Суперкомпьютерный консорциум университетов России
Российская академия наук

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ

Труды международной конференции

26-27 сентября 2022 г., Москва

МАКС ПРЕСС

МОСКВА – 2022

УДК 519.7
ББК 22.18
С89

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

Суперкомпьютерные дни в России : Труды международной конференции. 26–27 С89 сентября 2022 г., Москва / Под. ред. Вл. В. Воеводина. – Москва : МАКС Пресс, 2022. – 166 с.

ISBN 978-5-317-06875-2 e-ISBN 978-5-317-06876-9
https://doi.org/10.29003/m3109.RussianSCDays2022

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

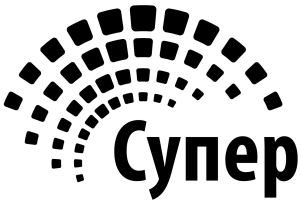
Russian Supercomputing Days: Proceedings of the International Conference. September 26–27, 2022, Moscow, Russia / Ed. by Vl. Voevodin. – Moscow : MAKS Press, 2022. – 166 p.

ISBN 978-5-317-06875-2 e-ISBN 978-5-317-06876-9
https://doi.org/10.29003/m3109.RussianSCDays2022

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 978-5-317-06875-2
e-ISBN 978-5-317-06876-9
https://doi.org/10.29003/m3109.RussianSCDays2022



Суперкомпьютерные дни в России

Полные и короткие статьи

Minimizing the average cost of redistribution when working with two work-stealing deques in two-level memory

Elena Aksenova, Andrew Sokolov

Institute of Applied Mathematical Research, Karelian Scientific Center of the Russian Academy of Sciences, Petrozavodsk

Abstract. In work-stealing parallel task balancers, each core has its own task buffer - deque. The owner of the deque uses one end to add and retrieve tasks, and from the other end, tasks are intercepted by other cores. The paper considers the problem of optimal control of two work-stealing deques in a two-level memory. The probabilities of parallel operations with deques are known. The task is to find the optimal division of fast memory for deques and determine the optimal number of elements for both ends of each deque, which storing in fast memory after memory reallocation. As an optimality criterion, we consider the minimum sum of average memory redistribution costs that occur in the case of overflow or emptying of fast memory by each deque. This criterion makes it possible to take into account specific speeds of access to memory levels and apply the developed methods to different combinations of fast and slow memory. A simulation model of the work process is constructed, the results of numerical experiments are presented.

Keywords: work-stealing balancers, work-stealing deques, data structures, random walks.

1 Introduction

To dynamically balance parallel computing in shared memory, the “work-stealing” strategy is often used, in which empty cores take tasks from other cores [1]. Each core has a deque, which contains pointers to cores, tasks or objects of these tasks. If a new task arises, the core adds a pointer (object) of this task to its deque; when the core needs a task, it reads the pointer (object) of the top of the deque. If the core determines that its deque is empty, it intercepts pointers (objects) from another core. The interception occurs from the bottom of the deque - as in a FIFO-queue, and the inclusion and exclusion operations are performed as in a LIFO-stack. In the work of D. Knuth, such a data structure is called a “deque with restricted input” [2]. One element, or, for example, half of the elements can be intercepted. "Work-Stealing" is used in systems such as Cilk, Cilk++, TPL, TBB, X10, JSR166 (package java.util.concurrent), Erlang, OpenMP in ICC implementation.

In works [3], [4] we proposed models for optimal control of deques in one level memory for sequential cyclic representation of deques. We have proposed a new method for representing deques in the memory of the same level, when they move one after another in the same area of the shared memory (the method is patented [5]). In [6] this method was analyzed for working with two deques, and in [7] - with three. Such a method for FIFO queues was analyzed in [8]. As our experience in software implementation has shown, optimization of work with cache memory in parallel task balancers working according to the Work-Stealing strategy is of great importance. For example, in the implementation [9], due to the fact that task objects instead of task pointers were written to the deques, it was possible to reduce the execution time of test tasks by 2.5 times and cache misses by up to 30% compared to the work-stealing balancers Intel TBB and Intel / MIT Cilk.

Work-stealing balancers will be of great value when used in embedded real-time systems and sensor networks. For example, in [10] there is a reference to [9]. The essence of this link is that the authors of the paper propose new solutions based on the use of embedded computer systems and sensor networks to optimize agricultural management. The use of such information processing systems will help displace the world based on aerial (drones) and ground robots. These tools will perform precise tasks with positioning systems that provide accurate information. To achieve all this, it is necessary to conduct a thorough study of the information systems of agricultural fields using accurate sensors. We also need new data management tools to process them in real time. The correct distribution of these data using various tools (and in the paper [9], according to the authors of the paper, one such tool is given), re-

quires their accurate preliminary processing, which will allow the information to be correctly processed to increase the yield and quality of agricultural fields. The paper [11] provides such a link to the paper [12]. “When distributing jobs in a CNN (convolutional neural network), the task scheduler plays a major role, as it includes task queues, task distribution strategies, and worker threads that actually execute tasks. There are two main approaches of the task scheduler: work sharing, when the scheduler "shares" tasks, and work stealing, when the scheduler "steals" tasks. It is shown [12] that the strategy of “stealing” the task gives a distribution of tasks that is close to optimal, therefore, the work-stealing strategy is used in the distribution of tasks of the CNN.” This confirms the importance of efficient implementation of work-stealing schedulers for optimizing work with neural networks in artificial intelligence systems. Therefore, it is important to explore special methods for working with deques in two-level memory, and not just try to work effectively with general-purpose cache memory implementations.

In the theory of virtual memory and cache memory, it has been proven that when fast memory overflows, it is optimal to move data to slow memory that will not be needed for as long as possible, but in practice this is rarely known. For the most commonly used data structures, you can offer your own methods of working in two-level memory, for example, registers - random access memory, that are different from the universal ones.

In case of stack overflow in fast memory, elements from the bottom of the stack should be transferred to slow memory, since they will not be needed in the near future (work will go with the top of the stack). If the queue overflows, elements from the newer part of the queue should be transferred to slow memory, since they arrived later and access to them will also occur later. In [13], a mathematical model was proposed for the optimal control of a FIFO queue, and in [14] and [15] - for the optimal control of one and two stacks growing towards each other in a two-level memory.

The concept of a stack is widely used in software and hardware development. When overflowing or emptying the register top of the stack, various software and hardware methods were used [16] - [19]. Important new applications are currently emerging that use stacked virtual machines. For example, in [20], a description is given of the organization of the Etherbox32vm virtual machine, designed to implement scenarios for the operation of sensor network nodes.

It should be noted that if many different methods of software and hardware implementation were proposed for stacks, then the methods of hardware implementation for deques are unknown to the authors of the article. This can be explained by the fact that deques began to be used in practice much later than stacks, since they became needed only in connection with the advent of parallel programming. In practice, for deques, you can use analogues of some methods that were used in computers with a stack architecture. For example, a modification of the Barometer-Pointer Algorithm method [17] can be applied, in which elements are moved between memory levels not during overflows or underflows, but when they deviate from half of the fast buffer in the background, i.e. in parallel with normal commands execution. Only movements should be started when there is a deviation from the optimal state that we are looking for.

Works that dealt with the control of deques in two-level memory (registers - OP), before our works, are not known to us.

2 Model for working with deques in two-level memory

Let's assume that we divided fast memory of size m units of memory between deques so that the first deque was given m_1 units of memory, and the second $m_2=m-m_1$ units of memory.

Consider a sequential cyclic method for representing of i-th deque in a fast memory of size m_i, $i=1,2$. Inclusion and exclusion operations are performed at one end of the deque, and steals (exclusion operations) are performed at the other end. The principle of deque management is to store elements from the ends of the deque in fast memory, since they are being worked with, and to store the middle part of the deque in slow memory. With this way of working, the deque can be larger than the size of the fast memory m_i. When the fast memory overflows, some of the elements from the middle of the deque are moved to the slow memory, while elements from the ends of the deque remain in the fast memory. When one of the ends of the deque in the fast memory is empty, some of the elements are moved from the slow to the fast memory, if the slow memory is not empty.

Let x_i denote the first end of the deque, where element inclusions and exclusions occur, and y_i denote the second end of the deque, where steals of elements occur, i=1,2. After redistribution, we get a new deque in fast memory, in which x^0_i elements from the first end, where elements are included and excluded, and y^0_i elements from the second end, where steals occur (see Fig. 1).

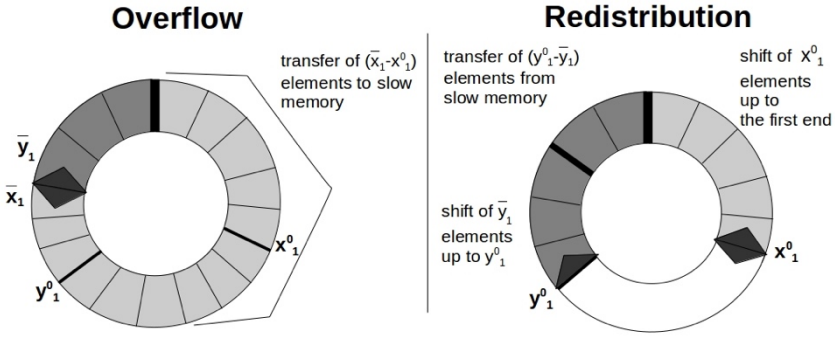


Fig. 1. Overflow and memory redistribution for the first deque.

In case of deque overflow or depletion of one of the ends, the elements are redistributed so that x^0_i and y^0_i elements from the ends of the deque always remain in fast memory. It is necessary to determine such values x^0_i and y^0_i of the elements for each deque, i=1,2, which should be store in fast memory during reallocation, so that the average cost of reallocating memory is minimal.

Let's assume that operations with deque are performed in parallel. Let p_i and q_i be the probabilities of inclusion and exclusion of elements in the deque at the first end; r_i is the probability that the deque length has not changed (it is possible to perform a subtask or an operation with other deques); qr_i - the probability of element steal (exclusion at the second end); qq_i - the probability of simultaneous exclusion from both ends of the deque (exclusion of the element from the first end and steal of the element by another deque from the second end at the same time); qp_i is the probability of simultaneous inclusion of the element and steal of the element by other deque, i=1,2.

The process of working with the first deque after redistribution can be described as a random walk over integer points (x_i, y_i) inside a triangular region $x_i+y_i \le m_i+1$, $x_i \ge -1$, $y_i \ge -1$ (x_i is the number of elements from the end of the deque, where elements are included; y_i is the number of elements from the end of the deque where elements are "stolen"), i=1,2.

The points on the lines $x_i+y_i=m_i+1$, $x_i=-1$ and $y_i=-1$ are absorbing states of the random process (redistribution states). If a random process enters an absorbing state, it means that the memory is redistributed, since hitting the line $x_i+y_i=m_i+1$ corresponds to deque overflow, and hitting the lines $x_i=-1$ or $y_i=-1$ corresponds to depletion of one of the ends of the deque, i=1,2 (see Fig. 2).

Transitions in a random walk will occur according to the scheme:

- from (x_i, y_i) → (x_i+1, y_i) with probability p_i,
- from (x_i, y_i) → (x_i−1, y_i) with probability q_i,
- from (x_i y_i) → (x_i, y_i) with probability r_i,
- from (x_i, y_i) → (x_i, y_i−1) with probability qr_i,
- from (x_i, y_i) → (x_i−1, y_i−1) with probability qq_i ,
- from (x_i, y_i) → (x_i+1, y_i−1) with probability qp_i.

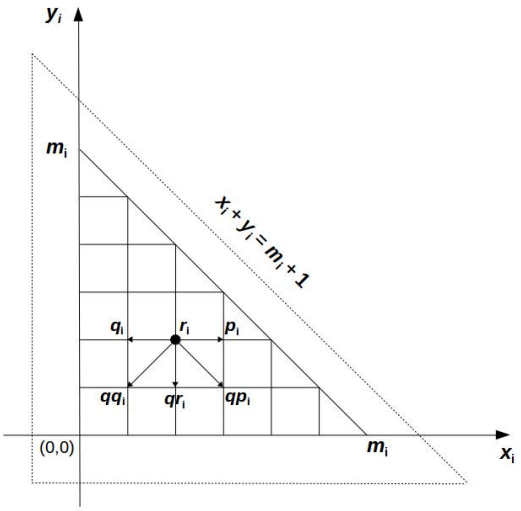


Fig. 2. Random walk area and transition probabilities.

For each initial point (x^0_i, y^0_i) from the area $x_i+y_i \le m_i$, $x_i \ge 0$, $y_i \ge 0$, we will simulate a random walk and calculate the estimates of the absorption probabilities of the process on the lines $x_i+y_i=m_i+1$, $x_i=-1$ and $y_i=-1$ using the simulation model, $i=1,2$. The calculated estimates correspond to the probabilities of overflowing and emptying the deque if x_i and y elements from the ends of the deque remained in the fast memory.

Thus, our task of optimal control of two deques will be the task of determining such values m_i, x^0_i, y^0_i, $i=1,2$, in order to minimize the sum of the average memory reallocation costs that occur when each deque overflows or empties fast memory.

In the problem of managing one sequential FIFO-queue in two-level memory [13], a special case of such a model for the optimality criterion is considered, which minimizes the average number of memory reallocations (maximizes the average time to memory reallocation when the data structure overflows or empties). In this problem, all probabilities, except for p_1 and q_1, are equal to zero, $x=0$. The problem is solved using the apparatus of difference equations [21], which can be applied to the average walk time and solved recursively, since the boundary conditions allow this. In the case of deque management, this method of building a model cannot be applied. Note that if all probabilities are equal to zero, except for p_1 and q_1 , we obtain the problem of managing one stack in a two-level memory [17], where $y=0$.

In [22], a variant of the optimal control problem for a deque in two-level memory is considered, when the optimality criterion is the criterion that minimizes the average number of memory reallocations, i.e., maximizes the average time of working with the deque before reallocating the memory.

In [23], the problem of optimal control of two deques in two-level memory is considered, when criteria that maximize the sum of average times and the minimum average time of working with deques before memory reallocation are used as an optimality criterion.

In [24], the problem of optimal control of n FIFO queues in memory of one level is considered, when the criterion that maximizes the average running time before memory overflow is used as an optimality criterion.

4 Optimality criterion

Let us first consider the optimality criterion in the case of managing the first deque in a two-level memory - the minimum average cost of memory reallocation that occurs in the event of overflow or emptying of fast memory.

To calculate the proposed optimality criterion, it is necessary to know the probabilities of memory reallocation and the time spent on reallocation. To solve the problem, we will build a simulation model, with the help of which we will simulate the process of a random walk.

For each point (x_i, y_i) from the walk area $x_1+y_1 \le m_1$, $x_1 \ge 0$, $y_1 \ge 0$, we will count the number of hits of the random process in each absorbing state (the number of memory reallocations). If we divide the

obtained values by the number of experiments N, then we obtain estimates of the absorption (memory redistribution) probabilities in all absorbing states for each point of the region $x_1+y_1 \le m_1$, $x_1 \ge 0$, $y_1 \ge 0$.

Consider the cost of redistribution. Let t_1 be the time of moving one element in fast memory (for example, registers).

We will assume that $t_1=1$ (clock), $t_2 = kt_1$ is the time of exchanging one element between memory levels, where k currently takes values from 100 to 1000. We will assume that transferring z elements in fast memory takes zt_1 units of time, and transferring z elements from fast memory to slow memory or from slow memory to fast memory takes $zt_2 = zkt_1$ units of time.

Consider the overflow state (x_1,y_1) in Figure 1, where $x_1>x^0_1$ and $0<y_1<y^0_1$. Let $c(x_1,y_1)$ is the reallocation cost in the overflow state (x_1,y_1). To reallocate memory and go to the initial state (x^0_1,y^0_1), you need to move $(x_1-x^0_1)$ elements of the deque to slow memory and shift x^0_1 elements in fast memory to the first end of the deque; move $(y^0_1-y_1)$ elements from slow to fast memory to the second end of the deque, while shifting y_1 elements in fast memory. Then the redistribution cost $c(x1,y1)$ in the state (x_1,y_1) for the initial state (x^0_1, y^0_1) is calculated as follows: $c(x_1,y_1) = (x_1-x^0_1)kt_1 + x^0_1t_1 + (y^0_1- y_1)kt_1 + y_1t_1$. Applying similar reasoning, we can calculate the cost of redistribution in the case of emptying one of the ends of the deque.

To calculate the optimality criterion, we determine the average cost C_1 for memory reallocation. For each initial state (x_1,y_1) in which $x_1+y_1 \le m_1$, $x_1 \ge 0$, $y_1 \ge 0$, we calculate the average time spent for memory redistribution, using the formula $C_1=\sum c_j f_j$, $j=1,\ldots,n$, where f_j is the redistribution probability in the j-th redistribution state (the probability of getting into the j-th absorbing state), c_j is the cost of redistribution in the j-th state of redistribution, n is the number of redistribution states. To calculate the average costs, we will use estimates of the redistribution probabilities calculated using the simulation program. The average cost of memory redistribution for the second deque C_2 is calculated similarly.

To solve the problem, we choose points (x^0_1,y^0_1) and (x^0_2,y^0_2) at which the average costs of reallocating memory C_1 and C_2 are minimal. The minimum sum of average costs for memory redistribution in the case of overflow or depletion of fast memory is considered as optimality criterion for the problem of controlling of two deques.

Let C is the average cost of deque reallocating, when fast memory overflows or empties, and T is the average time of working with a deque before memory redistribution. In a number of works [22] and others, we used *Max T* as an optimality criterion. In the paper [25], we began to use *Min C* as an optimality criterion, but then we realized that this does not take into account the average time of working with the deque before memory redistribution. It is logical that the time of work with the deque before redistribution should be maximized, because in the case of frequent memory redistribution the total cost of reallocation will increase.

Therefore, in [25], we introduced a new optimality criterion *Min C*/*T*, which minimizes the average cost of reallocating a deque when fast memory overflows or empties and at the same time maximizes the average time of working with a deque before reallocating memory, that is, it minimizes the average number of reallocations deque. Let us clarify that $1/T$ is the average loss factor introduced in the theory of cache memory [26], which is equal to the percentage of slow memory accesses to the total number of memory accesses. For example, if on average 1% of accesses are made to slow memory, and 99% to fast memory, then the value $1/T = 0.01$.

The criterion *Min* $(C_1/T_1 + C_2/T_2)$ is used in this work.

4 Numerical experiments

The tables show preliminary results of simulation experiments for some parameters. The optimal point, the division of memory into values m_1 and m_2, the calculated values of the optimality criteria are indicated.

Table 1. m = 10, p_1=0.1, q_1=0.1, r_1=0, qp_1=0.4, qq_1=0.4, qr_1=0, p_2=0.4, q_2=0.4, r_2=0, qp_2=0.1, qq_2=0.1, qr_2=0

Deque 1			Deque 2			
(x^0_1, y^0_1)	Min C_1/T_1	m_1	(x^0_2,y^0_2)	Min C_2/T_2	m_2	Min $(C_1/T_1+C_2/T_2)$
(0, 0)	80.13	1	(0, 0)	32.43	9	112.56
(0, 0)	80.27	9	(0, 0)	41.46	1	121.73
(0, 0)	80.24	2	(0, 0)	32.65	8	112.9
(0, 0)	80.27	8	(0, 0)	36.51	2	116.78
(0, 0)	80.26	3	(0, 0)	32.77	7	113.03
(0, 0)	80.85	7	(0, 0)	35.02	3	115.88
(0, 0)	80.85	4	(0, 0)	33.04	6	113.89
(0, 0)	80.85	6	(0, 0)	34.02	4	114.88
(0, 0)	80.85	5	(0, 0)	33.21	5	114.07

Table 1 shows the results of experiments in the case when the probabilities of parallel operations are more than the probabilities of sequential operations for the first deque, and vice versa for the second deque. Tables 2 and 4 show the results of experiments in the case when sequential operations and parallel operations are equally probable for both deques. Table 4 shows different values of the total memory m.

Table 2. m = 10, p_1=0.25, q_1=0.25, r_1=0, qp_1=0.25, qq_1=0.25, qr_1=0, p_2=0.25, q_2=0.25, r_2=0, qp_2=0.25, qq_2=0.25, qr_2=0

Deque 1			Deque 2			
(x^0_1,y^0_1)	Min C_1/T_1	m_1	(x^0_2,y^0_2)	Min C_2/T_2	m_2	Min $(C_1/T_1+C_2/T_2)$
(1, 0)	59.971	1	(2, 0)	58.136	9	118.11
(2, 0)	58.136	9	(1, 0)	59.971	1	118.11
(0, 0)	58.953	2	(2, 0)	58.132	8	117.08
(0, 0)	58.223	8	(0, 0)	59.271	2	117.49
(0, 0)	58.225	3	(0, 0)	58.223	7	116.45
(0, 0)	58.223	7	(0, 0)	58.225	3	116.45
(0, 0)	58.215	4	(0, 0)	58.223	6	116.44
(0, 0)	58.223	6	(0, 0)	58.215	4	116.44
(0, 0)	58.205	5	(2, 0)	57.832	5	116.04

Table 3 shows the results of experiments when only parallel operations with deques are possible. For the given parameters, the optimal memory redistribution is when all the memory after the is filled with elements that will be stolen. In the future, we plan to build also a mathematical model of random process as an absorbing Markov chain and check simulation experiments in this way.

Table 3. m = 10, p_1=0, q_1=0, r_1=0, qp_1=0.3, qq_1=0.5, qr_1=0.2, p_2=0, q_2=0, r_2=0, qp_2=0.5, qq_2=0.3, qr_2=0.2

Deque 1			Deque 2			
(x^0_1,y^0_1)	Min C_1/T_1	m_1	(x^0_2,y^0_2)	Min C_2/T_2	m_2	Min $(C_1/T_1+C_2/T_2)$
(0, 1)	200.0	1	(0, 9)	123.96	9	323.96
(0, 9)	107.51	9	(0, 1)	200.0	1	307.51
(0, 2)	129.4	2	(0, 8)	125.48	8	254.88
(0, 8)	107.88	8	(0, 2)	150.01	2	257.89
(0, 3)	127.54	3	(0, 7)	128.24	7	255.78
(0, 7)	109.98	7	(1, 2)	143.72	3	253.7
(0, 4)	116.52	4	(0, 6)	129.33	6	245.85
(0, 6)	110.83	6	(0, 4)	135.820	4	246.65
(0, 5)	114.89	5	(0, 5)	133.49	5	245.85

Table 4. p_1=0.25, q_1=0.25, r_1=0, qp_1=0.25, qq_1=0.25, qr_1=0, p_2=0.25, q_2=0.25, r_2=0, qp_2=0.25, qq_2=0.25, qr_2=0

	Deque 1			Deque 2			
m	(x^0_1,y^0_1)	Min C_1/T_1	m_1	(x^0_2,y^0_2)	Min C_2/T_2	m_2	Min $(C_1/T_1+C_2/T_2)$
10	(0, 0)	58.205	5	(2, 0)	57.832	5	116.04
20	(0, 0)	56.65	14	(0, 0)	56.66	6	113.31
30	(0, 0)	56.48	10	(0, 0)	56.48	20	112.96
40	(0, 0)	56.62	32	(0, 0)	56.71	8	113.33
50	(0, 0)	55.57	28	(0, 0)	55.24	22	110.82

Conclusion

In this paper, it was assumed that the probabilistic characteristics of deques are known. In practice, to assess the probabilities, it is necessary to conduct experiments in order to calculate the frequencies of operations with deques in a particular application. If dependency functions probabilities of operations on time are known, then there are several options for finding optimal deque control algorithms. It is possible to divide the program execution time into intervals, where the probabilities of operations can be considered constants, and control the deque at each interval according to the proposed algorithm. You can look for the dependence of the optimal point coordinates of the beginning of the process after the memory redistribution on time.

It would be interesting to study the problem of optimal control of two work-stealing deques in a two-level memory, when the fast memory is divided into two sections, so that one section contains the FIFO-parts of two deques, and the other section contains the LIFO-parts of two deques..

It would also be interesting to add to the problem of optimal control of deques the search for the optimal number of elements for stealing, depending on the model parameters such as the probabilities of operations, temporal characteristics of memory levels, and the size of fast memory. In practice, balancers were implemented that stole half of the deque elements. When working with registers, the implementation of such work scheme will be inefficient apparently, since then in order to “steal” it will be necessary to make exchanges between registers and RAM. Therefore, in the future it is useful to study some analogue of this scheme, when half of the deque elements are stolen from the FIFO-part of the deque, which is located on the registers.

Similar tasks for stacks, FIFO queues, priority queues, and other data structures are also of interest in the future. It is possible to consider different optimality criteria, the choice of which depends on the tasks being solved. If very strict restrictions on the time of solving problems are needed, then, for example, it is possible to minimize not the average, but the maximum costs for reallocating memory.

It is also planned to generalize these tasks to work with *n* data structures. These tasks will be important for optimizing work with 6G networks and with convolutional neural networks in artificial intelligence systems, since the data transfer rate in networks also depends on the optimization of work with various data structures in computers that are network elements.

References

1. Blumofe, R.D., Leiserson, C.E.: Scheduling Multithreaded Computations by Work Stealing. Journal of the ACM 46(5), 720–748 (1999).
2. Knuth D.: The Art of Computer Programming. Volume 1. Addison-Wesley Professional (1997).
3. Sokolov A.V., Barkovsky E.A.: The Mathematical Model and The Problem of Optimal Partitioning of Shared Memory for Work-Stealing Deques. Lecture Notes in Computer Science (9251), 102–106 (2015).
4. Aksenova E.A., Sokolov A.V.: Modeling of the Memory Management Process for Dynamic Work-Stealing Schedulers. In: IEEE Proceedings: Proceedings of 2017 Ivannikov ISPRAS Open Conference (ISPRAS), pp. 12–15. (2018).
5. Barkovsky E. A., Sokolov A. V.: Method for managing computer system memory. RF Patent № 2647627 (2018).
6. Barkovsky E. A., Lazutina A. A., Sokolov A. V.: The optimal control of two work-stealing deques, moving one after another in a shared memory. Program Systems: Theory and Applications, 10(1), 3–17 (2019).
7. Aksenova E.A., Barkovsky E.A., Sokolov A.V.: The Models and Methods of Optimal Control of Three Work-Stealing Deques Located in a Shared Memory. Lobachevskii Journal of Mathematics, 40(11), 1763–1770 (2019).
8. Barkovsky E. A., Sokolov A. V.: Management Model for Two Parallel FIFO Queues Moving One after Another in Shared Memory. Informatsionno-upravliaiushchie sistemy [Information and Control Systems], № 1, pp. 65–73. (2016). (in Russian)
9. Ruslan Kuchumov, Andrey Sokolov, Vladimir Korkhov: Staccato: shared-memory work-stealing task scheduler with cache-aware memory management. International Journal of Web and Grid Services 15(4), 394–407 (2019).
10. Amine Saddik, Rachid Latif,Abdelhafid El Ouardi, Mohamed Elhoseny&Adel Khelifi: Computer development based embedded systems in precision agriculture: tools and application. Acta Agriculturae Scandinavica 72(12), (2022).
11. Khaydarova R., , Mouromtsev D., Lapaev M., Fishchenko V.: Distributed convolutional neural network model on resource-constrained cluster. Scientific and Technical Journal of Information Technologies, Mechanics and Optics, vol. 20 5(129), 739–746 (2020). (in Russian)
12. Kuchumov R.I.: Implementation and analysis of the work-stealing task scheduler. Stochastic optimization in computer science 12(1), 20-39 (2016). (in Russian)
13. Sokolov A.V.: On optimal caching of FIFO queues. Stochastic optimization in computer science 9(2), 72–88 (2013). (in Russian)
14. Aksenova E.A., Lazutina A.A., Sokolov A.V.: Study of a Non-Markovian Stack Management Model in a Two-Level Memory. Programming and Computer Software 30(1), 25–33 (2004).
15. Aksenova E. A., Sokolov A. V.: Optimal control of two parallel stacks in two-level memory. Discrete Math. (19), 67–75 (2007).
16. Hasegava M., Shigei Y.: High-speed top-of-stack scheme for VLSI processor: a management algorithm and its analysis. In: Proceedings of 12th Symposium on Computer Architecture, pp. 48–54. (1985).

17. Stanley T., Wedig R.: A performance analysis of automatically managed top of stack buffers. In: Proceeding of 14th Symposuim on Computer Architecture, pp. 272-281. (1987).
18. Koopman P. J.: Stack Computers: The New Wave. Ellis Horwood Ltd. (1989).
19. Kim A. K., Perekatov V. I., Yermakov S. G.: Microprocessors and computing systems of the Elbrus family, Piter, SPb. (2013). (in Russian)
20. Yury Shevchuk, Andrey Shevchuk: Etherbox32vm virtual machine. Program systems: Theory and applications, 7:4(31), 119–143 (2016). (in Russian)
21. Feller W.: An Introduction to Probability Theory and its Applications. V. I, 3rd Edition, Wiley, (1984).
22. Lazutina A. A., Sokolov A. V.: On the optimal management of Work-Stealing deques in two-levelmemory. Vestn. Komp'yut. Inform. Tekhnol. 17(4), 51–60 (2020).
23. Aksenova, E.A., Lazutina, A.A., Sokolov, A.V. About Optimal Management of Work-Stealing Deques in Two-Level Memory. Lobachevskii Journal of Mathematics (42), 1475–1482, (2021).
24. Aksenova, E.A., Sokolov, A.V. Optimal Parallel Control of n FIFO-Queues in Shared Memory. Lobachevskii Journal of Mathematics (42), 44–49 (2021).
25. Elena A. Aksenova, Anna A. Lazutina, Andrew V. Sokolov: About optimal managment of work-stealing deques in two-level memory. Program Systems: Theory and Applications 12:2(49), 53–71, (2021). (in Russian)
26. Kohonen T.: Content-Adressable Memories. Springer Series in Information Sciences (1). Springer-Verlag, Berlin–Heidelberg (1980).

Quantum–Chemical Research of Some Imidazole Tetrazine Derivatives *

V.M. Volokhov[1], E.S. Amosova[1], V.V. Parakhin[2], A.V. Volokhov[1], D.B. Lempert[1], T.S. Zyubina[1]

[1]Federal Research Center of Problems of Chemical Physics and Medicinal Chemistry, Chernogolovka, Russian Federation,
[2]N.D. Zelinsky Institute of Organic Chemistry, Moscow, Russian Federation

The paper refers to the results of quantum–chemical calculations of structural parameters, enthalpy of formation, IR absorption spectra of some 5/6/5 tricyclic 1,2,3,4- and 1,2,4,5-tetrazines annelated with imidazoles. Methods of various complexity of the Gaussian program package, G4, G4MP2, M062X/6-311+G(2d,p), B3LYP/6-311+G(2d,p), ωB97XD/aug-cc-pVTZ, CBS-4M, have been used in calculations.

Keywords: quantum–chemical calculations, high–energy density materials, tetrazine derivatives, enthalpy of formation.

1. Введение

New high–energy density materials (HEDMs) are in high demand in the field of energy-intensive compounds [1–9]. And that makes the study of the existing ones and the search for the new ones a very important task. Most researches present derivatives of various nitrogenous heterocyclic nuclei as very promising new HEDMs. For example, several of the first representatives of a series of high-energy 5/6/5 tricyclic derivatives of 1,2,3,4–tetrazines have been recently proposed in [10–12] and reported to possess a number of attractive energetic and physicochemical properties. The presence of nitro groups in their structure provides an acceptable oxygen balance and a large number of N–N, N=N and C–N bonds of the heterocyclic system sets a high level of enthalpy of formation (ΔH_f), thus defining high energy potential of these compounds.

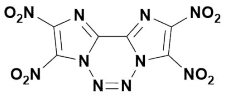
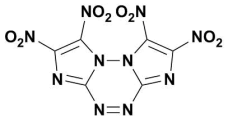
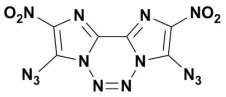
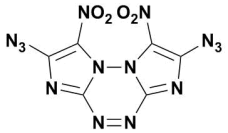
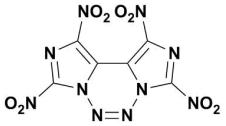
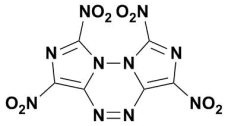
It should be emphasized that ΔH_f is the key parameter that determines the energy capabilities of the HEDM, and the assessment of the compound's energy characteristics highly depends on the accuracy of its value. Therefore, this work focused on calculation of the enthalpy of formation for 5/6/5 tricyclic tetrazine derivatives annelated with imidazoles (Table 1) in the gas phase at a temperature 298 K and pressure $p = 1$ atm. Quantum–chemical methods of varying complexity have been used to determine the enthalpy of formation and to detect regularities in the dependence of the value of $\Delta H^0_{f(g)}$ on the structure of the compound. Current work continues our research of 5/6/5 tricyclic derivatives 1,2,3,4–tetrazines annelated with different kinds of azoles [13].

2. Calculation Method

Experimental measurement of $\Delta H^0_{f(g)}$, requires time–consuming synthetic production and labour–intensive thermochemical research of compounds under investigation which can be

*The equipment of the Center for Collective Use of Super High–Performance Computing Resources of the Lomonosov Moscow State University [14, 15] (projects Enthalpy-2219 and Enthalpy-2065) and computational resources of IPCP RAS have been used for calculations. Volokhov V.M, Amosova E.S. and Volokhov A.V. conducted quantum–chemical research in accordance with the State task, state registration No. AAAA-A19-119120690042-9. Lempert D.B. assessed the energy potential in accordance with the State task, state registration No. AAAA-A19-119101690058-9. Zyubina T.S. performed calculation and analysis of the IR spectra and atom displacements in accordance with the State task, state registration No. AAAA-A19-119061890019-5. Calculations by resource–intensive methods were carried out with the support of the RFBR, project No. 20-07-00319.

Таблица 1. Molecules under consideration

Formula	α^a	N%^{b}	№	Structural formula	№	Structural formula
$C_6N_{10}O_8$	0.666	41.18	**1a**		**1b**	
$C_6N_{14}O_4$	0.333	59.04	**2a**		**2b**	
$C_6N_{10}O_8$	0.666	41.18	**3a**		**3b**	

[a] - $\alpha = 2z/(4x + y)$, the coefficient of oxygen saturation of $C_xH_yN_wO_z$ molecule;

[b] - nitrogen mass content.

disadvantageous for the research. At the same time quantum–chemical ab initio calculations provide means to determine the most accurate values of $\Delta H^0_{f(g)}$. Curtiss and coauthors [16] use a test set of 454 substances to analyze in detail the veracity of calculations of thermochemical quantities by quantum-chemical methods based on the G4 method of the Gaussian package. It is shown in their work that the average deviation of the calculated values from experimental data is less than 1 kcal/mol (for high–enthalpy substances it is less than 1%). Quantum–chemical methods enable the design of molecules of new compounds that do not exist yet, but seem promising for synthesis. They also help to determine with high accuracy physicochemical (especially thermochemical) properties of new compounds and analyze in detail how they depend on the structural parameters of the molecule.

Enthalpy of formation of $C_xN_yO_z$ molecule has been calculated in this work by two approaches:

1) $\Delta H^0_{f(g)}(I)$ – using ΔH^0 of the atomization reaction for the compound under consideration

$$xC_{(g)} + yN_{(g)} + zO_{(g)} = C_xN_yO_{z(g)};$$

2) $\Delta H^0_{f(g)}(II)$ – using ΔH^0 of the reaction of formation of the compound under consideration from simple substances

$$xC_{(g)} + (y/2)N_{(g)} + (z/2)O_{(g)} = C_xN_yO_{z(g)}.$$

3. Results and Discussion

3.1. Enthalpy of Formation

Figure 1 shows the essential geometric parameters of molecules under consideration determined at the ωB97XD/aug-cc-pVTZ level. Table 2 presents values of the enthalpy of formation of the studied molecules in the gas phase calculated by different (B3LYP/6-311+G(2d,p), M062X/6-311+G(2d,p), G4MP2, ωB97XD/aug-cc-pVTZ, G4, CBS-4M). Figures 2, 3 and 4 show IR absorption spectra and atom displacements for the most intense vibrations calculated at the ωB97XD/aug-cc-pVTZ level.

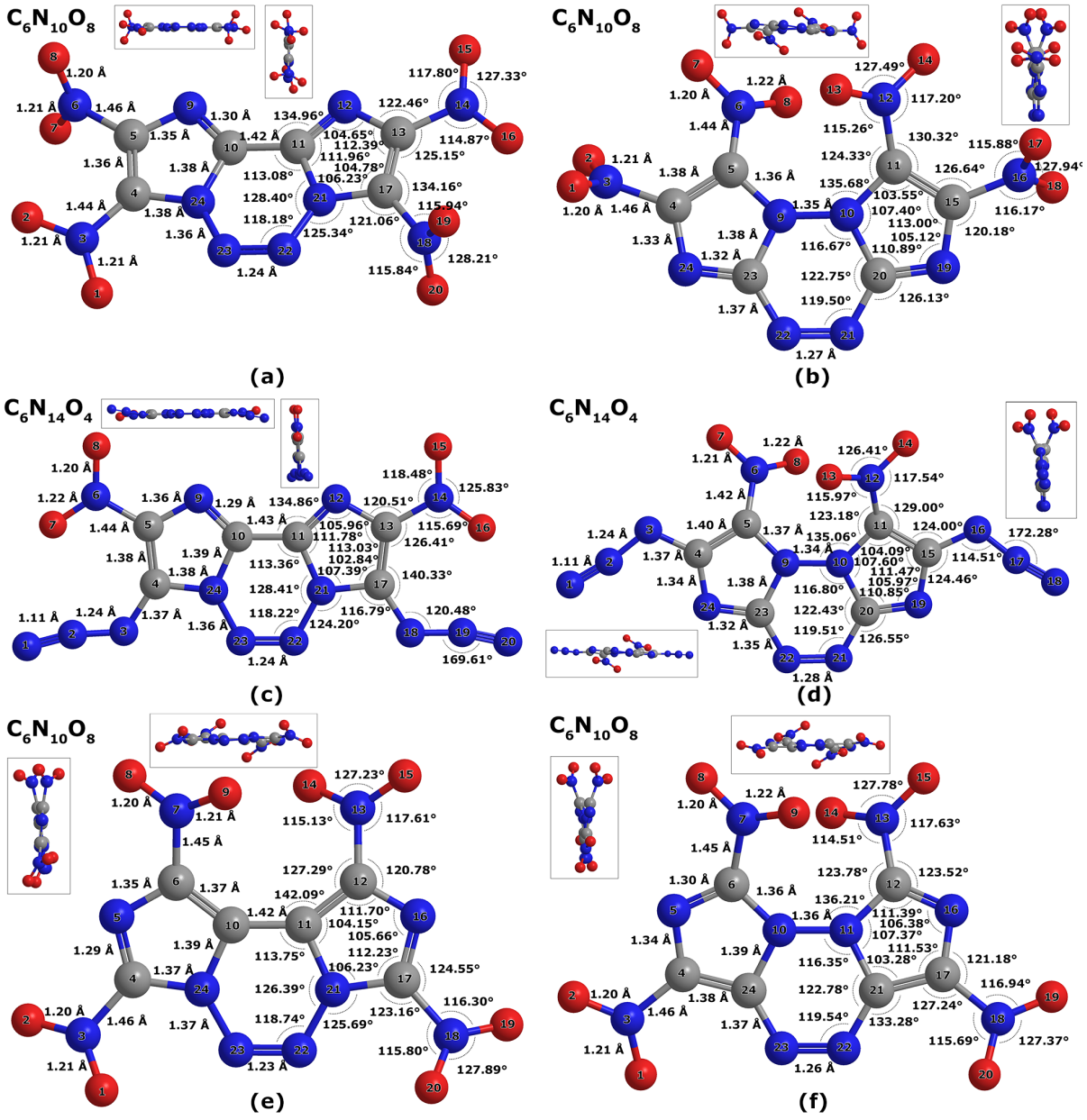


Рис. 1. Structures (in different angles) and the essential geometric parameters (in Å and °) of the molecules under consideration (calculated at the ωB97XD/aug-cc-pVTZ level): **a** — 1a, **b** — 1b, **c** — 2a, **d** — 2b, **e** — 3a, **f** — 3b.

Таблица 2. Enthalpy of formation of the considered molecules in the gas phase calculated at various levels (in kJ/mol).

Calculation level	$C_6N_{10}O_8$		$C_6N_{14}O_4$		$C_6N_{10}O_8$	
	1a	**1b**	**2a**	**2b**	**3a**	**3b**
B3LYP/6-311+G(2d,p)	924.55[a]	967.31[a]	1442.41[a]	1478.33[a]	922.74[a]	962.75[a]
	921.65[b]	964.41[b]	1415.34[b]	1451.27[b]	919.84[b]	959.85[b]
M062X/6-311+G(2d,p)	952.30[a]	986.38[a]	1498.04[a]	1537.07[a]	942.98[a]	979.59[a]
	856.11[b]	890.19[b]	1373.30[b]	1412.33[b]	846.79[b]	883.40[b]
G4MP2	813.90[a]	854.87[a]	1394.94[a]	1430.36[a]	809.65[a]	852.83[a]
	794.73[b]	835.70[b]	1390.55[b]	1425.96[b]	790.48[b]	833.66[b]
ωB97XD/aug-cc-pVTZ	811.48[a]	855.26[a]	1375.62[a]	1417.95[a]	807.82[a]	852.43[a]
	790.04[b]	833.82[b]	1294.27[b]	1336.59[b]	786.39[b]	830.99[b]
G4	**780.56**[a]	**818.83**[a]	**1315.56**[a]	**1395.85**[a]	**777.36**[a]	**818.89**[a]
	779.89[b]	**817.86**[b]	**1321.22**[b]	**1401.51**[b]	**776.40**[b]	**817.92**[b]
CBS-4M	758.90[a]	800.47[a]	1381.72[a]	1402.28[a]	752.46[a]	793.96[a]
	648.03[b]	689.60[b]	1243.70[b]	1264.26[b]	641.60[b]	683.09[b]

[a] $\Delta H^0_{f(g)}(I)$

[b] $\Delta H^0_{f(g)}(II)$

The smallest difference between values of the enthalpy of formation determined by approaches I and II has been observed at the G4 level. Calculations by other methods resulted in differences of 4–19 kJ/mol (0–2%), 21-81 kJ/mol (3–6%), 3–27 kJ/mol (0–2%), 96–125 kJ/mol (9–11%), 111–138 kJ/mol (11–16%) at the G4MP2, ωB97XD/aug-cc-pVTZ, B3LYP/6-311+G(2d,p), M062X/6-311+G(2d,p), CBS-4M levels accordingly. In these calculations values of $\Delta H^0_{f(g)}(I)$ in most cases are higher than those of $\Delta H^0_{f(g)}(II)$ (Table 2). The only exception is calculation by G4 method for structures **2a** and **2b**.

Comparison of values of $\Delta H^0_{f(g)}(II)$ calculated by different methods has shown that results of calculations at ωB97XD/aug-cc-pVTZ, G4MP2, M062X/6-311+G(2d,p, B3LYP/6-311+G(2d,p) and CBS-4M level diverge from the most accurate results of the G4 method by 10–65 kJ/mol (1–5%), 15–70 kJ/mol (2–5%), 11–76 kJ/mol (1–10%), 50–147 kJ/mol (4–18%) and 78–137 kJ/mol (6–17%) accordingly. Thus it might be observed that methods ωB97XD/aug-cc-pVTZ and G4MP2 provide results closest in accuracy to those obtained by G4 method. However, calculations by G4 method require more time than those by ωB97XD/aug-cc-pVTZ and G4MP2 methods: 123–129 days for the former one as opposed to 4–5 and 7–19 days for the latter ones accordingly. When such level of accuracy is acceptable, calculations by ωB97XD/aug-cc-pVTZ and G4MP2 methods could be recommended for use.

Table 2 shows that values of enthalpy of formation increase by 45 and 35 kJ/mol along the series **3a — 1a — 3b — 1b** and **2a — 2b**, accordingly. Though the enthalpy of formation changes significantly (by 504 kJ/mol) between the two series. It might be noted that ceteris paribus enthalpy of formation slightly increases when the chain of nitrogen atoms $N-N-N-N$ is being divided into two pairs of $N-N$ (transfer from structure **3a** to **3b**), when NO_2 groups are moved wider apart (transfer from structure **3a** to **1a**), when number of $N-C-N$ fragments in the rings increases (transfer from structure **3b** to **1b**) and they are located closer to each other (transfer from structure **3a** to **1a**). At the same time replacement of the NO_2 group by the N_3 group (structure **1a** to **2a** and **1b** to **2b**) leads to a noticeable increase (by 541 and 584 kJ/mol accordingly) of the enthalpy of formation.

Figure 1 shows that every pair of structures (**1a** and **1b**, **2a** and **2b**, **3a** and **3b**) differs by the type of central tetrazine ring. And in each case (as can be seen from Table 2) it's those based on 1,2,4,5-tetrazine that have higher enthalpy of formations as compared to the ones based on 1,2,3,4-tetrazine.

3.2. The IR spectra

Figures 2, 3 and 4 show the IR spectra of absorption and atom displacements for the most notable vibrations of the molecules under study calculated at the ωB97XD/aug-cc-pVTZ level. The scale factor 0.956 for the spectra that has been recommended in [17] has been used for better agreement with the experiment.

Structures **3a**, **1a**, **3b**, **1b** are characterized by the most intense vibrations in the range of 1602–1627 cm^{-1}, which can be accredited to the stretching vibrations of $N_{NO_2}-O$ bonds in NO_2 groups, and by vibrations in the range of 805–873 cm^{-1} which can be referred to the angular vibrations in NO_2 groups. Structures **2a** and **2b** are characterized by the most intense vibrations in the range of 2237–2252 cm^{-1}, they can be associated with the stretching vibrations of $N_{N_3}-N_{N_3}$ bonds in N_3 groups.

4. Computational Details

The computational node Lomonosov-2 with the following configuration: Intel(R) Xeon(R) Gold 6140 CPU @ 2.30 GHz, RAM 259 GB, 20 TB disk space has been used for calculations.

It's worth mentioning that the Gaussian package uses its own Linda software for parallelization purposes. In order to assess the efficiency of calculations we have performed some test jobs on computational nodes with different configuration. While core number increased up to

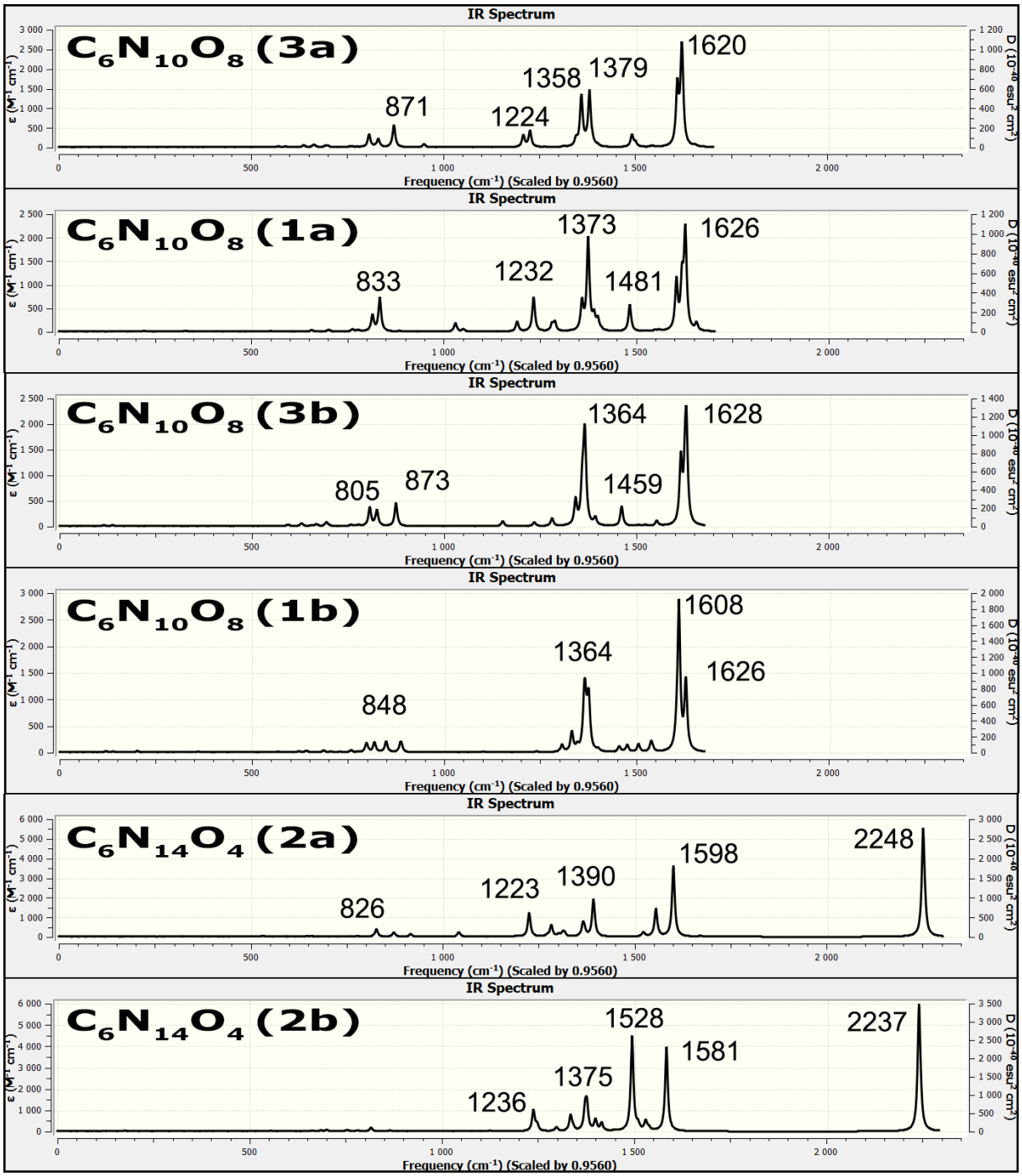


Рис. 2. The IR absorption spectra of the molecules under consideration

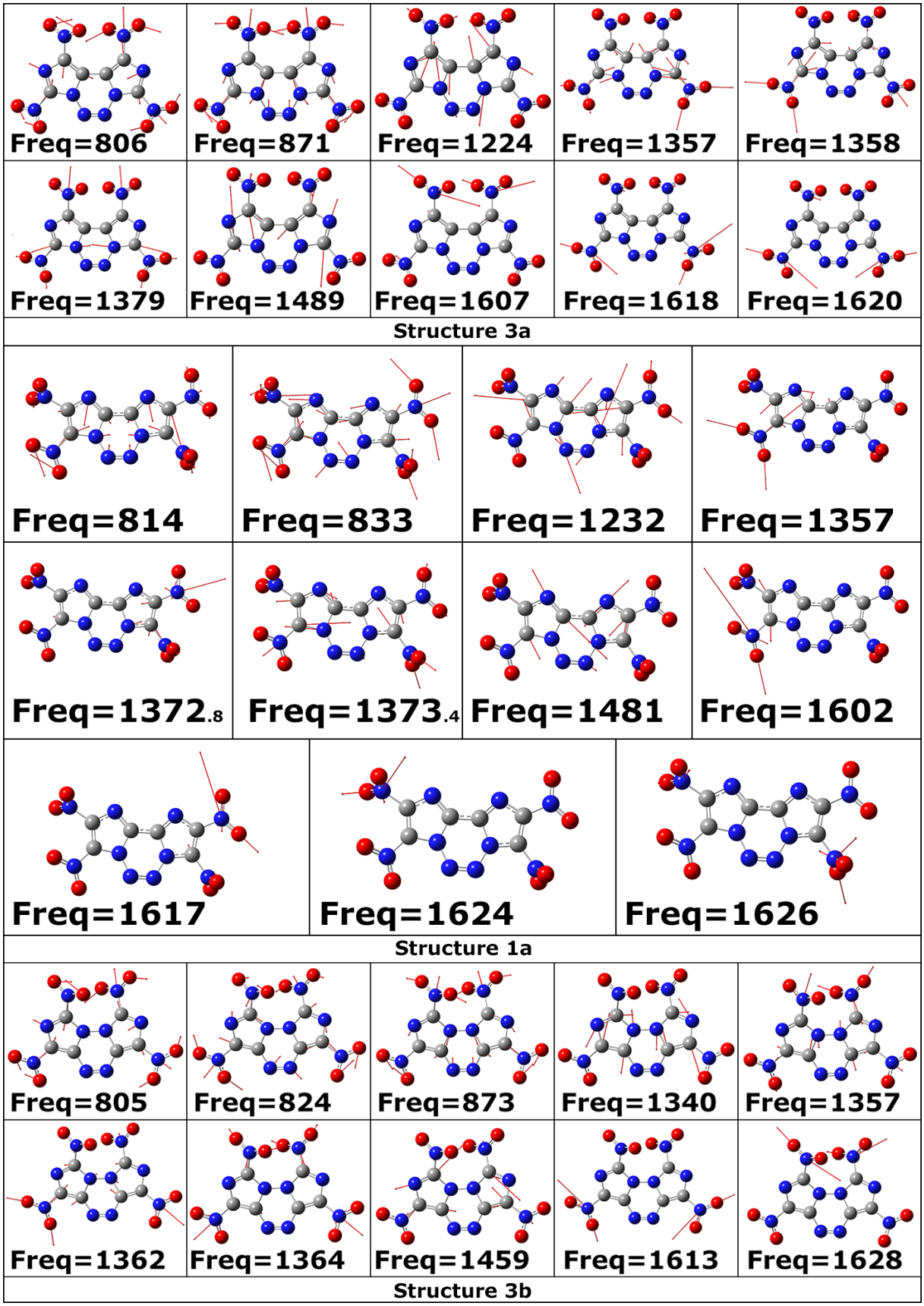


Рис. 3. Atom displacements for the most notable vibrations (structures **3a**, **1a** and **3b**)

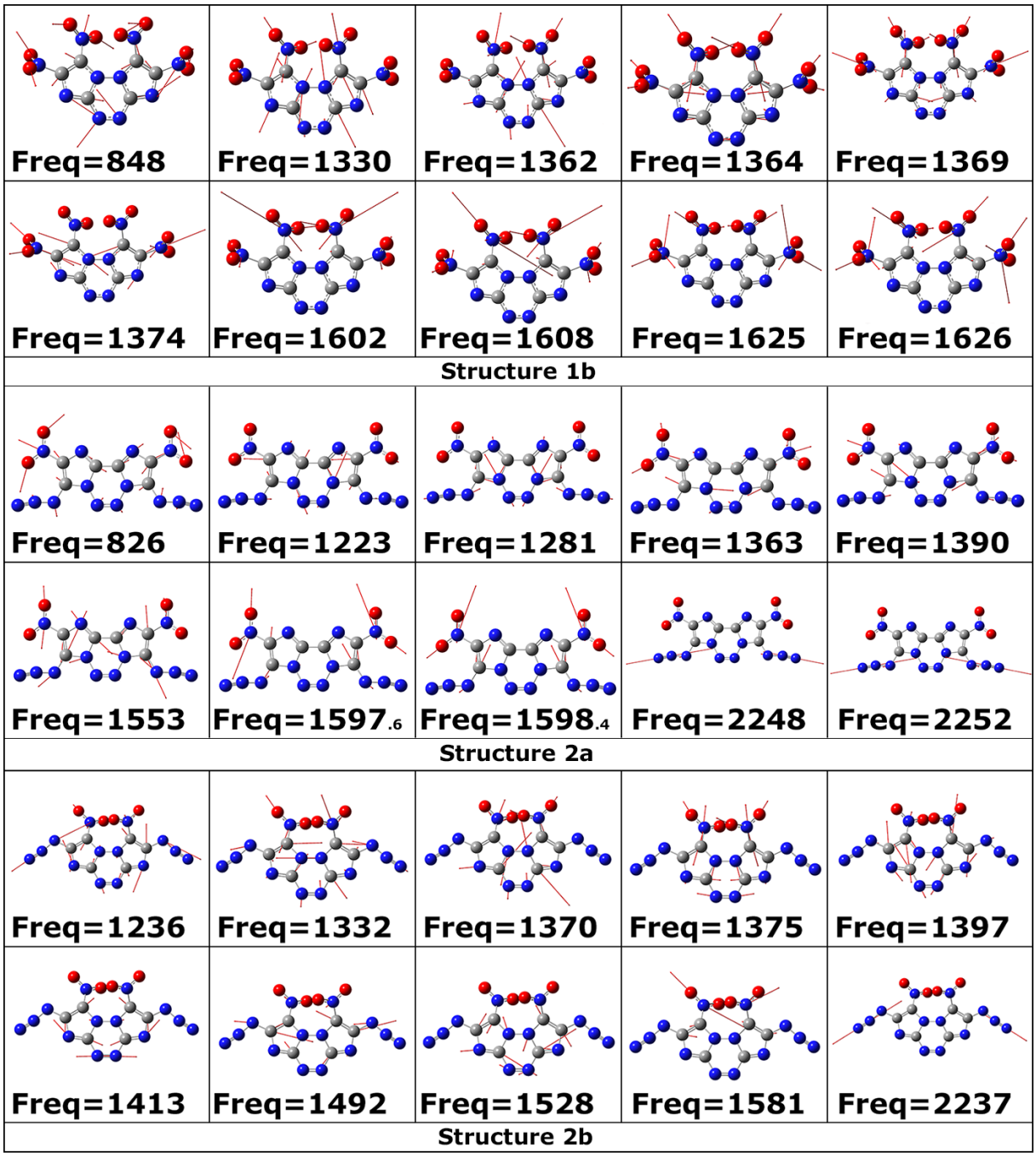


Рис. 4. Atom displacements for the most notable vibrations (structures **1b**, **2a** and **2b**)

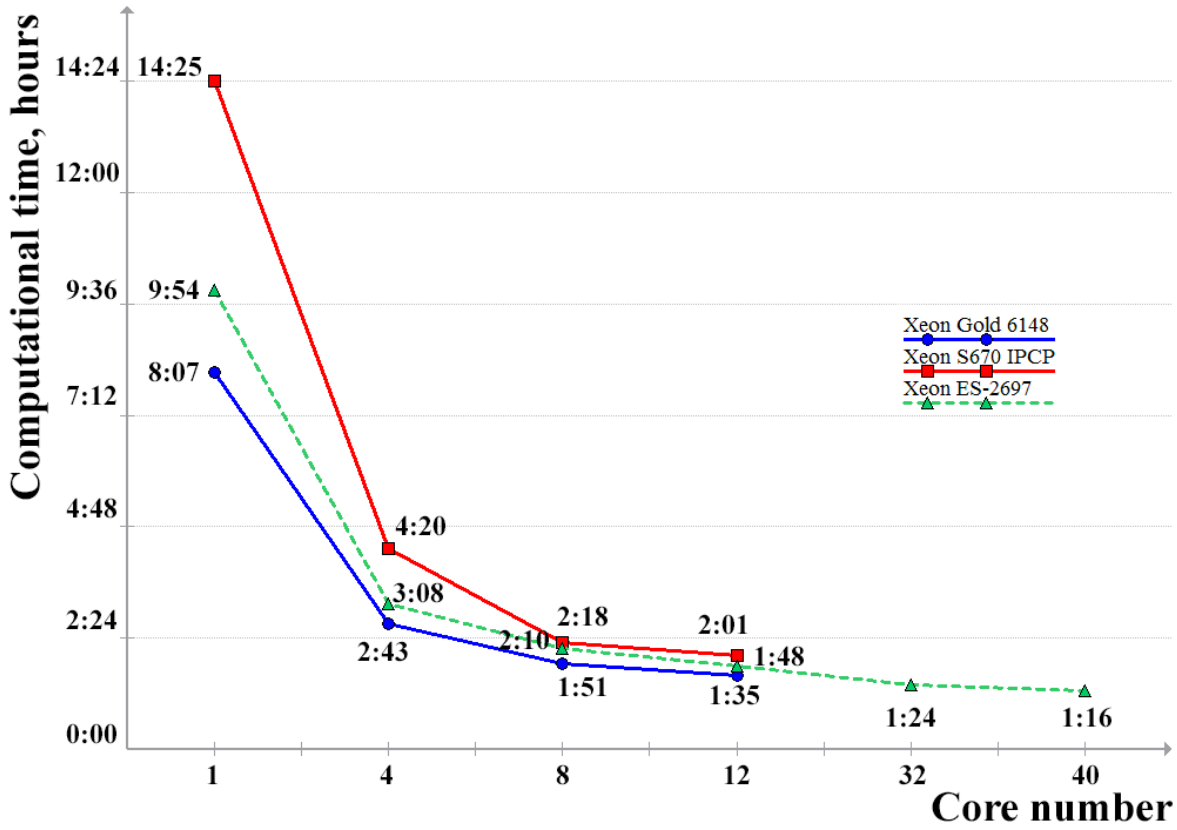


Рис. 5. Comparison of the computational time for the standard task in the Gaussian package on various computational configurations

8 cores, a stable accelaration of computational time has been observed (Fig. 5), but with further increase of core number this effect of accelaration decreased, which was the reason why we used 8 cores per task in our calculations. It's also noteworthy that the speed of calculations depends on the processors' support of the avx2 and sse42 instructions. The former one, in particular, when used on processors with a close clock rate, can profit by 8-10 times [18]. The Gaussian package produces enormous intermediate files up to 2 TB in the course of calculations. Recording them on an SSD disk can take up to 35–50 min, and noticeable longer in the case of SATA arrays. Therefore it is preferable to use SSD disks or high-speed SAS disks with a large amount of allocated disk memory for such calculations.

Calculation time by given conditions for structures under consideration varied from 10 core–hours (1 hour) for CBS-4M method to 23000 core–hours (123 days) for the most resource–expensive method G4.

5. Conclusions

Quantum–chemical calculations of the enthalpy of formation, structure and IR absorption spectra of the series of tricyclic tetrazines annelated with imidazoles have been performed for the first time. Various computational methods within the Gaussian program package have been compared. It has been shown that the values of the enthalpy of formation of the considered molecules in the gas phase lie within the range of 790–1426 kJ/mol. At the same time replacement of the NO_2 group by the N_3 group in the structures under consideration leads to a noticeable increase (more than 500 kJ/mol) of the enthalpy of formation. Structures based on 1,2,4,5-tetrazine are characterized by higher enthalpy of formation than those based on 1,2,3,4-tetrazine.

References

1. Yin P., Zhang Q., Shreeve J.M. Dancing with Energetic Nitrogen Atoms: Versatile N-Functionalization Strategies for N-Heterocyclic Frameworks in High Energy Density Materials // Acc. Chem. Res. 2016. Vol. 49. No. 1. P. 4–16. DOI: 10.1021/acs.accounts.5b00477.

2. Qu Y., Babailov S.P. Azo-linked high-nitrogen energetic materials // J. Mater. Chem. A. 2018. Vol. 6. P. 1915–1940. DOI: 10.1039/C7TA09593G.

3. Yongjin C., Shuhong B. High Energy Density Material (HEDM) - Progress in Research Azine Energetic Compounds // Johnson Matthey Technol. Rev. 2019. Vol. 63. P. 51–72. DOI: 10.1595/205651319x15421043166627.

4. Fershtat L L., Makhova N.N. 1,2,5-Oxadiazole-Based High-Energy-Density Materials: Synthesis and Performance // ChemPlusChem. 2020. Vol. 85. No. 1. P. 12–42. DOI: 10.1002/cplu.201900542.

5. Gao H., Zhang Q., Shreeve J.M. Fused heterocycle-based energetic materials (2012–2019) // J. Mater. Chem. A. 2020. Vol. 8. P. 4193–4216. DOI: 10.1039/C9TA12704F.

6. Zhang, J., Zhou, J., Bi, F., Wang, B.: Energetic materials based on poly furazan and furoxan structures. Chin. Chem. Lett. 31(9), 2375–2394 (2020). DOI: 10.1016/j.cclet.2020.01.026.

7. Zhou, J., Zhang, J. L., Wang, B. Z., Qiu, L. L., Xu, R. Q., Sheremetev, A. B.: Recent Synthetic Efforts towards High Energy Density Materials: How to Design High-Performance Energetic Structures? FirePhysChem. (2021). DOI: 10.1016/j.fpc.2021.09.005.

8. Tang, J., Yang, H., Cui, Y., Cheng, G.: Nitrogen-rich tricyclic-based energetic materials. Mater. Chem. Front. 5, 7108–7118 (2021). DOI: 10.1039/D1QM00916H.

9. Churakov, A. M., Voronin, A. A., Klenov, M. S., Tartakovsky, V. A.: Other Tetrazines and Pentazines. In Comprehensive Heterocyclic Chemistry IV, Black, D. S., Cossy, J., Stevens, C. V. (eds.) Elsevier Science, Amsterdam, 9, 640–661 (2022). DOI: 10.1016/B978-0-12-818655-8.00064-0.

10. Chavez, D. E., Bottaro, J. C., Petrie, M., Parrish, D. A.: Synthesis and Thermal Behavior of a Fused, Tricyclic 1,2,3,4-Tetrazine Ring System. Angew. Chem. Int. Ed. 54, 12973–12975 (2015). DOI: 10.1002/anie.201506744.

11. Tang, Y., Kumar, D., Shreeve, J. M.: Balancing Excellent Performance and High Thermal Stability in a Dinitropyrazole Fused 1,2,3,4-Tetrazine. J. Am. Chem. Soc. 139, 13684–13687 (2017). DOI: 10.1021/jacs.7b08789.

12. Tang, Y. X., He, C. L., Yin, P., Imler, G. H., Parrish, D. A., Shreeve, J. M.: Energetic Functionalized Azido/Nitro Imidazole Fused 1,2,3,4-Tetrazine. Eur. J. Org. Chem. 19, 2273–2276 (2018). DOI: 10.1002/ejoc.201800347.

13. Volokhov V.M., Amosova E.S., Volokhov A.V., Zyubina T.S., Lempert D.B., Parakhin V.V. 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 // Comput. Theor. Chem. 2022. Vol. 1209. 113608. DOI: 10.1016/j.comptc.2022.113608.

14. Voevodin V.V., Antonov A.S., Nikitenko D.A., Shvets P.A., Sobolev S.I., Sidorov I.Y., Stefanov K.S., Voevodin V.V., Zhumatiy S. A. Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community // Supercomput. Front. Innov. 2019. Vol. 6, P. 4–11. DOI: 10.14529/jsfi190201.

15. Nikitenko D., Voevodin V., Zhumatiy S. Deep Analysis of Job State Statistics on Lomonosov-2 Supercomputer // Supercomput. Front. Innov. 2019. Vol. 5. P. 4–10. DOI: 10.14529/jsfi180201.

16. Curtiss, L. A., Redfern, P. C., Raghavachari, K.: Gaussian-4 theory. J. Chem. Phys. 126, 084108 (2007). DOI: 10.1063/1.2436888.

17. CCCBDB Vibrational Frequency Scaling Factors URL: https://cccbdb.nist.gov/vsfx.asp(accessed: 04.04.2022).

18. Grigorenko B., Mironov V., Polyakov I., Nemukhin A. Benchmarking quantum chemistry methods in calculations of electronic excitations // Supercomput. Front. Innov. 2019. Vol. 5. No. 4. P. 62-66. DOI: 10.14529/jsfi180405.

XY model on self-avoiding walks : a large-scale Monte Carlo study

Kamilla Faizullina[1], Evgeni Burovski[1,2]

[1]HSE University, 101000 Moscow, Russia,
[2]Landau Institute for Theoretical Physics, 142432 Chernogolovka, Russia

We study a model where the monomers of a self-avoiding walk (SAW) on a lattice carry XY spins. We consider the regime where both spins and conformations are dynamic and a single coupling constant governs both magnetic and structural properties: the monomers interact via a short-range spin-dependent coupling. We perform large-scale Monte Carlo simulations of the model and characterize the joint transition where both structural and magnetic degrees of freedom order.

1. Introduction

A linear polymer, or macromolecule, is a long molecule formed by monomers joined in a sequence by covalent bonds. The simplest model of polymer is classical homopolymer model which is represented by interacting (also known as collapsing) self-avoiding walk (SAW) (see, e.g., [1] and references therein). This model captures the θ-transition from the extended, denaturated regime to the globular one (see, e.g., [2] and references therein).

An extension of the interacting SAW model for magnetic polymers was first introduced in Ref. [3], which considered a self-avoiding walk where monomers carry Ising spins and interact via short-range Ising-type coupling. Mean-field studies and early Monte Carlo simulations of Ref. [3] indicated a first-order transition between a paramagnetic coil and a ferromagnetic globular phase. Recent large-scale Monte-Carlo simulations of this model, [4, 5], have shown that the nature of the transition is, in fact, dependent on the dimensionality: in three spatial dimensions (i.e., on a cubic lattice) the transition is indeed first-order; however in two dimensions (i.e., on a square lattice), the transition is continuous.

In this paper, we extend our previous work on the Ising-type model [5] to consider the SAWs with monomers carrying XY spins and interacting via the short-range XY-type coupling. To this end, we construct a Monte Carlo (MC) simulation which samples both SAW conformations and XY spin configurations. To extract the thermodynamic limit behavior, we need to simulate large length SAWs with low statistical erorrbars. This is both CPU time and memory consuming, with total resource requirements beyond simple desktop capabilities. We therefore run our simulations on the HSE University HPC Cluster *cHARISMa*.

The rest of the paper is organized as follows. In Sec. 2 we introduce the model and relevant physical observables. In Sec. 3 we detail our MC algorithm. In Sec. 4 we discuss the details of our implementation of the algorithm and present simulation results. Finally, we present our conclusions and outlook for further work.

2. XY model on Self-avoiding walks.

A molecular conformation of the length N is represented by a self-avoiding walk (SAW) of length N—i.e., having $N-1$ edges and N nodes—on a regular lattice. Each i-th node of the walk carries a spin-like variable s_i which is associated with an angular variable $\theta_i \in [-\pi; \pi)$, with $i = 1, \ldots, N$.

Given a conformation, u, of N steps and a sequence of N spins, s, the Hamiltonian is defined

as the sum over all non-repeating neighbor pairs $\langle i,j\rangle$ in conformation u:

$$H(u,s) = -J\sum_{\langle i,j\rangle}\cos(\theta_i-\theta_j)\ . \tag{1}$$

Here J is the spin-spin coupling constant.

Let U_N be a set of all SAW conformations of N monomers. The equilibrium partition function for the chain of the length N is the sum over all SAW conformations of N monomers and the integral over all spin space:

$$Z(J) = \sum_{u\in U_N}\int_{-\pi}^{\pi}\frac{d\theta_1 d\theta_2\dots d\theta_N}{(2\pi)^N}\,e^{J(\cos\theta_1-\theta_2)}e^{J(\cos\theta_2-\theta_3)}\cdots e^{J(\cos\theta_{N-1}-\theta_N)} \tag{2}$$

Here we set the heat bath temperature $T=1$ to fix the energy scale. This way, the only dimensionless coupling constant is J, Eq. (1). Note that in Eq. (2) we both sum over the conformations and integrate over the spin variables.

Physical observables. The mean magnetization is defined as a vector:

$$\langle\vec{m}\rangle = \frac{1}{N}\left\langle(\sum_{i=1}^{N}\cos\theta_i,\sum_{i=1}^{N}\sin\theta_i)\right\rangle \tag{3}$$

The second moment of magnetization is a square of the norm:

$$\langle m^2\rangle = \frac{1}{N^2}\left\langle(\sum_{i=1}^{N}\cos\theta_i)^2+(\sum_{i=1}^{N}\sin\theta_i)^2\right\rangle \tag{4}$$

From measurements of the average magnetization per spin $\langle m\rangle(J)$, we can obtain the value of the magnetic cumulant (Binder parameters) of fourth order [6], which is helpful to study magnetic phase transition:

$$U_4(J) = 1-\frac{\langle m^4\rangle}{3\langle m^2\rangle^2} \tag{5}$$

To study the structural phase transition, we use the mean square end-to-end distance (radius) of self-avoiding-walks which is defined as the sum over all configurations:

$$\langle R_N^2\rangle = \frac{1}{Z_N}\sum_{|u|=N}|u|^2e^{-E_u}, \tag{6}$$

where $|u|$ is the Euclidean distance between the endpoints of conformation u, and Z_N is partition function in the canonical assemble (2). As $N\to\infty$, the mean radius of SAWs is believed to scale as

$$\langle R_N^2\rangle \sim N^{2\nu}. \tag{7}$$

Here ν is the metric exponent [1].

3. Numerical simulations.

A common approach to simulating SAW-like models relies on chain-growth Monte Carlo techniques—the variations of the pruned-enriched Rosenbluth method [7]. Methods of this family start losing efficiency on approach to the globular regime, and we instead formulate a Metropolis algorithm [8] Monte Carlo process of simulating the model (1)-(2) for a fixed SAW length N.

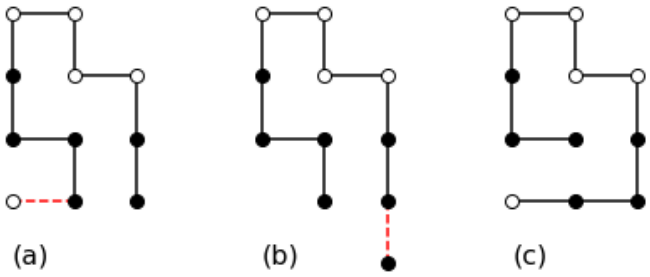


Figure 1. Monte-Carlo updates.

In this work, we construct Markov Chain Monte Carlo method for fixed-length chain consisting of three types of updates. We refer to them as a reptation update (also known as a BEE move in the literature), the reconnection update and a Wolff Cluster update. In each iteration, the algorithm chooses an update according to predefined probabilities P_{local},$P_{\text{reconnect}}$ and P_{Wolff}, respectively. The sum of probabilities is always equal to one: $P_{\text{local}} + P_{\text{reconnect}} + P_{\text{Wolff}} = 1$. We note that a similar algorithm was previously used for studying the Ising model on a SAW in Ref. [5]. Below we detail the algorithm specifically for the XY model (1)-(2).

BEE-move is a bilocal reptation update, see Figure 1 (a)-(b). The algorithm removes a monomer from one end of a SAW and adds a monomer to the other end. One iteration of the BBE move algorithm consists of the following operations:

1. Fix the Energy (1) of the current conformation u.
2. Choose randomly the endpoint of the chain with equal probabilities :
   - Remove the first monomer of chain and add a new node to the end.
   - Remove the last monomer of chain and add a new starting node to the chain.
3. Then the algorithm chooses at random a neighbor of the supplementable endpoint of current conformation with equal probability $p = \frac{1}{2d}$, where d is the lattice dimension.

   If the current conformation already has this point, then reject the attempted move, make null transition and count old configuration. Stop the iteration.
4. If the chosen lattice neighbor is free, the algorithm generates new spin variable $\theta_{\text{new}} \sim U(-\pi, \pi)$.
5. Add new node with new generated spin to the chain and delete the node from the endpoint chosen to be removed.

   Calculate the energy of new generated state u_{new}.
6. Make the transition to new generated system state accordingly to the Metropolis rule:

$$A(u_0 \to u_{new}) = \begin{cases} e^{-J(E_{u_{new}} - E_{u_0})}, & \text{if } E_{u_{new}} - E_{u_0} > 0; \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

   If the new state was not accepted, cancel the updates from Step 4.

The reptation update is very convenient to implement and has the time and memory complexity $O(1)$. However, the autocorrelation time is long for magnetic variables and for structure $\eta \sim N^2$ and the system can be locked in the frozen states when both ends of the conformation are surrounded by $2d$ neighbors. To address the problem, we also implement Reconnect update to accelerate conformations generation and Wolff-cluster algorithm to effectively explore spin configuration space.

Reconnect is a non-local update based on ideas of Worm algorithm [9], see 1 (a)-(c). In this system update, only connections of conformation are changed. The Metropolis acceptance probability is always equal to one as the energy does not change. The time complexity $O(N)$.

Cluster update. The Wolff cluster update allows to generate independent magnetic samples [10]. The algorithm effectively samples spin configurations *for a fixed conformation*.

The main idea of the update is to form the cluster C of spins and flip its spins. The update rules should obey the detailed balance condition. We follow the classical way of cluster update implementation for XY model discussed in Ref. [11].

To flip a spin's using chosen direction $\vec{n}$ means to change sign of vector $\vec{n}\vec{S}_i(\theta_i) = \vec{n}(\cos\theta_i \sin\theta_i)$ to the opposite.

1. Choose randomly the spin $\overrightarrow{S_{start}}$ from the chain using discrete uniform distribution.
2. Choose the direction on the unit circle $\overrightarrow{n}$ using uniform distribution $\theta_{new} = U(-\pi, \pi)$: $\overrightarrow{n} = [\cos(\theta_{new}), \sin(\theta_{new})]$.
3. Update the chosen monomer $\overrightarrow{S_{start}}$. To do it, we need to change sign of $\vec{n}\overrightarrow{S_{start}}$ to the opposite. This action is the subtraction the reflected $\vec{n}$ from $\overrightarrow{S_{start}}$. Therefore, the flip is implemented as following: $\overrightarrow{S_{start}} := \overrightarrow{S_{start}} - 2 \times (\overrightarrow{S_{start}}\vec{n}) \times \vec{n}$. We add this flipped spin to the cluster C and point it as flipped.
4. Take one spin $\overrightarrow{S_i}$ from the cluster to visit all its neighbors S_j. Add new spin to the cluster with the probabilities $P_{add}(\overrightarrow{S_i}, \overrightarrow{S_j}) = 1 - \exp\left(2J(\overrightarrow{S_i}\overrightarrow{n})(\overrightarrow{S_j}\overrightarrow{n})\right)$, where $\overrightarrow{S_i}\overrightarrow{n}$ is a dot product.

   If the spin $\overrightarrow{S_j}$ was added to cluster and has not been flipped, make update: $\overrightarrow{S_j} := \overrightarrow{S_j} - 2 \times (\overrightarrow{S_j}\vec{n}) \times \vec{n}$. Mark the spins as flipped.

   After visiting all neighbors, remove $\overrightarrow{S_i}$ from the cluster.
5. Repeat the previous step until the cluster is empty.

The Wolff cluster update effectively decorrelates spin configurations, while keeping the conformation fixed. The time complexity of the cluster update is $O(N)$.

4. Computational results.

The computational bottleneck of the Monte Carlo algorithm of Sec. 3 is finding nearest neighbors of a node of a SAW conformation on a $d = 3$ dimensional lattice. We thus use local hashing: we tabulate the nearest neighbors of a cubic lattice of linear size M before starting the simulations. This way, the time complexity of finding nearest neighbors of a given lattice site is $O(1)$, at the expense of the memory requirement of $2dM^d$. An alternative would be to use KD-tree type data structures for the nearest-neighbor search (see, e.g. [12]), with the worst time query complexity of $O(N^{1-1/d})$, where N is the SAW length. Since we are interested in the thermodynamic limit behavior, our simulations require $N \gg 1$, thus tabulating the neighbors is vastly preferrable.

To avoid finite-lattice effects we use $M \sim N$— in principle, this can be pushed down to $M \sim N^\alpha$ with α slightly larger then $1/2$ since a typical linear size of a non-interacting SAW is $\sim N^\nu$ with the metric exponent $\nu \approx 0.58$ (see e.g. [13] and references therein). With this choice, typical memory consumption for $N = 700$ is about 25 Gb.

Our open-source C++ implementation of the Monte Carlo algorithm is available in Ref. [14]. The implementation only has a command-line user interface, as appropriate for running large-scale simulations using batch queueing systems on HPC systems.

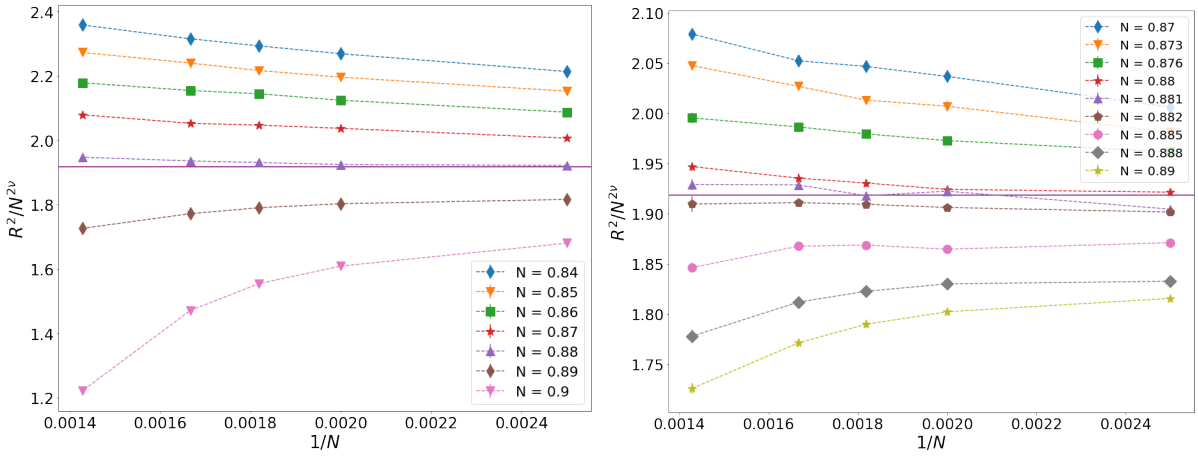


Figure 2. Scaled Mean-squared end-to-end distance using $\nu = 1/2$ as a function of $1/N$ where N is the chain length.

We performed MC simulations using the HSE University HPC *cHARISMa* cluster [15]. Simulations were run on Intel Xeon Gold 6152 2.1-3.7 GHz compute nodes with up to 150 Gb of memory.

To collect enough statistics for low enough errorbars, we use MC runs with up to 9×10^{10} MC steps. A single simulation of a chain with $N = 400$ at a given value of the coupling constant J takes approximately 18 CPU-hours; longer chains with $N = 700$ take up to 90 hours of CPU time. We employ the fact that MC simulations are embarrassingly parallel and run up to 100 independent simulations to speed up collecting the MC statistics. The total CPU time to date for the results reported in this work in progress is about $\sim 10^5$ CPU-hours.

4.1. The globule-coil transition

4.1.1. Structure

We calculate the mean square end-to-end distance of SAWs (6) to investigate the globule-coil transition. From the Flory prediction, at the theta-point, the classical interacting SAW has following value $\nu_\theta = \frac{1}{2}$ [1]. We investigate the asymptotics (7) in the phase region. Figure 2 shows the scaled mean-squared end-to-end distance by $\nu_\theta = \frac{1}{2}$ as a function of the chain length $1/N$ for a range of values interaction energy J. The horizontal line corresponds to the critical value ν at the point of phase transition. From these scaled numerical results, we deduce that the phase transition happens at $J \approx 0.88$.

Figure 3 (right) illustrates scaled curves of mean radius crossing at the transition point. Using the crossing points of curves, we obtain the following estimate as $N \to \infty$:

$$J_\theta \approx 0.88(1). \tag{9}$$

4.1.2. Magnetic transition

The Binder parameter could be used to determine the universality class [16]. At the phase transition, the Binder ratio has a divergent feature at the step if the system has the first order transition. Figure 3 (left) shows that Binder curves diverge for 3D XY model on SAWs. We can see that in the large systems at the transition region the amplitude of the dip of U_4 grows as N increases. This is a typical behavior for first order transitions [16].

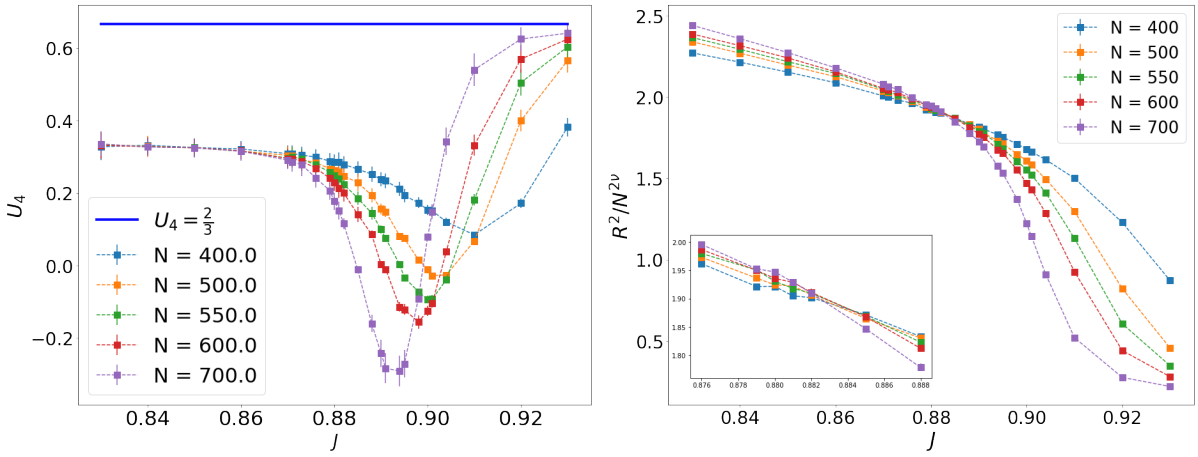


Figure 3. Binder cumulants (5) and mean radius (6). The mean radius is scaled by $\nu\frac{1}{2}$.

4.1.3. Conclusions and outlook

In this work, we study the XY model on self-avoiding walks on the 3D cubic lattice using Monte-Carlo simulations. We consider the model where both conformations and spins are dynamic, so that we numerically observe indications of a joint transition from a paramagneric coil to a ferromagnetic globule. Our computational results indicate that the transition is first order. More work is needed to fully characterize the structural properties of the model and accurately quantify critical properties of the observed transition.

An intriguing development in statistical physics is a suggestion to classify phases of classical [17] and quantum [18] systems and their critical behavior using artificial neural networks (NN). In this approach, training and testing sets for the NN are generated via MC sampling of the configuration space of the model, and our present work serves as generating the datasets for the NN training, testing and analysis.

4.1.4. Acknowledgements

Research is supported by the grant 22-11-00259 of the Russian Science Foundation. Numerical simulations were done using the computational resources of HPC facilities at HSE University [15].

References

[1] E.J.J. Van Rensburg. *The Statistical Mechanics of Interacting Walks, Polygons, Animals and Vesicles*. Oxford Lecture Series in Mathe. Oxford University Press, 2015. ISBN: 9780199666577. URL: https://books.google.ru/books?id=LIVMCAAAQBAJ.

[2] Carlo Vanderzande. *Lattice models of polymers*. Cambridge University Press, 1998.

[3] T Garel, H Orland, and E Orlandini. "Phase diagram of magnetic polymers". In: *Eur. Phys. J. B* 12 (1999), pp. 261–268.

[4] Damien Paul Foster and Debjyoti Majumdar. "Critical behavior of magnetic polymers in two and three dimensions". In: *Phys. Rev. E* 104 (2 2021), p. 024122. DOI: 10.1103/PhysRevE.104.024122. URL: https://link.aps.org/doi/10.1103/PhysRevE.104.024122.

[5] Kamilla Faizullina, Ilya Pchelintsev, and Evgeni Burovski. "Critical and geometric properties of magnetic polymers across the globule-coil transition". In: *Phys. Rev. E* 104 (5

2021), p. 054501. DOI: 10.1103/PhysRevE.104.054501. URL: https://link.aps.org/doi/10.1103/PhysRevE.104.054501.

[6] Kurt Binder and Dieter W. Heermann. *Monte Carlo Methods for the Sampling of Free Energy Landscapes*. 2010, pp. 153–174. ISBN: 9783642031625. DOI: 10.1007/978-3-642-03163-2_6.

[7] Hsiao-Ping Hsu et al. "Growth-based optimization algorithm for lattice heteropolymers". In: *Phys. Rev. E* 68 (2 2003), p. 021113. DOI: 10.1103/PhysRevE.68.021113. URL: https://link.aps.org/doi/10.1103/PhysRevE.68.021113.

[8] Nicholas Metropolis et al. "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092. DOI: 10.1063/1.1699114. eprint: https://doi.org/10.1063/1.1699114. URL: https://doi.org/10.1063/1.1699114.

[9] N. Prokof'ev and B. Svistunov. "Worm Algorithms for Classical Statistical Models". In: *Phys. Rev. Lett.* 87 (16 2001), p. 160601. DOI: 10.1103/PhysRevLett.87.160601. URL: https://link.aps.org/doi/10.1103/PhysRevLett.87.160601.

[10] Ulli Wolff. "Collective Monte Carlo Updating for Spin Systems". In: *Phys. Rev. Lett.* 62 (4 1989), pp. 361–364. DOI: 10.1103/PhysRevLett.62.361. URL: https://link.aps.org/doi/10.1103/PhysRevLett.62.361.

[11] Mark Newman and Gerard Barkema. *Monte Carlo methods in statistical physics chapter 1-4*. Oxford University Press: New York, USA, 1999.

[12] Mark de Berg et al. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[13] Nathan Clisby. "Accurate Estimate of the Critical Exponent ν for Self-Avoiding Walks via a Fast Implementation of the Pivot Algorithm". In: *Phys. Rev. Lett.* 104 (2010), p. 055702. DOI: https://doi.org/10.1103/PhysRevLett.104.055702.

[14] Kamilla Faizullina. *saw*. https://github.com/kamilla0503/saw_models. 2022.

[15] P. S. Kostenetskiy, R. A. Chulkevich, and V. I. Kozyrev. "HPC Resources of the Higher School of Economics". In: *J. Phys.: Conf. Ser.* 1740 (2021), p. 012050. DOI: 10.1088/1742-6596/1740/1/012050. URL: https://doi.org/10.1088/1742-6596/1740/1/012050.

[16] Kurt Binder. "Finite size scaling analysis of Ising model block distribution functions". In: *Zeitschrift für Physik B Condensed Matter* 43.2 (1981), pp. 119–140.

[17] Juan Carrasquilla and Roger G. Melko. "Machine learning phases of matter". In: *Nature* 13 (2017), pp. 431–434.

[18] Giuseppe Carleo and Matthias Troyer. "Solving the Quantum Many-Body Problem with Artificial Neural Networks". In: *Science* 355 (2017), p. 602.

Автоматизированное распараллеливание программ для гетерогенных кластеров с помощью системы SAPFOR *

Н.А. Катаев, А.С. Колганов

ИПМ им. М.В. Келдыша РАН

В статье будет рассмотрен подход к автоматизированному распараллеливанию программ для кластеров с помощью системы SAPFOR (System FOR Automated Parallelization). Главной целью системы SAPFOR является автоматизация процесса отображения последовательных программ на параллельные архитектуры в модели DVMH, которая является моделью программирования, основанной на директивах. Помимо этого система SAPFOR позволяет выполнять автоматически некоторый класс преобразований над исходным кодом программы по запросу пользователя через графический интерфейс. На определенных классах задач пользователь системы SAPFOR может рассчитывать на полностью автоматическое распараллеливание, если программа была написана или приведена к потенциально параллельному виду. Также в статье будут описаны подходы к построению схем распределения данных и вычислений на распределенную память в модели DVMH. Эффективность полученных алгоритмов построения схем распределения данных и вычислений будет продемонстрирована на примере некоторых приложений из пакета NAS Parallel Benchmarks.

Ключевые слова: SAPFOR, DVMH, автоматизация распараллеливания, распределение данных, распределение вычислений, гетерогенные кластеры

1. Введение

Высокопроизводительные вычислительные технологии получили широкое распространение среди большого количества областей науки: вычислительная гидродинамика, исследования климата и окружающей среды, нейронные сети и искусственный интеллект, и многие другие. Сложились разнообразные подходы к организации параллельных вычислений, предполагающие использование различных технологий параллельного программирования [1]. При этом при выборе и применении предпочтительных подходов для решения конкретной задачи часто приходится сталкиваться с различными трудностями [2].

Среди них можно выделить необходимость одновременного применения различных технологий программирования (MPI+OpenMP, MPI+OpenMP+CUDA и т.д.), чтобы задействовать все доступные вычислительные ресурсы. При этом прикладной программист должен обладать знаниями, позволяющими применять каждую из них. Множество используемых технологий может меняться по мере развития доступной вычислительной аппаратуры, что усложняет сопровождение и развитие уже написанных программных комплексов. Кроме того запуск вычислительной задачи на сложной, часто гибридной, вычислительной системе может сопровождаться необходимостью подбора многочисленных параметров для достижения максимальной производительности. В сложившейся ситуации крайне желательной становится автоматизация максимального количества этапов, составляющих процесс разработки и сопровождения параллельной программы.

Одним из направлений такой автоматизации является разработка единых подходов, охватывающих сразу несколько уровней параллелизма и позволяющих отображать программу на различные архитектуры вычислителей. Примером такого единого подхода является стандарт SYCL [3], который добавляет параллелизм в последние версии языка C++

*Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-01-00631 А.

для поддержки параллельного выполнения на GPU, CPU и FPGA. Параллельная программа должна явно описывать параллельное выполнение, при этом оставаясь программой на стандартном языке C++, а ответственность за реализацию параллелизма ложится на используемые компиляторы. Помимо открытых реализаций данного стандарта, активно развивается коммерческая версия от Intel (oneAPI). В исходном виде SYCL ориентирован на системы с общей памятью, но на его основе также разрабатывается открытый проект Celerity [4] для отображения программ на кластеры, оснащенные в узлах ускорителями.

Альтернативой такому подходу может быть применение директивных расширений стандартных последовательных языков программирования XcalabalACC [5], DVMH [6,7]. Преимущество таких подходов заключается в том, что они позволяют сначала разрабатывать и отлаживать последовательную программу, а потом добавлять в нее спецификации параллелизма, в то время как подходы, аналогичные SYCL, требуют, чтобы программа изначально разрабатывалась с использованием конструкций, описывающих параллелизм.

Являясь достаточно универсальной для описания различных уровней параллелизма, доступных на гибридном вычислительном кластере, DVMH модель скрывает конкретные технологии программирования внутри реализации компиляторов с DVMH языков. Это в свою очередь позволяет при необходимости расширять множество поддерживаемых архитектур, избегая значительных изменений в DVMH модели, и обеспечивает переносимость существующих DVMH программ. Еще одним уровнем автоматизации, который обеспечивает DVM система, является возможность динамической настройки запускаемых параллельных программ на выделенные для их выполнения вычислительные ресурсы [8].

Несмотря на то, что модели программирования, опирающиеся на директивные расширения существующих языков, являются высокоуровневыми, их применение прикладным программистом все равно требует достаточных знаний в области параллельных вычислений и может быть сопряжено с трудностями. Способствовать решению данной проблемы может дальнейшая автоматизация процесса распараллеливания, связанная с созданием автоматизирующих систем, которые упрощают перевод последовательной программы в параллельную. Однако полностью автоматическое распараллеливание произвольных программ сталкивается со значительными трудностями, не позволяющими достичь приемлемой эффективности получаемых параллельных версий. Поэтому на рассматриваемые последовательные программы могут накладываться существенные ограничения, а пользователь получает возможность описывать свойства программ, которые невозможно проанализировать [9–12].

В связи с этим перспективным видится смешанный подход, который объединяет использование высокоуровневой модели параллельного программирования, системы автоматизации, ответственной за выполнение наиболее трудоемких этапов распараллеливания, и возможность для пользователя контролировать ход распараллеливания и принимать в нем активное участие [13].

В качестве инструмента автоматизации может выступать система SAPFOR (System FOR Automated Parallelization) [14, 15], ориентированная на использование DVMH языков как целевых. С одной стороны система включает автоматически распараллеливающий компилятор, при этом инкапсулированные в DVMH модели возможности по динамической настройке запускаемых параллельных программ упрощают его разработку. С другой стороны, система обладает широкими возможностями по статическому и динамическому анализу программ [18, 19], автоматизирует выполнение преобразований исходной программы, позволяя пользователю выбирать фрагменты программы, которые должны быть преобразованы. Графический интерфейс пользователя позволяет управлять процессом распараллеливания [17].

В предыдущих работах [13, 16] рассматривались возможности системы SAPFOR, направленные на распараллеливание программ для систем с общей памятью (мультипроцессор, GPU). Данная статья охватывает дальнейшее расширение возможностей системы SAPFOR в направлении отображения программ на многоядерные гибридные вычислитель-

ные кластеры. В статье основное внимание уделяется алгоритмам распределения данных и вычислений между узлами кластера в модели DVMH.

Статья состоит из введения, 3 разделов и заключения. В разделе 2 приведен краткий обзор работ, направленных на автоматизацию разработки параллельных программ для систем с распределенной памятью. Раздел 3 посвящен алгоритмам распределения данных и вычислений, реализованным в системе SAPFOR. Результаты вычислительных экспериментов, охватывающие распараллеливание некоторых программ из набора NAS Parallel Benchmarks [20], приведены в разделе 4.

2. Обзор существующих работ

Распараллеливание для кластера обладает существенными особенностями, отличающими его от распараллеливания для систем с общей памятью. Разработчику приходится учитывать размещение данных на узлах системы наряду с распределением вычислений между отдельными вычислительными устройствами, чтобы обеспечить доступ к удаленным данным, максимально сократив при этом коммуникационные издержки. Таким образом, важную роль начинает играть требование локальности данных, используемых на каждом узле, и необходимость сбалансированного распределения данных, чтобы равномерно загрузить вычислительные устройства работой. Ситуация усугубляется тем, что приходится принимать глобальные решения в рамках всей программы в целом, так как отдельные ее фрагменты могут накладывать противоречивые требования, что в конечном счете приведет к дополнительным коммуникациям, направленным, на перераспределение данных.

В связи с этим рассматриваются различные подходы к распараллеливанию программ на вычислительный кластер. Так как процесс распараллеливания для систем с распределенной памятью можно разделить на три этапа: распределение данных, распределение вычислений, организация коммуникаций для доступа к удаленным данным или перераспределения данных, то автоматизации могут подвергаться только некоторые из них.

Например, в работе [21] рассматривается подход к построению оптимального распределения вычислений и организации коммуникаций, опирающийся на применение полиэдральной модели распараллеливаемой программы, для предварительно заданного фиксированного распределения данных. Подход, основанный на предварительно заданном распределении данных, также применялся в инструменте [22], при этом интересно, что инструмент обеспечивал пользователя диалоговой оболочкой, позволяющей управлять процессом распараллеливания, задавать распределение данных и управлять преобразованиями программ. Распределение вычислений выполнялось автоматически и основывалось на правиле собственных вычислений, когда запись значений выполняется на том процессоре, который владеет записываемыми данными. Подход, описанный в [21] позволяет ослабить это ограничение, в том числе обеспечивая размножение данных между узлами.

Инструмент Molly [11], расширяет возможности компилятора Polly [23] для систем с общей памятью, построенного на базе LLVM [24] и основанного на использовании полиэдральной модели. При этом вводится специальный тип данных, описывающий распределяемые массивы, что позволяет контролировать отсутствие операций адресной арифметики, применяемых к распределяемым данным. Распределение данных выполняется равными блоками распределяемых массивов между процессами, при этом выравнивание данных друг на друга не предусмотрено, а для распределения вычислений применяется правило собственных вычислений. Предполагается, что в дальнейшем пользователь сможет корректировать как распределение данных, так и вычислений, задавая соответствующие директивы. Кроме того накладываются дополнительные ограничения на структуру распараллеливаемых фрагментов (SCoP), вводится требование глобальности распределяемых данных, что позволяет избежать подробного межпроцедурного анализа, редукционные операции также не поддерживаются.

Подход с использованием директив, описывающих распределение данных, также предложен в работе [25]. Директивы содержат большое количество параметров, позволяющих гибко управлять требуемым распределением данных. Также предлагается нестандартное распределение массивов с перекрытиями, что позволяет сократить частоту обменов данными. При этом использование специальных директив для описания такого распределения упрощает разработку программы и вероятность ошибок при задании сложных индексных выражений в обращениях к массивам.

Подход, основанный на оптимизации распределения вычислений и необходимых коммуникаций, также приведен в работе [27], расширяющей возможности полиэдрального компилятора Pluto [26] для систем с общей памятью. При этом распределение данных как таковое не требуется, а размещение данных на узлах в каждый конкретный момент определяется выполняемыми над ними вычислениями. Данный подход не предусматривает глобального принятия решений по распределению данных, которое максимально бы соответствовало распределению вычислений, что может привести к увеличению частоты и объема коммуникаций.

Построение распределения данных наряду с распределением вычислений и оптимизацией коммуникаций было реализовано в инструменте Paradigm [28]. Исследования были направлены на распараллеливание последовательных программ на языке Фортран 77. Инструмент предполагал поддержку достаточно широкого класса задач, в том числе за счет поддержки нерегулярных вычислений и распараллеливания циклов с зависимостями за счет организации конвейерного выполнения. При этом в работе отмечаются ограничения связанные с определением правила выравнивания измерения массивов, а экспериментальные результаты приводятся для небольших вычислительных ядер.

3. Построение схем распределения данных и вычислений в системе SAPFOR

3.1. Варианты отображения последовательной программы на кластер

Распределение данных – одна из основных проблем отображения последовательной программы на кластер в случае использования регулярных сеток. Для эффективного задействования всей мощности вычислительного кластера требуется учитывать специфику обработки данных в циклах последовательной программы, так как зачастую в них содержится основная вычислительная нагрузка. Рассмотрим следующие варианты отображения последовательной программы на кластер:

- выполняется только распределение вычислений, а данные остаются размноженными на всех узлах. При таком подходе существенно упрощается преобразование последовательной программы в параллельную для кластера, так как все данные дублируются на каждом узле кластера. В свою очередь, каждый цикл может «требовать» свое распределение, что легко реализуется данной моделью;

- выполняется распределение как данных, так и вычислений. При таком подходе необходимо учитывать интересы всех циклов, участвующих в распараллеливании. В этом случае в силу того, что на каждом узле присутствует только часть данных, необходимо выполнять их пересылки для корректного выполнения последовательных участков программы.

Система SAPFOR преобразует исходную программу в параллельную с использованием модели DVMH. Данная модель использует второй вариант отображения последовательной программы на кластер – отображение как данных, так и вычислений на узлы кластера. В связи с этим необходимо обеспечить такое распределение данных, чтобы количество коммуникаций между процессорами, а также их объем были как можно меньше. Можно отме-

тить, что с помощью модели DVMH можно реализовать и первый вариант, но, во-первых, параллельная программа будет требовать столько же памяти на каждом узле, что и последовательная, и, во-вторых, эффективность выполнения такой программы может быть ниже из-за большого объема коммуникаций, возникающих в момент перераспределения данных.

Алгоритм распределения данных состоит из двух этапов. На первом этапе выполняется межпроцедурный анализ всей программы: анализируются циклы и используемые в них массивы. Затем собранная информация обобщается и выполняется поиск распределения данных с как можно меньшими издержками по пересылке данных между узлами.

3.2. Определение распределяемых массивов

По умолчанию система SAPFOR считает все массивы в программе распределяемыми. Распределяемый массив – это такой массив, для которого необходимо построить распределение данных с помощью DVMH-директив и выполнить соответствующее выбранному распределению отображение вычислений в параллельных циклах и одиночных операторах. Но не все массивы могут быть распределенными. Чтобы снизить количество распределяемых данных, системе SAPFOR необходимо уметь распознавать такие ситуации либо в автоматическом, либо в полуавтоматическом режиме. Рассмотрим случаи, когда массив не требует распределения.

К первой категории массивов, которые не требуется распределять, относятся те массивы, которые были определены как приватизируемые или редукционные системой SAPFOR или были указаны в соответствующих спецификациях пользователем в исходном коде программы или через графический интерфейс. Данные массивы являются вспомогательными в местах использования (в основном в циклах), при этом использовать в циклах распределенный массив как приватизируемый или редукционный запрещено DVM-системой.

Ко второй категории относятся массивы, которые участвуют в операторах ввода-вывода, а также массивы, которые передаются как параметры во внешние процедуры (например, процедуры стандартных библиотек, либо процедуры, которые не доступны для анализа системе SAPFOR). В операторах ввода-вывода разрешается задавать только по одному массиву и только целиком из-за ограничений DVM-системы (то есть можно выводить целиком весь массив). Все остальные случаи отменяют распределение данного массива. Для того чтобы не отменять распределение массивов, которые участвуют в сложных операторах ввода-вывода, а также во внешних процедурах, требуется заводить массивы-копии и вставлять копирования до и после соответствующих операторов. В текущий момент в системе SAPFOR такое преобразование не автоматизировано.

К третьей категории относятся массивы, которые система SAPFOR автоматически отфильтровывает по тем или иным причинам. Процесс фильтрации состоит в том, чтобы запретить использовать распределяемые массивы, которые в дальнейшем будут отображены на разные деревья выравнивания и соответственно на разные DVMH-шаблоны в общем гнезде потенциально параллельных циклов. Разные деревья выравнивания образуются из несвязанных между собой графов массивов. Также отфильтровываются массивы, объявленные в глобальной области видимости, которые могли быть использованы в процедурах, вызываемых из цикла, без явного использования в самом цикле. Такое ограничение связано с тем, что в DVMH-модели во всех параллельных циклах необходимо «видеть» все используемые массивы. Те данные, которые используются в вызываемых из цикла процедурах, но не используются в циклах, считаются «невидимыми» для DVMH-компилятора, что приведет к ошибке конвертации программы и ее выполнения.

После выполнения фильтрации происходит распространение состояния потенциальной распределенности массивов в соответствии со связями фактических и формальных параметров процедур в программе. По массивам, которые не были отфильтрованы, будет построен граф массивов, которые в свою очередь будет отображен в дерево выравнивания в модели DVMH.

3.3. Анализ программы: построение связей циклов и массивов

На данном этапе системой SAPFOR для каждой функции в исходной программе выполняется анализ всех ее операторов. Для каждого обращения к массиву, находящемуся внутри гнезда циклов, для каждого из измерений (отдельно и независимо) выполняется следующий анализ:

- выполняется поиск косвенной адресации в обращении. Косвенная адресация не может быть эффективно распараллелена DVMH-моделью в случае структурированных сеток;
- выполняется поиск более, чем одной итерационной переменной цикла в индексном выражении в обращении к массиву. Для корректного отображения данных и связывания распределения вычислений с распределением данных, необходимо однозначное соответствие индексного выражения в обращении к массиву с итерационной переменной одного из циклов в гнезде. Если имеется связь с одним циклом, то выполняется попытка сопоставления индексного выражения с итерационной переменной цикла I с шаблоном $a * I + b$ и вычисления данных коэффициентов;
- выполняется проверка всех измерений массива. Если обращение к массиву используется на запись, проверяется факт того, что все индексные выражения в обращении к массиву имеют хотя бы одну итерационную переменную цикла. Если итерационная переменная цикла не встречается в индексных выражениях, то для рассматриваемого цикла отмечается факт наличия неопределенной запись в массив.

Важно отметить, что вся информация привязывается к циклам, то есть для каждого цикла формируется список всех обращений к массивам. Для того, чтобы цикл был оптимально распараллелен системой SAPFOR, требуется, чтобы каждое индексное выражение в обращении по конкретному измерению массива содержало только одну итерационную переменную цикла или иными словами должно быть однозначное отображение измерений массива на циклы.

После обработки функции будет построена общая информация о «хороших» обращениях к массивам с привязкой этих обращений к циклам с коэффициентами $a * I + b$, где $a, b > 0$ – вычисленные константы, $a \neq 0$, а I – итерационная переменная цикла. Остальные обращения к массивам будут порождать неизбежные обмены между узлами кластера.

3.4. Анализ программы: построение графа массивов

Граф массивов является основной структурой данных, на основе которой строятся распределение данных и вычислений. Построение графа массивов обусловлено выбором целевой модели при создании параллельной версии программы. DVMH-модель требует выполнения правила собственных вычислений: каждый процессор изменяет только собственные данные, то есть данные, которые распределены на этот процессор. Кроме того вычисление одной итерации цикла должно целиком выполняться на одном процессоре, и следовательно элементы массивов, вычисляемые на одной итерации оказываются связаны между собой и должны быть размещены на одном процессоре.

Для соблюдения данного правила необходимо использовать взаимное выравнивание массивов между собой с помощью спецификации ALIGN. После распределения массивов с помощью спецификаций DISTRIBUTE и ALIGN получается дерево выравнивания, которое описывает связи между всеми массивами. Правило собственных вычислений требует, чтобы все массивы, используемые в одном цикле, принадлежали одному дереву выравнивания.

Граф массивов представлен в системе SAPFOR в формате CSR (Compressed Sparse Rows) и является неориентированным. Каждое измерение массива становится узлом гра-

фа, а дуги показывают связь одного измерения массива с другим. Для заполнения графа массивов необходимо обработать информацию о вычисленных коэффициентах a, b в обращениях к массивам и их связях с циклом. Каждая дуга в графе массивов связывает одно измерение массива Arr_1 с измерением массива Arr_2. Дуги добавляются по следующему принципу (W или Write означает обращение на запись, R или Read означает обращение на чтение):

- связываются измерения массивов, обращения по которым присутствуют в левой части операторов присваивания в цикле с типом дуги запись-запись (связь $W - W$) и весом $Loop_w * SumB$;
- связываются измерения массивов Arr_1 и Arr_2, причем обращение к массиву Arr_1 содержится в левой части операторов присваивания, а обращение к массиву Arr_2 содержится в правой части операторов присваивания или в условиях IF, причем не обязательно, чтобы массивы Arr_1 и Arr_2 были в одном операторе. Связь создается с типом дуги запись-чтение (связь $W - R$) по данному циклу с весом $Loop_w * SumB$;
- в случае отсутствия операций записи в массивы связываются измерения массивов, обращения по которым присутствуют в правой части операторов присваивания или условиях IF в данном цикле, причем два обращения к разным массивам не обязаны быть в одном операторе. Связь создается с типом дуги чтение-чтение (связь $R - R$) по данному циклу с весом $Loop_w * SumB$.

Под $SumB$ понимается совокупное количество байт, которое потребуется передать другим процессорам в случае неудовлетворения конкретной связи с циклом. В худшем случае необходимо передать целиком все измерение массива, отображенное на соответствующий цикл. Количество байт вычисляется из размерности типа используемого массива и номера измерения массива. Данные о размерах массивов всегда известны системе SAPFOR и должны быть получены либо от статического, либо от динамического анализов, либо от пользователя, иначе невозможно построить дерево выравнивания в модели DVMH.

Под $Loop_w$ понимается вес цикла. В зависимости от количества арифметических операций и обращений к массивам в телах циклов тот или иной цикл с меньшим количеством витков может выполняться дольше аналогичного цикла, но с большим количеством витков. Вес цикла оценивается статическим образом, либо путем динамического профилирования (получение времени выполнения данного цикла). Вес цикла показывает сколько раз цикл был выполнен за все время работы программы. Например, если цикл выполняется всего один раз (цикл инициализации), то можно пожертвовать количеством коммуникаций в данном цикле в пользу итерационного цикла, который может выполняться сотни, а то и тысячи раз, где каждый лишний переданный байт будет серьезно сказываться на производительности программы в целом. В случае недостаточности информации для оценки веса цикла система SAPFOR полагает $Loop_w = 1.0$, что означает равенство всех циклов в программе.

В случае добавления дуги с одинаковыми вершинами происходит увеличение веса данной дуги путем суммирования текущего веса и веса добавляемой дуги. Данное правило позволит выделить наиболее важные связи, что позволит уменьшить количество пересылок между процессами при выборе определенной схемы распределения данных.

Для того чтобы отличать дуги по типу связи ($W-W, W-R, R-R$), необходимо расставить приоритеты. Тип дуги с типом связи $W - W$ имеет самый высший приоритет, однако все дуги с типом $W - W$ имеют равный приоритет между собой. Это объясняется тем, что неудовлетворение данной связи приведет к невозможности распараллеливания данного цикла, так как не будет выполнено правило собственных вычислений для связываемых массивов, что в итоге повлечет за собой бОльшие коммуникации (так как все обращения к распределенным массивам на чтение в таком цикле в худшем случае должны быть «покрыты» с помощью доступа к удаленным данным). Дуги с типом связи $W - R$ имеют приоритет

над дугами с типом связи $R-R$. Дуги каждого из типов $W-R$ и $R-R$ имеют также равный приоритет между собой.

Чтобы обеспечить соотношение приоритетов для разных типов дуг, был реализован следующий алгоритм выделения дуг по приоритетам. Сначала посчитаем общую сумму всех дуг с типом $R-R$, $S1 = \sum_{[R-R]}$. Затем прибавим число $S1$ к дугам с типом $W-R$. Тем самым мы «выделим» приоритет $W-R$ типа над дугами типа $R-R$, сохранив между тем равный приоритет между схожим типом. Затем вычислим общую сумму весов всех дуг с типами связи $W-R$ и $R-R$, $S2 = S1 + \sum_{[W-R]}$ и добавим теперь число $S2$ ко всем дугам с типом $W-W$. Таким образом, будет выполнено правило приоритета дуг: вес любой дуги с типом $W-W$ будет больше, чем $W-R$ и $R-R$, вес любой дуги с типом $W-R$ будет больше, чем $R-R$.

3.5. Поиск наилучшего связывания массивов

После того, как был построен общий граф массивов, необходимо выбрать то множество дуг, которое будет отражать наилучшие связи между массивами и минимизировать коммуникации между процессорами. Таким образом, необходимо построить усеченный графа массивов (такой граф массивов, который не содержит циклов, порождающих конфликтные ситуации) для последующего создания распределения данных. Конфликты могут быть двух типов. Рассмотрим данные типы конфликтов:

- наличие цикла в графе массивов, для вершин которого нельзя построить единственный вариант выравнивания измерений массивов между собой (где каждая вершина соответствует измерению массива). Такие конфликты будем называть конфликтами первого типа (1);
- присутствует явная или косвенная дуга (через другие дуги графа) между двумя измерениями одного и того же массива. Такие конфликты будем называть конфликтами второго типа (2).

Рассмотрим два подхода, которые позволяют сократить количество друг в графе массивов и найти совокупность наиболее важных дуг. Первый подход – перебор совокупности наиболее важных дуг в графе. В данном подходе мы выполняем перебор всевозможных наборов дуг и их оценку общего веса. Набор дуг с максимальным весов и будет решением данной задачи. Решение данной задачи можно представить в виде нескольких этапов.

Первый этап – получение всех простых циклов в графе массивов. Простой цикл в графе – это замкнутый цикл без повторного прохода по ребру или посещения вершины дважды, за исключением начальной и конечной вершин. В системе SAPFOR цикл представляет собой набор дуг графа массивов с сохранением веса.

Нахождение всех простых циклов в графе массивов является NP-трудной задачей, поэтому для ограничения поиска по времени и ресурсам в случае больших графов вводится два параметра, которые позволят ограничить этот поиск: максимальный размер цикла (размером цикла будем называть количество входящих в него дуг), который необходимо добавить в список простых циклов и максимальная длина просматриваемой цепочки при рекурсивном поиске таких циклов в графе.

Первый параметр используется для ограничения по памяти, второй – по времени поиска всех простых циклов. Можно отметить, что накладываемые ограничения не всегда позволяют найти все простые циклы в графе массивов, а это значит, что алгоритм не всегда может найти точное решение для больших графов массивов. С другой стороны – чем больше размер (длина) цикла, тем больше массивов он связывает между собой. Тем самым для получения наилучшего выравнивания массивов между собой не обязательно находить все простые циклы в графе.

Второй этап – обработка найденных простых циклов. После нахождения всех простых

циклов (далее просто циклов), происходит их сортировка по размеру (длине), а также разделение на независимые группы по длине. Затем для каждого цикла выполняется сортировка его дуг по весу, и все циклы в каждой группе упорядочиваются по суммарному весу. Данные сортировки позволят наиболее быстрым образом обеспечить поиск и удаление конфликтных дуг в графе массивов.

Для устранения конфликта (1) можно удалить любую из дуг цикла, например, с минимальным весом. Для устранения конфликта (2) необходимо удалить такую дугу, чтобы явная или косвенная связь между двумя измерениями массивов, которая выражена дугами графа, пропала. Если есть конфликтные циклы, то запускается процесс устранения конфликтов. Рассмотрим такой подход, который позволяет удалять конфликтные и неконфликтные дуги. После применения данного подхода мы получим усеченный граф массивов, который не содержит циклов.

Каждый цикл характеризуется суммарным весом всех его дуг или просто общим весом. Ранее все циклы были отсортированы с учетом их веса и размерности от самых маленьких до самых больших циклов по размерности (длине), а циклы с одинаковой размерностью были отсортированы по убыванию их веса.

Для устранения конфликтов запускается рекурсивная процедура. Цель данной процедуры – удалить дуги с минимальным суммарным общим весом, что позволит получить наиболее оптимальное с точки зрения обменов между узлами распределение данных на основе построения графа массивов и задания соответствующих весов. В начале процедуры имеем нулевой общий вес удаленных дуг и пустой список удаляемых дуг. Выбираем очередной цикл из списка полученных конфликтных циклов. Пытаемся по очереди удалить каждую из дуг данного цикла и смотрим, что получается:

- если это первая дуга цикла, то удаляем эту дугу, прибавляем вес этой дуги к общему весу всех удаленных дуг и заносим эту дугу в список удаляемых. После удаления данной дуги некоторые циклы из полученного списка перестанут быть циклами и, соответственно, не будут принимать участие в дальнейшем выборе. Далее рекурсивно вызываем эту процедуру. Рекурсивный вызов завершается в том случае, если нет больше циклов с конфликтами. В этом случае получен суммарный вес и список необходимых для удаления дуг;

- если эта дуга цикла не первая, то у нас уже имеется общий вес и список удаляемых дуг на каком-то конкретном уровне рекурсивной вложенности вызова рассматриваемой процедуры. Также мы имеем текущий суммарный вес на текущем уровне рекурсивной вложенности. Если сумма веса очередной удаляемой дуги и текущего общего суммарного веса больше, чем найденный наименьший общий вес удаляемых дуг, то рекурсивный вызов удаления этой дуги не выполняется, так как удаление этой дуги повлечет за собой увеличение общего веса всех удаляемых дуг (таким образом выполняется отсечение перебора всех вариантов наборов дуг для удаления). Если мы завершили рекурсивный вызов и получили меньший вес, чем был найден до этого, то корректируется общий вес и соответствующий список дуг для удаления.

После того, как мы получили список дуг для удаления, происходит формирование усеченного графа массивов, такого графа, который не содержит циклов. Затем необходимо проверить, не возникнут ли конфликты второго типа в усеченном графе, или нет ли таких путей в графе, которые косвенно (через другие массивы и соответствующие им дуги) связывают измерения одного и того же массива. Данная конфликтная ситуация не образует цикл, но требует разрешения.

Для этого необходимо для каждого массива для всех уникальных комбинаций пар его измерений добавить фиктивную дугу между этими измерениями с очень большим весом (например, большим, чем сумма всех весов в графе) и повторить весь алгоритм поиска простых циклов и удаления конфликтов. Если мы нашли конфликтный цикл, то мы удаляем

дугу. Из-за особенности алгоритма (применение сортировок и удаление дуг с минимальным совокупным весом) будет выбрана не фиктивная дуга, а существующая дуга в графе. Таким образом, будут разрешены все конфликты второго типа.

Стоит отметить, что после такой операции по данному графу массивов не всегда можно построить выравнивание всех массивов между собой, особенно если в графе было много конфликтов или были применены ограничения на поиск простых циклов, что является минусом данного алгоритма. Оценка сложности данного алгоритма является экспоненциальной в зависимости от количества вершин в графе, что накладывает ограничения на его применимость на больших программах. Данный алгоритм может быть использован на сравнительно небольших программах, где количество узлов графа массивов не более 10.

Альтернативным алгоритмом поиска наиболее важных дуг является модифицированный алгоритм поиска минимального остовного дерева. Минимальное остовное дерево – такое дерево, которое является максимальным по включению ребер подграфом, не имеющее циклов, и в котором сумма весов ребер минимальна. Если исходный граф связный, то будет построено остовное дерево, если же в исходном графе несколько несвязных компонент, то результатом будет остовный лес.

В нашей задаче требуется найти набор дуг с наибольшим весом. Таким образом, без изменения общности алгоритма поиска минимального остовного дерева можно искать максимальное остовное дерево – такое дерево, в котором сумма весов ребер максимальна. Была использована самая простая реализация – алгоритм Дейкстры-Прима. Недостатком данного алгоритма является тот факт, что решение получается не самое лучшее, как в случае полного перебора, так как не выполняется перебор всех цепочек дуг и их сравнение между собой. Главное достоинство такого подхода заключается в линейной оценке его сложности в зависимости от количества вершин в графе. Данное достоинство вместе с возможностью распараллелить данный алгоритм делает возможным его использование на любой программе с любым количеством массивов.

3.6. Создание вариантов распределения данных

После того, как был получен усеченный граф (далее граф массивов), который связывает измерения массивов в соответствии с их использованием в циклах программы, можно построить варианты (схемы) распределения данных.

Так как граф может быть не связным, в нем ищутся все деревья, которые связывают массивы друг с другом. После такого поиска мы знаем, сколько поддеревьев у нас есть, и какие массивы в эти поддеревья входят. Для каждого дерева создается свой шаблон в DVMH-модели – DVMH TEMPLATE, который и будет распределяться с помощью директивы DISTRIBUTE и на который будут выровнены все массивы данного дерева. Шаблон представляет собой виртуальный массив, под который не отводится память в программе.

Шаблон создается по массиву с наибольшей размерностью, а среди одинаковых массивов одной размерности, выбирается тот, который занимает больше памяти. После того, как создан шаблон и найден массив, по которому строится этот шаблон, в граф добавляются дуги, связывающие измерения этого массива и измерения этого шаблона с весом 1.0 и типом связи $R-R$. Таким образом, в графе массивов появляется шаблон, до которого можно «добраться» по связям между массивами и узнать выравнивание на него.

Так как с шаблоном связан только один из массивов, необходимо уметь вычислять связь с шаблоном и для других массивов. Наибольшая сложность, которая может возникнуть при вычислении таких связей – не кратные коэффициенты при выравнивании на шаблон. Такие коэффициенты могут возникать в том случае, когда измерение одного массива требуется распределить, например, с раздвижкой $3*i$, а измерение второго массива требуется распределить на то же измерение шаблона с раздвижкой $2*i$. Таким образом, требуется вычислить наименьшее общее кратное и изменить соответствующие атрибуты в графе.

Варианты распределения данных создаются для каждого шаблона независимо, количе-

ство вариантов для каждого шаблона будет равно $2^{d(T_i)}$, а общее их количество – $\sum_{i=1}^{K} 2^{d(T_i)}$, где $d(T_i)$ – размерность шаблона (каждое измерение считается распределенным, либо размноженным), K – количество построенных шаблонов.

Правила выравнивания массивов на шаблон в модели DVMH не зависят от выбранного в дальнейшем распределения этих шаблонов. Для каждого массива из под-дерева, который не является шаблоном, выполняется поиск связи его измерений с шаблоном в этом поддереве. Поиск связи для конкретного измерения массива запускается в том случае, если это измерение присутствует в графе массивов. В результате поиска может быть не найдено связи с шаблоном по конкретному измерению, такое измерение будет размножено (указана * в спецификации ALIGN).

3.7. Создание директив распределения вычислений

Для создания директивы распределения вычислений необходимо найти массив из графа массивов, на который будет отображаться пространство витков потенциально параллельного цикла. Если в рассматриваемом цикле присутствует запись всего в один массив – он и будет выбран. Если массивов на запись несколько, то выбор происходит, прежде всего, среди массивов, которые участвуют в спецификации ACROSS, если таковая имеется. В модели DVMH требуется, чтобы цикл был отображен на массив, который участвует в спецификации ACROSS. Данная спецификация позволяет организовать конвейерное выполнение циклов с регулярными зависимостями по данным, необходимость ее задания для цикла определяется системой SAPFOR автоматически на основе результатов анализа программы.

В результате алгоритма поиска наилучшего выравнивания данных, который был описан выше, был получен усеченный граф массивов. Конфликтные ситуации первого и второго типов были разрешены с помощью удаления соответствующих ребер этого графа. В результате чего, интересы тех или иных циклов могли быть не учтены, что может привести к тому, что нельзя создать директиву распределения вычислений для этого цикла.

Если цикл не содержит ограничений на распараллеливание (считается потенциально параллельным), то директива «привязывается» к соответствующему циклу в дереве циклов. Для директивы заполняется информация о массиве, на который требуется отобразить вычисления, правила отображения, а также сохраняются свойства цикла, которые были получены в результаты его анализа системой SAPFOR и должны быть описаны в параллельной программе в виде спецификаций DVMH языков: приватные и редукционные переменные, информация о регулярных зависимостях по данным, теневые грани массивов и элементы массивов, которые должны быть отдельно получены с удаленного процессора с помощью спецификации REMOTE_ACCESS.

Далее выполняется объединение полученных директив для каждого из циклов, если имеет место тесная вложенность. Если цикл не тесно вложенный, то запускается преобразование, которое позволяет на основе информации, полученной от статического и динамического анализов, произвести объединение циклов и сделать их тесно вложенными (внесение инварианта цикла). Данное преобразование выполняется на лету и позволяет устранить не тесную вложенность, если это возможно, без потери результатов анализа и построенных структур.

В итоге, самый верхний цикл будет содержать объединенную информацию от всех тесно вложенных циклов, следующий по уровню вложенности цикл будет содержать объединенную информацию от всех тесно вложенных циклов, которые расположены ниже по уровню вложенности и т.д. Такой подход позволяет выбирать разные циклы для распараллеливания программы в зависимости от выбранного варианта распределения данных.

В зависимости от выбранного варианта распределения данных (шаблонов) будут выбраны соответствующие директивы распределения вычислений, а также созданы дополнительные директивы перераспределения данных и доступа к удаленным данным, если это потребуется.

4. Вычислительные эксперименты

В данном разделе исследуется эффективность программ в модели DVMH, получаемых в результате применения описанных алгоритмов распределения данных и вычислений. Нами было выполнено сравнение параллельных версий, полученных с помощью системы SAPFOR для вычислительных приложений BT, CG и EP из пакета NAS Parallel Benchmarks [20], с MPI-версиями данных программ, написанными их разработчиками вручную.

Исследование эффективности было выполнено на суперкомпьютере K10 [29], состоящем из процессоров Intel Xeon E5-2660 и графических ускорителей NVIDIA Tesla M2090. Каждый узел содержит два 8-ти ядерных процессора (CPU), связанных посредством общей памяти (архитектура NUMA), и три графических ускорителя (GPU). Для вычислительных экспериментов были использованы максимально возможные ресурсы только процессоров одного, двух и девяти узлов (графические ускорители не использовались). Время выполнения MPI программ и DVMH программ, полученных с помощью SAPFOR с использованием языков FDVMH и CDVMH, приведены в таблице 1.

Таблица 1. Время выполнения в секундах программ на языке Фортран и Си, NPB 3.3 класс C.

	MPI Fortran			**FDVMH**			MPI C			**CDVMH**		
	BT	CG	EP	BT	CG	EP	BT	CG	EP	BT	CG	EP
1 узел	100	21.7	27.4	80	25.9	22.6	123.6	35.9	30.6	97.1	25.8	28
4 узла	34	11.04	7.09	24.8	47	5.67	34.7	10.2	7.85	30	47	7
9 узлов	16.26	7.51	3.25	14.9	60.1	2.52	17.10	7.05	3.43	16.57	60.7	3.13

Время выполнения оригинальных версий, написанных разработчиками пакета NAS Parallel Benchmark, приведено в первой и третьей группе столбцов (MPI программы).

Изначально рассматриваемые последовательные программы были написаны только на Фортране, поэтому потребовалось выполнить перевод рассматриваемых программ на язык Си вручную. Чтобы получить автоматически распараллеливаемые версии программ с помощью системы SAPFOR, были выполнены их предварительные преобразования (подстановка функций, объединение циклов, сужение размерности приватных массивов и др.) [16]. Вторая и четвертая группа столбцов показывают время выполнения этих параллельных версий программ с использованием FDVMH и CDVMH языков. Большинство преобразований было выполнено системой SAPFOR, другая часть преобразований, не реализованных на данный момент в системе или специфичных для конкретной программы и трудно формализуемых в виде отдельного преобразования, была выполнена вручную.

Программы BT и EP, полученные с помощью системы SAPFOR, показывают практически схожие ускорения, что и MPI программы, написанные вручную разработчиками тестов. Но программа CG, начиная с 16 и более процессов, начинает замедляться по отношению к MPI программе. Это связно прежде всего с тем, что основная доля времени приходится на умножение разреженной матрицы на вектор, что приводит к косвенной индексации. Данная особенность не может быть эффективно распараллелена в текущей модели на регулярных сеток. Для этого требуется модификация системы SAPFOR для использования нового расширения DVMH-модели для нерегулярных сеток. При распараллеливании в текущей модели, системе SAPFOR постоянно требуется выполнять одну коллективную операцию по пересылке удаленных данных между всеми процессами, что приводит к замедлению на их большом количестве. Если использовать графические ускорители, то 16 процессов будет достаточно для достижения высокой производительности, а пересылки не будут так сильно сказываться.

5. Заключение

В статье был рассмотрен подход к автоматизированному распараллеливанию программ для кластеров с помощью системы SAPFOR. Работа полученных параллельных программы была продемонстрирована на примере некоторых приложений из пакета NAS Parallel Benchmarks.

Предлагаемый нами подход к разработке параллельных программ сочетает модель программирования на основе директив (DVMH) и инструменты автоматизации и взаимодействия с пользователем (SAPFOR). Разрабатываемые системы взаимно дополняют друг друга, что, в свою очередь, позволяет получить существенное преимущество по сравнению с другими моделями параллельного программирования, такими как MPI+OpenMP+CUDA, при разработке параллельных программ для гетерогенных кластеров.

Основная сложность при распараллеливании программ на кластер заключается в минимизации накладных расходов, вызванных коммуникационными обменами между вычислительными узлами. В системе SAPFOR был реализован алгоритм распределения данных, учитывающий глобальные связи между массивами программы. Такие связи порождаются совместным использованием разных массивов в одном цикле и необходимостью соблюдения правила собственных вычислений при распараллеливании циклов в модели DVMH. Выбор между противоречивыми связями выполняется на основе оценки потенциально возможных коммуникаций, в случае не соблюдения одной из связей.

В систему SAPFOR входит автоматически распараллеливающий компилятор, который успешно справляется с потенциально параллельными программами без вмешательств со стороны пользователя. Если же реализованный в системе SAPFOR анализ кода не справляется, то пользователь может помочь системе с помощью соответствующих директив, указав недостающие свойства программы, либо выполнить с помощью системы SAPFOR необходимые преобразования, которые не приводят к снижению производительности и при этом повышают доступный уровень параллелизма. Можно отметить, что преобразования производятся на уровне исходного кода и не требуют детального изучения параллельных архитектур и языков параллельного программирования.

В сочетании с ранее выполненными работами [13, 16] система SAPFOR может выполнять распараллеливание в модели DVMH на вычислительные системы в разной конфигурации, используя узлы кластера, а также графические ускорители и многоядерные процессоры внутри узла.

Таким образом, системы SAPFOR и DVM могут значительно сократить усилия, необходимые для распараллеливания программ, и позволяют задействовать все доступные внутри узла ресурсы (многоядерные процессоры и графические ускорители). Мы надеемся, что данный подход должен помочь эффективной разработке и оптимизации масштабируемых программ для суперкомпьютеров.

Литература

1. Штейнберг Б.Я., Штейнберг О.Б. Преобразования программ – фундаментальная основа создания оптимизирующих распараллеливающих компиляторов // Программные системы: теория и приложения, 2021, 12:1(48), с. 21–113. DOI: 10.25209/2079-3316-2021-12-1-21-113

2. Czarnul, P., Proficz, J., Drypczewski, K. Survey of methodologies, approaches, and challenges in parallel programming using high-performance computing systems // Scientific Programming, vol. 2020, P. 1058-–9244, 2020. DOI: 10.1155/2020/4176794

3. SYCL. URL: https://sycl.tech/

4. Celerity. High-level C++ for Accelerator Clusters. URL: /https://celerity.github.io/

5. Murai, H., Nakao, M., Shimosaka, T., Tabuchi, A., Boku, T., Sato, M. XcalableACC - a Directive-based Language Extension for Accelerated Parallel Computing // Supercomputing'14 poster, New Orleans, LA, USA, Nov. 2014

6. Konovalov, N.A., Krukov, V.A, Mikhajlov, S.N., Pogrebtsov, A.A. Fortan DVM: a Language for Portable Parallel Program Development // Programming and Computer Software. Vol. 21, No. 1. 1995. P. 35–38.

7. Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Сазанов Ю.Л. Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика, 2012. № 18(277). С. 82-92.

8. Бахтин, В.А., Колганов, А.С., Крюков, В.А., Поддерюгина, Н.В., Притула, М.Н. Методы динамической настройки DVMH-программ на кластеры с ускорителями // Суперкомпьютерные дни в России : Труды международной конференции, Москва, 28–29 сентября 2015 года, М.: Изд-во МГУ, 2015, С. 257-268), 2015. С. 257–268.

9. Hwu, W.-m., Ryoo, S., Ueng, S.-Z., Kelm, J.H., Gelado, I., Stone, S.S., Kidd, R.E., Baghsorkhi S.S., Mahesri, A.A., Tsao, S.C., Navarro, N., Lumetta, S.S., Frank, M.I., Patel, S.J.Implicitly parallel programming models for thousand-core microprocessors. // Proceedings of the 44th annual Design Automation Conference (DAC '07), ACM, New York, NY, USA. 2007. P. 754–759. DOI 10.1145/1278480.1278669

10. Baghdadi, R. et al. PENCIL: A Platform-Neutral Compute Intermediate Language for Accelerator Programming // 2015 International Conference on Parallel Architecture and Compilation (PACT). 2015. P. 138–149 DOI: 10.1109/PACT.2015.17.

11. Kruse, M. Introducing Molly: Distributed Memory Parallelization with LLVM // CoRR, vol. abs/1409.2088. 2014 DOI: https://doi.org/10.48550/arXiv.1409.208810.48550/arXiv.1409.2088

12. Vandierendonck, H., Rul, S., De Bosschere, K. The Paralax Infrastructure: Automatic Parallelization with a Helping Hand // 2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT). 2010. P. 389–399.

13. Катаев, Н. А., Колганов, А. С. Дополнительное распараллеливание MPI программ с помощью системы SAPFOR // Вычислительные методы и программирование. 2021. 22. С. 239–251. DOI 10.26089/NumMet.v22r415

14. Клинов М.С., Крюков В.А. Автоматическое распараллеливание Фортран-программ. Отображение на кластер // Вестник ННГУ, 2009. №2. С. 128–134.

15. Бахтин В.А., Бородич И.Г., Катаев Н.А., Клинов М.С., Ковалева Н.В., Крюков В.А., Поддерюгина Н.В. Диалог с программистом в системе автоматизации распараллеливания САПФОР // Вестник ННГУ, 2012. № 5(2). С 252-245.

16. Kataev, N. LLVM Based Parallelization of C Programs for GPU //Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2020. Communications in Computer and Information Science, vol 1331. Springer, Cham, 2020. P. 436–448. DOI: 10.1007/978-3-030-64616-5_38

17. Kataev, N. Interactive Parallelization of C Programs in SAPFOR // Scientific Services & Internet 2020. CEUR Workshop Proceedings, Vol. 2784, 2020. P. 139–148.

18. Kataev, N. Application of the LLVM Compiler Infrastructure to the Program Analysis in SAPFOR // Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2018.

Communications in Computer and Information Science, Vol. 965,. Springer, Cham, 2018. P. 487–499. DOI: 10.1007/978-3-030-05807-4_41

19. Kataev, N., Smirnov, A., Zhukov A. Dynamic data-dependence analysis in SAPFOR // CEUR Workshop Proceedings, Vol. 2543, 2020, P. 199–208. DOI: 10.20948/abrau-2019-62

20. NAS Parallel Benchmarks. URL: https://www.nas.nasa.gov/publications/npb.html (Дата обращения: 8 апреля 2021).

21. Amarasingh, S. P., Lam, M. S. Communication Optimization and Code Generation for Distributed Memory Machines // PLDI '93: Proceedings of the ACM SIGPLAN 1993 conference on Programming language design and implementation. 1993 P. 126–138. DOI: 10.1145/155090.155102

22. Zima, H., Bast H., Gerndt, M. SUPERB: A tool for semi-automatic MIMD/SIMD parallelization // Parallel Comput. vol. 6. 1988. P. 1–18. DOI: 10.1016/0167-8191(88)90002-6

23. Grosser, T., Groesslinger, A., Lengauer. C. Polly — performing polyhedral optimizations on a low-level intermediate representation // Parallel Processing Letters, Vol 22(04), 2012. DOI: 10.1142/S0129626412500107

24. Lattner, C., Adve, V.: LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation // Proc. of the 2004 International Symposium on Code Generation and Optimization (CGO'04). Palo Alto, California, 2004. DOI: 10.1109/CGO.2004.1281665

25. Гервич, Л.Р., Кравченко, Е.Н., Штейнберг, Б.Я., Юрушкин, М.В. Автоматизация распараллеливания программ с блочным размещением данных // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. - Новосибирск, 2015. – Т.18, № 1. С–41-53

26. Bondhugula, U., Hartono, A., Ramanujam, J., Sadayappan, P. A practical automatic polyhedral parallelizer and locality optimizer // SIGPLAN Notices, 43(6), 2008. P. 101–113. DOI: 10.1145/1375581.1375595

27. Bondhugula, U. Compiling Affine Loop Nests for Distributed-Memory Parallel Architectures // SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 2013. P. 1–12. DOI: 10.1145/2503210.2503289

28. Banerjee, P., et al. The Paradigm Compiler for Distributed-Memory Multicomputers // Computer, Vol. 28, Issue 10, IEEE Oct 1995, P. 37-47. DOI: 10.1109/2.467577

29. Гетерогенный кластер K10. URL: https://www.kiam.ru/MVS/resourses/k10.html

Валидационные расчеты задач гемодинамики с использованием программного комплекса FlowVision в режиме распараллеливания

М.Д. Калугина[1], В.С. Каширин[1], А.И. Лобанов[1,2]

[1]ООО «ТЕСИС», [2]Московский физико-технический институт (национальный исследовательский университет)

Проведены расчеты тестовой задачи, связанной с моделированием течения в идеализированном медицинском устройстве, в программном комплексе FlowVision. Расчеты проводились для ламинарного, турбулентного и переходного режимов течения. Исследована масштабируемость задачи. На основе решения тестовой задачи делается вывод, что программный комплекс FlowVision может быть использован для решения задач гемодинамики.

Ключевые слова: параллельные вычисления, программный комплекс FlowVision, гемодинамика, валидационные расчеты.

1. Введение

Задачи вычислительной гемодинамики приобрели значительный интерес в связи с возможностью перейти к созданию пациент-ориентированных (персонализированных) математических моделей кровообращения [1] и создания искусственных органов [2]. Современные математические модели гемодинамики представляют собой комбинацию математических моделей движения жидкости в сложном сосудистом дереве [1] и крупных сосудах сложной формы. Отметим, что оба класса задач требуют значительных вычислительных ресурсов и практически не могут быть реализованы без применения современных вычислительных технологий и использования мощных многопроцессорных вычислительных систем.

Возникают дополнительные требования к программному обеспечению для математического моделирования сложных задач гемодинамики. Так как системы уравнений существенно нелинейны (даже в случае использования системы Навье-Стокса для моделирования кровотока, учет реалистичной реологии крови приводит к появлению дополнительных нелинейностей), то получение точных решений для задач такого класса для областей сложной геометрии практически невозможно. Точные решения с учетом реологических уравнений могут быть получены только для упрощенных одномерных постановок [3].

Возросшее число работ по вычислительной гемодинамике выдвигает требования к возможностям программных комплексов. Эти требования существенны и для исследовательских проектов, и для задач проектирования новых устройств или препаратов. Для оценки точности и производительности программного обеспечения в настоящее время используются две задачи, предложенные американской Food and Drug Administration (FDA) [4, 5]. Это задачи о сопле (nozzle) и насосе для перекачки крови. Для решения практических задач гемодинамики, как правило, применяются те комплексы, которые показывают хорошие результаты на задачах FDA, что подтверждается соответствующими публикациями.

Задача о сопле широко используется для тестирования (бенчмаркинга) как коммерческого программного обеспечения, так и исследовательских программных комплексов. В литературе чаще встречаются результаты тестирования программных комплексов, основанных на методах конечных элементов. В [6] приведены результаты исследования задачи о сопле на основе использования комплекса мультифизического моделирования COMSOL.

Примерами программ для моделирования гидродинамики свободного доступа, основанными на методах конечных элементов, могут служить Feel++ [7], результаты расчетов задачи на его основе приведены в [8].

Некоторые системы математического моделирования задач гидродинамики, используемые для гемодинамических расчетов, специально разработаны для реализации на многопроцессорных вычислительных комплексах. Оценки решения задачи FDA с использованием солвера NEK 5000, основанного на методе спектральных элементов (вариант метода конечных элементов с базисом из кусочно-полиномиальных функций высокой степени), приведены в [9]. Выполнены оценки применимости параллельного комплекса LS-DYNA для данных задач [10]. Отметим, что программный комплекс LS-DYNA в настоящее время является частью популярной системы моделирования ANSYS. Использование программы ANSYS-FLUENT для решения задачи о сопле описано в [11].

В данной статье для решения тестовой задачи FDA использован программный комплекс FlowVision.

2. Характеристики программного комплекса FlowVision

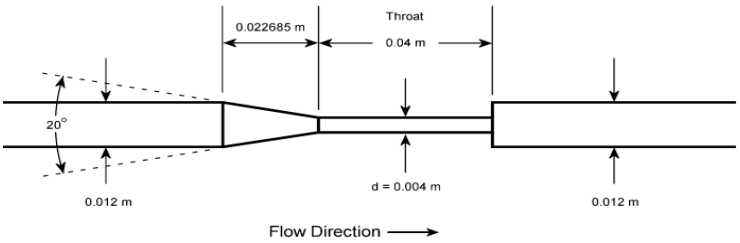
Программный комплекс вычислительной аэро- и гидродинамики FlowVision [12] предназначен для проведения математического моделирования различных физических процессов. Аппроксимация основных уравнений движения жидкости в форме Навье-Стокса в ПК основана на конечно-объемном подходе. Расчетная сетка во FlowVision является декартовой, ячейки сетки представляют собой гексаэдры. Во FlowVision имеется автоматический построитель неструктурированной сетки с возможностью ее локальной адаптации до указанного уровня на любой поверхности и в любом объеме расчетной области. Для моделирования характеристик пограничного слоя на стенке в функционал ПК заложены пристеночные функции, позволяющие пользователю без подробного разрешения пространства расчетной сеткой, получать достаточно точные результаты [13]. Для учета турбулентности течения во FlowVision реализовано 7 моделей турбулентности, которые можно использовать в низко- и высокорейнольдсовых расчетах [14]. Также в ПК реализован двухуровневый параллелизм, позволяющий эффективно проводить расчеты на компьютерах, имеющих распределенную и общую память одновременно.

3. Постановка задачи FDA

В тестовых задачах FDA рассматривается небольшое эталонное сопло, напоминающее упрощенные идеализированные медицинские устройства для переноса крови. Насадка обладает аналогичными характеристиками с наборами для гемодиализа, катетерами, иглами для подкожных инъекций, шприцами и т.д. Экспериментальные данные были получены в определенных сечениях сопла (рис. 1).

3.1 Модель сопла

Модель осесимметричного сопла имеет узкую горловину длиной 0,04 м с соединительным конусом на одном конце горловины и ступенчатым изменением диаметра на другом конце (рис. 1).



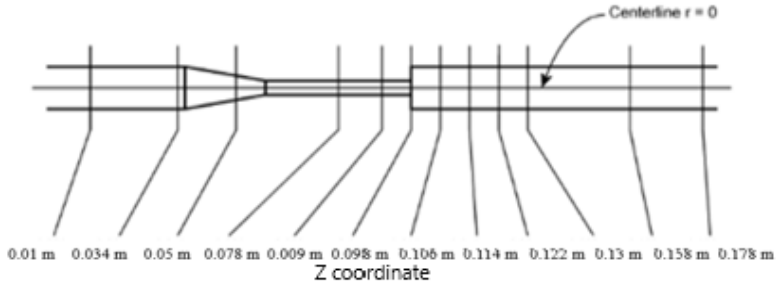


Рис. 1. Геометрические параметры сопла и сечения, в которых определялись параметры течения [5]

Предложенная FDA геометрия разработана с учетом ускоряющего и замедляющего потоков, изменения напряжения и скорости сдвига, а также рециркуляционного потока.

3.2 Численное моделирование

В программном комплексе (ПК) FlowVision в соответствии с начальными экспериментальными данными [5] были проведены трехмерные расчеты, соответствующие четырем числам Рейнольдса в горловине Re_t. Числа Рейнольдса выбирались таким образом, чтобы охватить ламинарный, переходный и турбулентный режимы. В таблице 1 представлены значения объемных расходов Q и соответствующих им чисел Рейнольдса.

Таблица 1. Объемные расходы и числа Рейнольдса

Q, m^3/s	Re_t
$5.21 \cdot 10^{-6}$	500
$2.08 \cdot 10^{-5}$	2000
$3.64 \cdot 10^{-5}$	3500
$6.77 \cdot 10^{-5}$	6500

Также при различных числах Рейнольдса использовались различные модели турбулентности и пристеночные функции (табл. 2).

Таблица 2. Числа Рейнольдса, модели турбулентности и пристеночные функции

Re_t	Модель турбулентности	Пристеночные функции
500	нет	равновесные
2000	SST	равновесные/без них
3500	SST	равновесные/без них
6500	SST, k–ε	равновесные

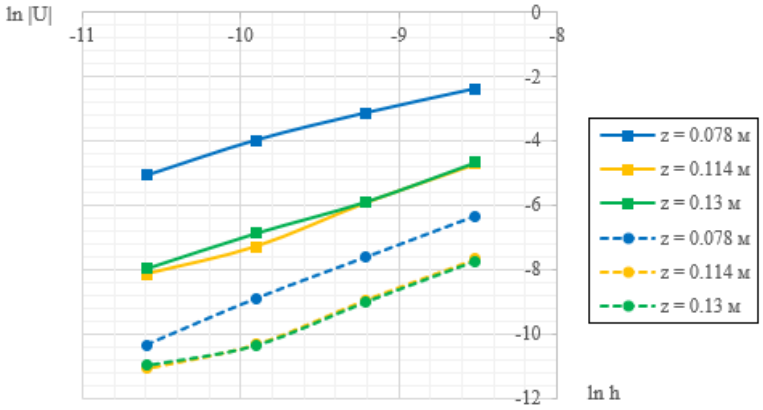
Поскольку при $Re_t = 500$ режим течения является полностью ламинарным, использование моделей турбулентности не потребовалось. Для остальных чисел Рейнольдса применялась SST-модель турбулентности, а при $Re_t = 6500$ использовалась также k–ε модель [13]. Стоит отметить, что для переходных режимов проводились расчеты как без использования пристеночных функций, т.е. пограничный слой полностью разрешался сеткой, так и с применением равновесных пристеночных функций.

Кровь упрощенно была определена как ньютоновская жидкость с плотностью и динамической вязкостью 1056 кг/м3 и 0,0035 Н·с/м2 соответственно. На входной границе сопла задавался параболический профиль осевой скорости, представляющий полностью развитое стационарное течение в круглой трубе.

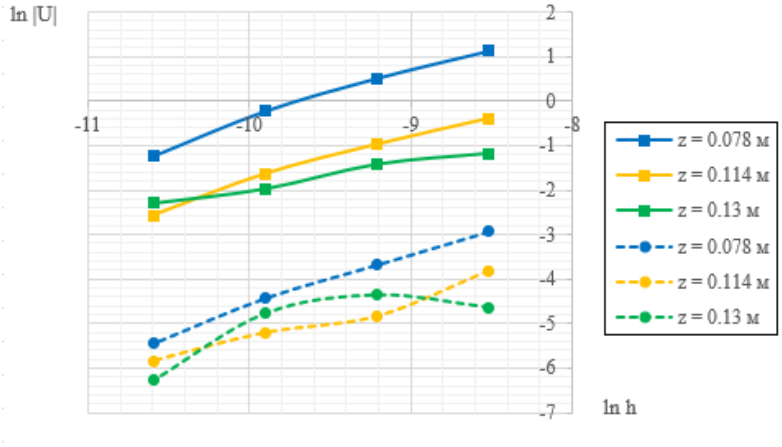
Для ламинарного течения и течения с развитой турбулентность было проведено исследование сеточной сходимости. Были рассчитаны значения «сильной» (аналог нормы Чебышева) и «слабой» (аналог нормы L_2) нормы для отклонения значения осевой скорости от значений, полученных на самой подробной сетке, в сечениях с координатами z = 0,078; 0,114 и 0,13 м в зависимости от размера ячейки h (м) вдоль оси z (рис. 2). На графиках «сильная» норма показана сплошной линией, «слабая» - пунктирной. Графики приведены в двойном логарифмическом масштабе, тангенс угла наклона примерно соответствует порядку сходимости для задачи. Отметим, что

численное значение тангенса оказалось практически не зависящим от конкретного выбора норм, что характерно для гладких решений.

Также для сравнения были использованы значения максимальной скорости в сопле U_{z_max} при различных уровнях адаптации сетки N = 0, 1, 2 и т.д, обеспечивающей измельчение ячеек расчетной сетки путем их разбиения пополам по всем направлениям (рис. 3). Наблюдается выход построенного графика на асимптоту при втором уровне адаптации сетки, т.е. дальнейшее измельчение сетки не приводит к существенному изменению характерной величины.

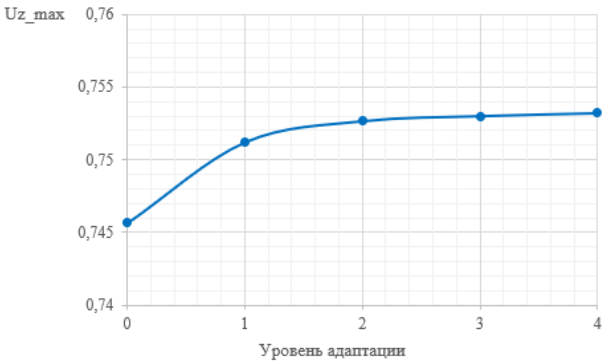


а)

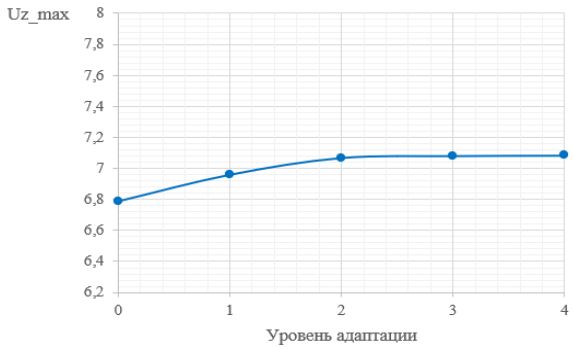


б)

Рис. 2. Зависимость «сильной» и «слабой» нормы от размера ячейки в двойном логарифмическом масштабе при: а) $Re_t = 500$; б) $Re_t = 6500$



а)

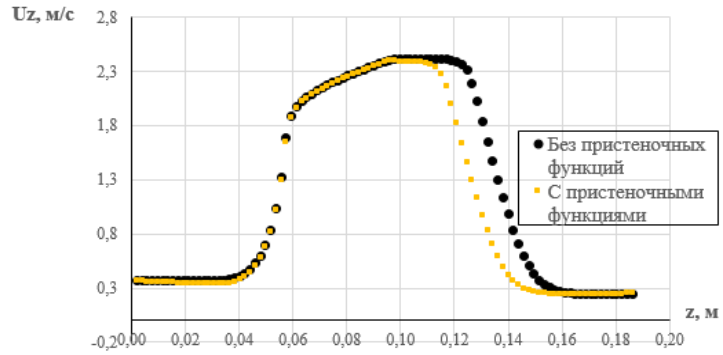


б)

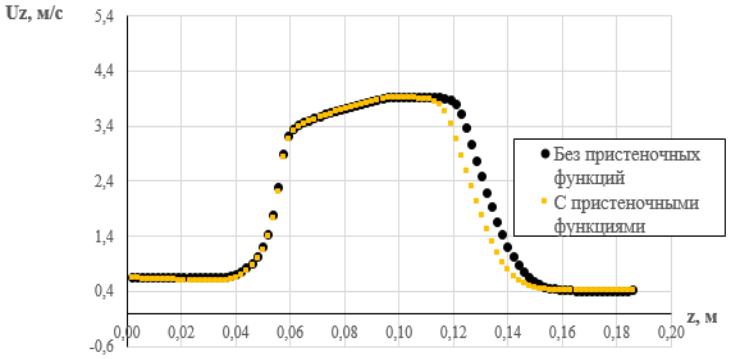
Рис. 3. Исследование сеточной сходимости: а) зависимость максимальной скорости в сопле от уровня адаптации сетки при $Re_t = 500$; б) зависимость максимальной скорости в сопле от уровня адаптации сетки при $Re_t = 6500$.

4. Результаты расчетов

В результате расчетов были получены значения осевой скорости U_z. При $Re_t = 2000$ и 3500 дополнительно проведены расчеты без пристеночных функций. Для этих расчетов сетка была построена так, чтобы безразмерный параметр Y^+ — первый пристеночный шаг по нормали к стенке в координатах закона стенки $\frac{Y \cdot V^*}{\nu}$, где $V^* = \sqrt{\frac{\tau_w}{\rho}}$ — динамическая скорость, имел значение меньше 1 (рис. 4).



а)

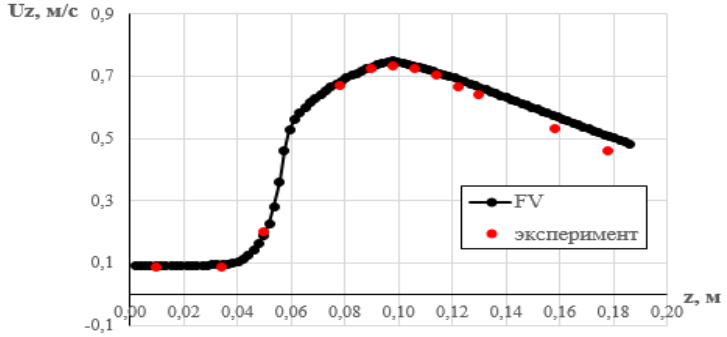


б)

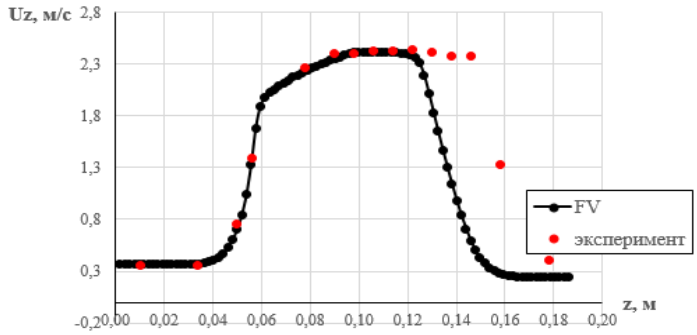
Рис. 4. Зависимость осевой скорости от координаты z при расчете с пристеночными функциями и без них: а) $Re_t = 2000$; б) $Re_t = 3500$.

Расчеты при $Re_t = 500$ и 6500 проводилось на сетке с количеством ячеек около 700000. Для расчета при $Re_t = 2000$ и 3500 были выбраны сетки, соответствующие $Y^+ < 1$ (пристеночные функции не использовались).

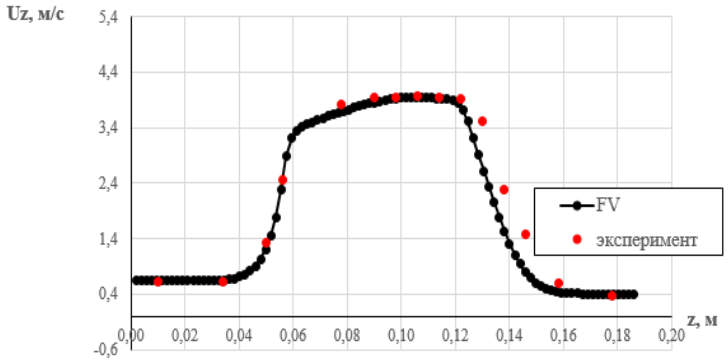
На рис. 5 показаны зависимости осевой скорости U_z от координаты z, полученные в программном комплексе FlowVision и экспериментально [5] (рис. 5).



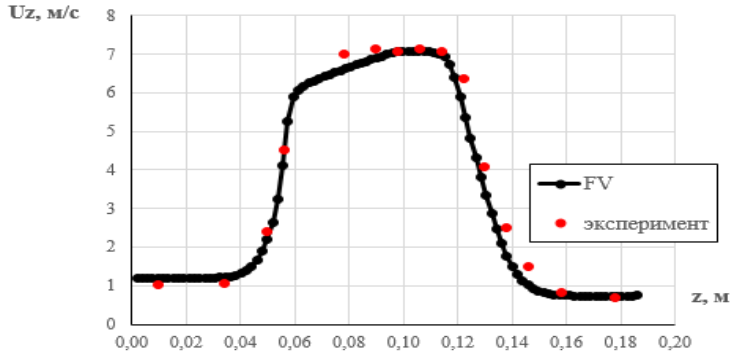
а)



б)



в)



г)

Рис. 5. Зависимость осевой скорости от координаты z: а) $Re_t = 500$; б) $Re_t = 2000$; в) $Re_t = 3500$; г) $Re_t = 6000$.

Наилучшее соответствие с экспериментальными данными наблюдается при полностью ламинарном течении (рис. 5, а) и течении с развитой турбулентностью (рис. 5, в, г). При $Re_t = 6500$

после резкого расширения сопла возникает вихревое течение, образуются две зоны возвратного течения в области с минимальной скоростью (рис. 6).

Численное моделирование ламинарно-турбулентного перехода всегда представляет собой достаточно сложную задачу, поэтому для $Re_t = 2000$ в результатах появлялись некоторые расхождения с экспериментальными данными. Отличия были не больше полученных в других пакетах. Результаты численного моделирования, полученные в других программных комплексах [5], обозначены на рис. 7 как CFD_1 - CFD_4. При этом результаты расчета для $Re_t = 2000$ и 3500 без пристеночных функций лучше соответствуют экспериментальным данным. При ламинарном течении после выхода потока из горловины скорость вдоль осевой линии сопла уменьшается постепенно, в остальных случаях происходит резкое снижение скорости.

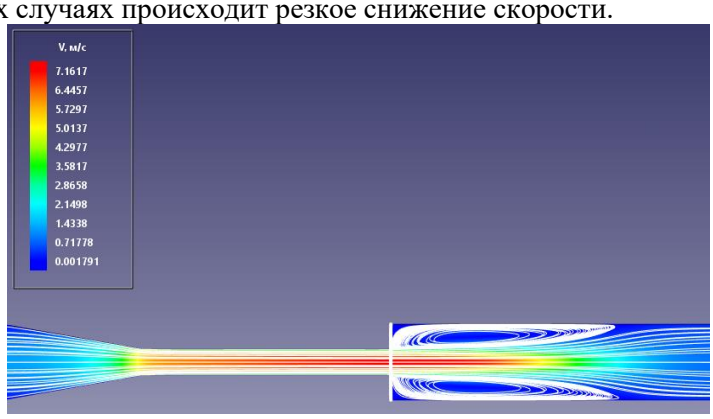


Рис. 6. Распределение скорости и линий тока в сопле

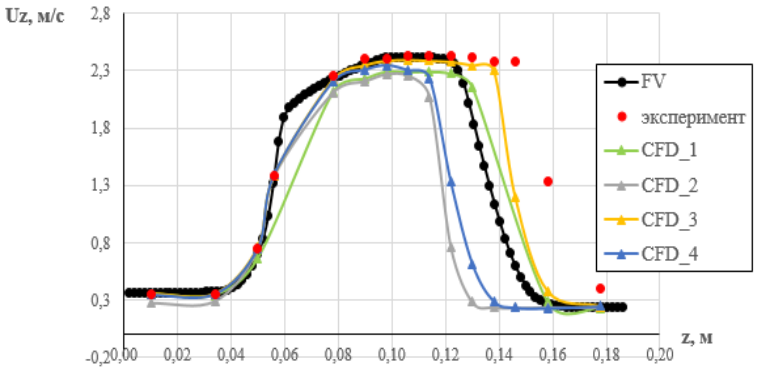
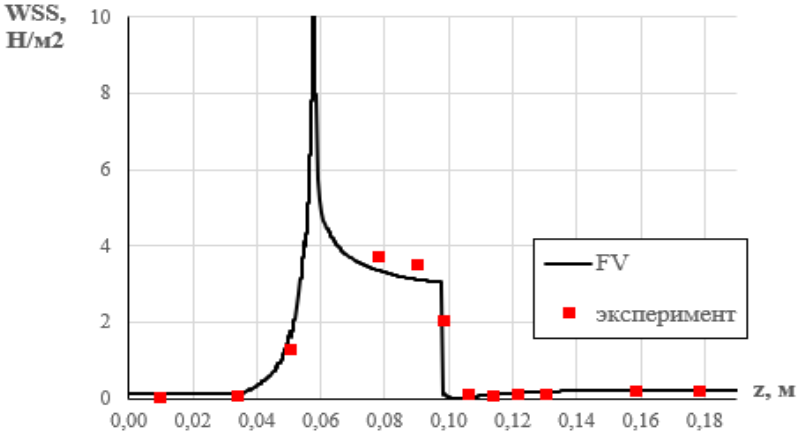
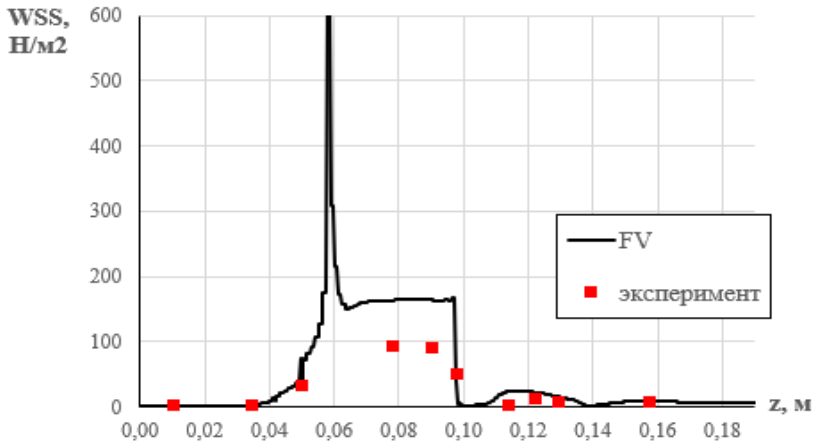


Рис. 7. Зависимость осевой скорости от координаты z при $Re_t = 2000$.

Помимо зависимости осевой скорости U_z от координаты z для чисел Рейнольдса $Re_t = 500$ и $Re_t = 6500$ было получено распределение сдвиговых напряжений на стенке (Wall shear stress - WSS) сопла, которое сравнивалось с данными эксперимента (рис. 8).



а)



б)

Рис. 8. Распределение сдвиговых напряжений на стенке при: а) $Re_t = 500$; б) $Re_t = 6500$.

Наилучшее соответствие расчетных данных и экспериментальных получено при ламинарном режиме течения жидкости. Наибольшее рассогласование соответствует горловине сопла.

Поскольку часть расчетов проводилась на вычислительном кластере HPC4, исследовалась масштабируемость задачи [15].

5. Характеристики параллельности

В многофункциональный вычислительный комплекс (МФК) НИЦ «Курчатовский институт» входит несколько высокопроизводительных кластеров. Часть представленных расчетов производилась на кластере HPC4, состоящем из 364 вычислительных узлов по 2 процессора Intel Xeon E5-2680 v3 и 23 вычислительных узлов по 2 процессора Intel Xeon E5-2680 v3 и по 3 ускорителя вычислений Supermicro NVIDIA Tesla K80 24GB GDDR5 PCIe 3.0 [16]. В качестве оценки эффективности распараллеливания расчетов использовалось отношение *S* (относительное ускорение) физического времени счета одного шага, измеряемого в секундах, полученного при расчете на 1 узле к времени счета одного шага, полученного при расчете на нескольких узлах *n*. Для всех исследуемых режимов расчет запускался на продолжение с определенного шага, до которого течение жидкости уже успело установиться. На рис. 9 представлена зависимость *S(n)* для расчетов при числах Рейнольдса $Re_t > 500$.

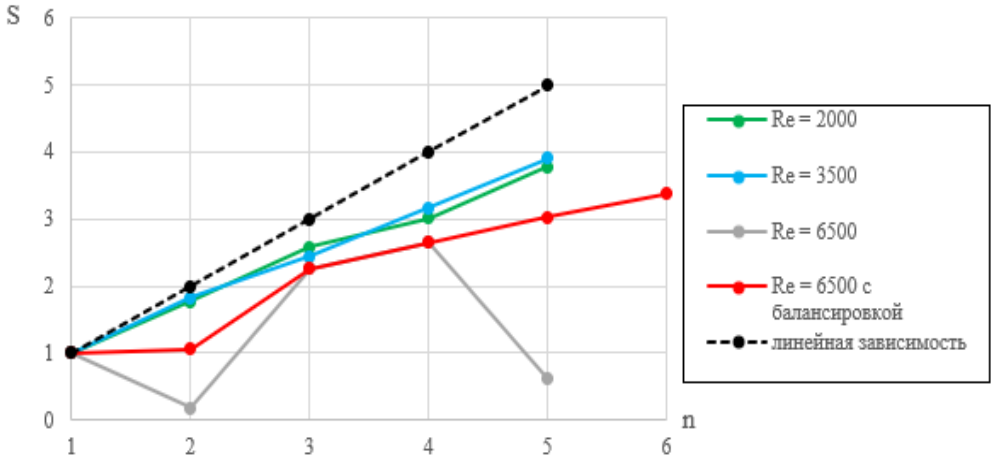


Рис. 9. Масштабируемость при различных числах Рейнольдса.

Зависимости *S(n)* для расчетов при числах Рейнольдса 2000 и 3500 близки к линейной. При расчете на 2 узлах для $Re_t = 6500$ возникает значительный дисбаланс загрузки процессоров, т.е. такой режим запуска является неоптимальным. Использование 5 узлов также ведет к возникновению дисбаланса, который можно объяснить наличием слишком большого количества

исполнителей для используемой сетки. Однако дисбаланс, возникающий из-за большого времени обменных операций MPI, можно снизить за счет применения специальной технологии FlowVision «Динамическая балансировка». На рис. 9 показано, что использование динамической балансировки при расчете на двух и пяти узлах позволяет значительно снизить время вычисления шага и улучшить масштабируемость.

6. Выводы

В работе представлено решение тестовой задачи FDA в программном комплексе FlowVision. Получены значения осевой скорости в зависимости от координаты z при ламинарном, переходном и турбулентном режимах течения. Так же проведено исследование масштабируемости при расчете на вычислительном кластере HPC4. Существенному увеличению производительности вычислений в случаях, когда особенности постановки расчетной задачи приводят к дисбалансу загрузки процессоров, способствует специальная технология FlowVision «Динамическая балансировка».

Решение тестовой задачи показало, что программный комплекс FlowVision позволяет получать результаты, имеющие хорошее согласование с экспериментом для задачи FDA о сопле. Хорошее совпадение тестовых результатов расчетов с гемодинамическими экспериментами в искусственной системе открывает возможность решать более сложные задачи гемодинамики.

Литература

1. Yu. Vassilevski, M. Olshanskii, S.Symakov, A.Kolobov, A. Danilov Personalized computational hemodynamics // Models, methods and application for vascular surgery and antitumor therapy. Academic Press. 2020. 270 p.
2. Chen, Z., Fan, Y., Deng, X. and Xu, Z. A New Way to Reduce Flow Disturbance in Endovascular Stents: A Numerical Study. Artificial Organs. 2011. Vol. 35. P. 392-397. https://doi.org/10.1111/j.1525-1594.2010.01106.x.
3. Krivovichev G.V. Comparison of inviscid and viscid one-dimensional models of blood flow in arteries // Applied Mathematics and Computation. Elsevier, 2022. Vol. 418. P. 126856 https://doi.org/10.1016/j.amc.2021.126856.
4. Computational Fluid Dynamics Round Robin Study. URL: https://ncihub.org/wiki/FDA_CFD (дата обращения: 23.04.2022).
5. S.F.C Stewart et al. Assessment of CFD Performance in Simulations of an Idealized Medical Device: Results of FDA's First Computational Interlaboratory Study // Cardiovascular Engineering and Technology. 2012. Vol. 3(2) P. 139-160. DOI 10.1007/s13239-012-0087-5.
6. Blood Damage Modeling of FDA Benchmark Nozzle. URL: https://www.comsol.ru/paper/blood-damage-modeling-of-fda-benchmark-nozzle-93171 (дата обращения: 21.04.2022).
7. C. Prud'homme, V. Chabannes, V. Doyeux, M. Ismail, A. Samake and G. Pena. Feel++: A Computational Framework for Galerkin Methods and Advanced Numerical Methods // ESAIM: Proceedings 2012. Vol. 38. P. 429–455. DOI:10.1051/proc/201238021.
8. V. Chabannes, C. Prud'Homme, M. Szopos, R. Tarabay. High order finite element simulations for fluid dynamics validated by experimental data from the FDA benchmark nozzle model // 5th International Conference on Computational and Mathematical Biomedical Engineering, CMBE 2017, April 10-12, 2017, Pittsburgh, PA, United States.
9. Abad, Nour & Vinuesa, Ricardo & Schlatter, Philipp & Andersson, Magnus & Karlsson, Matts. Simulation strategies for the FDA nozzle using Nek5000 Simulation strategies for the Food-and-Drug-Administration nozzle using Nek5000 // AIP Advances. 2020. Vol. 10 (2). P. 025033. DOI: 10.1063/1.5142703.

10. Huang, Chien-jung, Iñaki Çaldichoury, Facundo Del Pin and Rodrigo Paz. CFD Validations with FDA Benchmarks of Medical Devices Flows // 15th International LS-DYNA Users Conference. 2018.

11. Zmijanovic V., Mendez S., Moureau V., Nicoud F. About the Numerical Robustness of Biomedical Benchmark Cases: Interlaboratory FDA's Idealized Medical Device // International Journal for Numerical Methods in Biomedical Engineering. 2016. Vol. 33 (1). P. 1-19. DOI: 10.1002/cnm.2789.

12. А. А. Аксенов. FlowVision: индустриальная вычислительная гидродинамика // Компьютерные исследования и моделирование. 2017. Т. 9. №. 1. С. 5–20. DOI: 10.20537/2076-7633-2017-9-5-20.

13. С.В. Жлуктов, А.А. Аксенов. Пристеночные функции для высокорейнольдсовых расчетов в программном комплексе FlowVision // Компьютерные исследования и моделирование. 2015. Т. 7. № 6. С. 1221−1239.

14. С. В. Жлуктов, А. А. Аксенов, Д. В. Савицкий. Высокорейнольдсовые расчеты турбулентного теплопереноса в программном комплексе FlowVision // Компьютерные исследования и моделирование. 2018. Т. 10. Вып. 4. С. 461–481. DOI: 10.20537/2076-7633-2018-10-4-461-481.

15. В.С. Акимов, Д.П. Силаев, А.С. Симонов, А.С. Семенов. Исследование масштабируемости FlowVision на кластере с интерконнектом Ангара // Вычислительные методы и программирование. 2017. Т. 18. Вып. 4. С. 406-414.

16. Объединенный вычислительный кластер. URL: http://comp.nrcki.ru/pages/main/12530/12546/index.shtml (дата обращения: 21.04.2022)

Исследование причин аварийного завершения заданий на суперкомпьютере «Ломоносов-2»*

С.И. Соболев

Научно-исследовательский вычислительный центр МГУ имени М.В.Ломоносова

Предложен подход к анализу случаев аварийного завершения заданий, выполняемых на суперкомпьютере «Ломоносов-2». Подход нацелен на поиск первопричин этих ситуаций путем выявления возможных связей сбоев приложений с событиями, происходящими в это время как на вычислительных узлах, задействованных для выполнения задания, так и на суперкомпьютере в целом. В качестве источников информации о событиях рассматриваются системные журналы вычислительных узлов и служебных серверов суперкомпьютера. Представлены первые результаты анализа, показавшие перспективность предложенного подхода.

Ключевые слова: суперкомпьютер, Ломоносов-2, Slurm, InfluxDB, задания, приложения, аварийное завершение

1. Введение

Ежемесячно на суперкомпьютере «Ломоносов-2», установленном в Московском государственном университете имени М.В.Ломоносова [1], запускается от трех до десяти тысяч заданий. После постановки пользователем задания в очередь на выполнение СУПЗ Slurm [2] присваивает ему статус, который меняется в зависимости от текущего состояния задания: ожидает ресурсов, запускается, выполняется, завершается и т.д. В случае успешного выполнения и корректного завершения задания оно получает статус COMPLETED. Однако не все задания завершаются успешно. Задание может быть отменено самим пользователем или администратором, и в этом случае оно получает статус CANCELLED. Задание также может завершиться аварийно; таким заданиям Slurm присваивает статус FAILED. В данной работе предпринята попытка проанализировать случаи аварийного завершения заданий, выявить возможную связь с событиями, происходящими в это время как на вычислительных узлах, задействованных для выполнения задания, так и на суперкомпьютере в целом, и тем самым найти первопричины подобных сбоев.

На рис. 1 приведена статистика по соотношению задач, завершившихся со статусом FAILED (отмечены красным), и задач, завершившихся с иными статусами (отмечены зеленым), по каждому месяцу в течение 2021 года.

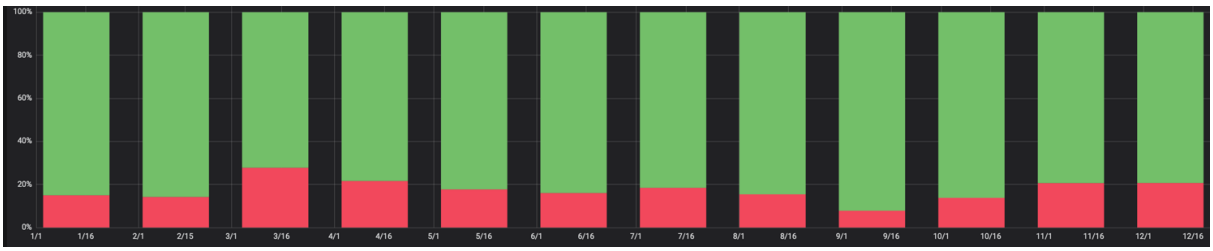


Рис. 1. Соотношение числа заданий, завершившихся со статусом FAILED, и заданий, завершившихся с другими статусами, за каждый месяц 2021 г.

На графике видно, что число аварийно завершившихся заданий не сильно изменяется от месяца к месяцу и составляет около 20% от общего числа заданий в среднем за год. Если рассматривать соотношение числа заданий, завершившихся аварийно (со статусом FAILED) и успешно

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-07-00818.

(со статусом COMPLETED), то его среднее значение за год составляет уже 27%. Это проиллюстрировано на графике на рис. 2, где доля успешно завершившихся заданий отмечена желтым. Если же говорить об абсолютных значениях, то число аварийно завершившихся заданий колеблется от нескольких сотен до почти двух тысяч в месяц. Эти данные приведены на графике на рис. 3.

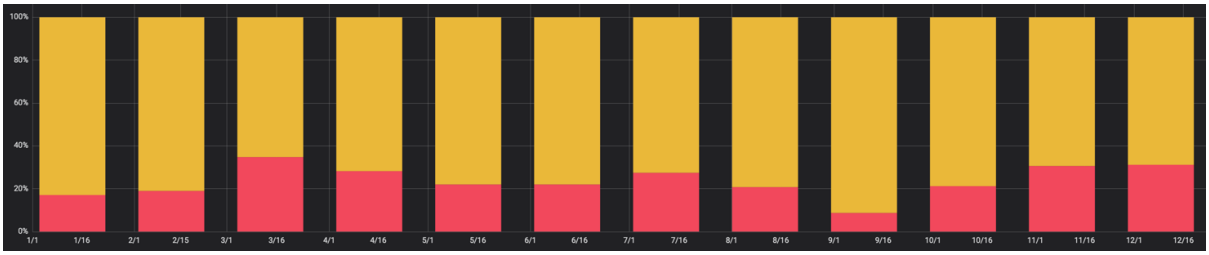


Рис. 2. Соотношение числа заданий, завершившихся со статусом FAILED, и успешно завершившихся заданий, за каждый месяц 2021 г.

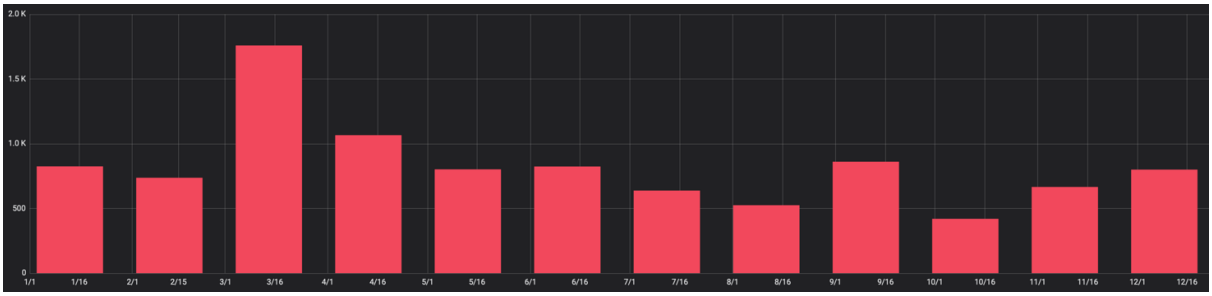


Рис. 3. Число FAILED-заданий за каждый месяц 2021 г.

Таким образом, доля завершившихся заданий оказывается достаточно существенной в общем потоке заданий суперкомпьютера, а их количество предоставляет обширный материал для анализа.

Данное исследование отчасти было вдохновлено работой [3], описывающей анализ потока событий суперкомпьютера IBM BlueGene/L. Представляемые в работе методы и результаты были получены в рамках выполнения проекта по разработке методов сохранения, реконструкции и анализа структурно-функциональных свойств суперкомпьютерных систем

2. Источники данных

В качестве источников данных о событиях используются системные журналы (syslog) вычислительных узлов и служебных серверов суперкомпьютера «Ломоносов-2». Было реализовано сохранение записей из системных журналов всех вычислительных узлов и служебных серверов суперкомпьютера Ломоносов-2 в базу данных InfluxDB [4]. На узлах и серверах была настроена отправка системных журналов на один из служебных серверов суперкомпьютера. На нем для приема журналов был установлен и настроен сервис rsyslog [5]. За один час в базе данных InfluxDB сохраняется в среднем 200 000 записей из системных журналов суперкомпьютера, из них 81% – записи из журналов вычислительных узлов, а оставшиеся 19% – записи из журналов служебных серверов и размещенных на них виртуальных машин.

Изначально в тестовом режиме журналы с узлов и серверов сохранялись в виде текстовых файлов. Этот режим продемонстрировал общую работоспособность конструкции, однако был крайне неудобен для практического использования и проведения исследований – поиск по текстовым файлам происходил достаточно медленно, а даже простейший поиск по нескольким файлам необходимо было организовывать вручную. Кроме того, на некоторых узлах и серверах отличались настройки системного времени (временн*ая* зона могла быть не прописана, некорректно

работала синхронизация с NTP-сервером и т.д.), из-за чего одновременно происходящие на разных серверах или узлах события могли иметь значительно отличающиеся временны*ы*е метки. Поэтому сохранение событий в СУБД было оправданным. Это было реализовано путем перенаправления (посредством того же сервиса rsyslog) всех приходящих системных журналов в агент мониторинга Telegraf [6]. В агенте был включен модуль inputs.syslog, и после его активации записи системных журналов стали отправляться в InfluxDB. При сохранении записей в InfluxDB исходная временная метка отбрасывалась, вместо нее проставлялось текущее системное время. Тем самым была решена проблема различающихся временны*ы*х меток.

В качестве альтернативы Telegraf как агрегатору и транслятору системных журналов был протестирован пакет fluentd [7]. Он обеспечивал аналогичную функциональность, однако создаваемые им структуры данных в InfluxDB отличались от таковых в Telegraf; как следствие, системных журналов «не видел» сервис визуализации данных Chronograf [8], разрабатываемый совместно с Telegraf и InfluxDB и оптимизированный для работы именно с этими пакетами. Кроме того, было принято решение минимизировать установку дополнительного программного обеспечения на достаточно сильно загруженный служебный сервер. По этой же причине не использовался известный «стек» ELK (ElasticSearch, Logstash, Kibana) [9], ориентированный на задачи хранения, обработки и визуализации событий из множественных журналов.

Для получения информации о заданиях, выполнявшихся на суперкомпьютере, используется система Job Digest [10]. В ее состав входит база данных, содержащая сведения о всех выполнявшихся на суперкомпьютере заданиях: идентификатор, имя учетной записи, командная строка, времена постановки в очередь, запуска и завершения, статус завершения задания и т.д.

Отметим, что далее в множество рассматриваемых заданий не включаются случаи их принудительного завершения со стороны Slurm – например, по превышению лимита доступного процессорного времени – или же со стороны пользователя или системного администратора. Такие задачи завершаются со статусом TIMEOUT («нехватка времени») или CANCELLED («прерванный») и потому могут быть легко исключены из анализа.

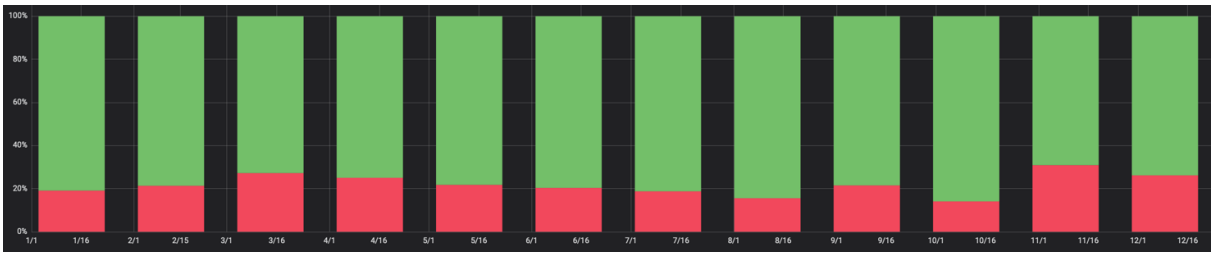


Рис. 4. Соотношение числа заданий из раздела test, завершившихся со статусом FAILED, и заданий, завершившихся с другими статусами, за каждый месяц 2021 г.

Введем еще одно ограничение – в список анализируемых заданий не включаются задания, выполненные на разделе test суперкомпьютера «Ломоносов-2». Время выполнения задач на нем ограничено 15 минутами, также существенно ограничено число одновременно находящихся в очереди заданий и т.д. Данный раздел предназначен прежде всего для тестирования приложений, в том числе в ходе их разработки (включая, например, выполнение заданий в рамках учебных практикумов), и реальные вычисления на нем не проводятся. Соотношение числа FAILED-заданий и заданий с иными статусами в этом разделе составляет 28%, что на 8% больше, чем аналогичное соотношение во всех остальных разделах. Соответствующая статистика по месяцам 2021 года приведена на графике на рис. 4. Очевидно, что для заданий из раздела test допустимо б*о*льшее число ошибок в кодах приложений, при этом в силу ограничения на максимальное время выполнения в них не успеют проявиться сбои, вероятные при длительном выполнении задач. Поэтому такие задания представляют гораздо меньший интерес для анализа. Добавим, что данные на рис. 1-3 приведены без учета раздела test.

3. Разработка инструментария

Для исследования возможных причин аварийного завершения заданий был разработан инструментарий, анализирующий события системных журналов задействованных приложениями

вычислительных узлах во временной окрестности остановки заданий. Основным объектом исследования стали задания, для которых СУПЗ Slurm выставила статус FAILED по завершению их выполнения. Такой статус присваивается заданию, если соответствующее ему приложение либо завершилось с ненулевым кодом возврата, либо завершилось аварийно. При этом ненулевой код возврата может свидетельствовать о сбое, обнаруженном самим приложением (типичная ситуация – корректная обработка нехватки памяти при попытке ее выделения), однако исходные тексты приложений нам недоступны, а коды возврата приложений не сохраняются. Вследствие этого отличить запланированный выход из приложения от его сбоя по иной причине не представляется возможным. Соответственно, FAILED-задания представляются в каком-то смысле «черными ящиками», поэтому исследованию подлежит их окружение – то, что происходит на суперкомпьютере в моменты их завершения.

Инструментарий, анализирующий события системных журналов вычислительных узлов во временной окрестности завершившихся заданий, состоит из 4-х программных компонентов (shell-скриптов), реализующих соответственно 4 этапа обработки данных. Рассмотрим их подробно.

Этап 1 – выборка событий из системных журналов (скрипт fails2log). Из базы данных Job Digest (СУБД PostgreSQL) выбираются задания по следующим критериям:

- задание было запущено в любом разделе суперкомпьютера, кроме раздела test;
- задание завершилось в течение суток, предшествующих дню выполнения выборки (при необходимости в качестве параметра скрипта можно задать другую дату);
- задание завершилось со статусом FAILED, NODE_FAIL или COMPLETED (почему в анализ включаются задания со статусами COMPLETED и NODE_FAIL, будет пояснено далее).

Полученные данные содержат в том числе список вычислительных узлов, на которых выполнялись задания. Для каждого задания и для каждого узла из базы данных системных журналов выбираются записи о событиях, произошедших во временной окрестности завершения задания. Если длительность выполнения задания составляла более 15 минут, то выбираются все записи о событиях, происходивших за 15 минут до завершения задания и в течение еще 3-х минут после его завершения. Если же время выполнения задания было менее 15 минут, то в качестве начала времени выборки берется время начала выполнения задания минус 3 минуты. Временные рамки (15 минут и 3 минуты) были определены эмпирически после консультации с системными администраторами суперкомпьютера. Журналы событий для каждого узла сохраняются в файлы, именуемые по имени (hostname) узла, и размещаются в каталоге, имя которого соответствует идентификатору, присвоенному заданию системой Slurm.

Этап 2 – анализ системных журналов (скрипт fails2process). Для каждого задания по системным журналам вычислительных узлов, полученных на 1-м этапе, проводится две серии поисковых операций. Первая серия сохраняет в отдельные файлы записи со всех узлов согласно их важности. Тем самым воедино сводятся все события во временной окрестности завершения задания, имеющие уровни важности emergency, alert, critical, error, warning, notice, info и debug. Вторая серия нацелена на поиск конкретных событий, например, сообщений о проблемах с оперативной памятью, перегреве процессоров, сбоев в системном программном обеспечении и т.д. Поиск осуществляется по вхождению ключевых слов, специфичных при появлении записи о проблеме. Подробный перечень определяемых проблем будет приведен далее

Этап 3 – генерация отчетов за сутки (скрипт fails2csv). Здесь по каждому заданию, прошедшему обработку 2-го этапа, формируются отчеты в формате CSV (comma-separated values) и HTML. Оба формата отчета в целом содержат одни и те же данные: информацию о задании (дата и время начала и завершения, имя пользователя, командная строка, раздел и т.д.) и статистику по каждому типу проанализированных событий. Отчет в формате HTML содержит также ссылки на системные журналы узлов и списки обнаруженных событий. Отчет в формате CSV предназначен для анализа всех заданий за сутки с помощью внешнего программного обеспечения, например, MS Excel.

Этап 4 – генерация сводного отчета (скрипт fails2html). На этом этапе создается общий HTML-документ, содержащий ссылки на статистику заданий по суткам, обработанным на предыдущих этапах.

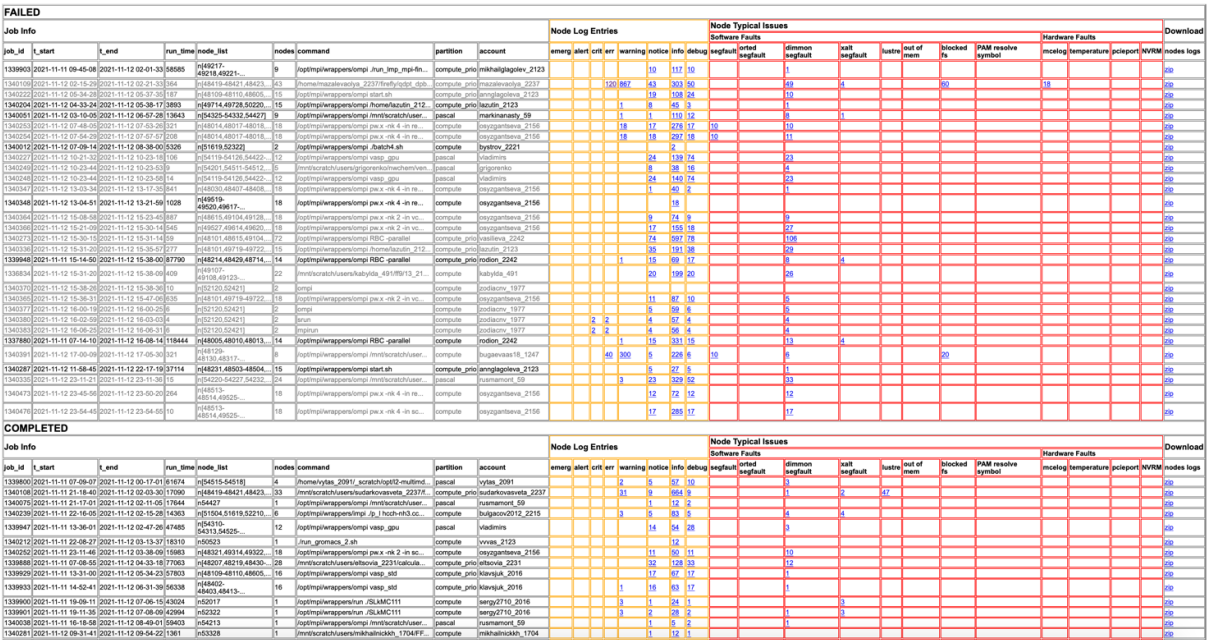
Если скрипты этапов 3 и 4 выполняются достаточно быстро – как правило, не более 1-2 минут – то для выполнения скрипта этапа 2 требуется порядка 10-20 минут, а скрипт 1-го этапа работает до 30-40 минут. Таким образом, автоматический анализ существенной части заданий одних суток работы суперкомпьютера занимает около часа. Вследствие этого описанные выше скрипты запускаются последовательно в 02:00 каждые сутки, когда загрузка служебных серверов суперкомпьютера минимальна.

На рис. 5 приведен фрагмент страницы сводного отчета. Каждая строка содержит дату, общее число проанализированных заданий и число заданий со статусами FAILED, NODE_FAIL и COMPLETED (каждое число представляет собой ссылку на соответствующий раздел суточного отчета). Подписи под общим количеством заданий – число заданий, для которых были обнаружены записи о событиях высокого уровня важности (emergency, alert, critical, error, warning). Последняя колонка – ссылка на суточный отчет в формате CSV.

Date	Tasks Processed *	FAILED Tasks	NODE_FAIL'ed Tasks	COMPLETED Tasks	Download Data
2021-11-12	67	Total: 30 warning 9 err 4 crit 2		Total: 37 warning 12 err 5 crit 3	csv
2021-11-13	55	Total: 18 warning 2 err 2 crit 2		Total: 37 warning 9 err 3 crit 1	csv
2021-11-14	64	Total: 15 warning 1 err 2		Total: 49 warning 12 err 11 crit 5	csv
2021-11-15	70	Total: 21 warning 2 err 1		Total: 49 warning 11 err 7	csv
2021-11-16	162	Total: 48 warning 8 err 16 crit 2	Total: 5 warning 1	Total: 109 warning 19 err 15 crit 4 alert 1	csv
2021-11-17	105	Total: 36 warning 8 err 2 crit 1		Total: 69 warning 9 err 10 crit 2	csv
2021-11-18	93	Total: 41 warning 4 err 9 crit 3		Total: 52 warning 7 err 5 crit 1	csv
2021-11-19	216	Total: 51 warning 11 err 2 crit 2		Total: 165 warning 19 err 100 crit 1 alert 1	csv
2021-11-20	80	Total: 12 warning 5 err 1 crit 1		Total: 68 warning 17 err 24	csv
2021-11-21	101	Total: 22 warning 6 err 6 crit 2		Total: 79 warning 37 err 22 crit 2	csv
2021-11-22	142	Total: 29 warning 6 err 2 crit 2 alert 1	Total: 2 warning 2 err 1 alert 1	Total: 111 warning 36 err 21 crit 1	csv
2021-11-23	267	Total: 44 warning 10 err 1 crit 3		Total: 223 warning 15 err 55 crit 8	csv
2021-11-24	123	Total: 16 warning 6 err 4	Total: 17 warning 3 err 1	Total: 90 warning 16 err 3 crit 5	csv

Рис. 5. Фрагмент страницы со сводным отчетом в веб-браузере

На рис. 6 приведен фрагмент отчета в формате HTML о заданиях за сутки. Отчет состоит из двух или трех таблиц, в которых сгруппированы задания (одна строка – одно задание). В примере на рисунке целиком уместилась таблица с заданиями со статусом FAILED и часть таблицы с заданиями со статусом COMPLETED. Еще одна таблица появляется, если за сутки были задания со статусом NODE_FAIL, однако такие задания встречаются не каждый день. Столбцы таблицы для заданий со всеми статусами одинаковы.



FAILED

Job Info									Node Log Entries									Node Typical Issues											Download
																		Software Faults								Hardware Faults					
job_id	t_start	t_end	run_time	node_list	nodes	command	partition	account	emerg	alert	crit	err	warning	notice	info	debug	segfault	orted segfault	dimmon segfault	xalt segfault	lustre	out of mem	blocked fs	PAM resolve symbol	mcelog	temperature	pcieport	NVRM	nodes logs
1339903	2021-11-11 09:45-08	2021-11-12 02-01-33	58585	n[49217-49218,49221-...	9	/opt/mpi/wrappers/ompi /run_lmp_mpi-fin...	compute_prio	mikhaligiagolev_2123						10	117	10			1										zip
1340109	2021-11-12 02-15-29	2021-11-12 02-21-33	364	n[48419-48421,48423...	43	/home/mazalevaolya_2237/firefly/qdpl_dpb...	compute_prio	mazalevaolya_2237				120	867	43	303	50			49	4		60		18				zip
1340222	2021-11-12 05-34-28	2021-11-12 05-37-35	187	n[48109-48110,48605...	15	/opt/mpi/wrappers/ompi start.sh	compute_prio	annglagoleva_2123						19	108	24			10										zip
1340204	2021-11-12 04-33-24	2021-11-12 05-38-17	3893	n[49714,49728,50220,...	15	/opt/mpi/wrappers/ompi /home/lazutin_212...	compute_prio	lazutin_2123					1	8	45	3			1										zip
1340051	2021-11-12 03-10-05	2021-11-12 06-57-28	13643	n[54325-54332,54427]	9	/opt/mpi/wrappers/ompi /mnt/scratch/user...	pascal	markinanastiy_59					1	1	110	12			8	1									zip
1340253	2021-11-12 07-48-55	2021-11-12 07-53-26	321	n[48714,48717-48718,...	18	/opt/mpi/wrappers/ompi pw.x -nk 4 -in re...	compute	osyzgantseva_2156					18	17	278	17	10		10										zip
1340254	2021-11-12 07-54-29	2021-11-12 07-57-57	208	n[48014,48017-48018,...	18	/opt/mpi/wrappers/ompi pw.x -nk 4 -in re...	compute	osyzgantseva_2156					18	18	297	18	10		11										zip
1340012	2021-11-12 07-09-14	2021-11-12 08-38-00	5326	n[51619,52322]	2	/opt/mpi/wrappers/ompi ./batch4.sh	compute	bystrov_2221						2															zip
1340227	2021-11-12 10-21-32	2021-11-12 10-23-18	106	n[54119-54126,54422-...	12	/opt/mpi/wrappers/ompi vasp_gpu	pascal	vladimirs						24	139	74			23										zip
1340249	2021-11-12 10-23-44	2021-11-12 10-23-53	9	n[54201,54511-54512,...	5	/mnt/scratch/users/grigorenko/machem/ven...	pascal	grigorenko						8	38	16			5										zip
1340248	2021-11-12 10-23-44	2021-11-12 10-23-58	14	n[54119-54126,54422-...	12	/opt/mpi/wrappers/ompi vasp_gpu	pascal	vladimirs						24	140	74			23										zip
1340347	2021-11-12 13-03-34	2021-11-12 13-17-35	841	n[48030,48407-48408,...	18	/opt/mpi/wrappers/ompi pw.x -nk 4 -in re...	compute	osyzgantseva_2156						1	40	2			1										zip
1340348	2021-11-12 13-04-51	2021-11-12 13-21-59	1028	n[49519-49520,49617-...	18	/opt/mpi/wrappers/ompi pw.x -nk 4 -in re...	compute	osyzgantseva_2156							18													zip
1340364	2021-11-12 15-08-58	2021-11-12 15-23-45	887	n[48615,49104,49128,...	18	/opt/mpi/wrappers/ompi pw.x -nk 2 -in vc...	compute	osyzgantseva_2156						9	74	9			9										zip
1340366	2021-11-12 15-21-09	2021-11-12 15-30-14	545	n[49527,49614,49620,...	18	/opt/mpi/wrappers/ompi pw.x -nk 2 -in vc...	compute	osyzgantseva_2156						17	155	18			27										zip
1340273	2021-11-12 15-30-15	2021-11-12 15-31-14	59	n[48101,48615,49104,...	72	/opt/mpi/wrappers/ompi RBC -parallel	compute_prio	vasilieva_2242						74	597	78			106										zip
1340336	2021-11-12 15-31-20	2021-11-12 15-35-57	277	n[48101,49719-49722,...	15	/opt/mpi/wrappers/ompi /home/lazutin_212...	compute_prio	lazutin_2123						35	191	38			29										zip
1339948	2021-11-11 15-14-50	2021-11-12 15-38-00	87790	n[48214,48429,48714,...	14	/opt/mpi/wrappers/ompi RBC -parallel	compute_prio	rodion_2242					1	15	69	17			8	4									zip
1338834	2021-11-12 15-31-20	2021-11-12 15-38-09	409	n[49107-49108,49123-...	22	/mnt/scratch/users/kabylda_491/ffl9/i13_21...	compute	kabylda_491						20	199	20			26										zip
1340370	2021-11-12 15-38-26	2021-11-12 15-38-36	10	n[52120,52421]	2	ompi	compute	zodiacnv_1977																					zip
1340365	2021-11-12 15-36-31	2021-11-12 15-47-06	635	n[48101,49719-49722,...	18	/opt/mpi/wrappers/ompi pw.x -nk 2 -in vc...	compute	osyzgantseva_2156						11	87	10			5										zip
1340377	2021-11-12 16-00-19	2021-11-12 16-00-25	6	n[52120,52421]	2	ompi	compute	zodiacnv_1977						5	59	6			5										zip
1340380	2021-11-12 16-02-59	2021-11-12 16-03-03	4	n[52120,52421]	2	srun	compute	zodiacnv_1977			2	2		4	57	4			4										zip
1340383	2021-11-12 16-06-25	2021-11-12 16-06-31	6	n[52120,52421]	2	mpirun	compute	zodiacnv_1977			2	2		4	56	4			4										zip
1337880	2021-11-11 07-14-10	2021-11-12 16-08-14	118444	n[48005,48010,48013,...	14	/opt/mpi/wrappers/ompi RBC -parallel	compute	rodion_2242					1	15	331	15			13	4									zip
1340391	2021-11-12 17-00-09	2021-11-12 17-05-30	321	n[48129-48130,48317-...	8	/opt/mpi/wrappers/ompi /mnt/scratch/user...	compute	bugaevaas18_1247				40	300	5	226	6	10		8			20						zip
1340287	2021-11-12 11-58-45	2021-11-12 22-17-19	37114	n[48231,48503-48504,...	15	/opt/mpi/wrappers/ompi start.sh	compute_prio	annglagoleva_2123						5	27	5			1										zip
1340335	2021-11-12 23-11-21	2021-11-12 23-11-36	15	n[54220-54227,54232,...	24	/opt/mpi/wrappers/ompi /mnt/scratch/user...	pascal	rusmamont_59					3	23	329	52			33										zip
1340473	2021-11-12 23-45-56	2021-11-12 23-50-20	264	n[48513-48514,49525-...	18	/opt/mpi/wrappers/ompi pw.x -nk 4 -in re...	compute	osyzgantseva_2156						12	72	12			12										zip
1340476	2021-11-12 23-54-43	2021-11-12 23-54-55	10	n[48513-48514,49525-...	18	/opt/mpi/wrappers/ompi pw.x -nk 4 -in sc...	compute	osyzgantseva_2156						17	285	17			17										zip

COMPLETED

Job Info									Node Log Entries									Node Typical Issues											Download
																		Software Faults								Hardware Faults					
job_id	t_start	t_end	run_time	node_list	nodes	command	partition	account	emerg	alert	crit	err	warning	notice	info	debug	segfault	orted segfault	dimmon segfault	xalt segfault	lustre	out of mem	blocked fs	PAM resolve symbol	mcelog	temperature	pcieport	NVRM	nodes logs
1339800	2021-11-11 07-09-07	2021-11-12 00-17-01	61674	n[54515-54518]	4	/home/vytas_2091/_scratch/opt/t2-multimd...	pascal	vytas_2091					2	5	57	10			3										zip
1340108	2021-11-11 21-18-49	2021-11-12 02-03-30	17090	n[48419-48421,48423,...	33	/mnt/scratch/users/sudarkovaaveta_2237/f...	compute_prio	sudarkovaaveta_2237					31	9	664	9			1	2	47								zip
1340075	2021-11-11 21-17-01	2021-11-12 02-11-05	17644	n54427	1	/opt/mpi/wrappers/ompi /mnt/scratch/user...	pascal	rusmamont_59						1	12	2													zip
1340239	2021-11-11 22-16-05	2021-11-12 02-15-28	14363	n[51504,51619,52210,...	6	/opt/mpi/wrappers/mpi ./p_l hcch-nh3.cc...	compute	bulgacov2012_2215					3	5	83	5			4	4									zip
1339947	2021-11-11 13-36-01	2021-11-12 02-47-26	47485	n[54310-54313,54525-...	12	/opt/mpi/wrappers/ompi vasp_gpu	pascal	vladimirs						14	54	28			3										zip
1340212	2021-11-11 22-08-27	2021-11-12 03-13-37	18310	n50523	1	./run_gromacs_2.sh	compute	vvvas_2123							12														zip
1340252	2021-11-11 23-11-46	2021-11-12 03-38-09	15983	n[48321,49314,49322,...	18	/opt/mpi/wrappers/ompi pw.x -nk 2 -in sc...	compute	osyzgantseva_2156						11	90	11			10										zip
1339888	2021-11-11 07-08-55	2021-11-12 04-33-18	77063	n[48207,48219,48430-...	28	/mnt/scratch/users/eltsovia_2231/calcula...	compute_prio	eltsovia_2231						32	128	33			12										zip
1339929	2021-11-11 13-31-00	2021-11-12 05-34-23	57803	n[48109-48110,48605,...	16	/opt/mpi/wrappers/ompi vasp_std	compute_prio	klavsjuk_2016						17	67	17			1										zip
1339933	2021-11-11 14-52-41	2021-11-12 06-31-39	56338	n[48402-48403,48413-...	16	/opt/mpi/wrappers/ompi vasp_std	compute_prio	klavsjuk_2016					1	16	63	17			1										zip
1339900	2021-11-11 19-09-11	2021-11-12 07-06-15	43024	n52017	1	/opt/mpi/wrappers/run ./SLMC111	compute	sergy2710_2016					3	1	24	1			3										zip
1339901	2021-11-11 19-11-35	2021-11-12 07-08-09	42994	n52322	1	/opt/mpi/wrappers/run ./SLMC111	compute	sergy2710_2016					3	2	28	2			1	3									zip
1340038	2021-11-11 16-18-58	2021-11-12 08-49-01	59403	n54213	1	/opt/mpi/wrappers/ompi /mnt/scratch/user...	pascal	rusmamont_59						1	5	2			1										zip
1340281	2021-11-12 09-31-41	2021-11-12 09-54-22	1361	n53328	1	/mnt/scratch/users/mikhalnickkh_1704/FF...	compute	mikhalnickkh_1704						1	12	1													zip

Рис. 6. Фрагмент отчета за сутки в веб-браузере

Для удобства восприятия информации столбцы в таблицах сгруппированы по цветам. В левой части черным или серым цветом приводятся данные о задании (Job Info): идентификатор Slurm, время начала выполнения, время завершения, общее время счета в секундах, список задействованных для расчета узлов, число узлов, командная строка, раздел, имя пользователя. Серым отмечены задания, длительность которых не превосходит 15 минут. Выше говорилось, что для таких заданий временное окно событий считается по-другому, чем для более длительных заданий, и в системные журналы, вообще говоря, могут попадать посторонние записи, не относящиеся к исследуемому заданию – например, от предыдущих заданий на узле. Уже на приведенном примере можно заметить, что большинство FAILED-заданий не выполнялись дольше 15 минут, тогда как время выполнения успешно завершившихся заданий превышает этот порог. Действительно, такая тенденция имеет место практически во все проанализированные дни. Впрочем, временами она нарушается – обычно за счет запусков одним-двумя пользователями большого количества коротких и успешно завершающихся заданий.

Правее секции Job Info размещается оранжевая секция таблицы, обозначенная как Node Log Entries. Здесь для каждого задания указано число записей в системных журналах, разбитых по категориям важности – от emergency до debug. При наличии ненулевого значения записей число представляет собой ссылку, по которой можно загрузить сводку таких записей. Можно заметить, что в левых колонках оранжевой секции таблицы, соответствующих наиболее важным уровням событий (emergency, alert, …), записи встречаются скорее как исключения, тогда как в правой части, где сгруппированы записи о менее важных событиях (notice, info, …), они присутствуют почти всегда.

В правой части таблицы находится красная секция Node Typical Issues, разбитая на две части: Software Faults и Hardware Faults. Числа здесь соответствуют количеству обнаруженных записей о проблемах, обнаруженных на этапе 2 анализа. На текущий момент реализовано определение следующих проблем:

- “segfault” – аварийное завершение («падение») пользовательского приложения;
- “orted segfault” – аварийное завершение процесса orted – процесса подсистемы обмена сообщениями OpenMPI;
- “dimmon segfault” – аварийное завершение процесса dimmon – агента системы мониторинга DiMMon;
- “xalt segfault” – аварийное завершение процесса xalt_submis – системного процесса пакета XALT, определяющего используемые пользователями библиотеки и пакеты прикладных программ;
- “lustre” – проблемы с общей файловой системой Lustre;

- “out of mem” – превышение приложением лимита доступной оперативной памяти;
- “blocked task” – проблемы с общей файловой системой;
- “PAM resolve symbol” – проблемы с механизмами аутентификации на узле;
- “mcelog” – аппаратные проблемы, определяемые демоном mcelog (MCE = Machine Check Exception), в основном касающиеся оперативной памяти;
- “temperature” – превышение заданного температурного порога ядром или конструктивом центрального процессора;
- “pcieport” – ошибки системной шины PCIe узла;
- “NVRM” – проблемы с графическим ускорителем NVIDIA.

Последняя, самая правая колонка таблицы содержит ссылку на архив, содержащий полный набор системных журналов всех вычислительных узлов, задействованных для выполнения задания, во временно*й* окрестности его завершения.

Аналогичные данные, за исключением ссылок, содержатся и в CSV-версии отчета за сутки. Работа с CSV-файлами обладает одним важным достоинством по сравнению с HTML-представлением данных – их можно объединить простой конкатенацией (с последующим удалением дублирующихся заголовков), что позволяет проводить статистический анализ данных за любой интересующий период времени, а не только за одни сутки.

4. Результаты анализа

Переходя непосредственно к результатам анализа возможных причин появления сбоев в работе пользовательских приложений, проведенного с помощью описанного выше инструментария, необходимо объяснить включение в состав рассматриваемых заданий таковые со статусами COMPLETED и NODE_FAIL. Если найденные ситуации, предположительно являющиеся причинами сбоев FAILED-заданий, обнаруживаются также для успешно завершенных заданий, то такие причины, очевидно, не являются критическими. Для заданий, завершенных со статусом NODE_FAIL, проблемы в работе вычислительных узлов фиксируются самой системой Slurm, которая проводит серию проверок после окончания выполнения каждого приложения. Соответственно, интересно понять, насколько эти проблемы могут быть выявлены путем исследования системных журналов.

За исследуемый период времени (два месяца в конце 2021 года) на вычислительных узлах во временно*й* окрестности завершения заданий не случалось событий с уровнем важности emergency. События с уровнем важности alert были обнаружены при завершении всего 8 заданий. При этом шесть из них завершились успешно (со статусом COMPLETED), а оставшиеся два – со статусами FAILED и NODE_FAIL. Эти два 50-узловых задания были последовательно запущены одним и тем же пользователем (хотя и с отличающимися командными строками) и попали на сходное множество узлов, среди которых был узел n50703. Именно этот узел при выполнении заданий выдал серию событий с важностью alert, говорящих об ошибках в работе клиента файловой системы PanFS; одновременно с этим в системном журнале узла появились и сообщения о сбоях в работе системной шины PCIe. Совокупность этих событий, вероятно, и привела к тому, что СУПЗ Slurm в итоге вывела упомянутый узел из счетного поля. Заданий с событиями уровня critical было уже гораздо больше – 162, из которых 90 были завершены успешно, 71 – со статусом FAILED и 1 – со статусом NODE_FAIL. Тем самым показано, что события, которые операционная система вычислительного узла записывает в системный журнал с отметкой об их высокой важности, далеко не всегда являются основными причинами сбоев пользовательских приложений.

В ходе дальнейшего анализа было отмечено аномально большое число случаев аварийного завершения процессов системы мониторинга DiMMon – это происходило в конце 2/3 всех попавших в рассмотрение заданий. Также были отмечены случаи «падения» процессов пакета XALT (671 за исследуемый период). Эти события представляют определенный интерес для системных администраторов суперкомпьютера, однако на работу пользовательских приложений они не влияют. Кроме того, было обнаружено 37 случаев «падения» процесса orted, являющегося частью подсистемы обмена сообщениями OpenMPI. Эти проблемы наблюдались в разные дни и на разных узлах, при этом большинство заданий было завершено успешно. Интересно, что в 29 случаях

одним и тем же пользователем запускался прикладной пакет GROMACS. Можно выдвинуть предположение о том, что особенности программного кода этого пакета могут приводить к сбоям в работе системного процесса, однако такое предположение требует дополнительных исследований. Но необходимо учитывать, что события фиксируются только в окрестности завершения приложений, и не исключено, что при анализе событий за все время выполнения приложений подобная зависимость может не подтвердиться. В 97 заданиях были отмечены случаи аварийного завершения процессов самого пользовательского приложения. Тем не менее, 52 из них были завершены успешно. Здесь можно сделать вывод об удачной программной архитектуре ряда прикладных пакетов (ORCA, NWCHEM, VASP и других), сохраняющих работоспособность даже при «падении» части процессов.

У 18% исследованных заданий отмечались проблемы с доступом к общей файловой системе Lustre. Подобные проблемы обычно носят временный характер и явно не приводят к аварийному завершению приложений, хотя, вероятно, могут замедлять их работу. У 5 заданий были выявлены проблемы превышения доступной оперативной памяти (out of memory), все они были принудительно завершены операционной системой вычислительных узлов. Для 79 заданий были зафиксированы аппаратные сбои, обнаруженные системным процессом mcelog, из них 30 (37%) завершились со статусом FAILED. Можно сделать вывод, что наличие аппаратных сбоев в конце работы приложения увеличивает вероятность его аварийного завершения. Обычно такие ситуации связаны с исправленными ошибками оперативной памяти ECC. Подобные ситуации в целом не являются критическими, однако рост числа исправленных ошибок ECC, как правило, в какой-то момент приводит к появлению неисправленных ошибок, что уже говорит о деградации модуля оперативной памяти. В конце работы 147 приложений отмечались случаи превышения температуры отдельных процессорных ядер и процессорных модулей целиком. В целом такие ситуации также не являются критическими, они отрабатываются на аппаратном уровне путем снижения тактовой частоты процессора (throttling), но это, в свою очередь, может привести к замедлению работы приложений. У 7 и 8 заданий отмечались сбои системной шины PCIe и ускорителей NVIDIA соответственно, однако некоторые из них завершались успешно.

5. Заключение

Анализ событий из системных журналов вычислительных узлов во временной окрестности завершения приложений позволяет прояснить причины аварийного завершения для некоторой их части, однако однозначного решения вопроса не дает. Целесообразно добавить и другие факторы для анализа, например, события из системных журналов части служебных серверов, необходимых для поддержания работы суперкомпьютера (в первую очередь это серверы метаданных файловой системы Lustre), события из буфера ядра операционной системы (dmesg), счетчики сетевых ошибок. Предполагается сделать это в дальнейшем.

Литература

1. Vl. Voevodin, A. Antonov, D. Nikitenko, P. Shvets, S. Sobolev, I. Sidorov, K. Stefanov, Vad. Voevodin, S. Zhumatiy. Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. In Journal: Supercomputing Frontiers and Innovations, Vol.6, No.2 (2019). pp.4–11. DOI:10.14529/jsfi190201
2. Slurm SchedMD. Slurm Support and Development. URL: https://www.schedmd.com (дата обращения: 10.08.2022).
3. Yinglung Liang, Yanyong Zhang, A. Sivasubramaniam, M. Jette and R. Sahoo. BlueGene/L Failure Analysis and Prediction Models. International Conference on Dependable Systems and Networks (DSN'06), 2006, pp. 425-434, doi: 10.1109/DSN.2006.18.
4. InfluxDB: Open Source Time Series Database | InfluxData. URL: https://www.influxdata.com (дата обращения: 10.08.2022).
5. The rocket-fast Syslog Server – rsyslog. URL: https://www.rsyslog.com (дата обращения: 10.08.2022).

6. Telegraf Open Source Server Agent | InfluxDB. URL: https://www.influxdata.com/time-series-platform/telegraf/ (дата обращения: 10.08.2022).
7. Fluentd | Open Source Data Collector | Unified Logging Layer. URL: https://www.fluentd.org (дата обращения: 10.08.2022).
8. Chronograf: Complete Dashboard Solution for InfluxDB. URL: https://www.influxdata.com/time-series-platform/chronograf/ (дата обращения: 10.08.2022).
9. ELK stack The ELK Stack: From the Creators of Elasticsearch | Elastic. URL: https://www.elastic.co/what-is/elk-stack (дата обращения: 10.08.2022).
10. A. V. Adinets, P. A. Bryzgalov, Vad V. Voevodin, S. A. Zhumatii, D. A. Nikitenko, and K. S. Stefanov. Job digest: an approach to dynamic analysis of job characteristics on supercomputers. Вычислительные методы и программирование, 13:160–166, 2012.

Моделирование течения воздуха в системе охлаждения блока верхнего РУ ВВЭР с применением CFD

В.Ю. Волков, Л.А. Голибродо, А.А. Крутиков, О.В. Кудрявцев

АО ОКБ «ГИДРОПРЕСС»

В настоящей работе рассмотрена разработка и применение CFD модели системы воздушного охлаждения БВ РУ АЭС-2006 – типового проекта атомной станции нового поколения «3+» с улучшенными технико-экономическими показателями. Сложность геометрии проточного тракта системы воздушного охлаждения БВ и необходимость учета нескольких влияющих друг на друга коллекторов, работающих при разной температуре, обуславливают необходимость применения нестандартных методов работы с исходной геометрией (технология пространственной морфологической фильтрации), а также необходимость применения расчетных сеток размерностью до 300 млн. ячеек. Для CFD-моделирования данной задачи применялся высокопроизводительный вычислительный кластер, насчитывающий 2000 ядер (IntelXeon E5-2698 v4 2,2 Гц) и 5 Тб оперативной памяти.

Ключевые слова: ВВЭР, CFD, блок верхний, математическое моделирование, привод системы управления и защиты, суперкомпьютерное моделирование.

1. Введение

Одним из важнейших элементов реакторной установки с водо-водяным энергетическим реактором (РУ ВВЭР) является блок верхний (БВ) реактора, надежная и безопасная эксплуатация которого во многом обусловлена возникающими при этом температурными нагрузками на важные и чувствительные к температуре элементы. БВ эксплуатируется в герметичной защитной оболочке РУ, входит в состав границ давления первого контура РУ и воспринимает давление теплоносителя первого контура и нагрузки от внутрикорпусных устройств (ВКУ) и тепловыделяющих сборок (ТВС). Следует отметить, что непосредственный контакт некоторых элементов БВ с теплоносителем первого контура не позволяет использовать теплоизоляцию для уменьшения подвода тепла, что обуславливает применение принудительной циркуляции воздуха для охлаждения приводов системы управления и защиты (СУЗ) и разъемов датчиков внутриреакторного контроля (ВРК).

В настоящей работе представлено CFD-моделирование гидродинамики и теплообмена при вынужденной конвекции воздуха в системе охлаждения блока верхнего реактора ВВЭР-1200 и в близлежащем надреакторном пространстве.

Сложность геометрии проточного тракта системы воздушного охлаждения БВ и необходимость учета нескольких влияющих друг на друга коллекторов, работающих при разной температуре, обуславливают необходимость применения нестандартных методов работы с исходной геометрией (технология пространственной морфологической фильтрации), а также необходимость применения расчетных сеток размерностью до 300 млн. ячеек. Для CFD-моделирования данной задачи применялся высокопроизводительный вычислительный кластер, насчитывающий 2000 ядер (IntelXeon E5-2698 v4 2,2 Гц) и 5 Тб оперативной памяти. Использование высокопроизводительных вычислительных систем при решении таких задач является обязательным.

В результате расчетного моделирования при различных расходах воздуха в системе воздушного охлаждения БВ были получены распределения расходов охлаждающего воздуха в патрубках СУЗ и ВРК, определяющие температурные условия работы расположенных в них элементов. Дополнительно был выполнен анализ чувствительности результатов расчетного моделирования к способу задания граничных условий.

Стоит отметить, что не смотря на большой опыт применения CFD-технологий в атомной отрасли, их использование до настоящего момента весьма ограниченно для моделирования систем вентиляции конструкций внутри защитной оболочки [1]. В тоже время аналогичные задачи

достаточно широко решаются в строительной отрасли при моделировании климата и микроклимата в помещениях больших объемов [2-4].

2. Описание конструкции

Блок верхний служит для уплотнения реактора, размещения приводов СУЗ, выводов датчиков ВРК, сигнализаторов течи и т.д. Основными элементами БВ являются: крышка реактора с приваренными патрубками, металлоконструкция с траверсой для транспортировки БВ, приводы СУЗ. К блоку верхнему присоединен трубопровод воздушника, электрические коммуникации приводов СУЗ, датчиков системы ВРК, сигнализаторы течи [5].

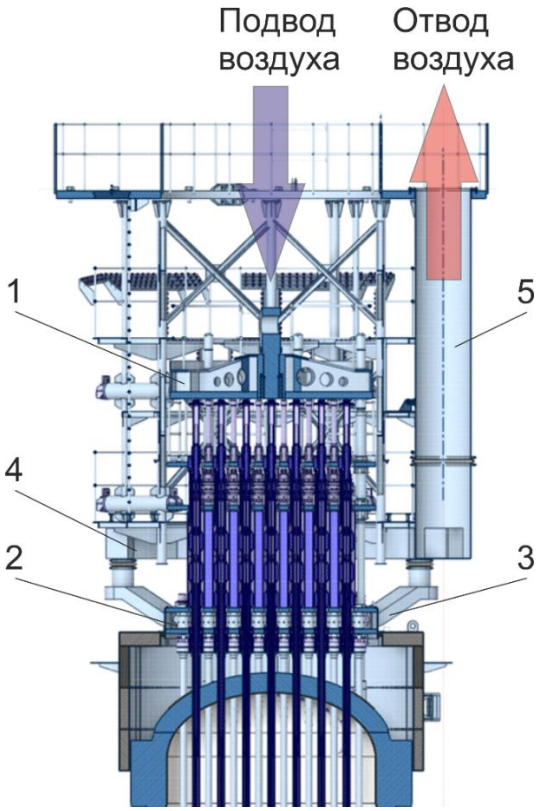
Для отвода тепла в состав БВ входит система воздушного охлаждения вытяжного типа, обеспечивающая подвод охлаждающего воздуха из центрального зала к элементам БВ, и удаление нагретого воздуха через систему воздушных коллекторов в вентиляционный канал и далее в специальный теплообменник.

На уровне нижней части приводов СУЗ организован плоскоцилиндрический воздушный коллектор, образованный двумя плитами.

На высоте электромагнитов вокруг каждого привода СУЗ установлены шестигранные трубы (патрубки СУЗ), которые закреплены на верхней плите воздушного коллектора и организуют проток охлаждающего воздуха вокруг электромагнитов.

В местах расположения патрубков вывода датчиков ВРК в соответствующих гнездах воздушного коллектора установлены страхующие устройства с перфорацией, через которую обеспечивается проток воздуха для охлаждения.

Воздушный коллектор соединен посредством шести отводящих коробов с вентиляционным коллектором, расположенным выше воздушного коллектора. От вентиляционного коллектора вверх в центральный зал отходит вентиляционный канал, на выходе из которого установлены вентиляторы. БВ, металлоконструкция и система воздушного охлаждения представлены на рисунке 1.



1 – БВ, 2 – воздушный коллектор, 3 – отводящие короба, 4 – вентиляционный коллектор, 5 – вентиляционный канал

Рис. 1. БВ, металлоконструкция и система воздушного охлаждения

3. Постановка задачи

Объектом исследования является проточный тракт системы охлаждения верхнего блока реактора ВВЭР-1200, отводящих коробов, вентиляционного коллектора, вентиляционного канала, а также непосредственно прилегающие области надреакторного пространства, ограниченные полом центрального зала, бетонной шахтой и теплоизоляцией реактора. Предмет исследования - процессы гидродинамики и теплообмена, определяющие расходы в патрубках СУЗ и ВРК, температуру охлаждающего воздуха и гидравлические потери в проточном тракте. Целевым параметром расчетного исследования является распределения расходов охлаждающего воздуха в патрубках СУЗ и ВРК.

Охлаждающая воздушная среда поступает из центрального зала и из прилегающего окружающего пространства, попадает на вход в шестигранные патрубки СУЗ и в трубы страхующих устройств и, охлаждая электрооборудование приводов СУЗ и электроразъемы СВРД и измерительных линий СВРК, поступает в плоскоцилиндрический воздушный коллектор. Из воздушного коллектора верхнего блока нагретый воздух отводится по шести отводящим коробам и далее через вентиляционный коллектор поступает в вентиляционный канал. Поток воздушной среды организуется за счет работы вентиляторов, которые создают необходимое разряжение в плоскоцилиндрическом воздушном коллекторе с шестью отводящими коробами и вентиляционном канале.

Расход воздушной среды в вентиляционном канале варьируется в широком диапазоне значений. Температура воздуха на входе в конструкцию соответствует температуре в центральном зале.

Тепловые граничные условия задаются из условия стационарного режима работы РУ ВВЭР-1200 на номинальной мощности. В расчетах рассмотрена работа 121 привода СУЗ, общие тепловые потери от которых составляют 0,5 МВт.

4. Разработка твердотельной модели расчетной области

Распределение воздуха по патрубкам СУЗ определяется как гидравлическим сопротивлением последних, так и особенностями подвода воздуха, обусловленными геометрией металлоконструкций БВ. В связи с этим при разработке твердотельной модели расчетной области необходимо учесть затеснение создаваемое металлоконструкциями БВ, хотя необходимость их детального моделирования отсутствует.

Исходными данными при создании твердотельной модели расчетной области является конструкторская твердотельная модель с избыточной для данной задачи детализацией (множество зазоров, крепежных элементов и т.д.).

Стандартные способы огрубления твердотельной модели подразумевают использование врапперов [6]. Однако в данном случае их использование усложняется обилием тонкостенных элементов (пластин, металлических профилей и т.д.), геометрия которых при использовании враппера может передаваться некорректно. Поэтому после удаления мелких элементов из конструкторской твердотельной модели была применена технология пространственной морфологической фильтрации, позволяющая получить качественную геометрию при автоматизированном закрытии крупных зазоров и упрощении деталей с характерным размером менее размера фильтра.

По сравнению со стандартным враппером, данный способ позволяет использовать размеры пространственного фильтра больше, чем характерные размеры некоторых значимых элементов (например, толщины пластин). Также способ позволяет сохранить качество ребер модели. Из недостатков стоит отметить то, что в результате операции внутренние углы превращаются в скругления, за счет которых образуется большое количество полигонов, для работы с которыми требуются значительные вычислительные ресурсы, как процессорные мощности, так и оперативная память (десятки гигабайт на одной рабочей станции). Однако на последующее расчетное моделирование данный факт не оказывает влияния, поскольку в данной области используется относительно грубая расчетная сетка (с характерным размером ячейки, соответствующему размеру

полигона или образовавшегося в результате процедуры морфологии скругления). При этом основные геометрические размеры и проходные сечения проточного тракта сохраняются. Также несмотря на то, что часть операций с геометрией распараллеливаются в пределах ядер одной рабочей станции, другая часть выполняется на одном ядре. Это приводит к тому, что общее время подобной обработки может исчисляться сутками. Таким образом, в дальнейшем представляется перспективной доработка и распараллеливание алгоритмов, особенно при необходимости обработки более масштабных моделей (РУ целиком, здания, металлоконструкции). Несмотря на указанные недостатки, использование данной технологии все равно быстрее, чем подготовка геометрии в полностью ручном режиме.

В качестве примера технологии пространственной морфологической фильтрации на рисунках 2 и 3 представлены твердотельные модели до и после упрощения соответственно.

Общий вид разработанной твердотельной модели представлен на рисунке 4. Расчетная область представляет собой инвертированную твердотельную модель металлоконструкций и является проточным трактом воздушной среды. Из рисунка видно, что расчетная модель сверху ограничена полом центрального зала, сбоку - бетонной шахтой, снизу - теплоизоляцией реактора.

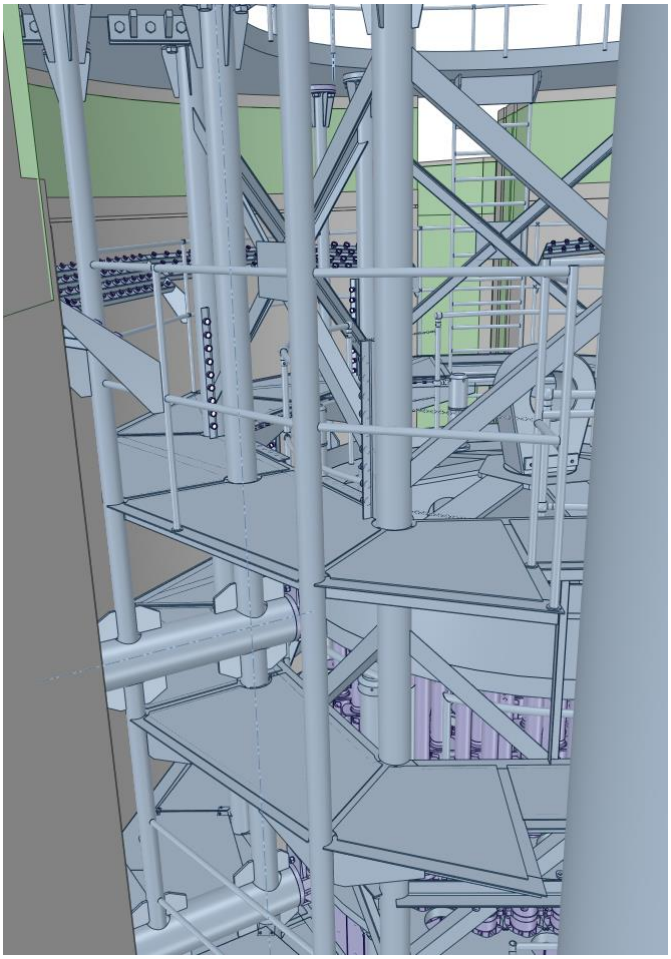


Рис. 2. Твердотельная модель БВ до упрощения

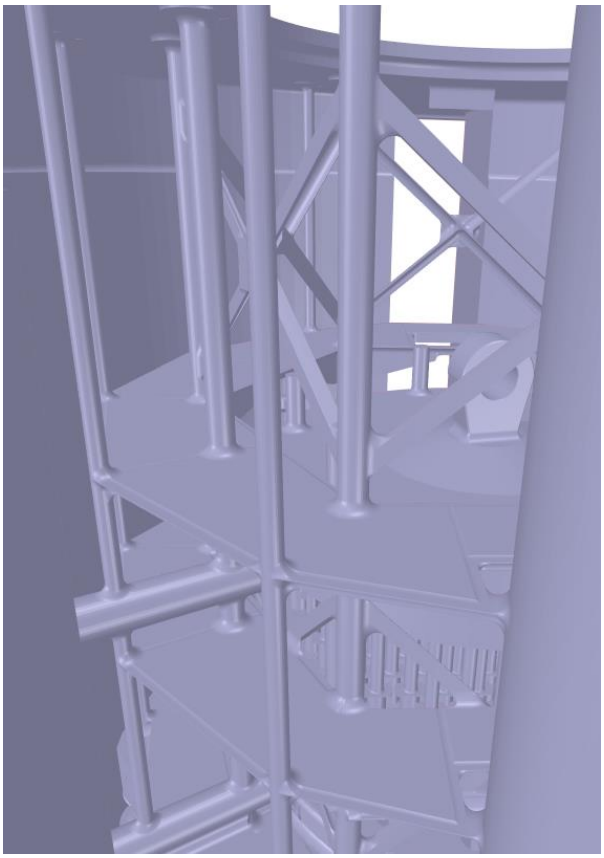


Рис. 3. Твердотельная модель БВ после упрощения

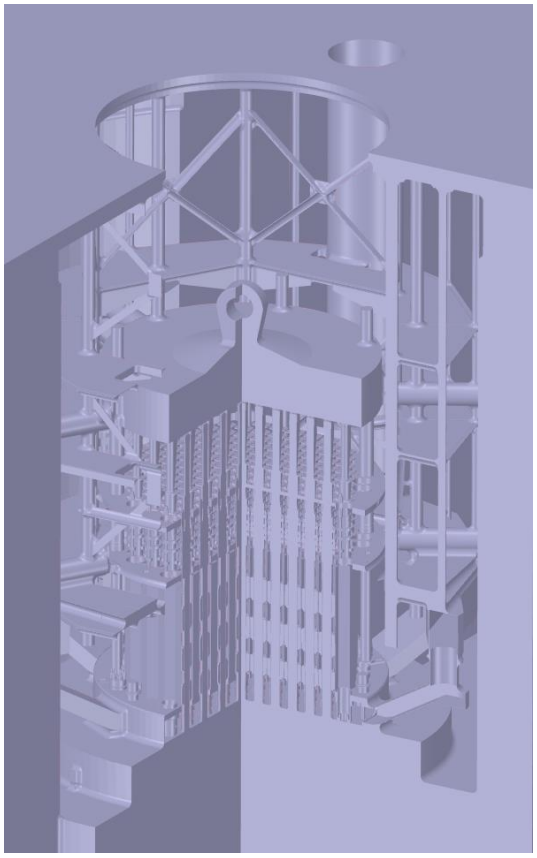


Рис. 4. Твердотельная модель расчетной области

5. CFD-модель

В соответствии с поставленной задачей в коде STAR-CCM+ была разработана CFD- модель системы воздушного охлаждения БВ РУ АЭС-2006 и непосредственно прилегающих областей. Выбор расчетного кода обусловлен наличием лицензии, возможностью эффективной работы с большими CFD моделями вплоть до 1 млрд. расчётных ячеек, и соответствием требованиям к вычислительным ресурсам и дисковому пространству.

Основные допущения математической модели: воздух считается ньютоновской и несжимаемой средой, течение воздуха стационарное, турбулентное, границы проточной части считаются гидравлически гладкими стенками, вибрации стенок отсутствуют.

Теплофизические свойства воздуха (плотность, динамическая вязкость, удельная теплоемкость и теплопроводность) зависят от температуры.

Для моделирования турбулентности была использована k-ε-Realizable модель турбулентности [2].

Входное граничное условие описывает подвод воздушной среды через отверстие в полу центрального зала и из окружающего пространства. Выходное граничное условие описывает отвод воздушной среды через вентиляционный канал. Стенки с подогревом –стенки чехлов приводов СУЗ и электромагнитов, патрубков СУЗ. Адиабатические стенки – все поверхности образующие расчетную область, за исключением перечисленных выше стенок с подогревом. Следует отметить, что для стенок с подогревом задавалась плотность теплового потока.

Разработанная сеточная модель учитывает все конструктивные особенности проточного тракта, способные в значительной мере оказывать влияние на целевой параметр. Подробность дискретизации расчетной области выбрана в соответствии с результатами моделирования тестовой задачи, на которой был проведен ряд численных исследований по валидации и верификации – течение воздуха в одном патрубке СУЗ. Для данной расчетной модели варьировались характерные размеры ячеек в диапазоне от 0.1 мм до 5 мм, а также количество пристенных сеточных слоев в области пограничного слоя – от 2 до 10. В рамках серии валидационных расчетов была выбрана приемлемая с точки зрения вычислительных ресурсов и точности результатов расчетная сетка с характерным размером ячеек 5 мм и пятью пристенными сеточными слоями. Дальнейшее уменьшение характерного размера ячеек расчетной сетки было признано нецелесообразным вследствие многократного роста размерности сетки общей модели при незначительном повышении точности.

Большая часть сеточной модели выполнена из полиэдрических элементов, за исключением пограничного слоя и каналов с постоянным поперечным сечением, где методом экструзии создана призматическая сетка. Сеточная модель собрана из отдельных модулей, в каждом из которых предусмотрена возможность локального изменения детализации сеточной модели, либо замены сеточной модели модуля целиком, что дополнительно увеличивает гибкость и универсальность CFD-модели.

Общий вид сеточной модели расчетной области представлен на рисунке 5 (внутренняя структура расчетной области не показана). Полная размерность сеточной модели составляет 273 млн. элементов. Из всех частей большая размерность у модуля, включающего в себя проточный тракт 121 патрубка СУЗ, - более 200 млн. элементов.

Вопросам эффективности распараллеливания задач данного класса частично посвящена работа [1]. Для организации параллельных вычислений пакет STAR-CCM+, как и большинство коммерческих пакетов, использует MPI. Протестировав в АО ОКБ «ГИДРОПРЕСС» в 2016 году эффективность распараллеливания для типичных задач атомной энергетики к этому вопросу далее не обращались, а выбор количества используемых узлов определялся в основном срочностью и допустимым временем решения задач, количеством лицензий на различное ПО, задействованное в конкретный момент времени на кластере. Ускорение счета по сравнению с использованием одного ядра для использования 40 ядер (один узел) составляло 17.4 раза. Для разработанной расчетной сетки количество ячеек, приходящихся на одно ядро, составляет 136500 при использовании 2000 ядер.

Время, необходимое для выполнения одного расчета составляет порядка суток. Указано только время выполнения собственно расчетов. Однако, стоит заметить, что общее время полу-

чения результатов на подобной модели с проведением валидационных и верификационных исследований составляет до нескольких месяцев. Для новых конструкций на данный момент невозможно ускорить ряд процессов за счет распараллеливания. Сокращается только объем валидационных и верификационных исследований.

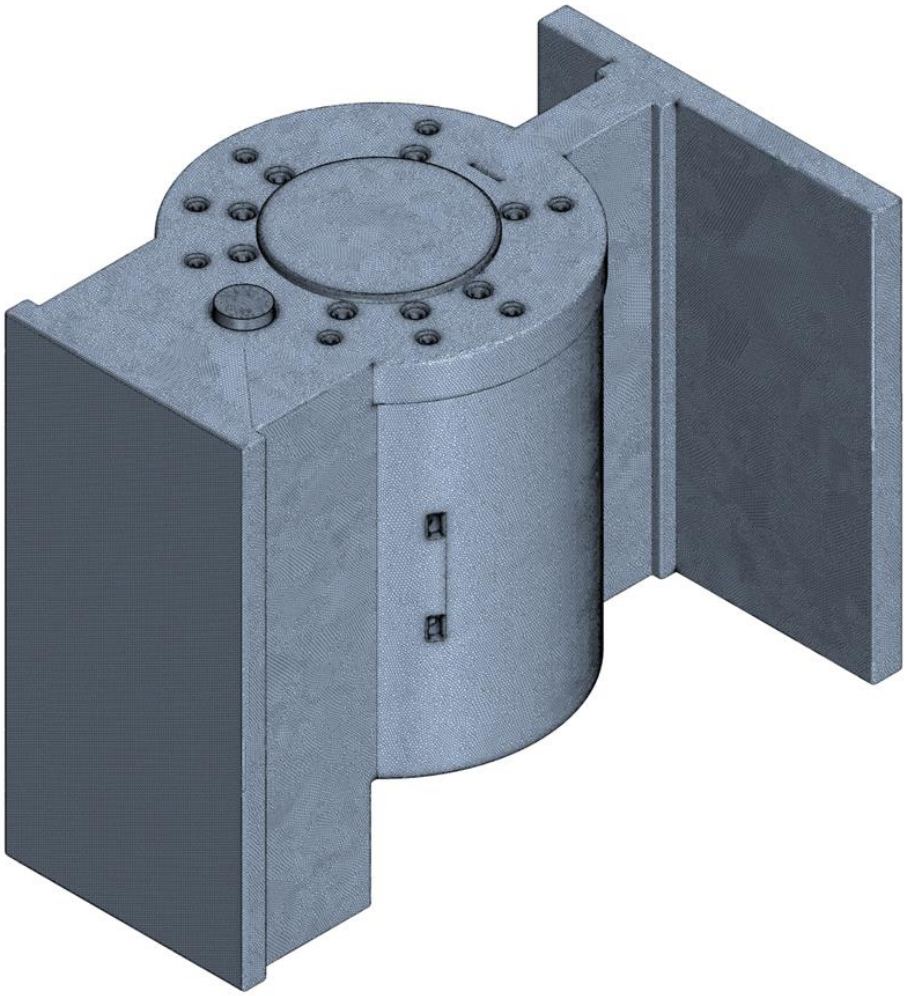


Рис. 5. Сеточная модель расчетной области

6. Результаты расчета

В рамках проведенного расчетного моделирования были выполнены расчеты с расходами воздуха в вентиляционном канале в рабочем диапазоне.

С целью исследования анализа чувствительности результатов расчета к входным и тепловым граничным условиям рассмотрены два варианта задания условий однозначности, отличающиеся распределением расхода воздуха на входе в расчетную область и профилем тепловых потерь.

В первом варианте граничных условий подвод воздушной среды к верхнему блоку осуществляется только через отверстие в полу центрального зала, при этом тепловые потери в каждом приводе СУЗ постоянны и не зависят от расхода воздушной среды в патрубке СУЗ.

Во втором варианте граничных условий подвод воздушной среды к верхнему блоку осуществляется через отверстие в полу центрального зала и из прилегающего окружающего пространства, при этом тепловые потери в каждом приводе СУЗ зависят от расхода воздушной среды в патрубке СУЗ.

В результате проведенного анализа следует, что с увеличением расхода воздушной среды в вентиляционном канале несколько увеличиваются неравномерности расходов по патрубкам СУЗ и СВРК. Также следует сделать важный вывод о слабой зависимости расходов в патрубках СУЗ

и СВРК от профиля тепловых потерь и от способа подвода воздуха к верхнему блоку. При этом профиль тепловых потерь влияет на температурный режим чехлов СУЗ.

Для демонстрации результатов выбран режим со вторым вариантом граничных условий. Осредненные по сечениям проточного тракта результаты моделирования тепломассообмена в верхнем блоке представлены на рисунках 6 - 7.

На рисунке 6 представлены осредненные по сечениям расходы воздушной среды в патрубках СУЗ, патрубках СВРК, шестью коробам и в вентиляционном канале (осредненные значения объемных расходов приведены к параметрам воздушной среды на входе в БВ). Следует отметить, что в рабочем диапазоне расход воздуха в вентиляционном канале на 5000 м3/ч больше, чем расход на входе в расчетную область.

На рисунке 7 показано распределение осредненной температуры воздушной среды на выходе из патрубков СУЗ и СВРК, а также в воздушном коллекторе и вентиляционном канале.

На рисунках 8, 9 представлены локальные распределения векторного поля скорости воздушной среды в верхнем блоке для продольного и поперечного сечений соответственно. Распределения представлены в логарифмической шкале.

На рисунке 8 хорошо видны низкоскоростные (примерно до 1 м/с) крупномасштабные вихревые течения до входа в патрубки СУЗ и СВРК. Из рисунка 9 хорошо видна сложная вихревая форма течения в проточном тракте верхнего блока.

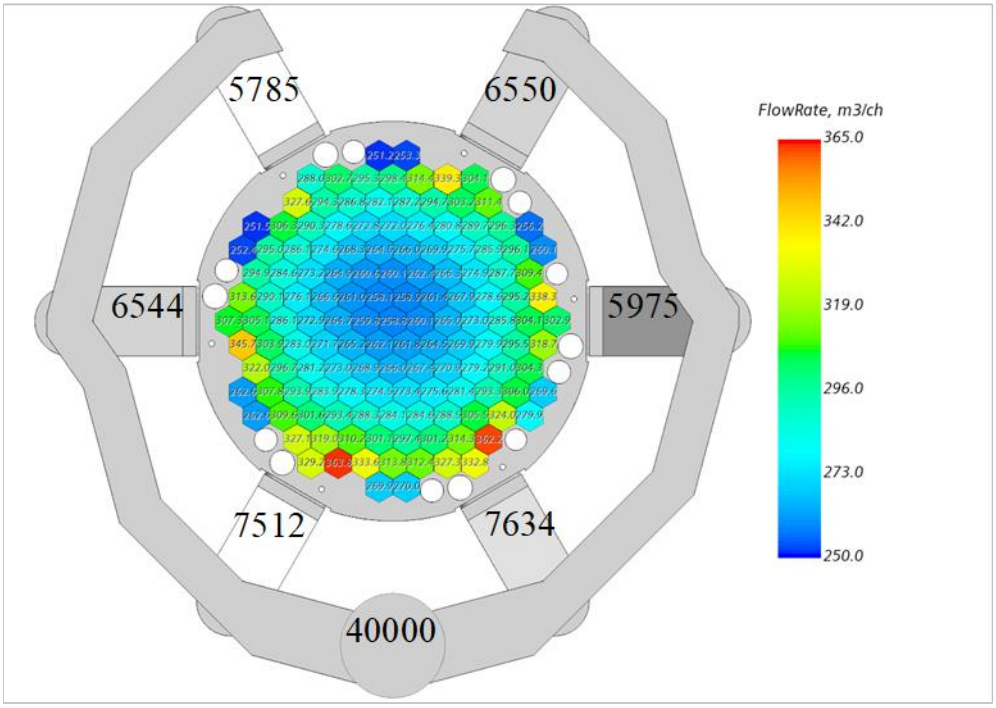


Рис. 6. Расходы воздуха на выходе из патрубков СУЗ и СВРК, а также в воздушном коллекторе и вентиляционном канале, приведенные к температуре входа

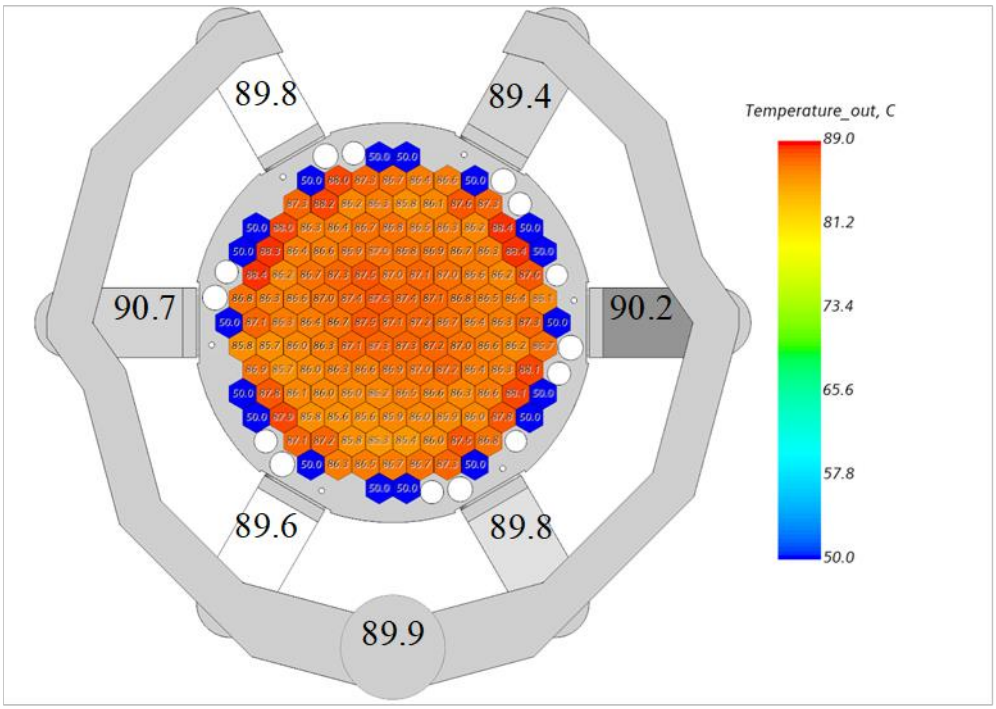


Рис. 7. Температура воздуха на выходе из патрубков СУЗ и СВРК

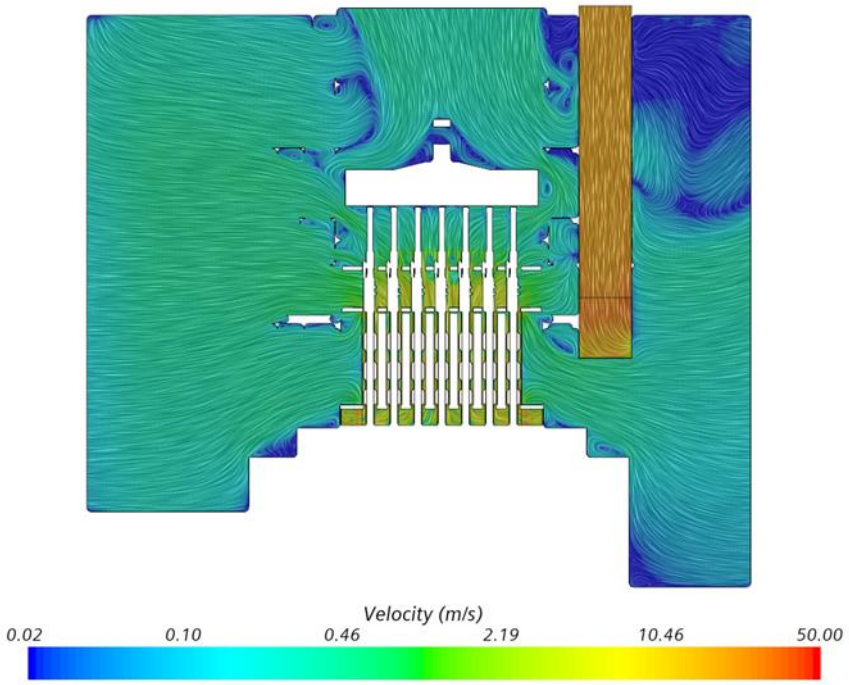


Рис. 8. Векторное поле скорости воздуха в продольном сечении

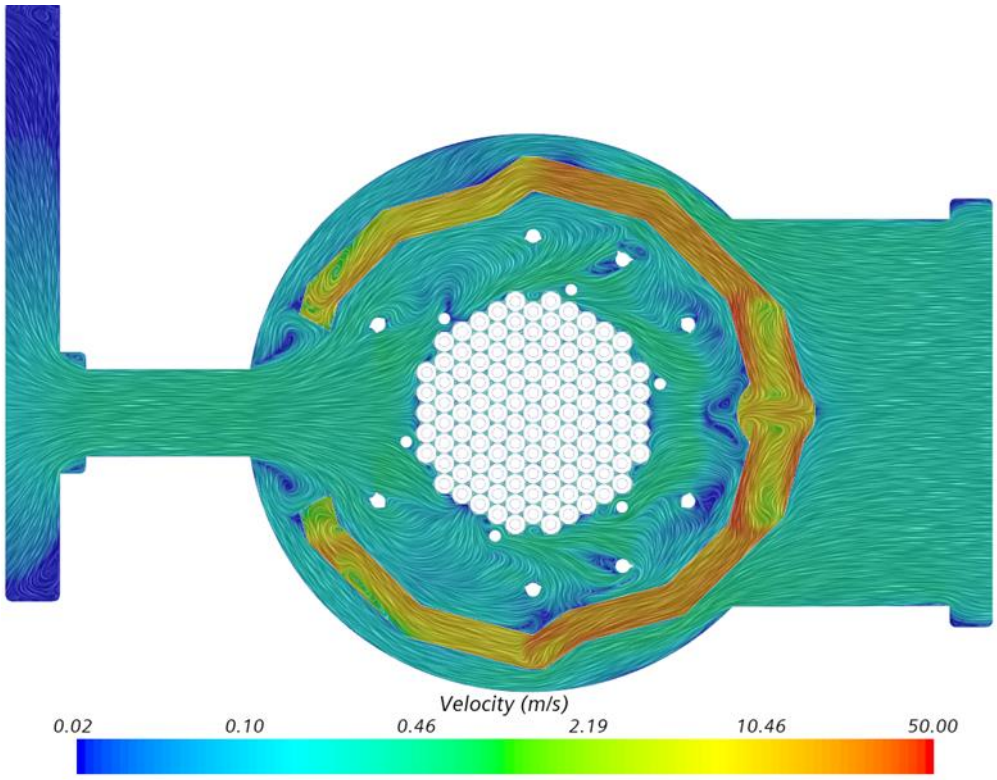


Рис. 9. Векторное поле скорости воздуха в поперечном сечении

7. Заключение

В рамках представленной в настоящем докладе работы было выполнено CFD-моделирование гидродинамики и теплообмена при вынужденной конвекции воздуха в системе охлаждения верхнего блока реактора ВВЭР-1200 и в близлежащем надреакторном пространстве. Моделирование выполнено с допущением работы РУ на номинальных параметрах и наличием 121 привода СУЗ. В работе было рассмотрены режимы различными расходами продуваемого воздуха через конструкцию, температура которого равна температуре в центральном зале.

По итогам анализа полученных результатов CFD-моделирования следует выделить следующие особенности процессов тепломассообмена в проточном тракте системы охлаждения верхнего блока реактора ВВЭР-1200:

• величина расхода через патрубки СУЗ уменьшается от периферии к центру;

• в распределении расхода через патрубки СУЗ отсутствует симметрия в тангенциальном направлении, а именно увеличивается расход в патрубках, ближе расположенных к вентиляционному каналу (разница в расходах может достигать 20% и более), что связано с влиянием отвода через один выход и приводит к смещению минимального расхода через патрубки СУЗ на один ряд в направлении от вентиляционного канала;

• для всех рассмотренных режимов минимальный объемный расход через патрубок СУЗ не опускается ниже проектных требований;

• для всех рассмотренных режимов средняя температура воздуха в вентиляционном канале соответствует проектным требованиям;

• расходы по патрубкам СУЗ практически не зависят от способа снятия тепла с чехлов СУЗ и от способа подвода воздуха к верхнему блоку;

• неравномерность температуры воздуха при равномерном теплосъеме с патрубков СУЗ (первый вариант граничных условий) до 5 раз выше по сравнению с неравномерностью температуры воздуха при теплосъеме с патрубков СУЗ, зависящем от расхода воздушной среды через патрубки СУЗ (второй вариант граничных условий); результаты CFD-расчетов могут использоваться при проведении конструкторских теплогидравлических расчетов верхнего блока ВВЭР-

1200, оптимизации конструкции воздушного тракта верхнего блока, при анализе эксплуатационных данных, и могут служить исходными данными для дальнейших расчетных исследований температурного состояния приводов СУЗ.

Литература

1. Волков В.Ю., Голибродо Л.А., Крутиков А.А. и др. Разномасштабные задачи тепломассообмена в атомной энергетике. Вестн. ЮУрГУ. Сер. Выч. матем. информ., 2017, том 6, выпуск 4. С. 60–73. DOI: 10.14529/cmse170405.
2. Денисихина Д.М., Русаков С.В. Изменение параметров микроклимата в течение хоккейного матча в зале крытой ледовой арены. АВОК: Вентиляция, отопление, кондиционирование воздуха, теплоснабжение и строительная теплофизика. 2019. № 6. С. 26-37.
3. Денисихина Д., Луканина М., Самолетов М. Математическое моделирование вентиляции завода. Здания высоких технологий. 2013. Т. 2. № 2. С. 60-63.
4. V.V. Agafonova, A.P. Skibin, V.Y. Volkov (20) (PDF) Modeling of Air Distribution in an Office Building Using Microperforated Textile Air Duct. Available from: https://www.researchgate.net/publication/350968470_Modeling_of_Air_Distribution_in_an_Office_Building_Using_Microperforated_Textile_Air_Duct [accessed Jun 15 2022].
5. Резепов В.К., Денисов В.П., Кирилюк Н.А., и др. Реакторы ВВЭР-1000 для атомных электростанций. Москва: ИздАТ, 2003. 333с.
6. Siemens. Simcenter STAR-CCM+ Documentation Version 2020.2.

Прототип глобальной негидростатической модели атмосферы на сетке кубическая сфера для прогноза погоды*

В.В. Шашкин[1,2], Г.С. Гойман[1,2,3], М.А. Толстых[1,2], Р.Ю. Фадеев[1,2,3]

[1]Институт вычислительной математики им. Г.И. Марчука РАН,
[2]Гидрометцентр России,
[3]Московский физико-технический институт

Представлен прототип глобальной модели атмосферы для системы численного прогноза погоды нового поколения. Решаются трехмерные уравнения динамики сжимаемой атмосферы без использования гидростатического приближения по вертикали (уравнения Эйлера). По горизонтали используется сетка «кубическая сфера», имеющая равномерное разрешение на сфере. На данном этапе не учитывается рельеф поверхности, для интегрирования по времени используется явная схема. Представленная модель испытана на общепринятых идеализированных задачах и показала результаты, соответствующие текущему мировому уровню. На сетке с разрешением 20 км по горизонтали модель масштабируется до 4000 процессорных ядер. Работы по развитию модели будут продолжены.

Ключевые слова: моделирование атмосферы, прогноз погоды, кубическая сфера

1. Введение

Математические модели атмосферы состоят из блока численного решения уравнений гидротермодинамики на вычислительной сетке и блока параметризаций процессов подсеточного масштаба. Блок параметризаций описывает вклад процессов гидродинамического (конвекция, турбулентность) и негидродинамического (солнечная радиация, фазовые переходы воды) характера в движения воздуха на масштабах, разрешаемых на сетке. Отдельные параметризации опираются на эмпирические/статистические закономерности.

Повышение разрешения сетки модели атмосферы позволяет перевести часть процессов из категории подсеточных в разрешаемые на сетке модели, что положительно влияет на точность их воспроизведения. Шаг сетки по горизонтали в современных глобальных моделях атмосферы для прогноза погоды 10-20 км. Предполагается, что рост доступных вычислительных ресурсов позволит в скором будущем перейти к моделям с шагом сетки 3-5 км по горизонтали. Подобное разрешение значительно повысит способность моделей воспроизводить мезомасштабные метеорологические процессы, которые отвечают за большую часть опасных метеорологических явлений в средних широтах.

Переход к моделированию с шагом сетки менее 10 км не может быть осуществлен путем масштабирования современных моделей, так как необходимо решить ряд проблем:

1. Отказ от регулярной широтно-долготной сетки (Рис. 1) и переход на одну из сеток с квазиравномерным разрешением на сфере [1]. Практические расчеты на регулярной широтно-долготной сетке, используемой сейчас в большинстве прогностических моделей, сильно затруднены из-за сходимости меридианов к полюсам и чрезмерного уменьшения шага сетки по долготе в высоких широтах.

2. Использование системы уравнений без гидростатического приближения по вертикали. В большинстве современных моделей используется квазистатическая система уравнений, которая не позволяет корректно описывать динамику, например, орографических волн при длине волны менее 10-20 км.

*Исследование выполнено в ИВМ РАН при поддержке Российского научного фонда (проект 21-71-30023)

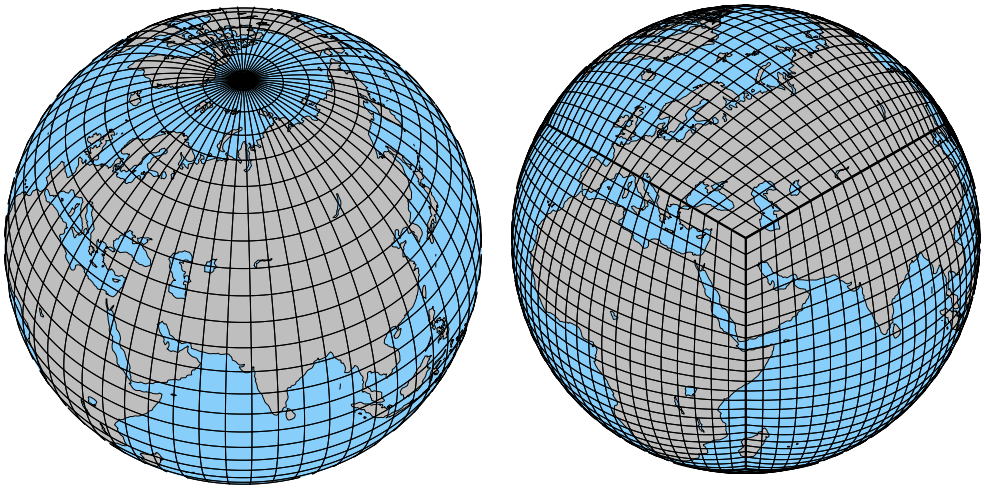


Рис. 1: Регулярная широтно-долготная сетка (слева), сетка кубическая сфера (права)

3. Применение численных методов, не требующих глобальных пересылок данных (для эффективности параллельных расчетов с использованием 10^4–10^5 процессорных ядер). Необходимо отказаться от спектрального метода, основанного на разложении полей в ряд по сферическим гармоникам или в тригонометрический ряд Фурье по долготе, который применяется, например, в глобальной модели IFS Европейского центра прогноза погоды [2] и модели ПЛАВ Гидрометцентра России [3].

В настоящее время системы прогноза погоды, для которых решены перечисленные выше проблемы, созданы только в Германии, США и Канаде. В этих странах в оперативную эксплуатацию внедрены глобальные негидростатические модели на сетках с квазиравномерным разрешением на сфере ICON [4], FV3 [5], GEM [6].

В ИВМ РАН и Гидрометцентре России ведется разработка блока динамики для негидростатической модели атмосферы на сетке кубическая сфера [7], [8]. В данной статье приводится описание версии динамического блока на основе явного метода интегрирования по времени без учета рельефа поверхности. Данная версия позволяет получать численные решения для ряда трехмерных идеализированных тестовых задач по воспроизведению динамики атмосферы и является важным промежуточным этапом на пути к динамическому блоку новой модели атмосферы (с рельефом поверхности и использованием явно-неявного или полностью неявного метода интегрирования по времени).

2. Система уравнений гидротермодинамики атмосферы

Существуют два принципиально разных подхода к моделированию динамики атмосферы без гидростатического приближения по вертикали. Первый основан на использовании уравнений Эйлера для сжимаемого газа. Во втором вводятся те или иные приближения квази-несжимаемости [9]. Второй подход представляется логичным, так как скорости течений воздуха в тропосфере и стратосфере существенно дозвуковые. На практике, однако, в глобальных моделях атмосферы [4–6, 10] преобладает использование «исходных» уравнений динамики сжимаемого газа, так как в этом случае не требуется ни задания фоновых профилей плотности, ни решения плохо-обусловленных уравнений типа Пуассона.

Для разрабатываемой перспективной модели атмосферы мы выбрали сжимаемую систему уравнений Эйлера в переменных потенциальная температура - функция давления

Экснера. По вертикали используется z-координата (высота над поверхностью Земли). Выбор прогностических переменных и вертикальной координаты определяется тем, что для дискретизации этой системы на разнесенной пространственной сетке достигаются оптимальные свойства распространения звуковых, инерционно-гравитационных и Россби-волн [11]. Уравнения записываются:

$$\frac{\partial u}{\partial t} = -\boldsymbol{v} \cdot \nabla u - C_p \theta \frac{\partial P}{\partial x} + fv, \tag{1}$$

$$\frac{\partial v}{\partial t} = -\boldsymbol{v} \cdot \nabla v - C_p \theta \frac{\partial P}{\partial y} - fu, \tag{2}$$

$$\frac{\partial w}{\partial t} = -\boldsymbol{v} \cdot \nabla w - C_p \theta \frac{\partial P}{\partial z} - g, \tag{3}$$

$$\frac{\partial \theta}{\partial t} = -\boldsymbol{v} \cdot \nabla \theta, \tag{4}$$

$$\frac{\partial P}{\partial t} = -\boldsymbol{v} \cdot \nabla P - \frac{R}{C_v} P \nabla \cdot \boldsymbol{v}, \tag{5}$$

где $\boldsymbol{v} = (u, v, w)^T$ – вектор скорости ветра, θ – потенциальная температура, $P = (p/p_0)^\kappa$ – функция давления Экснера, p – давление, $p_0 = 1000$ мБар – нормировочная константа, C_p – теплоемкость сухого воздуха при постоянном давлении, R – газовая константа сухого воздуха, $\kappa = R/C_p$, f – параметр Кориолиса (сила Кориолиса приведена для приближения «мелкой атмосферы» [12]).

Отметим,что в данной статье речь идет о блоке численного решения уравнений динамики адиабатической атмосферы. Не учитываются такие процессы как солнечное и тепловое излучение, фазовые переходы влаги и др., которые в моделях атмосферы традиционно относятся к блоку параметризованного описания процессов подсеточного масштаба. Блок параметризаций не вносит особенностей и трудностей в параллельную реализацию моделей атмосферы, так как все параметризации – локально одномерные алгоритмы по вертикали, а в моделях атмосферы на данном этапе используется только декомпозиция расчетной области по горизонтальным переменным.

3. Дискретизация уравнений динамики атмосферы на сетке кубическая сфера

3.1. Вычислительная сетка

Для реализации негидростатической модели сжимаемой атмосферы мы выбрали сетку типа кубическая сфера (см. например, [13]). Такая сетка получается путем центральной проекции прямоугольной сетки на гранях куба на вписанную сферу (Рис. 1). Для большей равномерности сетки на сфере, сетка на гранях куба берется равномерной в угловых координатах $(X, Y) = (\tan\alpha, \tan\beta)$, $\alpha, \beta \in [-\pi/4, \pi/4]$.

Сетка кубическая сфера, фактически, состоит из 6 отдельных криволинейных неортогональных сеток, стыкующихся на образах ребер куба. Координатные линии, при этом, испытывают излом. К достоинствам сетки кубическая сфера следует отнести высокую равномерность разрешения, логически прямоугольную структуру, которая позволяет эффективно программировать методы высокого порядка аппроксимации. Недостатки сетки – неортогональность, излом координатных линий на ребрах.

По горизонтали применяется разнесение сетки типа «C» [14], значения давления определяются в центрах ячеек сетки $(\alpha, \beta) = (-\pi/4 + (i+1/2)\Delta, -\pi/4 + (j+1/2)\Delta)$, $\Delta = \pi/2N$, где

N – размерность сетки. Компоненты горизонтальной скорости ветра определяются в точках, которые сдвинуты на половину шага от точек давления: $(\alpha, \beta) = (-\pi/4+i\Delta, -\pi/4+(j+1/2)\Delta)$ для компоненты скорости по направлению α, $(\alpha, \beta) = (-\pi/4+(i+1/2)\Delta, -\pi/4+j\Delta)$ для компоненты по направлению β.

По вертикали используется разнесенная сетка Чарни-Филлипса. Точки компонент горизонтальной скорости ветра лежат на одном вертикальном уровне с точками давления. Вертикальная компонента ветра w и потенциальная температура θ определены в точках, лежащих на одной вертикальной прямой с точками давления и сдвинутых на половину шага по вертикали относительно P-точек. Нижняя и верхняя точки, где определены вертикальная скорость и потенциальная температура, лежат на уровне земли и верхней границе модельной атмосферы соответственно.

3.2. Пространственная дискретизация

Производные по направлениям α, β, z в операторах градиента и дивергенции аппроксимировались с помощью конечно-разностных формул для первой производной, удовлетворяющих свойству суммирования по частям (summation by parts – SBP) [15]. SBP-свойство конечных разностей – аналог свойства интегрирования по частям для аналитических операторов:

$$\int_a^b q(x)\frac{\partial p(x)}{\partial x}dx = -q(a)p(a) + q(b)p(b) - \int_a^b p(x)\frac{\partial q(x)}{\partial x}dx, \tag{6}$$

где q и p – произвольные дифференцируемые функции.

Использование конечно-разностных SBP-операторов позволяет строить устойчивые численные методы высокого порядка аппроксимации для гиперболических уравнений на сетках, состоящих из нескольких блоков (таких, как кубическая сфера) или при наличии границ сетки (как у атмосферы по вертикали). Устойчивость доказывается энергетическим методом через сохранение квадратичного инварианта. Доказано, что существуют SBP-аппроксимации первой производной, с порядком аппроксимации $p/2p$, p=1, 2, 3, 5 (у границы/внутри блока сетки) [16].

В данной работе используются SBP-конечные разности порядка $2/1$, $4/2$ на разнесенных сетках, построенные в работе [17]. Используется методология применения SBP операторов для разнесенных криволинейных сеток [18]. Граничные условия на стыках блоков кубической сферы (ребрах куба) реализуются методом SAT [15] в его варианте для разнесенных сеток [17]. При постановке граничных условий используется непрерывность давления и контравариантной компоненты скорости горизонтального ветра, перпендикулярной ребру куба (с точностью до ориентации локальной системы координат).

Адвективные слагаемые в уравнениях аппроксимируются по формулам направленных разностей 1-го – 4-го порядков. При этом, необходимые значения полей в точках, находящихся за границей блока сетки, вычисляются с помощью интерполяции по точкам соседних блоков, как в [7]. Использование направленных разностей позволяет ввести диссипацию мелкомасштабных движений и избежать нелинейной неустойчивости без применения искусственных диссипаторов типа гипер-диффузии.

3.3. Интегрирование по времени

В рассматриваемой версии динамического блока для интегрирования по времени используется явная схема Рунге-Кутта 4-го порядка. В проведенных экспериментах шаг по времени ограничен числом Куранта для звуковых волн, распространяющихся по вертикали. В дальнейшем планируется перейти к явному по горизонтали и неявному по вертикали методу интегрирования по времени [19] или к полунеявному методу интегрирования.

Рис. 2: Разбиение сетки кубическая сфера на 192 подобласти

4. Параллельная реализация

Параллельно с динамическим ядром мы развиваем библиотеку для параллельных вычислений на сетке кубическая сфера ParCS [20], [8], которая реализует все необходимые для вычислений операции: декомпозиция области, обмены областями зависимости между MPI-процессами, ввод/вывод полей. Применяется технология MPI, двумерная декомпозиция расчетной области. Каждый блок кубической сферы разделяется на прямоугольные подобласти равного размера, см. Рис. 2. Число областей может превосходить число MPI-процессов, при этом, каждый MPI-процесс может обрабатывать несколько подобластей. Такой подход увеличивает гибкость разбиения области, способствует лучшему использованию кэш-памяти за счет дробления массивов, содержащих прогностические поля, на небольшие куски.

5. Результаты моделирования

5.1. Эксперимент «Распространение инерционно-гравитационной волны»

В эксперименте «Распространение гравитационной волны» возмущение потенциальной температуры амплитуды 1 K, с характерным горизонтальным масштабом 5 км накладывается на устойчиво стратифицированную атмосферу (частота Брента-Вяйсяля $N = 0.01\,\mathrm{c}^{-1}$). Начальное возмущение локализовано на экваторе, на Гринвичском меридиане. От начального возмущения распространяются циркуляные волны. Радиус планеты уменьшен по сравнению с радиусом Земли в 125 раз, чтобы выделить негидростатические эффекты. Эффект Кориолиса отключен.

Решение разработанного динамического ядра на сетке с вертикальным и горизонтальным разрешением 1 км изображено на Рис. 3. На рисунке представлено вертикальное сечение поля возмущения потенциальной температуры через 3600 с моделирования. Решения разработанного динамического ядра хорошо согласуются с решениями других моделей, полученными в эксперименте проекта Dynamical Core Model Intercomparison Project-2012 [21]. При сравнении решений со средним потоком и без него можно отметить снижение амплитуды волны на заднем фронте (ветер на рисунке дует слева направо), что характерно для всех моделей.

5.2. Эксперимент «Плотностное течение»

Этот эксперимент является адаптацией к сферической геометрии известного стандартного теста [22]. Рассматривается эволюция холодного пузыря воздуха (аномали потенциаль-

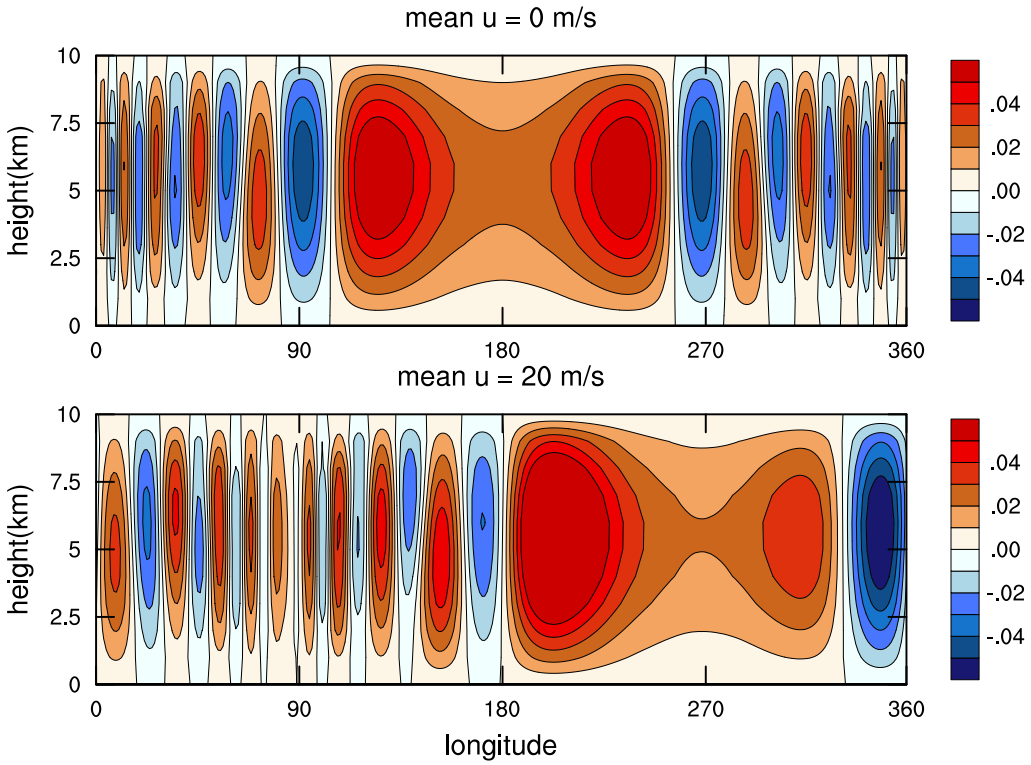


Рис. 3: Вертикальное сечение поля возмущения потенциальной температуры по экватору в эксперименте «Распространение гравитационной волны» через 3600 с моделирования. Сверху - в отсутствии среднего потока, снизу - средний поток со скоростью на экваторе 20 м/с.

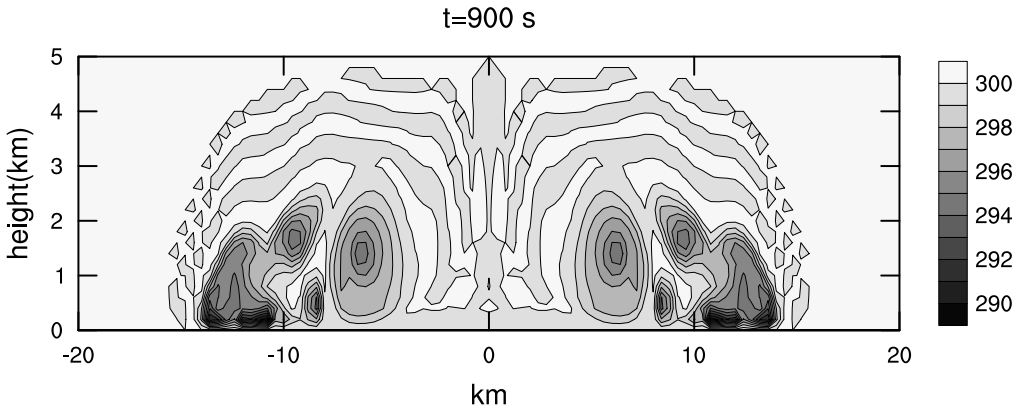


Рис. 4: Вертикальное сечение поля потенциальной температуры эксперименте плотностное течение.

ной температуры -15 K), опускающегося в нейтрально стратифицированной атмосфере и разбивающегося об Землю. Пузырек имеет полувысоту 2000 м, полуширину 4000 м, ширина пузырька в долготном направлении выбрана бесконечной, чтобы воспроизвести двумерную динамику (аномалия потенциальной температуры локализована вдоль экватора). Разработанный динамический блок воспроизводит узкий и быстрый поток плотного холодного воздуха и формирование вихрей при столкновении пузырька с поверхностью. На Рис. 4 изображено вертикальное сечение поля потенциальной температуры в решении с разрешением 200 м по горизонтали и вертикали. Решение хорошо согласуется с моделью Английской метеослужбы [23].

5.3. Эксперимент геострофическое равновесие

Рассматриваемый эксперимент [24] обобщает аналогичную задачу для моделей мелкой воды [25] на трехмерный случай. Течение типа твердого вращения с максимальной скоростью 20 м/с находится в устойчивом равновесии под действием силы градиента давления и силы Кориолиса и является стационарным решением системы уравнений динамики атмосферы. С целью увеличить роль негидростатических эффектов, мы уменьшили радиус Земли в 125 раз и в 125 раз увеличили скорость ее вращения. Ошибки численного решения и эффективные порядки сходимости в эксперименте после одного полного оборота (16000 с) приведены в Таблице 1. Максимальная ошибка решения определяется аппроксимацией у ребер и особенно углов куба. При использовании SBP-операторов первой производной порядка аппроксимации 4-2 можно ожидать сходимости со вторым порядком. Практический порядок сходимости больше второго.

5.4. Параллельная эффективность

Параллельная эффективность представленной версии динамического ядра испытывалась на вычислительной системе CRAY XC-40 Росгидромета. Был проведен эксперимент по исследованию сильной масштабируемости на сетке $6 \times 512 \times 512$, 30 уровней по вертикали (разрешение по горизонтали – 20 км, по вертикали – 300 м). Динамическое ядро масштабируется до 4032 процессорных ядер с эффективностью 70% (Рис. 5). Расчет на 4032 ядрах соответствует размеру подобласти одного MPI-процесса примерно 20×20 точек по горизонтали. Основная причина снижения эффективности параллельных расчетов при большом числе ядер – обмены в блоке интерполяции полей в «виртуальные точки». В настоящий момент они реализованы путем пересылки значений полей в прямоугольных областях, при этом пересылается значительное количество данных, не нужных для алгоритма интерпо-

Δx км	Δz км	N	Nlev	$\max\|w\|$ (м/с)	$\max\|P - P(t=0)\|$
2,5	1,0	32	10	$1,8 \times 10^{-3}$	$1,6 \times 10^{-5}$
1,25	0,5	64	20	$3,2 \times 10^{-4}$	$1,4 \times 10^{-6}$
0,765	0,25	128	40	$7,7 \times 10^{-5}$	$1,19 \times 10^{-7}$
Порядок сходимости				2,3	3,1

Таблица 1: Максимальные ошибки полей вертикальной скорости и функции давления Экснера после одного полного оборота в эксперименте геострофическое равновесие при использовании SBP-операторов первых производных порядка аппроксимации 4-2.

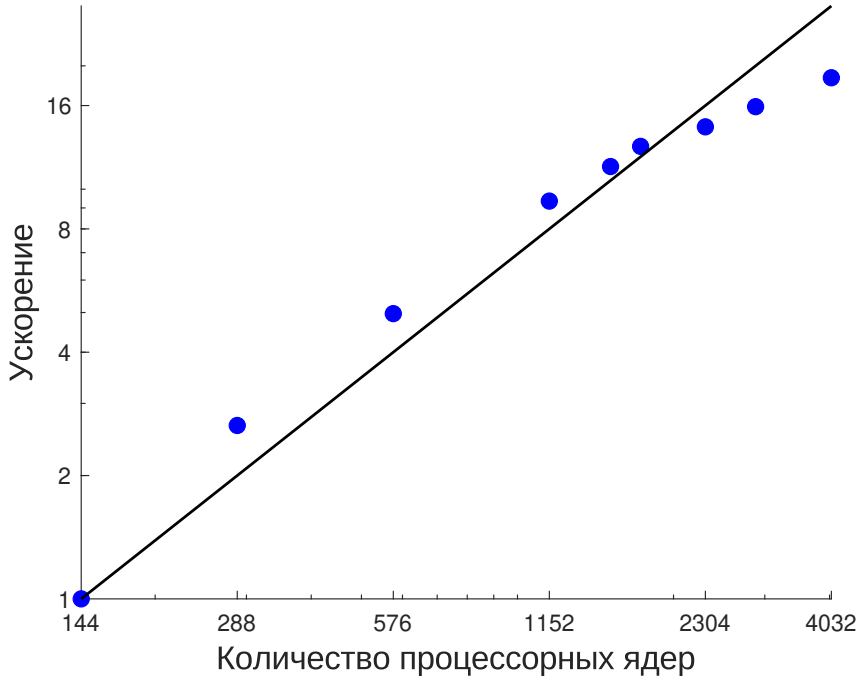


Рис. 5: Сильная масштабируемость негидростатического динамического ядра на сетке $6 \times 512 \times 512 \times 30$, по оси x – число процессорных ядер CRAY XC40.

ляции. Параллельная эффективность может быть также повышена с помощью внедрения технологии OpenMP в сочетании с MPI.

6. Заключение

Представлен прототип блока численного решения уравнений негидростатической динамики атмосферы для системы прогноза погоды нового поколения. Блок сформулирован на сетке с квази-равномерным разрешением на сфере «кубическая сфера», что в дальнейшем позволит измельчать шаг сетки сетки до 3-5 км и менее. Представленная версия блока не учитывает рельеф поверхности и использует явную схему интегрирования по времени. В дальнейшем планируется реализация версии в криволинейной вертикальной координате, огибающей рельеф, а также различных схем явно-неявного интегрирования по времени.

Прототип динамического блока испытан на ряде общепринятых идеализированных задач для тестирования блоков численного решения уравнений гидротермодинамики атмосферы. Точность разработанного блока соответствует современным мировым стандартам.

Предложенный прототип динамического блока в разрешении 20 км по горизонтали масштабируется до 4000 процессорных ядер (эффективность около 70%). Параллельная эффективность и максимальное количество ядер могут быть увеличены при использовании технологии OpenMP в сочетании с MPI. Также, необходима оптимизация обменов при вычислении значений полей в виртуальных точках, выходящих за границу блоков кубической сферы. Отметим, что при повышении разрешения до целевого (3-5 км) количество ядер, которые могут быть эффективно использованы, автоматически повысится в силу роста размерности задачи.

Представленный прототип блока решения уравнений динамики глобальной атмосферы показал результаты, которые позволяют продолжить работу по его развитию до реализации в системе прогноза погоды.

Список литературы

1. Staniforth A., Thuburn J. Horizontal grids for global weather and climate prediction models: a review // Quart. J. Roy. Met. Soc. 2012. Vol. 138. P. 1 – 26.

2. Hortal M. The development and testing of a new two-time-level semi-Lagrangian scheme (SETTLS) in the ECMWF forecast model // Quart. J.Roy. Met.Soc. 2002. Vol. 128. P. 1671–1688.

3. Vorticity-divergence semi-Lagrangian global atmospheric model SL-AV20: dynamical core / M. Tolstykh, V. Shashkin, R. Fadeev et al. // Geoscientific Model Development. 2017. Vol. 10, no. 5. P. 1961–1983. URL: https://www.geosci-model-dev.net/10/1961/2017/.

4. The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core / G. Zangl, D. Reinert, P. Ripodas et al. // Quart. J.Roy. Met.Soc. 2015. Vol. 141, no. 687. P. 563–579.

5. Harris L. M., Lin S. J. A Two-Way Nested Global-Regional Dynamical Core on the Cubed-Sphere Grid // Mon. Wea. Rev. 2013. Vol. 141. P. 283–306.

6. A New Dynamical Core of the Global Environmental Multiscale (GEM) Model with a Height-Based Terrain-Following Vertical Coordinate / S. Husain, C. Girard, A. Qaddouri et al. // Monthly Weather Review. Boston MA, USA, 01 Jul. 2019. Vol. 147, no. 7. P. 2555 – 2578.

7. Shashkin Vladimir V., Goyman Gordey S. Semi-Lagrangian exponential time-integration method for the shallow water equations on the cubed sphere grid // Russian Journal of

Numerical Analysis and Mathematical Modelling. 2020. Т. 35, № 6. С. 355–366. URL: https://doi.org/10.1515/rnam-2020-0029.

8. Goyman G., Shashkin V. Implementation of Elliptic Solvers Within ParCS Parallel Framework // Supercomputing / Ed. by V. Voevodin, S. Sobolev. Cham: Springer International Publishing, 2021. P. 137–147.

9. Arakawa Akio, Konor Celal S. Unification of the Anelastic and Quasi-Hydrostatic Systems of Equations // Monthly Weather Review. Boston MA, USA, 2009. Т. 137, № 2. С. 710 – 726. URL: https://journals.ametsoc.org/view/journals/mwre/137/2/2008mwr2520.1.xml.

10. A new dynamical core for the Met Office's global and regional modelling of the atmosphere / T. Davies, M. Cullen, A. Malcolm et al. // Quart. J.Roy. Met.Soc. 2005. Vol. 131. P. 1759–1782.

11. Thuburn John, Woollings T.J. Vertical discretizations for compressible Euler equation atmospheric models giving optimal representation of normal modes // Journal of Computational Physics. 2005. 03. Т. 203. С. 386–404.

12. Holton J. R. An introduction to dynamic meteorology. Fourth edition. Elsevier Academic Press, Int. Geophys. Ser., 2004. Vol. 88.

13. Rančić M., Purser R. J., Mesinger F. A global shallow-water model using an expanded spherical cube: Gnomonic versus conformal coordinates // Quarterly Journal of the Royal Meteorological Society. 1996. Т. 122, № 532. С. 959–982. URL: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.49712253209.

14. Arakawa A., Lamb V. Computational design of the basic dynamical processes of the UCLA general circulation model // Methods of Computational Physics. New York: Academic Press, 1977. Vol. 17. P. 173–265.

15. Del Rey Fernández David C., Hicken Jason E., Zingg David W. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations // Computers & Fluids. 2014. Т. 95. С. 171–196.

16. Strand Bo. Summation by Parts for Finite Difference Approximations for d/dx // Journal of Computational Physics. 1994. Т. 110, № 1. С. 47–67.

17. Gao Longfei, Keyes David. Explicit coupling of acoustic and elastic wave propagation in finite-difference simulations // Geophysics. 2020. Т. 85, № 5. С. T293–T308.

18. O'Reilly Ossian, Petersson N. Anders. Energy conservative SBP discretizations of the acoustic wave equation in covariant form on staggered curvilinear grids // Journal of Computational Physics. 2020. Т. 411. с. 109386.

19. Current and Emerging Time-Integration Strategies in Global Numerical Weather and Climate Prediction / G. Mengaldo, A. Wyszogrodzki, M. Diamantakis et al. // Archives of Computational Methods in Engineering. 2019. Vol. 26. P. 663 – 684.

20. Shashkin V. V., Goyman G. Parallel Efficiency of Time-Integration Strategies for the Next Generation Global Weather Prediction Model // Supercomputing - 6th Russian Supercomputing Days, RuSCDays 2020, Moscow, Russia, September 21-22, 2020, Revised Selected Papers / Ed. by V. V. Voevodin, S. Sobolev. Vol. 1331. Springer, 2020. P. 285–296.

21. URL: http://web.archive.org/web/20170212012614/https://earthsystemcog.org/projects/dcmip-2012/.

22. Numerical solutions of a non-linear density current: A benchmark solution and comparisons / J. M. Straka, R. B. Wilhelmson, L. J. Wicker et al. // International Journal for Numerical Methods in Fluids. 1993. Vol. 17, no. 1. P. 1–22.

23. An inherently mass-conserving iterative semi-implicit semi-Lagrangian discretization of the non-hydrostatic vertical-slice equations / T. Melvin, M. Dubal, N. Wood et al. // Quart. J.Roy. Met.Soc. 2010. Vol. 136. P. 799–814.

24. Idealized test cases for the dynamical cores of Atmospheric General Circulation Models / C. Jablonowski, P. H. Lauritzen, M. Taylor et al. // A proposal for the NCAR ASP 2008 summer colloquium. 2008. URL: http://www.cgd.ucar.edu/cms/pel/asp2008/idealized_testcases.pdf.

25. A standard test set for numerical approximations to the shallow water equations in spherical geometry / D. Williamson, J. Drake, J. Hack et al. // J. Comput. Phys. 1992. Vol. 102. P. 211 – 224.

Работа с данными в учебном языке программирования СИНХРО

Л.В. Городняя[1,2]

[1]Институт систем информатики СО РАН, [2]Новосибирский государственный университет

Аннотация. Статья посвящена уточнению понятий, полезных при подготовке много-поточных программ для обучения параллельному программированию. Подход сложился при создании языка СИНХРО. Дан приоритет парадигме функционального программирования, популярной при подготовке прототипов много-поточных программ. Приведено описание отличий от привычных представлений, сдерживающих решение задач организации параллельных вычислений и предельно распределённых систем из ряда потоков, взаимодействующих в терминах доступа к значениям переменных, возможно расположенных в общей памяти. Предложен механизм взаимодействия локальной и общей памяти.

Ключевые слова: дисциплина доступа к памяти, функциональное программирование, много-поточные программы, неизменяемость данных, восстановление данных, освобождение памяти.

1. Введение

В лаборатории информационных систем ИСИ СО РАН разрабатывается учебный язык программирования СИНХРО, предназначенный для ознакомления с явлениями параллелизма, мета-программированием и особенностями много-поточного программирования [1, 2]. Прагматика языка соответствует парадигме функционального программирования (ФП) без отказа от некоторых методов императивного программирования. При создании языка учтен опыт применения языков Lisp, Робик, SETL, БАРС, Sisal, Haskell, mpC, Oz. Приняты во внимания основные модели организации параллельных вычислений [3, 4] и уточнен ряд понятий и методов, изменяющихся при переходе от обычного программирования к представлению много-поточных и многопроцессорных программ. Прежде всего это преодоление зависимости порядка вычислений от последовательности вхождения выражений в текст программы. Далее изменяется базовый уровень воздействий на память. Часть из них усложнены для предотвращения неожиданностей из-за асинхронности и ослабления императивности элементов распределённых систем. Добавлено понятие команд-двойников для управления императивной синхронизацией взаимодействующих устройств.

Язык СИНХРО ориентирован на формирование навыков профилактики неудачных взаимодействий процессов, учёта равноправия независимых потоков, использования укрупнённых воздействий на общую память, прогнозирования результатов автоматизированных преобразований программ и данных, включая оптимизацию и компиляцию программ. Учтены требования отладки программ решений учебных задач. В данной статье описаны решения, принятые на уровне ядра языка программирования. Рассматриваются подходы к решению проблем работы много-поточных программ над общей памятью. Язык СИНХРО позволяет управлять компиляцией много-поточных программ в многопроцессорные. Обучение мета-программированию предполагается выполнять в форме подготовки сценариев отладки и преобразования программ, а также измерения вклада

программируемых решений в производительность программ, что выходит за пределы данной статьи.

Изложение начинается с освещения принципов ФП, затем на их основе описано уточнение отдельных понятий программирования на случай много-поточных программ, далее представлен переход от потоков к процессам и описаны вытекающие из этого решения по организации работы с общей памятью на уровне абстрактного многопроцессорного комплекса.

2. Функциональное программирование и параллелизм

Функциональное программирование - одна из первых парадигм, направленных не столько на получение эффективной реализации заранее созданных и хорошо изученных алгоритмов, сколько на решение новых и исследовательских задач, изучение которых продолжается при программировании [5-11]. Для таких задач ***правильность и полнота решений важнее эффективности и производительности полученных программ***. ФП основано на семантических принципах, таких как всеобщность (универсальность), само-применение и равноправие параметров функций. В результате функции и значения могут быть представлены такими же символами, что и любые данные для компьютерной обработки. Представления функций и значений могут использовать рекурсивные символьные формы. Порядок вычисления параметров функции не важен, они не зависят друг от друга.

ФП опирается на прагматические принципы реализации вычислений, такие как гибкость ограничений на размеры блоков памяти, неизменяемость обрабатываемых данных и строгость результата функции. Это позволяет системам ФП предотвращать простои памяти автоматизацией оперативного анализа достижимости данных, освобождения и повторного использования памяти, хранящей недостижимые данные. Представление каждого данного помещается в новую часть свободной памяти без искажения аргументов функции, они могут быть полезны для других функций. Любое количество результатов функции может быть представлено в виде единой символьной формы.

Представление алгоритмов в виде функциональных программ дает важные следствия. Из семантических принципов вытекает возможность мета-программирования, верификации и факторизации программ на автономные модули. Прагматические принципы поддерживают интуитивные модели непрерывности процессов, обратимости действий и унарных функций, служащих основой для прототипов на ранних стадиях жизненного цикла программ.

Комплекс семантических и прагматических принципов обеспечивает поддержку подготовки программ при организации параллельных вычислений, сводимых к наборам независимых потоков. Любой фрагмент, выполнение которого маловероятно, может быть перемещён из представления функции в отложенное действие. Ленивые или опережающие вычисления дают возможность оперативно перераспределять нагрузку процессоров. Любое конечное множество может работать как пространство итераций для определенной на нем функции при автоматическом распараллеливании. [12].

При переходе к многократно применяемым программам и параллельным вычислениям ***успех приложения и производительность программ становятся важнее, чем их формальная корректность и эффективность.*** В дополнение к принципам и следствиям в реальные языки и системы программирования для производственного ФП обычно включают механизмы практичного компромисса, которые, внешне не нарушая функциональный стиль

представления программ, выглядят как специальные функции. Например, Lisp 1.5, Clisp, Cmucl, Clojure и другие члены семейства языка Lisp обычно предоставляют возможности такого рода:

1. **контроль типов данных** несколько ограничивает *принцип универсальности* функциями статического и динамического анализа типов данных;
2. **схемы циклов** позволяют преодолеть типичные опасения по поводу сложности реализации *принципа само-применения*;
3. **возможность восстановления данных** позволяет исключить чрезмерное потребление памяти, аккуратно отклоняясь от *принципа неизменности данных*;
4. **программируемые прогнозы** объема памяти и скорости выполнения позволяют нейтрализовать грубоватые механизмы *принципа гибкости ограничений*;
5. **псевдофункции**, сохраняя общую схему управления вычислениями, одновременно выполняют нагрузку в виде взаимодействия с устройствами, включая ввод-вывод данных и доступ к файлам, что несколько уводит от *принципов равноправия параметров и неизменяемости данных*;
6. **мемоизация позволяет** снижать сложность многократно повторяемых вычислений, сохраняя доступ к успешному опыту ранее выполненных вычислений против *принципа строгого результата*.

Высокопроизводительное программирование требует учета перспектив многократного использования и улучшения программ [13]. Эксперименты на супер-компьютерах показали, что системные решения могут вносить значительный вклад в производительность параллельных вычислений, такой вклад может превышать теоретические оценки. Квалификация разработчика программных систем обычно включает в себя умение изобретать решения задач и навыки ответственного улучшения готовых решений.

В феврале 2021 года состоялась 22-я конференция, посвященная современным тенденциям ФП [14]. Представленные доклады убедительно показали нацеленность ФП на решение задач организации параллельных вычислений. Строго говоря, ФП способно выполнять роль проектно-конструкторского отдела для производственного программирования. В некоторых источниках по измерению трудоёмкости программирования и производительности программ утверждается, что опыт ФП повышает качество программирования в любой парадигме **[15].**

3. Подходы к обработке данных

При переходе к параллельному программированию многие привычные понятия претерпевают изменения, начиная с понятия «данное». Данными являются уже не только представления значений и функций, но ещё и процессоров, и разнообразных устройств. Такое обобщение распространяет сферу успешного программирования на параллельные вычисления и распределённые системы.

3.1. Равноправие элементов структур данных

Много-поточная программа допускает асинхронное выполнение потоков и действий, что означает в соответствии с принципом **равноправия параметров** функций независимость порядка вычисления выражений от порядка их вхождения в структуры данных, рассматриваемые как функции над этими выражениями. Для приобретения навыков учёта такой независимости в языке СИНХРО предлагается в записи структур данных разнести смысл

скобок и разделителей. Круглые скобки означают, что доступ к данным возможен последовательно, а квадратные символизируют произвольный доступ. Если разделителем в перечне данных является точка с запятой «;», то элементы перечня вычисляются последовательно, в порядке вхождения в запись выражения, а в случае запятой «,» - в произвольном порядке, независимо от порядка записи (см. таблица 1).

Таблица 1. Структуры данных

БНФ	Примечание
Структура::= '('Выр(';'Выр)…')'	Список, заполняемый последовательно вычисляемыми элементами, в порядке записи в программе
|'('Выр(','Выр)…')'	Список, заполняемый в порядке записи в программе асинхронно вычисляемыми элементами, возможно в другом порядке
|'['Выр (','Выр)…']'	Вектор, заполняемый в порядке записи в программе асинхронно вычисляемыми элементами, возможно в другом порядке
|'['Выр (';'Выр)…']'	Вектор, заполняемый последовательно вычисляемыми элементами, в порядке записи в программе

Часть проблем взаимодействия потоков алгоритмически не разрешима, поэтому в задачи ознакомления с параллелизмом входит научиться предвидеть такие опасности и отлаживать программы при обнаружении неудачных взаимодействий потоков. Наполнение много-поточной программы может развиваться независимо от схем управления вычислениями в отдельных потоках, а схемы можно реорганизовывать без дополнительной отладки наполнения в соответствии с **принципом факторизации** на автономно развиваемые модули. Схемы работают подобно макросам, но с контролем соответствия параметров объявленным видам фрагментов. Выражение может использовать размещённые в общей памяти данные, но не изменяет их. Директива обрабатывает память и при благополучном исходе даёт результат подобно выражению.

3.2. От потоков к процессам

Компилятор много-поточной программы строит функционально эквивалентную ей многопроцессорную программу для размещения на многопроцессорном комплексе. Набор потоков при компиляции обычно преобразуется в набор процессов, каждый из которых выполняется одним отдельным процессором, но может быть иначе при решении проблем балансировки нагрузки в соответствии с принципом **гибкости ограничений**.

Контекст исполнения программы при переходе к процессам — общая память, данные из неё доступны всем процессам многопроцессорной программы. Общая память содержит представления глобальных, возможно изменяемых, переменных. Существует и локальная память, соответствующая процессам, созданным по определениям потоков, функций и циклов, подчинённая иерархии определений в программе. Обработка общей и локальной памяти на уровне много-поточной программы выглядит одинаково, что позволяет при отладке перемещать фрагменты в разные позиции программы. Возможно хранение в общей памяти определений и имён. Типы именуемых данных можно устанавливать по виду их значения или функции. Имеются распознающие функции-предикаты ATOM, NUMBER, LIST, ARRAY и др.

Процесс выполняется как непустой ряд команд процессора, допускающий наращивание при исполнении. Очередная команда может начинать выполнение строго после начала предыдущей команды, но не обязана дожидаться её завершения. Каждый процессор работает

по шагам, соответствующим выполнению одной команды. После шага происходит переход к общему механизму управления многопроцессорной конфигурацией.

Компилятор по каждому потоку строит корректный ряд команд процесса, согласованных для выполнения действий потока. Завершение набора процессов может включать выполнение сверток для получения **единого результата**, что несколько влияет на их синхронизацию. Между двумя соседними командами процесса может быть выполнена команда другого процесса, что требует особого внимания при организации воздействий на общую память.

Решение учебной задачи строится в два шага. Сначала программа решения выглядит как функциональная схема, наполняемая выражениями - фрагментами вычислений без действий над общей памятью. Потом возможен гладкий переход к более эффективной версии программы методом замены некоторых выражений на директивы, т.е. чистых функций на функционально эквивалентные им псевдофункции или процедурные аналоги с воздействиями на память и синхронизацией потоков по мере необходимости – принцип **факторизации.** Для простоты изложения здесь не рассматривается разнообразие категорий систем команд процессоров и видов используемой памяти с различной дисциплиной доступа.

3.3. Абстрактный процессорный комплекс

Абстрактный процессорный комплекс способен выполнять вычисления несколько более широкого класса, чем обычно задано семантикой языка программирования. Кроме собственно процессоров к выполнению программы можно привлекать дополнительные устройства, которые рассматриваются как специальные процессоры без определённой в языке системы команд, способные выполнять некоторые действия по запросам от других процессоров. Система команд локального процессора поддерживает обработку данных, их размещение и реорганизацию в своей локальной или общей памяти, и управление ходом выполнения процессов, включая взаимодействие процессоров и устройств, и резервирование данных для их защиты от случайных изменений, подобно средствам операционных систем [16]. Программа использует общую память, данные из которой доступны отдельным процессорам, выполняющим программу. Шаги разных процессоров асинхронны, но могут происходить одновременно.

При спецификации команд процессора использовалась предложенная П.Лендиным машина SECD [7], дополненная учётом особенностей наиболее известных виртуальных машин, включая JVM [17-20]**.** SECD работает над четырьмя регистрами: S - стек для промежуточных результатов, E - контекст для размещения именованных значений, C - управляющая вычислениями программа, D - резервная память (Stack, Environment, Control_list, Dump). Регистры приспособлены для хранения выражений в форме символов или списков. Состояние машины полностью определяется содержимым этих регистров. К регистрам системы команд машины SECD добавлен регистр «M» (Memory), что даёт уточнённое обозначение многопроцессорного комплекса: **M(SECD)+ –** абстрактный многопроцессорный комплекс над общей памятью.

Это обозначение символизирует, что **M** — общая память для всех процессоров,

(SECD)+ – что хотя бы одни процессор обязателен, общее число процессоров произвольно и не исключено изменение их числа в динамике.

M – общая память, условно подчинена принципу **неизменяемости данных**, расширенному механизмом **восстановления данных** (см. 3.4). Она состоит из регистров произвольного доступа, хранящих счётчик числа потоков, использующих эти регистры, имя переменной, её текущее значение и протокол произошедших изменений, устроенный как вектор или список. Поддержано восстановление значений, оттеснённых присваиваниями.

m s e c d → m' s' e' c' d' – переход от старого состояния машины к новому.

Разница между SECD и M(SECD)+ сводится к операциям над данными в общей памяти. Регистр M является списком списков или стеков, каждый элемент которого начинается с имени глобальной переменной, вслед за которым расположено её текущее значение, а дальше следует протокол изменения значений, выглядящий как последовательность номеров процессов с каждым последующим установленным этим процессом значением. В определённом смысле состояние общей памяти можно рассматривать как непременное дополнение к формальному результату работы программы. Фактически исполняются команды по очереди, последовательно, начиная с первых в регистре управляющей программы, хотя можно считать, что они исполняются в произвольном порядке по мере готовности.

Такое определение может быть машинно-независимым и переносимым. Размер стека не ограничен. Каждая команда абстрактного многопроцессорного комплекса «знает» число используемых при ее работе элементов стека S и их форматы, элементы она удаляет из стека S и вместо них размещает один выработанный результат для обычных команд или набор результатов для многопроцессорных команд, размещая число процессов в верхнем элементе стека. Всегда известно число текущих результатов в стеке S, которые можно явно свернуть в единственный результат специальной свёрткой, необходимость в которой выясняет компилятор и размещает свёртку в регистре C абстрактной машины. Свёртка может быть встроенной в язык или программируемой, позволяющей указанное число элементов в стеке свести в одно данное. Например: список, структура, сумма, произведение, макс, мин, последний и т. п. Свёртки обычно коммутативны, за редким исключением — первый или последний по времени вычислений или позиции вхождения в программу. Фильтр можно рассматривать как частный случай свёртки.

Суммарно комплект команд включает в себя действия, часть из которых определены в книге П. Хендерсона (LD, LDC, LDF, AP, RTN, RAP, DUM, SEL, JOIN, CONS, CAR, CDR, CONS, ATOM, EQ, SUB, ADD, MUL, DIV, STOP) [7]. Этот набор команд в языке СИНХРО пополнен для решения проблем работы с общей памятью и внешними устройствами, особенности которых проявляются на командах воздействия на общую память, взаимодействия общей и локальной памяти, доступа к устройствам и организации параллельных процессов [17-20]:

LDM – **копирование** данного из общей памяти в стек;
SET – **запись** в общую память комплекса из стека;
LET-- сохранение локальных значений в общей памяти;
DEL – удаление верхнего элемента стека;
MLL – **пересылка в** головной элемент списка из другого списка;
MLV– **пересылка из списка** в указанный элемент вектора;
MVL– **пересылка из вектора** в головной элемент списка;
MVV – **пересылка в** указанный элемент вектора из другого вектора;

CHNG – обмен данными в общей памяти комплекса.

Технические детали определения этих и остальных команд приведены в статье [21].

3.4. Восстановление данных

Для языка СИНХРО уточнён принцип неизменяемости данных, характерный для ФП. Работа с памятью поддержана в транзакционном стиле, подобно обработке записей в базах данных, т. е. каждое воздействие на память или выполняется полностью, или восстанавливается состояние до начала выполнения не завершившейся команды [22]. Каждый элемент общей памяти в любой момент времени обрабатывается только в одном процессе программы, подобно захвату-освобождению файла. Хранение данных в общей памяти сопровождается протоколом изменений, чтобы при отладке можно было видеть какой процесс внёс изменения и, при необходимости, вернуть утраченное значение.

Возможен побочный эффект присваивания, допускающий при необходимости восстановление прежних значений переменных. Такая реализация позволяет поддержать транзакции и обратимость обработки памяти при отладке. В стеке размещается список результатов однократных присваиваний. К верхнему элементу контекста прицепляется этот список в хвост, вслед за аргументами. Предполагается, что имена формальных параметров и локальных данных различны.

Усложнённые команды пересылки и обмена данными в общей памяти (MLL, MLV, MVL, MVV, CHNG) нужны для профилактики возникновения временных интервалов между взаимосвязанными присваиваниями в общей памяти. Иначе может возникать так называемая «фантомная» память или доступ к формально уже удалённым данным, что иногда обнаруживается при использовании JVM в сетях, стандарт на которую был утверждён более 20-ти лет назад.

3.5. Внешние устройства и процессоры

При работе с устройствами через конечное время вырабатывается сигнал, говорящий или об успешном выполнении действия, или о причине отказа в его завершении. Выработка сигнала об отказе внешнего устройства приводит к запоминанию в протоколе информации об ущербно выработанных результатах, что можно учесть при отладке и выполнению остаточной программы в стиле смешанных вычислений.

WRT – вывод данных из стека на внешнее устройство с сигналом успеха-провала;
RD – ввод данных от внешнего устройства в стек с сигналом успеха-провала;
ANY – выравнивание стека, если данное с устройства не введено из-за провала.

Для организации параллельных вычислений требуются ещё команды:

KIT – комплекс из процессоров для выполнения действий;
ROW – ряд действий на одном процессоре;
REZ – **размещение** или использование заданного числа результатов процесса в свой стек;
WAIT – приостановка с ожиданием указанного процесса (завершения или сообщения);
SEND – **сообщение** указанному процессу;
NEXT – ожидание события или завершения действия указанного процесса.

Так обеспечивается подключение комплексов процессоров для неупорядоченных наборов потоков, мощность комплекса известна. На каждом процессоре происходит размещение процессов в виде ряда действий, длина ряда известна в каждый момент. Предполагается выполнение последовательности действий ряда на одном процессоре, завершаемых передачей результатов процесса в стек с локализацией воздействий на транзакционную общую память и явной обработкой ошибок.

Реализация передачи сообщений подобна рандеву в языке ADA – обмен происходит между двумя активными процессами и каждый знает что и кому он передаёт или от кого сообщение получает. Процесс-отправитель перед выполнением SEND может "дожидаться ожидания" получателя, то есть проверять, что получатель сейчас выполняет команду WAIT, иначе ожидать её, чтобы произошло "рандеву". Процесс может не знать, что его завершения кто-то ждет.

Многопроцессорная программа считается идеальной, компилятор её строит без учёта возможных аварийных ситуаций, считает, что все данные на устройстве ввода расположены в соответствии с запросами программы, именно те, что нужны. Состояния стеков к моменту выполнения команд сформированы вполне корректно. Регистры локальной памяти подчинены принципу **неизменяемости данных**. Новые данные размещаются в первом элементе списка, а к прежним значениям, хранимым в общей памяти, можно вернуться при необходимости, например, при отладке посмотреть историю изменения данных. При выполнении обратимых действий поддержана и обратимость воздействий на общую память. Идеальная конфигурация многопроцессорной программы работает на как бы бесконечной общей памяти, не решает проблемы её исчерпания и необходимости освобождения от ставших ненужными данных. Поскольку процессоры рассматриваются как одна из категорий данных, то можно от понимания памяти как одного из регистров машины перейти к отдельному процессору общей памяти, обладающему своей системой команд.

3.6. Общая память

Освобождение общей памяти требует механизмов, подобных опробованным в практике операционных систем при решении вопросов управления распределёнными системами с совмещением работы независимых программ [16]. Можно рассмотреть определение многопроцессорного комплекса, допускающее управление доступом к общей памяти с использованием паспортов, хранящих шкалу необходимых регистров общей памяти для доступа к глобальным переменным. Паспорт одновременно доступен и локальному процессору, и процессору общей памяти. В этом случае локальный процесс при «сборке мусора» формирует новое состояние такой шкалы, доступной процессору общей памяти. Локальные процессы вместо произвольного доступа к общей памяти обладают лишь шкалой для доступа к определённым регистрам. При освобождении общей памяти появляется возможность частичного освобождения памяти, не задействованной в объединении шкал регистров от локальных процессов, без приостановки всех процессоров. В частности, при отказе отдельного процессора можно по шкале глобальных переменных выполнить уменьшение счётчиков доступа к этим переменным для решений по освобождению или восстановлению памяти.

Несколько проще выглядит модель комплекса со специальным процессором общей памяти, поддерживающим **императивную синхронизацию**. Такой процессорный комплекс состоит из ряда обычных процессоров и одного процессора общей памяти, с которым могут взаимодействовать локальные процессоры. Между собой они взаимодействуют только через

общую память. Каждый процессор выполняет одну последовательность команд, среди которых встречаются команды запросов к процессору общей памяти. Это **активные команды**, выполняемые по ходу процесса без особенностей. Команды запроса к общей памяти имеют **двойников** среди команд процессора общей памяти. Это **пассивные команды**, работающие как ленивые вычисления, возбуждаемые при выполнении запросов от локальных процессоорв. Пассивные команды процессора общей памяти собраны в ряд очередей, каждая из которых соответствует локальному процессору и содержит двойников в том же порядке, что и последовательность запросов от активных команд. Пара запрос и его двойник выполняются неразрывно в стиле рандеву языка Ada. Как и при пересылках, механизм рандеву исключает случайное вмешательство сторонних процессов. При этом происходит обмен данными между локальной памятью активного процессора и общей памятью пассивного процессора (см. Таблица 2).

Таблица 2 . Дублеты - парные команды — срабатывают только неразрывно вместе

Активные команды локальных процессоров	**Пассивные команды-двойники процессора общей памяти, ждущие запросов от локальных процессоров**
GIVE – запрос регистра для переменной в общей памяти	**TAKE –** выбор регистра для переменной, счётчик кратности доступа к нему его надо увеличить на 1.
SAFE – запрос на хранение данного в переменной в общей памяти	**WRITE** – размещение данного. В протокол вносится указатель на данное и номер процесса.
READ – запрос на чтение данного	**VAR –** доступ процесса к данному из общей памяти
UNDO – запрос на получение предыдущего значения переменной	**UNDO –** доступ к предыдущему значению переменной, состояние регистров переменной не меняется.
FREE – объявление, что больше переменная не нужна.	**FREE** – освобождение памяти, счётчик кратности доступа уменьшается на 1.

Для локальных процессоров возможны запросы на регистр для переменной, на хранение данного в переменной по её регистру, на чтение данного из переменной по регистру, на получение предыдущего данного из переменной по адресу и на объявление, что больше переменная не нужна.

Команды-двойники процессора общей памяти, пассивно ждущего запросов от локальных процессоров или отладчика, одновременно с выполнебнием активных команд выполняют выбор регистра для переменной с указанным номером (счётчик кратности доступа к нему надо увеличить на 1), размещение данного (если данное — указатель на вектор или список, то они копируются полностью в общую память и в протокол вносится копия указателя на данное и номер активного процесса), чтение данного для передачи локальному процессору (состояние регистра не меняется), чтение предыдущего данного, освобождение памяти (счётчик кратности доступа уменьшается на 1).

Процессор общей памяти может функционировать как элемент распределённой системы, включающийся по мере поступления запросов от локальных процессоров, и не требующий приостановки всех процессоров на время решения проблемы освобождения памяти. Представленный здесь метод использует ряд процессоров со своей локальной памятью и процессор общей памяти. Локальные процессоры могут работать автономно, по мере необходимости обращаясь к общей памяти и время от времени освобождая свою локальную

память. В локальной памяти хранится шкала регистров доступа к общей памяти, которая может то расширяться, то сужаться по мере инструкций из программы или сужаться в результате «сборки мусора». Общая память состоит из двух частей — регистров прямого доступа и кучи для хранения протоколов, а также структур данных и старых значений после присваивания на случай необходимости восстановления состояний памяти. Время от времени при исчерпании той или иной области общей памяти или просто по графику запускается алгоритм, похожий на «Stop&Copy» в системах ФП. Система команд абстрактного комплекса образует два уровня на каждом из которых можно видеть подсистемы организации вычислений, работы с памятью, управления процессами и структурирования данных. Один уровень реализует работу отдельного потока над локальной памятью, другой поддерживает взаимодействие потоков над общей памятью и внешними устройствами.

4. Заключение.

На этапе ознакомления с проблемами параллелизма можно строить и отлаживать небольшие много-поточные программы взаимодействия процессов, выполняемых на модели произвольной многопроцессорной конфигурации над общей памятью. При обучении параллельному программирования опыт такой работы поможет видеть и понимать разные явления, происходящие при взаимодействии потоков, предвидеть проблемы и накапливать рецепты решения проблем, возникающих при подготовке и отладке много-поточных программ, что особенно ярко может влиять на переход к суперкомпьютерам [23]. Главный результат ознакомления — формирование интуитивных понятий, соответствующих реальной сложности параллельного программирования. Для этих целей предлагается принцип неизменяемости данных, характерный для ФП, распространить на работу с переменными в общей памяти и дополнить его механизмом восстановления данных, используя протоколы изменения значений переменных. Проблему освобождения общей памяти может быть решена как композиция методов «сборки мусора» и учёта кратности доступа к переменным из потоков, взаимодействующих через использование данных, хранящихся в общей памяти.

Автор благодарен Дмитрию Мажуге, выполнившему для языка СИНХРО реализацию виртуального многопроцессорного комплекса на языке Clojure.

Литература

1. Городняя Л.В. Язык параллельного программирования СИНХРО, предназначенный для обучения // Новосибирск, Препринт ИСИ СО РАН № 180, 2016, 30 с.
2. Городняя Л.В. О курсе «Начала параллелизма» // Ершовская конференция по информатике. Секция «Информатика образования». 27 июля 2011 года. Новосибирск. с. 51-54.
3. Воеводин В. В. Параллельные вычисления. / В. В. Воеводин, Вл. В. Воеводин. – СПб.: БХВ-Петербург, 2002. 608 С.
4. Хоар Ч. Взаимодействующие последовательные процессы /Издательство «Мир», 1989, 264 с.
5. McCarthy J. LISP 1.5 Programming Manual /J. McCarthy. The MIT Press, Cambridge, 1963. 106 p.
6. Backus J. Can programming be liberated from the von Neumann style? A functional stile and its algebra of programs. – Commun. ACM 21, 8, 1978, – p.613-641.
7. Хендерсон П. Функциональное программирование. – М.: Мир, 1983. – 349 с.

8. Лавров С.С. Функциональное программирование. // Компьютерные инструменты в образовании. – 2002, N 2-4.

9. Лавров С.С., Городняя Л.В. Функциональное программирование. Принципы реализации языка Лисп. // Компьютерные инструменты в образовании. – 2002, N5, с. 49-58

10. Городняя Л.В. Основы функционального программирования. – М.: Интернет-Университет Информационных технологий. − http://www.intuit.ru, 2004. − 272 с. http://www.intuit.ru/studies/courses/29/29/info

11. Городняя Л.В. Первые реализации языка Lisp в СССР // Материалы второй Международной конференции Развитие вычислительной техники и ее программного обеспечения в России и странах бывшего СССР (SoRuCom-2011) с. 95-100 https://www.computer-museum.ru/histsoft/lisp_sorucom_2011.htm

12. Cann D. C. SISAL 1.2: A Brief Introduction and tutorial. Preprint UCRL-MA-110620. Lawrence Livermore National Lab., Livermore – California, May, 1992. – 128 p.

13. Бурдонов И.Б., Косачев А.С. Семантики взаимодействия с отказами, дивергенцией и разрушением. Часть 2. Условия конечного полного тестирования // Вестник Томского государственного университета, № 2(15), 2011.

14. Pieter Koopman, Steffen Michels, Rinus Plasmeijer. Dynamic Editors for Well-Typed Expressions // Trends in Functional programming/ 22nd International Symposium, TFP 2021, February 17–19, 2021. Springer, LNCS 12834. P. 44–66.

15. Erann Gat Lisp as an Alternative to Java https://flownet.com/gat/papers/lisp-java.pdf

16. Иртегов Д.В. Введение в операционные системы СПб.: БХВ-Петербург, 2008. – 1040 с.: ил. – ISBN 978-5-94157-695-1.

17. Кнут Д.Э. Искусство программирования, том 1, выпуск 1. MMIX — RISC-компьютеры нового тысячелетия. /М.: Вильямс, 2017, 160 с.

18. Вирт Н. Построение компиляторов. М.: ДМК Пресс, 2010. ISBN 978-5-94074-585-3, 0-201-40353-6

19. Айлиф Дж. Принципы построения базовой машины. − М.: Мир, 1973. − 119 с.

20. Эванс Б., Гоф Дж, Ньюленд К. Java: оптимизация программ. Практические методы повышения производительности приложений в JVM. — М.: Диалектика, 2019. — 448 с. — ISBN 978-5-907114-84-5.

21. Городняя Л. В. Абстрактная машина языка программирования учебного назначения СИНХРО // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, No 4. С. 16–35.

22. Грабер М. Введение в SQL. − М.: Лори, 1996. − 377 с.

23. Левин В.К. Отечественные суперкомпьютеры семейства МВС. URL: http://parallel.ru/mvs/levin.html .

Решение трудоемких задач многомерной глобальной оптимизации с использованием набора инструментов Intel oneAPI [1*]

К.А. Баркалов, И.Г. Лебедев, Я.В. Кольтюшкина

Нижегородский государственный университет им. Н.И. Лобачевского

В рамках данной статьи рассматривается решение серии сложно вычислимых задач глобальной оптимизации. В качестве целевой выбирается функция, удовлетворяющая условию Липшица с заранее не известной константой. Инструментом для реализации параллельного алгоритма выбран Intel oneAPI, который позволяет писать один код как для центрального процессора, так и для графического ускорителя. В решении многомерных задач применяется метод, построенный на идее редукции размерности при помощи кривой Пеано, которая непрерывно и однозначно отображает отрезок вещественной оси на гиперкуб.

Ключевые слова: глобальная оптимизация, параллельные вычисления, многоэкстремальные функции, редукция размерности, графические ускорители, Intel oneAPI.

1. Постановка задачи

Задачи поиска экстремумов многоэкстремальных функций часто возникают в различных областях науки, таких как химия, физика, машиностроение и т.д. В рамках данной работы решается задача минимизации многомерной функции $\varphi(y)$ в гиперинтервале $D = \{y \in R^N : a_i \leq y_i \leq b_i, 1 \leq i \leq N\}$. Будем предполагать, что информация о виде самой функции недоступна, целевая функция задается в формате «черного ящика». Дополнительно для целевой функции предполагаем выполнение условия Липшица, причем константа L априори неизвестна.

$$\varphi(y^*) = min\{\varphi(y) : y \in D\}, \tag{1}$$

$$|\varphi(y_1) - \varphi(y_2)| \leq L||y_1 - y_2||, \qquad \forall y_1, y_2 \in D, \tag{2}$$

$$\frac{|\varphi(y_1) - \varphi(y_2)|}{||y_1 - y_2||} \leq L, \quad 0 < L < \infty. \tag{3}$$

Сведение многомерной задачи (1) к одномерной проводится за счет использования редукции размерности при помощи кривой Пеано $[1, 2]$ $y(x)$, которая непрерывно и однозначно отображает на N-мерный гиперкуб отрезок вещественной оси [0,1]:

$$\{y \in R^N : -2^{-1} \leq y_i \leq 2^{-1}, \ 1 \leq i \leq N\} = \{y(x) : 0 \leq x \leq 1\}. \tag{4}$$

$$\varphi(y) = \varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0,1]\}. \tag{5}$$

У данного способа редукции размерности имеется важное свойство, необходимое для последующих расчетов: ограниченность относительных разностей функции сохраняется, т.е.

[1*] Работа выполнена при поддержке программы Центра компетенций OneAPI в ННГУ, Министерства науки и высшего образования РФ (проект № 0729-2020-0055) и научно-образовательного математического центра «Математика технологий будущего» (проект № 075-02-2021-1394).

если в области D функция $\varphi(y)$ удовлетворяла условию Липшица, то на интервале $[0,1]$ функция $\varphi(y(x))$ будет удовлетворять равномерному условию Гельдера

$$|\varphi(y(x_1)) - \varphi(y(x_2))| \le H\,|x_1 - x_2|^{\frac{1}{N}}, x_1, x \in [0,1], \quad (6)$$

$$H = 4Ld\sqrt{N}, d = \max\{b_i - a_i : 1 \le i \le N\}. \quad (7)$$

Пользуясь этим свойством, можно трактовать исходную задачу, как минимизацию одномерной гельдеровой функции $f(x) = \varphi\big(y(x)\big),\ x \in [0,1]$.

Факт того, что функция задана как «черный ящик», сокращает число подходящих алгоритмов. Для исследования выбран алгоритм глобального поиска [3, 4], который является одним из эффективных методов глобальной оптимизации, т.к. при решении многих задач опережает (по числу итераций) другие методы аналогичного назначения [5 – 9]. Алгоритм глобального поиска относится к классу характеристических алгоритмов оптимизации, хорошо масштабируется и отлично подходит для работы на параллельных вычислительных кластерах.

В процессе своей работы алгоритм порождает последовательность точек x^k, в каждой из которых проходит вычисление значения минимизируемой функции $z^k = f(x^k)$. Далее под *испытанием* мы будем понимать подразумевать процесс вычисления значения одномерной функции, в который, помимо прочего, включается построение образа $y^k = y(x^k)$, а *результатом испытания* назовем пару (x^k, z^k). Процесс распараллеливания построен таким образом, что при выполнении одной итерации метода одновременно проходят P испытаний, $P \ge 1$. Введем обозначение $k(n)$, как общее число испытаний, которые были проведены после n выполненных параллельных итераций. Приведем последовательность этапов параллельного алгоритма глобального поиска с модифицированием для решения задач с функциями, удовлетворяющими условию Гельдера:

Подготовительный этап. Изначально проводится испытание в произвольной внутренней точке $x^1 \in [0,1]$. Когда выполнено $n \ge 1$ итераций, и соответствующее им число $k = k(n)$ испытаний в точках $x^i, 1 \le i \le k$, то точки $x^{k+1}, \dots, x^{k+P}$ испытаний последующей $(n+1)$ итерации будут определять следующим образом:

1 этап. Упорядочить по координате точки уже проведенных испытаний и граничные точки:

$$0 = x_0 < x_1 < \dots < x_{k+1} = 1 \quad (8)$$

2 этап. Вычислить текущее значение M, оценивающее неизвестную константу Липшица L:

$$\mu = max\left\{\frac{|z_i - z_{i-1}|}{(x_i - x_{i-1})^{1/N}}, i = 1, \dots, k\right\}, M = \begin{cases} r\mu, \mu > 0, \\ 1, \mu = 0, \end{cases} \quad (9)$$

где $r > 1$ заданный параметр метода.

3 этап. Найти значение характеристики $R(i)$ для всех интервалов $(x_{i-1}, x_i),\ 0 < i < k+1$,

$$R(1) = 2\Delta_1 - 4\frac{z_1}{M} \quad (10)$$

$$R(k+1) = 2\Delta_{k+1} - 4\frac{z_k}{M} \quad (11)$$

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{M^2\Delta_i} - 2\frac{z_i + z_{i-1}}{M}, 1 < i < k+1 \quad (12)$$

где $\Delta_i = (x_i - x_{i-1})^{1/N}$.

4 этап. Упорядочить характеристики $R(i), 1 \le i \le k+1$ в порядке невозрастания

$$R(t_1) \ge R(t_2) \ge \dots \ge R(t_k) \ge R(t_{k+1}) \quad (13)$$

и выбрать P интервалов с номерами $t_j, 1 \le j \le P$, значение характеристики в которых наибольшее.

5 этап. В выбранных интервалах вычислить точки $x^{k+j}, 1 \le j \le P$ и провести в них новые испытания:

$$x^{k+j} = \frac{x_{t_j} + x_{t_j-1}}{2}, t_j = 1, t_j = k+1 \quad (14)$$

$$x^{k+1} = \frac{x_{t_j} + x_{t_j-1}}{2} - sign\left(z_{t_j} - z_{t_j-1}\right)\frac{1}{2r}\left[\frac{\left|z_{t_j} - z_{t_j-1}\right|}{\mu}\right]^N, 1 < t_j < k+1 \quad (15)$$

Важно отметить, что в рассматриваемых задачах самый трудоемкий и времязатратный этап алгоритма – это вычисление значения функции в точке. Поэтому параллельное вычисление этих значений положительно влияет на скорость работы алгоритма.

Используемые критерии остановки:

- по длине интервала, как только происходит выполнение условия $\Delta_{t_j} \le \varepsilon$ хотя бы для какого-то номера $t_j, 1 \le j \le P$. Основной критерий остановки, используется в прикладных задачах оптимизации и в тестовых наборах, для которых точка глобального минимума заранее неизвестна.
- по попаданию в окрестность глобального минимума, $\left|x_{t_j} - x^*\right| < \varepsilon,\ t_j, 1 \le j \le P$, где x^* – точка глобального минимума, если ее значение известно. Используется в тестовых задачах, когда x^* известна до начала работы алгоритма.

Как оценка глобально-оптимального решения рассматриваемой задачи (1) выбираются значения

$$f_k^* = \min_{1 \le i \le k} f\left(x^i\right),\ x_k^* = \arg\min_{1 \le i \le k} f\left(x^i\right), \quad (16)$$

2. Реализация с использованием Intel oneAPI

Рассматриваемые многомерные многоэкстремальные задачи обладают высокой трудоемкостью численного решения, поскольку при увеличении размерности задачи наблюдается экспоненциальный рост вычислительных затрат. Вместе с тем быстрый процесс развития современных вычислительных средств, включая распараллеливание, предоставляет всё больше различных новых возможностей для решения проблем оптимизации. Однако на фоне этого возникает задача эффективного распараллеливания, а многообразие различных типов современных ускорителей и средств разработки для них дают пользователям немалый выбор [10].

Но перейдя от частной задачи к общей ситуации, разберем, какие предлагаются варианты решения. Для обеспечения высокой производительности вычислений в рамках каких-то новых рабочих задач требуются различные вычислительные архитектуры. Например, компанией Intel предлагается следующая классификация ускорителей: скалярные (CPU), векторные (GPU), матричные и ПЛИС (FPGA). Эти классы отличаются архитектурой, которая прямым образом влияет на их функционал.

Рассмотрим их по порядку. Скалярные (CPU), являются кэш-ориентированными, основаны на оптимизации однопоточной производительности и на решение задач общего назначения. Несмотря на то, что большинство процессоров сейчас многоядерные, достаточно много ресурсов тратится на обеспечение их однопоточной производительности. В GPU большая часть ресурсов задействована под вычисления [11], а не под кэш, как в случае с CPU, поэтому в GPU применима стратегия SIMD[12]. Третий класс – это матричные процессоры, рассчитанные на быструю работу с матрицами. Прежде всего это ускорители из области искусственного интеллекта, нейронные процессоры, например, процессоры машинного зрения, тензорные и другие. И последний класс – программируемые логические интегральные схемы (ПЛИС), FPGA [13]. Основу структуры составляет матрица логических элементов, функции этих элементов и связи между ними могут модифицироваться – программироваться, в процессе использования. Сфера применения ПЛИС достаточно широка, они используются в бытовой

электронике, телекоммуникационном оборудовании, разнообразной робототехнике и при прототипировании микросхем.

Однако использование преимуществ нескольких типов архитектур является сложной задачей для разработчиков. Для каждой архитектуры требуются разные языки, отдельные инструменты, а повторное использование кода ограничено. Это делает разработку сложной, дорогостоящей и трудоемкой.

Например, когда возникает потребности выполнения алгоритма на нескольких типах ускорителях, то при написании кода программы, возникают проблемы связанные с несовместимости разных средств разработки и разными архитектурными особенностиями. Для более четкого понимания этого, проведем небольшой обзор инструментов и средств разработки для программирования ускорителей. Для GPU часто используются графические API и шейдерные языки: DirectX, OpenGL, Vulcan, Metal. Некоторые производители, к примеру, NVidia и AMD, создают свои специальные средства, которые подходят под их ускорители (NVidia CUDA, AMD ROCm). Для программирования под FPGA применяются языки описания архитектур, например, Verilog и VHDL. Также есть набор общих средств, которые отличаются по применимости, сложности написания кода, текущей поддержке: OpenMP, OpenCL, Python, SYCL и другие.

Как правило, если остановиться на одной из вычислительных архитектур, то для возможности запуска на абсолютно другой может потребоваться адаптация части кода, а может даже придется написать программу с нуля, применяя другие инструменты. В большинстве случаев, будет создан новый проект, и необходимо будет поддерживать несколько программ, в которых используются разные технологии.

Для создания универсального кода работающего на различных устройствах можно воспользоваться набором инструментов Intel oneAPI. Он прост, открыт и позволяет разработчику обеспечивать высокую производительность в разных архитектурах. А поскольку oneAPI основан на стандартах и открытых спецификациях, риски при переносе снижаются. Это дает возможность один раз написать код и в дальнейшем запустить его на другом устройстве. Также к преимуществам oneAPI можно отнести возможность применения в различных прикладных решениях, например, в задачах машинного обучения.

В рамках данной задачи необходимы возможности распараллеливания, которые обеспечиваются за счет включения в oneAPI языка Data Parallel C++ и набора библиотек, облегчающих межархитектурную разработку. Data Parallel C++ основан на широко известном языке C++ и включает в себя SYCL от группы Khronos и расширения от комьюнити.

Все это позволяет повторно использовать код в разных архитектурах и выполнять пользовательскую настройку ускорителей. Что дает разработчикам гибкость, позволяющую отказаться от патентованных подходов, и открывает возможности для использования аппаратных средств, которые ранее были невозможны.

Руководствуясь приведенным ранее алгоритмом, реализована возможность одновременно вычислять сразу несколько значений целевой функции, используя при этом инструменты распараллеливания Intel oneAPI. Также имеется возможность выбрать устройство, на котором будут проходить вычисления значений функции. Остальные этапы рассмотренного алгоритма необходимо проводить последовательно, потому что в процессе их выполнения проходит работа с достаточно большим количеством накопленной ранее поисковой информации, в связи с этим реализуем их на CPU.

Таким образом, первые этапы рассматриваемого алгоритма выполняются на центральном процессоре (CPU). Далее, полученные в ходе выполнения одной итерации новые P координат из интервалов с наибольшими характеристиками, передаются с помощью промежуточного буфера на устройство (CPU, GPU, FPGA), выбранное для исполнения распараллеленного блока кода, для вычисления значения функции в них. Затем найденные значения функции в этих точках передаются через промежуточный буфер обратно на центральный процессор. Общая схема организации параллельных вычислений приведена на рис. 1.

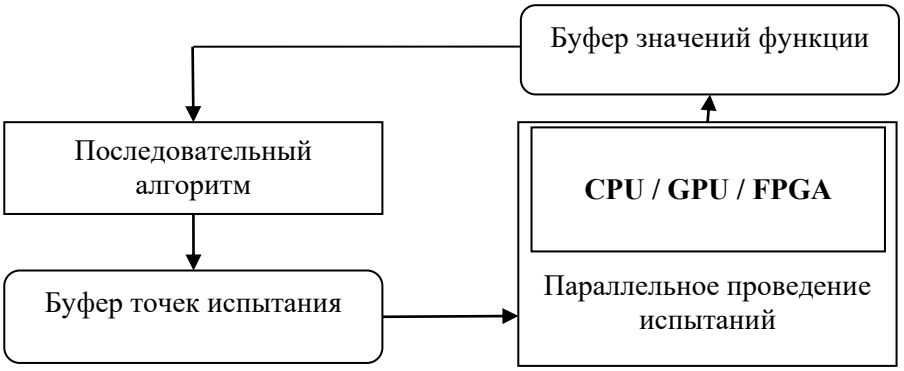


Рис. 1. Общая схема организации параллельных вычислений.

3. Результаты численных экспериментов

В рамках работы были проведены вычислительные эксперименты на персональном компьютере с процессором Intel Core i5-10600 3.3GHz, 32 GB оперативной памяти и встроенной графической картой Intel UHD Graphics 630. Для получения исполняемого программного кода использовался компилятор Intel oneAPI DPC++ 2021.1. Вычислительные эксперименты выполнялись с использованием разработанного программного комплекса Globalizer [14, 15].

Большинство известных тестовых задач из области многомерной глобальной оптимизации характеризуются небольшим временем вычисления значений целевой функции. Из-за этого при проведении экспериментов бывает сложно понять с чем может быть связано относительно большое время выполнения параллельного алгоритма: с появившимися, сравнительно с последовательной версией, накладными расходами или причина в неэффективном распараллеливании. В реальных же задачах вычисление функции самая трудоемкая и время затратная операция, поэтому такой проблемы не возникает.

Нами предлагается вычислительно трудоемкая модификация существующих тестовых задач, расчет целевой функции заключается в интегрировании исходной тестовой функции по части параметров. Для этого изначально порождается задача, размерность которой в два раза больше искомой. При вычислении значения функции, первые N координат фиксируются, а по остальным производится численное интегрирование по области определения функции. Для интегрирования используется метод средних прямоугольников.

$$\varphi(y_1, \dots, y_N) = \int_{a_{N+1}}^{b_{N+1}} \int_{a_{N+2}}^{b_{N+2}} \dots \int_{a_{2N}}^{b_{2N}} \psi(y_1, \dots, y_N, y_{N+1}, .., y_{2N})\, dy_{N+1} dy_{N+2} .. dy_{2N}$$

$$= \sum_{i_1=0}^{M-1} \sum_{i_{2=0}}^{M-1} .. \sum_{i_N=0}^{M-1} \left(\left(\prod_{j=1}^{2N} h_j \right) \psi(y_1, y_2, .., y_N, y_{(N+1)_{i_1}}, y_{(N+2)_{i_2}}, .., y_{(2N)_{i_N}}) \right) \tag{17}$$

$$D = \{y \in R^{2N} : a_i \leq y_i \leq b_i, 1 \leq i \leq 2N\}.$$

$$y_{(N+k)_{i_k}} = a_k + i_k * h_k + \frac{h_k}{2} \tag{18}$$

$$h_k = \frac{b_k - a_k}{M}, \tag{19}$$

где M – количество участков интегрирования по одной координате, а ψ – исходная тестовая функция. Очевидно, чем больше значение M, тем больше проводится вычислений. Изменяя число узлов в сетке интегрирования по всем координатам или число участков по одной, можно регулировать время выполнения вычислений.

Вначале проведем вычислительные эксперименты на классической задаче – функции Растригина. Она задается простой формулой, сепарабельна и не имеет ограничений по размерности.

$$\psi(y_1, \dots, y_{2N}) = \sum_{i=1}^{2N} (y_i^2 - 10\cos(2\pi y_i) + 10) \qquad (20)$$
$$-2.2 < y_i < 1.8\,, 1 \le i \le 2N$$

На рис. 2 изображены линии уровня двухмерной функции Растригина (слева) и интегрированной четырехмерной функции Растригина (справа). Как можно видеть, новая функция не сильно отличается от двухмерного оригинала, и даже глобальный минимум остался прежним – точка (0, 0).

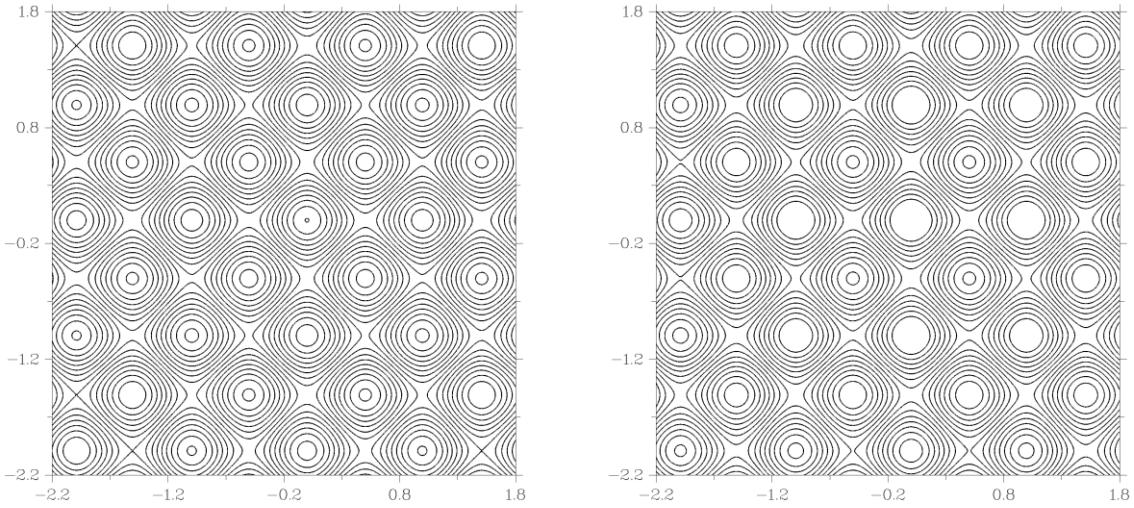


Рис. 2. Линии уровня двумерной функции Растригина (слева) и интегрированной четырехмерной функции Растригина (справа).

Число итераций параллельного алгоритма глобального поиска (ПАГП) было ограничено 1000000, точность поиска $\varepsilon = 0.01$ и параметр метода $r = 3$. Размерность задачи N = 4 и 5. При вычислениях на CPU число потоков варьировалось от 1 до 8, а при вычислении на GPU от 256 до 1024. Использовался критерий остановки по попаданию в окрестность. В таблице 1 приведено число итераций ПАГП, в таблице 2 – ускорение по сравнению с однопоточным запуском.

N	CPU					GPU		
	P=1	*P=2*	*P=4*	*P=8*		*P=256*	*P=512*	*P=1024*
4	61231	26592	21007	6728		340	301	92
5	703548	328141	258351	99524		4642	2194	995

Таблица 1. Число итераций ПАГП при решении интегрированной функции Растригина.

N	CPU				GPU		
	P=2	*P=4*	*P=8*		*P=256*	*P=512*	*P=1024*
4	2.1	2.1	5.6		2.8	3.1	9.4
5	1.9	2.0	4.3		2.4	4.9	9.7

Таблица 2. Ускорение ПАГП при решении интегрированной функции Растригина

Как можно видеть из полученных результатов, использование инструментов Intel oneAPI, при распараллеливании алгоритма глобального поиска, показала хорошие результаты на тестовой функции.

Далее проведем эксперименты на серии задач полученных интегрированием функций из генератора GKLS. Данный генератор описан в работах [16, 17], дает возможность создавать серии задач многоэкстремальной оптимизации и заранее задавать их свойства, такие как: размерность задачи, количество локальных минимумов, размеры их областей притяжения, координата точки глобального минимума, значение функции в ней и т.п. На рис. 3 изображены линии уровня двухмерной функции GKLS номер 9 (слева) и интегрированной функции GKLS номер 9, полученной интегрированием четырехмерной функции по последним двум параметрам (справа).

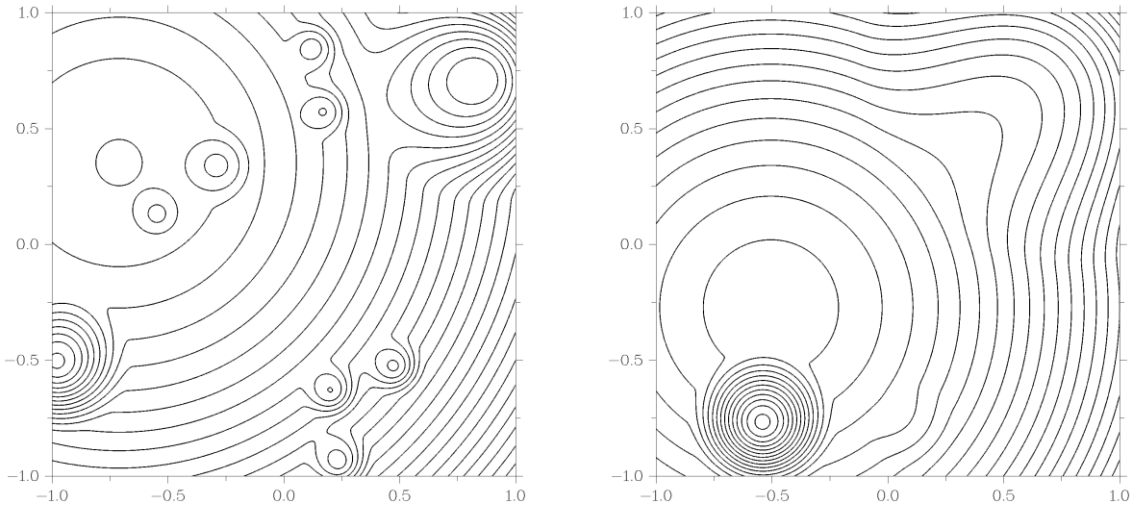


Рис. 3. Функции GKLS (слева) и интегрированной функции GKLS (справа).

Число итераций ПАГП было ограничено 100000, точность поиска $\varepsilon = 0.01$ и параметр метода $r = 4$. Размерность задачи $N = 4$ и 5. При вычислениях на CPU число потоков варьировалось от 1 до 8, а при вычислении на GPU от 256 до 1024. В таблице 3 приведено ускорение по сравнению с однопоточным запуском.

N	CPU				GPU		
	P=2	*P=4*	*P=8*		*P=256*	*P=512*	*P=1024*
4	1.9	3.0	1.5		1.2	2.3	4.5
5	1.9	3.0	1.6		1.2	2.3	4.5

Таблица 3. Ускорение ПАГП при решении серии интегрированных функций GKLS.

Заключение

Написать одну программу, которая в дальнейшем может запуститься на различных устройствах, очевидно проще, чем создавать для каждого инструмента свой проект. При использовании Intel oneAPI, не потребуется портирование программы на другое устройство, и можно будет избежать связанных с ним проблем. В современном мире данный подход приобретает всё большую актуальность, так как быстрый технологический прогресс способствует более частому изменению архитектуры устройств, их усовершенствованию, появлению новых технологий. Всё это только усложняет процесс переноса программ.

Рассмотренный в рамках данной работы параллельный алгоритм решения задач многомерной многоэкстремальной оптимизации был реализован с использованием набора инструментов Intel oneAPI. Для проверки работоспособности параллельного алгоритма была предложена вычислительно трудоемкая модификация существующих тестовых задач.

Эксперименты, проведенные на разных архитектурах, подтвердили целесообразность использования инструментов Intel oneAPI для распараллеливания алгоритмов глобальной оптимизации.

Литература

1. Sergeyev, Ya.D. Introduction to global optimization exploiting space-filling curves / Ya.D. Sergeyev, R.G. Strongin, D. Lera – Springer, 2013. 125 p.
2. Strongin, R.G. Global Optimization with Non-convex Constraints. Sequential and Parallel Algorithms / R.G. Strongin, Ya.D. Sergeyev – Kluwer Academic Publishers, 2000. 704 p.
3. Стронгин, Р.Г. Параллельные вычисления в задачах глобальной оптимизации / Р.Г. Стронгин, В.П. Гергель, В.А. Гришагин, К.А. Баркалов – М.: Издательство Московского университета, 2013. 280 с.
4. Pinter (Ed.), J.D. Global Optimization: Scientific and Engineering Case Studies / J.D. Pinter – Springer, 2006. 546 p.
5. Стронгин, Р.Г. Параллельные методы решения задач глобальной оптимизации / Р.Г. Стронгин, В.П. Гергель, К.А Баркалов, // Известия высших учебных заведений. Приборостроение, 2009. Т. 52. № 10. С. 25 – 33.
6. Захарова, Е. Обзор методов многомерной оптимизации. / Е.М. Захарова, И.К. Минашина, // Информационные процессы, 2014. Т 14. №3. С. 256 – 274.
7. Gaviano, M. Software for generation of classes of test functions with known local and global minima for global optimization/ M. Gaviano, D. Lera, D. E. Kvasov, Y. D. Sergeyev // ACM Transactions on Mathematical Software. – 2003. – Vol. 29. – P. 469-480.
8. Сергеев, Я.Д. Диагональные методы глобальной оптимизации / Я.Д. Сергеев, Д.Е. Квасов – М.: Физматлит, 2008. – 352 с.
9. Gablonsky, J.M. A Locally-Biased Form of the DIRECT Algorithm / J.M. Gablonsky, C.T. Kelley, // Journal of Global Optimization, – 2001, Vol. 21, No. 1, – P. 27–37.
10. Paulavicius, R. Parallel branch and bound for global optimization with combination of Lipschitz bounds / R. Paulavicius, J. Zilinskas and A. Grothey // Optimization Methods & Software, 2011. – Vol. 26, No. 3. P. 487–498.
11. Боресков А.А. Параллельные вычисления на GPU. Архитектура и программная модель CUDA / Боресков А.А., Харламов А.А., Марковский Н.Д., Микушин Д.Н., Мортиков Е.В., Мыльцев А.А., Сахарных Н.А., Фролов В.А. – М.: Издательство Московского университета, 2015. 336 с.
12. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум, Т. Остин; перевод с английского Е. Матвеева. — 6-е изд. — Санкт-Петербург [и др.]: Питер, 2014. 816 с.
13. Комолов Д.А. Системы автоматизированного проектирования фирмы Altera MAX+Plus II и Quartus II. / Комолов Д.А. Мяльк Р.А. Зобенко А.А. Филиппов А.С., Издательство РадиоСофт, 2002. 355 с.
14. Gergel, V.P. A novel supercomputer software system for solving time-consuming global optimization problems / V.P. Gergel, K.A. Barkalov A.V. Sysoyev // Numerical Algebra, Control and Optimization, 2018. Vol. 8, No. 1. P. 47-62.
15. Sysoyev, A., Barkalov, K., Sovrasov, V., Lebedev, I., Gergel, V. AGS NLP solver. URL: https://github.com/sovrasov/ags_nlp_solver
16. Сергеев, Я.Д. Диагональные методы глобальной оптимизации / Я.Д. Сергеев, Д.Е. Квасов – М.: Физматлит, 2008. 352 с.

17. Gaviano, M. Software for generation of classes of test functions with known local and global minima for global optimization/ M. Gaviano, D. Lera, D. E. Kvasov, Y. D. Sergeyev // ACM Transactions on Mathematical Software, 2003. Vol. 29. P. 469-480.

Универсальная Многосеточная Технология для параллельного численного решения начально-краевых задач*

С.И. Мартыненко[1,2,3], П.Д. Токталиев[1], В.А. Бахтин[4], Е.В. Румянцев[1], Г.А. Тарасов[1], Н.Н. Середкин[1], К.А. Боярских[1]

[1]Институт Проблем Химической Физики РАН
[2]Объединённый институт высоких температур РАН
[3]Московский Государственный Технический Университет им. Н.Э. Баумана
[4]Институт Прикладной Математики им. М.В. Келдыша РАН

Представлено обобщение параллельной универсальной многосеточной технологии (УМТ) для численного решения начально-краевых задач. Целью данной работы является разработка параллельной вычислительной технологии для решения широкого класса (не)стационарных задач механики сплошных сред при помощи перспективных комплексов программ, устроенных по принципу «чёрного ящика». Параллельная УМТ построена с использованием грубых сеток только по пространству. Показано, что нет принципиальной разницы между параллельным решением краевых и начально-краевых задач, однако снижение объёма выполняемой вычислительной работы сильно зависит от обусловленности матрицы коэффициентов результирующей СЛАУ.

Ключевые слова: начально-краевые задачи, многосеточные методы, параллелизм, OpenMP.

1. Введение

Начально-краевые задачи для уравнений математической физики и эффективные алгоритмы для их численного решения представляют значительный научный и практический интерес. Для этого существуют следующие причины:
1) отдельные физические процессы (например, турбулентность) являются сугубо трёхмерными и нестационарными явлениями;
2) в настоящее время широкое распространение программы, устроенные по принципу «чёрного ящика». Зачастую заранее неизвестно имеет ли задача стационарное решение. Кроме того, в инженерной практике системы нелинейных дифференциальных уравнений в частных производных используют для описания совокупности физико-химических процессов, протекающих в технологическом оборудовании. В этом случае шаг по времени можно использовать в качестве параметра нижней релаксации, обеспечивающий сходимость итераций по нелинейности. Поэтому в программах, устроенных по принципу «чёрного ящика», предпочтительнее использовать начально-краевые задачи в научно-технических приложениях.

Постепенно удалось сформулировать требования к перспективным вычислительным технологиям для программного обеспечения, устроенного по принципу «чёрного ящика»:
1) универсальность (минимальное количество проблемно-зависимых компонентов);
2) эффективность (близкая к оптимальной алгоритмическая трудоёмкость);
3) параллелизм (быстрее самого быстрого последовательного алгоритма).
Данные требования являются взаимоисключающими и очень трудно разработать вычислительную технологию в должной мере удовлетворяющей всем приведённым выше требова-

*Исследовательские работы проведены при финансовой поддержке государства в лице РНФ по соглашению № 21-72-20023 по теме: «Суперкомпьютерное моделирование высокоскоростных ударов по искусственным космическим объектам и Земле».

ниям [2].

Прекрасный обзор достижений в области параллельного интегрирования по времени изложен в статье [3]. В настоящее время ожидается, что уменьшение времени решения задач будет происходить за счёт увеличения числа ядер, а не за счёт более высоких тактовых частот. Таким образом, по мере совершенствования методов пространственного параллелизма распараллеливание по времени предлагает возможность дальнейшего развития параллельных алгоритмов для численного решения начально-краевых задач.

Особенность распараллеливания во времени состоит в принципе причинности: более поздние события определяются более ранними событиями. Алгоритмы, в которых пытаются использовать распараллеливание по времени, должны учитывать этот принцип причинности. Исследования параллельного интегрирования по времени начались около 50 лет назад с работы Нивергельта [4].

Целью данной работы является разработка параллельной вычислительной технологии для решения широкого класса (не)стационарных задач механики сплошных сред при помощи перспективных комплексов программ, устроенных по принципу «чёрного ящика». Чтобы избежать нарушение принципа причинности, использован только пространственный параллелизм.

2. Замечания об универсальном алгоритме

Достаточно трудно дать точное описание классов прикладных задач, которые можно решать унифицированным образом. Тем не менее, сначала попробуем сформулировать проблему о построении универсального алгоритма для линейных задач, затем постараемся обобщить полученные результаты на более сложные случаи. Пусть область Ω есть d-мерный куб, в котором построена равномерная сетка посредством разбиения сторон куба на n частей. Тогда количество узлов составит $N = n^d$, где $d = 2, 3, \ldots$. Тогда численное решение краевых или начально-краевых задач сводится к решению системы линейных алгебраических уравнений (СЛАУ)

$$Ax = b,$$

на одном или нескольких временных слоях. Воспользуемся методом Зейделя для решения результирующей СЛАУ. Выбор данного итерационного алгоритма обусловлен следующими причинами:

а) метод Зейделя эффективно удаляет высокочастотные компоненты погрешности и часто используется в многосеточных методах в качестве сглаживателя;

б) метод Зейделя со специальным блочным упорядочением неизвестных (сглаживатель Ванки) позволяет эффективно решать седловые задачи, причём без использования проблемнозависимых параметров релаксации.

Для общности положим, что использовано блочное упорядочение неизвестных, т. е. количество неизвестных составит

$$N = n_b N_b,$$

где n_b есть количество блоков неизвестных, а N_b – количество неизвестных, образующих блок. Тогда итерации метода Зейделя сводятся к решению n_b СЛАУ прямым методом Гаусса с выбором главного элемента (сглаживатель Ванки), поэтому вычислительная стоимость ($\mathcal{W}$) одной итерации составит

$$\mathcal{W} = C n_b N_b^3 = C n_b^{-2} N^3$$

арифметических операций (ao). Полагая, что количество итераций метода Зейделя равно

$$n_s = C_n n^k = C_n N^{k/d},$$

где константа k зависит от обусловленности матрицы коэффициентов A результирующей СЛАУ. Тогда алгоритмическая трудоёмкость метода Зейделя составит

$$\mathcal{W} = \hat{C} n_b^{-2} N^{3+k/d} \quad \texttt{ao}, \qquad \hat{C} = C C_n.$$

Существенными упрощением анализа является допущение о равномерности вычислительной сетки, которое сводит зависимость алгоритмической трудоёмкости итерационного алгоритма от константы k, которая зависит от обусловленности матрицы коэффициентов A.

Рассмотрим частные случаи. Сначала положим, что $n_b = 1$, т. е. итерационный метод Зейделя совпадает с прямым методом Гаусса ($k = 0 \Rightarrow C_n = 1$). Тогда алгоритмическая трудоёмкость составит

$$\mathcal{W} = C N^3 \quad \texttt{ao},$$

т. е. трудоёмкость $\mathcal{W}$ не зависит от константы k. Поэтому ожидается, что

$$n_b \to 1 \;\Rightarrow\; k \to 0.$$

Прямой метод Гаусса реализуется без проблемно-зависимых компонентов, однако высокая трудоёмкость позволяет использовать данный прямой метод для решения СЛАУ сравнительно небольшого размера.

Второй важный частный случай $n_b = N$, т. е. итерационный метод Зейделя с точечным упорядочением неизвестных. В этом случае трудоёмкость метода Зейделя составит

$$\mathcal{W} = \hat{C} n_b N_b^3 = \hat{C} N^{1+k/d} \quad \texttt{ao}.$$

Очевидно, что итерационный алгоритм обладает, как минимум, двумя проблемно-зависимыми компонентами: критерием останова итераций и упорядочиванием неизвестных, которое влияет на константу k. Здесь предполагается, что упорядочение уравнений подчинено упорядочению неизвестных. Далее, если $k \to 0$ то метод Зейделя обладает оптимальной трудоёмкостью:

$$\mathcal{W} \to \hat{C} N \quad \texttt{ao}.$$

Подобная ситуация возникает, если матрица A имеет сильное диагональное преобладание. Как правило, попытки ускорить сходимость итерационных методов в данном случае не приводят к ощутимым результатам.

В приложениях чаще встречается случай, когда матрица коэффициентов A плохо обусловлена. Полагая, что итерационный метод Зейделя потребует меньшего объёма вычислительной работы, чем прямой метод Гаусса при больших N получим, что $k < 2d$

Таким образом, задачу о построении универсального алгоритма для решения линейных (начально-)краевых задач на равномерной сетке сформулируем следующим образом:
1) Если матрица коэффициентов A хорошо обусловлена ($k \to 0$), то универсальный алгоритм должен совпадать с методом Зейделя.
2) Если матрица коэффициентов A плохо обусловлена ($0 < k < 2d$), то необходимо добавить минимальное количество проблемно-зависимых компонентов к методу Зейделя, чтобы:
а) снизить трудоёмкость до близкой к оптимальной, т. е.

$$\mathcal{W} = \bar{C} n_b^{-2} N^3 \log N \quad \texttt{ao}.$$

б) достичь крупнозернистого параллелизма, причём параллельный алгоритм, обладающий близкой к оптимальной трудоёмкостью, должен исполняться быстрее самого быстрого (оптимального) последовательного алгоритма.

В действительности, в научно-технических приложениях (не)стационарные физико-химическими процессы описываются (начально-)краевыми задачами для систем нелинейных дифференциальных уравнений в частных производных. Как правило, для аппроксимации дифференциальных задач используют сильно неравномерные сетки, глобально или

локально линеаризованные дискретные аналоги дифференциальных уравнений решают сегрегированно или совместно. Поэтому получить зависимость трудоёмкости от количества неизвестных практически невозможно. Однако и в общем случае проблема построения универсального алгоритма остаётся схожей: если метод Зейделя сходится медленно, то следует ускорить сходимость при помощи минимального количества дополнительных проблемно-зависимых компонентов.

Следует ещё раз подчеркнуть, что формализация вычислений не связана с разработкой нового метода решения новых задач.

3. Параллельный многосеточный цикл

Последовательная Универсальная Многосеточная Технология (УМТ) подробно представлена в [1, 7].

Рассмотрим некоторую нелинейную краевую задачу в области Ω. Предположим, что в данной области Ω построена глобально-структурированная сетка G_1^0, т. е. сетка G_1^0 порождает фундаментальную многосеточную структуру [5]. В 3D случае первый сеточный уровень состоит из 27 сеток, поэтому количество независимых вычислителей (нитей при использовании технологии OpenMP) должно быть 3, 9 или 27. Предположим, что сетки первого уровня являются самыми мелкими в первой многосеточной итерации (т. е. самой мелкой сетки нет, а сетки первого уровня являются динамическими [5]). Несколько многосеточных итераций на многосеточных структурах, порождаемых динамическими сетками первого уровня, позволяют получить достаточно точное приближение к решению на самой мелкой сетке G_1^0. Нетрудно показать, что для схем второго порядка аппроксимации различие между приближением к решению и решением будет не более одной верной значащей цифры

$$\|u^h - u^{3h}\| \le 10\|u_a - u^h\|.$$

Здесь u^h и u^{3h} есть численные решения, полученные на сетках с шагом h и $3h$, а u_a – аналитическое решение. Вторая и последующие многосеточные итерации необходимы для получения этой значащей цифры.

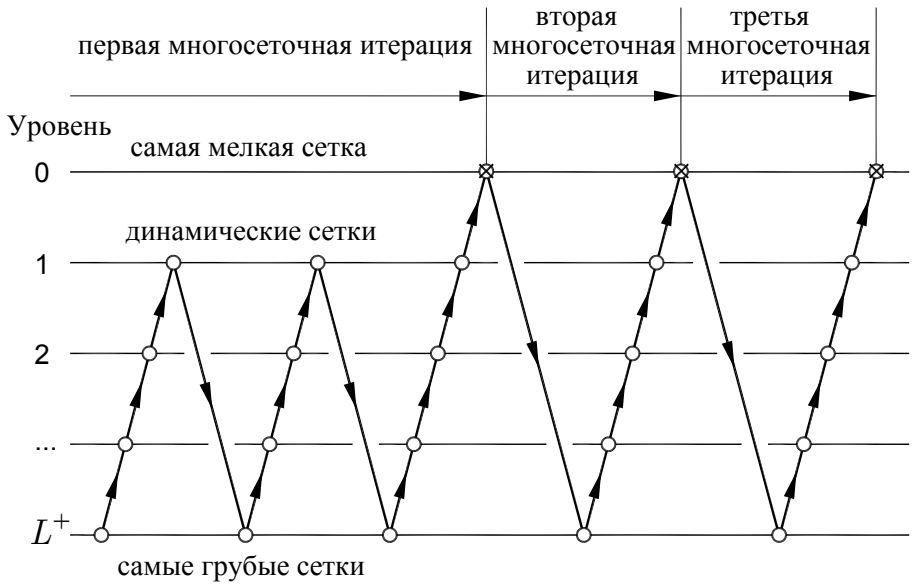


Рис. 1. Параллельный многосеточный цикл для численного решения нелинейных начально-краевых задач.

Первая многосеточная итерация состоит из решения 27 независимых подзадач, приблизительно одного размера, которые можно решать независимо друг от друга и без обмена данными. Именно первая многосеточная итерация обуславливает крупную зернистость универсальной многосеточной технологии (УМТ) [4].

Этот многосеточный цикл можно использовать и для решения нелинейных начально-краевых задач, если не осуществляется распараллеливание по времени. Данное обстоятельство очень удобно для унификации алгоритмов для численного решения нелинейных краевых и начально-краевых задач на (не)структурированных сетках.

Напомним, что операторы переходов УМТ не зависят от решаемой задачи: оператор сужения является следствием свойства аддитивности определённого интеграла относительно подобластей, а оператор пролонгации фактически определяется количеством узлов сетки и не связан с интерполяцией.

В настоящее время активно развивают два семейства алгоритмов для решения (начально-)краевых задач на неструктурированных сетках. Алгебраические многосеточные методы основаны на построении дискретных задач различной размерности без использования геометрической информации о вычислительной сетке. Фактически, решение начинают с СЛАУ, исходными данными является матричная формулировка. Построение СЛАУ меньшего размера есть проекция дискретной задачи из пространства большего размера в пространство меньшего размера. В действительности, большинство математических моделей являются системами нелинейных дифференциальных уравнений в частных производных, причём сильно связными между собой. Поэтому очень трудно построить вариант алгебраических многосеточных методов для совместного решения локально линеаризованных дискретных аналогов исходных нелинейных дифференциальных уравнений.

В качестве альтернативы можно рассмотреть метод вспомогательного пространства, в котором помимо исходной неструктурированной сетки так же используют вспомогательную (структурированную) сетку для отыскания поправки (разницы между искомым решением и текущим приближением к нему). В этом случае УМТ можно использовать для отыскания поправки на вспомогательной сетке. Этот итерационный метод в линейном случае имеет вид

$$b_0 - A_0\varphi_0^{(q+1)} = M(b_0 - A_0\varphi_0^{(q)})$$

где матрица итераций данного двухсеточного метода выглядит следующим образом

$$M = A_0 S_0^{\nu}(A_0^{-1} - \mathcal{P}_{A\to 0}A_A^{-1}\mathcal{R}_{0\to A}).$$

Здесь $\mathcal{P}_{A\to 0}$ и $\mathcal{R}_{0\to A}$ есть проблемно-независимые операторы переходов УМТ, а S_0 – матрица сглаживающих итераций. Нижние индексы O и A обозначают принадлежность к исходной и вспомогательной сеткам. Основной объём вычислений связан с численным решением вспомогательной СЛАУ

$$A_A c_A = \mathcal{R}_{0\to A}(b_0 - A_0\varphi_0)$$

для отыскания поправки c_A, т. е. с численным обращением матрицы A_A.

Фактически, УМТ используют для построения предобуславливателя

$$P^{-1}A_0\varphi_0 = P^{-1}b_0,$$

где

$$P^{-1} = A_0^{-1}(I - M).$$

Двухсеточный метод легко обобщить для решения нелинейных задач (используя схему FAS), при этом с исходной сетки на вспомогательную необходимо интерполировать невязку и приближение к решению (для нелинейных задач), а обратно – поправку. В отличие от алгебраических многосеточных методов здесь нет необходимости интерполировать дискретные (начально-)краевые задачи с сетки на сетку.

4. Вычислительные сетки нулевого и первого уровней

Для наглядности рассмотрим 1D вычислительную сетку для аппроксимации краевых задач методом конечного объёма. Данная сетка состоит из двух множеств точек, которые являются либо узлами, либо гранями конечного объёма. (Рис. 2).

Пусть неизвестная сеточная функция есть u_i, $i = 1, 2, \ldots, N+1$. Будем полагать, что поставлены граничные условия Дирихле, т. е. в граничных узлах значения сеточной функции (искомого решения) u_1 и u_{N+1} известны точно. Пусть количество независимых вычислителей составит $P = 3, 9, 27$.

Объединим неизвестные u_i в блоки из n_b неизвестных. Если в граничных узлах заданы условия Дирихле, то для реализации метода Зейделя с двухцветным упорядочением неизвестных зададим количество узлов самой мелкой сетки в виде

$$N + 1 = cPn_b + 2,$$

где $c = 2$ есть количество цветов в упорядочении неизвестных. Будем называть неизвестные u_i, где

$$2 + 2(m-1)n_b \le i \le 1 + (2m-1)n_b, \quad m = 1, 2, \ldots, P,$$

блоками неизвестных первого цвета, а неизвестные u_i, где

$$2 + (2m-1)n_b \le i \le 1 + 2mn_b, \quad m = 1, 2, \ldots, P,$$

блоками неизвестных второго цвета. Метод Зейделя с данным двухцветным упорядочением неизвестных будет использован в качестве сглаживателя на самой мелкой сетке [6].

Распараллеливание сглаживающих итераций на грубых сетках первого уровня является тривиальным (Рис. 2).

5. Геометрический и алгебраические параллелизмы УМТ

Декомпозиция самой мелкой сетки на независимые подсетки определяет геометрический параллелизм УМТ. На динамических сетках первого уровня начально-краевую задачу решают традиционным образом на нескольких временных слоях. На рис. 3 показано независимое решение начально-краевой задачи на четырёх сеточных уровнях (значения $u_i^{(n+1)}$ являются известными). Полученные $u_i^{(n+k)}$, $k = 2, 3, 4, 5$ являются приближениями к решению на самой мелкой сетке. Их необходимо хранить в оперативной памяти, что и определяет количество временных слоёв на которых одновременно отыскивают решение.

Заметим, что количество сеточных уровней определяется числом обусловленности матрицы коэффициентов результирующей СЛАУ, получаемой в результате аппроксимации

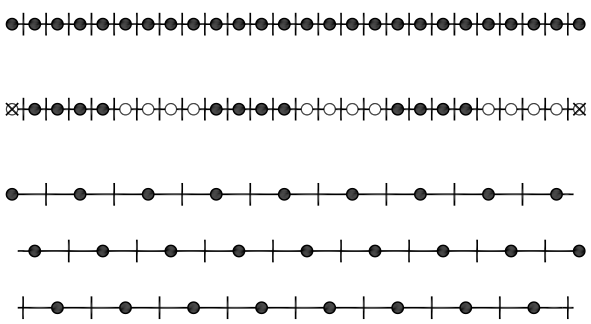


Рис. 2. 1D вычислительные сетки: самая мелкая, самая мелкая с двухцветным упорядочением неизвестных, грубые сетки первого уровня.

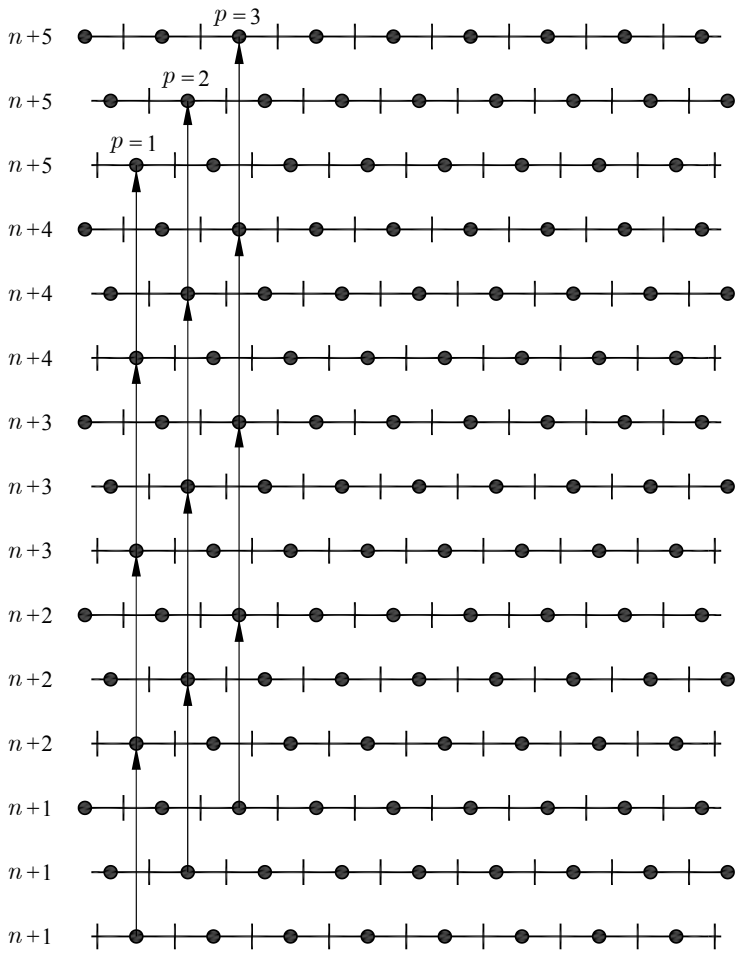


Рис. 3. Геометрический параллелизм УМТ: сглаживание на грубых сетках.

начально-краевой задачи на самой мелкой сетке. Чем лучше обусловлена дискретная начально-краевая задача, тем меньшее количество сеточных уровней будет в многосеточной структуре.

Алгебраический параллелизм определяется тем, что количество независимых вычислителей больше количества вычислительных сеток. Поэтому необходимо искусственно разделять исходную задачу на подзадачи, которые можно обрабатывать независимо. Это возможно при помощи многоцветных упорядочений неизвестных (рис. 2).

Параллельное сглаживание на самой мелкой сетке показано на рис. 4. Сначала осуществляется обработка неизвестных, образующих блоки одного цвета, затем – другого цвета. Сглаживание на самой мелкой сетке завершает многосеточную итерацию. После чего вычисляют новое значение приближение к решению, обнуляют поправку и проверяют критерий останова многосеточных итераций. В случае необходимости выполняют последующую многосеточную итерацию.

6. Иллюстративный пример

Пусть Ω является единичным кубом и на границе S поддерживается заданная температура u_s, т. е.

$$u(t, P) = u_s(t, P), \quad P \in S,\ t \geq 0.$$

Кроме того, известна температура тела u_0 в некоторый начальный момент $t = 0$

$$u(0, M) = u_0(0, M), \quad M \in \bar{V}.$$

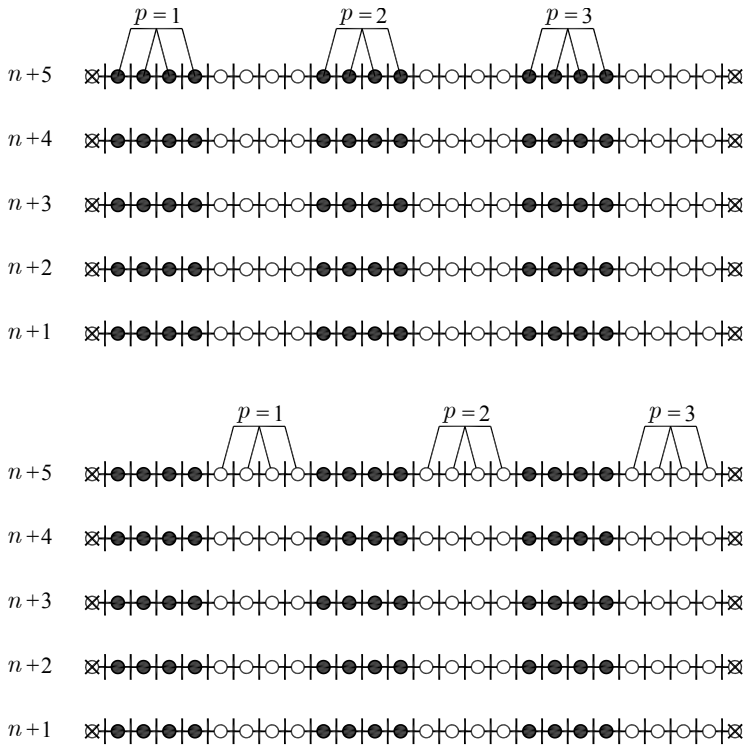


Рис. 4. Алгебраический параллелизм: сглаживание на самой мелкой сетке.

Классическим решением начально-краевой задачи для уравнения теплопроводности

$$u'_t = a^2 \Delta u + f$$

называется функция $u(t, M)$, непрерывная вместе с первыми производными по координатам в замкнутом цилиндре $V \times [0, T]$, имеющая непрерывные производные первого порядка по t и второго порядка по M в открытом цилиндре $V \times (0, T]$, удовлетворяющая в $V \times (0, T]$ уравнению теплопроводности, граничному и начальному условиям.

Выберем следующую функцию

$$u(t, x, y, z) = \sin(2\pi t) + Q(x)Q(y)Q(z)$$

в качестве точного решения модельной начально-краевой задачи для уравнения теплопроводности. Здесь

$$Q(\theta) = \exp(\alpha\theta) + (1 - \exp(\alpha))\theta - 1, \quad \theta = (x\ y\ z)^T,$$

а $\alpha \geq 0$ есть некий параметр.

Для аппроксимации начально-краевой задачи для уравнения теплопроводности воспользуемся схемой Кранка-Николсона

$$\frac{u_{ijk}^{(n+1)} - u_{ijk}^{(n)}}{h_t} = \frac{a^2}{2}\Big(\Delta^h u_{ijk}^{(n)} + \Delta^h u_{ijk}^{(n+1)}\Big) + \frac{1}{2}\Big(f(t^{(n)}, x_i, y_j, z_k) + f(t^{(n+1)}, x_i, y_j, z_k)\Big).$$

Эффективность параллельной УМТ определим традиционным образом: ускорением (S) и эффективностью (E) параллельного алгоритма называется величина

$$S = PE = \frac{T(1)}{T(P)},$$

где $T(1)$ – время выполнения последовательного алгоритма, а $T(P)$ – время выполнения параллельного алгоритма на системе из P процессоров [6].

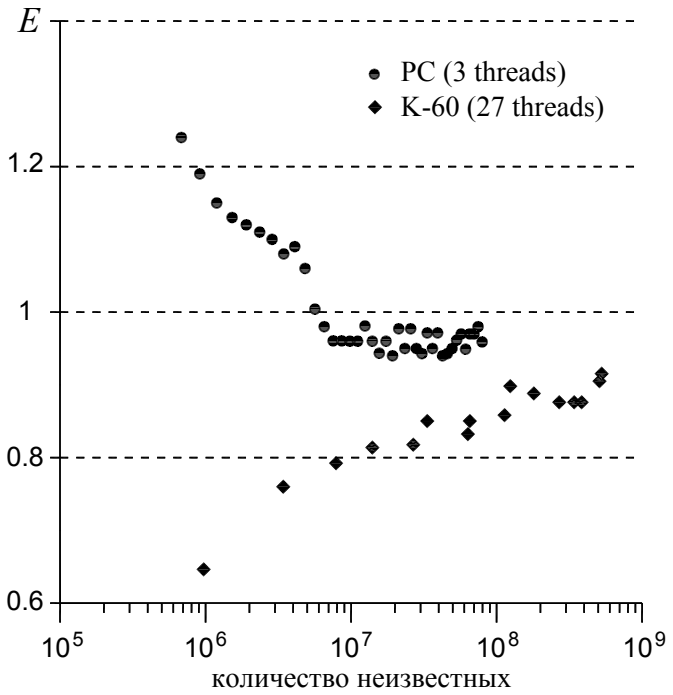


Рис. 5. Зависимость эффективности параллельной УМТ от количества неизвестных.

Персональный компьютер (Intel(R) Core(TM) i7-4790 CPU@3.60 GHz) и кластер K-60 (Институт Прикладной Математики имени М.В. Келдыша РАН) использованы для вычислительных экспериментов [9]. Параллельный код разработан с привлечением технологии распараллеливания OpenMP. Эффективность E определялась после получения численного решения на семи временных слоях при $h_t = h_x = h_y = h_z$. УМТ реализована со сглаживателем Ванки с блоками неизвестных $3 \times 3 \times 3$. Результаты вычислительного эксперимента показаны на рис. 5. Сверхлинейное ускорение $E > 1$ при использовании трёх нитей определяется кэш-памятью, которая позволяет хранить бóльшую часть данных? если количество неизвестных достаточно мало.

Эффективность последовательных многосеточных методов применительно к численному решению начально-краевых задач зависит от обусловленности матрицы коэффициентов результирующей СЛАУ: для хорошо обусловленных матриц многосеточные методы не позволяют существенно уменьшить объём требуемой вычислительной работы, однако для плохо обусловленных матриц многосеточные методы будут наиболее быстрыми алгоритмами. С точки зрения программной реализации, параллельное решение краевых и начально-краевых задач различается лишь количеством временных слоёв, но которых параллельно отыскивается решение: один временной слой для краевых задач и несколько временных слоёв для начально-краевых задач. Эффективность параллельной УМТ в обоих случаях приблизительно одинакова.

Неэффективный доступ к памяти — одна из наиболее распространённых проблем с производительностью в параллельных программах. Скорость загрузки данных из памяти традиционно отстаёт от скорости их обработки процессором. Тенденция размещать все больше и больше ядер на чипе означает, что каждое ядро имеет относительно более узкий канал для доступа к ресурсам общей памяти. Анализ программы, выполненный с помощью Intel VTune Performance Analyzer, показывает, что при реализации 27 потоков с использованием 2 процессоров Intel Xeon Gold 6142 v4 16-20% обращений к памяти приходится на обращения к удалённой памяти. Это приводит к снижению эффективности параллельной программы.

Литература

1. Мартыненко С.И. Многосеточная технология: теория и приложения / Под. ред. М. П. Галанина. – М.: Физматлит, 2015. 208 с.

2. Мартыненко С.И., Бахтин В.А., Румянцев Е.В. и др. Параллельное решение краевых задач с помощью технологии OpenMP. Вестник МГТУ им. Н.Э. Баумана. Сер. Естественные науки, 2022, No. 2 (101), С. 36–56. DOI: 10.18698/1812-3368-2022-2-36-56

3. Gander M.J. 50 years of Time Parallel Time Integration. In: Multiple Shooting and Time Domain Decomposition. Springer. 2015. P. 69–113. DOI: 10.1007/978-3-319-23321-5_3

4. Nievergelt J. Parallel methods for integrating ordinary differential equations. Commun. ACM 7(12). 1964. P. 731–733. DOI: 10.1145/355588.365137

5. Мартыненко С.И. Параллельное программное обеспечение для универсальной многосеточной технологии. М.: Триумф, 2020. 150 с. URL: https://github.com/simartynenko/Robust_Multigrid_Technique_2021_OpenMP

6. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем. М: Мир, 1991. 367 с.

7. Martynenko S.I. The robust multigrid technique: For black-box software. Berlin: De Gruyter, 2017.

8. Martynenko S., Zhou W., Gökalp İ., V. Bakhtin V., Toktaliev P. Parallelization of Robust Multigrid Technique Using OpenMP Technology. In: Malyshkin V. (eds) Parallel Computing Technologies. PaCT 2021. Lecture Notes in Computer Science, vol 12942. Springer, Cham, 2021.

9. KIAM Homepage, www.kiam.ru/MVS/resourses/k60.html

Учебный курс «Программирование с использованием модели oneAPI»*

А.В. Сысоев, И.Б. Мееров, А.В. Горшков, В.Д. Волокитин

Нижегородский государственный университет им. Н.И. Лобачевского

Современные высокопроизводительные вычислительные системы в массе своей являются гетерогенными. Разработка параллельных программ, способных использовать весь потенциал таких систем, сопряжена со значительными сложностями – требуется не только использовать соответствующие языки и технологии программирования, но и учитывать особенности центральных и графических процессоров, влияющие в том числе на схемы организации параллелизма и работу с памятью. На упрощение процесса разработки таких программ направлена модель гетерогенного программирования oneAPI, представленная компанией Intel, и ее ключевой компонент – язык Data Parallel C++ [1], позволяющий разрабатывать переносимые высокопроизводительные программы для CPU, GPU, FPGA и других устройств [1-3]. В статье представлен учебный курс по oneAPI [4], разработанный в ННГУ им. Н. И. Лобачевского [5]. Курс направлен на изучение широкого спектра вопросов, связанных с высокопроизводительными вычислениями с использованием моделей, методов и инструментов параллельного программирования на платформах Intel. В статье представлена концепция курса, описана его структура, категории слушателей, которым он может быть интересен, и варианты построения курса в зависимости от уровня подготовки аудитории.

Ключевые слова: образование, высокопроизводительные вычисления, параллельное программирование, гетерогенные вычислительные системы, Data Parallel C++, SYCL

1. Введение

Одна из тенденций последнего десятилетия заключается в появлении широкого спектра вычислительных архитектур, уникальные особенности которых требуют разработки, освоения и применения соответствующих языков и средств программирования. Известно, что затраты на подготовку специалистов, способных квалифицированно использовать языки и средства программирования для создания высокопроизводительных программ, существенно отличаются. Так, например, языки C, C++ и Fortran, широко применяемые для разработки программных средств численного моделирования, изучаются при подготовке бакалавров по всему миру, тогда как CUDA, позволяющая программировать для GPU, требует существенно больших трудозатрат (и предварительных знаний) при ее освоении. Принимая во внимание тот факт, что основные средства программирования для CPU и GPU не совместимы, а особенности разных вычислительных архитектур требуют использования специфичных приемов оптимизации кода, нередко требуется разработка, поддержка и развитие нескольких версий программного кода. На это обычно не хватает ресурсов.

Для решения этой проблемы разрабатываются универсальные средства разработки программ (языки, расширения для языков, библиотеки), которые позволяют использовать один язык параллельного программирования для разных вычислительных устройств. Среди таких средств можно отметить OpenCL, OpenACC, SYCL, alpaka, KOKKOS и другие разработки. В данной работе мы основываемся на недавно представленном языке гетерогенного программирования Data Parallel C++ (DPC++), который основан на стандарте языка SYCL и является частью модели oneAPI, вобравшей в себя практически весь набор библиотек и инструментов программирования, разработанный в компании Intel за последние десятилетия. Наличие обширно-

* Разработан в Центре компетенций oneAPI ННГУ при поддержке компании Intel. Работа частично поддержана Министерством науки и высшего образования, проект № 0729-2020-0055.

го и хорошо проработанного инструментария (компиляторы, профилировщики, отладчик, анализатор кода, математические библиотеки, средства организации и поддержки параллелизма, проблемно-ориентированные библиотеки) и сообщества разработчиков и пользователей является достоинством oneAPI и DPC++, и в целом дает надежду на успешное развитие этого проекта. Несмотря на то, что вопрос переносимости производительности (performance portability) далек от своего решения, сам факт использования единого инструментария уже является существенным шагом вперед. Этими соображениями и обусловлен наш интерес к разработке учебного курса по модели oneAPI и языку DPC++.

Учебный курс по любой технологии параллельного программирования опирается на два предположения относительно слушателя, приступающего к его освоению.

1. Слушатель владеет базовыми знаниями по основам алгоритмизации, модульному программированию, алгоритмам и структурам данных, а также имеет опыт применения этих знаний, как минимум, на учебных задачах.
2. Слушатель владеет языком программирования, на котором построена рассматриваемая технология, как минимум, в той степени, при которой в представленных в курсе примерах программ ему не встречаются незнакомые конструкции языка.

Другими словами, в рамках курса по технологии параллельного программирования слушателя не нужно учить разработке последовательных программ на требуемом этой технологией языке.

Однако в силу новизны языка DPC++ курс по модели oneAPI может быть интересен разным категориям слушателей с существенно разным уровнем подготовки на входе. Таким образом, в основу построения структуры курса была положена идея модульности и возможности адаптации его наполнения и длительности, исходя из целей обучения и особенностей аудитории.

Дальнейшее изложение в статье построено следующим образом. В разделе 2 изложены требования к уровню знаний слушателя курса, как обязательные, так и желательные. В разделе 3 описаны категории слушателей, которым курс по oneAPI может быть полезен/интересен. В разделе 4 описана структура курса, в разделе 5 обобщен первый опыт его прочтения.

2. Предварительные требования к уровню знаний слушателей курса

Модель oneAPI представляет язык разработки гетерогенных параллельных программ DPC++. Язык DPC++ основан на стандарте языка программирования SYCL, который, в свою очередь, в значительный степени является расширением языка программирования C++ [6-8] (стандарта 11 и выше). Таким образом, курс по oneAPI добавляет к базовым требованиям к слушателю – знаниям по основам алгоритмизации и структурам данных – еще и владение языком программирования C++, а также объектно-ориентированным программированием (ООП). Как ООП, так и базовый C++ не могут быть с хоть сколько-нибудь необходимой детальностью изложены в рамках курса по oneAPI и являются обязательными входными требованиями.

При этом, несмотря на то, что существенное обновление языка, привнесенное стандартом C++ 11 и, пусть и в меньшей степени, продолженное последующими стандартами (14, 17 и 20), произошло уже более десяти лет назад, многие учебные курсы, использующие язык C++ в качестве базового при изложении концепций и технологий программирования, включая ООП, все еще ориентированы на изложение C++ 98/03. Как следствие, те конструкции DPC++, которые основаны на возможностях современного C++, могут быть слушателям, с ними незнакомыми, не вполне понятны. Выходов из ситуации видится два: 1) включить во входные требования к курсу по oneAPI знания по C++ стандартов 11 и выше; 2) изложить используемые в DPC++ конструкции современного C++ непосредственно в курсе по oneAPI.

Проанализировав совместно с коллегами из компании Intel язык DPC++ с точки зрения набора используемых им элементов C++ из стандартов, начиная с 11-го, мы пришли к выводу, что этот набор ограничен в достаточной степени, чтобы в рамках курса по oneAPI можно было реализовать именно второй вариант, не повышая тем самым входные требования к потенциальным слушателям.

Аналогичным образом обстоит дело еще с двумя областями знаний: архитектуры вычислительных систем и операционные системы. Качественное освоение любого курса по параллель-

ному программированию и высокопроизводительным вычислениям невозможно без определенного уровня знаний об архитектурах параллельных вычислительных систем [9, 10] и ключевых возможностях современных операционных систем [11, 12], на которых базируется разработка параллельных программ. Предварительное изучение соответствующих курсов безусловно будет полезно перед тем, как приступать к знакомству с миром разработки параллельных программ, но, тем не менее, не является строго обязательным. Необходимый минимум сведений из архитектур и операционных систем может быть изложен непосредственно в курсе по параллельному программированию в виде дополнительных начальных модулей, которые для подготовленных слушателей могут быть как сокращены, так и опущены вовсе.

3. Категории слушателей

Как уже было отмечено во введении, курс по oneAPI может быть интересен разным категориям слушателей. В первую очередь, это студенты бакалавриата, уже прошедшие через курсы по основам программирования, алгоритмам и структурам данных, и освоившие язык C++ и технологию ООП. Курс может быть полезен магистрантам, возможно уже знакомым с какими-то из технологий разработки параллельных программ (OpenMP, TBB, MPI и др.). Курс может заинтересовать и преподавателей вузов, читающих учебные курсы по технологиям параллельного программирования и желающих их актуализировать. Наконец, в силу новизны языка DPC++ курс может привлечь и специалистов в области высокопроизводительных вычислений и суперкомпьютерных технологий, желающих попробовать и/или освоить новую технологию. Очевидно, что каждой из указанных категорий слушателей требуется разная степень детальности изложения материала, каждая из них на входе в курс обладает разными входными знаниями и опытом. Именно по этой причине структура курса по oneAPI построена в виде отдельных модулей, часть из которых может быть опущена и/или сокращена при соответствующем уровне подготовки слушателей.

4. Структура курса

Курс состоит из 6 модулей:

1. Введение в мир суперкомпьютерных вычислений.
2. Архитектурные механизмы, влияющие на производительность. Уровни параллелизма.
3. Операционные системы. Аспекты параллелизма.
4. Программирование на языке Data Parallel C++.
5. Программные библиотеки и инструменты oneAPI.
6. Высокопроизводительные вычисления и научное моделирование. Примеры использования.

Модули 2 и 3 относятся к необязательным и, как уже было сказано выше, могут быть сокращены или опущены.

Модуль по DPC++ является основным в курсе. Именно к нему, в первую очередь, относятся сформулированные в разделе 2 входные требования к слушателям.

Модуль 5 содержит обзорные материалы по библиотекам, включенным компанией Intel в oneAPI, а также инструментам, поддерживающим процесс разработки параллельных программ.

В модуле 6 демонстрируется использование библиотек и инструментов oneAPI при решении задач в научных проектах.

Рассмотрим содержание модулей более подробно.

4.1 Модуль «Введение в мир суперкомпьютерных вычислений»

Продолжительность модуля – одна лекция.

Предварительные требования к освоению модуля отсутствуют.

В модуле дается обзор предметной области суперкомпьютерных вычислений, показывается разнообразие типов вычислительных систем и подходов к разработке параллельных программ. Обсуждаются сильные и слабые стороны подходов. Демонстрируется, чем DPC++ может помочь разработчикам параллельных программ для гетерогенных вычислительных систем.

4.2 Модуль «Архитектурные механизмы, влияющие на производительность. Уровни параллелизма»

Продолжительность модуля – две лекции.

Предварительные требования к освоению модуля отсутствуют.

В модуле рассматриваются ключевые особенности современных вычислительных архитектур, с точки зрения высокопроизводительных вычислений и суперкомпьютерных технологий. Обсуждается их влияние на производительность и эффективность параллельных программ.

4.3 Модуль «Операционные системы. Аспекты параллелизма»

Продолжительность модуля – две лекции, одна практика.

Предварительные требования к освоению модуля отсутствуют.

В модуле дается обзор ключевых особенностей современных операционных систем с точки зрения разработки параллельных программ. Рассматриваются процессы, потоки и планирование времени центрального процессора. Обсуждаются вопросы синхронизации выполнения потоков.

4.4 Модуль «Программирование на языке Data Parallel C++»

Модуль состоит из пяти лекций и пяти практик.

Предварительные требования к освоению модуля: знание технологий структурного, модульного и объектно-ориентированного программирования. Знание C++ 98/03 или C++ 11. Представление об архитектуре многоядерных центральных процессоров. Представление об архитектуре графических процессоров. Представление о потоках и их синхронизации.

Поскольку данный модуль является в курсе основным, его структуру и наполнение рассмотрим более детально.

Лекция 0. Элементы современного C++

Рассматриваются конструкции языка C++, появившиеся в нем в стандарте C++ 11 и последующих, активно используемые при разработке параллельных программ на языке DPC++.

Практика 0. Элементы современного C++ 11, используемые в DPC++.

Лекция 1. Введение в Data Parallel C++

Основные темы лекции:

- обзор Intel oneAPI – концепция, языки, библиотеки, инструменты;
- базовые понятия гетерогенного программирования – хост, устройство, ядро, очередь, типы памяти;
- язык DPC++, основные особенности – язык высокого уровня, единый исходный код для всех поддерживаемых устройств, стандартный C++;
- модель SPMD (single program, multiple data);
- обработка ошибок;
- вывод информации с устройства.

Практика 1. Hello World

Задачи практики:

- вывести в консоль все доступные платформы и устройства средствами DPC++;
- разработать ядро (kernel), чтобы напечатать “Hello” из ядра каждого из устройств.

Лекция 2. Исполнение ядер в Data Parallel C++

Основные темы лекции:

- модель исполнения ядра DPC++;
- запуск ядра – анатомия ядра, типы параллелизма, индексация потоков/групп;
- измерение времени выполнения ядра;
- групповые алгоритмы.

Практика 2. Вычисление двойного интеграла методом сумм Римана
Задачи практики:

- реализовать на DPC++ вычисление заданного интеграла с использованием средней суммы Римана;
- сравнить производительность ядра для разных платформ и числа разбиений.

Лекция 3. Управление памятью в Data Parallel C++
Основные темы лекции:

- модель памяти;
- буферы и объекты доступа (аксессоры);
- типы аксессоров;
- атомарные операции;
- унифицированная разделяемая память;
- пример: матрично-векторное умножение.

Практика 3. Решение систем линейных уравнений методом Якоби
Задачи практики:

- реализовать на DPC++ решение системы уравнений методом Якоби;
- разработать три версии – с использованием аксессоров, разделяемой памяти (модель USM), памяти устройства;
- сравнить производительность для разных версий, устройств и входных данных.

Лекция 4. Оптимизация программ на Data Parallel C++
Основные темы лекции:

- базовые рекомендации;
- глобальная память;
- локальная память;
- подгруппы;
- приватная память;
- векторизация;
- высокопроизводительные библиотеки.

Практика 4. Умножение матриц
Задачи практики:

- реализовать на DPC++ умножение матриц;
- разработать четыре версии – «наивную», блочную, векторизованную блочную, с использованием oneMKL;
- сравнить производительность для разных версий, устройств и входных данных.

4.5 Модуль «Программные библиотеки и инструменты oneAPI»

Продолжительность модуля – две лекции, одна практика, три мастер-класса.

Предварительные требования к освоению модуля отсутствуют.

В модуле дается краткий обзор библиотек IPP, oneVPL, oneDAL, oneMKL, oneTBB, oneDPL с точки зрения их использования при разработке параллельных программ. Более детально рассматривается работа с библиотекой oneVPL. В виде мастер-классов демонстрируется оптимизация программ с использованием Intel VTune и Intel Advisor, а также вопросы высокопроизводительного вывода нейронных сетей с Intel Distribution of OpenVINO toolkit.

4.6 Модуль «Высокопроизводительные вычисления и научное моделирование. Примеры использования»

Продолжительность модуля – четыре мастер-класса.

Предварительные требования к освоению модуля: знание технологий структурного, модульного и объектно-ориентированного программирования; знание C++ 11; знание языка DPC++.

В модуле предполагается продемонстрировать использование библиотек и инструментов oneAPI при решении задач в научных проектах. Запланированы следующие мастер-классы:

- портирование модуля движения частиц [13-15] на DPC++ (анализ и оптимизация производительности на CPU и GPU с использованием Intel VTune);
- анализ финансовых рынков: вычисление формулы Блэка-Шоулса (анализ и оптимизация производительности на CPU и GPU с использованием Intel VTune);
- интегрирование уравнений Максвелла методом FDTD (анализ и оптимизация производительности на CPU и GPU);
- вычисления в смешанной точности на CPU и GPU на примере задачи интегрирования уравнений Максвелла методом FDTD.

В настоящий момент готовы первый и второй мастер-класс, остальные находятся в разработке.

5. Апробация

При разработке курса мы ориентировались на его всестороннюю апробацию, сбор отзывов от разных категорий слушателей и соответствующую адаптацию материалов. В начале мы провели серию открытых лекций, преимущественно ориентированных на студентов 3-6 курсов, которые ранее уже занимались проблематикой высокопроизводительных вычислений. На основе полученной обратной связи, а также в ходе консультаций с разработчиками oneAPI была сформирована программа курса, описанная в предыдущем разделе статьи. После разработки основной части материалов курс был прочитан студентам магистерской программы «Вычислительные методы и суперкомпьютерные технологии» [16]. С использованием материалов курса был проведен мастер-класс на конференции «Суперкомпьютерные дни в России» (Москва, сентябрь 2021 года). Материалы курса прошли рецензирование у разработчиков oneAPI. По итогам была разработана и опубликована текущая версия материалов курса на русском [17] и английском [18] языках.

В январе 2022 года была организована в онлайн-формате программа повышения квалификации по программированию с использованием модели oneAPI для преподавателей вузов и сотрудников НИИ России. Получены положительные отзывы по результатам прохождения обучения, материалы переданы коллегам для дальнейшего использования. В настоящее время курс дополняется примерами из реальных научных проектов Центра суперкомпьютерных технологий ННГУ.

Литература

1. Reinders J., Ashbaugh B., Brodman J., Kinsner M., Pennycook J., Tian X. Data Parallel C++ Mastering: DPC++ for Programming of Heterogeneous Systems using C++ and SYCL. URL: https://link.springer.com/book/10.1007/978-1-4842-5574-2.
2. oneAPI GPU Optimization Guide. URL: https://software.intel.com/content/www/us/en/develop/documentation/oneapi-gpu-optimization-guide/top.html.
3. oneAPI FPGA Optimization Guide. URL: https://software.intel.com/content/www/us/en/develop/documentation/oneapi-fpga-optimization-guide/top.html.
4. Программирование с использованием модели oneAPI. URL: https://hpc-education.unn.ru/ru/центр-компетенций-oneapi-в-ннгу/курс-oneapi.
5. Центр компетенций oneAPI. URL: https://hpc-education.unn.ru/ru/центр-компетенций-oneapi-в-ннгу.
6. Bjarne Stroustrup. The C++ Programming Language (4th Edition), Addison-Wesley, 2013.

7. Nicolai M. Josuttis. The C++ Standard Library – A Tutorial and Reference, second edition, Addison-Wesley, 2012.
8. C++ FAQ. URL: https://isocpp.org/wiki/faq.
9. Паттерсон Д., Хеннесси Д. Архитектура компьютера и проектирование компьютерных систем. – 2012.
10. Таненбаум Э. Архитектура компьютера. СПб.: Питер, 2015.
11. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. – СПб.: Питер, 2015.
12. Коньков К.А., Карпов В.Е. Основы операционных систем: курс лекций: учеб. пособие для вузов – М.: Национальный Открытый Университет “ИНТУИТ”, 2016.
13. Код Hi-Chi. URL: https://github.com/hi-chi/pyHiChi.
14. О проекте. URL: http://hpc-education.unn.ru/en/research/overview/laser-plasma.
15. Доклад на oneAPI DevSummit at ISC. URL: https://www.oneapi.io/event-sessions/porting-boris-particle-pusher-dpc-performance-analysis-optimization-intel-cpus-gpus/.
16. Meyerov I.B., Sysoev A.V., Pirova A.Yu., Shestakova N.V., Ivanchenko M.V. Bridging the Gap Between Applications and Supercomputing: A New Master’s Program in Computational Science // Communications in Computer and Information Science. V. 1129. 2019. P. 529-541.
17. Программирование с использованием модели oneAPI. URL: https://hpc-education.unn.ru/ru/центр-компетенций-oneapi-в-ннгу/курс-oneapi.
18. Programming with oneAPI. URL: https://hpc-education.unn.ru/en/the-oneapi-center-of-excellence/programming-with-oneapi.

Численное моделирование одной задачи атмосферного электричества

И.Г. Милешин[1,2], В.М. Головизнин[3], М.М. Хапаев[3]

[1]НГУ имени Н.И. Лобачевского, математический центр
[2]Филиал МГУ имени М. В. Ломоносова, Саров
[3]МГУ имени М.В. Ломоносова

В работе рассматривается математическая модель атмосферного электричества, известная как концепция глобальной электрической цепи. Для этой модели формулируется нестандартная стационарная эллиптическая краевая задача с неклассическим граничным условием. Для численного решения этой задачи, с целью изучения возможности и эффективности распараллеливания вычислений, используются два численных алгоритма на основе метода конечных элементов. Приводятся результаты предварительных расчетов простых структур.

Ключевые слова: глобальная электрическая цепь, ионосферный потенциал, метод конечных элементов, OpenMP

1. Введение

В настоящее время основной моделью атмосферного электричества является концепция глобальной электрической цепи [1], [2]. Современное состояние этой теории изложено в [3].

В этой модели поверхность земли и ионосфера каждая эквипотенциальны и являются границами атмосферы. Атмосфера является проводящей средой с неоднородной проводимостью. В атмосфере существует разность потенциалов между поверхностью земли и ионосферой и протекают токи проводимости. Кроме того, составной частью модели являются облака, которые являются независимым источником токов. Таким образом, потенциалы земли и ионосферы, токи проводимости в атмосфере и источники тока в облаках образуют распределенную электрическую цепь.

Математическое моделирование реальных структур для этой задачи требует суперкомпьютерных вычислений и эффективного распараллеливания. В нашей работе мы рассматриваем два базовых подхода к численному моделированию электрического потенциала и токов в атмосфере. Оба подхода основаны на методе конечных элементов, но один из них основан на регулярных сетках и многопоточных вычислениях для итерационного решения сеточных уравнений, а другой использует нерегулярные адаптивные сетки и может использовать методы разделения областей, MPI и реализован в пакете FreeFEM++. Целью нашей работы является проведение расчетов для базовых модельных задач с целью исследования и адаптации алгоритмов параллельных вычислений.

2. Постановка стационарной задачи

Коротко изложим вывод основных уравнений и постановку стационарной задачи. Подробно эти вопросы обсуждаются в работах [1], [3] - [6].

Основой стационарной модели глобальной электрической цепи служит одно из уравнений Максвелла:

$$\nabla\times \vec{H}(x) = \sigma(x)\vec{E}(x) + \vec{J}^{ext}(x). \tag{1}$$

Здесь $\vec{H}(x)$ - вектор магнитного поля, $\sigma(x)$ - проводимость атмосферы, $\vec{E}(x)$ - вектор электрического поля, $\vec{J}^{ext}(x)$ - плотность сторонних токов, $x \in R^3$.

В задаче атмосферного электричества рассматривается область гомеоморфная шаровому слою с границей, состоящей из двух компонент связности. Её вид представлен на Рис.

1. В рассматриваемой области Γ_1 - Поверхность Земли. Γ_2 - условная граница, разделяю-

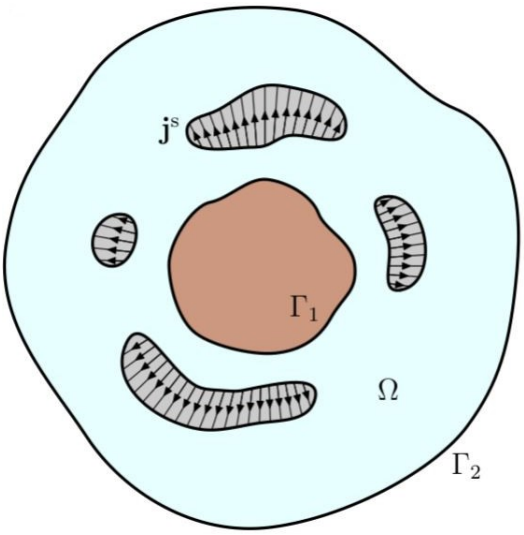


Рис. 1. Схематичный разрез модели земного шара. Границами атмосферы Ω является поверхность земли Γ_1 и нижняя граница ионосферы Γ_2. Облака являются источниками тока.

щая атмосферу и ионосферу. Области J^S (облака) являются генераторами электрических токов в атмосфере. В дальнейшем, будем обозначать рассматриваемую область, где ищется решение, Ω.

Введем потенциал электрического поля

$$\vec{E} = -\nabla\phi(x). \tag{2}$$

Из (1) следует уравнение для нахождение электрического потенциала и вектора магнитного поля

$$\nabla\times\vec{H}(x) = -\sigma(x)\nabla\phi(x) + \vec{J}^{ext}(x). \tag{3}$$

В (3) исключим магнитное поле $\vec{H}(x)$. В результате получим уравнение для нахождения скалярного электрического потенциала $\phi(x)$:

$$\nabla\cdot(\sigma(x)\nabla\phi(x)) = \nabla\cdot\vec{J}^{ext}(x). \tag{4}$$

Уравнение (4) является основным уравнением рассматриваемыой математической модели. Однако кроме (4) из (3) следуют дополнительные интегральные соотношения.

Исходя из того, что область Ω гомеоморфна шаровому слою, для того чтобы переход от (3) к (4) был равносильным, необходимо выполнение интегральных равенств для потоков [4]:

$$\nabla\times\vec{u} = \vec{f} \Leftrightarrow \begin{cases} \nabla\cdot\vec{f} = 0 \\ \oint\limits_{\Gamma_i} \vec{f}\cdot\vec{n}\, d\Gamma_i = 0, \ i = 1,2. \end{cases} \tag{5}$$

Эти соотношения являются следствием теоремы Стокса.

Проводимость земной поверхности намного больше проводимости нижних слоев атмосферы, следовательно нижнюю границу можно считать идеальным проводником. Как известно, проводимость атмосферы растет экспоненциально с ростом высоты, так что проводимость на высоте 70 км намного выше проводимости атмосферы у поверхности Земли. Таким образом верхнюю границу на достаточно большой высоте также можно считать идеальным проводником. Положим [5] $\phi(x) = 0, \quad x \in \Gamma_1$, $\phi(x) = C, \quad x \in \Gamma_2$, где C -

неизвестная константа, которую называют ионосферным потенциалом. В результате получим следующую постановку задачи для нахождения скалярного электрического потенциала (6)-(9):

$$\nabla\cdot(\sigma(x)\,\nabla\,\phi(x)) = \nabla\cdot\vec{J}^{ext}(x), \tag{6}$$

$$\oint\limits_{\Gamma_2}\sigma(x)\cdot\nabla\,\phi(x)\cdot\vec{n}\;d\Gamma_2 = \oint\limits_{\Gamma_2}\vec{J}^{ext}(x)\cdot\vec{n}\;d\Gamma_2, \tag{7}$$

$$\phi(x) = 0, \quad x\in\Gamma_1, \tag{8}$$

$$\phi(x) = C, \quad x\in\Gamma_2. \tag{9}$$

В задаче (6-9) ионосферный потенциал C является искомой величиной. Источники тока $\vec{J}^{ext}(x)$ считаются заданными. Интегральное соотношение (7) позволяет определить ионосферный потенциал C и электрический потенциал $\phi(x)$.

Заметим, что ионосферный потенциал можно найти следующим образом. Представим решение (6-9) в виде суммы $\phi(x) = C\cdot\phi_1(x) + \phi_2(x)$, где $\phi_1(x)$ и $\phi_2(x)$ являются решением следующих стандартных краевых задач:

$$\nabla\cdot(\sigma(x)\,\nabla\,\phi_1(x)) = 0, \tag{10}$$

$$\phi_1(x) = 0, \quad x\in\Gamma_1, \quad \phi_1(x) = 1, \quad x\in\Gamma_2 \tag{11}$$

а также

$$\nabla\cdot(\sigma(x)\,\nabla\,\phi_2(x)) = \nabla\cdot\vec{J}^{ext}\;(x), \tag{12}$$

$$\phi_2(x) = 0, \quad x\in\Gamma_1 = 0, \quad \phi_2(x) = 0, \quad x\in\Gamma_2 = 0. \tag{13}$$

Подставляя выражение для $\phi(x)$ в виде суммы в контурный интеграл, получим

$$\oint\limits_{\Gamma_2}\sigma(x)\cdot(C\cdot\nabla\,\phi_1(x) + \nabla\,\phi_2(x))\cdot\vec{n}\;d\Gamma_2 = \oint\limits_{\Gamma_2}\vec{J}^{ext}\cdot\vec{n}\;d\Gamma_2. \tag{14}$$

откуда следует выражение для ионосферного потенциала

$$C = \frac{\oint\limits_{\Gamma_2}\vec{J}^{ext}\cdot\vec{n}\;d\Gamma_2 - \oint\limits_{\Gamma_2}\sigma(x)\,\nabla\,\phi_2(x)\cdot\vec{n}\;d\Gamma_2}{\oint\limits_{\Gamma_2}\sigma(x)\,\nabla\,\phi_1(x)\cdot\vec{n}d\Gamma_2}. \tag{15}$$

2.1. Вывод обобщенной формулировки задачи

Покажем, что можно определить ионосферный потенциал C иным способом, в котором требуется решать только одну краевую задачу.

Для того, чтобы выполнить конечно-элементную дискретизацию задачи (6)-(9) необходимо сформулировать ее в виде некоторого интегрального тождества (билинейной формы). Приведем необходимые выкладки для получения интегрального тождества для задачи (6)-(9). Пусть $\psi(x)$ - некоторая пробная функция. Домножим правую и левую часть уравнения (6) на эту функцию и проинтегрируем по области Ω. Получим

$$\int\limits_{\Omega}\nabla\cdot(\sigma(x)\,\nabla\,\phi(x))\psi(x)d\Omega = \int\limits_{\Omega}\nabla\cdot\vec{J}^{ext}(x)\psi(x)d\Omega. \tag{16}$$

Воспользуемся известным интегральным тождеством для интегрирования по частям

$$\nabla\cdot\psi\vec{A} = \vec{A}\cdot\nabla\,\psi + \psi\,\nabla\cdot\vec{A}.$$

Преобразовывая (6), получим, что

$$\int_\Omega \nabla\cdot(\sigma(x)\,\nabla\,\phi(x)\psi(x) - \vec{J}^{ext}(x)\psi(x))d\Omega - \int_\Omega \sigma(x)\,\nabla\,\phi(x)\cdot\nabla\,\psi(x)d\Omega = -\int_\Omega \vec{J}^{ext}\,\nabla\,\psi(x)d\Omega. \quad (17)$$

Воспользовавшись теоремой Гаусса-Остроградского для первого интеграла в левой части, получим уравнение

$$\oint_\Gamma (\sigma(x)\,\nabla\,\phi(x) - \vec{J}^{ext})\psi(x)\cdot\vec{n}d\Gamma - \int_\Omega \sigma(x)\,\nabla\,\phi(x)\cdot\nabla\,\psi(x)d\Omega = -\int_\Omega \vec{J}^{ext}\,\nabla\,\psi(x)d\Omega. \quad (18)$$

В (6)проинтегрируем левую и правую часть равенства, а затем воспользуемся теоремой Гаусса-Остроградского. Получим:

$$\oint_\Gamma (\sigma(x)\,\nabla\,\phi(x) - \vec{J}^{ext})\cdot\vec{n}d\Gamma = 0. \quad (19)$$

Далее, функцию $\psi(x)$ будем выбирать в классе функций, равных 0 на границе Γ_1 и равных некоторой константе, например 1, на границе Γ_2. Тогда из (17), (18), (7) а также с учетом того, что главным краевым условием для $\phi(x)$ на Γ_1 является 0, а на границе Γ_2 $\phi(x) = C$, получим:

$$\oint_\Gamma (\sigma(x)\,\nabla\,\phi(x) - \vec{J}^{ext})\psi\cdot\vec{n}d\Gamma = 0. \quad (20)$$

Таким образом, основное интегральное тождество имеет простой вид

$$\int_\Omega \sigma(x)\cdot\nabla\,\phi(x)\cdot\nabla\,\psi(x)\;d\Omega = \int_\Omega \vec{J}^{ext}\cdot\nabla\,\psi(x)\;d\Omega. \quad (21)$$

В этом тождестве, или билинейной форме, неявно присутствующий ионосферный потенциал C является одной из искомых величин. Сама билинейная форма (21) очень похожа на билинейную форму для задачи Дирихле.

Для завершения постановки задачи формально определим пространство, в котором ищется решение $\phi(x)$: $V(\Omega) = \{\phi(x) \in H^1(\Omega), \phi(x) = 0, x \in \Gamma_1, \phi(x) = C, x \in \Gamma_2\}$. В этом же пространстве следует выбирать пробную функцию $\psi(x)$, полагая $\psi(x) = 1$, $x \in \Gamma_2$.

Корректность задачи (21) доказывается с помощью леммы Лакса-Мильграма. Доказательство существования и единственности задачи подробно рассмотрено в [6].

3. Численная реализация двумерной задачи на структурированной четырехугольной сетке

Все расчеты проводились для двумерных областей в полярной системе координат. Этими областями являлись кольцо и сектора с периодическими краевыми условиями ("цилиндрическая земля").

Рассмотрим задачу в кольце с радиусами, соответствующими радиусу Земли и радиусу ионосферы: $r \in [6370000, 6440000]$(м), $\theta \in [0, 2\pi]$. Тут 6440000.0 м - расстояние от центра Земли до условной границы, разделяющей атмосферу и ионосферу. Таким образом, данная область в полярных координатах представляет тонкий прямоугольник со склеенной стороной (цилиндр). Проводимость имеет вид

$$\sigma(r) = \sigma_0 e^{\frac{r-r_0}{r_s}}\ \text{См/м}, \quad (22)$$

$$r_s = 5000\ \text{м}, \quad r_0 = 6370000.0\ \text{м}, \quad (23)$$

$$\sigma_0 = 5\cdot 10^{-14}\ \text{См/м}. \quad (24)$$

Сторонние токи $\vec{J}^{ext}(r,\theta)$ локализованы внутри рассматриваемой области.

В дальнейшем, для краткости будем опускать единицы размерности.

Простая структура области позволяет ввести структурированную сетку с прямоугольными ячейками. С помощью этой сетки мы строим известный метод конечных элементов, в котором по каждому направлению используется кусочно-линейная аппроксимация [7]. Элементы матрицы сеточных уравнений имеют вид

$$A_{i,j} = \int\limits_{\Omega} \sigma(r,\theta) \cdot \nabla u_i(r,\theta) \cdot \nabla u_j(r,\theta)\ d\Omega,$$

где $u_i(r,\theta)$ являются билинейными функциями конечно-элементной аппроксимации. Метод конечных элементов настоящей работы отличается от классического только тем обстоятельством, что все точки на границе ионосферы оказываются склеены, так как они соответствуют единственной неизвестной, ионосферному потенциалу. В свою очередь, это обстоятельство приводит к повышенной заполненности соответствующего столбца и строки разреженной матрицы сеточных уравнений.

Элементы вектора правой части имеют вид

$$b_i = \int\limits_{D} \vec{J}^{ext} \cdot \nabla \phi_i(x)\ ds. \tag{25}$$

В этой формуле область D представляет собой совокупность ячеек сетки, в которых правая часть $\vec{J}^{ext}(x)$ отлична от 0. В проводившихся расчетах это была некоторая прямоугольная область, а ток $\vec{J}^{ext}(x)$ имел постоянное значение и направление.

Матрица системы сеточных уравнений такого метода конечных элементов является сильно разреженной. В строке матрицы для внутренних узлов имеется не более 9 ненулевых элементов. Матрица является симметричной и положительно определенной по построению.

Для работы с разреженными матрицами использовался пакет Eigen [8]. Так как матрица системы сеточных уравнений симметрична и положительно определена, для решения системы линейных уравнений был использован алгоритм сопряженных градиентов с диагональным предобуславливанием. Для сборки матрицы и умножения матрицы на вектор использовались средства OpenMP.

Программа написана на C++.

С помощью описанной программы был проведен ряд расчетов. В этих расчетах число узлов достигало нескольких миллионов.

Один из расчетов был проведен для сравнения с известными результатами работы [5]. Источники тока в этой работе имели радиальное направление. Вычислялись значения полярного потенциала, график результатов приведен на Рис. 2. Эти результаты согласуются с данными [5].

Представляет также интерес зависимость времени счета от размерности системы и числа потоков. Эти результаты приведены на Рис. 3. Такая простая многопоточность позволяет легко получить заметное практическое ускорение вычислений. Однако, как видно из графиков, ускорения, пропорционального числу процессов, достичь не удается.

4. Численная реализация с использованием пакета FreeFEM++

Кроме максимально простого подхода, описанного в предыдущем разделе, представляет интерес алгоритм, допускающий нерегулярные треугольные (в двумерном случае) сетки, области сложной структуры, адаптацию сетки к решениям, эффективный метод разделения областей, полномасштабные параллельные вычисления с помощью аппарата MPI, удобные средства визуализации и постпроцессинга решения, потенциальный переход к решению реальных трехмерных задач. Такими возможностями обладает известный пакет FreeFEM++,

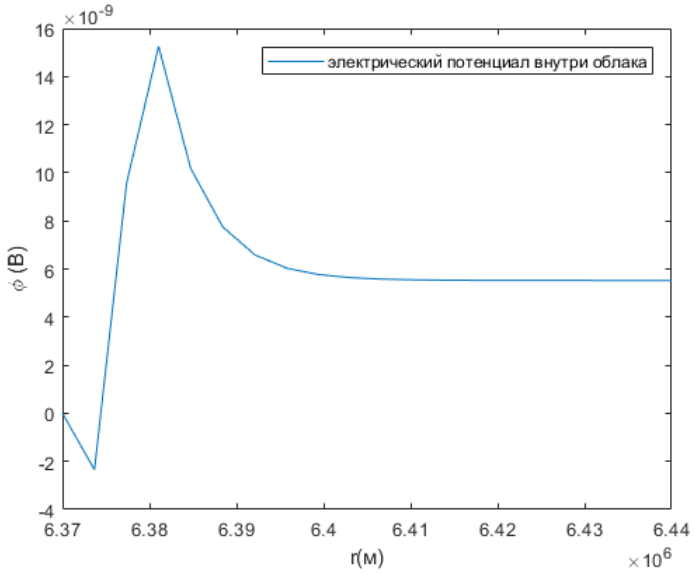


Рис. 2. Зависимость электрического потенциала от радиуса. Структура решения хорошо согласуется с физическими особенностями задачами. Наблюдается скачок потенциала около облака. Также наблюдается плавное изменение решения при удалении от облака.

который является в значительной степени энциклопедией метода конечных элементов и сопутствующих алгоритмов, а также имеет внутренний язык программирования, основанный на C/C++.

Хотя пакет FreeFEM++ допускает программирование билинейной формы (21), в этой работе мы рассматриваем подход на основе решения двух краевых задач (10, 11), (12, 13) и формулы (15).

Коротко опишем этапы решения задачи с помощью пакета FreeFEM++.

1. Средствами пакета (C++ -подобный скриптовый язык) описывается геометрия задачи, поверхность земли и ионосферы, а также облака. В рассматриваемой геометрии облако моделируется прямоугольником в полярных координатах, но в принципе может иметь любую форму. Для этого необходимо аналитически задать 2 границы (окружности радиуса 6370000.0 и 6440000.0 соответственно) и внутренние границы для облака. Каждая из границ приближается множеством отрезков. Далее с помощью логических операций необходимо описать внутренность области.

2. Генерация начальной сетки. Сгенерируем начальную сетку с использованием триангуляции Делоне для исходной области. Исходя из величин радиусов окружностей (6370000.0 и 6440000.0) оказывается, что для подробного описания области требуется несколько миллионов узлов и треугольных ячеек сетки.

3. С использованием внутреннего синтаксиса FreeFEM++ необходимо записать решаемые задачи в виде интегральных тождеств. Это делается на основе (10,11,), (12, 13). После описания билинейных форм и правых частей можно выполнить решение задач.

4. Сначала вычисления проводятся на начальной простой сетке, затем можно выполнить адаптацию сетки к решению. Это делается с помощью итерационной процедуры. Для измельчения или загрубления сетки пакет использует индикаторы точности решения, которые вычисляются для ячеек сетки.

5. Визуализация решения. Пакет имеет встроенные средства, однако в нашем случае сетка настолько велика, что требуется использование внешних программ, для которых

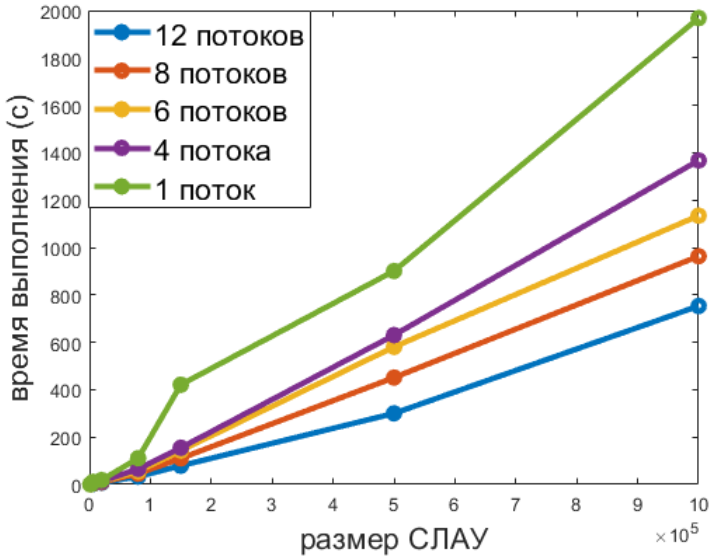


Рис. 3. Зависимость времени выполнения программы от размерности системы и числа потоков OpenMP. Расчеты проводились на 6 ядерном процессоре Intel Core i7-9750h. Для расчетов было доступно 16 гб оперативной памяти. Простая многопоточность позволяет заметно ускорить вычисления в задачах, не требующих алгоритмов разделения области.

пакет FreeFEM++ может подготовить все данные.

6. Обработка (постпроцессинг) полученного решения. Это делается средствами внутреннего скриптового языка пакета, который является полноценным языком программирования, дополненным библиотеками алгоритмов для удобных вычислений.

На Рис. 4 для той же задачи, что и в предыдущем разделе, представлен пример части сгенерированной сетки внутри области с распределением скалярного электрического потенциала после проведения необходимой серии расчетов. Для решения полной исходной задачи требуются значительные вычислительные ресурсы, для чего планируется использование суперкомпьютера.

Следует отметить, что при адаптации сетки она измельчается при приближении к облаку. Кроме того, структура сетки учитывает экспоненциальную структуру проводимости. Эти особенности приводят к тому, что строится детальная сетка в тех подобластях, где происходит резкое изменение решения, и крупная сетка в тех подобластях, где изменение решения происходит не столь значительно.

Для решения систем линейных уравнений, которые возникают в ходе решения задач, имеется возможность распараллеливания с использованием средств FreeFEM++. Доступно большое число как прямых, так и итерационных алгоритмов. Для данной задачи использовался метод сопряженных градиентов из известного пакета PETSc.

По сравнению с четырехугольной структурированной сеткой, алгоритм FreeFEM++ допускает построение треугольной сетки с меньшим количеством узлов. Тем не менее, при решения задач близких к реальным, число узлов все равно остается достаточно большим, что делает необходимым распараллеливание алгоритма. Поэтому решение системы линейных уравнений осуществлялось с использованием алгоритмов распараллеливания MPI, который является частью пакета.

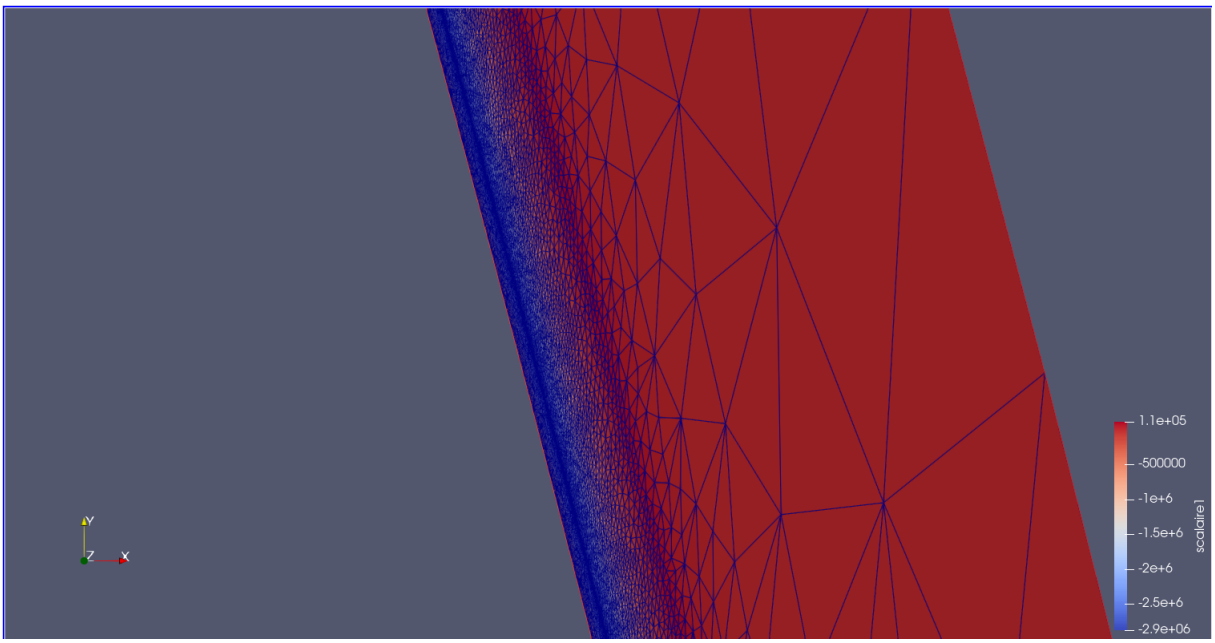


Рис. 4. Пример адаптивной расчетной сетки и распределения электрического потенциала. После проведения сеточных адаптаций сетка учитывает физические особенности задачи. Произошло сгущение сетки в областях с высоким градиентом решения. Сетка более подробная в облаке и его окрестности. Кроме того, сетка учитывает экспоненциальную структуру проводимости по радиальному направлению.

5. Выводы

В работе описаны два варианта метода конечных элементов решения задачи атмосферного электричества.

В первом подходе используются структурированные сетки и простой билинейный метод конечных элементов, а параллельность вычислений достигается многопоточностью на этапе формирования матрицы системы сеточных уравнений и итерационного решения этой системы. Можно сказать, что этот подход следует идее векселизации вычислений (простой однородный метод, прямоугольные регулярные сетки), которая в нашем случае служит хорошим приближением для физической проблемы. Достоинством этого подхода является также достаточно простой переход к полной трехмерной задаче, где, однако, потребуется полное распараллеливание с использованием алгоритмов разделения области.

Второй вариант использует более универсальный и эффективный метод конечных элементов и реализованный в пакете FreeFEM++. Этот подход имеет много практических преимуществ перед простым методом конечных элементов, так как данный пакет является полноценной средой программирования вычислительных алгоритмов и не требует программирования на уровне сетки и конечных элементов, поскольку содержит обширный набор известных конечноэлементных аппроксимаций. FreeFEM++ включает в себя множество свободно доступных библиотек решения систем сеточных уравнений, а также включает в себя алгоритмы разделения областей. Также этот пакет содержит готовую реализацию параллельных вычислений в системе MPI. Однако переход к полной трехмерной задаче в этом пакете не является тривиальным.

Интересно, что простой метод на прямоугольной сетке сохраняет конкурентоспособность на достаточно больших задачах и заслуживает дальнейшего развития, например, применения иерархически измельчаемых сеток.

Литература

1. Leblanc F., Aplin K.L., Yair Y., Harrison R.G., Lebreton J.P., Blanc M. Planetary Atmospheric Electricity. Springer, 2008. 531 p.

2. Wilson C.T.R. Investigations on lighting discharges and on the electric field of thunderstorms // Phil. Trans. Roy. Soc. Lon. A. 1921. V. 221. P. 73–115.

3. Anisimov S.V., Bakastov S.S., Mareev E.A. Spatiotemporal structures of electric field and space charge in the surface atmospheric layer // Journal of Geophysical Research. 1994. V. 99, P. 10603–10610

4. Kalinin A.V., Slyunyaev N.N. Initial-boundary value problems for the equations of the global atmospheric electric circuit // Journal of Mathematical Analysis and Applications 2017. vol. 450(1). P. 112–136

5. Slyunyaev N.N., Mareev E.A., Kalinin A.V., Zhidkov A.A. // Influence of large-scale conductivity inhomogeneties in the atmosphere on the global electric circuit. J.Atmos. Sci. 2014. V.71, no. 11. P. 4382–4396.

6. Жидков А.А., Калинин А.В. Корректность одной математической задачи атмосферного электричества // Вестник ННГУ им. Н.И. Лобачевского. 2009. № 4. С. 123–129.

7. Jian-Ming Jin. The Finite Element Method in Electromagnetics. Wiley: 2014. 876 p.

8. https://eigen.tuxfamily.org/index.php

Аннотации стендовых докладов

Development Of The ADP Interatomic Potential Model Using The Kokkos C++ Parallel Programming Library For The Supercomputer Molecular Dynamics Package LAMMPS[1]

V.S. Galigerov[1], V.P. Nikolskiy[1,2]

National Research University Higher School of Economics[1], Joint Institute for High Temperatures of the Russian Academy of Sciences[1,2]

One of the most frequently performed tasks on supercomputers is the research of chemical properties using molecular dynamics techniques. For example, in materials science, it is sometimes too expensive or even impossible to experiment to study the properties of alloys at high pressures or high and low temperatures, so scientists use computer simulations.

The LAMMPS is an open-source package for classical molecular dynamics computation written in C++ [4, 5]. It is designed to be compiled and run both on local computers and supercomputers and allows the simulation of up to tens of millions of particles. It implements a large number of models of interaction potentials, ranging from the most popular to exotic ones.

Angular dependent potential (ADP) is a semi-empirical method, introduced by Mishin [1], and it is an extension of an earlier Embedding Atom Method [2]. ADP is mostly used to simulate the behavior of atoms in metals and metal alloys. The LAMMPS package implements this type of potential, and this implementation runs only on CPU cores [3].

It appears that the use of scientific computing on graphics processing units (GPUs) is growing considerably nowadays because of their multi-core nature, which can show better performance in molecular dynamics tasks, and this fact applies both to computing on personal computers and supercomputers. The problem is that different GPU and CPU architectures can differ in the most productive data layout and access pattern, so we need to write not only portable code that will work on these architectures, but also code that is equally productive on these platforms. Described problem is called the performance portability issue. We can solve this problem with the Kokkos library, which is a performance portable open-source project designed for almost all existing supercomputer systems [6, 7]. To use it, it is enough to write the code once, and then it is only needed to specify the necessary compilation parameters for the target platforms. The other two problems are the process communication for the code executed on GPU via MPI and the communications between host and device, which are the bottlenecks of the hybrid computations.

This work describes our implementation of the ADP method in the LAMMPS package using the Kokkos library. We first outline the LAMMPS source code structure and how it implements the Angular dependent potential. We end up with implementation details on our PairADPKokkos class inside the LAMMPS package.

We tested our implementation on the HSE supercomputer cHARISMa [8] and got sevenfold acceleration on the GPU Nvidia V100 compared to the CPU-based implementation on 22 cores of the Intel Xeon Gold. In addition, this new version should be equally performant on different platforms (this fact was not tested in this work).

As a result, we created a pull request to the LAMMPS repository on GitHub [9]. This implementation was reviewed by the source code maintainers, merged to *develop* branch and published in 4 May 2022 patch release. This functionality will be published with the Stable Release Summer 2022 [10].

References

1. Mishin Y., Mehl M., Papaconstantopoulos D. Acta Mater 2005;53:4029.
2. Daw M.S., Baskes M.I.. Phys Rev B 1984;29:6443.

[1] The article was prepared within the framework of the HSE University Basic Research Program.

3. Singh C.V., Warner D.H.. Acta Mater. 2010.; DOI:10.1016/j.actamat.2010.06.055. – p. 5798
4. LAMMPS documentation. [online]. Available at: https://docs.lammps.org/Manual.html (Accessed 10.03.2022).
5. Thompson A.P., Aktulga H.B., et al. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. Computer Physics Communications. 2022. DOI:10.1016/j.cpc.2021.108171.
6. Kokkos API reference. [online]. Available at: https://github.com/kokkos/kokkos/wiki (Accessed: 10.03.2022).
7. Carter Edwards H., Trott C. R., Sunderland D. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. Journal of Parallel and Distributed Computing. 2014. DOI:10.1016/j.jpdc.2014.07.003.
8. 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. DOI: https://doi.org/10.1088/1742-6596/1740/1/012050
9. Kokkos accelerated variant for adp pair style. [online]. Available at: https://github.com/lammps/lammps/pull/3247 (Accessed: 21.06.2022).
10. Stable Release Summer 2022. [online]. Available at: https://github.com/lammps/lammps/milestone/9?closed=1 (Accessed: 21.06.2022).

Learning ice accretion with Graph Neural Networks

S.S. Shumilin

Joint Supercomputer Center of Russian Academy of Sciences

Graph Neural Networks (GNNs) have become a hot topic recently [1]. Many problems with unstructured data are now effectively solved using various GNN architectures. For instance, predicting molecular properties where molecules are represented as graphs became widespread.

As a subfield Geometric Deep Learning [2] studies application of GNNs to spatial data. It works with problems such as point cloud segmentation, analyzing spatial properties of graphs, and so on [3].

In present work we study application of GNNs to remeshing. As a target task we chose ice accretion which assumes evolution of surface triangular mesh. We represent a surface mesh as a spatial graph and train a GNN to model the process of ice accretion. The work is inspired by recent advances in physical modeling with GNNs [4].

To do so, we implement modern architectures such as GCN (Graph Convolutional Network), EdgeConv (convolution on edges). We compare out model with traditional remeshing methods in terms of complexity and show that our model produces comparable results. It may reduce computation time by orders of magnitude.

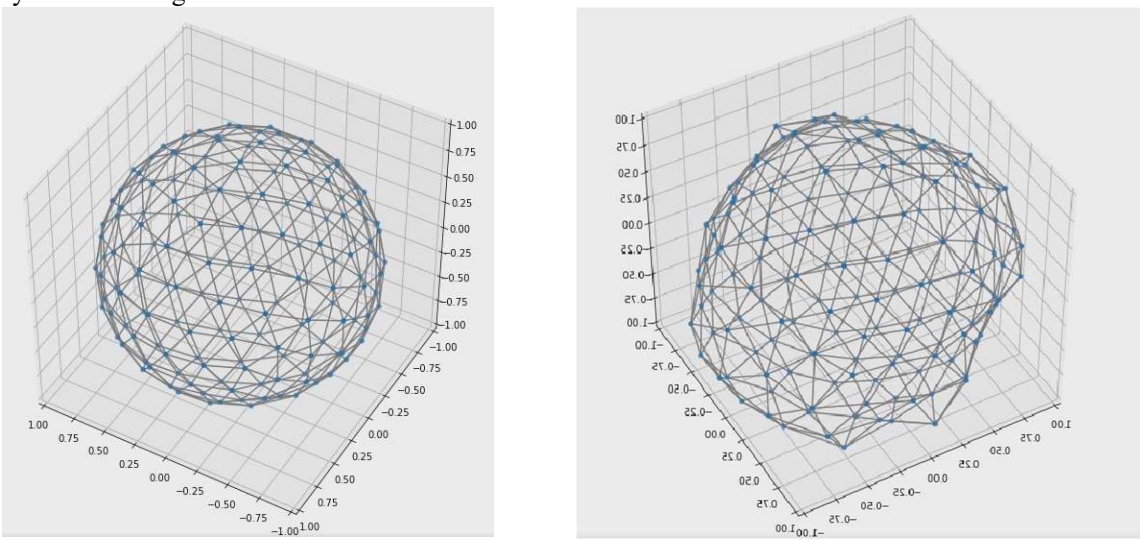


Figure 1. Surface mesh before (left) and after (right) "ice accretion" Results (right) of our model reproduce results of the traditional algorithm with 10e-5 error rate so the right mesh coincide with the original one

As a ground truth algorithm we used remeshing from [5].

The algorithm includes several steps:

solving MSE problem with spectral decomposition of covariance matrix of face normals
mesh smoothing algorithm called null-space smoothing
distribution of ice volume among faces
normal smoothing

All (except for the first step) may be extended by increasing the number of iterations. The better result we want to get the more iterations we need to do. So the time complexity of the algorithm raises. We train a GNN, which incorporates the whole algorithm, and show, in terms of volume error, that our approach helps get similar results with less time.

Our model consists of several EdgeConv layers with skip connections. As a training set, we use generated meshes of different configurations. Output of our model is a 4-dimentional vector representing direction and magnitude of displacement. Thus for every node in the mesh we get displacement and then we move the nodes.

Physical simulation of ice accretion may be performed in parallel using supercomputers. However, remeshing is hard to parallelize so our approach reduces the wall time of the whole simulation pipeline. We may achieve this by pretraining a model and then use it, as a step is pipeline. We trained our model on a single GPU with time proportional to the mesh size.

References

1. Zhou, Jie & Cui, Ganqu & Hu, Shengding & Zhang, Zhengyan & Yang, Cheng & Liu, Zhiyuan & Wang, Lifeng & Li, Changcheng & Sun, Maosong. (2020). Graph neural networks: A review of methods and applications. AI Open. 1. 57-81. 10.1016/j.aiopen.2021.01.001.
2. Bronstein, Michael & Bruna, Joan & Cohen, Taco & Veličković, Petar. (2021). Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. DOI: https://doi.org/10.48550/arXiv.2104.13478
3. Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. DOI: https://doi.org/10.48550/arXiv.2111.14522
4. Pfaff, Tobias & Fortunato, Meire & Sanchez-Gonzalez, Alvaro & Battaglia, Peter. (2020). Learning Mesh-Based Simulation with Graph Networks. DOI: https://doi.org/10.48550/arXiv.2010.03409
5. Xialoing Tong, David Thompson, Qiuhan Arnoldus, Eric Collins and Edward Luke. Three-Dimensional Surface Evolution and Mesh Deformation for Aircraft Icing Applications. Journal of aircraft. 2016.

Numerical simulation of catalyst grain during oxidative regeneration with MPI technologies*

O. V. Grishaeva[1], I. M. Gubaydullin[2], E. E. Peskova[1], O. S. Yazovtseva[1]

National Research Mordovia State University[1],
Institute of Petrochemistry and Catalysis of the Russian Academy of Sciences[2]

The development of mathematical models of catalytic processes is a non-trivial task that requires both an extensive experimental base and effective methods for solving the resulting problems. It is necessary to take into account heat and mass transfer and chemical transformations for modelling [1]- [3]. The combination of such processes with a large difference in the characteristic times leads to stiff systems of equations and, accordingly, a multiple increase in the computational complexity of the algorithms for their solution [4].

The catalyst grain model during oxidative regeneration is written in dimensionless form:

$$\begin{cases} \dfrac{\partial \Theta}{\partial \tau} = \dfrac{D^* \tau_k}{R_z^2} \dfrac{1}{\rho^2} \dfrac{\partial}{\partial \rho} \left(\rho^2 \dfrac{\partial \Theta}{\partial \rho} \right) + \dfrac{\hat{S} c_0}{T_{op} c} \sum\limits_{j=1}^{5} Q_j \omega_j, \\ \dfrac{\partial y_i}{\partial \tau} = \dfrac{D^* \tau_k}{R_z^2 \varepsilon} \dfrac{1}{\rho^2} \dfrac{\partial}{\partial \rho} \left(\rho^2 \dfrac{\partial y_i}{\partial \rho} - \rho^2 \hat{\mu} y_i \right) + \dfrac{\hat{S}}{\varepsilon} \sum\limits_{j=1}^{5} \nu_{ij} \omega_j, \\ \dfrac{\partial}{\partial \rho} \left(\rho^2 \hat{\mu} \right) = \rho^2 \dfrac{R_z^2 c_0 \hat{S}}{D^* \tau_k \gamma} (-\omega_1 + \omega_3 + \omega_5), \\ \dfrac{\partial q_c}{\partial \tau} = -\dfrac{M_C c_0}{\gamma} \hat{S} (\omega_2 + \omega_3 + \omega_5), \\ \dfrac{\partial z_1}{\partial \tau} = \dfrac{c_0}{\gamma q_c} \hat{S} (\omega_6 + z_1 M_C (\omega_2 + \omega_3 + \omega_5)), \\ \dfrac{\partial z_2}{\partial \tau} = \dfrac{c_0}{\gamma q_c} \hat{S} (\omega_7 + z_2 M_C (\omega_2 + \omega_3 + \omega_5)), \\ \dfrac{\partial \theta_1}{\partial \tau} = -\hat{S} \left(\omega_4 + \dfrac{c_0}{\gamma} \omega_6 \right), \\ \dfrac{\partial \theta_2}{\partial \tau} = \hat{S} \left(2\omega_1 - \omega_3 + \omega_4 - 2\omega_5 - \dfrac{c_0}{\gamma} \omega_7 \right). \end{cases} \quad (1)$$

Here ρ – dimensionless catalyst grain radius, $\rho \in [0, 1]$ (independent spatial variable); τ – dimensionless time, $\tau \in [0, +\infty)$ (independent time variable); $\Theta(\rho, \tau)$ – dimensionless catalyst grain temperature; $y_i(\rho, \tau)$, $i = \overline{1,4}$ – mole fraction of components in the gas phase of the reaction (index 1 corresponds to oxygen, 2 – carbon monoxide, 3 – carbon dioxide, 4 – water); $\hat{\mu}(\rho, \tau)$ is the dimensionless velocity of the Stefan flow; $q_c(\rho, \tau)$ – mass fraction of coke on catalyst grain; $z_1(\rho, \tau)$ and $z_2(\rho, \tau)$ – mass fractions of hydrogen and oxygen in the coke sediments; $\theta_1(\rho, \tau)$ and $\theta_2(\rho, \tau)$ are fractions of hydrogen-carbon and oxygen-carbon complexes on the coke granule surface; $\hat{S}(\rho, \tau)$ – dimensionless area of coke granules; $\omega_j(\rho, \tau)$, $j = \overline{1,5}$ – dimensionless rates of quasi-homogeneous reactions taken from the kinetic scheme; $\omega_j(\rho, \tau)$, $j = \overline{6,7}$ – rates of heterogeneous reactions taken from the kinetic scheme, g/mol; D^* – effective diffusion coefficient, m^2/sec; τ_k – contact time, sec; R_z – catalyst grain radius, m; ε – catalyst grain porosity; ν_{ij}, $i = \overline{1,4}$ – stoichiometric coefficients from the reaction scheme; c_0 – gas molar density, mol/m^3; $T_{op} = 520°$ is the temperature at which the reaction rate constants are experimentally determined, K; c – volumetric heat capacity of the catalyst, J/(m^3· K); Q_j, $j = \overline{1,5}$, – thermal effects of chemical reactions, J/mol; γ – catalyst bulk density, g/m^3; M_C – molecular weight of coke, g/mol.

*The work was carried out within the state task of the Institute of Petrochemistry and Catalysis of Russian Academy of Sciences (No. FMRS-2022-0078).

It should be noted that there is a decrease in the volume of coke sediments over time. This fact is taken into account in the (1) model as a decrease of the reaction surface area $\hat{S}(\rho,\tau) = \left(\frac{q_c(\rho,\tau)}{q_c(\rho,0)}\right)^{\frac{2}{3}}$, while the catalyst grain size remains unchanged.

The (1) system is supplemented with boundary and initial conditions chosen based on the experimental conditions:

$$\rho = 0: \ \hat{\mu} = 0, \ \frac{\partial y_i}{\partial \rho} = 0, \ \frac{\partial \Theta}{\partial \rho} = 0; \rho = 1: \ \frac{\partial y_i}{\partial \rho} = 0, \ \frac{\partial \Theta}{\partial \rho} = \beta_0 \left(\frac{T_0}{T_{op}} - \Theta\right);$$

$$\tau = 0: \ q_c = q_C^0, \ z_1 = z_1^0, \ z_2 = 0, \ \theta_1 = \theta_1^0, \ \theta_2 = 0, \Theta = \frac{T_0}{T_{op}}, \ y_1 = 1, \ y_i = 0, i = \overline{2,4},$$

where T_0 is the initial temperature of the catalyst grain, K.

The solution of the diffusion and heat transfer equations in the (1) system is based on the integro-interpolation method. The solution of kinetic equations is carried out separately (splitting by physical processes) by the three-stage implicit Runge-Kutta method of the fifth accuracy order – RADO II A [4].

The resulting difference scheme formed the basis of software written in C++. A theoretical investigation of stability and convergence is impossible due to the complexity of the right side of the (1) system. The investigation was conducted for a different number of computational cells. A conclusion about numerical algorithm stability was made from its correct operation.

Calculations are performed using the MPI standard in this article. The choice of technology is due to several factors. Earlier, the authors developed a parallel algorithm using Open MP technology for numerical simulation of the oxidative regeneration process taking into account the simple geometry of the region. But at the same time, solving kinetic problems in the general body of the program leads to a strong grinding of the step, which entails large time costs for calculations. In this study, splitting by physical processes is applied in order to solve the chemical kinetics equations separately, which makes the use of the OpenMP standard ineffective, since parallel calculations will be applied only for only three equations.

The developed parallel algorithm has shown its effectiveness in simulation of oxidative regeneration at various technological parameters. The model studied in this article will be used as part of the catalyst layer model in the process of oxidative regeneration, and the developed algorithm will be expanded to a new model in the future.

References

1. Masagutov, R. M., Morozov, B. F., Kutepov, B. I.: Regeneration of catalysts in oil processing and petrochemistry. Moscow, USSR (1987)

2. Gubaydullin, I. M.: Mathematical modelling of dynamic modes of oxidative regeneration of catalysts in motionless layer. Ufa, Russia (1996)

3. Gubaydullin, I. M., Yazovtseva, O. S.: Investigation of the averaged model of coked catalyst oxidative regeneration. In: Computer Research and Modeling, vol. 13, no. 1, pp. 149–161 (2021). DOI: 10.20537/2076-7633-2021-13-1-149-161.

4. Hairer E., Wanner G.: Solving Ordinary Differential Equations. Stiff and Difderential-Algebraic Problems. 2nd edition. Springer Series in Comput. Math, vol. 14 (1996)

Numerical simulation of unsteady chemically non-equilibrium flow problems with parallel-in-time algorithms

P.D. Toktaliev[1], S.I. Martynenko[1,2,3]

Institute of Problems of Chemical Physics of RAS[1], Joint Institute for High Temperatures of RAS [2], Bauman Moscow State Technical University[3]

The importance of simulating chemically non-equilibrium flows can't be underestimated: the vast majority of the environmental and industrial flows are chemically non-equilibrium. And for many industries prediction, design and optimization of characteristics of such flows are crucial, these areas include energy, transport, oil, chemical industries and many others. Despite the great importance and significant development of non-continuous approaches last decades which are used extensively in the scope of molecular dynamics and quantum chemistry, continuum mechanics stays the most widespread known area of application of parallel and HPC algorithms. Thus, we restrict our attention in the current paper to computational fluid mechanics and numerical solutions of reactive Navier-Stokes equations.

Usually to accomplish the parallelization of a serial numerical procedure computational domain decomposition methods or redistribution of objects for meshless methods are applied, thereby providing parallel-in-space execution of the whole problem. And for many particular steady problems, these approaches could give good or even close to theoretically optimal performance. However, the time-stepping procedure for unsteady problems, is usually sequential and doesn't allow optimal use of a set of parallel workers. For systems with exaflops performance, it is already clear that a sequential-in-time procedure is the critical bottleneck of the entire computational algorithm. One potential solution to this problem is to use a multigrid time approach, where the time variable is treated in a similar way to the spatial variables, with the construction of a grid hierarchy in the time dimension. Potentially, this approach allows to obtain linear scalability in time for the systems of the exaflops range and, due to the use of the multigrid approach, also in space – close to the optimal computational performance.

There are several ways to extend the multigrid approach to the time dimension including space-time multigrid. We consider in the current paper multigrid-reduction-in-time algorithm (MGRIT)[1] which generalizes a well-known parareal algorithm to multiply levels of grid hierarchy and implementation of MGRIT from [2]. One of the advantages of the implementation [2] is that it can be used over the existing implementation of the solver with only minor source modifications. Thus, we used in-house solver for incompressible reactive Navier-Stokes implemented in PETSc [3], based on parallel unstructured mesh interface DMPLEX and set of the time stepping routines TS, to couple with parallel-in-time algorithms. As a linear solver we used GMRES.

Two model benchmarks are considered and analyzed in the paper – reactive flow in the plane channel with two reactions in the bulk phase, reactions mimic the real oxidation mechanism and introduce stiffness into the system. Comparison between serial and two-level time stepping algorithm performance for non-stiff set of first benchmark parameters is represented in Fig.1. It should be mentioned, that for these computations explicit time stepping scheme was taken for both levels of MGRIT algorithm and stability restrictions were satisfied through benchmark parameters tuning. It can be clearly seen from the figure, that for small number of processes the serial procedure is much faster than parallel one and

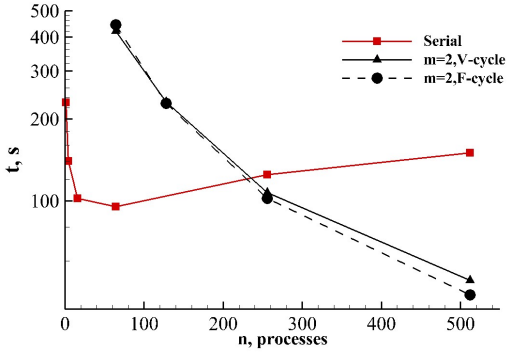


Fig.1. Performance comparison for serial and two versions of MGRIT time stepping procedure

graphs have intersection approximately near the point n~210 processes. But for n>210, both MGRIT versions give much better results for absolute CPU effort and performance graph slope than the serial procedure. The second benchmark is the well-known unsteady laminar flow problem around a plane cylinder with imposed surface adsorbtion-desorbtion reactions.

For both benchmarks performance of the sequential in time and parallel in time cases in HPC environment are compared and performance bottlenecks are identified.

Funding: The reported study was funded by RFBR and TUBITAK, project number 21-51-46007.

References

1. S. Friedhoff, R. D. Falgout, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel, in Sixteenth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, United States, Mar 17 - Mar 22, 2013, 2013, online via http://www.osti.gov/scitech/servlets/purl/1073108.
2. N. Abel, J. Chaudhry, R. D. Falgout, J. B. Schroder, Multigrid-Reduction-in-Time for the Rotating Shallow Water Equations, 2020, LLNL Technical Report, LLNL-TR-813511
3. S. Balay, W. D. Gropp, L. Curfman McInnes, B. F. Smith. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries // Modern Software Tools in Scientific Computing, 1997, pp.163-202

Виртуальный тур по суперкомпьютерному комплексу

М.А Тимошкин, С.И. Соболев

Научно-исследовательский вычислительный центр Московского государственного университета имени М.В. Ломоносова

Суперкомпьютерный комплекс – это огромная и чрезвычайно сложная техническая конструкция, занимающее специально подготовленное помещение или даже здание. Для обеспечения безопасности к подобным объектам поддерживается ограниченный режим доступа персонала. При этом интерес к устройству суперкомпьютера и желание увидеть его собственными глазами отмечается у людей, так или иначе связанных с высокопроизводительными расчетами, у студентов, изучающих технологии параллельных вычислений и даже у школьников в рамках курса информатики. В НИВЦ МГУ многие годы существует практика организации экскурсий в Суперкомпьютерный комплекс МГУ [1]. Экскурсии включают в себя осмотр аппаратных помещений суперкомпьютера «Ломоносов-1» и рассказ об устройстве суперкомпьютера в целом и его отдельных компонентов. Однако из-за пандемии COVID-19 в 2020 году экскурсии были приостановлены. При этом было принято решение о создании сервиса виртуальных экскурсий, позволяющего подробно ознакомиться с устройством суперкомпьютера без посещения места его установки.

В качестве технологической основы сервиса была взята технологий трехмерной визуализации суперкомпьютерного комплекса, разработанная в рамках проекта Octotron [2]. Для задач этого проекта была создана модель суперкомпьютерного комплекса, отражающая его основные компоненты и связи между ними, средства визуализации этой модели и отображения на ней состояния компонентов суперкомпьютера. Функциональность данного сервиса была значительно расширена в части представления информации о суперкомпьютере и методов ориентации пользователей в виртуальном пространстве.

Сервис виртуального тура работает в веб-браузере. Он создается с помощью JavaScript, HTML и CSS с использованием популярной библиотеки для трехмерной визуализации Three.js [3]. В настоящее время реализован виртуальный тур по суперкомпьютерному комплексу «Ломносов-1» [4]. Все использованные текстуры – фотографии реальных компонентов суперкомпьютера. Созданная трёхмерная модель суперкомпьютера очень близка к реальности, но не является его точной копией.

Отличительной особенностью виртуального тура является интерактивное взаимодействие с компонентами суперкомпьютера. Сервис предоставляет пользователю широкий функционал для взаимодействия с элементами суперкомпьютера, начиная от получения информации по каждому компоненту и разбиения сложных объектов на составляющие и заканчивая возможностью изменения способа перемещения в виртуальном пространстве или включения режима автоматической экскурсии.

На данный момент сервис лучше всего поддерживается на компьютерах из-за сравнительно высоких требований к графической подсистеме. Сейчас ведутся работы по оптимизации тура и снижению минимальных требований. В разработке находится аналогичный тур по суперкомпьютерному комплексу «Ломоносов-2».

Литература

1. Суперкомпьютерные технологии - школьникам. URL: https://parallel.ru/info/excursion.html (дата обращения: 01.09.2022).
2. Alexander Antonov, Dmitry Nikitenko, Pavel Shvets, Sergey Sobolev, Konstantin Stefanov, Vadim Voevodin, Vladimir Voevodin, and Sergey Zhumatiy. An approach for ensuring reliable functioning of a supercomputer based on a formal model. In: Parallel Processing and Applied Mathe-

matics. 11th International Conference, PPAM 2015, Krakow, Poland, September 6-9, 2015. Revised Selected Papers, Part I, volume 9573 of Lecture Notes in Computer Science, pp. 12–22. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-32149-3/_2

3. Библиотека Three.js. URL: https://threejs.org (дата обращения: 01.09.2022)
4. Суперкомпьютер «Ломоносов» – виртуальный тур. URL: http://lom1.virtual-hpc.parallel.ru/ (дата обращения: 01.09.2022).

Исследование механизма пакетирования суперкомпьютерных заданий в средстве моделирования Alea*

А.В. Баранов, Д.С. Ляховец

Межведомственный суперкомпьютерный центр Российской академии наук

Для проведения расчетов на суперкомпьютере пользователи оформляют т.н. задания, включающие в себя расчетные программы и входные данные. Специальные системы управления заданиями (СУЗ) ведут очереди заданий, выделяют и конфигурируют для них ресурсы, осуществляют запуск заданий и контролируют их выполнение. Для большинства заданий накладные расходы СУЗ являются пренебрежимо малыми, но существуют классы заданий с большими накладными расходами, для которых инициализация задания, включающая в том числе выделение и конфигурирование ресурсов, занимает длительное относительно последующей обработки время. Высокая доля времени инициализации является следствием либо длительного времени инициализации, либо малого времени обработки задания. Исследования показывают, что когда доля времени инициализации превышает 1% от времени выполнения задания, накладные расходы на инициализацию перестают быть пренебрежимо малыми, и их сокращение становится актуальной задачей.

Длительное время инициализации характерно для заданий, для которых необходимы подготовка и разворачивание виртуальной платформы, подготовка и загрузка сопроцессоров, копирование входных данных для случая территориально распределенной вычислительной системы и т.п. Малое время обработки заданий характерно для областей, в которых законченной единицей обработки является небольшой фрагмент работы. Такая ситуация возникает при решении задач фото- и видеообработки, таких как анализ движения, расчёт освещённости после добавления или удаления объектов, компьютерная анимация. Обработка одного кадра может занимать непродолжительное время. Схожая ситуация возникает в таких областях, как секвенирование генома, тестирование лекарств, ядерная физика и био-информатика.

На базе работы [1] проиллюстрируем численную оценку доли накладных расходов в Aneka. Aneka – это способ организации облачных вычислений в виде платформы-как-сервис, предоставляющий возможность построения распределённых приложений на базе Грид-систем или в облачной среде. В работе [1] приведены данные, что для решаемой задачи динамическое предоставление виртуальных машин с применением Aneka на базе облачного провайдера Azure от Microsoft в среднем занимает 150 секунд (усреднённое время для 10 запусков). При этом среднее время выполнения задания составляет 10 минут (600 секунд). Таким образом, накладные расходы на инициализацию составляют 150/(150+600) = 20%.

Задания одного пользователя с одинаковой инициализацией могут быть объединены в один пакет (мета-задание), для которого инициализация будет выполнена однократно, а затем будет произведена обработка всех заданий пакета. Для повышения эффективности использования ресурсов для заданий с высокой долей накладных расходов на инициализацию задания авторами был предложен способ пакетирования заданий [2]. Способ пакетирования был реализован для отечественной системы управления заданиями СУППЗ, применяемой в МСЦ РАН.

Для определения границ применимости системы пакетирования решено было использовать имитационное моделирование [3], для этого модель системы пакетирования была реализована в симуляторе Alea [4]. В Alea встроены различные алгоритмы планирования, такие как обычная очередь (FCFS), алгоритм обратного заполнения (Backfill) и другие, при этом существует возможность реализовать собственный алгоритм планирования заданий.

В работе [5] приводятся результаты вычислительных экспериментов, показавшие, что точность модели Alea незначительно меньше точности симулятора СУППЗ, но при этом Alea позволяет проводить длительные эксперименты (недели реальной обработки) за короткое время (десятки минут эксперимента).

* Работа была выполнена в МСЦ РАН в рамках государственного задания по теме FNEF-2022-0016.

С целью анализа системы пакетирования в разных условиях было разработано программное средство генерации входного потока, которое создает поток заданий, статистически схожий с реальным входным потоком суперкомпьютера. Генерируемыми параметрами задания являются: время выполнения, количество процессоров, время поступления задания, его тип. Каждый из параметров моделировался случайной величиной, вид распределения и его параметры были основаны на работе [6], в которой проведён статистический анализ журналов работы нескольких суперкомпьютеров за длительные периоды времени.

Для вычислительных экспериментов были сгенерированы различные входные потоки по 5000 заданий с общим временем обработки около 5 дней. Изменялось число типов заданий, время инициализации задания (измерялась средняя доля времени инициализации), интенсивность поступления заданий в очередь.

Сравнение производилось для алгоритмов обычной очереди (FCFS) в качестве наиболее плохого варианта, алгоритма обратного заполнения (Backfill) в качестве наиболее часто используемого в СУЗ алгоритма и реализованного алгоритма формирования пакетов (Packet). В качестве анализируемых показателей качества рассчитывались полная утилизация (процент вычислительных модулей, занятых инициализацией или выполнением заданий), полезная утилизация (процент занятых выполнением заданий вычислительных модулей, то есть модули во время инициализации задания считаются свободными), средняя длина очереди, среднее время ожидания заданий в очереди в секундах. Системы мониторинга СУЗ показывают полную утилизацию вычислительных модулей, так как не существует единого способа определения инициализации вычислительного модуля во время работы задания.

Реализованная авторами модель пакетирования в Alea является инструментальным средством, которое позволяет спрогнозировать эффект от внедрения алгоритма пакетирования для конкретного входного потока и определить наилучшее значение параметра алгоритма пакетирования (порог целесообразности формирования пакета), обеспечивающего требуемые показатели качества. В докладе предложен способ определения предельных характеристик входного потока заданий, при которых появляется статистически значимое улучшение показателей качества СУЗ, и результаты вычислительных экспериментов по определению границ применимости системы пакетирования заданий.

Литература

1. Tuli S., Sandhu R., and Buyya R. Shared data-aware dynamic resource provisioning and task scheduling for data intensive applications on hybrid clouds using Aneka // Future Gener. Comput. Syst. 106, 595–606 (2020). DOI: 10.1016/j.future.2020.01.038.

2. Lyakhovets D.S., Baranov A.V. Group Based Job Scheduling to Increase the High-Performance Computing Efficiency // Lobachevskii J Math 41, 2558–2565 (2020). DOI: 10.1134/S1995080220120264.

3. Баранов А., Ляховец Д. Симулятор системы управления суперкомпьютерными заданиями как научный сервис // Суперкомпьютерные дни в России: Труды международной конференции. 21-22 сентября 2020 г., Москва / Под редакцией Вл.В. Воеводина. – Москва: МАКС Пресс, 2020. – 172 с. С. 140-141. DOI: 10.29003/m1406.RussianSCDays-2020.

4. Klusáček D., Soysal M., Suter F. Alea – Complex Job Scheduling Simulator // Parallel Processing and Applied Mathematics. PPAM 2019. Lecture Notes in Computer Science, 217-229 (2020). DOI: 10.1007/978-3-030-43222-5_19.

5. Baranov A.V., Lyakhovets D.S. Accuracy Comparison of Various Supercomputer Job Management System Models // Lobachevskii J Math 42, 2510–2519 (2021). DOI: 10.1134/S199508022111007X.

6. Lublin U. and Feitelson G. The workload on parallel supercomputers: Modeling the characteristics of rigid job // J. Parallel Distrib. Comput. Arch. 63, 542–546 (2003). DOI: 10.1016/S0743-7315(03)00108-4.

Исследование основ для методов анализа пользовательской активности в суперкомпьютерном центре*

Д.А. Никитенко[1], М.В. Гомозов[1]

[1] Московский государственный университет имени М.В. Ломоносова

Для суперкомпьютерных центров и центров коллективного пользования актуальна задача анализа пользовательской активности. Для полноты понимания активности проекта, что может потребоваться, в частности, при рассмотрении вопросов расширения квот и продления исследовательского проекта, держателям систем может быть интересна активность пользователя не только в виде запусков задач, но активность, связанная с подготовкой к запускам. В данной работе рассматриваются потребности в таких оценках для разных классов пользователей – администраторов, экспертов, непосредственных пользователей и т.д., а также источники данных для анализа проектной и пользовательской активности в рамках суперкомпьютерного центра. Предлагается подход к анализу активности пользователей путем использования инструментария ручного и автоматического анализа и его программная реализация в виде модуля системы поддержки функционирования СКЦ Octoshell.

Ключевые слова: центр коллективного пользования, суперкомпьютерный центр, контроль использования вычислительных ресурсов, активность исследовательских проектов.

1. Введение

Суперкомпьютерный центр — это большая комплексная система, в рамках которой могут работать тысячи пользователей. Помимо непосредственных пользователей, можно выделить и другие роли, такие как администраторы, руководители проектов, эксперты и т.д. В рамках их отличающихся рабочих процессов могут возникать потребности в анализе разного рода данных, связанных как с непосредственно использованием, так и с доступом к вычислительным ресурсам, которые могут варьироваться в зависимости от того, какую роль в системе исполняет пользователь [1].

Описываемое в данной работе исследование проводится и апробируется в рамках СКЦ МГУ, в рамках которого ведут работы более ста научных проектов, объединяющих несколько сотен пользователей [2]. Для поддержки функционирования основных рабочих процессов суперкомпьютерного центра была разработана система Octoshell [3-5]. Она реализует проектный подход к работе пользователей с вычислительными ресурсами [6, 7]. Каждый пользователь имеет одну или несколько учетных записей, привязанных к определенным проектам. В свою очередь проект определяет деятельность в рамках суперкомпьютерного центра. Руководителем проекта, координирующего работу, запрашиваются и выделяются ресурсы определенных вычислительных систем, доступные для его участников. Вычислительных систем в суперкомпьютерном центре может быть несколько, каждый из них может иметь несколько разделов и определенные квоты использования его ресурсов.

Для регулирования использования ресурсов, поддержки актуальной информации о пользователях и их потребностях, а также понимания результатов проектной деятельности в системе Octoshell предусмотрена ежегодная процедура перерегистрации. В ее рамках каждый проект обязан предоставить отчет о проделанной работе. Отчеты оцениваются экспертами - специальными пользователями, способными достойно оценить проектную деятельность в определенном направлении исследований. Отдельно стоит выделить роли администраторов, имеющих полный доступ ко всем параметрам и данным системы.

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта №20-07-00864.

Ранее доступная реализация статистического раздела Octoshell предоставляет возможность анализировать распределение запусков задач по машинам, разделам и статусам завершения [8]. Участники проектов имеют доступ к индивидуальной статистике, руководители к статистике своего проекта, а администраторы - к статистике всех проектов системы. Статистика представлена с помощью сводных таблиц и круговых распределительных диаграмм [9, 10].

Целью данного проекта является расширение возможностей и функционала статистического раздела системы Octoshell с помощью анализа пользовательской активности в целом.

2. Предлагаемый подход

По результатам исследования сформирован перечень потребностей пользователей суперкомпьютерного центра в статистическом анализе и подробно разобранный набор источников информации. Предлагаемый подход к удовлетворению описанных потребностей состоит в построении для каждой из ролей: участника проекта, руководителя проекта, эксперта и администратора - инструментария, на основе которого можно проводить ручной и автоматический анализ данных, полученных из сущностей 3 описанных доступных источников информации: планировщиков систем, вычислительной системы и системы Octoshell (рисунок 1).

Инструментарий для каждой отдельной роли пользователя определяет перечень инструментов ручного и автоматического анализа, основанных на базовом наборе инструментов описательной и аналитической статистики, а также методы фильтрации данных и отображение некоторой дополнительной специфичной для рассматриваемого объекта информации.

Рис. 1. Схема предлагаемого подхода к анализу пользовательской активности.

3. Заключение

В статье рассмотрены основы для реализации комплексного подхода к пользовательской активности в рамках суперкомпьютерного центра – выделены ключевые пользовательские роли и источники данных. На основе рассмотренного, ведется программная реализация, апробация которой планируется к завершению осенью 2022г., что будет описано в отдельной статье.

Работа выполнена с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова и при финансовой поддержке РФФИ.

Литература

1. Role-dependent resource utilization analysis for large hpc centers / D. Nikitenko, P. Shvets, V. Voevodin, S. Zhumatiy // Parallel Computational Technologies. — Vol. 910 of Communications in Computer and Information Science (CCIS). — SPRINGER, 2018. — P. 47–61.
2. Supercomputer Lomonosov-2: Large scale, deep monitoring and fine analytics for the user community / V. V. Voevodin, A. S. Antonov, D. A. Nikitenko et al. // Supercomputing Frontiers and Innovations. — 2019. — Vol. 6, no. 2. — P. 4–11.
3. Никитенко Д. А., Воеводин В. В., Жуматий С. А. Octoshell: система для администрирования больших суперкомпьютерных комплексов // Вестник Южно-Уральского государственного университета. Серия "Вычислительная математика и информатика". — 2016. — Т. 5, No 3. — С. 76–95.
4. Nikitenko D.A., Zhumatiy S.A., Paokin A.V., Voevodin V.V., Voevodin Vl V., Octoshell: Large supercomputer complex administration system // Communications in Computer and Information Science. - Springer International Publishing, 2019. - С. 19-33.
5. Исходный код Octoshell v2 URL: https://github.com/octoshell/octoshell-v2 (дата обращения: 30.06.2022).
6. Paokin A. V., Nikitenko D. A. Unified approach for provision of supercomputer center resources // Bulletin of South Ural State University: Computational Mathematics and Software Engineering. — 2022. — Vol. 11, no. 1. — P. 5–14.
7. Паокин А. В., Никитенко Д. А., Жуматий С. А. Исследование поддержки вариативности рабочих процессов в системе octoshell // Параллельные вычислительные технологии – XV международная конференция, ПаВТ'2021, г. Волгоград, 30 марта – 1 апреля 2021 г. Короткие статьи и описания плакатов. — Челябинск: Челябинск, 2021. — С. 233–243.
8. Nikitenko D.A., Zhumatiy S.A., Voevodin V.V. Deep Analysis of Job State Statistics on Lomonosov-2 Supercomputer // Supercomputing Frontiers and Innovations. - 2018. - С. 4-10.
9. Kapridov A. A., Nikitenko D. A. The Top50 performance ranking statistical data processing and visualization methods // Параллельные вычислительные технологии - XIII международная конференция, ПаВТ'2019, г. Калининград, 2-4 апреля 2019 г.
10. Nikitenko D., Zhumatiy S., Shvets P. Making large-scale systems observable — another inescapable step towards exascale // Supercomputing Frontiers and Innovations. — 2016. — Vol. 3, no. 2. — P. 72–79.

Исследование эффективности механизма прогнозирования и корректировки времени выполнения вычислительных заданий в территориально распределенной сети суперкомпьютерных центров*

А.И. Тихомиров, А.В. Баранов

Межведомственный суперкомпьютерный центр Российской академии наук – филиал Федерального государственного учреждения «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук»

Эффективность использования высокопроизводительной вычислительной установки (ВУ) во многом определяется возможностью составить точное расписание запусков пользовательских заданий, не допускающее простоя ресурсов ВУ. Задачу планирования расписания выполняет система управления заданиями (СУЗ). Для планирования СУЗ использует ресурсные требования заданий, которые указывают пользователи. СУЗ строго следует соблюдению требований, указанных пользователем, и неточность, допущенная пользователем при оценке ресурсных требований задания, может привести к неэффективному планированию ресурсов ВУ.

На протяжении последних нескольких лет в мире и в России ведутся исследования [1-3] возможностей автоматической корректировки (уточнения) характеристик задания, заказанных пользователем, а также влияния корректировки на качество планирования заданий. При этом одной из характеристик задания, требующей корректировки, является заказанное время выполнения. Наблюдение [1] за потоком пользовательских заданий показало, что почти всегда заказанное пользователем время превышает фактическое время выполнения. В современных СУЗ появились механизмы, позволяющие корректировать время выполнения задания в пределах требований, заданных пользователем. Такая корректировка призвана повысить эффективность планирования. В настоящей работе предлагается исследовать эффективность такой корректировки для случая территориально распределённой сети ВУ.

Заказанное время выполнения определяет время, в течение которого для задания будут выделены ресурсы ВУ. Фактическое время выполнения задания может отличаться от заказанного как в большую, так и меньшую сторону. Если фактическое время выполнения меньше заказанного, то задание выполнится, но ресурсы ВУ могут простаивать до завершения заказанного времени. Если фактическое время больше заказанного, то задание будет снято СУЗ с ресурсов до завершения расчетов. Зная принцип работы СУЗ, пользователи осознано почти всегда заказывают время выполнения со значительным (в среднем до 2,3 раз) запасом.

Обзор актуальных исследований [1-3] показывает, что во всех работах исследовалась одна из следующих моделей.

1. Модель, в которой заказанное время выполнения корректировалось путем подмены его на прогнозируемое [1]. Характеристики входного потока задания до распределения на ресурсы ВУ корректировались с помощью дополнительного модуля СУЗ, который на основе статистики ранее выполненных заданий прогнозировал время выполнения задания и подменял в требованиях задания заказанное время на прогнозируемое. Неточность прогноза приводила к тому, что часть заданий была снята СУЗ с выполнения до их завершения, потому что их прогнозируемое время оказывалось меньше фактического. При этом СУЗ работала с такими заданиями, как с обычными заданиями, и снимала с выполнения все задания, которые выполнялись дольше заказанного времени.
2. Модель с одной очередью заданий [2]. Весь поток пользовательских заданий обрабатывался СУЗ одной ВУ.
3. Модель с синтетическим потоком заданий [3]. Характеристики входного потока пользовательских заданий задавались с помощью классических математических распределений или были получены путем какого-либо преобразования реального потока, например, ускорения.

* Работа была выполнена в МСЦ РАН в рамках государственного задания по теме FNEF-2022-0014

В то же время современные СУЗ, такие как PBS Pro и применяемая в МСЦ РАН СУППЗ, обладают механизмом корректировки требований задания. Этот механизм позволяет для каждого задания указать дополнительно скорректированные ресурсные требования, в том числе время выполнения. Скорректированное время выполнения должно быть строго меньше времени заказанного пользователем. Корректировку требований выполняет дополнительный модуль СУЗ – модуль прогнозирования. СУЗ использует скорректированное время для составления расписания запусков заданий. При этом в случае, если скорректированное время окажется неточным и меньше фактического, СУЗ не снимет задание с выполнения, пока не будет исчерпано время, заказанное пользователем. Использование этого механизма более безопасно для задания, в отличие от подхода с заменой [1].

Особый научный интерес представляет задача по исследованию эффективности использования механизма корректировки для случая распределённой сети ВУ. В настоящей работе авторы исследуют модель территориально распределённой сети, в которой:

- используется реальный входной поток заданий;
- прогнозируемое время выполнения задания указывается для задания дополнительным требованием;
- для каждого задания скорректированное время задается с помощью модуля прогнозирования, который на основе статистики обработки заданий может с заданной точностью прогнозировать время выполнения.

Целью работы является оценка порога точности прогноза, при котором будет достигнут положительный эффект от использования механизма корректирования требований задания в распределённой сети ВУ. Под положительным эффектом понимается статистически значимые улучшения показателей качества планирования заданий в СУЗ, таких как загрузка ресурсов, время нахождения задания в очереди, полное время пребывания задания в системе и других.

Для проведения исследования авторами подготовлен макет распределённой сети ВУ. Сеть объединяет ВУ разной производительности, каждая ВУ – это модель-симулятор одного из разделов суперкомпьютера МВС-10П ОП, установленного в МСЦ РАН. Модельный поток заданий формируется путем объединения реальных потоков заданий, поступивших за определенный период на соответствующие разделы МВС-10П ОП. Модельный поток заданий помещается в единую глобальную очередь, откуда в дальнейшем задания распределяются в ВУ из состава сети. Распределение заданий глобальной очереди осуществляет коллектив равноправных диспетчеров ВУ путем проведения аукциона за каждое задание глобальной очереди. Аукционную цену задания – ставку каждого диспетчера – определяет прогнозированное время выполнения задания для соответствующей диспетчеру ВУ. Задание помещается в ту ВУ, в которой оно выполнится быстрее всего с учетом корректировки времени выполнения в соответствии с прогнозом. Построенная модель позволит оценить влияние точности прогноза заданий на показатели эффективности планировщика СУЗ.

Литература

1. Shabanov B., Baranov A., Telegin P., Tikhomirov A. Influence of Execution Time Forecast Accuracy on the Efficiency of Scheduling Jobs in a Distributed Network of Supercomputers. Lecture Notes in Computer Science, vol. 12942, pp. 338-347, 2021. DOI: 10.1007/978-3-030-86359-3_25.
2. Klusáček D., Chlumský V. Evaluating the Impact of Soft Walltimes on Job Scheduling Performance. Lecture Notes in Computer Science, vol. 11332, 2019. DOI: 10.1007/978-3-030-10632-4_2.
3. Klusáček D., Tóth V., Podolníková G. Complex job scheduling simulations with Alea 4. Ninth EAI International Conference on Simulation Tools and Techniques (SimuTools 2016), pp. 124–129, ACM, 2016.

Разработка MPI-реализации генератора многопучковых конфигураций электромагнитного поля*

Е.А. Панова[1], И.Б. Мееров[1], Е.С. Ефименко[2]

[1] Нижегородский государственный университет им. Н. И. Лобачевского
[2] Институт прикладной физики РАН

В последнее время всё большее внимание привлекает область моделирования квантово-электродинамических (КЭД) каскадов в условиях сверхсильных электромагнитных полей [1]. Получение сверхмощных полей на перспективных лазерных системах, включая проект XCELS [2], подразумевает использование многопучковых лазерных систем. Ранее нами в работе [3] был представлен разрабатываемый в рамках программного комплекса PICADOR [4] модуль для задания многопучковых конфигураций, который позволяет задавать любое количество независимых электромагнитных импульсов, распространяющихся в произвольных направлениях и фокусирующихся в заданной пользователем точке расчетной области. Предложенный нами метод подразумевает использование вспомогательной сетки для расчета методом FDTD [5] электромагнитного поля одного пучка, распространяющегося параллельно оси координат, а затем его отображение на границу основной расчетной области с учетом угла поворота, поляризации, сдвига и других параметров. Часто пучки в расчете имеют схожие характеристики, что позволяет рассчитывать поля нескольких пучков на одной вспомогательной сетке. Пример работы генератора многопучковых конфигураций представлен на рис. 1.

Одной из основных задач являлась реализация параллельной версии многопучкового генератора на распределенной памяти, которая представляла некоторую сложность, обусловленную тем, что вспомогательная расчетная область находится под некоторым углом к основной (рис. 2). В этом случае возникают вопросы, связанные с тем, как оптимально разделить вспомогательную сетку на домены и распределить их между процессами; в каком виде хранить и передавать данные; как определить, какие процессы должны взаимодействовать друг с другом. Поскольку обновление полей занимает большую часть времени, для балансировки нагрузки было принято решение делить вспомогательную сетку, как и основную, на равные по размеру домены. В рамках текущей реализации на каждом процессе, хранящем вспомогательную сетку, с помощью линейной интерполяции вычисляются значения в граничных узлах основной сетки, упаковываются в подходящую структуру данных и передаются нужным процессам основной сетки. Для того, чтобы определить, с какими доменами вспомогательной сетки нужно взаимодействовать отдельно взятому домену основной сетки, перед началом моделирования один раз происходит обмен информацией каждого процесса с каждым.

Для определения эффективности полученной параллельной реализации на распределенной памяти в качестве тестовой задачи рассматривалась близкая к реальным задачам конфигурация, состоящая из двух одинаковых по форме пучков, при этом один пучок распространялся из угла трехмерной коробки к центру под углом $3\pi/4$ рад к оси z, а другой – параллельно оси z. Схожий двумерный случай изображен на рис. 1 и 2. Для оптимизации расчета использовалась одна общая вспомогательная сетка. Производилось несколько итераций обновления значений электромагнитного поля. Размер основной сетки был равен 512×512×512, автоматически вычисленный размер вспомогательной сетки 381×1304×1304.

Запуски производились на нескольких вычислительных узлах суперкомпьютера МВС-10П Объединенного суперкомпьютерного центра РАН (2x Intel Xeon Gold 6248R, 3.0 ГГц, 48 ядер, 192 ГБ оперативной памяти). Рассмотрено два случая: несколько (до 8) процессов MPI в рамках одного вычислительного узла и по одному процессу на каждый узел. Деление на домены на основной и вспомогательной сетках происходило вдоль оси x пополам для 2 процессов; вдоль осей

* Исследование выполнено при финансовой поддержке РФФИ и Госкорпорации «Росатом» в рамках научного проекта №20-21-00095.

x и y для 4 процессов; вдоль осей x, y, z для 8 процессов. Полученная в результате экспериментов эффективность сильной масштабируемости представлена на рис. 3. Можно видеть, что в рамках одного узла масштабируемость падает с ростом числа процессов, в то время как при запуске на нескольких узлах она остается высокой. Также можно видеть, что некоторые разбиения были более удачными для одного пучка и менее удачными для другого, однако в совместном расчете они компенсировали друг друга. Эффективность больше 1 достигается за счет лучшей локальности данных при увеличении числа процессов. В целом затраты на формирование пакета данных и MPI-коммуникацию одного пучка составили около 10% от всего времени расчета для запусков на общей памяти и 20% для запусков на распределенной памяти.

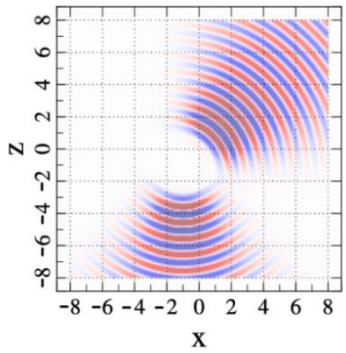


Рис. 1. Пример работы многопучкового генератора: электромагнитное поле в сечении $y = 0$. Генерируются два ортогональных оси y импульса: параллельно оси z и под углом $3\pi/4$ рад к оси z. Фокус обоих импульсов расположен в точке (-1, 0, -1). Длина волны 1 см; разрешение 12 точек на длину волны.

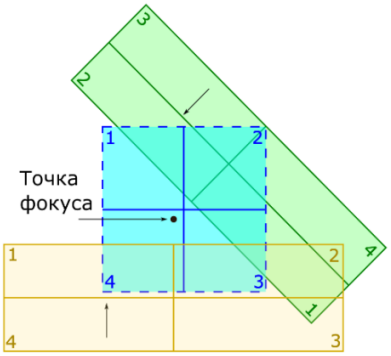


Рис. 2. Пример взаимного расположения основной (синяя) и вспомогательных (зеленая, желтая) сеток. От вспомогательной сетки к основной передаются значения полей на границе основной сетки (пунктирная линия). Можно видеть, что домен 2 основной сетки должен взаимодействовать с четырьмя, домены 3 и 4 с двумя, а домен 1 с одним доменом вспомогательной сетки. Если параметры импульсов на вспомогательных сетках совпадают, то в памяти хранится только одна вспомогательная сетка.

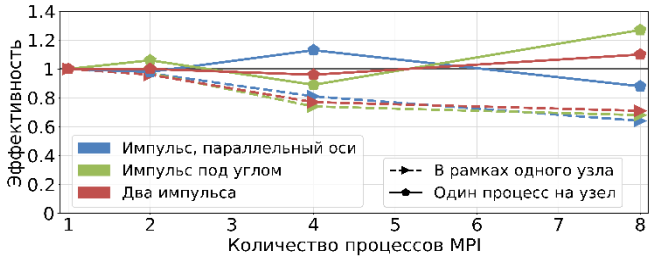


Рис. 3. Эффективность сильной масштабируемости версии MPI на одном вычислительном узле и на нескольких вычислительных узлах (один процесс на узел) для тестовой конфигурации: расчеты для каждого из 2-х импульсов по отдельности и вместе.

Литература

1. Volokitin V. et al. Optimized routines for event generators in QED-PIC codes // Journal of Physics: Conference Series. – IOP Publishing, 2020. – Т. 1640. – №. 1. – С. 012015.
2. XCELS: https://xcels.ipfran.ru/.
3. Панова Е. А. и др. Разработка генератора многопучковых конфигураций электромагнитного поля // Математическое моделирование и суперкомпьютерные технологии. Труды XXI Международной конференции (Н. Новгород, 22–26 ноября 2021 г.) / Под ред. проф. Д.В. Баландина – Нижний Новгород: Изд-во ННГУ, 2021. – С. 257-261.
4. Bastrakov S. et al. Particle-in-cell plasma simulation on heterogeneous cluster systems // Journal of Computational Science. – 2012. – Т. 3. – №. 6. – С. 474-479.
5. Taflove A., Hagness S. C. Computational electrodynamics: the finite-difference time-domain method. – Artech house, 2005.

Реализация алгоритмов раскраски графов с использованием SYCL и KOKKOS*

А.А. Курникова, А.Ю. Пирова, В.Д. Волокитин, И.Б. Мееров

Нижегородский государственный университет им. Н.И. Лобачевского

Разнообразие вычислительного оборудования создает большие возможности для численного моделирования при решении задач в науке и индустрии. Эффективное использование разнородных устройств (CPU, GPU, FPGA) часто невозможно без применения языков программирования, специфичных для конкретных устройств, что приводит к необходимости разработки и поддержки нескольких кодов, решающих одну и ту же задачу. Для преодоления этой проблемы развиваются подходы к гетерогенному программированию, в основе которых лежит идея единого языка (библиотеки) для программирования произвольных вычислительных устройств. В данной работе мы демонстрируем приемы оптимизации кода при решении задачи о раскраске графа с использованием двух перспективных моделей программирования – KOKKOS и SYCL, анализируем производительность в сравнении с реализацией на C++/OpenMP.

Алгоритмы раскраски графа позволяют находить точное или приближенное значение хроматического числа произвольного графа и соответствующую этому значению раскраску вершин [1]. Широко известен жадный алгоритм раскраски с различными оптимизациями по упорядочиванию вершин [2]. Распространены параллельные алгоритмы раскраски Джонса-Плассмана [3], Чаталюрека (Çatalyürek) [4], Боумана [4] и другие. В данной работе мы рассматриваем один из наиболее эффективных алгоритмов – алгоритм Чаталюрека – и обсуждаем способы ускорения расчетов при решении задачи раскраски графа без потери качества раскраски. Мы рассматриваем стандартную версию алгоритма Чаталюрека, реализованную с использованием различных моделей программирования: OpenMP (для центральных процессоров), KOKKOS и SYCL (для центральных и графических процессоров). Основная цель работы – сравнить производительность кода, разработанного с использованием различных моделей программирования, и проанализировать влияние различных методик оптимизации кода на время работы.

В начале была разработана параллельная реализация алгоритма Чаталюрека с использованием OpenMP (*default omp*). Для этого множество вершин графа разделялось поровну между потоками, каждый поток выполнял раскраску подграфа. По окончании производилось объединение результатов и разрешение конфликтов раскраски. Далее этот код был адаптирован для использования средств распараллеливания SYCL (*default dpc*) и KOKKOS (*default kokkos*). Для этого потребовалось переработать механизмы выделения/освобождения памяти, подготовить функции-ядра, выполняющие раскраску подграфов и разрешение конфликтов. Кроме этого, была исследована реализация одного из алгоритмов раскраски из стандартного комплекта поставки библиотеки KOKKOS (*kokkos lib*). Эксперименты, проведенные на процессорах Intel Xeon Gold (32 ядра) показали, что все рассматриваемые реализации приводят к одинаковому качеству раскраски (числу цветов). При этом сравнение времени работы показало, что версии на OpenMP, SYCL и KOKKOS работают практически за одинаковое время, но уступают около 60% реализации из комплекта поставки библиотеки KOKKOS. Было обнаружено, что данное отставание полностью устраняется путем перехода к использованию битовых операций при разрешении конфликтов раскраски (*bit omp*, *bit kokkos*, *bit dpc*). Это подразумевает замену вставки в массив конфликтных цветов на шаге разрешения конфликтов на битовый сдвиг по номеру добавляемого цвета. Таким образом, было установлено, что использование моделей SYCL и KOKKOS не приводит к дополнительным накладным расходам в этой задаче, при этом важным преимуществом является возможность запуска кода не только на центральных, но и на графических процессорах. Результаты представлены на рисунке 1.

Далее мы провели эксперименты на дискретных GPU Intel Iris XE Max. Было обнаружено, что качество раскраски ожидаемо не изменилось. При этом сравнение времени работы показало, что оптимизированные с использованием битовых операций реализации на KOKKOS и SYCL

работают за одинаковое время, но немного уступают встроенной реализации из комплекта поставки KOKKOS. Было установлено, что дополнительное ускорение может быть получено за счет кеширования данных в локальной памяти GPU (*bitcache kokkos*, *bitcache dpc*). Таким образом удалось дополнительно ускорить вычисления на 22% и обогнать стандартную реализацию из комплекта KOKKOS. Важно отметить, что такое кеширование приводит к ускорению и на CPU (рисунок 2). Было обнаружено, что построенная с использованием кеширования реализация является в этом смысле универсальной и может быть использована по умолчанию и на центральных, и на графических процессорах Intel в силу переносимости реализаций на KOKKOS и SYCL.

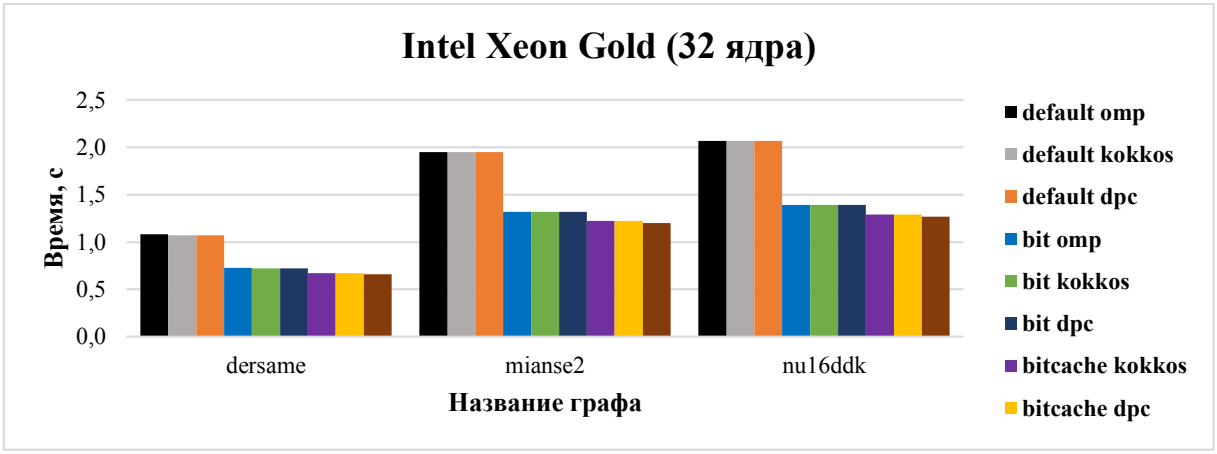


Рисунок 1. Результаты запусков на центральном процессоре

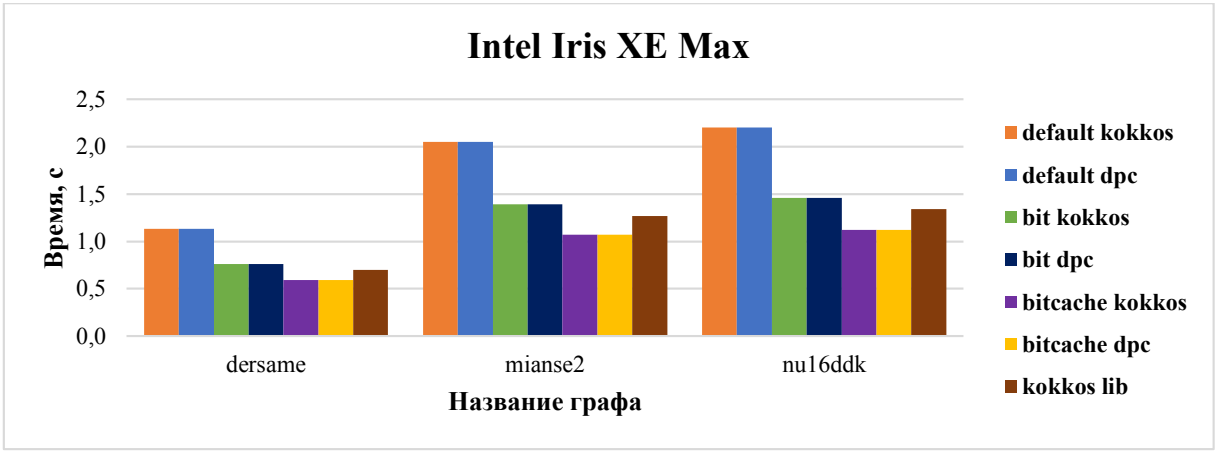


Рисунок 2. Результаты запусков на графическом процессоре

Литература

1. Karp R.M. Reducibility among combinatorial problems // Complexity of computer computations. Springer, Boston, MA, 1972. P. 85-103. DOI: 10.1007/978-1-4684-2001-2_9
2. Gross J.L. et al. Graph Theory and Its Applications (3rd ed.). Chapman and Hall/CRC. 2018. DOI: 10.1201/9780429425134
3. Jones M.T., Plassmann P.E. A parallel graph coloring heuristic // SIAM Journal on Scientific Computing. 1993. Vol. 14, No. 3. P. 654-669. DOI: 10.1137/0914041
4. Boman E.G. et al. A Scalable Parallel Graph Coloring Algorithm for Distributed Memory Computers // Cunha J.C., Medeiros P.D. (eds) Euro-Par 2005 Parallel Processing. Vol. 3648. Springer, Berlin, Heidelberg, 2005. DOI: 10.1007/11549468_29M

* Работа выполнена при поддержке Министерства науки и высшего образования РФ, проект № 0729-2020-0055

Реализация на СуперЭВМ алгоритма решения обратных задач на основе операторов чувствительности в рамках платформы обратного моделирования IMDAF [*]

А.В. Пененко, Е.В. Русин

Институт вычислительной математики и математической геофизики СО РАН

Платформа обратного моделирования с усвоением данных IMDAF (Inverse Modeling and Data Assimilation Framework) - это оригинальный программный комплекс решения задач обратного моделирования для дифференциальных уравнений и различных типов данных измерений. Основными классами решаемых задач являются: прямые, обратные, усвоения данных и оценки чувствительности. Основными областями применения платформы являются оценка и прогнозирование качества воздуха [1] и исследование биологических систем [2].

В рамках платформы реализованы три основные группы «решателей» обратных задач, которые требуют различной сложности реализации вспомогательных процедур, необходимых для работы в данной области приложений. Для использования первой группы необходимо реализовать только процедуру решения прямой задачи. После этого используются стандартные реализации алгоритмов минимизации «без производных» (derivative-free) [3]. Следующая группа «решателей» требует реализации процедуры решения как прямой, так и сопряженной задач для оценки градиента функционала невязки измеренных и смоделированных значений. Для решения задач оптимизации используются стандартные реализации градиентных алгоритмов [3]. Третья группа «решателей» на основе операторов чувствительности обратных задач относится к уникальным особенностям IMDAF. При этом обратная задача сводится к семейству квазилинейных операторных уравнений с операторами чувствительности. Последние формируются с помощью решения ансамблей сопряженных уравнений и вычисления соответствующих функций чувствительности, задаваемых набором функций проектирования данных измерений. Для применения этих алгоритмов требуется дополнительно реализовать процедуры решения ансамбля сопряженных уравнений. Детальное описание алгоритмов можно найти в [1,2]. Численные эксперименты показывают более высокую относительную эффективность «решателей» из третьей группы [2,3].

Ранее распараллеливание в IMDAF было реализована на основе стандарта OpenMP для машин с общей памятью. Результаты оценки эффективности такого распараллеливания представлены в [4]. Однако необходимость работать с гетерогенными данными измерений высокой детализации (контактные измерения и спутниковые снимки полей концентраций) [1] и использовать более реалистичные модели химии атмосферы требует привлечения более существенных вычислительных ресурсов, чем одна машина с общей памятью.

В данной работе мы рассматриваем реализацию IMDAF на основе MPI. Прежде всего, с помощью стандарта MPI была реализована процедура вычисления ансамбля решений сопряженных уравнений и оператора чувствительности на его основе. Различные сопряженные уравнения из ансамбля являются условно независимыми и при этом для их вычисления используется общий набор коэффициентов численных схем. Это открывает возможности для естественного распараллеливания: параллельно по ансамблю решаются сопряженные уравнения, массив коэффициентов численных схем вычисляется параллельно согласно схе-

[*]Работа поддержана грантом № 075-15-2020-787 в форме субсидии на крупный научный проект Министерства науки и высшего образования Российской Федерации (проект «Основы, методы и технологии цифрового мониторинга и прогнозирования экологической обстановки на Байкальской природной территории»).

ме расщепления многомерного уравнения адвекции-диффузии-реакции. Для решения обратной задачи обычно требуется вычислить оператор чувствительности несколько сотен раз. Количество итераций алгоритма определяется в зависимости от задачи и требуемой точности приближенного решения.

В Таблице приводятся предварительные оценки эффективности распараллеливания вычисления оператора чувствительности при решении обратной задачи идентификации источников загрязнений по данным мониторинга для трехмерной модели переноса и трансформации примеси в атмосфере в тестовом сценарии для Байкальского региона (аналогично [1]). Расчеты проводились на двух «блейд»-серверах, установленных в кластер НКС-1П ССКЦ СО РАН. На серверах установлены двойные процессоры Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz с 24 ядрами на каждом процессоре. Объемы памяти 376 GB и 754 GB соответственно. В будущем планируется измерить масштабируемость на большем числе узлов.

Таблица 1. Сравнение времени вычисления оператора чувствительности обратной задачи идентификации источников загрязнений для модели переноса и трансформации примеси в атмосфере в модельном сценарии для Байкальского региона на разных конфигурациях узлов

Узлов	Потоков на узле	Время, с	Ускорение	Эффективность, %
1	2	13694	1	100
1	12	2964	4.6	77
1	24	1769	7.7	65
1	48	1395	9.8	41
2	24	1183	11.6	48
2	48	912	15	31

Литература

1. Penenko A., Penenko V., Tsvetova E., Gochakov A., Pyanova E. and Konopleva V. Sensitivity operator framework for analyzing heterogeneous air quality monitoring systems. // Atmosphere, 12(12):1697, dec 2021. DOI: 10.3390/atmos12121697.

2. Penenko A., Nikolaev S., Golushko S., Romashenko A. and Kirilova I. Numerical algorithms for diffusion coefficient identification in problems of tissue engineering. // Mathematical Biology and Bioinformatics, 11(2):426–444, 2016. (In Russian). DOI: 10.17537/2016.11.426.

3. Penenko A., Konopleva V., and Bobrovskikh A. Numerical comparison of the adjoint problem-based and derivative-free algorithms on the coefficient identification problem for a production-loss model. // 2021 17th International Asian School-Seminar "Optimization Problems of Complex Systems (OPCS). IEEE, sep 2021. DOI: 10.1109/opcs53376.2021.9588680.

4. Penenko A. and Gochakov A. Parallel speedup analysis of an adjoint ensemble-based source identification algorithm //Journal of Physics: Conference Series, 1715:012072–1–012072–6, 2021. DOI: 10.1088/1742-6596/1715/1/012072.

Реализация схемы переноса пассивной примеси на графических ускорителях при использовании половинной точности.

Е.М. Гащук[1, 2], Е.В. Мортиков[3, 2], А.В. Дебольский[3, 2]

Московский государственный университет им. М.В.Ломоносова, механико-математический факультет, Москва, 119234, Российская Федерация[1]
Московский Центр фундаментальной и прикладной математики, Москва, 119992, Российская Федерация[2]
Московский государственный университет им. М.В. Ломоносова, Научно-исследовательский вычислительный центр, Москва, 119234, Российская Федерация[3]

Численное моделирование геофизических процессов на сегодняшний день остается одной из самых сложных задач, на решение которых требуются значительные вычислительные ресурсы. Использование пониженной точности является одним из возможных способов повышения эффективности вычислений, что становится все более распространенным в последнее время [1]. При этом существование множества алгоритмов [2], позволяющих компенсировать ошибки округления, делает возможным использование пониженной точности при решении все большего спектра вычислительных задач.

В данной работе рассматривается реализация алгоритма переноса примеси при использовании половинной точности (fp16) на GPU в модели прямого численного моделирования (DNS - direct numerical simulation), развиваемой в НИВЦ МГУ и ИВМ РАН [3], на основе системы уравнений Навье–Стокса. Программная реализация основана на использовании библиотеки MPI и технологии CUDA.

Уравнение переноса в безразмерном виде, описывающее изменение поля концентрации примеси:

$$\frac{\partial C_k}{\partial t} + \frac{\partial u_i C_k}{\partial x_i} = \frac{1}{ReSc_k}\frac{\partial^2 C_k}{\partial x_i \partial x_i} - T_k^{-1} C_k,$$

где $\mathbf{x} = (x_1, x_2, x_3)^T$ – пространственные координаты, $\mathbf{u}(\mathbf{x}, t) = (u_1, u_2, u_3)^T$ – вектор скорости, $C_k(\mathbf{x}, t)$ – концентрация k-ой примеси, Sc_k – число Шмидта для k ой примеси, Re – число Рейнольдса, T_k – безразмерная величина, определяющая характерное время жизни k-ой примеси.

Для аппроксимации пространственных производных используются разностные схемы второго и четвертого порядка [4] на прямоугольных сетках с разнесенным расположением переменных в ячейке. Для конечно-разностной формы системы уравнений выполняются законы сохранения импульса и энергии.

Проведены вычислительные эксперименты по воспроизведению турбулентного течения Куэтта при нейтральной стратификации. Получено, что ускорение вычислений на GPU при использовании половинной точности на задаче размерности $96 \times 64 \times 64 (Re = 5200)$ достигает 30%, на задаче размерности $192 \times 128 \times 128 (Re = 20000)$ достигает 50%.

Проведено сравнение результатов численных экспериментов. Показано, что результаты численных экспериментов в fp16 (первые и вторые моменты гидродинамических полей: дисперсия, поток концентрации скаляра и вертикальное распределение концентрации примеси) значительно не отличаются от результатов численных экспериментов в одинарной точности (fp32). Ниже представлены результаты численных экспериментов в fp16 для задачи размерности $192 \times 128 \times 128$. На рисунке 1 изображен средний профиль концентрации скаляра (ошибка fp16 относительно fp32 равняется 0.02%[1]), полный поток скаляра, который для течения Куэтта не зависист от расстояния до стенок, и его отдельные компоненты. На рисунке 2 показано распределение турбулентного потока скаляра при осреднении по времени и в продольном к потоку направлении, где неоднородность связана с воспроизводимой в расчетах с fp16 крупномасштабной вторичной к среднему потоку циркуляцией (погрешность составляет 0.08%[1]).

[1] $\frac{\|res_{fp32} - res_{fp16}\|_{L_2}}{\|res_{fp32}\|_{L_2}} \cdot 100\%, \ \|\mathbf{x} = (x_1, \ldots, x_n)^T\|_{L_2} = \sqrt{x_1^2 + \ldots + x_n^2}.$

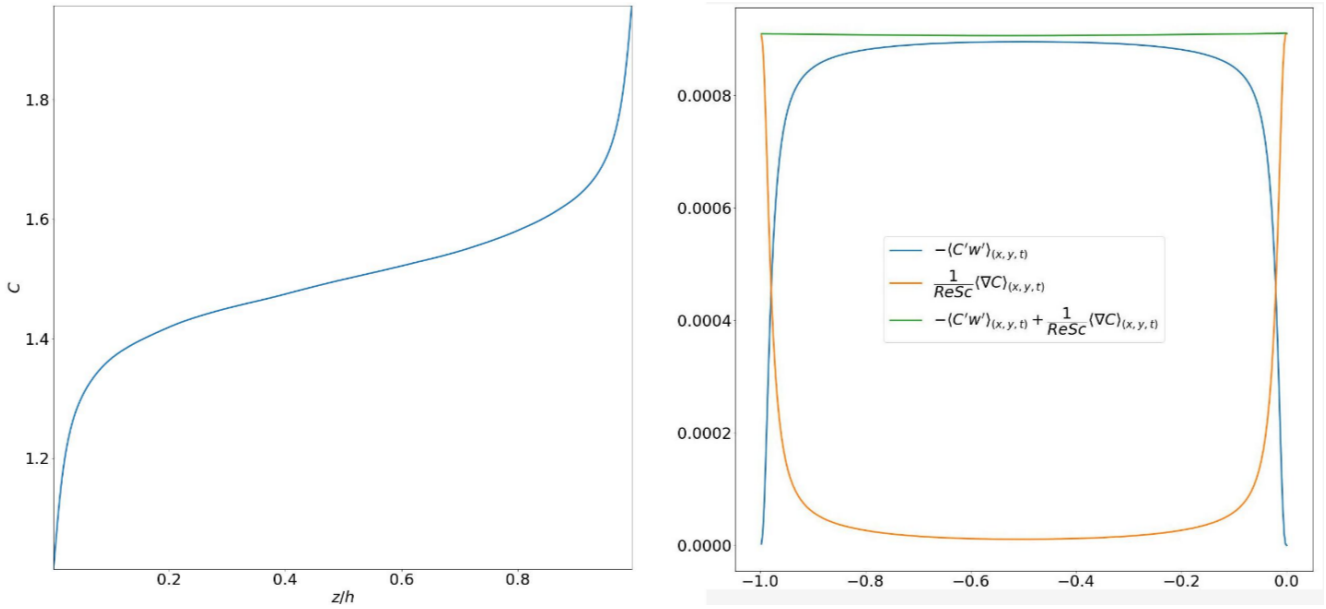


Рис. 1. Средний профиль примеси *(слева)* и средний полный поток скаляра *(справа)* и его компоненты: трубулентная и вязкая часть.

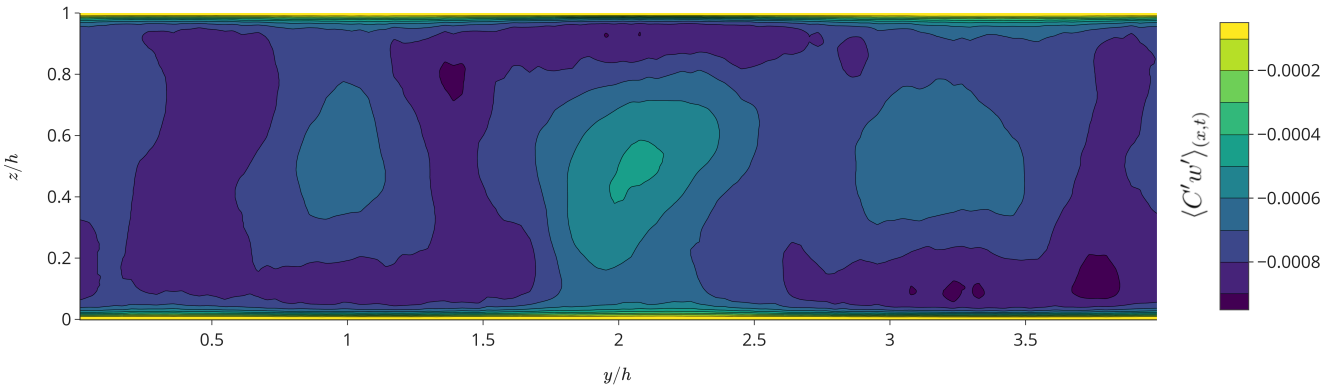


Рис. 2. Осредненный по (x, t) вертикальный турбулентный поток примеси.

Работа выполнена с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова.

Литература

1. Chantry M., Thornes T., Palmer T. and Düben P., 2019: Scale-selective precision for weather and climate forecasting Mon. Wea. Rev., 147, 645–655, https://doi.org/10.1175/MWR-D-18-0308.1.

2. Croci M., Fasi M., Higham N.J., Mary T. and Mikaitis M. 2022: Stochastic rounding: implementation, error analysis and applications R. Soc. open sci. vol. 9, iss: 3, http://doi.org/10.1098/rsos.211631.

3. Mortikov E., Glazunov A. and Lykosov V. 2019: Numerical study of plane Couette flow: turbulence statistics and the structure of pressure–strain correlations Russ. J. Numer. Anal. Math. Model., vol. 34, iss: 2, 119-132, https://doi.org/10.1515/rnam-2019-0010.

4. Morinishi Y., Lund T., Vasilyev O. and Moin P. 1998: Fully conservative higher order finite difference schemes for incompressible flows J. Comp. Phys. 143, 90–124, https://doi.org/10.1006/jcph.1998.5962

Сравнение производительности параллельной СХД суперкомпьютера с разными версиями файловой системы Lustre

Р.А. Чулкевич, В.И. Козырев, А.Б. Шамсутдинов, П.С. Костенецкий

Национальный исследовательский университет "Высшая школа экономики"

Суперкомпьютер "cHARISMa" [1] активно используется 64 подразделениями НИУ ВШЭ для проведения научных исследований и учебной работы. Суперкомпьютер представляет собой высокопроизводительный вычислительный кластер с 46 вычислительными узлами, и параллельной СХД. Шесть вычислительных узлов кластера оснащены восьмью GPU NVIDIA A100 80 ГБ SXM в каждом, 29 узлов с большим объемом оперативной памяти 768-1536 ГБ оснащены четырьмя графическими ускорителями NVIDIA Tesla V100 32 ГБ SXM в каждом, а для задач, не требующих GPU, в составе кластера есть 11 вычислительных узлов без GPU с более мощными центральными процессорами. Система хранения данных (СХД) суперкомпьютера построена на базе параллельной сетевой файловой системы Lustre [3]. СХД построена на базе эталонной архитектуры, рекомендуемой Dell и состоит из двух Object Storage Server (OSS), двух Lustre Metadata Service (MDS), Lustre Metadata Target (MDT), Integrated Manager for Lustre (IML).

СХД является одним из важнейших компонентов суперкомпьютера, поскольку на ней хранятся как пользовательские данные для расчетов, так и результаты этих расчетов. Эффективная работа СХД тесно связана с поддержанием актуальности ее управляющего ПО. В марте 2022 года вышла новая LTS-версия Lustre 2.15.x, в связи с этим и было решено обновить Lustre на суперкомпьютере. Процесс обновления СХД суперкомпьютера не всегда тривиален, поэтому различные нюансы, упомянутые в данной статье, возможно, помогут системным администраторам, которым данные работы еще предстоят. Основные из них:

- в официальной таблице совместимости Lustre с ОС уже нет Centos 7 (который является довольно популярным для вычислительных кластеров), в то же время поддержка Centos 7 упоминается разработчиками в git-репозитории проекта, а также проверяется в тестах - это означает, что пакеты для данной ОС нужно собирать самостоятельно из исходных кодов, а ядро Centos 7 при этом должно быть актуальным;
- для установки новейшей версии Lustre необходим свежий IB-стэк (OFED/MOFED) - как минимум из-за различий в заголовочных файлах;
- сборка клиентской части не требует модификаций ядра, а часть для управляющих серверов (OSS/MDS) - требует: необходимо вручную пересобрать ядро с применением патчей для наилучшей производительности файловой системы.

В июне 2022 года отделом суперкомпьютерного моделирования НИУ ВШЭ выполнен комплекс работ по обновлению системного программного обеспечения суперкомпьютера "cHARISMa". Обновление проходило в несколько этапов. На всех вычислительных узлах кластера было обновлено ядро и релиз ОС, стек программного обеспечения вычислительной сети InfiniBand и микропрограммное обеспечение. Далее были обновлены клиенты Lustre на вычислительных узлах кластера. На заключительном этапе были обновлены ОС (включая самостоятельно собранное ядро новой версии), драйвера и управляющее ПО серверов СХД Lustre. Версии основного используемого программного обеспечения приведены в таблице 1. Файловая система Lustre была обновлена до актуального LTS-релиза 2.15.0. Особенностью данной версии стала поддержка NVIDIA GPUDirect Storage [2], благодаря которой можно ускорить обучение искусственных нейронных сетей. Обновление Lustre позволило значи-

тельно повысить скорость работы с файлами и обеспечить совместимость с новейшими версиями прикладного и системного программного обеспечения.

Таблица 1. Версии системного программного обеспечения на суперкомпьютере

Программное обеспечение	**До обновления**	**После обновления**
Клиент Lustre	2.11.0	2.15.0
Сервер Lustre	2.10.6	2.15.0
Ядро на вычислительных узлах	3.10.0-957.5	3.10.0-1160.59
Ядро на серверах Lustre	3.10.0-957.5	3.10.0-1160.49
Драйвера Mellanox (InfiniBand)	4.5-1.0.1.0	5.6-1.0.3.3

При обновлении отслеживались зависимости между новыми и старыми версиями программного, микропрограммного и аппаратного обеспечения. Чтобы сократить период обновления и не беспокоить пользователей, предварительно была создана виртуальная копия суперкомпьютера со всем установленным ПО: набор виртуальных машин, в точности повторяющих базовую конфигурацию основных систем (версии ОС, ядер, библиотек, системных пакетов, а также локальную версию СХД Lustre с аналогичными действующей параметрами). На этой копии были отработаны сценарии обновления. Также важным моментом было сохранение функциональности Integrated Manager for Lustre (IML) - управляющего менеджера Lustre, представляющего собой web-систему с возможностью управления файловой системой, и включающего в себя сервисы для автоматического failover (corosync и pacemaker). Следует отметить, что новейшие версии IML (5/6) лишены многих возможностей управления, которые имелись в версии 3 - именно эту версию было решено оставить. Подготовительные работы длились три месяца. Конфигурация СХД и узлов кластера была для тестов воссоздана в виртуальной среде и на ней подбирался оптимальный сценарий обновления ПО вычислительного кластера. Далее сценарий обновления был выполнен «на чистовик» за 48 часов. В результате работ сохранены все данные пользователей и обеспечена обратная совместимость, позволяющая запускать ранее подготовленные пользователями научные задачи.

Для сравнения производительности старой и новой версий программного обеспечения СХД проводилось тестирование скорости чтения/записи с использованием утилиты *dd*. На каждом вычислительном узле 25 раз выполнялась работа с файлами размером 1 ГБ. В процессе тестирования очереди задач были остановлены, а доступ пользователям закрыт - таким образом, никакие иные процессы не влияли на результаты тестов. Полученные результаты на тестах записи для четырех типов вычислительных узлов с различными характеристиками [1] приведены в таблице 2. Более сложные тесты не проводились из-за жесткого временного ограничения SLA.

В результате обновления ОС и файловой системы Lustre существенно выросла средняя скорость записи. При относительно похожих конфигурациях, наибольшее ускорение от обновления получили вычислительные узлы на базе CPU Intel Xeon Gold 6240R и AMD EPYC 7702 (32,8% и 30% соответственно), а вычислительные узлы с более старыми процессорами Intel Xeon Gold 6152 дали меньшее ускорение - 3,8%. Предположительно, разница в ускорении вызвана оптимизацией новой ОС и Lustre под новейшее программное и аппаратное обеспечение. Более детальное изучение этих результатов является дальнейшим направлением этого исследования.

Важно отметить, что после обновления значительно повысилась эффективность системы кэширования файловой системы. Теперь при повторном обращении к файлу на СХД скорость его чтения увеличивается независимо от клиента, т.к. содержимое файла разме-

Таблица 2. Сравнение скорости записи до и после обновления

Вычислительные узлы	**До обновления**	**После обновления**	**Ускорение,%**
	AVG, GB/s	AVG, GB/s	
01-26 Dell C4140K, Xeon Gold 6152	1.06	1.1	3.77
26-29 Dell C4140M, Xeon Gold 6240R	1.19	1.58	32.77
30-40 Dell R640, Xeon Gold 6248R	1.24	1.47	18.54
41-46 HPE XL675dG10+, EPYC7702	1.2	1.56	30

щается в кэше файловой системы. Ускорение достигает четырех раз при доступе с вычислительного узла, выполнявшего первое чтение, и до трех раз – с других узлов. По этой причине мы не приводим таблицу с результатами сравнения скоростей чтения: до обновления значимое ускорение обращения к одному и тому же файлу достигалось только на том же узле, с которого было повторное обращение, но не с других.

Благодаря произведенному комплексу работ, выполнение программ, осуществляющих ввод/вывод в файлы, заметно ускорилось, что повысило общую производительность суперкомпьютера НИУ ВШЭ. Авторы рекомендуют обновление ОС и файловой системы Lustre до версии 2.15.0 на всех современных вычислительных кластерах.

Литература

1. Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I. HPC Resources of the Higher School of Economics // Journal of Physics: Conference Series. 2021. Т. 1740, № 1.

2. NVIDIA GPUDirect Storage Benchmarking and Configuration Guide:: NVIDIA GPUDirect Storage Documentation. (n.d.). Retrieved May 10, 2022, from https://docs.nvidia.com/gpudirect-storage/configuration-guide/index.html

3. Dai, D., Gatla, O. R., Zheng, M. (2019). A Performance Study of Lustre File System Checker: Bottlenecks and Potentials. IEEE Symposium on Mass Storage Systems and Technologies, 2019-May, 7–13. https://doi.org/10.1109/MSST.2019.00-20

Экспериментальное наблюдение свойств квантового компьютера как открытой квантовой системы*

П.Е. Ведруков[1], Д.С. Куландин[1], А.В. Линев[1], И.Б. Мееров[1], С. Денисов[1,2]

Нижегородский государственный университет им. Н. И. Лобачевского[1]

Oslo Metropolitan University[2]

Введение. В работе представлены результаты экспериментов, выполненных в рамках исследования квантового компьютера как открытой квантовой системы, описываемой уравнением Линдблада. Использовались квантовые схемы, состоящие из предварительного перевода одного или нескольких кубитов в собственный базис {x,y,z}, выжидающего тождественного преобразования определенной глубины, возвращающего перевода в исходный базис и измерения, что позволило наблюдать динамику изменения значений наблюдаемых квантовой системы в зависимости от времени ее работы [1]. Обработка результатов измерений с помощью метода гармонической инверсии [2] показала зависимости поведения наблюдаемых от времени работы и степени локальности наблюдаемой, качественно похожие на представленные в работах [1,3]. Эксперименты проведены на платформе IBM Quantum Computing [4], использовался 5-кубитный квантовый компьютер ibmq_belem.

Моделирование диссипативной системы кубитов. Динамика открытой квантовой системы описывается уравнением Линдблада: $\frac{\partial}{\partial t}\rho = -i[H,\rho] + \sum_{\alpha,\beta=1}^{N^2-1} K_{\beta\alpha}\left(F_\alpha \rho F_\beta^\dagger - \frac{1}{2}\left\{F_\beta^\dagger F_\alpha, \rho\right\}\right)$, где $\rho(t)$ – матрица плотности, $\{F_\alpha\}$ – каналы диссипации. Динамику квантового компьютера можно анализировать, собирая значения наблюдаемых. В работах [1,3] было предсказано и частично подтверждено экспериментально, что при локальной диссипации возникают несколько уровней времен релаксации, отражающих степень локальности наблюдаемых, а также было продемонстрировано в численном эксперименте качественное сходство спектра квантовой системы и набора собственных значений, восстановленных по измерениям наблюдаемых.

Методика проведения эксперимента. Идея эксперимента состоит в использовании квантовых схем, которые выполняют тождественное преобразование, что позволяет нам исключить оператор Гамильтониана. Мы используем диссипацию, вызванную ошибкой гейта $U(\theta,\varphi,\lambda)$, и предполагаем, что диссипация локальна, то есть воздействует на каждый кубит независимо и никак не влияет на остальные. Используемые квантовые схемы имеют следующую структуру.

1. Перевод системы кубитов в состояние $|\sigma^{l_n} \dots \sigma^{l_2}\sigma^{l_1}\rangle$, где каждый кубит находится в каком-либо локальном базисе $l_j \in \{x, y, z, I\}$, выполняется посредством применения гейтов H и $S^\dagger$ к начальному состоянию $|00000\rangle$. Число кубитов, для которых был выполнен перевод в локальные базисы $\{x, y, z\}$, определяет порядок локальности наблюдаемой.

2. Выжидающее тождественное преобразование состоит из прямого и обратного диссипативных блоков. Прямой блок представляет собой t подблоков глубины 1 (параметр t в данном случае называется глубиной диссипативного блока), состоящих из операторов $U(\theta,\varphi,\lambda)$, где θ,φ,λ – случайные параметры, обозначающие углы вращения на сфере Блоха. Структура обратного блока симметрична прямому, но в нем используется обратные операторы (см. Рис 1).

3. Поворот каждого кубита в соответствии с его начальным локальным базисом и измерение состояния системы. При многократном повторении эксперимента статистика результатов измерений позволяет вычислить значение наблюдаемой $Tr\left(\rho_0 O^{(k)(t)}\right) = \sum_{i=0}^{2^n-1} p_i \lambda_i$, где k – уровень локальности наблюдаемой, t – глубина диссипативного блока, p_i – частота измерения в базисном состоянии $|i\rangle$, λ_i – произведение собственных чисел матриц Паули, соответствующих используемому набору локальных базисов и базисному состоянию $|i\rangle$.

Общее возможное количество схем – 3^n*T, где n – число кубитов, T – максимальная глубина выжидающего преобразования. Мы выполняли эксперименты для n=5, T=100 и набора соче-

* Исследования выполнены при поддержке научно-образовательного математического центра «Математика технологий будущего»

таний локальных базисов с разным порядком локальности. В каждом эксперименте выполнялось 20 000 запусков (измерений).

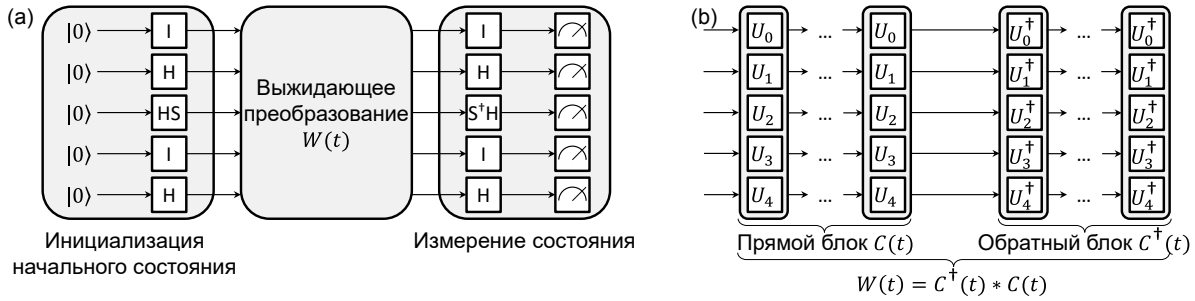


Рис. 1. (a) Структура квантовых схем и (b) выжидающего преобразования

Результат серии экспериментов для наблюдаемой $O^{(k)(t)}$ формирует сигнал $Tr\left(\rho_0 O^{(k)(t)}\right), t \in [0, T]$. Его обработка методом гармонической инверсии [2] позволяет получить основные частоты сигнала из задаваемого частотного диапазона и приближенно представить его виде суммы гармонических колебаний $Tr\left(\rho_0 O^{(k)(t)}\right) = \sum_n c_n e^{\lambda_n t}$ (см. Рис. 2).

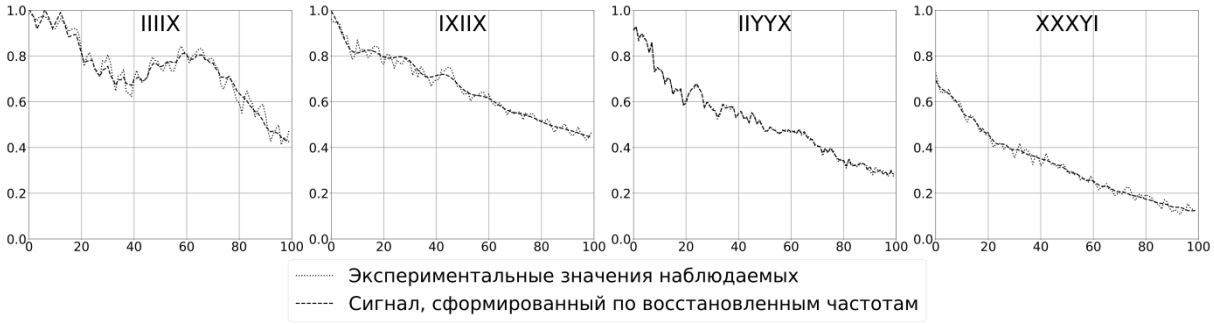


Рис. 2. Зависимость средних значений наблюдаемых IIIIX, IXIIX, IIYYX, XXXYI от глубины диссипативного блока квантовой схемы

Согласно [1], множество всех собственных чисел восстановленных сигналов может сформировать диаграмму, качественно сходную с полным спектром лиувиллиана квантовой системы (Рис. 3). Дальнейшие исследования позволят проверить это утверждение экспериментально.

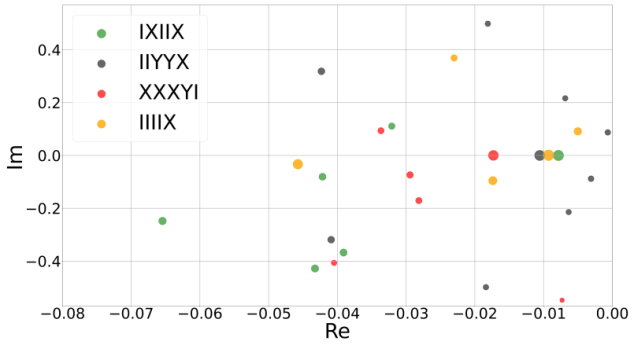


Рис. 3. Собственные значения восстановленных сигналов, диаметры кругов пропорциональны $|c_n|$

Литература

1. Sommer, O. E., Piazza, F., & Luitz, D. J. Many-body hierarchy of dissipative timescales in a quantum computer // Physical Review Research, 2021, 3(2). DOI: 10.1103/physrevresearch.3.023190
2. Mandelshtam, V. A., & Taylor, H. S. Harmonic inversion of time signals and its applications // The Journal of Chemical Physics, 1997, 107(17), 6756–6769. DOI: 10.1063/1.475324
3. K. Wang, F. Piazza, and D. J. Luitz, Hierarchy of Relaxation Timescales in Local Random Liouvillians // Physical Review Letters, 2020, 124(10). DOI: 10.1103/physrevlett.124.100604
4. IBM Quantum. URL: https://quantum-computing.ibm.com/ (дата обращения: 31.03.2022)

Содержание

Научное издание

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ
Труды международной конференции
26–27 сентября 2022 г. Москва

Издательство «МАКС Пресс»
Главный редактор: *Е. М. Бугачева*
Обложка: *М. А. Еронина*

Напечатано с готового оригинал-макета

Подписано в печать 26.09.2019 г. Формат 60x90 1/8.
Усл.печ.л. 20,75. Тираж 10 экз. Изд. №. 148.

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