуто ускорение в 33 раза.

Ключевые слова: дистанционное зондирование Земли, интерферометрический радар с синтезированной апертурой, развертывание фазы, параллельные вычисления, SIMD, OpenMP, MPI

1. Введение

Космическая радиолокационная интерферометрия (РСА-интерферометрия) широко используется для получения цифровых моделей рельефа и карт изменения рельефа в различных отраслях науки, техники и экономики: картографии, геодезии, в экологическом мониторинге, в геологических, гидрологических и гляциологических исследованиях, а также для контроля состояния транспортных коммуникаций. Интерферометрическая обработка включает несколько этапов преобразования радиолокационной информации [1, 2].

Основными задачами интерферометрической обработки радиолокационных данных дистацнионного зондирования Земли из космоса являются подавление фазового шума и развертывание фазы. Развертывание представляет собой процесс преобразования картины двумерной относительной фазы, принимающей значения только в

интервале $[-\pi, \pi)$, в абсолютную фазу, диапазон принимаемых значений для которой не ограничен. Абсолютная фаза содержит информацию о рельефе поверхности и его малых изменениях за период между радиолокационными наблюдениями. Развертывание фазы до сих пор остается самым проблемным этапом интерферометрической обработки, т.к. подобная задача в принципе не имеет точного аналитического решения и решается приближенно. Обработка интерферограмм занимает продолжительное время и может достигать нескольких суток, что затрудняет использование данных для оперативного решения задач мониторинга. Для развертывания фазы разработаны алгоритмы, многие из которых имеют высокую вычислительную сложность и требуют использования приближений и упрощений, приводящих к снижению точности [4].

В работах авторов [3, 4] предложен метод развертывания фазы, основанный на построении и выравнивании встречного вихревого поля фазы, обладающий высокой точностью и квазилинейной вычислительной сложностью.

В настоящей работе на основе метода развертывания фазы интерферограмм космических радиолокаторов предлагается параллельный алгоритм и его реализация на многоядерных процессорах. Исследована эффективность алгоритма в зависимости размера исходных данных и количества параллельных потоков OpenMP с использованием векторизации.

В разделе 2 настоящей работы описан алгоритм развертывания фазы. Приведена блок-схема алгоритма. В разделе 3 приводится параллельная реализация алгоритма развертывания фазы. В разделе 4 приведены результаты вычислительных экспериментов по оценке эффективности алгоритма при реализации на многоядерных процессорах.

2. Метод развертывания интерферометрической фазы

Предложенный метод использует непосредственное устранение разрывов фазы интерферограммы искусственными разрывами (элементарными вихрями фазы) встречного направления [5, 6]. Совокупность искусственных разрывов формирует встречное вихревое поле фазы. Далее осуществляется рекурсивное выравнивание этого поля и его адаптивная фильтрация.

Для математического описания интерферометрической фазы используется модель в виде комплексной интерферограммы — функции комплексной переменной $\dot{I}(z)$, составленной следующим образом:

$$\dot{I}(z) = \mathrm{e}^{j\Delta\phi_{m,n}}\Big|_{z=m+j\cdot n}, \tag{1}$$

где j — мнимая единица, $\Delta\phi_{m,n}$ — интерферограмма, m, n — координаты элементов (пикселей) интерферограммы, $z = m + j \cdot n$ — дискретный аргумент комплексной интерферограммы. Разрывы фазы (элементарные вихри) описываются в виде функции комплексной переменной $\dot{J}(z)$

$$\dot{J}^{\pm}(z) = \exp\left(\pm j \cdot q \cdot \arg\{z - z_{(0/p)}\}\right), \tag{2}$$

где $z_{(0/p)} = m_{(0/p)} + j \cdot n_{(0/p)}$ — координаты точки разрыва с положительным (ноль) или отрицательным (полюс) знаком.

Для восстановления непрерывности фазы в каждую точку разрыва интерферограммы q-го порядка и положительного знака помещается элементарный вихрь $J^{-}(z)$ q-го порядка, а в точку разрыва с отрицательным знаком — $J^{+}(z)$ q-го порядка. Совокупность элементарных вихрей образует встречное вихревое поле фазы интерферограммы $\dot{C}_{*}(z)$:

$$\dot{C}_{*}(z) = \exp\left(j \cdot \arg\left\{\frac{(z-z_{p1})^{q_{p1}} \cdots (z-z_{p\mu})^{q_{p\mu}}}{(z-z_{01})^{q_{01}} \cdots (z-z_{0\nu})^{q_{0\nu}}}\right\}\right), \tag{3}$$

где μ, ν — количество точек разрыва интерферограммы положительного и отрицательного знака, соответственно. Умножение комплексной интерферограммы $\dot{I}(z)$ на встречное вихревое поле $\dot{C}_{*}(z)$ должно приводить к исчезновению точек разрыва фазы и формированию безразрывной относительной фазы, после чего окончательное развертывание можно было бы произвести путем суммирования разностей фаз соседних элементов интерферограммы (свернутых в интервал однозначности) вдоль любой траектории на интерферограмме, получив, таким образом, безразрывную абсолютную фазу $\widetilde{\Psi}_{m,n}$ (рис. 1):

$$\widetilde{\Psi}_{m,n} = \Upsilon_{r}\left\{\arg\{\dot{I}(z) \cdot \dot{C}_{*}(z)\}\big|_{z=m+jn}\right\}, \tag{4}$$

где $\Upsilon_{r}\{\}$ — символическое обозначение развертывания по правилу суммирования разностей соседних элементов интерферограммы.

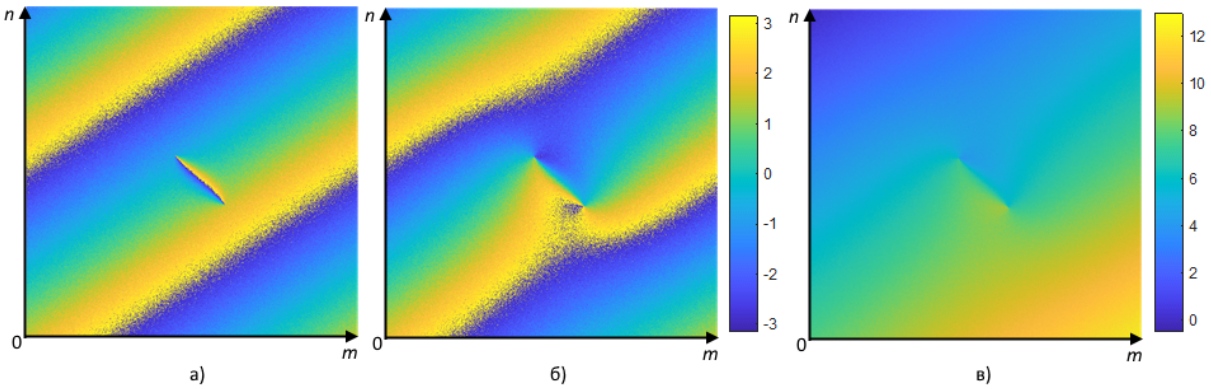


Рис. 1: Развертывание интерферограммы с помощью встречного вихревого поля фазы: а) исходная интерферограмма; б) фаза произведения исходной интерферограммы и встречного вихревого поля; в) развернутая интерферограмма

В силу дискретного характера интерферограммы и квантования значений фазы после умножения возможно появление новых точек разрыва или их перемещение на новые позиции. Для их компенсации процедура осуществляется итерационно до тех пор, пока все разрывы фазы не будут полностью устранены. Итоговое встречное вихревое поле фазы $C_{**}(z)$ будет представлять собой произведение встречных вихревых полей, полученных на отдельных итерациях алгоритма:

$$\dot{C}_{**}(z) = \dot{C}_{\mathrm{I}}(z) \cdot \dot{C}_{\mathrm{II}}(z) \cdot \dot{C}_{\mathrm{III}}(z) \cdot \ldots \cdot \dot{C}_{K_{\mathrm{iters}}}(z), \tag{5}$$

где $\dot{C}_{\mathrm{I}}(z), \dot{C}_{\mathrm{II}}(z), \ldots$ — встречные вихревые поля, полученные на первой, второй и последующих итерациях алгоритма, K_{iters} — число итераций. Далее абсолютная фаза формируется по правилу суммирования разностей соседних элементов.

Алгоритм 1 (последовательный) на основе предложенного метода состоит из следующих этапов.

0. Инициализация встречного вихревого поля $C(z)$ нулевым массивом действительных чисел с двойной точностью размером, равным размеру интерферограммы ($M_I \times N_I$).
1. Предварительное вычисление элементарного вихря $J(z) = \arg\{z\}$ на поле размером $2M_I \times 2N_I$ элементов.
2. Построение функции локализации точек разрыва с учетом построенного встречного вихревого поля и определение количества точек разрыва $\mu + \nu$ с координатами $z_{(0/p)i}$ и весами $q_{(0/p)i}$. Если количество точек разрыва равно нулю, то переходим к шагу 6.
3. Для i-й точки разрыва производится считывание фрагмента элементарного вихря размером $M_I \times N_I$ с центром в точке с координатами $z_{(0/p)i}$.
4. Умножение всех элементов фрагмента элементарного вихря на вес разрыва $q_{(0/p)i}$ и прибавление этого фрагмента к встречному вихревому полю $C(z)$, если $q < 0$, или вычитание, если $q > 0$. Если $i < \mu + \nu$, то переходим к следующей точке $i + 1$ и затем к п. 3.
5. Переходим к п. 2.
6. Преобразование полученного поля в комплексный вид: $C(z) \to \dot{C}(z) = \exp\{j \cdot C(z)\}$.
7. Вычисление безразрывной абсолютной фазы по формуле 4 и завершение работы.

На рис. 2 представлена блок-схема алгоритма. Заметим, что алгоритм позволяет легко распараллеливать вычисления.

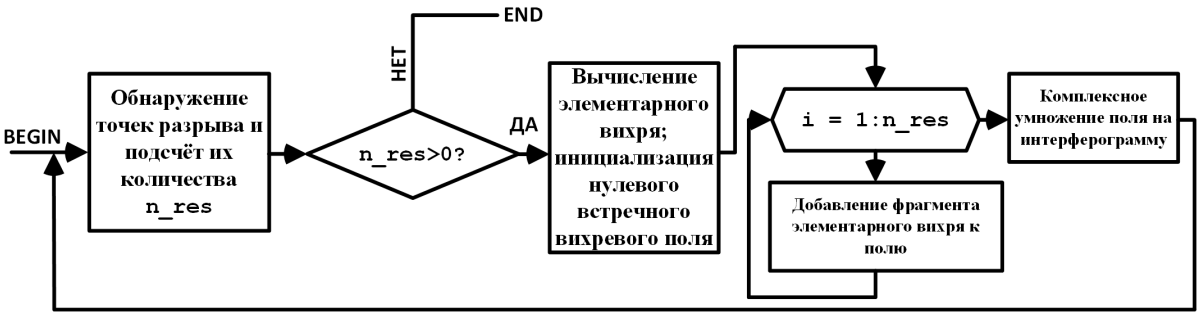


Рис. 2: Блок-схема алгоритма развертывания фазы

3. Построение параллельного алгоритма

Эффективная параллельная реализация алгоритма развертывания фазы основана на геометрической декомпозиции интерферограммы.

На верхнем уровне интерферограмма разбивается на горизонтальные полосы, хранящиеся и обрабатывающиеся в отдельных MPI-процессах. Внутри каждой подобласти декомпозиция осуществляется путем использования распределения работы циклов по потокам с помощью технологии OpenMP.

На данном этапе исследований разработана эффективная параллельная реализация алгоритма развертывания фазы в рамках одного узла с многоядерным процессором.

Алгоритм 2 (параллельный многопоточный), реализованный на языке C++ с использованием технологии OpenMP, имеет следующий вид (*отличия от Алгоритма 1 выделены курсивом*):

0. Инициализация встречного вихревого поля $C(z)$ нулевым массивом действительных чисел с двойной точностью размером, равным размеру интерферограммы ($M_I \times N_I$).
1. Предварительное вычисление элементарного вихря $J(z) = \arg\{z\}$ на поле размером $2M_I \times 2N_I$ элементов.
2. Построение функции локализации точек разрыва с учетом построенного встречного вихревого поля. Определение количества точек разрыва $\mu + \nu$, их координат $z_{(0/p)i}$ и *весов со знаком* $q_{(0/p)i}$.
   Данный этап реализуется как два вложенных цикла, проходящих по всем элементам интерферограммы. Используется конструкция `#pragma omp for` *для распределения работы по OpenMP-потокам. Найденные точки разрыва, их координаты и веса сохраняются в экземпляры контейнера типа vector для каждого потока. После завершения этапа экземпляры контейнера объединяются в один.*
   Если количество точек разрыва равно нулю, то переход к шагу 6.
3. Для каждой i-й точки разрыва *производится предвычисление указателя на массив, соответствующий фрагменту элементарного вихря размером* $M_I \times N_I$ *с центром в точке с координатами* $z_{(0/p)i}$. *Указатели сохраняются в общий контейнер, построенный на шаге 3.*
4. Для каждой точки разрыва умножить все элементы фрагмента элементарного вихря на вес разрыва $q_{(0/p)i}$ и прибавить полученный фрагмент вихря ко встречному вихревому полю $C(z)$.
   Данный этап реализуется как три вложенных цикла: проход по всем элементам встречного поля и по всем точкам разрыва. Для эффективного распределения работы по потокам и использования векторных инструкций процессора используется следующий порядок циклов:
   (a) *Внешний цикл прохода по строкам поля распределен по потокам с использованием конструкции* `#pragma omp for`;
   (b) *Проход по контейнеру, хранящему точки разрыва, и считывание весов и указателей на фрагмент элементарного вихря*;
   (c) *Проход по элементам строки поля и прибавление элемента вихря, умноженного на вес точки разрыва векторизован с помощью конструкции* `#pragma omp simd`.
   Для балансировки распределения работы по потокам внешний цикл (a) использует режим планирования `schedule(dynamic,1)`. *Таким образом, потоки обрабатывают изображение порциями по одной строке. После того как поток обработал очередную строку, планировщик «выдает» ему следующую из еще не обработанных.*
   Для обеспечения эффективного использования кеша внутренний цикл (c) разбит на блоки так, чтобы фрагменты обрабатываемой строки поля и вих-

ря помещались в кеш первого уровня. Эксперименты показали, что оптимальный размер блока составляет 1024 элемента. Размер блока составляет $2 \times 1024 \times 8 = 16$ Кбайт, поэтому блок помещается в L1-кэш (32 КБ у используемых процессоров).

5. *Переход к шагу 2.*
6. Преобразование полученного поля в комплексный вид: $C(z) \to \dot{C}(z) = \exp\{j \cdot C(z)\}$.
   Данный этап реализуется как два вложенных цикла, проходящих по всем элементам изображения с распределением работы по OpenMP-потокам.
7. Вычисление безразрывной абсолютной фазы по формуле 4 и завершение работы.

Для реализации параллельного алгоритма построения встречного вихревого поля на системах с разделяемой памятью с использованием технологии MPI [7] предлагается

Алгоритм 3.

1. Для каждого MPI-процесса по очереди:
   (a) Процесс находит точки разрыва внутри своей подобласти (см. шаг 2 Алгоритма 2).
   (b) Процесс рассылает свой экземпляр контейнера с найденными точками разрыва всем другим процессам.
   (c) Другие процессы получают контейнер с точками разрыва.
   (d) Каждый процесс строит фрагмент элементарного вихря, соответствующий взаимному расположению подобластей процесса-получателя и процесса-отправителя.
   (e) Строится встречное вихревое поле (см. шаги 4–5 Алгоритма 2).
   (f) Переход на шаг (a) для следующего процесса.
2. Суммирование встречных вихревых полей.
3. Пересылка граничных строк между процессами. Проверка элемента (m, n) на наличие разрыва фазы требует значения четырех элементов: (m, n), $(m+1, n)$, $(m+1, n+1)$, $(m, n+1)$. Для поиска разрывов в крайней нижней строке подобласти интерферограммы требуются значения элементов крайней верхней строки ниже расположенной подобласти (перекрытие шириной 1 элемент, см. рис. 3).

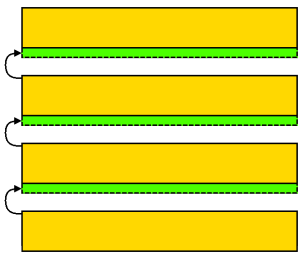


Рис. 3: Декомпозиция интерферограммы и схема передачи граничных строк между подобластями.

4. Переход к шагу 1. Если на очередной итерации ни один процесс не нашел ни одной точки разрыва в своей подобласти, то алгоритм завершен.

4. Вычислительные эксперименты и исследование кода

Эксперименты по исследованию эффективности параллельного **Алгоритма 2** проводились на двух системах:

- Рабочая станция с процессором Intel Core i9-12900K с восемью мощными ядрами (P-core) и восемью энергоэффективными ядрами (E-core);
- Узел суперкомпьютера «Уран» с двумя восемнадцатиядерными процессорами Xeon Gold 6254.

Программа компилировалась на рабочей станции компилятором Intel C++ Compiler 2024 с использованием набора векторных инструкций AVX2 (`/QxCORE-AVX2`), на суперкомпьютере «Уран» — Intel C++ Compiler Classic 19.1 с использованием инструкций AVX512 (`/QxCORE-AVX-512 /Qopt-zmm-usage=high`), для исследования использовались средства Intel Vtune Profiler и Intel Advisor [8].

Для тестов использовался набор модельных интерферограмм с размерами 1024×1024, 2048×2048, 4096×4096, 8192×8192 элементов с плотностью точек разрыва 25%.

Таблица 1 содержит времена счета тестовой задачи на процессоре Core i9-12900K с различным числом OpenMP-потоков и различной конфигурацией процессорных ядер, исполняющих потоки. Здесь «C» означает P-ядро, «C HT» — P-ядро с использованием гиперпоточности (2 потока на ядро) и «c» — E-ядро.

В таблице 1 также приведены коэффициенты ускорения $S_n = T_1/T_n$ и параллельной эффективности $E_n = S_n/n$ (только для P-ядер), где n — число задействованных процессорных ядер, T_n — соответствующее время счета. Кроме того, в таблице приведены показатели производительности программы в Гфлопс/с, измеренные с помощью инструмента Intel Advisor.

Таблица 1: Времена счета для тестовой задачи с размером интерферограммы 2048×2048 на процессоре Core i9-12900K

Число потоков n	Конфигурация ядер	Время счета T_n, с	Ускорение $S_n = T_1/T_n$	Эффективность $E_n = S_n/n$	Производительность (реальная/пиковая), Гфлоп/с
1	1C	390	—	—	
2	2C	205	1.90	0.95	
4	4C	98	3.98	0.99	
8	8C	49	7.96	0.99	260/600
16	8C HT	46	8.50	—	
16	8C+8c	37	10.54	—	
24	8C HT + 8c	35	11.14	—	360/860

При использовании только однородных P-ядер наблюдается линейный рост производительности. E-ядра обладают более слабой производительностью, как и аппаратная гиперпоточность P-ядер. Тем не менее, их использование увеличивает суммарную производительность процессора в 1.4 раза.

Таблица 2 содержит времена счета тестовой задачи на узле суперкомпьютера «Уран» с процессорами Xeon Gold 6254, а также коэффициенты ускорения и параллельной эффективности.

Наблюдается практически линейный рост производительности.

Таблица 2: Времена счета для тестовой задачи с размером интерферограммы 2048×2048 на процессорах Xeon Gold 6254

Число потоков n	Конфигурация ядер	Время счета T_n, с	Ускорение $S_n = T_1/T_n$	Эффективность $E_n = S_n/n$
1	1C	730	—	—
2	2C	365	2	1
4	4C	189	3.86	0.96
8	8C	92	7.93	0.99
12	12C	61	11.97	0.99
16	16C	47	15.53	0.97
18	18C AVX-512	42	17.38	0.96
36	2 x 18C	22	33.18	0.92
18	18C AVX2	60	—	—

Отметим, что хотя данный процессор поддерживает гиперпоточность, использование более одного потока на физическое ядро не дало прироста производительности, поэтому эти результаты не приведены в таблице. Последняя строка таблицы показывает время счета для программы, скомпилированной с использованием только векторных инструкций AVX2. Такой вариант дает в 1.4 раза меньшую производительность, чем AVX-512.

Таблица 3 содержит времена счета для серии тестовых задач на процессоре Core i9-12900K. Наблюдается ожидаемый рост времени счета при увеличении размера интерферограммы (в 4 раза больше элементов $\times$ в 4 раза больше точек разрыва). Ожидается, что обработка реальной интерферограммы размером 10973×11001 элементов потребует более 10 часов. Таким образом, дальнейшее увеличение производительности требует использования либо нескольких вычислительных узлов, либо графических процессоров.

Таблица 3: Времена счета для серии тестовых задач на процессоре i9-12900K

Размер интерферограммы	Время счета	Фактор роста времени (ожидаемый — 16)
1024x1024	2.6 с	—
2048x2048	35 с	15.21
4096x4096	9.7 мин.	16.57
8192x8192	2 ч. 42 мин.	16.84
Реальные данные 10973x11001	10 ч. (оценочное)	—

5. Заключение

Разработан параллельный алгоритм решения задачи развертывания интерферометрической фазы для интерферограмм космических радиолокаторов дистанционного зондирования Земли с синтезированной апертурой антенны. Алгоритм реализован в виде параллельной программы для систем с многоядерными центральными процессорами с использованием гибридной технологии — векторизации и OpenMP. Параллельный алгоритм имеет высокую эффективность распараллеливания. На 16-ядерном процессоре достигнуто ускорение в 11 раз. На узле с двумя 18-ядерными процессорами достигнуто ускорение в 33 раза. Планируется реализация алгоритма для многопроцессорных систем с использованием MPI.

Список литературы

[1] Joughin I. R., Li F. K., Madsen S. N., Rodrigues E., Goldstein R. M., et al. Synthetic Aperture Radar Interferometry // IEEE Proc. 2000. Vol. 88. No. 3. P. 33–82.

[2] Hanssen R. F. Radar interferometry. Data interpretation and error analysis. Dordrecht; Boston: Kluwer academic publishers. 2002. 308 p.

[3] Sosnovsky A. V., Kobernichenko V. G. An InSAR phase unwrapping algorithm with the phase discontinuity compensation // CEUR Workshop Proceedings. 2017. Vol. 2005. P. 127–136.

[4] Martyshko P. S., Akimova E. N., Sosnovsky A. V., Kobernichenko V. G. An Algorithm for Solving the Problem of Phase Unwrapping in Remote Sensing Radars and Its Implementation on Multicore Processors // Mathematics. 2024, Vol. 12. No. 5. 727.

[5] Aoki T., Sotomaru T., Ozawa T., Komiyama T., Miyamoto Y., Takeda M. Two-dimensional phase unwrapping by direct elimination of rotational vector fields from phase gradients obtained by heterodyne techniques // Opt. Rev. 1998. No. 5. P. 374–379.

[6] Tomioka S., Heshmat S., Miyamoto N., Nishiyama S. Phase unwrapping for noisy phase maps using rotational compensator with virtual singular points // Appl. Opt. 2010. No. 49. P. 4735–4745.

[7] Лаборатория параллельных информационных технологий НИВЦ МГУ. MPI: The Message Passing Interface.
`https://parallel.ru/tech/tech_dev/mpi.html`
(дата обращения: 14.04.2024).

[8] Intel Corporation. 64 and IA-32 Architectures Optimization Reference Manual.
`https://www.intel.com/content/www/us/en/content-details/671488/intel-64-and-ia-32-architectures-optimization-reference-manual.html`
(дата обращения: 14.04.2024).

Применение параллельных вычислений при реализации метода выборки переходных поверхностей

С.В. Полуян[1], Н.М. Ершов[2]

[1]Государственный университет «Дубна»,
[2]Московский государственный университет имени М.В. Ломоносова

Численная оценка константы скорости реакции является важной задачей в области биоинформатики, поскольку константа предоставляет информацию о кинетике связывания компонент белкового комплекса. Один из подходов к выполнению данной оценки заключается в применении метода выборки переходных поверхностей, в основе которого лежит симуляция переходов между различными состояниями моделируемой системы по нескольким траекториям. Каждое состояние и переход характеризуются оценками энергии взаимодействия между компонентами белкового комплекса. В настоящей работе описаны схемы применения параллельных вычислений при построении множества траекторий. Приводятся принципы применения технологии MPI и библиотеки Intel oneTBB, а также демонстрируются результаты различных экспериментов. Целью исследования является определение наиболее эффективного инструмента параллельного программирования при построении множества траекторий в методе выборки переходных поверхностей в используемых вычислительных системах.

Ключевые слова: метод выборки переходных поверхностей, белок-белковые взаимодействия, константа скорости реакции, MPI, Intel oneTBB.

1. Введение

В настоящей работе рассматриваются белковые комплексы вида белок-белок, где в роли двух компонент комплекса выступают белки, являющиеся макромолекулами, образованными из полипетидных цепей. Процесс образования комплекса включает в себя один основной этап – связывание: процесс образования устойчивого комплекса при нековалентном взаимодействии друг с другом в растворе компонент комплекса, а также соответствующее этой связи пространственное положение компонентов. При этом возможно рассматривать компоненты без каких-либо структурных изменений относительно свободного несвязного состояния в виде «твёрдых» тел [1], которые совершают в растворителе только поступательные и вращательные движения. Такое упрощение рассматривается в настоящем исследовании. При этом для описания межмолекулярных взаимодействий и получения численной оценки энергии взаимодействия достаточно использовать оценочную функцию [2]. В оценочной функции для описания межмолекулярных взаимодействий используются классические эмпирические парные потенциалы Леннард–Джонса и Кулона, а энергия растворителя

вычисляется в рамках модели неявного растворителя. Такие упрощения позволяют рассмотреть взаимодействия между компонентами на больших временных масштабах.

Исходя из основного термодинамического постулата молекулярной биологии, образование комплекса происходит потому, что комплекс является более термодинамически стабильным, чем свободные несвязанные компоненты [3]. В простейшем случае процесс связывания описывается моделью вида ключ-замок, которая представляется в виде $A + B \leftrightarrow AB$, где A и B являются компонентами комплекса.

Кинетика образования комплекса определяется через кинетику полуреакций. Образование и распад комплекса, как правило, записывают в виде $A + B(k_1) \rightarrow AB$ и $AB(k_2) \rightarrow A + B$, где k_1 – константа скорости прямой реакции, k_2 – константа скорости обратной реакции. Образование комплекса зависит от скорости, с которой он образуется при ассоциации реагентов [3] ($k_{\text{on}}[A][B]$, где k_{on} – константа скорости ассоциации), и скорости, с которой комплекс распадается ($k_{\text{off}}[AB]$, где k_{off} – константа скорости диссоциации). Приведенные выше величины определяют константы связывания.

Численная оценка константы скорости ассоциации k_{on} позволяет судить о кинетике процесса связывания, а оценка k_{on} для белковых комплексов различного вида является одной из основных задач для биохимиков, медицинских химиков и биоинформатиков [3]. Возможно выполнить оценку констант связывания компонентов белковых комплексов экспериментально. Например, с помощью систем поверхностного плазмонного резонанса или флуоресцентной спектроскопии [3]. Такие методы накладывают ряд ограничений, которые сопутствуют проведению физических экспериментов.

На сегодняшний день можно отметить ограниченное число исследований, в которых происходит оценка констант связывания посредством проведения вычислительных экспериментов, имеющих в качестве входных параметров пространственное представление комплексов. Например, в работе [4] с помощью метода молекулярной динамики были произведены оценки констант ассоциации. Очевидно, что применение метода молекулярной динамики требует значительных вычислительных затрат. Недостатком приведенного в исследовании способа является малый временной масштаб, несмотря на применение крупнозернистой модели белкового комплекса.

Помимо метода метода молекулярной динамики исследователи применяют различные численные методы для оценки указанной константы. Например, метод TransComp [5], основанный на теории «переходного» комплекса (transient-complex theory) [6]. В методе рассматривается иная кинетическая модель вида $A + B \rightleftharpoons A^*B \rightarrow AB$, где A^*B – «переходный» комплекс. К недостаткам метода следует отнести наличие верхней и нижней границы оценок, которые заданы авторами в рамках разработанной теории.

Следующим примером метода для оценки константы ассоциации является кинетический метод Монте-Карло [7]. В работе демонстрируется применение метода с использованием крупнозернистой модели для нескольких комплексов вида белок-белок. Недостатком метода является необходимость использования референтного набора белковых комплексов с известными значениями констант связывания, без которых оценка константы связывания невозможна.

Рассматриваемый в настоящей работе метод выборки переходных поверхностей (multiple state transition interface sampling) [8] лишен вышеупомянутых недостатков

и ранее использовался для оценки констант скорости в различных молекулярных системах. Важно отметить, что до настоящего времени не были разработаны конкретные программные инструменты для оценки константы на основе трехмерной структуры с использованием рассматриваемого метода.

Помимо перечисленных методов существуют различные способы «ускорения» молекулярной динамики, позволяющие выполнить численную оценку. Однако, среди существующих подходов только два вышеупомянутых метода были разработаны специально для оценки константы для комплексов вида белок-белок. Методы продемонстрировали положительную корреляцию с экспериментальными значениями на небольших наборах тестовых комплексов, каждый из которых формировался авторами по различным критериям. Следует отметить, что на данный момент для упомянутых методов не проводилось исследование, направленное на оценку степени корреляции с использованием независимого тестового набора комплексов.

Верифицировать работу рассматриваемого метода планируется на основе базы данных SKEMPI [9]. В ней для различных комплексов приводятся константы скорости ассоциации, а сама база сформирована на основе экспериментальных данных. В базе для каждого комплекса присутствует ссылка на работу (или на несколько работ) с результатами эксперимента. В базе данных SKEMPI более 1500 комплексов белок-белок с известной константой скорости ассоциации.

В качестве объекта исследования при проведении вычислительных экспериментов выступает белковый комплекс 2OOB (код в базе данных Protein Data Bank [10]), для которого известна константа связывания, а также численная оценка методом TransComp.

Реализация параллельной версии первого шага исследуемого метода численной оценки константы скорости производится на языке программирования C++. Предметом настоящего исследования являются средства синхронизации и многопоточности языка программирования C++ и высокоуровневые шаблоны библиотеки oneTBB [11]. Основной целью исследования является определение наиболее эффективного примитива синхронизации в используемых вычислительных системах:

- служебный сервер для параллельных вычислений, предоставляемый в рамках инфраструктуры университета «Дубна»;
- учебно-тестовый полигон гетерогенной платформы «HybriLIT» [12] ОИЯИ, позволяющий выполнять вычисления на нескольких вычислительных узлах.

Платформа HybriLIT будет задействована на этапе исследования метода на обширном наборе тестовых комплексов, в то время как использование служебного сервера запланировано после создания программного инструмента, который позволит загружать комплекс через веб-сервер для выполнения оценки. В перечисленных системах присутствуют различия в программно-аппаратном окружении, что может повлиять на эффективность применения средств параллельного программирования.

Основным ожидаемым преимуществом искомого инструмента параллельного программирования является высокая эффективность. Для поиска данного инструмента в исследовании проводится сравнение реализаций одного из паттернов параллельного программирования и демонстрируются результаты экспериментов, которые выполнены в различных вычислительных средах. Практическая значимость заключается в возможности определения оптимальных стратегий параллельной реализации для конкретных условий использования и типа решаемой задачи.

2. Постановка задачи

Связывание компонентов белкового комплекса можно рассматривать в парадигме классической теории переходного состояния, где в процессе моделирования система через последовательность квазиравновесных состояний движется в сторону наименьшей полной энергии по пути с наименьшими энергетическими барьерами. Метод выборки переходных поверхностей основан на идее генерации траекторий, которые позволяют исследовать редкие переходные события между различными состояниями системы. Первым шагом этого метода является выборка траекторий вдоль так называемых переходных путей (или «поверхностей»), которые соединяют различные начальные состояния с конечным. Каждый путь характеризуется одним или несколькими переходным состояниями, часть из которых для нескольких путей совпадает. Объединив все возможные состояния возможно выполнить кластеризацию переходных состояний, а затем выделить основные траектории связывания. Используя выделенные траектории связывания, возможно выполнить численную оценку кинетических параметров связывания, в том числе выполнить оценку константы k_{on}, моделируя переходы между состояниями [8].

При генерации множества траекторий и при поиске переходных путей связывания необходимо многократно оценивать энергию взаимодействия между компонентами комплекса с использованием оценочной функции. Расположение компонентов комплекса на различных расстояниях оказывает влияние на время, затрачиваемое на оценку энергии. На рис. 1 представлены компоненты комплекса 2OOB и график, который демонстрирует зависимость времени выполнения оценки от расстояния между компонентами комплекса при смещении вдоль одной оси одного из компонентов комплекса.

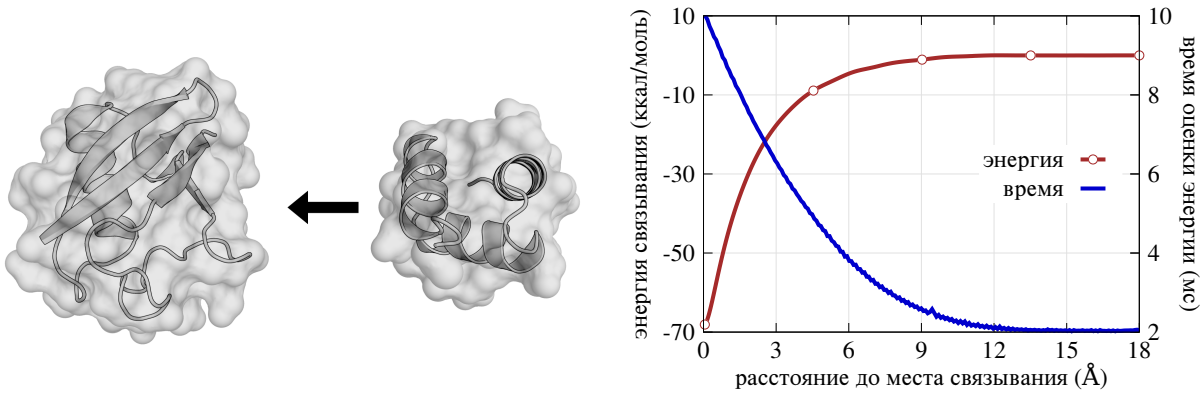


Рис. 1: Связывание и компоненты белка 2OOB (слева). Энергия связывания и время выполнения оценки энергии связывания в зависимости от расстояния между компонентами комплекса (справа).

Следует отметить, что генерируемые траектории и соответствующие найденные пути перехода между состояниями имеют разную «длину». Каждый путь определяет множество различных расположений компонент в пространстве относительно друг друга. При этом структурное представление компонент значительно варьируется, что определяет неравномерность времени оценки энергии взаимодействия. На рис. 2 представлены различные пути связывания, соответствующие им профили и диаграмма размаха, которая демонстрирует значительное отклонение времени поиска при определении пути.

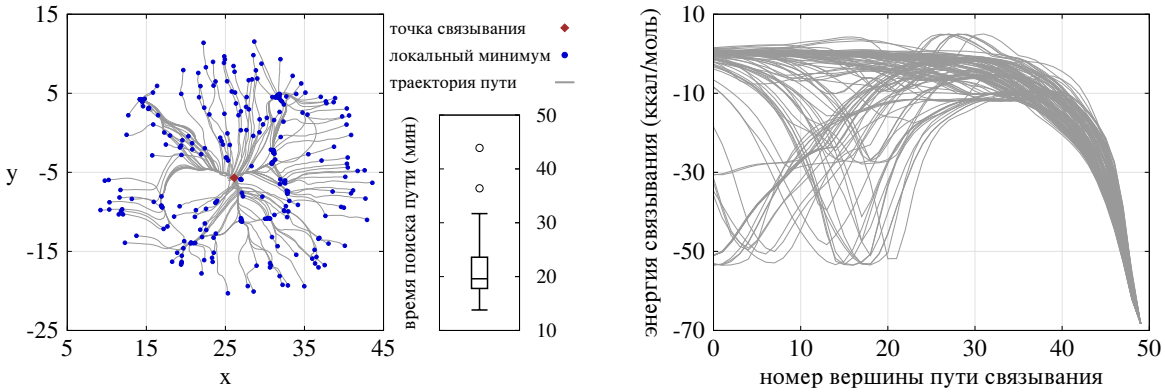


Рис. 2: Пути связывания и диаграмма размаха (слева). Профили путей связывания (справа).

В исследовании поиск пути с наименьшими энергетическими барьерами производился с помощью метода струны (string method) [13]. В методе происходит итеративная эволюция точек, каждая из которых характеризует пространственное положение одной компоненты комплекса. Точки перемещаются под действием градиентных сил до достижения критерия сходимости.

Очевидно, что разделение множества траекторий на блоки одинакового размера с параллельной обработкой каждого блока неэффективно. Задача заключается в организации параллельных вычислений на основе исходного множества траекторий с учетом значительного отклонения времени нахождения каждого пути. Следует отметить, что постановка задачи является в значительной мере стандартной и, в определенном смысле, тривиальной, однако она предусматривает возможность достижения максимальной эффективности параллельных вычислений.

Для решения задачи целесообразно использовать паттерн параллельного программирования «пул потоков» (thread pool). Помимо упомянутого шаблона при использовании системы с несколькими вычислительными узлами применялась парадигма параллельного программирования «ведущий–ведомый» (master-slave) [16]. Если рассматривать паттерн в упрощенном виде, задача делится на несколько подзадач, которые выполняются параллельно на различных вычислительных узлах с использованием паттерна «пул потоков», а затем результаты этих подзадач объединяются.

3. Параллельная реализация

В исследовании реализация паттерна «пул потоков» производилась с использованием очереди задач. Такая реализация возможна с использованием средств языка программирования C++ или шаблонов библиотеки oneTBB. Следует отметить, что возможно смоделировать задачу с помощью синтетического теста, результаты которого, как будет продемонстрировано ниже, не могут полностью указать на наиболее эффективный подход к реализации.

Рассматриваемый шаг метода может быть описан в упрощенной форме следующим образом. Каждая траектория характеризуется фиксированным набором точек, которые распределяются по определенному правилу в области связывания. Конечная точка каждой траектории определяет место связывания и одинакова для всех траек-

торий. Таким образом, имеется массив траекторий размером `paths_number`, в котором каждая траектория определяется уникальным `path_id`. Для каждой траектории требуется выполнить процедуру `calculate` поиска пути связывания, что сопряжено с существенными вычислительными затратами.

В результате выполненной работы произведена параллельная реализация, которая поддерживает вычисления на нескольких вычислительных узлах. Для организации распределенных вычислений использовалась технология MPI. Принцип распределения MPI процессов следующий: на одном вычислительном узле создается один MPI процесс, внутри которого запускаются вычислительные потоки, работающие в системе с общей памятью. Следует отметить, что временные расходы по передаче при обмене между вычислительными узлами минимизированы, поскольку каждая траектория передается в формате одномерного массива чисел с двойной точностью. Множество траекторий распределяется главным процессом на вычислительные узлы блоками фиксированного размера перед началом вычислений. Поскольку существенная часть траекторий исключается после окончания вычислений, в главный процесс возвращается существенно меньшее число путей, которые также передаются в формате одномерного массива.

На этапе реализации метода использовались следующие инструменты: Valgrind [14] для отладки с целью минимизации выделения динамической памяти и Intel Advisor [11] для анализа производительности контексте параллельных вычислений.

В ходе исследования были рассмотрены три способа параллельной организации вычислений в рамках одного вычислительного узла, каждый из которых представлен ниже.

3.1. Блочная реализация

В качестве простейшего подхода к параллельной обработке массива траекторий использовалось блочное параллельное выполнение. Подхода представлен для демонстрации необходимости и преимуществ применения паттерна «пул потоков». Исходный массив траекторий разбивался на блоки фиксированного размера. Для этого использовался кратный числу потоков размер блоков. После этого производилась независимая друг от друга обработка каждого блока соответствующим потоком. Очевидно, что по вышеперечисленным в предыдущем разделе причинам такой подход приводит к неравномерной загрузке потоков и снижению эффективности параллельной обработки и был реализован для демонстрации преимуществ применения следующих способов.

Реализация блочного выполнения произведена на языке программирования C++ с использованием инструментария `std::thread`. Фрагмент реализации представлен в репозитории [17]. В начале создается массив потоков `threads` размером `n_threads`. Затем вычисляется размер каждого блока `part_size`. В первом цикле создаются и запускаются потоки, каждый из которых использует собственный уникальный идентификатор потока `thread_id`, а также начальный и конечный индекс в массиве траекторий (`a` и `b`). В последнем цикле каждый поток ожидает завершения своего выполнения путем вызова `join()`. Этот механизм гарантирует, что главный поток приостанавливает свое выполнение до тех пор, пока все дочерние потоки не завершат свою работу.

3.2. Использование Intel oneTBB

Реализация паттерна «пул потоков» с использованием очереди и средств библиотеки oneTBB выполнена двумя способами. В обоих случаях создавался потокобезопасный контейнер `tbb::concurrent_queue`, в который добавлялись все уникальные идентификаторы траекторий `path_id` (индексы массива траекторий). В первом случае применялся высокоуровневый шаблон `tbb::parallel_for` как наиболее универсальный подход организации параллельных вычислений средствами библиотеки oneTBB. Во втором случае использовался шаблон `tbb::task_group`, предоставляемый библиотекой для управления задачами с различной продолжительностью выполнения. Фрагмент реализации представлен в репозитории [17].

3.3. Использование встроенных средств языка программирования C++

В настоящее время стандарт языка программирования C++ не включает в себя потокобезопасные контейнеры. Однако, можно создать потокобезопасную очередь с использованием различных механизмов синхронизации, доступных в языке. Поскольку результаты синтетического теста [17] демонстрируют практически идентичное время выполнения различных примитивов синхронизации в исследовании был выбран базовый механизм синхронизации – `std::mutex`. Фрагмент реализации, код теста и результаты представлены в репозитории [17]. Внутри цикла `while` создается объект типа `lock_guard`, который блокирует мьютекс при входе в блок. Это гарантирует, что только один поток имеет доступ к очереди в определенный момент времени. Дополнительный локальный блок внутри цикла используется для ограничения области видимости объекта и для того, чтобы мьютекс был разблокирован после выполнения операций, которые требуют защиты, так как объект `lock_guard` автоматически разблокирует мьютекс при выходе из своей области видимости. Такой подход реализует идиому RAII (Resource Acquisition Is Initialization), что гарантирует освобождение ресурсов автоматически при выходе из области видимости объекта.

4. Вычислительные эксперименты

Поскольку основной целью исследования является определение наиболее эффективного примитива синхронизации в используемых вычислительных системах, важно учитывать их особенности и конфигурацию при запуске экспериментов. В случае служебного сервера, предоставляемого в рамках инфраструктуры университета «Дубна», использовался один процессор Intel Xeon E5-2650 с 8 физическими ядрами. При экспериментах на учебно-тестовом полигоне гетерогенной платформы «HybriLIT» вычисления производились на узлах, каждый из которых включает в себя два 12-ядерных процессора Intel Xeon E5-2695 v2.

Необходимо отметить, что использованные в экспериментах вычислительные узлы не поддерживают архитектуру NUMA. Естественно предположить, что наличие двух сокетов на вычислительном узле позволит использовать преимущества NUMA-архитектуры. Однако в случае некоторых узлов платформы HybriLIT это не так. На момент проведения исследования для рассматриваемой группы узлов, которые присутствуют в очереди задач для вычислений с названием «cpu», использовался гипервизор, конфигурация которого исключает поддержку NUMA. Поскольку под-

держка отключена на уровне операционной системы, запускаемые приложения не могут использовать NUMA-архитектуру.

При запуске устанавливалась переменная окружения `OMP_PLACES` для распределения потоков по процессорным ядрам. Поскольку поддержка NUMA-архитектуры на узлах отсутствует, необходимость в создании группы потоков и управлении их размещением отпадает, поэтому для переменной устанавливалось значение `cores`. Помимо этого, через систему управления задачами SLURM устанавливалась опция с указанием максимально допустимого количества процессорных ядер, чтобы на вычислительных узлах присутствовала только одна задача.

На каждой итерации в методе струны присутствует один времязатратный шаг – вычисление оценки энергии взаимодействия, который состоит из двух этапов: поиска взаимодействующих атомов и вычисления энергии взаимодействия с использованием эмпирических парных потенциалов. При поиске весомым фактором, ограничивающим производительность, является пропускная способность памяти, поскольку в простейшем случае предполагается перебор всех атомов комплекса. Поиск может быть значительно оптимизирован применением k-d-дерева, что, однако, никак не исключает упомянутый фактор, поскольку только сокращает количество обращений к памяти.

Несмотря на то, что используемые потенциалы [2] в строгом смысле нелинейны и часть из них построена с применением элементарных функций, при вычислении требуется получение коэффициентов в соответствии с типом взаимодействующих атомов, которое также предполагает чтение памяти.

В результате применения Valgrind выявлено, что при поиске пути количество выделенной динамической памяти с течением времени остается постоянным, а наибольшее время выполнения занимают функции, определяющие смещение компоненты и оценку энергии взаимодействия, в то время как наиболее часто вызываемая функция используется для получения координат атомов. Помимо этого, с помощью утилиты Perf определен приемлемый уровень кэш-промахов. Процентное отношение кэш-промахов к общему числу обращений для всех уровней кэша процессора составило 3.37%.

На первом этапе экспериментов проведен анализ производительности параллельной реализации с базовым механизмом синхронизации. На рис. 3 представлены результаты использования инструмента Intel Advisor при запуске эксперимента по поиску 72 путей на одном вычислительном узле кластера Hybrilit с использованием 24 потоков. На графике представлена Cache-Aware Roofline-модель [15], на которой отмечены полученные пропускные способности различных уровней кэша и памяти, а также пики производительности для различных типов вычислений. Метками обозначены наиболее нагруженные функции и циклы. Цвета и размеры отражают относительное время выполнения, то есть показывают долю времени, затраченного на выполнение определенного «участка» программы (функции отмечены только цветом, циклы обведены), от общего времени выполнения программы.

Выделенная зеленой областью группа функций выполняется при вычислении парных потенциалов. Участки, которые выделены тремя метками циклов по центру графика (желтым, красным и зеленым цветами), выполняются при обходе k-d-дерева и при вычислении расстояния между потенциально взаимодействующими атомами. Находящаяся в «scalar compute bound» области функция вызывается при изменении пространственного положения компонента комплекса.

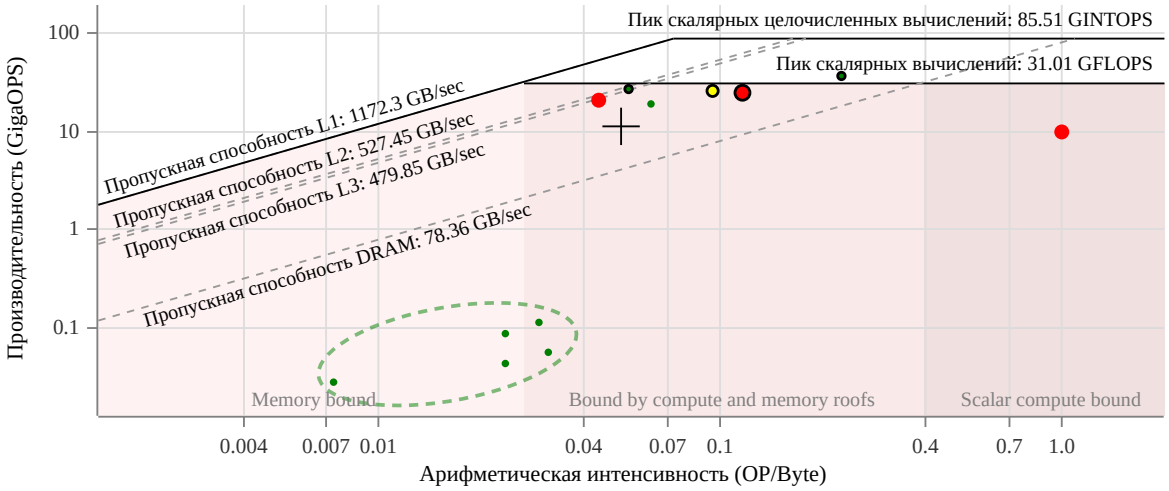


Рис. 3: Построенная средствами Intel Advisor Roofline-модель.

Маркер в виде креста на графике означает совокупную производительность всей программы, включая все функции и циклы. Несмотря на то, что маркер находится выше линии пропускной способности памяти, производительность значительно ограничена пропускной способностью памяти. Используя метрики производительности Intel Advisor, можно сделать вывод, что текущая реализация ограничена как вычислительной мощностью, так и пропускной способностью памяти.

На втором этапе экспериментов использовался служебный кластер, состоящий из одного процессора Intel Xeon E5-2650 с 8 физическими ядрами. Эксперименты проводились с использованием 8 потоков без применения технологии MPI. Использовался компилятор Clang (версия 17) с предоставляемой стандартной библиотекой шаблонов (libc++). Результаты ускорения работы и эффективности блочной реализации (block), реализации с использованием библиотеки oneTBB (TBB pf, TBB tg) и реализация с использованием средств языка программирования C++ (mutex) представлены на рис. 4. Сравнение проводилось относительно последовательной версии программы, с помощью который были найдены 120 путей. Они изображены на рис. 1. Выполнение последовательной версии программы заняло 20 часов. Единственным преимуществом последовательной версии является малая пространственная сложность, поскольку для оценки энергии взаимодействия на основе одного из компонентов комплекса (недвижимого) требуется однократное построение k-d-дерева для эффективного поиска взаимодействующих атомов. В случае многопоточной версии для каждого потока однократно создавалось дерево, то есть использование памяти возрастает кратно числу потоков. Следует отметить, что предъявляемые методом требования к памяти незначительны, а сама память при поиске пути выделяется однократно.

На третьем этапе экспериментов использовались ресурсы учебно-тестового полигона гетерогенной платформы «HybriLIT» ОИЯИ. Использовался компилятор GCC (версия 12) с предоставляемой стандартной библиотекой шаблонов (libstdc++). Распределенные вычисления проводились с использованием технологии MPI двумя процессами, что отмечено на рис. 5. В экспериментах рассматривалось применение шаблона `tbb::parallel_for`.

Как видно на рис. 4 и рис. 5, во всех экспериментах наибольшую эффективность показала параллельная реализация с использованием библиотеки oneTBB. Причины большей эффективности связаны с механизмом планирования вычислений [18],

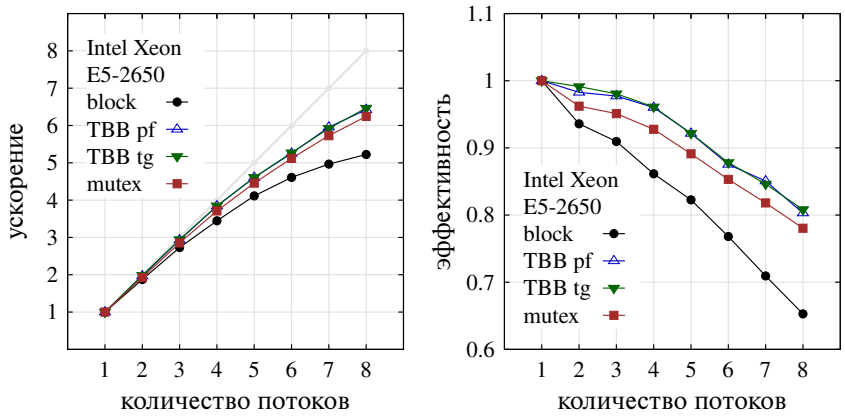


Рис. 4: Ускорение и эффективность на служебном сервере университета «Дубна».

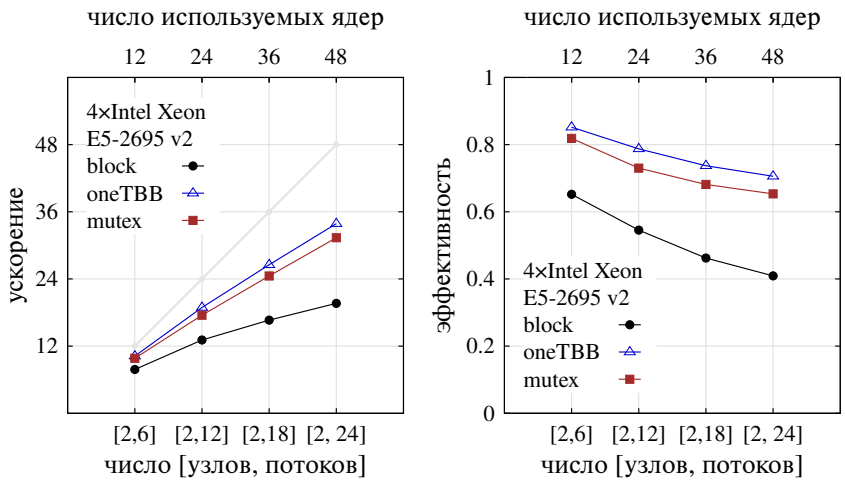


Рис. 5: Ускорение и эффективность на двух вычислительных узлах платформы «HybriLIT».

который реализован в библиотеке oneTBB. Указанный механизм, использование которого происходит при вызове `tbb::parallel_for` и `tbb::task_group`, представляет собой автоматизированный способ управления выполнением потоков, который способствует оптимальному использованию ресурсов и повышению производительности вычислений, причем это наблюдается в обоих случаях. Следует отметить, что в экспериментах использовались разные компиляторы с различными реализациями стандартной библиотеки шаблонов, но с одинаковой версией библиотеки oneTBB (версия 2021.12).

Во всех экспериментах время вычисления последовательной версии ограничивалось 24 часами. Число обработанных за указанное время траекторий в дальнейшем использовалось при запуске параллельных версий.

5. Заключение

При использовании двух вычислительных узлов с распределенной памятью для параллельных вычислений с применением библиотеки oneTBB была достигнута эффективность приблизительно в 70%. Хотя полученная эффективность достаточно высока, её можно охарактеризовать как недостаточную. Дальнейшее исследование

будет посвящено оптимизации процесса формирования выборки с целью повышения эффективности, параллелизации следующих этапов метода выборки переходных поверхностей, а также изучению масштабируемости разработанной параллельной реализации.

В результате выполненной работы проведена параллельная реализация основного этапа метода выборки переходных поверхностей несколькими различными способами и достигнута основная цель исследования – выявлен наиболее эффективный способ организации параллельных вычислений для рассматриваемых вычислительных систем, который основан на применении библиотеки oneTBB. Следует отметить, что применение библиотеки oneTBB позволило увеличить эффективность вычислений примерно на 5% по сравнению с реализацией, осуществленной стандартными средствами языка программирования C++.

Для рассматриваемого белкового комплекса получен ряд численных оценок константы k_{on}, который коррелируют с известными численными и экспериментальными оценками. Расширение тестового набора белковых комплексов и проведение вычислительных экспериментов для оценки констант у различных комплексов являются темами дальнейших исследований.

Список литературы

[1] Хрущев С.С. и др. Моделирование белок-белковых взаимодействий с применением программного комплекса многочастичной броуновской динамики ProKSim // Компьютерные исследования и моделирование. 2013.
https://doi.org/doi:10.20537/2076-7633-2013-5-1-47-64

[2] Полуян С.В., Никулин Д.А., Ершов Н.М. Разработка и верификация оценочной функции для учета межмолекулярных взаимодействий в белковых комплексах // Информационно-телекоммуникационные технологии и математическое моделирование высокотехнологичных систем: материалы Всероссийской конференции с международным участием. Издательство РУДН, Москва. – 2023. – С. 231–235.

[3] Борисов Д.В., Веселовский А.В. Кинетика связывания лиганда с рецептором в разработке лекарств // Биомедицинская химия. – 2020.
https://doi.org/10.18097/PBMC20206601042

[4] Souza P., et al. Protein-ligand binding with the coarse-grained Martini model // Nature Communications. – 2020.
https://doi.org/10.1038/s41467-020-17437-5

[5] Qin S., Pang X., Zhou HX Automated prediction of protein association rate constants // Structure. – 2011.
https://doi.org/10.1016/j.str.2011.10.015

[6] Alsallaq R., Zhou HX. Electrostatic rate enhancement and transient complex of protein-protein association // Proteins. – 2008.
https://doi.org/10.1002/prot.21679

[7] Dhusia K, Su Z., Wu Y. Using Coarse-Grained Simulations to Characterize the Mechanisms of Protein-Protein Association // Biomolecules. – 2020.
https://doi.org/10.3390/biom10071056

[8] Rogal J., Bolhuis P.G. Multiple state transition path sampling // J. Chem. Phys. – 2008.
https://doi.org/10.1063/1.3029696

[9] Jankauskaitė J., et al. SKEMPI 2.0: an updated benchmark of changes in protein–protein binding energy, kinetics and thermodynamics upon mutation // Bioinformatics. – 2018.
https://doi.org/10.1093/bioinformatics/bty635

[10] Berman H.M., et al. The Protein Data Bank // Nucleic Acids Research. – 2000.
https://doi.org/10.1093/nar/28.1.235

[11] Сысоев А.В., и др. Учебный курс «Программирование с использованием модели oneAPI» // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. – 2022.
https://doi.org/10.14529/cmse220303

[12] Adam Gh., et al. IT-ecosystem of the HybriLIT heterogeneous platform for high-performance computing and training of IT-specialists // CEUR Workshop Proceedings, Dubna, Russia. – 2018. – Vol. 2267. – P. 638–644.

[13] E Weinan, Ren W., Vanden-Eijnden E. Simplified and improved string method for computing the minimum energy paths in barrier-crossing events // The Journal of Chemical Physics. – 2007.
https://doi.org/10.1063/1.2720838

[14] Инструмент Valgrind.
https://valgrind.org/
(дата обращения: 08.06.2024).

[15] Williams S., Waterman A., Patterson D. Roofline: An Insightful Visual Performance Model for Multicore Architectures // Communications of the ACM. – 2009.
https://doi.org/10.1145/1498765.1498785

[16] Sahni S., Vairaktarakis G. The master-slave paradigm in parallel computer and industrial settings // Journal of Global Optimization. – 1996.
https://doi.org/10.1007/BF00121679

[17] Репозиторий с синтетическим тестом и фрагментами кода.
https://vcs.uni-dubna.ru/psm/multithreading
(дата обращения: 08.06.2024).

[18] Voss M., Asenjo R., Reinders J. C++ Parallel Programming with Threading Building Blocks. – 2021.
https://doi.org/10.1007/978-1-4842-4398-5

Развитие отечественной модели динамики атмосферы нового поколения

В.В. Шашкин[1,2,3], Г.С. Гойман[1,2,3], И.Д. Третьяк[3,1,2]

[1]Институт вычислительной математики РАН,
[2]Гидрометцентр РФ,
[3]Московский физико-технический институт

В ИВМ РАН и Гидрометцентре России разрабатывается модель динамики атмосферы нового поколения для численного прогноза погоды и исследований климата. В модели используется сетка кубическая сфера с квазиравномерным разрешением. Модель программно реализована как набор блоков, которые можно использовать для построения широкого класса алгоритмов решения уравнений динамики атмосферы (дифференциальные операторы, схемы интегрирования по времени, параллельные обмены и т.д.). В настоящее время мы разработали алгоритм численного решения негидростатических уравнений динамики для глобального прогнозирования погоды с высоким разрешением, а также алгоритм численного решения уравнений динамики в гидростатическом приближении по вертикали для моделирования климата. Оба алгоритма были протестированы на общепринятых идеализированных задачах и показали результаты, соответствующие передовому мировому уровню. В этой статье мы обосновываем основные идеи построения модели и описываем особенности ее программной реализации. Приводятся результаты исследования параллельной эффективности. Модель хорошо масштабируется как минимум до 6912 ядер системы Cray XC-40 Главного вычислительного центра Росгидромета.

Ключевые слова: моделирование атмосферы, численный прогноз погоды, моделирование климата

1. Введение

Гидродинамические модели общей циркуляции атмосферы в настоящее время являются основными инструментами численного прогноза погоды и исследований климата. Также, подобные модели необходимы для оценки текущего состояния атмосферы по данным наблюдений (анализ и реанализ), что необходимо для множества научных приложений.

Текущее горизонтальное разрешение глобальных атмосферных моделей составляет около 10 км в задаче прогноза погоды и 50-100 км при моделировании климата. Такие модели имеют $10^9 - 10^{10}$ степеней свободы и используют тысячи вычислительных ядер.

Повышение разрешения сетки – магистральный путь повышения точности моделирования атмосферы. В настоящее время считается важным достичь такого разрешения, при котором мезомасштабные метеорологические явления, формирующие значительную часть опасных погодных явлений, будут явно представлены на сетке.

Таким образом, предсказуемость подобных явлений, ранее возможная только в рамках регионального краткосрочного прогноза (1-2 дня), станет уделом глобального среднесрочного прогноза погоды (3-7 дней).

Текущее разрешение глобальных моделей атмосферы является своеобразным рубежом, для перехода которого необходимо отказаться от регулярной широтно-долготной сетки и использовать одну из сеток с квазиравномерным разрешением на сфере. Проблема регулярной широтно-долготной сетки – уменьшение шага сетки по долготе до очень малых величин около полюсов, что приводит к падению вычислительной эффективности (жесткое ограничение на шаг по времени для явных методов или же замедление сходимости итерационных методов решения систем уравнений для неявных методов). Это делает невозможным практическое моделирование с высоким разрешением [31]. Кроме того, необходимо отказаться от гидростатического приближения по вертикали, которое не является достаточно точным при описании явлений с горизонтальным масштабом около 10 км [35].

В силу обозначенных причин, а также развития средств разработки программного обеспечения, для дальнейшего повышения разрешения глобальных метеорологических моделей целесообразна разработка именно новых программных комплексов. В литературе принято обозначать негидростатические глобальные модели атмосферы на сетках с квазиравномерным разрешением на сфере как модели «нового поколения».

Несколько исследовательских центров уже представили модели нового поколения: модель ICON [47] на икосаэдральной сетке (Германия), модель FV3 [20] на основе сетки кубическая сфера (США) и модель GEM с использованием сетки Инь-Янь (Канада), все они используется для прогноза погоды. В области исследования климата Национальный центр атмосферных исследований (США) использует гидростатическую модель CAM на сетке кубическая сфера [17]. В Метеорологической службе Великобритании и Европейском центре среднесрочного прогноза погоды ведется разработка атмосферных моделей нового поколения GUNGHO [15] и FVM [4],[16] (сетки кубическая сфера и редуцированная широтно-долготная соответственно).

Мы считаем, что разработка глобальной модели атмосферы нового поколения для прогноза погоды, разрешающей мезомасштабные экстремальные погодные явления (т.е. с целевым шагом сетки 3-5 км), исключительно важна для России. Глобальный охват прогноза особенно актуален, учитывая большую географическую протяженность и еще большую протяженность области атмосферы, которая может влиять на погоду над Россией на временном масштабе нескольких дней. В последнее время все большее внимание уделяется проблемам изменения климата, что делает актуальным обновление атмосферной компоненты отечественной модели Земной системы. Новая модель общей циркуляции атмосферы для исследования климата должна отвечать требованиям Международной группы экспертов по изменению климата, которые, например, включают многолетние расчеты эксперимента HighResMIP [9] с разрешением $0,25^\circ$ (около 28 км).

Некоторое время назад мы начали разработку модели атмосферы нового поколения, с помощью которой можно было бы решить указанные выше задачи. В основном, мы рассматриваем блок численного решения уравнений гидротермодинамики атмосферы, поскольку на эту часть больше всего влияют геометрия сетки, вычислительная инфраструктура и т.д. В то же время мы считаем, что блок параметризаций (приближенных моделей процессов подсеточного масштаба) из текущей оперативной

модели Гидрометцентра РФ ПЛАВ можно использовать в новой модели практически без изменений, поскольку ранее он использовался в региональной модели атмосферы с разрешением, близким к целевому разрешению новой модели. В этой статье мы описываем текущее состояние разработки, последние результаты и перспективы дальнейших исследований.

2. Обзор реализованных численных методов

Область применения глобальных численных моделей атмосферы весьма обширна. Помимо численного прогноза погоды и моделирования климата, она также включает моделирование средней и верхней атмосферы, моделирование состава атмосферы и другие задачи. Требования к моделям в этих задачах (формулировка уравнений динамики, целевое разрешение, выбор численных методов, вертикальная система координат) сильно различаются, что обуславливает большое разнообразие моделей.

Тем не менее, большинство возможных формулировок моделей могут иметь общие компоненты программной реализации: процедуры межпроцессорной коммуникации и ввода-вывода, базовые дифференциальные операторы (например, градиент). Мы считаем, что при правильно выстроенной программной реализации, в одном программном комплексе могут сосуществовать несколько моделей, каждая из которых может быть применена для своей конкретной задачи.

Проблема разработки модели атмосферы нового поколения может трактоваться как реализация набора базовых численных методов, которые можно комбинировать для создания эффективного решения (то есть модели) в широком спектре приложений. В данном разделе мы приводим обзор подобных методов, реализованных в модели на данный момент.

2.1. Уравнения динамики

Выбор системы уравнений является основным аспектом построения моделей атмосферы, во многом определяющим их свойства. Три основных фактора, определяющие систему уравнений – используемые аппроксимации (например, гидростатическая аппроксимация по вертикали), выбор прогностических переменных [36] и вертикальных координат.

В настоящее время мы реализовали алгоритмы решения негидростатических уравнений (для нескольких наборов прогностических переменных) в вертикальной координате по высоте и алгоритм решения гидростатических уравнений в координатах, основанных на давлении. Кроме того, была реализована модель мелкой воды на сфере, которая лишь грубо аппроксимирует динамику атмосферы, но оказалась чрезвычайно полезной для проверки аппроксимаций по горизонтали. Также, разработан автономный алгоритм решения уравнений переноса для проверки схем аппроксимации адвективных слагаемых уравнений. В будущем мы планируем использовать его для расчетов переноса газовых примесей в атмосфере при моделировании климата.

2.2. Сетка с квазиравномерным разрешением на сфере

Наша модель основана на сетке кубическая сфера [24], которая получается путем центральной проекции сетки на гранях куба на вписанную сферу. Сетка на грани куба получается путем tan-отображения: если $x, y \in [-1, 1]$ являются координатами на

квадрате, сетка равномерна по $\alpha, \beta : x = \tan\alpha, y = \tan\beta$. Такой подход обеспечивает более равномерное разрешение после проекции на сферу, чем в случае сетки равномерной по x, y. Соотношение между площадями самой большой и самой маленькой ячейки составляет $\sqrt{2}$. Рассматриваемая сетка неортогональная, угол между линиями α и β варьируется от 60° до 120° с наибольшими (наименьшими) значениями вблизи вершин куба.

2.3. Конечно-разностная пространственная аппроксимация

Мы реализовали метод конечных разностей со свойством суммирования по частям [32] для построения устойчивой пространственной аппроксимации высокого порядка точности [29]. В рамках этого метода используются операторы первой производной, построенные так, что выполняется свойство суммирования по частям – дискретный аналог свойства интегрирования по частям:

$$\int_a^b u\frac{\partial v}{\partial x}dx = -\int_a^b v\frac{\partial u}{\partial x}dx - u(a)v(a) + u(b)v(b), \tag{1}$$

для любых дифференцируемых функций u и v. Свойство суммирования по частям необходимо и во многих случаях достаточно для устойчивости, сохранения массы и энергии.

Реализованы операторы первой производной 2-го, 4-го и 6-го порядка аппроксимации. Можно использовать неразнесенную (тип A по Аракаве [1]) и разнесенную (тип C) сетки по горизонтали. По вертикали реализованы разнесенные сетки Чарни-Филипса и Лоренца. Для сеток с разнесением переменных используется подход [6].

Также, мы реализовали стандартную конечно-разностную горизонтальную аппроксимацию с использованием подхода виртуальных точек [26], [49] вблизи ребер куба. Эта аппроксимация более точна вблизи ребер, чем аппроксимация со свойством суммирования по частям, но приводит к возникновению неустойчивых акустических мод, которые необходимо демпфировать. В дальнейшем, возможно будут реализованы пространственные аппроксимации конечно-объемного типа [44], а также аппроксимации на основе метода спектральных элементов и разрывного метода Галеркина [34], [22].

2.4. Методы интегрирования по времени

Явные методы интегрирования по времени не могут использоваться для практического моделирования негидростатической атмосферы из-за ограничения шага по времени условием Куранта, что связано с малым шагом сетки по вертикали и высокой фазовой скоростью звуковых волн. Типичное значение шага по времени в реальных задачах будет порядка $0,1$ с. Несмотря на это, явные методы Рунге-Кутты, оптимизированные для гиперболических задач, такие как [23], могут быть конкурентоспособными для моделирования гидростатической динамики. Кроме того, явные методы полезны для отладки и тестирования модели на идеализированных задачах. Мы реализовали классический РК метод 4-го порядка и оптимизированный метод [23].

Популярным подходом к интегрированию по времени в негидростатических моделях являются методы с неявностью по вертикали [19], в которых слагаемые, включающие вертикальные производные (т.е. ответственные за вертикальное распростра-

нение звуковых волн), интегрируются по неявной схеме, а слагаемые с горизонтальными производными интегрируются явно. Система алгебраических уравнений, порожденная вертикально-неявной формулировкой, распадается на ряд систем вдоль каждого вертикального столбца сетки модели. Таким образом, эти методы не влияют на эффективность параллельных вычислений, поскольку атмосферные модели традиционно используют декомпозицию расчетной области только в горизонтальном направлении. Шаг по времени в вертикально-неявных методах ограничен условием Куранта для звуковых волн, распространяющихся горизонтально, т.е. величиной, пропорциональной $\Delta x/c_s$, где Δx — шаг сетки по горизонтали, а $c_s \approx 343\,\text{м/с}$ – фазовая скорость звуковых волн. Типичная величина шага по времени при $\Delta x = 5\,\text{км}$ будет около 10 с, что на два порядка больше, чем для явных методов. Мы реализовали семейство вертикально-неявных схем интегрирования по времени на основе аддитивных методов РК (с двойной таблицей Бутчера) [2]. Для разделения уравнений на части, интегрируемые явно и неявно, реализованы подходы из [7].

Другой возможный подход к интегрированию по времени – использовать неявную формулировку для слагаемых, ответственных как за вертикальное, так и горизонтальное распространение звуковых волн, а адвективные слагаемые интегрировать явно. Мы реализовали несколько схем этого класса, все они основаны на аддитивных РК методах с подходами явного/неявного разделения из [7].

Максимальная величина шага по времени для этого класса методов определяется условием Куранта для горизонтальной адвекции, т.е. величиной, пропорциональной $\Delta x/v_{max}$, где $v_{max} \approx 100$ м/с — максимальная скорость горизонтального ветра. Следовательно, значение шага по времени может быть примерно в три раза выше, чем для вертикально-неявных методов. Однако, этот подход требует решения системы алгебраических уравнений, которая связана как по горизонтали, так и по вертикали, и, таким образом, возможный прирост производительности по сравнению с вертикально-неявными методами зависит от реализации и настройки эффективного и масштабируемого алгоритма решения такой системы. В настоящее время мы используем итерации Ньютона-Крылова, при таком подходе вертикально-горизонтально-неявный метод работает медленнее, чем вертикально-неявный. Мы также реализовали геометрический многосеточный метод [5, 42, 8] для линейной системы уравнений с матрицей Якоби и протестировали его на изолированной задаче. Результаты следует признать перспективными, но необходима дополнительная работа для внедрения многосеточного алгоритма в расчетную схему модели.

Ограничение Куранта на шаг по времени для адвективных слагаемых можно обойти, если использовать полулагранжев метод [30]. Полунеявный полулагранжев метод довольно распространен среди текущего поколения глобальных моделей атмосферы [37], [3], [10], но почти не используется в моделях нового поколения. Это связано с трудностями параллельной реализации, отсутствием локального сохранения массы, проблемами с устойчивостью гравитационно-волновых мод при решении негидростатических уравнений [27], а также необходимостью реализации быстрого массивно-параллельного алгоритма решения линейных систем. Однако эти проблемы были в основном решены в последнее десятилетие [45], [25], [11], [48], [21]. Поэтому мы считаем перспективным объединить эти методы и реализовать устойчивую массивно-параллельную полунеявную полулагранжеву схему, локально-сохраняющую массу.

В настоящее время, в модели реализована связка массивно-параллельного полулагранжев алгоритма [45] с полунеявной формулировкой [11].

2.5. Локальное повышение разрешения

Наш текущий подход к моделированию с локальным повышением разрешения состоит в измельчении шага сетки внутри одного ее блока. Это может быть как грань сетки кубическая сфера целиком, так и блок, встроенный в эту грань. Для пространственной аппроксимации используется конечно-разностный метод со свойством суммирования по частям [18]. В настоящее время мы реализовали этот подход только в модели мелкой воды [41] и приступили к реализации в трехмерных моделях. В будущем, мы рассматриваем возможность реализации подходов одно- и двунаправленного вложения, таких как [46] (более традиционных для вычислительной метеорологии).

3. Особенности программной реализации

Код модели написан на языке программирования Fortran с использованием конструкций языка, представленных в последних стандартах. Для повышения модульности и адаптивности кода мы используем парадигму объектно-ориентированного программирования, которая позволяет максимально отделить описание логики алгоритма от конкретных особенностей выполнения параллельных операций или взаимодействия с аппаратным обеспечением при выполнении вычислений. В результате большинство вычислений выполняется в изолированных секциях кода (ядрах), которые могут быть адаптированы к использованию различных вычислительных систем. На данный момент все ядра реализованы для выполнения вычислений на центральных процессорах, однако в будущем такая организация кода позволит относительно легко перейти к использованию других архитектур, например, графических ускорителей.

3.1. Представление вычислительной области и сеток

Для описания направления обхода точек внутри каждого блока, количества точек в каждом направлении, связности блоков и их граней между собой используется абстрактный тип данных `topology_t`. Использование такой абстракции позволяет описывать логическую структуру многоблочных логически прямоугольных сеток для различных типов вычислительных областей: поверхность сферы, периодический отрезок, бипериодическая плоскость (поверхность тора) а также области с внешними граничными условиями. Практическая ценность такой универсальности заключается в том, что она позволяет проводить предварительное тестирование программных компонент и алгоритмов модели в упрощенных постановках. Кроме того, это расширяет диапазон применений реализованной инфраструктуры, позволяя в будущем реализовать на основе данного кода региональную модель атмосферы или, например, модель океана.

Предполагается, что внутри каждого блока определена обобщенная криволинейная система координат. Абстрактный класс `metric_t` описывает интерфейсы функций для вычисления основных метрических величин таких как ко- и контравариантные базисные вектора, компоненты метрического тензора, символы Кристоффеля и многие другие. Вычислительная область нашей модели полностью определяется комбинацией объектов типа `metric_t` и `topology_t`.

Наконец, тип `mesh_t` описывает расположение точек в логически прямоугольной сетке внутри блока. Этот объект также содержит некоторые заранее вычисленные

значения метрических величин в узлах сетки. В нашем коде мы выделяем следующие типы точек: «o»-точки располагаются в центрах ячеек сетки; точки «x», «y» – центры граней ячеек; точки «xy» – вершины ячеек. Каждый тип точек описывается при помощи экземпляра типа `mesh_t`. Такой подход к описанию узлов сетки позволяет естественным образом реализовать в рамках модели различные варианты сеток с разнесением переменных [1].

3.2. Параллельные вычисления и декомпозиция области

Стратегия распределенного хранения данных и организации параллельных вычислений, используемая в нашем коде, описана в [28], поэтому здесь лишь кратко описываются ключевые концепции.

В качестве элементарной единицы разбиения сетки используется так называемый тайл (tile) – логически прямоугольную часть сетки в рамках одного блока, не пересекающаяся ни с какими другими тайлами. Параллельная декомпозиция расчетной области достигается за счет распределения этих тайлов по вычислительным узлам. Количество тайлов может превышать количество вычислительных процессов, и каждый процесс может обрабатывать разное количество тайлов, что позволяет при необходимости использовать статическую балансировку вычислительной нагрузки между процессами.

На данный момент в коде реализован алгоритм разбиения, основанный на двумерной геометрической декомпозиции области с использованием $N_b \times p \times q$ тайлов и MPI-процессов, где N_b – количество блоков сетки (6 в случае сетки кубическая сфера), а p, q – некоторые натуральные числа. В будущем мы планируем реализовать более сложные варианты, основанные на использовании заполняющих пространство кривых.

Вся необходимая информация о декомпозиции области хранится в объекте типа `partition_t`. Совокупная информация о вычислительной области хранится в рамках одного объекта типа `domain_t`, который включает в себя экземпляры типов `topology_t`, `metric_t`, `mesh_t` и `partition_t`. Возможность создания нескольких экземпляров `domain_t` удобна при необходимости работать с сетками различного разрешения. Такая необходимость возникает, например, при реализации многосеточного алгоритма.

Распределенное хранение значений сеточных функций осуществляется при помощи объектов типа `grid_field_t`. Этот объект состоит из набора массивов, каждый из которых представляет функцию в рамках одного тайла разбиения. Этот тип данных поддерживает основные операции векторной алгебры, такие как суммирование векторов и умножение на константу. Для выполнения параллельных операций с горизонтальными зависимостями используется так называемый подход гало-зон, то есть область каждого тайла расширяется таким образом, чтобы включать в себя точки, которые требуются для выполнения вычислений в точках внутри тайла.

Наконец, абстрактный класс `exchange_t` предоставляет интерфейсы для выполнения параллельных обменов между MPI-процессами.

В настоящее время реализовано распараллеливание с использованием технологии MPI, однако в будущем мы планируем реализовать гибридный подход MPI+OpenMP, в рамках которого вычисления на разных тайлах, принадлежащих одному и тому же MPI-процессу, распределяются между OpenMP-потоками.

3.3. Сеточные операторы, операторы модели, схемы интегрирования по времени

Реализация сеточных операторов организована следующим образом: каждому виду сеточного оператора (дивергенция, градиент, адвекция и т.д.) соответствует абстрактный класс, определяющий интерфейс для выполнения расчета данного оператора. При этом, различные алгоритмы для аппроксимации операторов реализуются как конкретные экземпляры соответствующих абстрактных классов. Таким образом, реализация модели основана только на знании интерфейсов методов абстрактных классов, а не на их конкретных реализациях. Это позволяет нам изменять выбор операторов (например, аппроксимацию градиента второго или четвертого порядка) во время запуска модели без необходимости модификации кода модели.

Абстрактные классы, реализующие различные сеточные операторы, объединяются в рамках конкретной реализации абстрактного класса `operator_t`, который отвечает за вычисление правых частей уравнений модели. Этот тип также определяет операцию решения нелинейного уравнения, которое возникает, когда некоторые или все члены уравнения аппроксимируются неявно. Алгоритм вычисления правой части системы уравнений модели в качестве входного и выходного аргумента использует экземпляр абстрактного класса (`stvec_t`), который описывает вектор-состояния модели. Абстрактный класс `stvec_t` также реализует основные операции векторной алгебры для прогностических переменных. Конкретные реализации `stvec_t`, как правило, являются коллекциями экземпляров типа `grid_field_t`.

Абстрактный класс `timescheme_t` используется для реализации различных схем интегрирования по времени. Этот класс определяет интерфейс для вычисления значений прогностических величин на новом шаге по времени с использованием в качестве аргументов объектов типа `operator_t`, `stvec_t` и величины шага по времени `dt`. Поскольку процедура выполнения шага по времени не зависит от особенностей реализации отдельных операторов и векторов состояния, одна и та же реализация какого-либо метода интегрирования по времени может без изменений использоваться для интегрирования различных уравнений: уравнений мелкой воды, гидростатических или негидростатических уравнений динамики атмосферы и других.

3.4. Возможности тестирования

Одним из важных преимуществ представленной программной инфраструктуры является возможность независимого тестирования отдельных компонент и алгоритмов модели. Ниже приводятся некоторые примеры тестов, которые рутинно выполняются в процессе разработки модели. Для всех реализованных типов обменов отдельно тестируется правильность заполнения гало-зон. Свойства аппроксимации и сходимости всех сеточных операторов (градиент, дивергенция, члены силы Кориолиса и т.д.) проверяются с использованием аналитических полей. Кроме того, проверяются схемы интегрирования по времени для получения ожидаемого характеристического полинома для уравнения Дальквиста. Все операторы модели, предназначенные для использования с неявными методами интегрирования по времени, проверяются с точки зрения правильного решения возникающей нелинейной системы уравнений.

Возможность такого тестирования значительно упрощает процесс разработки и модификации модели. Как правило, каждый новый реализованный в рамках кода

компонент (например, новая аппроксимация градиента или оптимизированный вариант процедуры параллельных обменов) может быть индивидуально протестирован. Кроме того, сохранение результатов тестирования для уже реализованных операций позволяет выполнять тесты на воспроизводимость при переработке или оптимизации кода. В случае появления каких-либо ошибок в коде, они, скорее всего, будут идентифицированы и точно локализованы в рамках автоматизированных запусков элементарных тестов.

4. Численные эксперименты

В этой статье мы анализируем производительность пяти конфигураций модели, использующих три горизонтальных сетки: сетку с разрешением около 1° размерностью $6 \times 90 \times 90$, релевантную для базовых экспериментов проекта по сравнения моделей климата (CMIP), сетку с разрешением около $0,25°$ размерностью $6 \times 360 \times 360$, подходящую для HighResMIP (моделирование климата с высоким разрешением), и сетку с разрешением около 10 км $6 \times 1024 \times 1024$, которая соответствует текущим моделям численного прогноза погоды. Конфигурация сетки для систем численного прогноза погоды нового поколения (3–5 км) не рассматривается из-за ограниченности вычислительных ресурсов для проведения экспериментов в данный момент. Все сетки имеют 80 уровней по вертикали. Эксперименты проводятся с использованием как гидростатической, так и негидростатической версии модели на каждой из сеток, за исключением сетки 10 км, где тестируется только негидростатическая версия.

Применяется пространственная аппроксимация 4-го и 2-го порядка точности по горизонтали и вертикали соответственно. Интегрирование по времени производится с помощью вертикально-неявной схемы для негидростатических конфигураций и явной схемы РК4 для гидростатических. Величина шага по времени составляет 240 с, 60 с и 20 с для сеток с разрешением 1°, $0,25°$ и 10 км соответственно.

Все 5 рассматриваемых конфигураций модели были протестированы на идеализированных тестах (без блока физических процессов подсеточного масштаба) проекта по сравнению динамических ядер моделей атмосферы (DCMIP) [43], [12] и показали результаты, близкие к эталонным решениям. Анализ точности модели выходит за рамки данной статьи, поэтому мы приведем только рис. 1 с полем относительной завихренности ($\vec{k} \cdot \nabla \times \vec{v}$) в эксперименте «развитие бароклинной неустойчивости» [13] согласно решениям негидростатической модели. Видно, что по мере увеличения разрешения сетки структура поля становится более четкой и выявляются детали все более мелкого масштаба. Спиральная структура трех вихрей на сетке 10 км разрешается на уровне лучших результатов аналогичных моделей [14], [47].

Результаты исследования вычислительной эффективности и сильной масштабируемости рассматриваемых конфигураций модели представлены на рис. 2. Все эксперименты проводились на компьютерной системе Cray XC-40 Главного вычислительного центра Росгидромета. Вычислительные узлы этой системы соединены коммуникационной сетью Aries, каждый из них включает в себя два 18-ядерных процессора Intel Xeon E2697v4. Система имеет 936 узлов и пиковую производительность 1,293 Пфлопс. Мы используем компилятор Intel Fortran версии 19.1.2.254.

На сетке 1° негидростатическая и гидростатическая конфигурации модели масштабируются до 1440 вычислительных ядер (40 узлов Cray XC-40). В зависимости

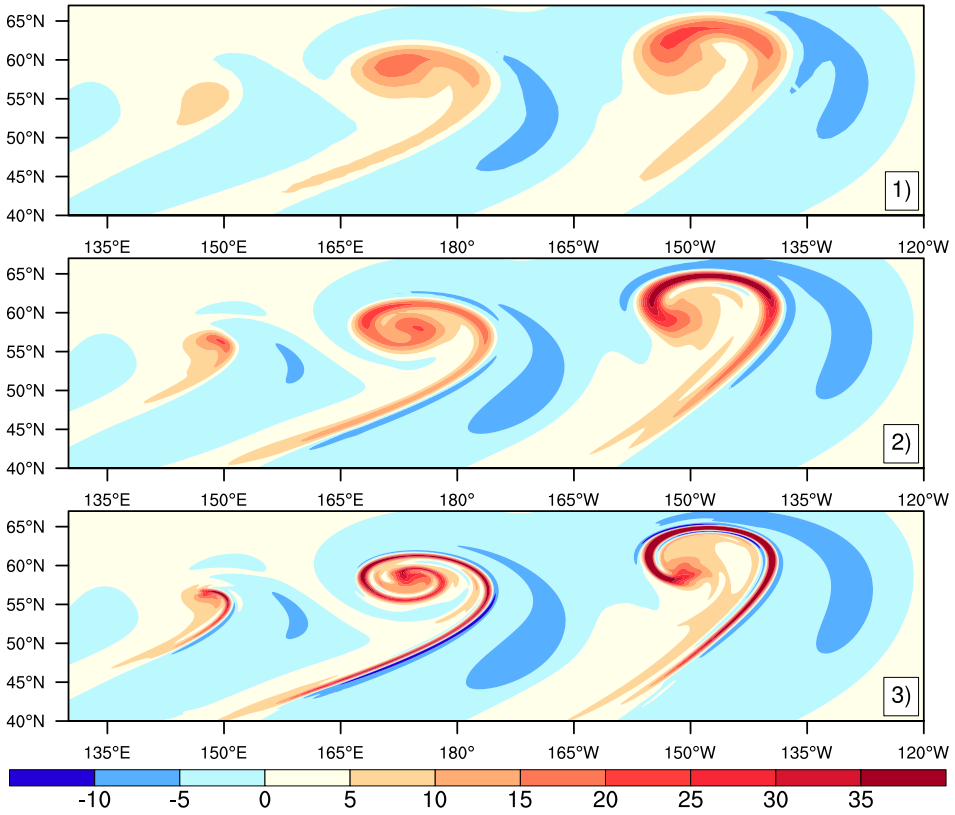


Рис. 1: Эксперимент «бароклинная неустойчивость», поле относительной завихренности ($10^{-5}\,\text{с}^{-1}$) на 9-й день по негидростатической модели на сетке с разрешением 1) 1°, 2) 0,25° и 3) 10 км.

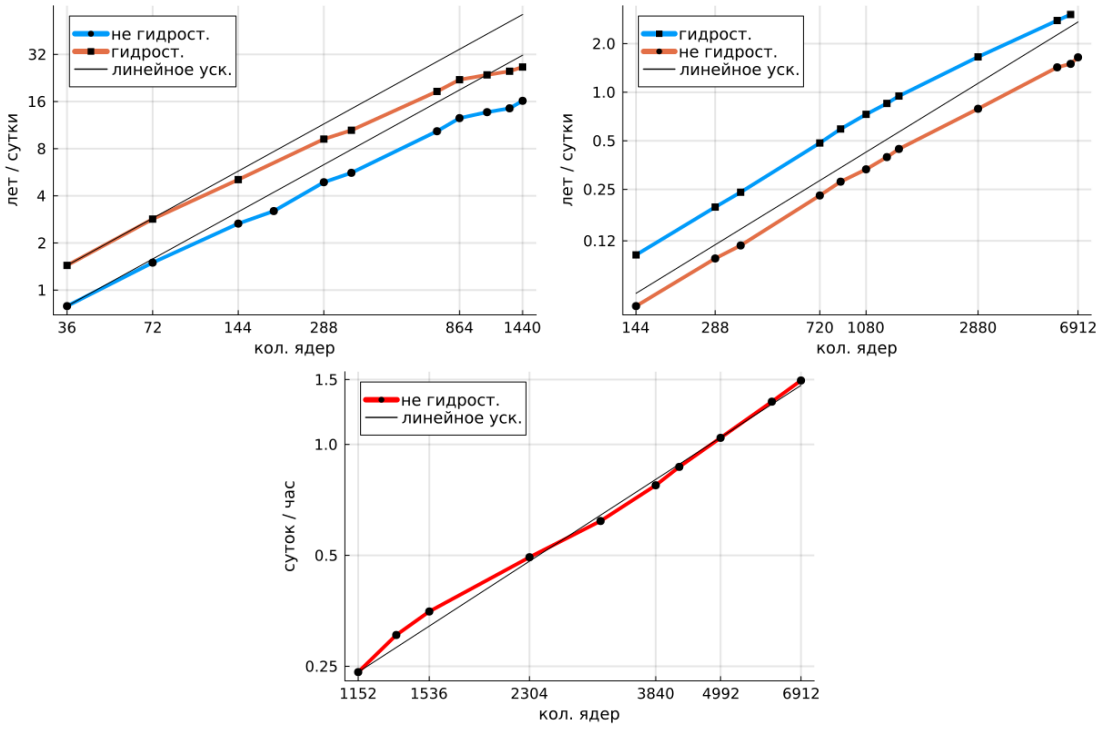


Рис. 2: Сильная масштабируемость конфигураций модели: сверху-слева – сетка 1°, сверху-справа – сетка 0,25°, снизу – сетка 10 км.

от количества ядер негидростатическая модель в $1,6$ – $1,8$ раз медленнее гидростатической. Параллельная эффективность (ускорение по отношению к 36 ядрам, разделенное на идеальное ускорение) при 1440 ядрах составляет 51% и 46% для негидростатической и гидростатической конфигураций соответственно. Параллельная эффективность превосходит модель INMCM6 с аналогичным горизонтальным разрешением [33], используемую в настоящее время для климатических экспериментов в России. Скорость вычислений на 1440 ядрах составляет 16 и 26 модельных лет в сутки для негидростатической и гидростатической конфигураций соответственно. Грубо оценить скорость вычислений в реальных приложениях с включенным пакетом параметризаций процессов подсеточного масштаба можно, разделив скорость вычислений без подсеточной физики на два (что справедливо для моделей SL-AV [39] и INMCM6 [33]).

Гидростатические и негидростатические конфигурации модели на сетке с разрешением $0,25^\circ$ эффективно масштабируются как минимум до 6912 ядер (максимально доступное нам количество на текущий момент). Ускорение при использовании 6912 ядер по сравнению со 144 ядрами составляет 72% и 68% линейного ускорения для негидростатической и гидростатической конфигурации соответственно. В абсолютных величинах, скорость вычислений составляет 1,6 и 3,0 года в сутки (негидростатическая/гидростатическая конфигурации).

Негидростатическая конфигурация модели на сетке 10 км масштабируется почти линейно до 6912 ядер. Такое количество ядер значительно ниже величины, при которой параллельная эффективность модели на сетке такой размерности должна снижаться. Скорость вычислений $1,5$ модельных суток за час расчета значительно ниже как требований оперативного прогноза (10–20 минут на сутки), так и текущей производительности модели ПЛАВ на аналогичной сетке (11 минут на сутки) [38]. Однако, существуют возможности для оптимизации текущего кода, и мы надеемся значительно ускорить модель, аналогично тому, как была ранее ускорена модель ПЛАВ [40].

5. Заключение

В ИВМ РАН и Гидрометцентре России разрабатывается модель динамики атмосферы нового поколения. Целью проекта является создание точного, быстрого и масштабируемого программного комплекса, который можно универсально использовать для широкого спектра задач. Решение этой проблемы невозможно без тщательной проработки вопросов организации программной инфраструктуры модели.

На данный момент нами разработаны основные концепции программного комплекса модели и реализованы его существенные части. Кроме того, мы реализовали алгоритмы численного решения уравнений негидростатической и гидростатической динамики атмосферы, которые были испытаны на общепринятых идеализированных задачах и показывали точность, соответствующую современному мировому уровню. Таким образом, блоки решения уравнений динамики подготовлены к финальному этапу разработки модели – соединению с пакетом параметризаций физических процессов подсеточного масштаба.

В этой статье мы исследовали производительность модели на массивно-параллельной вычислительной системе. Конфигурация модели, соответствующая следующему проекту по сравнению моделей климата (CMIP), эффективно масштабируется

до 1440 вычислительных ядер и, таким образом, превосходит INMCM6 – текущую российскую климатическую модель. Максимальная скорость вычислений – до 26 модельных лет в сутки, что является высоким показателем. В конфигурации, релевантной для проекта по сравнению моделей климата высокого разрешения (HighResMIP), представленная модель эффективно масштабируются как минимум до 6912 вычислительных ядер. Негидростатическая конфигурация модели, соответствующая численному прогнозу погоды (горизонтальное разрешение 10 км), масштабируется линейно (и слегка сверхлинейно) как минимум до 6912 вычислительных ядер (максимальное число ядер, доступное нам на текущий момент).

Нам удалось разработать гибкую программную платформу модели атмосферы, которая дает возможность эффективной разработки алгоритмов численного решения уравнений динамики для различных приложений моделирования атмосферы. Например, унификация программного кода между весьма различными блоками решения негидростатических и гидростатических уравнений динамики составляет около 90% (в строках). В то же время, такая степень гибкости не является бременем для вычислительной эффективности модели.

Существует определенный потенциал к дальнейшему повышению эффективности расчетов модели – реализация геометрического многосеточного алгоритма решения систем линейных алгебраических уравнений в сочетании с полулагранжевым методом, внедрение вычислений с одинарной точностью и адаптация кода для графических ускорителей. Этот потенциал необходимо использовать, чтобы реализовать практическую возможность расчетов конфигураций для численного прогноза погоды и моделирования климата с высоким разрешением.

Список литературы

[1] A Arakawa and V. Lamb. *Computational design of the basic dynamical processes of the UCLA general circulation model*, volume 17, pages 173–265. New York: Academic Press, 1977.

[2] Uri M. Ascher, Steven J. Ruuth, and Raymond J. Spiteri. Implicit-explicit runge-kutta methods for time-dependent partial differential equations. *Appl. Numer. Math.*, 25(2–3):151–167, November 1997.

[3] T. Davies, M. Cullen, A. Malcolm, M. Mawson, A. Staniforth, A. White, and N. Wood. A new dynamical core for the Met Office's global and regional modelling of the atmosphere. *Quart. J.Roy. Met.Soc.*, 131:1759–1782, 2005.

[4] Willem Deconinck, Peter Bauer, Michail Diamantakis, Mats Hamrud, Christian Kühnlein, Pedro Maciel, Gianmarco Mengaldo, Tiago Quintino, Baudouin Raoult, Piotr K. Smolarkiewicz, and Nils P. Wedi. ATLAS : A library for numerical weather prediction and climate modelling. *Computer Physics Communications*, 220:188 – 204, 2017.

[5] Radii Petrovich Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Computational Mathematics and Mathematical Physics*, 1(4):1092–1096, 1962.

[6] Longfei Gao, David C. Del Rey Fernández, Mark Carpenter, and David Keyes. SBP-SAT finite difference discretization of acoustic wave equations on staggered block-wise uniform grids. *Journal of Computational and Applied Mathematics*, 348:421–444, 2019.

[7] D. J. Gardner, J. E. Guerra, F. P. Hamon, D. R. Reynolds, P. A. Ullrich, and C. S. Woodward. Implicit–explicit (IMEX) Runge-Kutta methods for non-hydrostatic atmospheric models. *Geoscientific Model Development*, 11(4):1497–1515, 2018.

[8] G. Goyman and V. Shashkin. Implementation of elliptic solvers within ParCS parallel framework. *Communications in Computer and Information Science*, 1510:137–147, 2021.

[9] R. J. Haarsma, M. J. Roberts, P. L. Vidale, C. A. Senior, A. Bellucci, Q. Bao, P. Chang, S. Corti, N. S. Fučkar, V. Guemas, J. von Hardenberg, W. Hazeleger, C. Kodama, T. Koenigk, L. R. Leung, J. Lu, J.-J. Luo, J. Mao, M. S. Mizielinski, R. Mizuta, P. Nobre, M. Satoh, E. Scoccimarro, T. Semmler, J. Small, and J.-S. von Storch. High resolution model intercomparison project (HighResMIP v1.0) for CMIP6. *Geoscientific Model Development*, 9(11):4185–4208, 2016.

[10] M. Hortal. Aspects of the numerics of the ECMWF model. In *Procs. of the ECWMF Seminar, 7-11 September 1998*, pages 127–143, Reading, UK, 1999.

[11] S. Husain, C. Girard, A. Qaddouri, and A. Plante. A new dynamical core of the Global Environmental Multiscale (GEM) model with a height-based terrain-following vertical coordinate. *Monthly Weather Review*, 147(7):2555 – 2578, 01 Jul. 2019.

[12] C. Jablonowski, P. H. Lauritzen, M. Taylor, and R. D. Nair. Idealized test cases for the dynamical cores of atmospheric general circulation models. *A proposal for the NCAR ASP 2008 summer colloquium*, 2008.

[13] C. Jablonowski and D. L. Williamson. A baroclinic instability test case for atmospheric model dynamical cores. *Quarterly Journal of the Royal Meteorological Society*, 132:2943 – 2975, 2006.

[14] C. Jablonowski and D.L. Williamson. A baroclinic wave test case for dynamical cores of general circulation models: Model intercomparisons. ncar technical note ncar/tn-469+str. 2006.

[15] J. Kent, T. Melvin, and G. A. Wimmer. A mixed finite element discretisation of the shallow water equations. *Geoscientific Model Development Discussions*, 2022:1–17, 2022.

[16] C. Kühnlein, W. Deconinck, R. Klein, S. Malardel, Z. P. Piotrowski, P. K. Smolarkiewicz, J. Szmelter, and N. P. Wedi. FVM 1.0: a nonhydrostatic finite-volume dynamical core for the IFS. *Geoscientific Model Development*, 12(2):651–676, 2019.

[17] Peter Lauritzen, Ramachandran Nair, Adam Herrington, P. Callaghan, Steve Goldhaber, John Dennis, Julio Bacmeister, Brian Eaton, Colin Zarzycki, Mark Taylor, Paul Ullrich, T. Dubos, Andrew Gettelman, Richard Neale, B. Dobbins, Kevin Reed, Cecile Hannay, Brian Medeiros, James Benedict, and J.J. Tribbia. NCAR release

of CAM-SE in CESM2.0: A reformulation of the spectral-element dynamical core in dry-mass vertical coordinates with comprehensive treatment of condensates and energy. *Journal of Advances in Modeling Earth Systems*, 10, 05 2018.

[18] Tomas Lundquist, Fredrik Laurén, and Jan Nordström. A multi-domain summation-by-parts formulation for complex geometries. *Journal of Computational Physics*, 463:111269, 2022.

[19] Gianmarco Mengaldo, Andrzej Wyszogrodzki, Michail Diamantakis, Sarah-Jane Lock, Francis X. Giraldo, and Nils P. Wedi. Current and emerging time-integration strategies in global numerical weather and climate prediction. *Archives of Computational Methods in Engineering*, 26:663 – 684, 2019.

[20] J. Mouallem, L. Harris, and R. Benson. Multiple same-level and telescoping nesting in GFDL's dynamical core. *Geoscientific Model Development*, 15(11):4355–4371, 2022.

[21] E. Müller and R. Scheichl. Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction. *Quart. J.Roy. Met.Soc.*, 140(685):2608–2624, 2014.

[22] R.D. Nair, S. J. Thomas, and R. D. Loft. A discontinuous galerkin global shallow water model. *Mon. Wea. Rev.*, 133:876 – 888, 2005.

[23] Jens Niegemann, Richard Diehl, and Kurt Busch. Efficient low-storage Runge-Kutta schemes with optimized stability regions. *Journal of Computational Physics*, 231(2):364–372, 2012.

[24] M. Rančić, R. J. Purser, and F. Mesinger. A global shallow-water model using an expanded spherical cube: Gnomonic versus conformal coordinates. *Quarterly Journal of the Royal Meteorological Society*, 122(532):959–982, 1996.

[25] V. Shashkin, R. Fadeev, and M Tolstykh. 3D conservative cascade semi-Lagrangian transport scheme using reduced latitude-longitude grid (CCS-RG). *Journal of Computational Physics*, 305:700–721, 2016.

[26] V. V. Shashkin and G. S. Goyman. Semi-Lagrangian shallow water equations solver on the cubed-sphere grid as a prototype of new-generation global atmospheric model. *Journal of Physics: Conference Series*, 1740(1):012073, 2021.

[27] Vladimir V. Shashkin. Stability analysis of implicit semi-Lagrangian methods for numerical solution of non-hydrostatic atmospheric dynamics equations. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 36(4):239–253, 2021.

[28] Vladimir V. Shashkin and Gordey S. Goyman. Semi-Lagrangian exponential time-integration method for the shallow water equations on the cubed sphere grid. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 35(6):355–366, 2020.

[29] Vladimir V. Shashkin, Gordey S. Goyman, and Mikhail A. Tolstykh. Summation-by-parts finite-difference shallow water model on the cubed-sphere grid. part I: Non-staggered grid. *Journal of Computational Physics*, 474:111797, 2023.

[30] A. Staniforth and J. Côté. Semi-Lagrangian integration schemes for atmospheric models – a review. *Monthly Weather Review*, 119:2206–2223, 1991.

[31] A. Staniforth and J. Thuburn. Horizontal grids for global weather and climate prediction models: a review. *Quarterly Journal of the Royal Meteorological Society*, 138:1 – 26, 2012.

[32] Bo Strand. Summation by parts for finite difference approximations for d/dx. *Journal of Computational Physics*, 110(1):47–67, 1994.

[33] M. Tarasevich, A. Sakhno, D. Blagodatskikh, R. Fadeev, E. Volodin, and A. Gritsun. Scalability of the INM RAS Earth system model. In *Lecture Notes in Computer Science*, pages 202–216. 2024.

[34] Mark A. Taylor and Aimé Fournier. A compatible and conservative spectral element method on unstructured grids. *Journal of Computational Physics*, 229(17):5879–5895, 2010.

[35] John Thuburn. Some basic dynamics relevant to the design of atmospheric model dynamical cores. In Peter Lauritzen, Christiane Jablonowski, Mark Taylor, and Ramachandran Nair, editors, *Numerical Techniques for Global Atmospheric Models*, pages 3–27. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[36] John Thuburn and T.J. Woollings. Vertical discretizations for compressible Euler equation atmospheric models giving optimal representation of normal modes. *Journal of Computational Physics*, 203:386–404, 03 2005.

[37] M. Tolstykh, V. Shashkin, R. Fadeev, and G. Goyman. Vorticity-divergence semi-Lagrangian global atmospheric model SL-AV20: dynamical core. *Geoscientific Model Development*, 10(5):1961–1983, 2017.

[38] Mikhail Tolstykh, Gordey Goyman, Ekaterina Biryucheva, Vladimir Shashkin, and Rostislav Fadeev. Reduced precision computations in the SL-AV global atmosphere model. In Vladimir Voevodin, Sergey Sobolev, Mikhail Yakobovskiy, and Rashit Shagaliev, editors, *Supercomputing*, pages 190–201, Cham, 2023. Springer Nature Switzerland.

[39] Mikhail Tolstykh, Gordey Goyman, Rostislav Fadeev, and Vladimir Shashkin. Implementation of SL-AV global atmosphere model with 10 km horizontal resolution. In *Supercomputing: 6th Russian Supercomputing Days, RuSCDays 2020, Moscow, Russia, September 21–22, 2020, Revised Selected Papers 6*, pages 216–225. Springer, 2020.

[40] Mikhail A. Tolstykh, Rostislav Yu. Fadeev, Vladimir V. Shashkin, and Gordey S. Goyman. Improving the computational efficiency of the global SL-AV numerical weather prediction model. *Supercomputing Frontiers and Innovations*, 8(4):11–23, Feb. 2022.

[41] Ilya D. Tretyak, Gordey S. Goyman, and Vladimir V. Shashkin. Multiresolution approximation for shallow water equations using summation-by-parts finite differences. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 38(6):393–407, 2023.

[42] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2000.

[43] P. A. Ullrich, C. Jablonowski, J. Kent, P. H. Lauritzen, R. D. Nair, and M. A. Taylor. Dynamical core model intercomparison project (DCMIP) test case document. 2012.

[44] Paul A. Ullrich, Christiane Jablonowski, and Bram van Leer. High-order finite-volume methods for the shallow-water equations on the sphere. *Journal of Computational Physics*, 229(17):6104 – 6134, 2010.

[45] J.B. White III and J.J. Dongarra. High-performance high-resolution tracer transport on a sphere. *Journal of Computational Physics*, 230:6778 – 6799, 2011.

[46] G. Zängl, D. Reinert, and F. Prill. Grid refinement in ICON v2.6.4. *Geoscientific Model Development*, 15(18):7153–7176, 2022.

[47] G. Zangl, D. Reinert, P. Ripodas, and M. Baldauf. The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core. *Quarterly Journal of the Royal Meteorological Society*, 141(687):563–579, 2015.

[48] M. Zerroukat and T. Allen. SLIC: A Semi-Lagrangian implicitly corrected method for solving the compressible Euler equations. *Journal of Computational Physics*, 421:109739, 2020.

[49] M. Zerroukat and T. Allen. On the corners of the cubed-sphere grid. *Quarterly Journal of the Royal Meteorological Society*, 148(743):778–783, 2022.

Ускорение модели ПЛАВ в версии для среднесрочного прогноза погоды

Р.Ю. Фадеев, Г.С. Гойман, М.А. Толстых, В.В. Шашкин

Институт вычислительной математики
им. Г.И. Марчука РАН, Москва, Россия
ФГБУ «Гидрометцентр России», Москва, Россия
Московский физико-технический институт (НИУ), Москва, Россия

Глобальная модель атмосферы ПЛАВ в настоящее время применяется в Гидрометцентре России для оперативного среднесрочного прогноза погоды. Одним из важнейших направлений работ по развитию связанного с ПЛАВ программного комплекса является уменьшение времени, затрачиваемого на получение прогностической продукции. В рамках данной работы показано, что использование специализированного программного обеспечения для работы с файловой системой позволяет ускорить расчет одного среднесрочного прогноза на 15% и более. Данный результат получен для двух технологий: среднесрочный детерминированный прогноз на основе ПЛАВ10 и ансамблевый среднесрочный прогноз на основе ПЛАВ20.

Ключевые слова: численный прогноз погоды, модель ПЛАВ, ввод-вывод, параллельные вычисления

1. Введение

Всемирная метеорологическая организация не налагает ограничений на время расчета одного среднесрочного прогноза – это является прерогативой гидрометслужбы каждой страны. Например, в Европейском центре среднесрочных прогнозов погоды (ЕЦСПП) время расчета прогноза на 24 часа ограничено 10 минутами. В Гидрометцентре России это ограничение более мягкое – время расчета прогноза на сутки ограничивается 20 минутами. Важно отметить, что данное ограничение относится к численной модели динамики атмосферы и не включает время работы системы усвоения данных наблюдения, программ подготовки начального состояния для последующих расчетов и обработки результатов моделирования. Таким образом, совершенствование модели с целью ускорения вычислений на ее основе является практически важной задачей, решение которой реализуется различными способами.

При повышении пространственного разрешения значительно увеличивается объем данных, генерируемых прогностическими моделями. Повышение эффективности операций ввода/вывода может быть достигнуто, в том числе, применением специализированного программного обеспечения (ПО), таких как Message passing interface–input/output [1], netCDF [2], parallel netCDF [3], parallel I/O [4], XML Input/Output Server [5]. Ускорение операций ввода/вывода при использовании этих библиотек достигается за счет распараллеливания процесса взаимодействия с файловой системой. При этом вычисления модели и операции ввода/вывода всё равно происходят по-

следовательно, что означает приостановку расчетов до момента завершения записи данных в файловую систему.

Дальнейшее увеличение производительности и масштабируемости моделей может быть достигнуто за счет выделения части процессов исключительно для выполнения операций ввода/вывода. При этом расчетным процессам модели не нужно ожидать завершения выполнения операций записи на диск, им нужно только отправлять данные на узлы, ответственные за выполнение операций ввода/вывода (т.н. сервера ввода-вывода). К настоящему моменту на основе такого подхода реализовано несколько универсальных программных решений: ADIOS [6] и Damaris [7]. В задачах прогноза погоды и моделирования изменений климата применяются CDI-pio [8], Climate Fast Input/Output (FIO) [9] и XIOS [10], а также подходы собственной разработки [11, 12, 13].

В 2023 году в качестве основного численного метода оперативного детерминированного среднесрочного прогноза погоды Гидрометцентра России была внедрена модель ПЛАВ10, отличающаяся повышенным до 10 км горизонтальным разрешением (более чем вдвое) по сравнению с ее предыдущей версией ПЛАВ20. К концу 2024 года ожидается представление к оперативным испытаниям ансамблевой технологии среднесрочного прогнозирования на основе ПЛАВ20. Результаты параллельной профилировки ПЛАВ10 и ПЛАВ20 выявили, что доля времени выполнения операций записи выходных данных составляет заметную часть от времени всех вычислений. С целью ускорения работы с файловой системой в программный комплекс ПЛАВ10 было внедрено разработанное ранее ПО ParIO [14]. Использование внешних серверов ввода-вывода ParIO позволили не только повысить производительность модели, но и сократить объем диагностического вывода модели за счет использования специализированного функционала этого ПО. Данный подход был также реализован в ПЛАВ20, что потребовало адаптации сопутствующего модели ПО для использования файлов с данными в формате NetCDF.

В рамках данной работы обсуждаются особенности внедрения ПО ParIO в программный комплекс модели ПЛАВ и влияние использования выделенных серверов для работы с файловой системой на скорость расчетов по этой модели. В Разделе 2 обсуждается технология среднесрочного прогноза на основе ПЛАВ. Особенности программной реализации ПЛАВ в конфигурации с ПО ParIO обсуждается в Разделе 2.1. Анализ результатов численных экспериментов проводится в Разделе 3.

2. Система среднесрочного прогноза на основе модели ПЛАВ

Модель атмосферы ПЛАВ имеет несколько конфигураций, отличающихся, главным образом, горизонтальным разрешением, числом уровнем по вертикали и шагом по времени. Так, например, модель среднесрочного прогноза на основе ПЛАВ10 [15] внедрена в оперативную практику Гидрометцентра России в 2023 году. Модель ПЛАВ10 заменила собой ПЛАВ20, на основе которой сейчас разрабатывается новая система ансамблевого среднесрочного прогноза погоды [16].

Помимо повышения точности прогноза и ресурсоемкости прогностической технологии переход с ПЛАВ20 на ПЛАВ10 привел к увеличению объема оперируемых моделью данных (см. Табл. 1). Размер файла с начальными данными вырос почти в

10 раз: с 2.8 Гб в ПЛАВ20 до 22.7 Гб в ПЛАВ10. Аналогичная ситуация с выходными данными модели на стандартных уровнях давления: размер этих файлов увеличился с 0.7 Гб до 3.3 Гб.

Таблица 1: Характиристики версий модели ПЛАВ

Идентификатор модели	ПЛАВ20	ПЛАВ10
Число узлов по долготе	1600	3600
Число узлов по широте	865	1945
Число уровней по вертикали	51	104
Шаг по времени, с.	450	270
Число шагов по времени в сутки	192	320
Размер файла с контрольной точкой, Гб.	2.8	22.7
Размер файла с выходной продукцией, Гб,	0.7	3.3

Технология среднесрочного прогнозирования на основе модели ПЛАВ работает в циклическом режиме, где шаг цикла составляет 6 часов и включает последовательную работу системы усвоения и модели ПЛАВ. Данный цикл иллюстрируется на Рис. 1. На первом этапе с использованием данных наблюдений, реанализа и первого приближения модели происходит подготовка начального состояния для ПЛАВ. После этого модель рассчитывает прогноз на 10 суток. В процессе счета каждые 3 часа

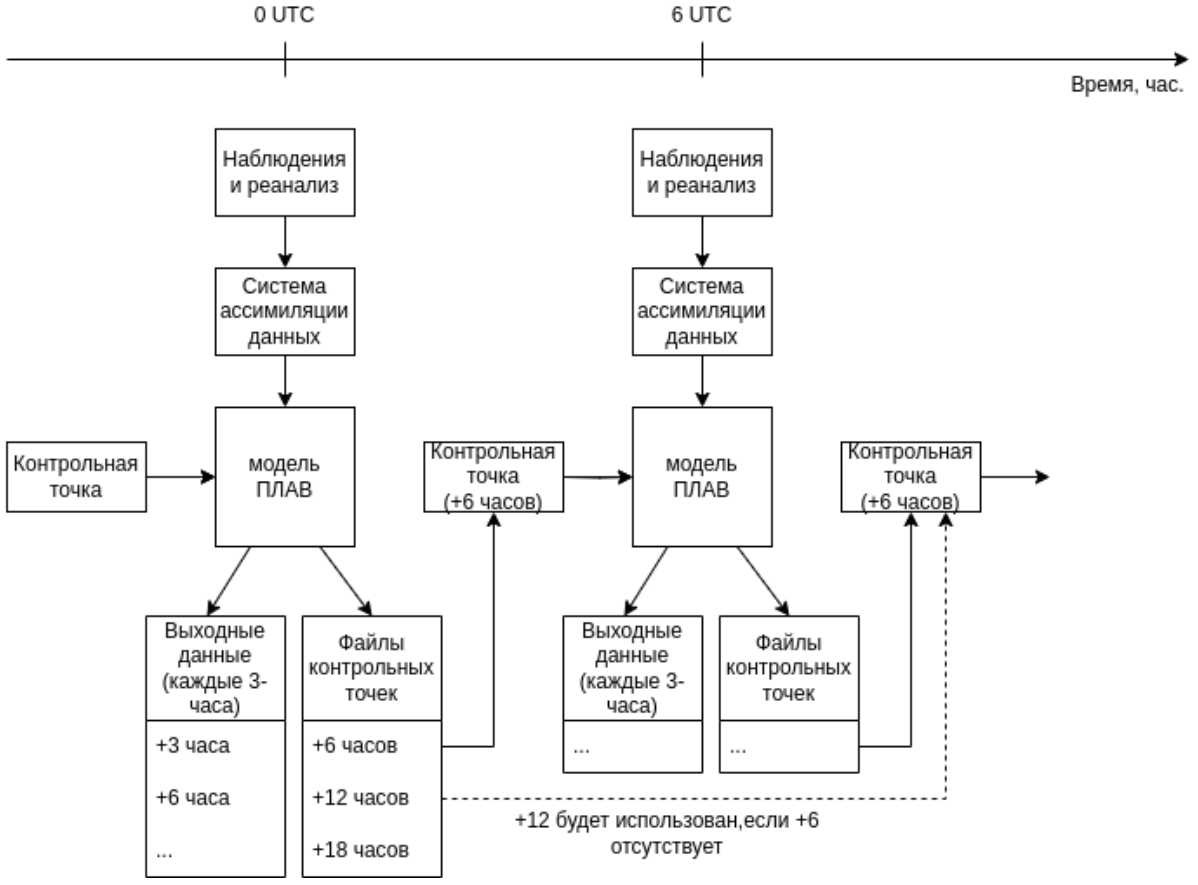


Рис. 1: Цикл усвоение–прогноз в системе оперативного среднесрочного прогноза погоды на основе ПЛАВ

модельного времени генерируются выходные файлы на сетке модели и на стандартных изобарических уровнях по вертикали. Размер данного типа файлов в Табл. 1 указан в последней строке.

Для подготовки начальных данных ПЛАВ для прогноза, стартующего спустя 6 часов, модель в процессе счета записывает файл, содержащий полную информацию о состоянии атмосферы на расчетной сетке по горизонтали и на всех модельных уровнях по вертикали. С целью повышения надежности работы циклической системы усвоение–прогноз ПЛАВ также записывает эти файлы с данными, соответствующими 12– и 18–часовой заблаговременности прогноза. Таким образом, модель ПЛАВ20 записывает в течение первых модельных суток около 8.4 Гб, а в течение вторых суток прогноза – 5.6 Гб. В случае ПЛАВ10 с существенно более высоким разрешением расчетной сетки общий размер вывода в файловую систему увеличивается до 49.1 Гб в первые сутки и до 26.4 Гб, в последующие.

Запись данных осуществляется в блокирующем режиме, приостанавливающем расчеты на время работы с файловой системой. Результаты профилировки программного кода ПЛАВ20 показали, что модель в версии для оперативного прогноза тратит от 31% до 35% времени расчета первого дня на запись данных в зависимости от способа работы с файловой системой: мастер–процесс и параллельный вывод на основе NetCDF, соответственно. В случае ПЛАВ10 технология вывода на основе мастер–процесса не применима из-за большого объема оперируемых данных, поэтому в прогнозе применяется только технология на основе NetCDF. Доля времени записи выходной продукции и контрольных точек по сравнению с временем расчета первых суток также существенна: около 27%. Во второй и последующие дни в процессе счета в файловую систему записывается существенно меньший объем данных, поэтому доля времени работы соответствующих процедур уменьшается до 14% в ПЛАВ20 и 11% в ПЛАВ10 (по сравнению со временем расчета вторых суток прогноза). Таким образом, ускорение вычисления среднесрочного прогноза на основе ПЛАВ может быть достигнуто за счет оптимизации работы с файловой системой. Для этих целей в программный комплекс ПЛАВ было внедрено разработанное ранее ПО ParIO.

2.1. Особенности использования ParIO в ПЛАВ

Для работы с файловой системой в ПЛАВ используется программное обеспечение HDF5 и NetCDF, подключаемые к модели в виде библиотек на этапе ее сборки. В оперативной технологии ПЛАВ20 до сих пор применяется устаревший подход на основе мастер–процесса, где расчетные данные собираются на один MPI–процесс, который осуществляет запись в файловую систему в режиме прямого доступа. Отличие прогностических технологий ПЛАВ10 и ПЛАВ20 с точки зрения оперирования данными объясняется невозможностью использования подхода мастер–процесс в ПЛАВ10. Поэтому вся технологическая цепочка ПЛАВ10 (включая пре– и постпроцессинг данных) были реализованы с поддержкой формата данных NetCDF.

Для ускорения вычислений на основе модели ПЛАВ в ее программный код была внедрена возможность использования разработанного ранее программного обеспечения ParIO. В этом случае предназначенные для записи данные пересылаются на выделенный MPI–коммуникатор, основным и единственным назначением которого является работа с файловой системой. Таким образом, в идеальном случае, расчеты

модели должны приостанавливаться на время, соответствующее пересылке с расчетных MPI–процессов на MPI–процессы коммуникатора ParIO. Особенность такого подхода заключается в возможности вывода не только полных двух и трехмерных полей на расчетной сетке ПЛАВ, но и их части. В случае ПЛАВ10 такая возможность особенно полезна в работах по совершенствованию и тестированию модели.

Работа с ParIO включает две основные стадии:

- регистрация переменной, включая передачу ParIO соответствующего ей уникального идентификатора, указателя на массив данных и его пределы на каждом MPI–процессе.

- вызов процедуры, ответственной за запись данных. На этом этапе в ParIO передаются названия выходных файлов и список уникальных идентификаторов для записи в эти файлы.

В настоящее время процедура вычисления одного шага по времени включает несколько этапов:

- **step1**: первая часть расчета динамики атмосферы;

- **out1**: процедура записи контрольной точки модели, процедура записи трехмерных полей в файлы с выходной продукцией модели;

- **step2**: вторая часть расчета динамики атмосферы;

- **out2**: процедура подготовки и записи двухмерных полей данных в файлы с выходной продукцией ПЛАВ;

- **step3**: завершение вычислений одного шага по времени.

Отметим, что выполнение этапов **out1** и **out2** происходит в соответствии с настройками модели (каждые 3 и/или 6 часов). Внедрение вывода данных ПЛАВ на основе ParIO потребовало введение в конец этого списка еще одной процедуры, которую мы обозначим с помощью **pio_flush**. Назначением данной процедуры является передача ParIO данных с целью последующей их записи на диск. Процедуры **out1** и **out2** в данном случае выполняют подготовку передаваемых в ParIO данных. Отметим, что в процедуре **out1** осуществляется создание выходного NetCDF файла с записью информации о расчетной сетке.

3. Результаты численных экспериментов

Особенность работы прогностической системы среднесрочного прогноза погоды на основе ПЛАВ заключается в необходимости сохранять два типа файлов, соответствующих выходной продукции и контрольным точкам, необходимых для последующего запуска модели через 6, 12 или 18 часов спустя. Это означает неравномерную интенсивность взаимодействия с файловой системой в процессе счета модели. Поэтому для оценки эффекта от внедрения ПО ParIO для записи выходных файлов в рамках численных экспериментов с продолжительностью интегрирования модели ПЛАВ в 48 часов замерялось время расчета первого и второго дня по отдельности. Изучались версии модели, применяемые для среднесрочного прогноза погоды (ПЛАВ10 и

ПЛАВ20). Параллельные конфигурации этих моделей выбирались таким образом, чтобы обеспечить максимальную скорость счета в заданном диапазоне доступных процессорных ядер и выбранной стратегии работы с файловой системой.

На рис. 2 приводится зависимость скорости счета ПЛАВ20 (выраженной в числе модельных дней, которые способна рассчитать ПЛАВ за 10 минут) в зависимости от числа задействованных процессорных ядер системы Cray XC40–LC, установленной в Главном вычислительном центре Росгидромета. Рисунок 2а соответствует первому дню интегрирования модели, рис. 2б – второму дню. Стратегии работы с файловой системой отличаются цветом точек: зеленый соответствует подходу мастер–процесс, красный – ПО NetCDF, черный – ParIO и NetCDF. Кружками меньшего диаметра на рис. 2 представлена масштабируемость ПЛАВ20 без учета операций вывода данных в файловую систему. Таким образом, разница между кружками большого и малого диаметров отражает накладные расходы, обусловленные необходимостью записи промежуточных результатов моделирования. Сплошная линия на обоих рисунках соответствует идеальному ускорению модели с ростом вычислительных ядер.

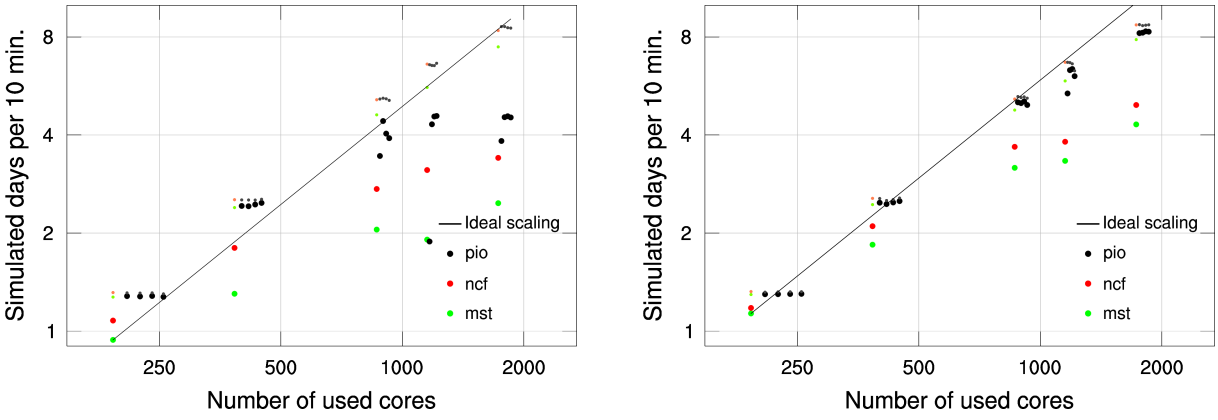


Рис. 2: Скорость расчета первого (а) и второго (б) дней прогноза в зависимости от числа процессорных ядер, делегированных ПЛАВ20

На рис. 2а можно видеть, что хуже всего масштабируется модель ПЛАВ20 с выводом данных на основе мастер–процесса. Несколько лучший результат показывает вывод с помощью ПО NetCDF. Максимальная скорость счета достигается при использовании ПО ParIO. В оперативной версии ПЛАВ для детерминированного среднесрочного прогноза погоды ускорение модели в первый день составляет 25% и 54% по сравнению с выводом на основе мастер–процесса. Для второго дня (рис. 2б) объем записываемых в файловую систему существенно меньше, поэтому ускорение более скромное: 14% и 37%, соответственно. Отметим, что ParIO в этой конфигурации ПЛАВ20 делегировано 32 MPI–процесса, что составляет около 3.7% от общего числа ядер, используемых ПЛАВ20 (всего делегировано – 864 процессорных ядра).

В версии ПЛАВ20, использующей 192 процессорных ядра, среднее время расчета одного шага модели по времени в течение первого дня в режиме с мастер–процессом составляет 3.3 с, при записи данных на основе NetCDF – 2.9 с, с внешними делегатами – 2.4 с. Таким образом, ускорение модели в случае использования NetCDF и ParIO

вместо мастер–процесса составляет 13% и 27%, соответственно. Ускорение расчетов в течение второго дня – 0.4% и 12%, сответственно. Отметим, что данная конфигурация ПЛАВ20 является наиболее вероятным кандидатом для использования в ансамблевой системе среднесрочного прогноза погоды. Такой подход позволит путем оценки дисперсии членов ансамбля рассчитать вероятность реализации метеорологического события или явления. Для этих целей потребуется выполнить несколько десятков расчетов на основе ПЛАВ20, поэтому даже небольшое ускорение модели позволит быстрее получать результат.

При сравнении кружков малого и большого диаметров на рис. 2б можно заметить, что накладные расходы ParIO незначительны по сравнению с накладными расходами при использовании других подходов к записи выходных данных. В первый день интегрирования наблюдается другая ситуация. На рис. 2а видно, что накладные расходы ParIO быстро растут с увеличением числа используемых ПЛАВ20 процессорных ядер. Начиная с конфигурации, использующей 864 процессорных ядра, ускорение модели с ParIO с ростом объема предоставленных ПЛАВ вычислительных ресурсов практически не происходит, в то время как модель продолжает ускоряться в версиях, использующих NetCDF и мастер–процесс. Такой результат объясняется достижением предела производительности файловой системы Lustre вычислительной системы Cray XC40–LC. Модель генерирует данные для записи быстрее, чем файловая система способна их записать.

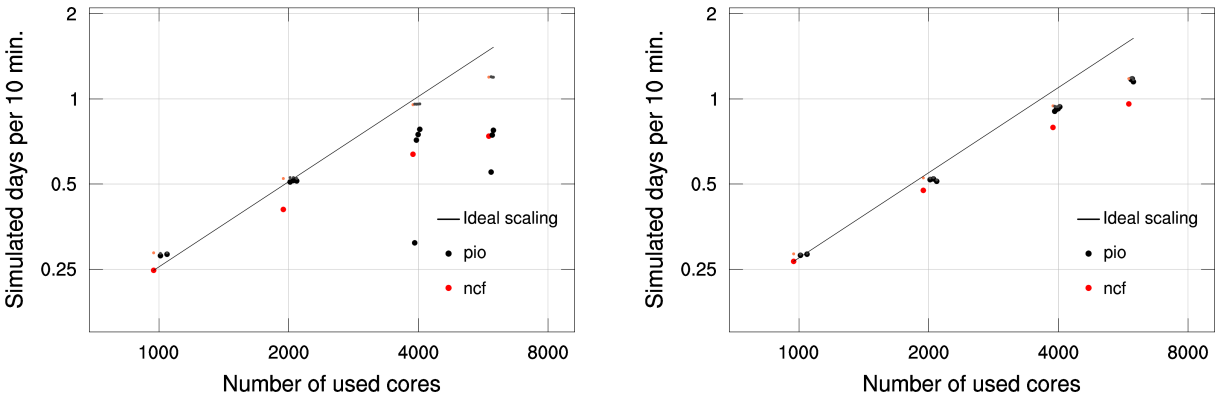


Рис. 3: Скорость расчета первого (а) и второго (б) дней прогноза в зависимости от числа процессорных ядер, делегированных ПЛАВ10

Аналогичный результат насыщения наблюдается в версии ПЛАВ10 для детерминированного среднесрочного прогноза погоды. На рис. 3а можно видеть, что используемая сейчас в Гидрометцентре России параллельная конфигурация модели с 3888 процессорными ядрами является предельной по скорости счета в первый день интегрирования: дальнейшее увеличение числа делегированных ПЛАВ процессорных ядер приводит к незначительному росту ее производительности. В то время как без накладных расходов, обусловленных записью данных, модель имеет потенциал к ускорению (кружки с малым диаметром). Ускорение модели в конфигурации с 3888 процессорными ядрами, где еще 108 ядер (18 MPI–процессов) делегируется системе ParIO, составляет около 15% (среднее время расчета одного шага по времени уменьшается с 2.94 с до 2.51 с). Близкие значения ускорения модели за счет

перехода на ParIO получены для второго дня интегрирования модели: среднее время расчета одного шага ПЛАВ10 по времени уменьшается с 2.37 с до 2 с, что составляет около 14%. Рост объема используемых вычислительных ресурсов при этом около 2.8%.

3.1. Заключение

Прогноз погоды на основе математической модели должен быть практически полезен. Поэтому наряду с точностью прогноза к таким моделям предъявляют строгие требования по скорости счета. Ориентиром могут являться модели ведущих метеорологических центров мира, где расчет прогноза погоды на 24 часа не превышает 10 минут. В данной работе показано, что использование внешнего MPI–коммуникатора для агрегации с целью последующей записи в файловую систему выходных данных ПЛАВ10 позволяет ускорить модель на 15%. Скорость расчетов второго и последующих дней прогноза с использованием оперативной версии ПЛАВ10 для детерминированного среднесрочного прогноза погоды в этом случае вплотную приближается к производительности 24 часа за 10 минут. Скорость расчета первого дня прогноза на основе ПЛАВ10 тоже увеличивается, но остается далекой от этой производительности. Однако, переход на ParIO для вывода данных в файловую систему позволяет применить для прогноза вдвое менее ресурсоемкую версию модели ПЛАВ10: скорость счета этой модели во все дни составляет величину около 24 часов за 20 минут, что является обязательным требованием Гидрометцентра России, предъявляемым к моделям для оперативного среднесрочного прогноза погоды.

К направлениям дальнейшей работы можно отнести создание версии ParIO, допускающей работу в виде отдельно исполняемого файла. Такой подход позволит индивидуально конфигурировать параметры окружения для ParIO и, вероятно, уменьшить число используемый этой системой вычислительных ресурсов за счет изменения значения OMP_NUM_THREADS, которое равно 4 в ПЛАВ20 и 6 для ПЛАВ10. Еще одной возможностью для дальнейшего ускорения модели атмосферы является использование нескольких независимых экземпляров ParIO для выполнения специализированных задач по работе с файловой системой (запись в файловую систему контрольных точек и выходной продукции, например).

Работа выполнена при поддержке Российского научного фонда (РНФ), проект No. 21-71-30023 и Отделения математического центра мирового уровня «Московский центр фундаментальной и прикладной математики» в ИВМ РАН (Соглашение с Министерством науки и высшего образования РФ No. 075-15-2022-286.

Список литературы

[1] P. Corbett, D. Feitelson, S. Fineberg, Y. Hsu, B. Nitzberg, J.-P. Prost, M. Snirt, B. Traversat, and P. Wong, "Overview of the mpi-io parallel i/o interface," *Input/Output in Parallel and Distributed Computer Systems*, pp. 127–146, 1996.

[2] "Network common data form (netcdf) home page."
`https://www.unidata.ucar.edu/software/netcdf/`
Accessed: 24.04.2024.

[3] J. Li, W.-k. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale, "Parallel netcdf: A high-performance scientific i/o interface," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, p. 39, 2003.

[4] J. M. Dennis, J. Edwards, R. Loy, R. Jacob, A. A. Mirin, A. P. Craig, and M. Vertenstein, "An application-level parallel i/o library for earth system models," *The International Journal of High Performance Computing Applications*, vol. 26, no. 1, pp. 43–53, 2012.

[5] "Xios home page."
`https://forge.ipsl.jussieu.fr/ioserver`
Accessed: 24.04.2024.

[6] C. Jin, S. Klasky, S. Hodson, W. Yu, J. Lofstead, H. Abbasi, K. Schwan, M. Wolf, W.-k. Liao, A. Choudhary, *et al.*, "Adaptive io system (adios)," *Cray User's Group*, 2008.

[7] M. Dorier, G. Antoniu, F. Cappello, M. Snir, and L. Orf, "Damaris: How to efficiently leverage multicore parallelism to achieve scalable, jitter-free i/o," in *2012 IEEE International Conference on Cluster Computing*, pp. 155–163, IEEE, 2012.

[8] M. Hanke, J. Biercamp, C. O. Escamilla, and T. Jahns, "Deike kleberg, paul selwood, and steve mullerworth," *Deliverable 7.3–Reference implementations of Parallel I/O and of I/O Server*.

[9] X. Huang, W. Wang, H. Fu, G. Yang, B. Wang, and C. Zhang, "A fast input/output library for high-resolution climate models," *Geoscientific Model Development*, vol. 7, no. 1, pp. 93–103, 2014.

[10] S. Joussaume, A. Bellucci, J. Biercamp, R. Budich, A. Dawson, M.-A. Foujols, B. Lawrence, L. Linardikis, S. Masson, Y. Meurdesoif, *et al.*, "Modelling the earth's climate system: data and computing challenges," in *2012 SC Companion: High Performance Computing, Networking, Storage and Analysis (SCC)*, vol. 1, pp. 2325–2356, IEEE Computer Society, 2012.

[11] "Parallelization of hirlam: Model parallelization and asynchronous i/o."
`https://www.ecmwf.int/sites/default/files/elibrary/2004/14140-parallelization-hirlam-model-parallelization-and-asynchronous-io.pdf`
Accessed: 24.04.2024.

[12] V. V. Kalmykov and R. A. Ibrayev, "A framework for the ocean-ice-atmosphere-land coupled modeling on massively-parallel archi-tectures," *Vychisl. Metody Programm.*, vol. 14, pp. 88–95, 2013.

[13] V. V. Kalmykov, R. A. Ibrayev, M. N. Kaurkin, and K. V. Ushakov, "Compact modeling framework v3.0 for high-resolution global ocean–ice–atmosphere models," *Geoscientific Model Development*, vol. 11, no. 10, pp. 3983–3997, 2018.

[14] M. Tolstykh, G. Goyman, R. Fadeev, *et al.*, "Structure and algorithms of SL-AV atmosphere model parallel program complex," *Lobachevskii J Math*, vol. 39, p. 587–595, 2018.

[15] M. Tolstykh, G. Goyman, R. Fadeev, and V. Shashkin, "Implementation of SL-AV global atmosphere model with 10 km horizontal resolution," in *Supercomputing* (V. Voevodin and S. Sobolev, eds.), (Cham), pp. 216–225, Springer International Publishing, 2020.

[16] K. Alipova, V. Mizyak, M. Tolstykh, and G. Goyman, "Stochastic perturbations in the semi-lagrangian advection algorithm of the SL-AV global atmosphere model," *Rus. J. of Num. An. and Math. Mod.*, vol. 34, no. 1, pp. 1–11, 2024.

Учебный курс «Тензорные компиляторы для глубоких нейросетевых моделей»

Ю.А. Родимков, Е.П. Васильев, И.Б. Мееров, А.В. Сысоев, В.Д. Кустикова

Нижегородский государственный университет им. Н.И. Лобачевского

Тензорные компиляторы для глубоких нейронных сетей позволяют ускорить вывод за счет оптимизации модели на разных уровнях. В лекционной части курса обосновывается необходимость использования тензорных компиляторов. Рассматривается общая схема их построения, основные компоненты и последовательность работы. Описываются типовые подходы к оптимизации графа вычислений, соответствующего глубокой модели, и генерируемой программной реализации операторов на слоях сети. Практика включает пошаговую оптимизацию нейросетевой модели средствами известного тензорного компилятора Apache TVM и анализ производительности вывода. В отличие от существующих курс не требует предварительного погружения в тематику глубокого обучения и рассчитан на прикладных исследователей, которые занимаются внедрением глубоких моделей в реальные системы.

Ключевые слова: учебный курс, тензорный компилятор, глубокое обучение, нейронная сеть, вывод, тензор, граф вычислений нейронной сети, оптимизация, Apache TVM

1. Введение

Глубокое обучение получило широкое распространение и в настоящее применяется для решения большого числа практических задач в разных областях [1–4]. Внедрение нейросетевой модели в реальную систему предполагает выполнение многократного *вывода* (*inference*) – прямого прохода по обученной сети с целью получения результата. При этом накладываются требования на качество решения задачи и скорость вывода на целевом аппаратном обеспечении. Основные вычисления в процессе прямого прохода – это операции над многомерными матрицами – *тензорами* (умножение матриц, применение функции активации к элементам тензора и другие).

Тензорные компиляторы (*tensor compiler*) – инструменты, которые предназначены для ускорения вывода за счет оптимизации модели на разных уровнях представления и компиляции под конкретное устройство [5]. В рамках тензорных компиляторов модель нейронной сети описывается *графом вычислений* (*computational graph*) [6]. Узлы графа соответствуют данным (тензор входных примеров, обученные матрицы весов или вектора сдвига нейронной сети) или преобразованиям на слоях, а ребра – потоку данных. Тензорные компиляторы обеспечивают оптимизацию графа вычислений нейронной сети и автоматическую программную оптимизацию преобразований, называемых *операторами*, достигая при этом высоких показателей производительности по сравнению с альтернативными программными решениями [7–9].

Практическое применение тензорных компиляторов предполагает наличие умений и навыков настройки параметров оптимизации нейросетевой модели.

Цель учебного курса – изучение устройства тензорных компиляторов и используемых техник высокоуровневой оптимизации глубоких моделей, а также освоение процедуры анализа и сравнения производительности вывода нейронных сетей. В отличие от существующих курс является самодостаточным, не требует предварительного изучения методов глубокого обучения и принципов построения трансляторов, хотя, разумеется, владение этими тематиками является полезным для освоения курса.

Статья построена следующим образом. Вначале дается обзор существующих курсов по тематике тензорных компиляторов. Далее формулируются основные принципы разработки курса, описывается структура теоретической и практической частей курса. В заключение рассматриваются возможные пути реализации курса и перспективы его развития.

2. Обзор существующих курсов

Авторам не удалось найти полных аналогов разрабатываемого курса. Тематика тензорных компиляторов, как правило, встречается как отдельный раздел в рамках более широких учебных курсов [10–12]. Так в [10] рассматривается современное состояние исследований в области разработки систем машинного обучения, в частности, компиляторов моделей машинного обучения, фреймворков для обучения этих моделей, систем управления ресурсами для задач машинного обучения и других. Формат курса предполагает изучение современных статей, представление и обсуждение описанных результатов. Итоговая оценка за курс складывается из выступлений на семинарах (презентация методов и результатов, предложенных в статьях) и группового выполнения проекта, который предполагает реализацию какой-либо идеи, востребованной при разработке подобных систем. Пример – аппаратно-зависимая реализация обучения или вывода глубоких моделей.

Основная цель курса [11] состоит в том, чтобы у слушателя сложилось всестороннее понимание, каким образом проектируются и работают системы машинного обучения в целом. В курсе рассматриваются типовые нейронные сети, метод обратного распространения ошибки для их обучения, программные модели для представления глубоких нейронных сетей (в частности, граф вычислений нейронной сети), компиляторы для моделей машинного обучения, алгоритмы распределенного обучения, и другие связанные вопросы. Материалы лекционной части курса представлены набором презентаций, в которых дается обзор современных исследований по теме каждой лекции. В частности, одна из лекций посвящена компиляции моделей машинного обучения и тензорным компиляторам Apache TVM [13] и MLIR [14]. Практическая часть курса по аналогии с [10] предполагает представление результатов, описанных в статьях по тематике лекций, а также командное выполнение проекта. Темы проектов доступны только для слушателей курса. Курс [12] является существенной переработкой [11], исходя из содержания опубликованных материалов (сроки прочтения курсов не указаны, поэтому достоверно не известен порядок их разработки). Здесь опущены начальные лекции по глубокому обучению. Сразу вводится общая схема построения систем машинного обучения, рассматриваются основные фреймворки и некоторые методы, применяемые в процессе их реализации, в частности, автоматическое дифференцирование [15], оптимизация графа вычислений нейронной сети,

распределенное обучение, аппаратно-зависимые оптимизации, компиляция моделей машинного обучения.

Курс [16] изучает продвинутые методы компиляции и программирования для параллельных архитектур. Он построен следующим образом. Вначале дается введение в область: текущее состояние области компиляторов, история развития параллельных архитектур, компиляторы в области машинного (глубокого) обучения. Далее на примерах рассматриваются типовые техники программной оптимизации. Описывается программная среда MLIR для разработки компиляторов под различные платформы (GPUs, DPUs, TPUs, FPGAs, AI ASICs и QPUs). Демонстрируются особенности компиляции предметно-ориентированных языков на примере задач обработки изображений и вывода глубоких нейронных сетей. Практика предполагает разработку программного решения в рамках MLIR.

Разработанный нами учебный курс в отличие от [10–12] является самостоятельным и систематизирует знания в области тензорных компиляторов. При этом является более высокоуровневым по сравнению с [16] и рассчитан на прикладных исследователей, а не на разработчиков компиляторов. **Цель курса** – понять общее устройство тензорных компиляторов и научиться использовать инструментарий для оптимизации вывода нейросетевых моделей при решении задач. Далее описываются основные принципы построения курса и его структура.

3. Основные принципы построения курса

Основополагающий принцип, на который ориентировались авторы при разработке курса, – самодостаточность разрабатываемых материалов. Курс не требует предварительного изучения моделей и методов глубокого обучения, а также теории трансляторов. Необходимая базовая информация дается в начальных лекциях. С одной стороны, этот материал вводит слушателя в область глубокого обучения, с другой, формирует понимание того, зачем используются тензорные компиляторы в процессе вывода. Последующая теория и практика объясняет, на верхнем уровне, каким образом устроены указанные инструменты и как их правильно использовать, чтобы достигнуть высоких показателей производительности.

Курс является практико-ориентированным. Практическая часть предполагает, что слушатель последовательно осваивает этапы оптимизации, анализа и сравнения производительности вывода глубоких нейросетевых моделей с использованием тензорных компиляторов, которые возникают в процессе их внедрения. При этом отсутствуют существенные ограничения на выбор моделей и фреймворков, а также на выбор аппаратной платформы, на которой проводятся вычислительные эксперименты.

4. Структура курса

4.1. Описание курса

В курсе рассматриваются следующие вопросы.

1. Введение в глубокое обучение. Общая схема решения задач с использованием глубокого обучения, важность производительности вывода на этапе внедрения глубоких моделей.

2. Основные виды нейросетевых моделей. Преобразования в слоях сети как операции над многомерными тензорами.

3. Общепринятые программные инструменты для обучения и вывода глубоких моделей: программный интерфейс, примеры, демонстрирующие полный цикл программной разработки нейронных сетей.

4. Понятие тензорных компиляторов, общая схема их построения. Различные уровни оптимизации моделей с целью ускорения вывода, примеры типовых оптимизаций.

5. Обзор существующих тензорных компиляторов, сравнение их возможностей.

6. Тензорный компилятор Apache TVM (TVM): схема построения, набор применяемых оптимизаций, программный интерфейс для реализации вывода нейронных сетей.

7. Гибкая настройка параметров TVM. Анализ производительности вывода поэтапно оптимизированных глубоких нейронных сетей.

Курс включает в себя 16 часов лекций (7 лекций продолжительностью от 1 до 5 академических часов в зависимости от сложности темы) и 20 часов практических занятий (таблица 1). Лекции проводятся в классической форме с элементами мастер-класса и сопровождаются примерами решения практических заданий. Для демон-

Таблица 1: Структура учебно-образовательного курса «Тензорные компиляторы для глубоких нейросетевых моделей»

Тема	**Содержание**	**Часы**
Лекция 1. Введение в глубокое обучение.	1. История развития глубокого обучения. 2. Примеры практических задач, успешно решаемых с использованием глубоких нейронных сетей. 3. Общая схема решения практических задач средствами глубокого обучения (подготовка данных, обучение, тестирование, внедрение). 4. Важность оптимизации вычислений, выполняемых в процессе вывода, на этапе внедрения глубоких моделей. Вычисления – операции над тензорами.	1
Лекция 2. Основные виды нейросетевых моделей.	1. Модель нейрона. 2. Полносвязные нейронные сети (полносвязный слой, функции активации). 3. Сверточные нейронные сети (свертка, активация, пространственное объединение). 4. Примеры архитектур полносвязных и сверточных нейронных сетей. Анализ результатов качества решения на примере задачи классификации, оценка производительности вывода (определение оптимальных параметров запуска) на примере PyTorch или другого фреймворка.	2
Лекция 3. Обзор инструментов глубокого обучения, поддерживающих обучение и тестирование глубоких моделей.	По каждому фреймворку предполагается рассмотреть следующие вопросы. 1. Общая информация. 2. Программный интерфейс. 3. Представление модели в рамках фреймворка, особенности реализации обучения и тестирования. 4. Пример использования фреймворка (демонстрация кода обучения и тестирования моделей, рассмотренных в предыдущей лекции).	2

Тема	Содержание	Часы
Практика 1. Программная реализация вывода глубоких нейронных сетей для классификации изображений средствами нескольких фреймворков.	1. Постановка задачи классификации изображений. 2. Выбор публичной модели для решения поставленной задачи. Изучение ее архитектуры. 3. Изучение программного интерфейса двух-трех фреймворков для вывода открытых нейросетевых моделей (PyTorch, TensorFlow, TensorFlow Lite, другие). 4. Программная реализация вывода выбранной модели с использованием изученных фреймворков. 5. Анализ качества решения задачи для всех разработанных реализаций. 6. Анализ производительности вывода и определение оптимальных параметров запуска вывода на выбранной аппаратуре (размер пачки входных данных, количество потоков, другие).	6
Лекция 4. Тензорные компиляторы. Понятие, общая схема построения.	1. Понятие тензорного компилятора. 2. Общая архитектура тензорных компиляторов для глубокого обучения. 3. Основные компоненты тензорных компиляторов для глубокого обучения. 4. Представление нейросетевой модели в виде графа вычислений. 5. Типовые подходы к оптимизации графа вычислений. 6. Типовые подходы к оптимизации кода операторов (аппаратно-зависимые оптимизации).	5
Лекция 5. Обзор существующих тензорных компиляторов.	1. Обзор существующих тензорных компиляторов для глубокого обучения (Apache TVM, nGraph, XLA, MLIR, другие). 2. Сравнение возможностей тензорных компиляторов.	2
Лекция 6. Обзор возможностей Apache TVM.	1. Общая информация об инструменте. 2. Схема работы Apache TVM. 3. Оптимизация графа вычислений. 4. Оптимизация операторов, соответствующих преобразованиям на слоях нейросетевой модели. 5. Автоматическая оптимизация кода операторов. Доступные алгоритмы оптимизации. 6. Программный интерфейс Apache TVM для вывода глубоких моделей.	2
Практика 2. Вывод глубокой модели средствами Apache TVM.	1. Загрузка, конвертация и компиляция глубокой модели. 2. Программная реализация вывода. 3. Проверка корректности разработанной реализации.	4
Лекция 7. Гибкая настройка параметров TVM. Анализ производительности вывода глубоких нейронных сетей	Автоматическая оптимизация вывода для глубоких нейросетевых моделей средствами Apache TVM, с элементами анализа производительности с использованием roofline-модели.	2

Тема	Содержание	Часы
Практика 3. «Ручная» оптимизация оператора в рамках TVM на примере полносвязного и сверточного слоя.	1. «Ручная» оптимизация оператора, соответствующего полносвязному слою (умножение матриц). 2. «Ручная» оптимизация оператора, соответствующего сверточному слою. 3. Интеграция оптимизированных реализаций операторов в глубокую модель.	4
Практика 4. Автоматическая оптимизация моделей в рамках Apache TVM.	1. Автоматическая оптимизация программной реализации операторов сети. Подбор оптимальных параметров, сравнение и анализ полученных результатов. 2. Построение roofline-модели для анализа производительности вывода исходной и оптимизированных моделей и сравнения достигнутых показателей производительности с пиковыми возможностями оборудования.	6

страции выполнения практических заданий используется глубокая модель, обеспечивающая решение задачи классификации изображений – классической задачи компьютерного зрения. В процессе выполнения практических работ осуществляется последовательный запуск и пошаговая оптимизация широко известных открытых моделей с целью достижения наилучших показателей производительности вывода. Итоговый контроль знаний предполагает представление результатов выполнения практических заданий и анализ достигнутых показателей производительности вывода.

Курс ориентирован на студентов и аспирантов технических и естественно-научных специальностей, а также инженеров ИТ-компаний, преподавателей вузов и исследователей. Предполагается, что слушатели имеют базовые навыки разработки программ на языках C, C++, Python. Наряду с этим, изучение курса требует наличия теоретических знаний и практических навыков в следующих областях: дискретная математика, линейная алгебра, системное программирование, алгоритмы и структуры данных. Наличие базовых знаний в области оптимизации программ, параллельном программировании, глубоком обучении, компьютерном зрении, теории трансляторов ускорит освоение материала, но не является обязательным.

4.2. Теоретическая часть

Рассмотрим более детально теоретическую часть курса. Лекция 1 является введением в глубокое обучение. В данной лекции рассматриваются примеры практических задач, которые решаются средствами нейронных сетей, демонстрируя высокие показатели качества. Дается общая схема решения задач с использованием глубокого обучения. Основное внимание уделяется вопросам оптимизации вычислений на этапе внедрения моделей. Во второй лекции рассматриваются основные виды нейронных сетей (полносвязные и сверточные сети). Приводится пример решения задачи средствами классических инструментов, выполняется анализ качества решения задачи и производительности вывода. В лекции 3 дается обзор существующих инструментов глубокого обучения, которые используются в процессе разработки глубоких моделей. По каждому фреймворку предлагается общая информация, кратко рассматривается программный интерфейс для реализации обучения и вывода, демонстрируется пример в исходных кодах. Четвертая лекция является центральной. В данной лекции

вводится понятие тензорного компилятора, приводится общая архитектура и основные компоненты, отвечающие за оптимизацию модели на разных уровнях. Формализуется представление нейронной сети в виде графа вычислений. Описываются базовые оптимизации графа вычислений и программной реализации операторов. Далее предлагается обзор существующих тензорных компиляторов и их сравнение (лекция 5), и более детально рассматривается компилятор Apache TVM [13] (лекция 6). В седьмой лекции описывается процедура настройки параметров TVM с целью достижения наилучших показателей производительности вывода.

4.3. Практическая часть

Практическая часть курса предполагает выполнение четырех заданий. Студенту предлагается выбрать какую-либо открытую обученную модель, которая позволяет решать задачу классификации изображений. Модель можно загрузить, например, из открытого репозитория OpenVINO – Open Model Zoo [17]. В первом задании необходимо разработать программную реализацию вывода средствами двух-трех фреймворков глубокого обучения, включая исходный фреймворк, средствами которого обучалась эта модель; выполнить анализ качества решения задачи классификации и производительности вывода нейронной сети; подобрать оптимальные параметры запуска. Второе задание предусматривает программную реализацию вывода с использованием TVM и проверку ее корректности – валидацию результатов классификации для отдельных изображений и анализ качества классификации на всей тестовой выборке. В третьем задании предлагается выполнить «ручную» оптимизацию операторов, соответствующих преобразованиям на слоях модели, интегрировать оптимизированные реализации операторов в модель, запустить вывод средствами TVM и проанализировать производительность вывода. Цель четвертого задания – показать, что «ручная» оптимизация – длительная и трудоемкая процедура, которая может быть заменена набором автоматических оптимизаций. В этом задании необходимо выполнить настройку параметров компиляции и автоматической оптимизации модели. Наряду с этим, сравнить и проанализировать производительность вывода оптимизированных моделей с результатами, полученными в практике 1. Выполнение практических заданий сопровождается демонстрацией примера в открытых исходных кодах и анализом результатов производительности на архитектуре RISC-V [18–20].

5. Заключение

В настоящее время разработано большое количество глубоких нейронных сетей, которые способны решать значительный спектр прикладных задач, демонстрируя приемлемые показатели качества. Внедрение этих моделей в реальные программные системы – отдельный этап жизненного цикла модели, требующий систематизации знаний и процессов анализа качества и производительности вывода. Учебный курс «Тензорные компиляторы для глубоких нейросетевых моделей» является актуальным и практически значимым, поскольку позволяет подготовить специалиста, который понимает основные подходы к оптимизации вывода и способен проводить последовательный анализ производительности вывода на этапе внедрения моделей с использованием существующего инструментария.

К моменту подачи работы материалы первых трех лекций согласно плану курса подготовлены полностью, имеется черновик лекции 6 (таблица 1). Наряду с этим, выполнена необходимая подготовительная работа для последующей разработки материалов практической части курса. К началу сентября планируется полностью разработать презентации лекционной части курса и первых двух практических заданий. Полный пакет материалов после рецензирования и апробации будет готов к концу календарного года. Все материалы предполагается выложить в открытый доступ на сайте университета под лицензией Apache 2.0.

Разрабатываемый курс может быть встроен в качестве спецкурса в основную программу обучения студентов старших курсов технических и естественно-научных специальностей или программу повышения квалификации. Наряду с этим, отдельные материалы можно задействовать в ходе проведения молодежных школ для студентов и аспирантов.

Один из возможных путей развития курса – углубление понимания внутреннего устройства тензорных компиляторов и расширение практических навыков разработки тензорных компиляторов под какую-либо аппаратную платформу на базе среды MLIR.

Список литературы

[1] Mehta, P., et al. A high-bias, low-variance introduction to Machine Learning for physicists.
`https://arxiv.org/abs/1803.08823`
(дата обращения 24.01.2024).

[2] Yoon S. lncRNAnet: Long Non-coding RNA Identification using Deep Learning. Bioinfor-matics. 2018. 34(22). P. 3889–3897.

[3] Yu T. A graph-embedded deep feedforward network for disease outcome classification and feature selection using gene expression data. Bioinformatics. 2018. 34(21). P. 3727–3737.

[4] CS231n: Deep Learning for Computer Vision.
`http://cs231n.stanford.edu`
(дата обращения 24.01.2024).

[5] Разработка тензорного компилятора под RISC-V CPU с помощью OpenVINO и MLIR.
`https://engineer.yadro.com/article/tensor-compiler`
(дата обращения 24.01.2024).

[6] Ng A., et al. Deep Learning Specialization. Course 1: Neural Networks and Deep Learning (Computation Graph).
`https://www.deeplearning.ai/courses/deep-learning-specialization`,
`https://www.youtube.com/watch?v=hCP1vGoCdYU`
(дата обращения 24.01.2024).

[7] Ding Y., Yu C.H., Zheng B., Liu Y., Wang Y., Pekhimenko G. Hidet: Task-Mapping Pro-gramming Paradigm for Deep Learning Tensor Programs // In the Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2023. Vol. 2. P. 370–384.

[8] Feng S., el al. TensorIR: An Abstraction for Automatic Tensorized Program Optimization // In the Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2023. Vol. 2. P. 804–817. https://doi.org/10.1145/3575693.3576933

[9] Li M., et al. The Deep Learning Compiler: A Comprehensive Survey // IEEE Transactions on Parallel and Distributed Systems. 2021. Vol. 32, No. 3. P. 708–727. https://doi.org/10.1109/TPDS.2020.3030548

[10] ML for ML Systems.
https://courses.cs.washinton.edu/courses/cse599m/23sp
(дата обращения 24.01.2024).

[11] Machine Learning Systems.
https://www.cs.cmu.edu/~zhihaoj2/15-849
(дата обращения 24.01.2024).

[12] Machine Learning Systems.
https://catalyst.cs.cmu.edu/15-884-mlsys-sp21
(дата обращения 24.01.2024).

[13] Apache TVM.
https://tvm.apache.org
(дата обращения 24.01.2024).

[14] MLIR. https://mlir.llvm.org
(дата обращения 24.01.2024).

[15] Automatic differentiation.
https://www.cs.toronto.edu/∼rgrosse/courses/csc321_2018/slides/lec10.pdf
(дата обращения 24.01.2024).

[16] Advanced Techniques in Compilation and Programming for Parallel Architectures.
https://www.csa.iisc.ac.in/~udayb/e0358
(дата обращения 24.01.2024).

[17] OpenVINO Toolkit – Open Model Zoo repository.
https://github.com/openvinotoolkit/open_model_zoo,
https://docs.openvino.ai/2023.2/model_zoo.html
(дата обращения 24.01.2024).

[18] Asanović K., Patterson D.A. Instruction sets should be free: The case for risc-v // EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146. – 2014.

[19] Asanović K. Advancing HPC with RISC-V // Invited talk at Supercomputing Conference. 2022.
`https://sc22.supercomputing.org/presentation/?id=inv111&sess=sess240`
(дата обращения 24.01.2024).

[20] History – RISC-V International.
`https://riscv.org/about/history`
(дата обращения 24.01.2024).

Аннотации стендовых докладов

Evolving the system for managing a computational cluster through containerization[1]

R. Kostromin, A. Feoktistov

Matrosov Institute for System Dynamics and Control Theory of SB RAS

In recent years, a noticeable trend of expansion of supercomputer application areas on new challenging trends in the field of machine learning, artificial intelligence, microservices, and other advanced technologies is targeted [1, 2]. Historically, supercomputer centres (SCCs) have been complex, specialist vendor systems with relatively high maintenance requirements. The infrastructure of an SCC's compute cluster typically does not undergo frequent changes in applications and system libraries, which limits the range of scientific applications that can be run. In order to extend this range, this paper proposes an architecture for managing a computational cluster based on containerization.

The need to update the current architecture of the Joint Usage Center "Irkutsk SC SB RAS" is driven by new user demands and rapidly changing computing requirements for the cluster environment. Unfortunately, the renewal of the cluster is hindered by limited funding, making it difficult to acquire suitable domestic software solutions. At the same time, open-source solutions such as OpenHPC are not a all-in-one solution for containerization, necessitating the adaptation of general-purpose software for control system updates within the current SCC archituecture. Special attention should be paid to the developments of RAS institutes, such as Asperitas, Fanlight (ISPRAS), HPC TaskMaster (HSE University Supercomputing Modeling Unit), and Octoshell (RCC MSU). These organizations have contributed part of their software to open-source and their solutions can be integrated into the proposed architecture.

The existing typical architecture of this SCC is commonly found in SCCs provided by the Russian supercomputer companies (Figure 1, a). It includes the dedicated role of the master virtual machine (VM), which has data storage replication configured through DRBD on two servers (master-1 and master-2). The master VM manages the cluster network, computing nodes (CNs), contains a local resource manager (LRM), LDAP user directories, monitoring system, etc. Access to the cluster is provided by primary and backup access nodes (access-1 and access-2). All systems are connected via a high speed network to a high performance Data Storage System (DSS).

In the early stages of developing the existing cluster management model, the use of virtualization based on the freely available Proxmox hypervisor was considered [3]. However, virtualization has significant overhead costs for compute nodes [2, 4]. As shown in previous studies, a popular and well-established approach to building HPC environments is software containerization, particularly using Docker. Unlike virtualizations, containerization reduces the overhead of preparing, storing and deploying customized software

[1]The work was supported by the Ministry of Science and Higher Education of the Russian Federation, the grant № 075-15-2024-533 for implementation of Major scientific projects on priority areas of scientific and techno-logical development (the project ≪Fundamental research of the Baikal natural territory based on a system of interconnected basic methods, models, neural networks and a digital platform for environmental monitoring of the environment≫).

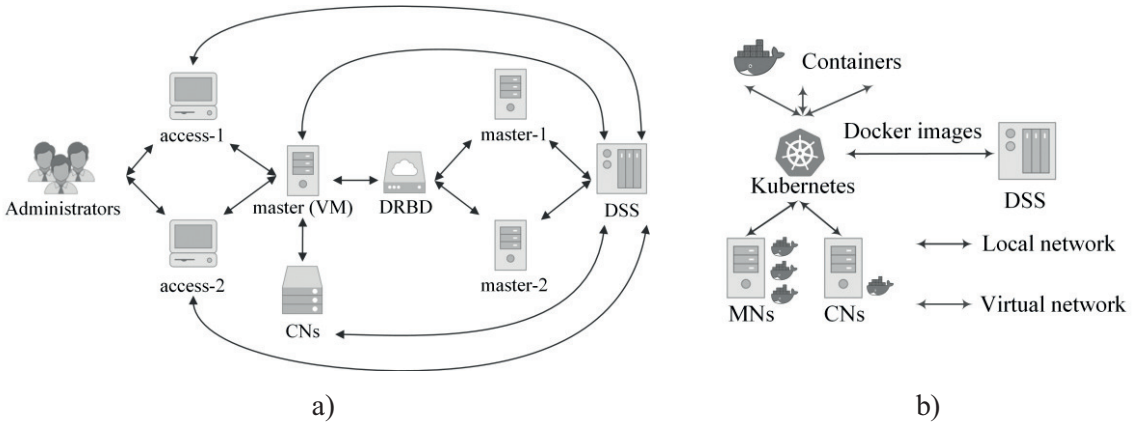


Figure 1: The typical HPC-cluster architecture (a) and the new architecture (b)

images. However, Docker-based containerization shows a significant drop in performance, in contrast to the Singularity containerization system. Singularity provides full support for Docker images and overcomes the limitations of Docker for running containers.

The new SCC management architecture is shown schematically in Figure 1, b. It is based on a Kubernetes control cluster (k8s) running on multiple management nodes (MNs). k8s is used to launch Docker images (including Singularity) stored in the local DSS. Some containers include management components (LDAP, monitoring, LRM, etc.) that run on MNs. k8s creates isolated networks (virtual networks) to facilitate interaction between containers on CNs. CNs run Singularity containers that support LRM (PBS, Slurm, etc.) within the established quotas for users. Images stored in DSS are prepared by SCC administrators and provide layered integration (Figure 2) of necessary components for applications and their modules running on the compute cluster. In some cases, it is possible to run LXC containers through Proxmox for services with long-term lifecycle.

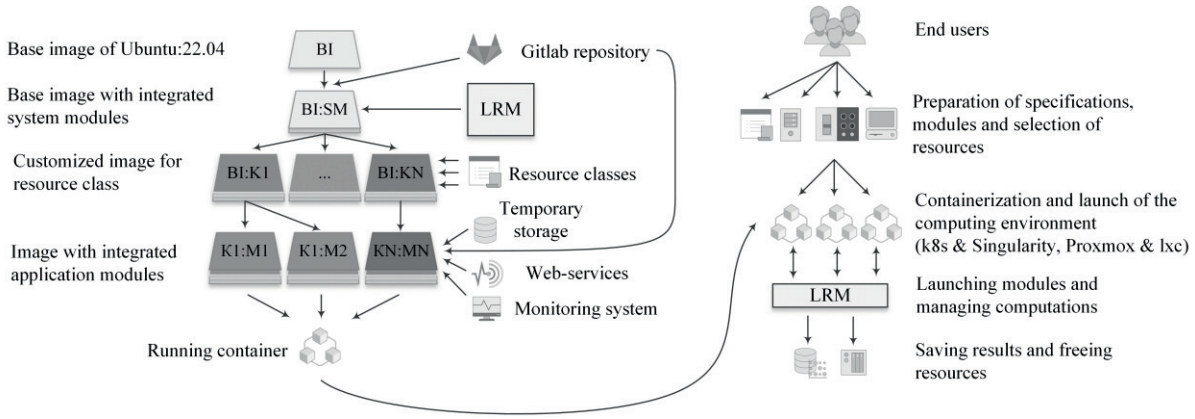


Figure 2: Scheme for preparation and executing of a containerized computing environment

Despite the fact that work on the development of the cluster architecture is still in its early stages, some work has already been done on testing a prototype of the new architecture. The current version of Rocky Linux 9.3, which has replaced Centos as the operating system, has been deployed on five nodes allocated for testing purposes. A Singularity-based container image has also been prepared, focusing on running MPI applications. These containers were deployed using Minicube, a lightweight version of Kubernetes. Running Linpack tests and other application tasks showed that the execution

time of tasks and overall node performance did not degrade when using containerization. Based on these preliminary results, we can continue working in this area. This model goes beyond the traditional use of classical parallel applications, opening up new possibilities for carrying out complex and large-scale scientific research.

References

[1] Shabanov B.M., Samovarov O.I. Building the Software-Defined Data Center. Programming and Computer Software. 2019. Vol. 45. P. 458–466.
https://doi.org/10.1134/S0361768819080048

[2] Deng S. eet al. Cloud-Native Computing: A Survey From the Perspective of Services. Proceedings of the IEEE. 2023. Vol. 112. P. 12–46.
https://doi.org/10.1109/JPROC.2024.3353855

[3] Feoktistov A. et al. An Approach to Implementing High-Performance Computing for Problem Solving in Workflow-based Energy Infrastructure Resilience Studies // Computation. 2023. Vol. 11, no. 12. P. 243.
https://doi.org/10.3390/computation11120243

[4] Rodriguez M.A., Rajkumar B. Container-based cluster orchestration systems: A taxonomy and future directions. Software: Practice and Experience. 2018. Vol. 49. P. 698–719.
https://doi.org/10.1002/spe.2660

Metadynamics Simulations of Antibody and SARS-CoV-2 RBD Complex in MARTINI 3 Force Field

Ya.V. Chuiko[1], A.V. Golovin[2]

[1]Sirius University of Science and Technology,
[2]Faculty of Bioengineering and Bioinformatics Moscow State University

Introduction. Neutralizing MAbs are efficient therapeutic drugs for the treatment of viral and autoimmune diseases. Given the emergence of new SARS-CoV-2 variants and the lack of efficacy of several neutralizing monoclonal antibodies, a rapid way to design new antibodies is needed [1]. Nowadays, diffusion machine learning models for the design of new protein sequences are becoming increasingly popular. But it is still a problem to sort the new structures obtained using the generative diffusion model [2]. Metadynamics allows one to quickly and efficiently scan different states of a system by adding Gaussian potentials to local free energy minima during a trajectory [3]. Representation of structures in the Martini force field together with the use of metadynamics speeds up FES calculations for protein-protein complexes [4]. Here, we demonstrate application of the metadynamics in the averaged most common antibody and SARS-CoV-2 RBD in the MARTINI 3 force field. We created a complex in which the antibody targets one of the most neutralizing RBD epitopes. For this complex, we performed metadynamics and obtained the states of the complex corresponding to the free energy minima. The obtained structures will become a starting point for the further design of CDR loops, and metadynamics in the MARTINI 3 force field is a potential effective method of ranging the generated structures.

Materials and methods. SARS-Cov-2 neutralizing MAbs data were downloaded from The Coronavirus Antibody Database CoV-AbDab [5]. We use the MODELLER to construct a structure of the antibody based on multiple templates [6]. The most frequency antibody CDRH3 and CDRL3 lengths and amino acid sequences were found for constructing the averaged common antibody sequence and modeling its structure. All-atom structure of the constructed complex was converted to CG-resolution structures in Martini 3.0.beta.3.2 force field [7]. Metadynamics simulations were run under NPT ensemble at 300 K by GROMACS version 2022.5 software with PLUMED version 2.9.0. The distance between the antibody and RBD centers of mass (COM) and the dihedral angle characterizing the rotation of the antibody relative RBD were selected as CV. For a system consisting of 7969 atoms including water molecules, the computation time was 50 core*hours using 1 CPU thread.

Results and discussions. With an averaged antibody, a complex with the RBD domain of the SARS-CoV-2 virus was created so that the CDR loops of the antibody were targeted to the epitope that most overlaps with the binding site of the ACE-2 (Fig. 1). The amino acid residues of this epitope are most frequently mutated, and the further design of new antibodies to this epitope is most current.

After conversion to CG-structures and its preparation metadynamics in two CVs was performed. Reweighted free energy surface for complex is shown in Fig. 2. First five free energy minima (values less than 0) are pointed out with numbers from 1 to 9.

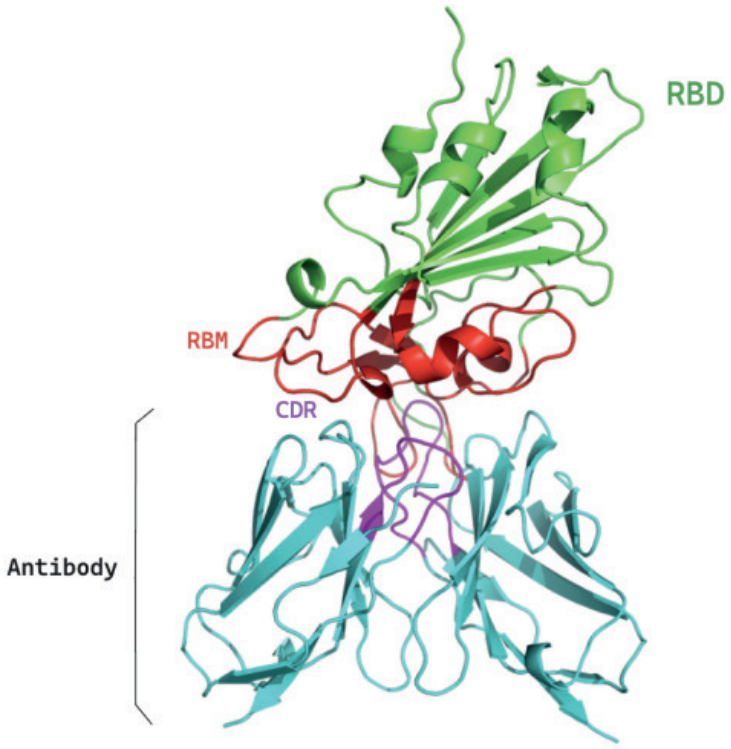


Figure 1: The averaged antibody (cyan) and RBD (green) complex. CDRH3 and CDRL3 (pink) of antibodies target epitope A and epitope B (red).

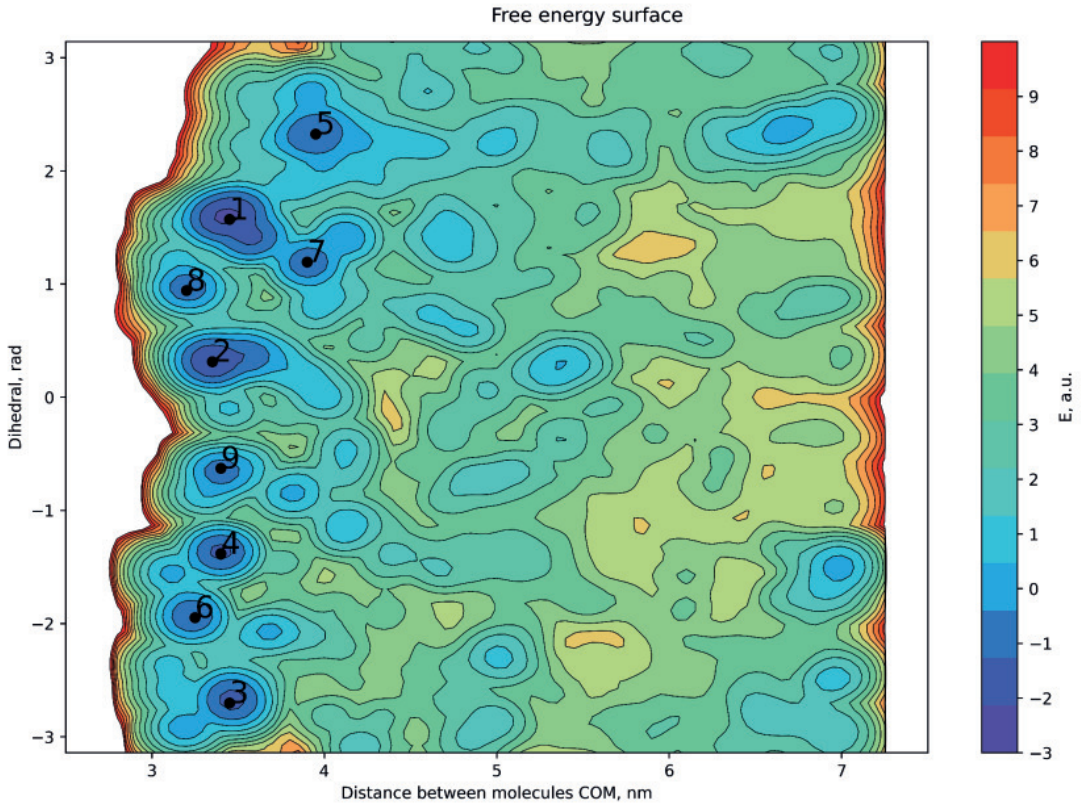


Figure 2: Reweighting free energy surface plot for the averaged antibody and RBD complex.

A map of RBD contact residues for each of the minimal states obtained after metadynamics is shown in Fig. 3.

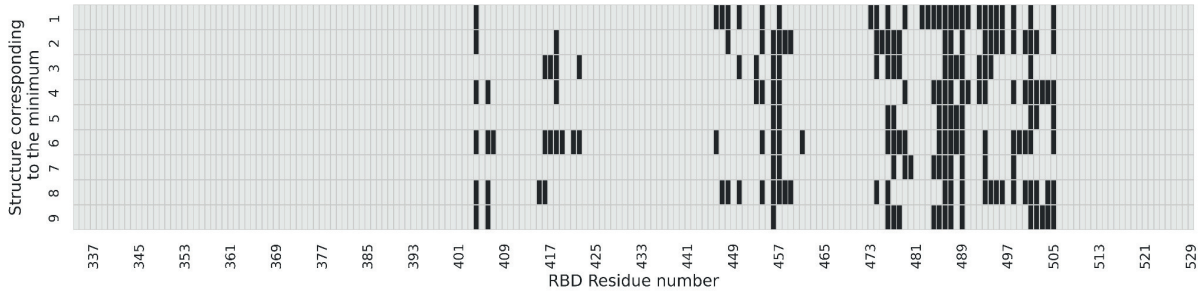


Figure 3: Representation of RBD residues involved in binding to the antibody for minimal states.

We used the metadynamics in the Martini 3 force-field for investigating the antibody and SARS-CoV-2 RBD complex states. In each, the antibody targets different previously known epitopes in the region of the receptor-binding motif. Using such a rapid approach could help in the design of new neutralizing antibodies. It is possible to rank the hundreds of the design results of new antibody structures, obtaining the free binding energy value for each of them.

References

[1] Wang L. et al. Immune evasion of neutralizing antibodies by SARS-CoV-2 Omicron // Cytokine Growth Factor Rev. 2023. Vol. 70. P. 13–25.

[2] Yin R., Pierce B.G. Evaluation of AlphaFold Antibody-Antigen Modeling with Implications for Improving Predictive Accuracy // bioRxiv. 2023.

[3] Ray D., Parrinello M. Kinetics from Metadynamics: Principles, Applications, and Outlook // J. Chem. Theory Comput. 2023. Vol. 19, № 17. P. 5649–5670.

[4] Lamprakis C. et al. Evaluating the Efficiency of the Martini Force Field to Study Protein Dimerization in Aqueous and Membrane Environments // J. Chem. Theory Comput. 2021. Vol. 17, № 5. P. 3088–3102.

[5] Raybould M.I.J. et al. CoV-AbDab: the coronavirus antibody database // Bioinformatics. 2021. Vol. 37, № 5. P. 734–735.

[6] Webb B., Sali A. Protein Structure Modeling with MODELLER // Methods Mol. Biol. 2017. Vol. 1654. P. 39–54.

[7] Souza P.C.T. et al. Martini 3: a general purpose force field for coarse-grained molecular dynamics // Nat. Methods. 2021. Vol. 18, № 4. P. 382–388.

Измерение производительности базовых блоков на архитектурах, отличных от x86

А.Ю. Баташев

Нижегородский государственный университет им. Н.И. Лобачевского

Оптимизирующие компиляторы опираются на эвристики при принятии решений. Однако, существующие модели затрат часто страдают от неточностей и ограниченной применимости к различным архитектурам. Так, Пол и др. [3] показывают, что текущая модель векторизации в рамках фреймворка LLVM слабо коррелирует с реальными измерениями, коэффициент корреляции Пирсона составляет 0,55. Наша работа фокусируется на измерении производительности отдельных базовых блоков на различных архитектурах с целью последующего анализа и улучшения моделей принятия оптимизационных решений.

Базовый блок (*ББ*) — это линейная последовательность инструкций без ветвлений. В терминах LLVM базовый блок должен заканчиваться т. н. терминатором (англ. *terminator*), таким как ветвление или возврат к вызываемой процедуре. Однако в ходе данного исследования мы также рассматриваем вызовы функций или системные вызовы в качестве завершающих элементов блока.

Процесс измерения производительности можно условно разбить на три этапа: извлечение базовых блоков, проведение экспериментов на реальном оборудовании (а именно измерение количества тактов процессора, затраченных для исполнения кода, и последующая статистическая обработка с целью исключения шума в данных), анализ полученных данных. Данный процесс подробно описан нами в предыдущем исследовании[1]. На рисунке 1 показана схема работы разработанного нами ПО с открытым исходным кодом[1].

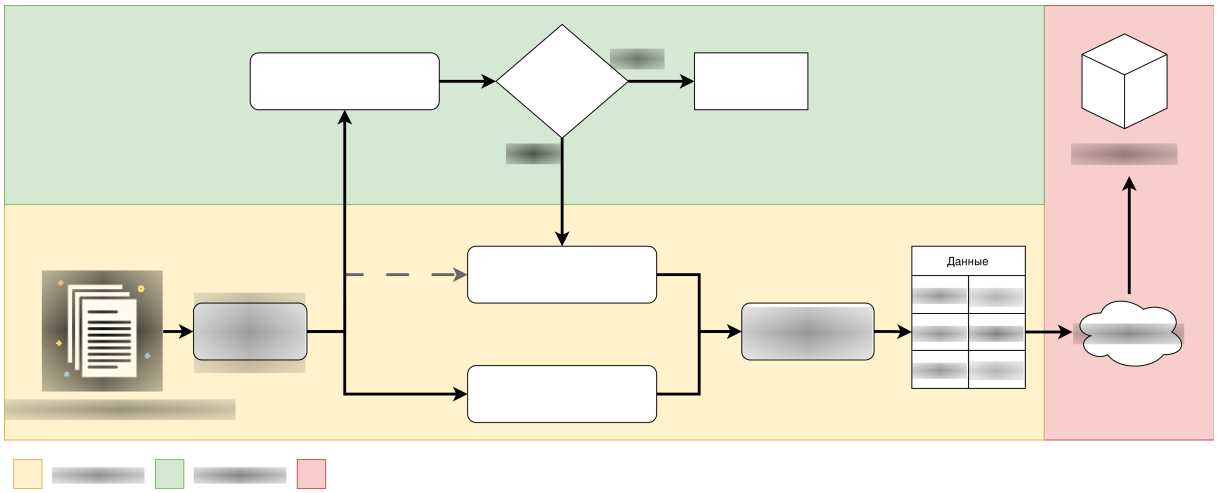


Рис. 1: Схема работы инструмента автоматического анализа производительности, включая результаты прошлых исследований, промежуточные результаты текущего доклада и направление будущих исследований.

[1]https://github.com/perf-toolbox/llvm-ml

Получение базовых блоков для проведения экспериментов может выполняться одним из трех способов:

1) *Генерация базовых блоков*. Такой метод позволяет породить произвольное количество входных данных, но такие данные не будут репрезентативны реальным нагрузкам.
2) *Динамическое извлечение* с помощью таких инструментов как DynamoRIO [2]. Такой способ затратен по вычислительным ресурсам и требует значительной подготовки.
3) *Статическое извлечение*, основанное на дизассемблировании бинарных файлов и последующей их очистке. Именно этот способ применяется в нашей работе.

Для измерения производительности базовый блок дублируется N раз, а затем запускается на случайных входных данных. Проблема выхода за границы аллоцированной памяти решается посредством отображения уже выделенной страницы памяти на соответствующие адреса. Для каждого бенчмарка записывается число процессорных тактов. Чтобы исключить шум при измерении, отслеживаются дополнительные метрики: количество переключений контекста операционной системы, количество загрузок по невыравненным адресам, количество кэш-промахов. Данные события являются внешними по отношению к исполняемой программе и не могут быть проконтролированы в рамках эксперимента. Чтобы результаты измерений можно было сравнивать, мы исходим из предположения, что программа исполняется в идеальных условиях, когда все процессорное время принадлежит одному процессу, а все доступы в память имеют константное время исполнения.

Сбор метрик производится при помощи специальных архитектурно-зависимых регистров и счетчиков (*PMU, Performance Monitoring Unit*). К сожалению, многие счетчики, доступные на платформе x86, недоступны на других платформах. Для решения этой проблемы мы предлагаем использовать симулятор архитектуры набора инструкций (*ISA, Instruction Set Architecture*).

Симулятор производит разбор входной программы на языке ассемблера целевой платформы и переводит её в промежуточное представление. Инструкции данного представления оперируют регистрами. Симулятор хранит отображение виртуальных страниц памяти на реальные страницы хостовой операционной системы и по запросу загружает данные в регистры. Перед запуском симуляции пользователь может проинициализировать внутреннее состояние процессора, а после исполнения каждой инструкции – проинспектировать его. Таким образом, детерминированные метрики, не зависящие от микроархитектурных особенностей, могут быть собраны независимо от поддержки соответствующих счетчиков конкретной реализацией платформы.

Добавление симулятора в процесс измерения меняет алгоритм работы бенчмарка. Если раньше все метрики снимались при помощи PMU, то теперь процесс разбивается на два шага. На первом этапе мы генерируем ассемблерный код бенчмарка, инициализируем внутреннее состояние симулятора и запускаем в нем полученный код. Если в ходе эксперимента хотя бы одна из инструкций обращалась к невыравненным адресам, то процесс останавливается, а базовый блок признается непригодным для исследования. В противном случае тот же самый ассемблерный код компилируется и запускается на реальном процессоре, проинициализированном аналогично симулятору. Обновленный процесс также показан на рисунке 1.

В результате получается набор, состоящий из пар вида (базовый блок, число тактов). В докладе будут представлены результаты экспериментов для платформы RISC-V. Полученные данные в дальнейшем можно использовать для анализа производительности процессоров или алгоритмов оптимизации в компиляторах, что является темой наших будущих исследований.

Список литературы

[1] *Batashev A.* Toolkit for Micro-Benchmarking of Assembly Basic Blocks on Modern CPU Architectures // 2023 Ivannikov Ispras Open Conference (ISPRAS). — 12.2023. — С. 7—14.
https://doi.org/10.1109/ISPRAS60948.2023.10508160

[2] BHive: A Benchmark Suite and Measurement Framework for Validating X86-64 Basic Block Performance Models / Y. Chen [и др.] // 2019 IEEE International Symposium on Workload Characterization (IISWC) (2019 IEEE International Symposium on Workload Characterization (IISWC)). — 11.2019. — С. 167—177.
https://doi.org/10.1109/IISWC47752.2019.9042166

[3] *Pohl A., Cosenza B., Juurlink B.* Portable Cost Modeling for Auto-Vectorizers // 2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) (2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)). — Rennes, FR: IEEE, 10.2019. — С. 359—369.
https://doi.org/10.1109/MASCOTS.2019.00046

Квантовый алгоритм поиска ближайшего

К.Р. Захарова[1], А.А. Черников[2]

[1]Факультет Математики и компьютерных наук СПбГУ,
[2]Математико-механический факультет СПбГУ

Лов Гровер в 1996 году предложил квантовый алгоритм поиска [1] точного совпадения с искомым значением среди монотонного массива, содержащего весь спектр состояний от 0 до $N-1=2n-1$. Данная работа послужила опорой для ряда интересных модификаций [2]. Однако в реальной жизни в данных могут быть повторяющиеся элементы или, напротив, отсутствовать часть спектра, как и само искомое значение. В таком случае: а) создание суперпозиции требует отдельной проработки; б) необходимо осуществить поиск ближайшего значения вплоть до точного совпадения, если таковое имеется. Цель нашего исследования – разработка алгоритма поиска ближайшего к искомому значения среди случайных элементов массива посредством перераспределения вероятности пропорционально удалённости данного элемента от искомого значения.

Нашей научной группой была предложена как общая схема алгоритма поиска ближайшего (рис. 1), так и рабочая цепь, использующая квантовые логические гейты, в специальной среде «Qiskit» [3], где также были проведены первые симуляции.

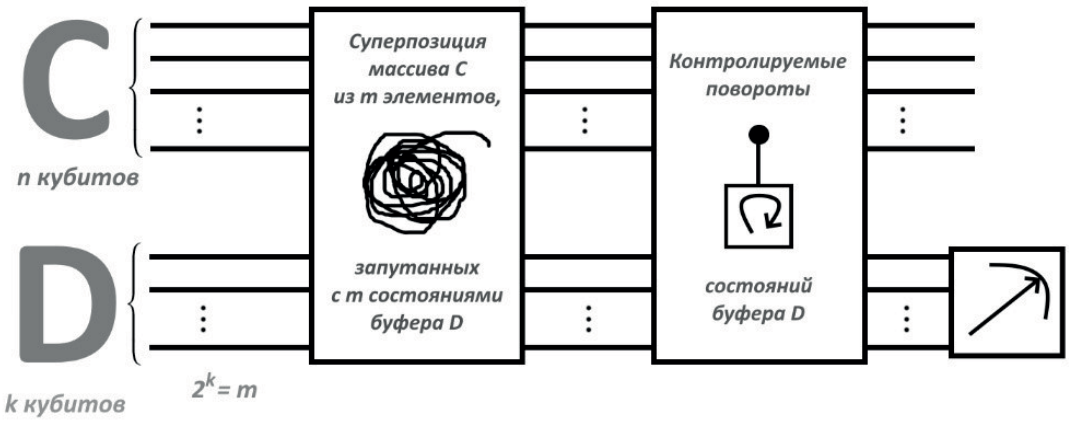


Рис. 1: Принципиальная схема алгоритма поиска ближайшего

Предложенный алгоритм принимает на вход неизвестный массив C в количестве m элементов, каждый из которых занимает n кубитов: Далее с помощью специального буфера D, состоящего из k кубитов (размерность всех состояний буфера $2k$ равна количеству элементов m), создаётся равновзвешенная суперпозиция всех элементов массива таким образом, что каждый элемент запутывается с единственным состоянием буфера. После чего происходит одновременная обработка всего массива: каждый элемент сравнивается с искомым, и чем меньше совпадение – тем сильнее уменьшается вероятность получить отсылку к данному элементу при измерении. Сравнения происходит с помощью контролируемых поворотов пространства состояний буфера, общая матрица поворотов представлена на рис. 2. В конце вычислений происходит измерение буфера, и полученное значение указывает на порядковый номер запутанного с ним элемента.

$$\begin{pmatrix} \cos\frac{\varphi}{2} & \pm\frac{\sin\frac{\varphi}{2}}{\sqrt{2^n-1}} & \dots & \pm\frac{\sin\frac{\varphi}{2}}{\sqrt{2^n-1}} \\ \pm\frac{\sin\frac{\varphi}{2}}{\sqrt{2^n-1}} & \cos\frac{\varphi}{2} & \dots & \pm\frac{\sin\frac{\varphi}{2}}{\sqrt{2^n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \pm\frac{\sin\frac{\varphi}{2}}{\sqrt{2^n-1}} & \pm\frac{\sin\frac{\varphi}{2}}{\sqrt{2^n-1}} & \dots & \cos\frac{\varphi}{2} \end{pmatrix}$$

Рис. 2: Общий вид матрицы поворотов пространств буфера D

Пример цепи с результатом представлен на рис. 3, где в верхней части создаётся равновзвешенная суперпозиция массива [15 14 10 0 1 4 6 9], в нижней части изображена специально разработанная цепь поворотов пространств буфера, с помощью которой в массиве производится поиск минимума. Результаты перераспределения вероятностей представлены на графике справа: вдоль горизонтальной оси расположены состояния буфера D, красным подписаны запутанные с каждым из них значения массива, синие столбики – количество измерений данного состояния при симуляции. На рис. 4 представлены результаты для обработки другого массива. Важно отметить, что для обоих случаев достаточно было вызвать алгоритм один раз.

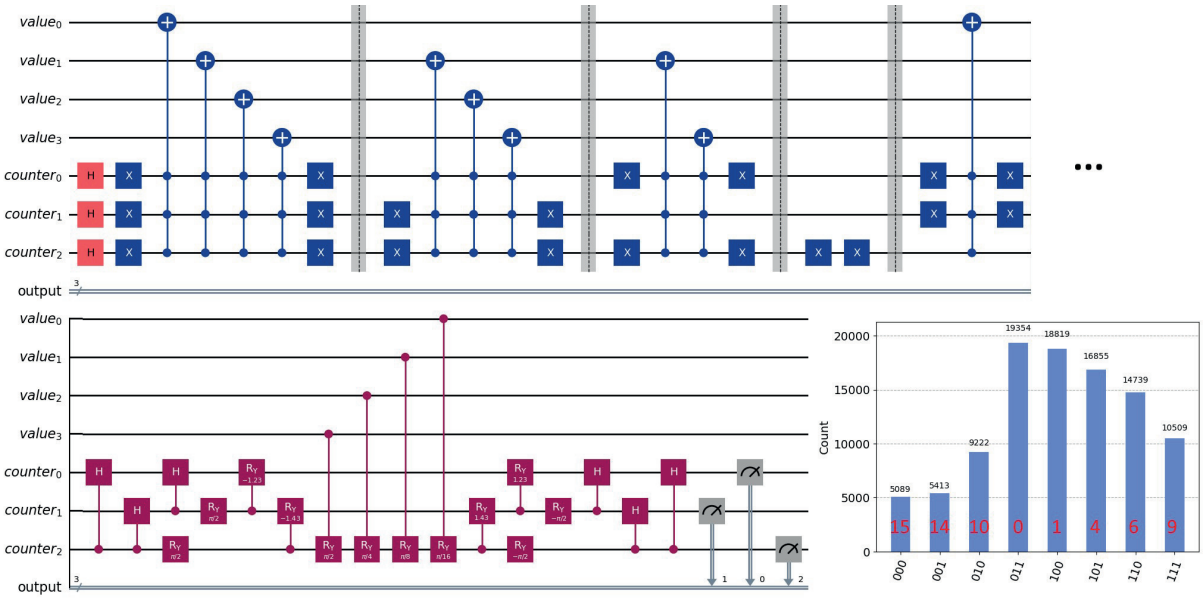


Рис. 3: Схема для поиска ближайшего в массиве [15 14 10 0 1 4 6 9], разработанная в специальной среде «Qiskit». На графике изображены результаты 100000 измерений

На рис. 4 представлены результаты поиска в массивах с повторяющимися элементами, при работе с которыми в схеме требуется две серии поворотов с промежуточным измерением.

В ходе дальнейшего исследования планируется нахождение способа оценки эффективности алгоритма, обоснования действия промежуточного измерения, а также поиск ответа на вопрос, можно ли сделать алгоритм итеративным с дальнейшим увеличением необходимых амплитуд.

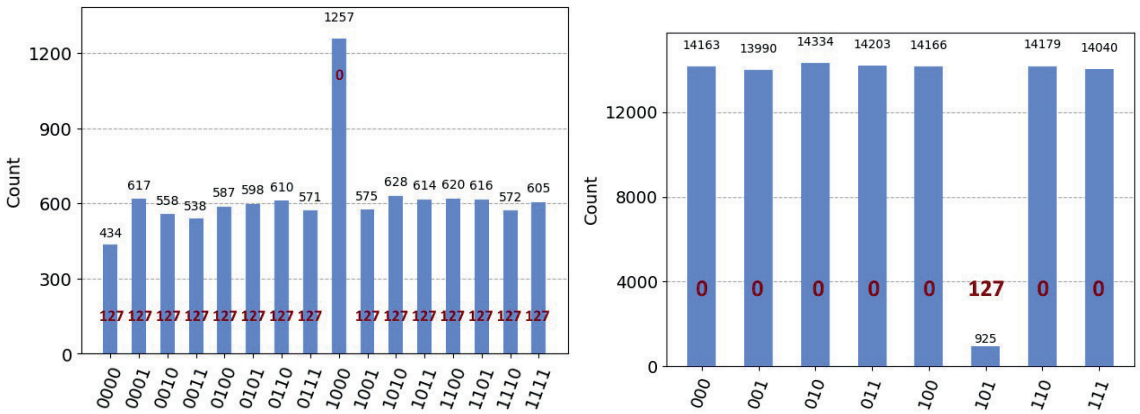


Рис. 4: Графики результатов поиска в массивах с повторяющимися элементами

Список литературы

[1] Lov K. Grover. A fast quantum mechanical algorithm for database search. Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. STOC'96. Philadelphia, Pennsylvania (1996), pp. 212–219.

[2] Christoph Durr, Peter Hoyer. A Quantum Algorithm for Finding the Minimum. `https://doi.org/10.48550/arXiv.quant-ph/9607014`, `https://www.ibm.com/quantum/qiskit`

Компьютерное моделирование распространения нерелятивистских джетов на многопроцессорных ЭВМ

Б.П. Рыбакин[1], Г.В. Секриеру[2]

[1]МГУ им. М.В. Ломоносова,
[2]Институт математики и информатики им. В. Андрунакиевича, МолдГУ, Молдова

Представлены результаты компьютерного моделирования формирования и распространения джета возникающего около коллапсирующего протозвездного облака. Звездообразованию во Вселенной способствует газ в молекулярных облаках (МО). При соударении МО между собой, при соударении ударных волн с молекулярными облаками возникают гравитационно-связанные области [1], в которых, при определенных условиях формируются протозвездные системы, из которых образуются молодые звезды (YSO). Наблюдения показывают, что фаза аккреции очень часто сопровождается мощным выбросом заметных биполярных потоков, состоящих из быстрых струй – джетов [2]. Это явление начинается на ранней стадии образования протозвезды и предполагается [3], что эти потоки в большой степени определяют конечную массу звезды. Это происходит из-за удаления массы из ядра будущей звезды и формирующейся турбулентности и потерю энергии за счет радиационного охлаждения. Качественное и реалистическое магнитогидродинамическое моделирование джетов зависит от сложных эффектов, которые до сих пор являются предметом интенсивных исследований. Сложности в таких задачах возникают при учете турбулентной вязкости, удельного сопротивления, эффектов неидеальности и т.д. Кроме того, на морфологию гидродинамического течения, вызванного джетом, сильно влияет расслоение плотности окружающего ядро вещества. Кроме того, эти струи переносят массу от аккреционного диска, что влияет на эффективность звездообразования (SFE) и может помочь объяснить низкую скорость звездообразования.

Будем рассматривать нестационарные течения идеального сжимаемого газа. Эффектами вязкости и трения будем пренебрегать. Тогда можно записать систему дифференциальных уравнений Эйлера, которые моделируют законы сохранения массы, количества движения и энергии в трехмерной декартовой системе координат:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial x_i}(\rho v_i) = 0,$$
$$\frac{\partial \rho v_i}{\partial t}\frac{\partial}{\partial x_i}(\rho v_i v_j + P\delta_{ij}) = 0,$$
$$\frac{\partial e}{\partial t}\frac{\partial}{\partial x_i}[(e+P)V_i] = 0.$$

В расчетах используется уравнение состояния идеального газа:

$$P = \rho(\gamma - 1)[1/2(u^2 + v^2 + w^2)],$$

здесь γ – отношение удельных теплоемкостей, ρ – плотность, $v = \{u, v, w\}$ – вектор скорости и p – давление [4]. Система уравнений Эйлера решается с помощью схемы расщепления по физическим координатам второго порядка точности типа TVD.

Для моделирования задачи распространения нерелятивистских джетов был разработан параллельный трехмерный адаптивный магнитогидродинамический код для исследования явлений соударения, формирования джетов и изотермического коллапса в моделях центрально конденсированных молекулярных облаков и взаимодействия с джетами. Для валидации программы численного моделирования взаимодействия молекулярных облаков и проверки точности расчета гравитационного потенциала аналитические решения [5] сравнивались с полученными численными результатами. Компьютерное моделирование крупномасштабных процессов формирования филамент и сверхплотных гравитационно-связанных сгустков при соударении молекулярных облаков проводилось на параллельных вычислительных кластерах с гибридной архитектурой. Для вычислений использовалась авторская программа гетерогенных технологий Coarray Fortran и OpenACC для GPU. В программе применяется гетерогенное распараллеливание вычислений на CPU и GPU. Для работы на графических процессорах используется технология OpenACC, на процессорах Intel Xeon — технология Coarray, что позволяет существенно сократить время выполнения команд программы.

На рисунке 1 приведен снимок джета в созвездии Carina Nebula (слева) и результат математического моделирования взаимодействия джета с МО.

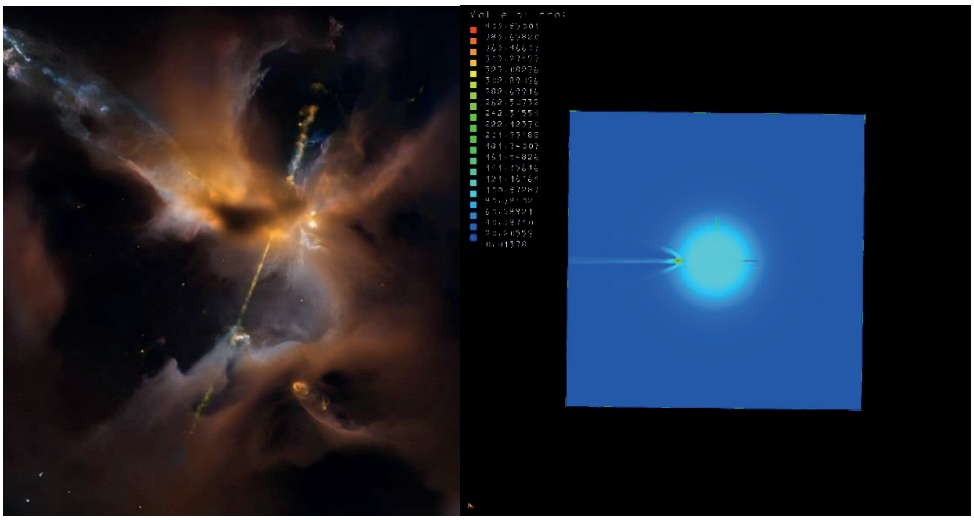


Рис. 1: Слева – снимок Hubble джета в Carina Nebula, справа результат моделирования взаимодействия джета с молекулярным облаком

Список литературы

[1] B.P. Rybakin. Dynamic evolution and morphological analysis of supersonic turbulence arising during the collision of prolate and spherical clouds. Acta Astronautica, Pergamon Press Ltd, (2024), v. 215, p. 325-332.

[2] Whelan E.T. Jets from Young Stars and Brown Dwarfs. arXiv:1406.7627v1 [astro-ph.SR] 30 June 2014.

[3] V.R. Krumholz, C. Federrath. The Role of Magnetic Fields in Setting the Star Formation Rate and the Initial Mass Function. Front. Astron. Space Sci., 20 February 2019, Sec. Stellar and Solar Physics. Volume 6 - 2019.
https://doi.org/10.3389/fspas.2019.00007

[4] B. Rybakin, V. Goryachev. The supersonic shock wave interaction with low-density gas bubble. Acta Astronautica, 94, (2014), 749–753.

[5] J.M. Stone and M.L. Norman, "ZEUS-2D: A radiation MHD code for astrophysical flows in two space dimensions, I. The hydrodynamic algorithms and tests," Astrophys. J. Suppl. Ser. 80, 753–790 (1992).

Кроссплатформенная реализация маскированного умножения разреженных матриц[1]

А.В. Устинов, А.Ю. Пирова, И.Б. Мееров

Нижегородский государственный университет им. Н.И. Лобачевского

Задачи, связанные с графами, часто возникают в задачах анализа сетей, в биоинформатике, в химии. Эффективная реализация алгоритмов на графах – сложная задача. Графы с большим числом вершин невозможно хранить в виде плотной матрицы смежности, поэтому приходится использовать разреженные форматы хранения. Это влечёт нелокальный и непоследовательный доступ к данным, поэтому кэш процессора используется неэффективно, а векторизация наиболее важных циклов часто невозможна. Одним из подходов для повышения производительности является использование матричных операций в графовых алгоритмах. В этом направлении наиболее известен стандарт GraphBLAS [1], в котором определены основные операции разреженной линейной алгебры для реализации алгоритмов на графах. Согласно стандарту, если алгоритм можно представить в виде последовательности операций над разреженными матрицами, то можно сократить время вычислений за счёт эффективной реализации этих операций. Также важным преимуществом подхода является упрощенная переносимость кода, поскольку для получения реализации алгоритма на новой архитектуре достаточно реализовать для нее набор матрично-векторных операций.

Маскированное умножение разреженных матриц – это операция, в которой к операндам применяется маска так, что в произведении отличны от нуля только входящие в маску элементы. Эта операция используется во многих алгоритмах на графах: в поиске в ширину, в подсчёте числа треугольников и других. В рамках исследования разработана кроссплатформенная параллельная реализация маскированного умножения разреженных матриц с использованием библиотеки Kokkos [2]. Затем алгоритм умножения применяется для подсчёта числа треугольников в графе [3] и вычисления степени посредничества вершин в алгоритме Брандеса [4].

Маскированное умножение позволяет использовать структуру маски для уменьшения затрат памяти и объёма вычислений. Обычное умножение разреженных матриц состоит из символьной стадии, где определяется структура произведения и выделяется память для матрицы, и численной стадии, где вычисляются элементы матрицы. В маскированном умножении можно исключить символьную стадию, сразу создавая матрицу на основе маски. Такой подход позволяет заранее ограничить объём памяти, занимаемый итоговой матрицей, и не вычислять структуру произведения матриц. Произведение вычисляется параллельно по строкам, каждая строка есть произведение соответствующей строки левой матрицы на правую матрицу. Для хранения промежуточного результата используется аккумулятор MSA (Masked Sparse Accumulator) [5].

Подсчёт числа треугольников – полных подграфов из трёх вершин – возможно реализовать с использованием маскированного умножения матриц. Для этого из

[1]Работа поддержана Минобрнауки РФ, проект № FSWR-2023-0034.

Таблица 1: Сравнение времени работы алгоритма подсчёта треугольников при использовании OpenMP и Kokkos. Время указано в секундах

	Матрица									
Число	delaunay_n20		pdb1HYS		Freescale		human_gene1		af_shell2	
потоков	OMP	Kokkos	OMP	Kokkos	OMP	Kokkos	OMP	Kokkos	OMP	Kokkos
1	0,096	0,121	0,177	0,159	0,125	0,277	13,853	15,116	0,225	0,305
2	0,044	0,066	0,103	0,082	0,080	0,178	7,114	10,550	0,123	0,184
4	0,027	0,040	0,057	0,046	0,055	0,126	3,799	6,279	0,077	0,096
8	0,020	0,028	0,032	0,032	0,050	0,084	2,186	3,783	0,049	0,074
16	0,024	0,020	0,029	0,018	0,048	0,066	1,615	2,728	0,048	0,052

Таблица 2: Сравнение времени работы алгоритма Брандеса при использовании OpenMP и Kokkos. Время указано в секундах

	Матрица									
Число	human_gene1		kron_g500-logn18		kron_g500-logn20		roadNet-PA		2cubes_sphere	
потоков	OMP	Kokkos	OMP	Kokkos	OMP	Kokkos	OMP	Kokkos	OMP	Kokkos
1	18,451	20,132	29,278	33,524	213,09	172,760	355,45	438,00	5,395	6,901
2	9,768	13,451	15,822	17,974	114,02	96,128	183,18	230,02	2,899	4,058
4	5,593	7,550	9,063	10,486	66,416	55,512	98,664	129,89	1,693	2,670
8	3,488	4,979	5,750	6,696	45,718	38,506	58,315	83,984	1,099	1,889
16	2,385	3,157	4,073	5,287	22,987	27,694	33,052	65,494	0,832	1,338

матрицы смежности графа выделяется нижнетреугольная часть L и выполняется умножение L на L с применением маски L. Числом треугольников является сумма ненулевых элементов полученной матрицы. Использование нижней половины оправдано симметричностью матрицы смежности, в то же время это сокращает вдвое число ненулевых элементов и заметно ускоряет умножение.

Алгоритм Брандеса позволяет вычислить степень посредничества вершин графа – меру важности вершины на основании числа проходящих через неё кратчайших путей. Он состоит из прямого и обратного ходов. Во время прямого хода выполняется поиск в ширину с сохранением порядка обхода вершин, во время обратного хода происходит обработка вершин в обратном порядке с вычислением степени посредничества. Для разных вершин шаги алгоритма можно выполнять независимо, на этом наблюдении основана матричная реализация алгоритма Брандеса. В ней с помощью маскированного умножения разреженных матриц выполняются поиск в ширину в прямом ходе и обход вершин в обратном порядке в обратном ходе. Отметим, что степень посредничества можно вычислить приближенно, если для подсчета числа кратчайших путей использовать не все вершины.

Разработаны две реализации указанных алгоритмов, с использованием технологии OpenMP и библиотеки Kokkos. Вычислительные эксперименты проведены на узле суперкомпьютера «Лобачевский» с двумя 8-ядерными процессорами Intel Sandy Bridge E5-2660 (2.2 Hz), 64 ГБ оперативной памяти, компилятором Intel C++ Compiler 2023. Для тестов использованы матрицы из коллекции Suite Sparse (`https://sparse.tamu.edu/`) размерности от 14 000 до 1 000 000. При тестировании алгоритма Брандеса для вычисления степени посредничества использованы первые 256 вершин каждого графа, то есть поиск в ширину выполнялся именно из них.

В первом эксперименте реализация авторов с использованием OpenMP сравнивалась с реализацией [5] и оказалась быстрее, в среднем, на 30%. Во втором эксперименте сравнивались собственные реализации с использованием OpenMP и Kokkos (таблицы 1, 2) на CPU.

Реализация на Kokkos пока уступает оптимизированной версии на OpenMP, однако является кроссплатформенной и может быть запущена как на CPU, так и на GPU. Накладные расходы в сравнении с OpenMP-реализацией не превышают 1,6 раза для подсчета треугольников и 2 раза на алгоритме Брандеса. Реализация на Kokkos находится в разработке, по опыту предыдущих исследований ожидается, что накладные расходы впоследствии удастся сократить за счет более эффективного использования стандартных примитивов библиотеки. В докладе будут представлены результаты работы оптимизированной версии на Kokkos на CPU и GPU.

Список литературы

[1] Kepner J., Bader D., Buluç A. et al. Graphs, Matrices, and the GraphBLAS: Seven Good Reasons // Procedia Computer Science, 2015, Vol. 51, P. 2453-2462. `https://doi.org/10.1016/j.procs.2015.05.353`

[2] Edwards H.C., Trott C.R. Kokkos: Enabling performance portability across manycore architectures // 2013 Extreme Scaling Workshop (xsw 2013). IEEE, 2013. P. 18-24.

[3] Kepner J., Gilbert J. (ed.). Graph algorithms in the language of linear algebra // Philadelphia: Society for Industrial and Applied Mathematics, 2011.

[4] Brandes U. A faster algorithm for betweenness centrality // Journal of mathematical sociology. 2001. Vol. 25. №. 2. P. 163-177.

[5] Milaković S. et al. Parallel Algorithms for Masked Sparse Matrix-Matrix Products // Proceedings of the 51st International Conference on Parallel Processing, ICPP 2022. Association for Computing Machinery, New York, NY, USA, Article 10, P. 1–11. https://doi.org/10.1145/3545008.3545048

Об исследовании параллельных алгоритмов условной глобальной оптимизации на новом классе модельных задач[1]

Е.С. Пинежанин, К.А. Баркалов

Нижегородский государственный университет им. Н.И. Лобачевского

В работе рассматривается параллельный алгоритм для поиска глобального минимума многоэкстремальной функции $\varphi(y)$ при наличии m ограничений типа неравенств с невыпуклыми левыми частями $g_j(y)$, $1 \le j \le m$. Предполагается, что функции, входящие в постановку задачи, удовлетворяющих условию Липшица с априори неизвестными константами L_j. Таким образом, задача рассматривается в виде

$$\varphi(y^*) = \min\{\varphi(y),\ y \in D,\ g_j(y) \le 0,\ 1 \le j \le m\},$$
$$D = \{y \in R^N \colon a_i \le y_i \le b_i,\ i = \overline{1, N}\}.$$

Предположение липшицевости функций задачи является типичным для многих подходов к разработке алгоритмов глобальной оптимизации [1]. В отличие от широко распространенных метаэвристических алгоритмов, методы липшицевой глобальной оптимизации обладают детерминированным поведением и более быстрой сходимостью к решению задачи [2].

Используемый в работе подход основан на редукции исходной многомерной задачи к эквивалентной ей одномерной задаче с использованием кривых, заполняющих пространство (кривых Пеано) $y(x)$. Непрерывное однозначное отображение $y(x)$, отображающее отрезок $[0, 1]$ на N-мерный гиперкуб D, позволяет свести многомерную задачу условной минимизации в области D к одномерной задаче условной минимизации на отрезке $[0, 1]$

$$\varphi(y(x^*)) = \min\{\varphi(y(x)) \colon x \in [0, 1],\ g_j(y(x)) \le 0,\ 1 \le j \le m\}.$$

Решение редуцированной одномерной задачи может быть выполнено эффективными алгоритмами минимизации функции одной переменной. В качестве такого алгоритма, основанного на оригинальной схеме учета ограничений и не использующего идей метода штрафных функций, исследовался индексный метод глобальной оптимизации [3].

Распараллеливание метода было организовано следующим образом. Каждой подобласти поиска ставится в соответствие число, называемое *характеристикой*. Она определяет перспективность данной подобласти для проведения в ней нового испытания. На каждой итерации алгоритма определяется p лучших подобластей (в соответствии с их характеристиками), в каждой из которых параллельно проводится одно поисковое испытание. Указанный способ обладает общностью (т.к. может быть применен для широкого класса divide-the-best алгоритмов) и эффективностью (т.к. проведение испытаний в прикладных задачах является трудоемкой операцией).

[1]Работа выполнена при поддержке Минобрнауки РФ (проект № FSWR-2023-0034) и НОМЦ «Математика технологий будущего».

Для проведения экспериментов с параллельными оптимизационными алгоритмами обычно используются некоторые генераторы тестовых задач. Как правило, они порождают непосредственно саму целевую функцию, и в случае размерности $N > 2$ здесь сложно наблюдать сам процесс оптимизации.

В связи с этим представляет интерес новый подход, инициированный в работе [4] и продолженный в данном исследовании. В рамках предлагаемого подхода оптимизационная задача возникает как задача идентификации (подбора) параметров ω, c дифференциального уравнения, решение которого $u(t)$ является одномерной кривой (см. рис. 1), что позволяет независимо от числа переменных наглядно наблюдать как текущее, так и окончательное решение задачи. В задаче требуется, чтобы кривая решения $u(t)$ проходила бы через выделенные области (см. рис. 1), причем в последней из них имела бы максимальный наклон, т.е. решается задача

$$|\dot{u}(t, \omega, c)| \to \max$$

при наличии дополнительных ограничения на параметры ω, c.

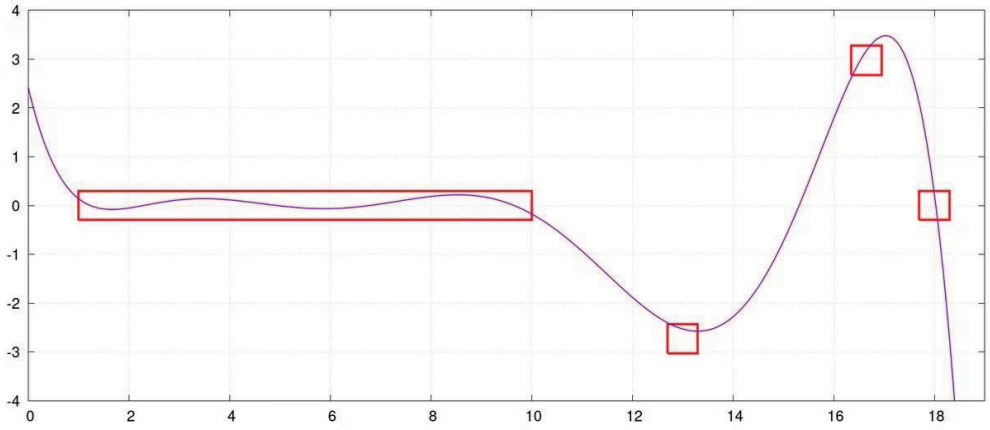


Рис. 1: Пример решения задачи подбора параметров дифференциального уравнения

Программная реализация параллельного алгоритма глобальной оптимизации выполнена на C++ с использованием OpenMP. Эксперименты проводились на узле суперкомпьютера «Лобачевский» (2 процессора AMD EPYC 7742 2.25 GHz, 64 физических ядра, до 128 параллельных потоков в режиме MultiThreading); решалась серия из 20 модельных задач.

Результаты, приведенные в таблице, показывают уменьшение среднего числа итераций, необходимого для решения одной задачи, за счет параллельного проведения нескольких испытаний в разных процессах; однако ускорение по времени достигается

Таблица 1: Среднее число итераций K, время работы T и ускорение S параллельного алгоритма.

p	K	T, сек.	S
1	64 617	49.8	1.0
16	3 971	20.8	2.4
32	2 326	16.5	3.0
64	1 247	11.0	4.5
128	561	5.9	8.5

небольшое. Это объясняется наличием в задаче нескольких ограничений, трудоемкость проверки которых существенно различается в зависимости от значений параметров. Синхронная схема распараллеливания, использованная в проведенных экспериментах, не обеспечила полную загрузку вычислительных потоков. Направлением дальнейших исследований будет использование асинхронной схемы распараллеливания вычислений. Асинхронная схема будет равномерно распределять нагрузку на потоки для задач, в которых время проведения испытаний в разных точках области поиска является различным.

Список литературы

[1] Евтушенко Ю.Г., Посыпкин М.А. Метод неравномерных покрытий для решения задач многокритериальной оптимизации с заданной точностью // Автомат. и телемех. 2014. № 6, С. 49–68.

[2] Sergeyev Y., Kvasov D., Mukhametzhanov M. On the Efficiency of Nature-Inspired Metaheuristics in Expensive Global Optimization with Limited Budget // Scientific Reports. vol. 8(1). Art. No. 435.

[3] Баркалов К.А. Оценки эффективности параллельного индексного метода глобальной оптимизации // Вестник Нижегородского университета им. Н.И. Лобачевского. 2011. №3(2), С. 13–19.

[4] Barkalov K., Strongin R. Test problems for parallel algorithms of constrained global optimization // Lecture notes in computer science. 2017. vol. 10556. P. 18–33.

Определение диалектов для тайлинга циклов в MLIR

А.В. Левченко

Суперкомпьютерный центр СПбПУ

Актуальность работы обусловлена необходимостью совершенствования тайлинга циклов в многоуровневом промежуточном представлении (Multi-Level Intermediate Representation, MLIR) библиотеки LLVM для применения в HPC. Развитие архитектур суперкомпьютеров порождает необходимость определения и массовой генерации различных тайлинговых диалектов в MLIR. Подобное масштабирование тайлинга в MLIR влечет усложнение расписания трансформаций для кодизайна тайлинга совместно с другими преобразованиями циклов в рамках полиэдральной модели [1]. Полученные варианты расписания тайлинга циклов должны быть инкапсулированы в диалекте MLIR для повторного использования.

Вклад данной работы — метод определения и автоматической генерации кода диалектов для тайлинга циклов в MLIR с использованием метапрограммирования (супертайлинг). В качестве метаязыка использован диалект IRDL (Intermediate Representation Definition Language), который был использован для определения новых диалектов для тайлинга в MLIR [2]. Макроструктура взаимодействия IRDL и полученного тайлингового диалекта с ранее разработанным методом низкоуровневого тайлинга продемонстрирована на рис. 1.

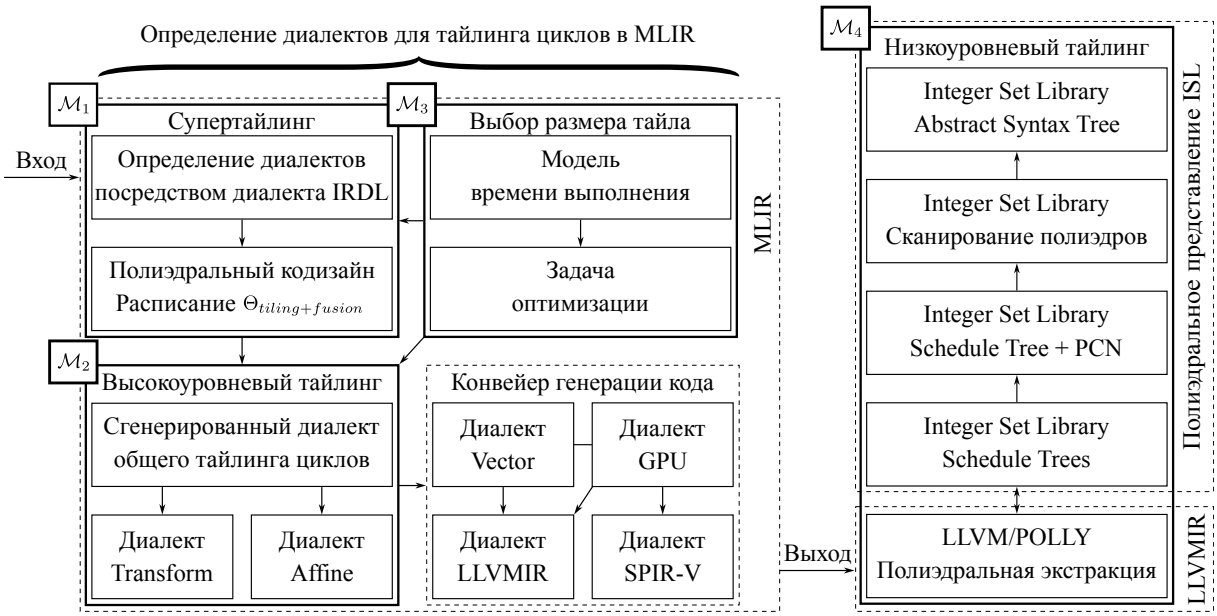
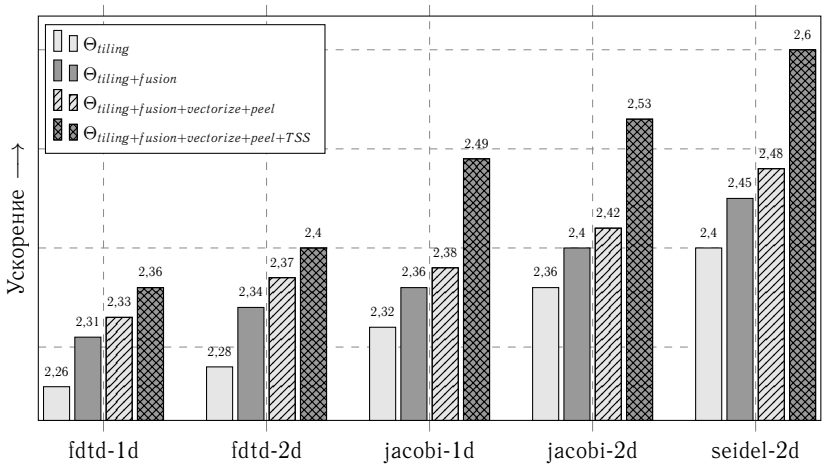


Рис. 1: Супертайлинг (стадии $\mathcal{M}_1$–$\mathcal{M}_3$) в контексте низкоуровневого тайлинга (стадия $\mathcal{M}_4$)

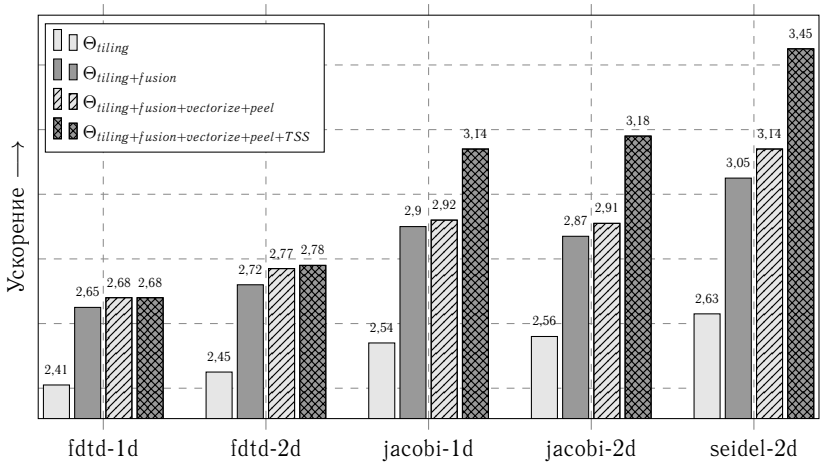
Входными данными метода являются циклы в MLIR. Стадия $\mathcal{M}_1$ содержит применение расширения IRDL-C++ для определения и генерации кода тайлинговых диалектов посредством декларативного описания их операций, атрибутов и типов. При этом определен кодизайн тайлинга и слияния циклов как примитивов полиэдральной модели [1] в виде расписания с использованием диалектов Affine и Transform. Стадия

$\mathcal{M}_2$ содержит применение полученного расписания тайлинга $\Theta_{tiling+fusion}$ совместно с определенным диалектом. Расписание понижено в диалект Transform, что позволяет инкапсулировать ряд трансформаций циклов в виде единой операции, которую можно интегрировать и в другие тайлинговые диалекты, сделав доступной для инфраструктуры MLIR. Стадия $\mathcal{M}_3$ содержит параметризацию операций диалекта посредством подстановки предварительно полученных оптимальных размерностей тайлов (Tile Size Selection, TSS), которые представляют собой совместный цифровой отпечаток программы и целевой архитектуры. Размерности тайлов определены в виде атрибутов операций посредством IRDL-C++. *Выходными данными* метода является LLVMIR. В дополнение рис. 1 демонстрирует расширение метода стадией $\mathcal{M}_4$, которая содержит низкоуровневый тайлинг посредством библиотек LLVM/Polly и ISL. Здесь выполняется генерация деревьев частичного расписания циклов с параметрическими управляющими узлами (Parametric Control Nodes, PCN), в которые включаются функции расписания для тайлинга. Таким образом, тайлинг масштабирован и за пределы MLIR.

Экспериментальные результаты включают оценку влияния тайлинга циклов для CPU на ускорение тестовых программ с AMD EPYC 7763 (рис. 2). Для оцен-



(a) Ускорение тайлового кода MLIR ($\mathcal{M}_1$–$\mathcal{M}_3$)



(b) Общее ускорение тайлового кода ($\mathcal{M}_1$–$\mathcal{M}_4$)

Рис. 2: Ускорение тайлового кода тестовых программ относительно базового тайлинга MLIR

ки ускорения были использованы наиболее репрезентативные задачи класса трафаретных вычислений (stencils) FDTD-1D/2D, Jacobi-1D/2D и Seidel-2D. Так, на рис. 2, *a* представлены результаты диалекта для тайлинга (стадии $\mathcal{M}_1$–$\mathcal{M}_3$), полученного в результате применения метода определения диалектов посредством IRDL-C++. Продемонстрированы результаты для расписания тайлинга, включающие изолировано тайлинг (Θ_{tiling}), тайлинг и слияние циклов ($\Theta_{tiling+fusion}$), пилинг и векторизацию ($\Theta_{tiling+vectorize+peel}$) с дополнительным уточнением размерностей тайлов ($+TSS$). Кроме того, на рис. 2, *b* результаты стадий $\mathcal{M}_1$–$\mathcal{M}_3$ объединены с результатами конвейера, который дополнительно подключает и низкоуровневый тайлинг LLVMIR на стадии $\mathcal{M}_4$. Таким образом, показано, что во всех случаях применение диалекта, определенного на стадиях $\mathcal{M}_1$– $\mathcal{M}_3$, дало ускорение как по сравнению с базовым тайлингом в MLIR, так и на общем фоне конвейера компиляции, представленного на рис. 1.

Выводы о практической значимости метода обоснованы возможностью определения диалектов (стадия $\mathcal{M}_1$) и повторного использования расписания для тайлинга (стадия $\mathcal{M}_2$). Будущая работа будет сфокусирована на возможности дополнительной параметризации операций диалектов с использованием специфической информации о целевой архитектуре.

Список литературы

[1] Jie Zhao, Jinchen Xu, Peng Di, Wang Nie, Jiahui Hu, Yanzhi Yi, Sijia Yang, Zhen Geng, Renwei Zhang, Bojie Li, Zhiliang Gan, Xuefeng Jin. Modeling the Interplay between Loop Tiling and Fusion in Optimizing Compilers Using Affine Relations // ACM Transactions on Computer Systems. January, 2024. Vol. 41. Issue 1–4. Article 5. `https://doi.org/10.1145/3635305`

[2] Mathieu Fehr, Jeff Niu, River Riddle, Mehdi Amini, Zhendong Su, Tobias Grosser. IRDL: an IR Definition Language for SSA Compilers // Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation. San Diego, USA. June 13–17, 2022. P. 199–212. `https://doi.org/10.1145/3519939.3523700`

Оптимизация нейронных сетей для запуска на ускорителях с использованием компиляторных технологий[1]

А.А. Оболенский, А.В. Горшков, И.Б. Мееров

Нижегородский государственный университет им. Н.И. Лобачевского

В настоящее время задача оптимизации вывода (inference) и тренировки (training) нейронных сетей с использованием различного рода ускорителей стоит достаточно остро. Различные производители предлагают множество решений для запуска нейронных сетей на своем оборудовании, крупные компании разрабатывают свои платформы для запуска сетей. Однако ключевой проблемой в этой области остается неоднородность инфраструктур, предоставляемых различными поставщиками. Это создает сложности в интеграции и масштабировании решений, так как каждый производитель стремится поддерживать лишь собственные программные и аппаратные платформы, что приводит к изоляции технологических решений. Например, использование тензорных процессоров (Tensor Processing Units, TPU) от Google представляет собой один из примеров закрытой архитектуры, которая эффективно работает в рамках экосистемы Google, но может вызывать сложности при попытке интеграции с другими системами.

Понимая проблему несовместимости существующих решений, разработчики аппаратуры пытаются предоставить поддержку для популярных программных продуктов, в частности, для решения задач машинного обучения. В данной работе рассматривается одна из аппаратных платформ для вывода нейронных сетей – Ascend NPU [1], производимая Huawei. В качестве программной модели разработчик предоставляет язык программирования высокого уровня Ascend C, основанный на C++. Главным препятствием на пути широкого распространения этого языка является необходимость его изучения программистами, разрабатывающими и применяющими методы машинного обучения для решения задач. Одним из способов упрощения использования платформы Ascend является поддержка какого-либо упрощенного языка программирования нейронных сетей, а также транслятора с этого языка в Ascend C. Данный подход имеет смысл в случае, если «промежуточный язык» будет прост в освоении, а реализованный транслятор во многом возьмет на себя оптимизацию кода для Ascend NPU.

Долгосрочной целью нашей работы является изучение и реализация трансляции из общепринятого высокоуровневого представления описания нейронных сетей в высокопроизводительные коды, оптимизированные для различных ускорителей нейронных сетей (в том числе, Ascend NPU) и CPU (в том числе, на архитектуре ARM 64-bit).

Для достижения поставленной цели требуется решить ряд задач:

1) Исследовать возможные входные представления слоев нейронных сетей и самих сетей для трансляции, выбрать наиболее подходящие по ряду критериев.

[1]И.Б. Мееров благодарит Минобрнауки РФ за поддержку в рамках проекта № FSWR-2023-0034.

2) Выбрать инфраструктуру и подход для трансляции кода.
3) Оценить особенности, которые нужно учесть при трансляции, определить список оптимизаций, которые нужно реализовать.
4) Построить математические модели для обоснования расширяемости, применимости и масштабируемости используемых оптимизаций.
5) Представить прототип транслятора, позволяющего сконвертировать выбранное представление в Ascend C с сопоставимым уровнем производительности.

Обзор литературы по данной тематике показал, что существуют программные инструменты, которые позволяют конвертировать код, написанный для GPU в код, который может быть исполнен на CPU. Были рассмотрены два обобщенных подхода к трансляции: прямая конверсия бинарного кода GPU в бинарный код CPU (binary-to-binary) и поэтапная конвертация GPU-кода в код для CPU – из исходного кода GPU в бинарный код для CPU (source-to-binary) или в исходный код для CPU (source-to-source).

В данный момент мы рассматриваем возможность использования различных представлений для конвертации. Исследования показали, что CUDA C++ представление зачастую оказывается слишком низкоуровневым для проведения сложных трансформаций. Инструмент Polygeist [2] конвертирует код из C++ в MLIR, однако полученный в результате код не удается адаптировать для высокоуровневого API Ascend C. Это проявляется в том, что в CUDA C++ коде операции могут быть разбиты до элементарных операций, которые не могут быть описаны абстракциями типа тензоров. Это, в свою очередь, лишает компилятор возможности установить, что операции могут быть векторизованы, а также лишает возможности использовать высокоуровневые векторные и матричные инструкции, доступные в Ascend C. Таким образом, нужно рассмотреть более подходящие программные модели. В рамках обзора существующих решений были найдены следующие варианты: OpenAI Triton [3], MLIR-бекенды пакетов машинного обучения (PyTorch, Tensorflow, ONNX) [4].

Далее было произведено сравнение в контексте использования, исходя из особенностей конкретной архитектуры. Так, в архитектуре NPU Ascend 910 принята модель разделения вычислительных операций на множество специализированных блоков. В частности, архитектура включает в себя скалярные, векторные блоки и блоки для выполнения матричных операций. Скалярный блок выполняет функции координации вычислений, направляя задачи в аппаратные очереди векторного блока и блока кубических вычислений. Несмотря на то, что скалярный блок обладает способностью к выполнению вычислений, его вычислительная мощность ограничена, и он значительно уступает по скорости другим блокам.

Таким образом, при конвертации кода нужно учесть несколько важных аспектов, характерных для данного железа:

- Наличие внутреннего кэша достаточно маленького размера и отсутствие прямого доступа в глобальную память. Это означает, что операции загрузки из глобальной памяти в локальную и наоборот (ввод-вывод) критически влияют на производительность.
- Правильное разделение доступа к входным и к выходным данным, организация работы произвольного доступа в глобальную память, учитывая ограничения оборудования.

- Сложность архитектуры, которая предполагает разделение разного рода вычислений на скалярные, векторные и матричные. Основные сложности заключаются в разделении векторных и матричных вычислений из-за особенностей разделения памяти на более новом оборудовании.

По итогам исследования был выбран стек технологий, подходящий для использования компиляторных методов для конвертации и использования оптимизаций на оборудовании Huawei Ascend с возможностью запуска на ARM с использованием векторных расширений NEON/SVE. За основу взят компиляторный стек LLVM, а именно инфраструктура MLIR [5]. На данный момент уже реализована базовая конвертация для некоторых векторных операций из MLIR в код Ascend C. В докладе будет представлено текущее состояние исследований и разработок, в частности общая схема предлагаемого подхода и первые примеры его использования.

Список литературы

[1] Ascend Computing.
`https://e.huawei.com/en/products/computing/ascend`

[2] Moses W.S. et al. Polygeist: Raising C to polyhedral MLIR // 2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT). – IEEE, 2021. – P. 45–59.

[3] Banerjee S. et al. Triton: Software-Defined Threat Model for Secure Multi-Tenant ML Inference Accelerators // Proceedings of the 12th International Workshop on Hardware and Architectural Support for Security and Privacy. – 2023. – P. 19-28.

[4] Zhang H. et al. Compiler technologies in deep learning co-design: A survey // Intelligent Computing. – 2023. – V. 2. – P. 0040.

[5] Lattner C. et al. MLIR: Scaling compiler infrastructure for domain specific computation // 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO). – IEEE, 2021. – P. 2–14.

Оценка быстродействия параллельного алгоритма пакетного решения линейных систем с трехдиагональными матрицами различной размерности на графических процессорах

А.С. Добровольцев, М.А. Сохатский, А.В. Юлдашев

Уфимский университет науки и технологий

Метод AIPS [1], аппроксимирующий обратную матрицу на основе степенного разложения в ряд Неймана, зарекомендовал себя в качестве масштабируемого предобусловливателя, позволяющего обеспечить ускорение итерационного решения систем линейных алгебраических уравнений (СЛАУ) с разреженной матрицей при гидродинамическом моделировании нефтегазовых месторождений на графических процессорах (GPU) [2]. Применение данного предобусловливателя в итерационном процессе предполагает выполнение на каждой итерации умножения на вектор матрицы M_N^{-1}, аппроксимирующей обратную матрицу при выполнении условия $\rho(P^{-1}R) < 1$ по следующей формуле:

$$A^{-1} \approx M_N^{-1} = \left[E + \sum_{k=1}^{N} (-1)^k (P^{-1}R)^k\right] P^{-1}, \tag{1}$$

где A – матрица решаемой СЛАУ, E – единичная матрица, P, например, – трехдиагональная часть матрицы A, $R = A - P$, N – количество используемых членов ряда Неймана, например, равное 10. Процедуру применения данного предобусловливателя целесообразно выполнять неявным способом путём умножения на вектор правой части выражения (1), что требует выполнения N умножений на вектор матрицы R, а также многократного ($N+1$ раз) решения систем с трехдиагональной матрицей P.

В работе [1] был предложен ориентированный на архитектуру CUDA параллельный алгоритм пакетного решения линейных систем с трехдиагональными матрицами различной размерности. Идея данного алгоритма заключается в параллельном применении метода прогонки для независимых блоков СЛАУ с трехдиагональной матрицей, которые предварительно сортируются, а элементы которых переставляются, чтобы обеспечить балансировку нагрузки и более производительный доступ к памяти GPU. В настоящее время ведётся развитие предложенного алгоритма в целях расширения сферы его применимости, а также повышения быстродействия. Так для повышения ресурса параллелизма алгоритма рассматривается применение повторной встречной прогонки [3] вместо повторной монотонной прогонки.

Проведено сравнение быстродействия алгоритма при решении СЛАУ с трехдиагональными матрицами, характеристики которых приведены в таблице 1, с альтернативными параллельными алгоритмами, реализованными в библиотеке cuSPARSE [4]: cusparse<t>gtsv2 и cusparse<t>gtsv2_nopivot – функции, предназначенные для решения трехдиагональных СЛАУ общего вида, а также cusparse<t>gtsv2StridedBatch и cusparse<t>gtsvInterleavedBatch – функции для пакетного решения множества

трехдиагональных СЛАУ одинаковой размерности, которые асинхронно вызывались в нескольких потоках (CUDA Streams), чтобы обеспечить возможность распараллеливания обработки трехдиагональных СЛАУ различного размера.

Таблица 1: Характеристики трехдиагональных матриц

Название матрицы	Размерность матрицы	Минимальный размер блока	Максимальный размер блока	Количество независимых блоков
immn3	768 034	1	34	221 402
krrv3	1 440 307	1	39	233 866
fdrv3	2 203 421	1	55	633 202
kmms3	2 876 965	1	121	389 199
lkms3	4 555 235	1	27	1 167 013

Отметим, что функция cusparse<t>gtsvInterleavedBatch реализует более производительный относительно cusparse<t>gtsv2StridedBatch доступ к памяти GPU, но требует выполнения перестановок элементов трехдиагональных СЛАУ, как и в исследуемом алгоритме. Также необходимо учесть, что в ходе выполнения функции cusparse<t>gtsvInterleavedBatch происходит обновление элементов матрицы.

В таблице 2 приведены осредненные замеры времени решения СЛАУ с использованием различных алгоритмов на вычислительном узле, содержащем 2 x CPU Intel Gold 6226, GPU NVIDIA V100 и GPU NVIDIA A100 (ОС CentOS 7.9, CUDA Toolkit 11.6). Расчеты проведены в двойной точности.

Таблица 2: Время решения СЛАУ с трехдиагональными матрицами, мс

GPU	Алгоритм	immn3	krrv3	fdrv3	kmms3	lkms3
V100	gtsv2	0,36	0,55	0,89	1,07	1,60
	gtsv2_nopivot	0,33	0,59	0,89	1,17	1,83
	gtsv2StridedBatch	0,48	0,70	1,30	1,47	2,55
	gtsvInterleavedBatch	0,25	0,39	0,58	0,83	0,91
	Алгоритм 1	0,12	0,23	0,37	0,48	0,77
A100	gtsv2	0,26	0,38	0,60	0,73	1,08
	gtsv2_nopivot	0,19	0,31	0,46	0,58	0,88
	gtsv2StridedBatch	0,26	0,41	0,69	0,82	1,37
	gtsvInterleavedBatch	0,18	0,19	0,27	0,40	0,44
	Алгоритм 1	0,08	0,13	0,20	0,29	0,36

Проведенное сравнительное тестирование показывает, что разработанный параллельный алгоритм позволяет ускорить решение СЛАУ в 1,2–3,9 раза относительно параллельных алгоритмов решения трехдиагональных линейных систем, реализованных в библиотеке cuSPARSE. Далее планируется провести оценку быстродействия алгоритма на большем наборе матриц, включающем трехдиагональные матрицы с мелкозернистой блочностью, в которых диагонали состоят не из скалярных элементов, а квадратных блоков небольшого размера: 2×2, 3×3 и т.д.

Список литературы

[1] Юлдашев А.В., Репин Н.В., Спеле В.В. Параллельный предобусловливатель на основе степенного разложения обратной матрицы для решения разреженных линейных систем на графических процессорах // Вычислительные методы и программирование. 2019. Т. 20. С. 444-456.
https://doi.org/10.26089/NumMet.v20r439

[2] Сайфуллин И.Ф., Шарипов Т.Р., Спеле В.В., Штангеев А.Л., Юлдашев А.В. Опыт использования графических процессоров для ускорения расчетов при гидродинамическом моделировании // Тезисы докладов научно-технической конференции «Цифровые технологии в добыче и переработке углеводородов: от моделей к практике» (6-9 октября 2020 г., г. Уфа). – М.: ЗАО «Изд-во «Нефтяное хозяйство»», 2020. – С. 83-84.

[3] AlgoWiki: Открытая энциклопедия свойств алгоритмов. Повторная встречная прогонка, точечный вариант.
http://algowiki-project.org/ru/Повторная_встречная_прогонка,_точечный_вариант
(дата обращения: 25.04.2024).

[4] cuSPARSE.
https://docs.nvidia.com/cuda/cusparse/
(дата обращения: 25.04.2024).

Оценка производительности суперкомпьютера “cHARISMa” и его компонент для квантовой химии на примере пакетов в CP2K и Quantum Espresso

И.Э. Недомолкин[1], М.П. Коников[1], В.В. Стегайлов[2,1], А.В. Тимофеев[2,1], И.Д. Федоров[2,1]

[1]Национальный исследовательский университет «Высшая школа экономики», [2]ОИВТ РАН

Суперкомпьютеры важны для науки и инженерии, их применение растет. Современные способы оценки производительности универсальны и базируются на общих математических задачх. Однако для большей точности лучше использовать научные пакеты, при оценке производительности в специфических задачах, например, в квантовой химии.

Существует множество авторитетных рейтингов, основанных на различных бенчмарках, таких как NAS parallel benchmark [1], SPEC benchmark [2], HPC Challenge benchmark [3] и LINPACK [4].

На основе анализа статей и обсуждения с пользователями программ были выбраны программные пакеты Quantum Espresso [5][6] и CP2K [7][8]. Они являются достаточно популярными в научной сфере, распространяются бесплатно и имеют открытый исходный код.

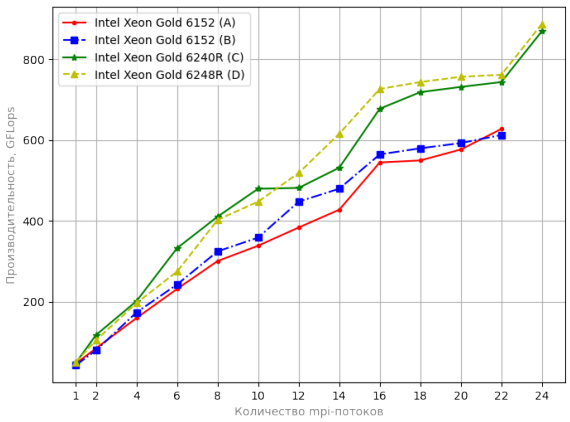


Рис. 1: Производительность в GFLops (CP2K)

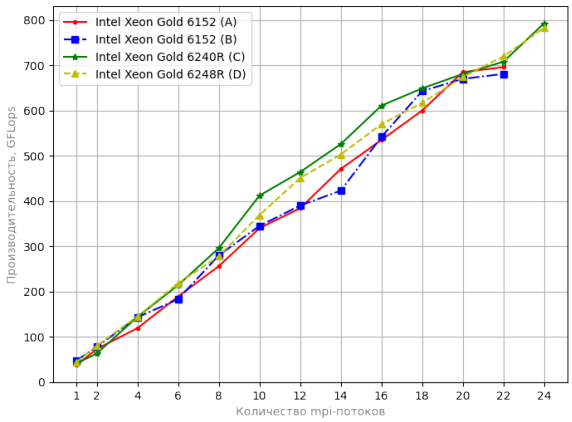


Рис. 2: Производительность в GFLops (QE)

Был проведен анализ производительности узлов суперкомпьютера “cHARISMa”. На графиках представлены показатели производительности в двух приложениях, рис. 1 и рис. 2 демонстрирующие производительность узлов суперкомпьютера при

разном числе mpi потоков. Узел типа D (Intel Xeon Gold 6248R) демонстрирует наиболее эффективные вычисления среди узлов суперкомпьютера при решении квантовохимических задач. А по результатам нормировки, данные которых представлены в таблице 1 и таблице 2, узел типа C (Intel Xeon Gold 6240R) оказался наиболее экономически эффективным

Таблица 1: Нормировка на цену и на энергопотребление (Quantum Espresso)

$\mathbf{N}_{cores}$	Нормировка на цену в (GFLops/$)				Нормировка на энергопотребление (GFLops/W)			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**
1	0,010	0,009	0,011	0,012	0,263	0,223	0,291	0,317
full	0,190	0,186	0,217	0,214	4,972	4,864	5,665	5,594

Таблица 2: Нормировка на цену и на энергопотребление (CP2K)

$\mathbf{N}_{cores}$	Нормировка на цену в (GFLops/$)				Нормировка на энергопотребление (GFLops/W)			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**
1	0,0129	0,0118	0,019	0,016	0,336	0,309	0,299	0,251
full	0,172	0,167	0,335	0,279	4,448	4,382	5,276	4,334

В будущем планируется произвести анализ узких мест в работе суперкомпьютера. Также уже проводится анализ программных пакетов с использованием профилировщиков, для нахождения самых трудозатратных функций при работе пакета.

Благодарности: Создание и оптимизация бенчмарков, а также анализ результатов выполнены при финансовой поддержке Российского научного фонда (проект № 20-71-10127), `https://rscf.ru/project/20-71-10127/`. Тестирование узлов суперкомпьютера осуществлено в рамках Программы фундаментальных исследований НИУ ВШЭ с использованием суперкомпьютерного комплекса НИУ ВШЭ [9].

Список литературы

[1] Bailey D.H. et al. The NAS parallel benchmarks-summary and preliminary results //Proceedings of the 1991 ACM/IEEE Conference on Supercomputing, 1991. С. 158–165. (the NAS parallel benchmark suites)

[2] SPEC: Standard performance evaluation corporation.
`http://www.spec.org`
2005. (the SPEC benchmark)

[3] Luszczek P. et al. Introduction to the HPC challenge benchmark suite. – 2005. (the HPC Challenge benchmark)

[4] Dongarra J.J., Luszczek P., Petitet A. The LINPACK benchmark: past, present and future // Concurrency and Computation: practice and experience. – 2003. – Т. 15. – №. 9. – С. 803-820. (LINPACK)

[5] Quantum ESPRESSO site.
https://www.quantum-espresso.org

[6] User's Guide for QUANTUM ESPRESSO (v.7.2).
https://www.quantum-espresso.org/Doc/user_guide/

[7] CP2K.
https://www.cp2k.org/

[8] Thomas D.K. "CP2K: An electronic structure and molecular dynamics software package Quickstep: Efficient and accurate electronic structure calculations". B: The Journal of Chemical Physics 152.19 (2020), c. 194103.

[9] Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I. HPC Resources of the Higher School of Economics // Journal of Physics: Conference Series. 2021. Vol. 1740, No. 1. P. 012050.
https://doi.org/10.1088/1742-6596/1740/1/012050

Оценочное тестирование эффективности вычислительных узлов суперкомпьютера cHARISMa для задач глубокого обучения

В.В. Писарев[1,2], Г.В. Промыслов[1], А.В. Тимофеев[2,1]

[1]Национальный исследовательский университет
"Высшая школа экономики",
[2]Объединенный институт высоких температур РАН

В настоящее время суперкомпьютерные системы играют важную роль в научных и инженерных исследованиях, в том числе в решении сложных задач машинного обучения. Этому способствует возможность суперкомпьютеров выполнять огромные объемы вычислений в кратчайшие промежутки времени, что позволяет находить решения, поиск которых был бы невозможен для традиционных вычислительных систем.

Одним из ключевых показателей производительности суперкомпьютеров является вычислительная эффективность выполнения поставленных задач. Стандартным подходом для оценки производительности математических операций является тест High Performance Linpack [1], однако данный бенчмарк не даёт исчерпывающее представление о эффективности расчётов для других вычислительных задач. В связи с этим появляются альтернативные оценки производительности, в том числе при помощи бенчмарков приближенных к реальным научным приложениям.

Цель работы состоит в подготовке методики оценивания эффективности суперкомпьютеров и их компонент при работе с задачами машинного обучения [2]. На данном этапе выбран бенчмарк ai-benchmark [3], позволяющий оценить эффективность расчётов на примеры работы 19 нейронных сетей на стадиях обучения и работы, что покрывает большинство архитекутур глубокого обучения, используемых в современных задачах. На текущем этапе расчёт проводился в основном на графических ускорителях, доступных на суперкомпьютере cHARISMa. Результаты выполнения бенчмарка на нескольких типах узлов с одинаковой видеокартой позволяют исследовать влияние окружения, такого как центральный процессор и оперативная память на эффективность выполнения задач глубокого обучения.

Исследование выполнено при финансовой поддержке Российского научного фонда (проект № 20-71-10127), `https://rscf.ru/project/20-71-10127/` с использованием суперкомпьютерного комплекса НИУ ВШЭ [4].

Список литературы

[1] Dongarra J. The LINPACK Benchmark: An explanation // Lecture Notes in Computer Science. 1988. Vol. 297. P. 456–474.

[2] Alpaydin E. Machine Learning. MIT Press, 2021. 280 p.

[3] Ignatov A. AI Benchmark.
`https://ai-benchmark.com`
(дата обращения: 25.04.2024)

[4] Kozyrev V.I., Kostenetskiy P.S., Chulkevich R.A. HPC Resources of the Higher School of Economics // J. Phys.: Conf. Series. 2021. Vol. 1740. P. 012050.

Разработка бенчмарка для устройств на архитектуре RISC-V на основе одной задачи биоинформатики[1]

М.А. Козлов, В.Д. Волокитин, Е.А. Панова, И.Б. Мееров

Нижегородский государственный университет им. Н.И. Лобачевского

В последнее время всё большее внимание привлекает область разработки программного обеспечения для устройств с процессорами новой архитектуры RISC-V. На данный момент их производительность требует улучшения, однако в будущем она может стать сопоставимой с производительностью устройств на более устоявшихся архитектурах. Особый интерес представляет работа подсистемы памяти. Мы исследуем возможности ускорения алгоритмов, производительность которых ограничена работой с памятью, за счет использования различных техник ускорения вычислений на архитектурах RISC-V и x86. Наша цель – сравнить полученные результаты и показать, что уже сейчас устройства на новой архитектуре могут быть сравнимы по производительности с устройствами на более распространенных архитектурах.

В качестве примера мы рассматриваем задачу из области биоинформатики. Задача заключается в поиске наиболее часто встречающихся паттернов небольшой длины во фрагменте ДНК. Предполагается, что такие последовательности могут нести в себе генетическую информацию. ДНК можно представить как последовательность четырехбуквенного алфавита (“A”, “C”, “G”, “T”), где каждая буква соответствует одному из азотистых оснований. Таким образом, задача эквивалентна поиску наиболее часто встречающихся подстрок в строке и может быть решена при помощи строковых алгоритмов. Известно, что производительность таких алгоритмов ограничена операциями с памятью. Бенчмарк подсистемы памяти, написанный на основе таких алгоритмов, будет востребован во многих областях. Данная работа является продолжением исследования производительности строковых алгоритмов на различных архитектурах [1].

В работе рассматриваются три различных алгоритма по поиску количества вхождения подстроки в строку. В качестве базового рассматривается наивный алгоритм, сравнивающий посимвольно каждую непрерывную подстроку определенной длины с искомой последовательностью. В качестве альтернативы наивному алгоритму рассматривается основанный на хешировании паттернов алгоритм Рабина-Карпа [2] в двух вариациях. В одной из версий для паттерна и каждой подстроки вычисляется кольцевая полиномиальная хеш-функция (Rolling hash) [3], посимвольное сравнение происходит только в случае совпадения хешей. Во второй реализации используется хеш-функция, учитывающая совпадение только двух первых и двух последних символов (SWAR подход, [4]). Стоит отметить, что данная хеш-функция приводит к гораздо большему числу коллизий относительно полиномиальной хеш-функции, однако может вычисляться эффективно в векторном режиме. Третий рассмотренный алгоритм Hash3 [5] весьма эффективен на задачах с небольшим алфавитом, практически не производит посимвольных сравнений, поэтому время вычислений зависит

[1]Работа поддержана Минобрнауки РФ, проект № FSWR-2023-0034. Авторы благодарят ННГУ за доступ к вычислительным ресурсам суперкомпьютера «Лобачевский».

только от работы с памятью. Все алгоритмы имеют сложность $O(mn)$, где m – длина искомой подстроки, а n – длина исходного текста. С учетом того, что поиск проходит для каждой подстроки исходного текста, итоговая сложность алгоритма – $O(mn^2)$. В приведенных ниже экспериментах $n = 43794$, $m = 128$.

Запуски производились на платформах 2x Intel Xeon Silver 4310T (x86, 2.30 ГГц, 2x10 ядер, AVX512, 64 ГБ DDR4-2667) и TH1520 RISC-V Xuantie C910 (RISC-V, 2.0 ГГц, 4 ядра, 16 ГБ LPDDR4X-3733). Для каждой платформы были разработаны две реализации алгоритмов: скалярная и векторная. В векторных реализациях для всех алгоритмов производилось векторное посимвольное сравнение строк. В алгоритме Рабина-Карпа с хэш-функцией SWAR также выполнялось векторное сравнение двух первых и последних символов. Векторные версии были разработаны с использованием специфичных для архитектур RISC-V и AVX512 функций-интринсиков. Исходные коды публично доступны [6].

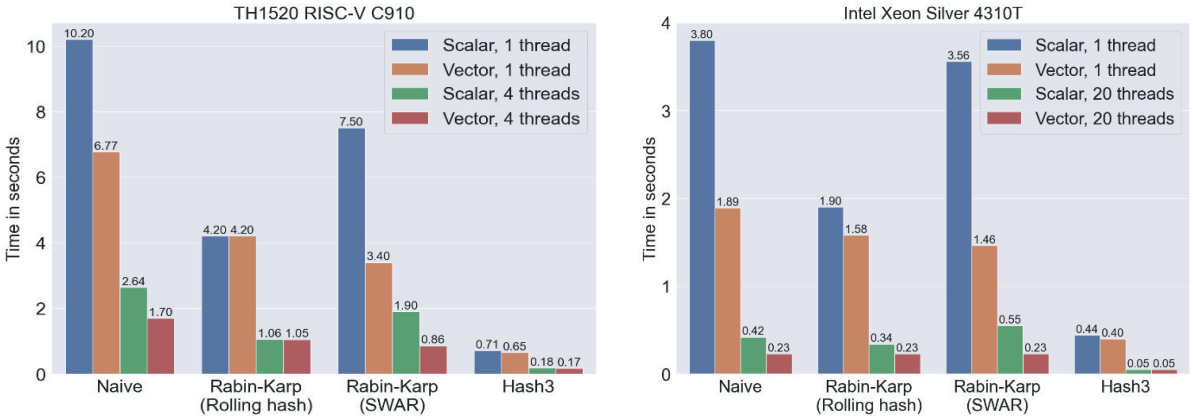


Рис. 1: Время работы алгоритмов поиска наиболее частых паттернов в ДНК на платформах архитектуры RISC-V (слева) и x86 (справа). Рассмотрены конфигурации в однопоточном и многопоточном режиме с использованием и без использования векторных вычислений.

Можно видеть (рис. 1), что на архитектуре RISC-V скалярная и векторная версии всех алгоритмов демонстрируют линейное ускорение при распараллеливании. Векторные реализации многопоточной и однопоточной версии ускоряются ожидаемым образом. Наивный алгоритм и алгоритм Рабина-Карпа с хеш-функцией SWAR, производящие большое количество посимвольных сравнений, получили ускорение ~ 2 раз от векторизации, что является хорошим результатом относительно других рассмотренных алгоритмов. Алгоритм Рабина-Карпа с кольцевым хешем и Hash3 не ускорились за счет векторизации, поскольку посимвольные сравнения строк занимают малую часть их работы. Таким образом, за счет векторизации и параллелизма наивный алгоритм ускорился в 6.2 раза, алгоритм Рабина–Карпа с подсчетом кольцевого хеша в 4 раза, алгоритм Рабина-Карпа с хеш-функцией SWAR в 8.7 раз, алгоритм Hash3 в 4 раза.

Для сравнения были разработаны реализации перечисленных алгоритмов для архитектуры x86. Результаты концептуально схожи с результатами на архитектуре RISC-V. Можно заметить, что, во-первых, относительная производительность векторной версии некоторых алгоритмов на архитектуре x86 получилась на 20-40% больше, чем на архитектуре RISC-V, во-вторых, наблюдается больший прирост от параллелизма по причине большего количества физических ядер и каналов памяти. На основании проведенных экспериментов мы можем сделать вывод, что результаты

ускорения строковых алгоритмов при использовании параллелизма и векторизации на RISC-V сравнимы с результатами, полученными на x86. Таким образом, даже маломощная версия устройства архитектуры RISC-V позволяет проводить эксперименты с векторизацией (с использованием интринсиков) и распараллеливанием и получать существенное ускорение, несмотря на достаточно сложные алгоритмы с малым числом вычислительных операций.

Список литературы

[1] Козлов М.А., Панова Е.А., Мееров И.Б. Реализация поиска наиболее часто встречающихся последовательностей ДНК с использованием библиотеки Kokkos // Проблемы информатики. – 2024 (принято к печати).

[2] Plaxton G. String Matching: Rabin-Karp Algorithm //Theory in Programming Practice. University of Austin, Texas. – 2005.

[3] Kuznetsov A.V., Maysnikov V.V. A fast plain copy-move detection algorithm based on structural pattern and 2D Rabin-Karp rolling hash // Image Analysis and Recognition: ICIAR 2014, Proceedings, Part I 11. – Springer International Publishing, 2014. – P. 461-468.

[4] Mula W. SIMD-friendly algorithms for substring searching.
`http://0x80.pl/articles/simd-strfind.html`
(дата обращения: 25.03.2024).

[5] Lecroq T. Fast exact string matching algorithms // Information Processing Letters. – 2007. – Т. 102. – № 6. – С. 229-235.

[6] Репозиторий проекта.
`https://github.com/Mishaizlesa/most_common_string`
(дата обращения: 25.03.2024).

Разработка и применение сервис-ориентированных рабочих процессов в гетерогенной вычислительной среде[1]

М.Л. Воскобойников, А.Г. Феоктистов

Институт динамики систем и теории управления им. В.М. Матросова СО РАН

Рассматриваются актуальные вопросы разработки и применения рабочих процессов научных приложений в гетерогенной распределенной вычислительной среде. Такая среда интегрирует высокопроизводительные ресурсы пользователей, суперкомпьютерные ресурсы центров коллективного пользования (например, HPC-кластеры) и облачные ресурсы. В этой связи возникают проблемы предоставления разнообразных способов доступа к прикладному программному обеспечению (ПО), его выполнения, масштабирования вычислений и эффективного использования интегрируемых ресурсов. В известных системах управления рабочими процессами (СУРП), таких как Pegasus, HyperFlow, Workflow-as-a-Service Cloud Platform, Galaxy, Apache Airflow и др. [1, 2], решению этих проблем уделяется пристальное внимание. К сожалению, анализ возможностей СУРП показывает, что в полной мере эти проблемы до сих пор не решены [3].

Представлен прототип нового инструментального комплекса Framework for Development and Execution of Scientific WorkFlows (FDE-SWFs) для разработки, отладки, тестирования, доставки, развертывания и выполнения научных рабочих процессов (НРП). FDE-SWFs наследует основной функционал своего предшественника Orlando Tools [4] и существенно развивает его возможности. Он поддерживает разработку как НРП, базирующихся на запуске исполняемых модулей, так и сервис-ориентированных НРП. Архитектура FDE-SWFs включает следующие основные подсистемы: пользовательский интерфейс, конструкторы вычислительной модели и библиотек модулей, менеджеры ПО, вычислений и состояния среды, API для доступа к ресурсам. База знаний содержит описание предметных областей приложений, вычислительных моделей и информацию о ресурсах. Исходные данные и результаты вычислений хранятся в расчетных базах данных. Доступ к НРП осуществляется через интерфейс приложения. Кроме того, сервис-ориентированные НРП представляются на стандартизированном языке Business Process Execution Language (BPEL) и могут быть автономно использованы в любых системах, поддерживающих данный стандарт. Дополнительно FDE-SWFs позволяет реализовать доступ к НРП с помощью стандарта Web Processing Service (WPS). Это обеспечивает гибкую интеграцию научных приложений с геоинформационными системами в рамках решения ресурсоемких задач экологического мониторинга.

[1]Работа выполнена в рамках гранта № 075-15-2024-533 Министерства науки и высшего образования РФ на выполнение крупного научного проекта по приоритетным направлениям научно-технологического развития (проект «Фундаментальные исследования Байкальской природной территории на основе системы взаимосвязанных базовых методов, моделей, нейронных сетей и цифровой платформы экологического мониторинга окружающей среды»).

FDE-SWFs реализован на программной платформе Node.js. В дополнение к прикладному ПО разрабатываемых приложений он предоставляет следующие системные модули для конструирования НРП: модули публикации и вызова сервисов WPS; конфигурирования и настройки вычислительной среды, а также формирования заданий для традиционных систем управления заданиями (СУЗ), таких как PBS Torque, HTCondor, Slurm и др., и метапланировщиков (Condor DAGMan, GridWay и др.) среды; взаимодействия с СУЗ и метапланировщиками среды; взаимодействия с базами данных; визуализации данных и НРП. В FDE-SWFs реализованы три схемы выполнения НРП (рис. 1). Все три схемы предполагают применение HPC-ресурсов ГРВС. Масштабирование НРП, СУЗ и метапланировщиков в рамках вычислительной среды достигается за счет контейнеризации прикладного и системного ПО с помощью системы Docker. Запуск, масштабирование и балансировка Docker-контейнеров на ресурсах среды и управление ими осуществляется средствами системы Kubernetes. В рамках данного исследования FDE-SWFs применен для проведения экспериментов по решению задач исследования живучести энергетических инфраструктур (ЭИ) на разных уровнях их территориально-отраслевой иерархии. В сравнении с известными СУРП, использующими традиционные системы хранения данных, FDE-SWFs дополнительно поддерживает обработку данных с помощью технологии In-Memory Data Grid, обеспечивая тем самым существенное сокращение времени решения задач. Это в свою очередь обеспечивает возможность исследования более сложных ЭИ за приемлемое для расчетов время.

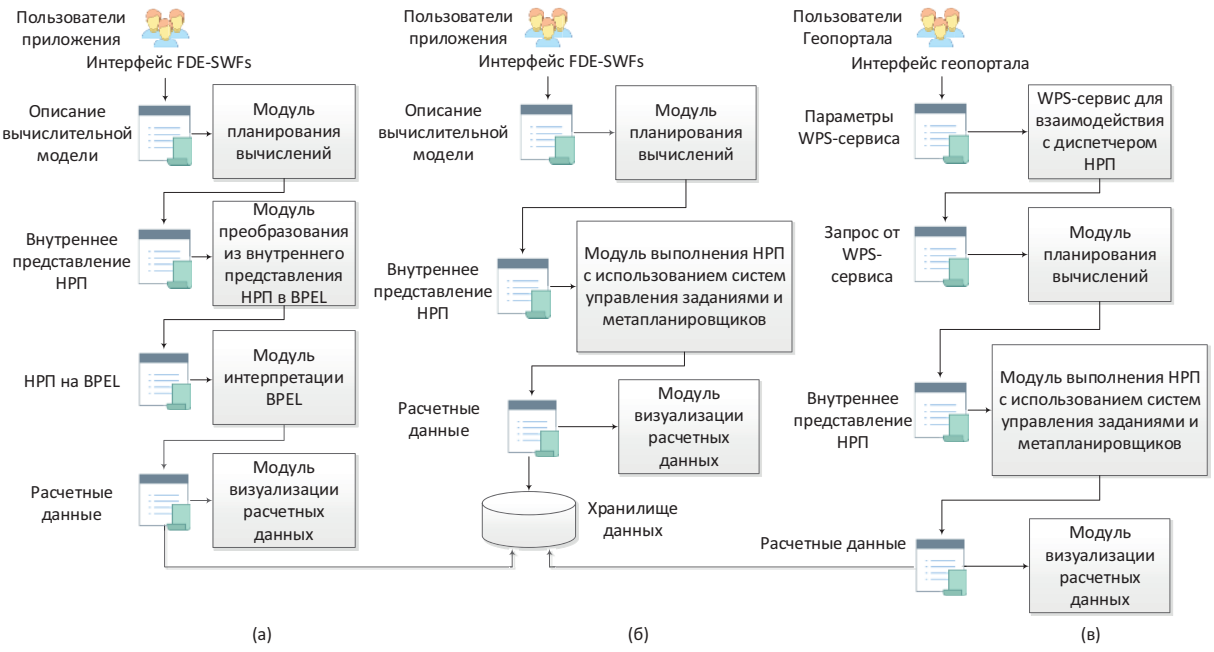


Рис. 1: Схемы выполнения НРП: интерпретация НРП на BPEL (а), выполнение НРП с помощью СУЗ (б), запуск НРП с помощью WPS-сервисов (в)

Другим преимуществом FDE-SWFs по сравнению с подобными инструментами, является возможность построения испытательных стендов разрабатываемых приложений. Такие стенды предназначены для тестирования, анализа работы, структурной и параметрической настройки компонентов вычислительных и управляющих систем среды, а также связанных с ней инженерного и контрольно-измерительного оборудования, программно-аппаратного и телекоммуникационного обеспечения. Они

строятся на основе специальных системных НРП, в которых разработчикам предоставляется доступ к сервисам систем мониторинга среды, специализированным сервисам многокритериального выбора, оценки погрешности вычислений, анализа временных рядов и др. Кроме того, в рамках системных НРП имеется возможность организации проведения многометодных расчетов с целью многокритериального выбора наиболее подходящих методов решения задач с учетом показателей эффективности решения задачи и использования ресурсов.

В настоящее время FDE-SWFs успешно используется при разработке ряда приложений для решения задач анализа производительности и уязвимости ЭИ и исследования эффективности работы алгоритмов структурно-параметрической оптимизации таких инфраструктур.

Список литературы

[1] Ahmad Z. et al. Scientific workflows management and scheduling in cloud computing: taxonomy, prospects, and challenges // IEEE Access. 2021. Vol. 9. P. 53491–53508. https://doi.org/10.1109/ACCESS.2021.3070785

[2] Wratten L., Wilm A., Göke J. Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers // Nature methods. 2021. Vol. 18, no. 10. P. 1161–1168. https://doi.org/10.1038/s41592-021-01254-9

[3] Feoktistov A. et al. An Approach to Implementing High-Performance Computing for Problem Solving in Workflow-based Energy Infrastructure Resilience Studies // Computation. 2023. Vol. 11, no. 12. P. 243. https://doi.org/10.3390/computation11120243

[4] Gorsky S.A. Continuous integration, delivery, and deployment for scientific workflows in Orlando Tools // Proceedings of the 2nd International Workshop on Information, Computation, and Control Systems for Distributed Environments. CEUR-WS Proceedings, 2020. Vol. 2638. P. 118–128. https://doi.org/10.47350/ICCS-DE.2020.11

Разработка новых индикаторов для системы HPC TaskMaster

А.А. Раимова, В.И. Козырев, Р.А. Чулкевич, П.С. Костенецкий

Национальный исследовательский университет
«Высшая школа экономики»

Для отслеживания эффективности выполняемых программ на суперкомпьютере отделом суперкомпьютерного моделирования была разработана система *HPC TaskMaster* [1]. В рамках данной системы, у каждой вычислительной задачи есть агрегированные метрики, описывающие загрузку ресурсов и параметры, характеризующие свойства задачи: длительность, число выделенных CPU и GPU, типы выделенных узлов. Результат анализа вычислительной задачи состоит из трех частей: индикаторы, теги, выводы. Индикаторы свидетельствуют о неверном использовании компонентов суперкомпьютера. Теги описывают характеристики выделенных ресурсов на вычислительную задачу, а также расширяют информацию о свойствах задачи. Выводы строятся на основе индикаторов, тегов и наборах диапазонах параметров [2]. При анализе задачи сравниваются ее индикаторы и теги с теми же сущностями в выводе, а также вхождение параметров задачи в наборы диапазонов параметров у выводов.

Во время анализа графиков загрузки у пользователей были обнаружены следующие типы графиков: на рисунке (a) пример графиков непараллельной задачи, запущенной на нескольких ядрах, на рисунке (b) пример хорошей загрузки ядер у параллельной задачи. Нужен индикатор, который будет обнаруживать задачи с первым типов графиков, то есть непараллельные задачи.

Рассмотрим два подхода к реализации индикатора. Первый, на базе коэффициента корреляции Спирмена, второй на базе меры САП-трансформ. При определении индикатора с использованием коэффициента корреляции Спирмена, обе вычислительных задачи были отмечены индикатором. Это объясняется тем, что значение коэффициента корреляции зависит от показателя стандартного отклонения. Так как в примере (b) отклонение значений загрузки очень мало, то относительно данных временных рядов этой задачи, небольшое отклонение значений в одну десятую может повлиять на значение коэффициента корреляции. Такой подход не рассматривает загрузку задач относительно масштаба всех задач, но при этом имеет сложность $O(n^2)$.

Решением проблемы может быть мера сравнения временных рядов *САП-трансформ* [3]. Для подсчета меры сначала вычисляется значение локального тренда a_i для определенного размера окна $k \in (2, \ldots, n-1)$ для временного ряда y

$$a_i = \frac{6\sum_{j=0}^{k-1}(2j-k+1)y_{i+j}}{hk(k^2-1)}, \quad i(1, \ldots, m),$$

где n – число показателей, h – временной интервал, $m = n - k + 1$.

Далее для двух временных рядов x и y считается мера локальных трендовых ассоциаций:

$$\cos s_k(y,x) = \frac{\sum_{i=1}^{m} a_{yi} \cdot a_{xi}}{\sqrt{\sum_{i=1}^{m} a_{yi}^2 \cdot \sum_{j=1}^{m} a_{xj}^2}}.$$

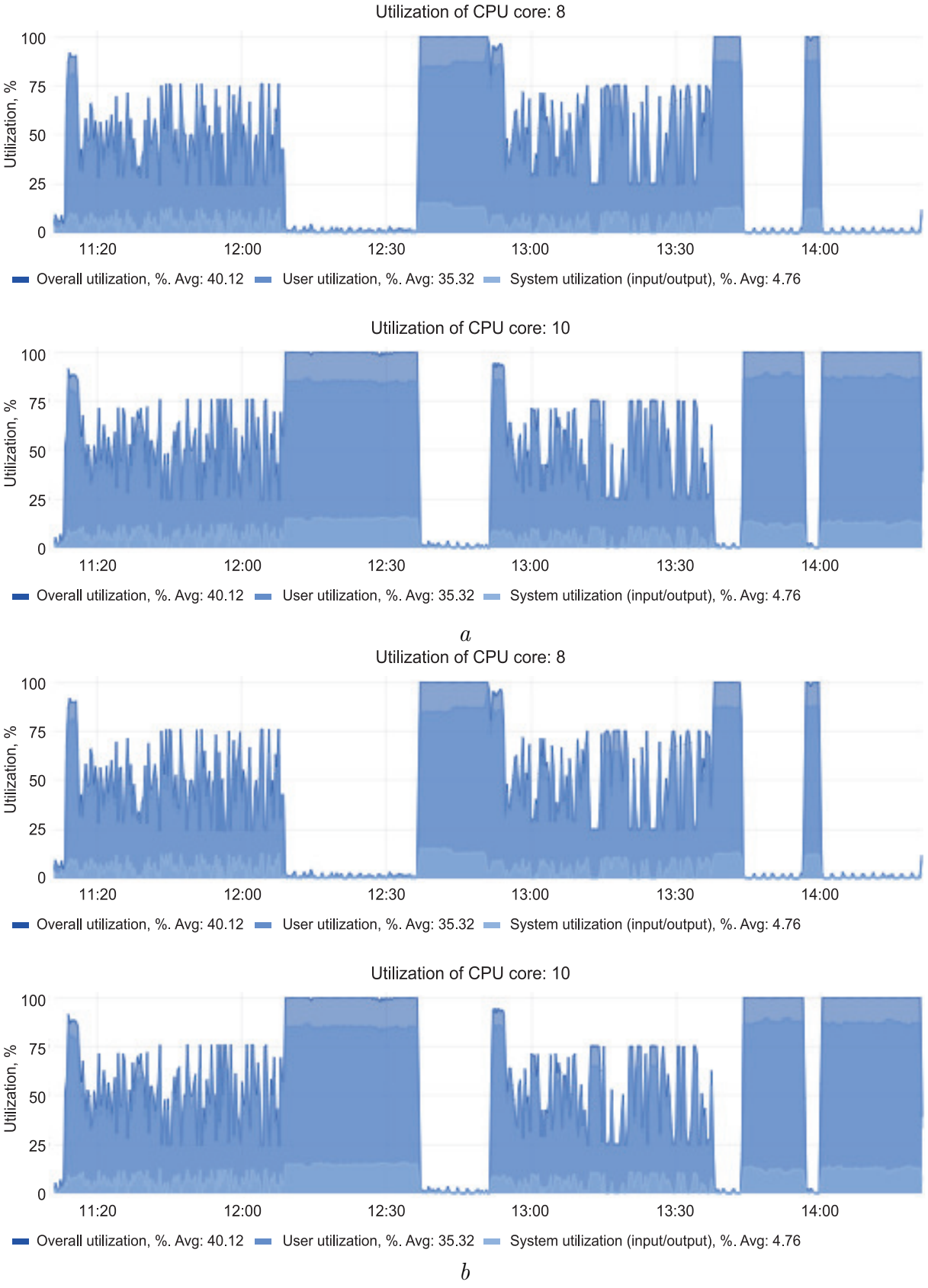


Рис. 1: Примеры графиков загрузки ресурсов у непараллельной (a) и параллельной (b) задачи

Для подсчета значения сравнения двух временных рядов берется среднее по всем размерам окон. Сложность алгоритма сравнения двух временных рядов $O(mn^2+mn)$. Для проверки на наличие индикатора необходимо сравнить каждый график загрузки ядра с графиками всех ядер, выделенных задаче. В системе HPC TaskMaster подсчет метрик и анализ задач производится одновременно для нескольких вычислительных задач, поэтому сложность $O(mn^2 + mn)$ является неоправданно большой для проверки одного индикатора.

В рамках данного исследования был оптимизировать алгоритм САП-трансформ путем подсчета меры только за последние p минут. Значение p на данный момент подсчитывается эмпирически, исходя из того, какое максимальное время алгоритм может потратить на вычисление метрик. Было проведено сравнение между двумя алгоритмами по двум критериям: время выполнения и значение метрики F1-мера. Для эксперимента была собрана выборка из 1502 вычислительных задач пользователей, выполненных на кластере за последние несколько дней. Из таблицы видно, что по F1-мере корреляция Спирмена дает качество намного ниже, чем САП-трансформ, при этом ускоренный и обычный САП-трансформ по качеству почти идентичны. Среднее время подсчета метрик у ускоренного САП-трансформ уменьшилось в 3,5 раза, но все еще на порядок отличается от корреляции Спирмена.

Таблица 1: Результаты исследования алгоритмов

Критерий	Корреляция Спирмена	САП-трансформ	Ускоренный САП-трансформ
F1-мера	0.407186	0.918699	0.91498
Минимальное время	0.00119872	0	0
Максимальное время	0.0196644	369.336	2.95074
Среднее время	0.00234576	1.42191	0.0529625

Индикатор пропуска метрик. Во временных рядах задач иногда могут присутствовать пропуски данных. Такие пропуски возникают по техническим причинам, например, во время профилактических работ и не зависят от пользователя. Для индикатора была написана функция, которая проверяет графики загрузки вычислительных ресурсов на каждом узле и возвращает значение от 0 до 1, где 1 означает, что метрики не были записаны в 100% времени выполнения задачи. Данный индикатор используется для того, чтобы исключить из анализа задачи пользователей, в которых по техническим причинам пропущена часть метрик. Это делается для предотвращения ситуации, когда пользователю ошибочно может быть отправлено предупреждение о запуске неэффективной задачи.

На базе разработанных новых индикаторов система HPC TaskMaster может делать новые выводы, такие как: «Неэффективное использование параллельных вычислений на GPU», «Неэффективное использование параллельных вычислений на CPU». Добавление новых индикаторов и выводов позволяет делать систему HPC TaskMaster более полезной и понятной для пользователей суперкомпьютера и, как следствие, повышать эффективность суперкомпьютерного комплекса.

Список литературы

[1] Kostenetskiy, P.S., Chulkevich, R.A., Kozyrev, V.I., Shamsutdinov, A.B., Antonov, D.A.: HPC TaskMaster - task efficiency monitoring system for the supercomputer center. Communications in Computer and Information Science 1618 (2022). https://doi.org/10.1007/978-3-031-11623-0_2

[2] Kostenetskiy, P.S., Chulkevich, R.A., Kozyrev, V.I., Raimova, A.A.: Enhancement of the Data Analysis Subsystem for the Task Efficiency Monitoring System HPC TaskMaster for the cHARISMa Supercomputer Complex of the HSE University. Communications in Computer and Information Science (2024).

[3] Batyrshin, I., Herrera-Avelar, R., Sheremetov, L., & Suarez, R. Moving approximations in time series data mining. Proc. Int. Conf. Fuzzy Sets and Soft Computing in Economics and Finance FSSCEF, 2004, pp. 62-72.

Содержание

Полные и короткие статьи

Аннотации стендовых докладов

Научное издание

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ
Труды международной конференции
23–24 сентября 2024 г. Москва

Издательство «МАКС Пресс»
Главный редактор: *Е. М. Бугачева*
Компьютерная верстка: *К. Е. Панкратьев*
Обложка: *А. В. Кононова*

Напечатано с готового оригинал-макета

Подписано в печать 05.11.2024 г. Формат 60x90 1/8.
Усл.печ.л. 30,25. Тираж 8 экз. Изд. № 176.

Издательство ООО “МАКС Пресс”. Лицензия ИД N 00510 от 01.12.99 г.
119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В. Ломоносова,
2-й учебный корпус, 527 к. Тел. 8(495) 939-3890/91. Тел./Факс 8(495) 939-3891.

Отпечатано в полном соответствии с качеством
предоставленных материалов в ООО «Фотоэксперт»
109316, г. Москва, Волгоградский проспект, д. 42,
корп. 5, эт. 1, пом. I, ком. 6.3-23Н

9 785317 070717

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ

Труды международной конференции

23–24 сентября 2024 г., Москва

Данный сборник содержит полные статьи на русском языке, короткие статьи и аннотации стендовых докладов, включенных в программу Международной конференции «Суперкомпьютерные дни в России».

Proceedings of the International Conference

September 23–24, 2024, Moscow

RUSSIAN SUPERCOMPUTING DAYS

Proceedings of the International Conference

September 23–24, 2023, Moscow, Russia

МАКС ПРЕСС