HYBRID MPI + OPENMP ALGORITHM FOR SYMMETRIC SPARE MATRIX REORDERING AND ITS APPLICATION TO THE SOLVING SYSTEMS OF LINEAR EQUATIONS

A. Yu. Pirova

Lobachevsky State University,
603950, Nizhny Novgorod, Russia

DOI: 10.24412/2073-0667-2022-1-28-41

Systems of linear equations (SLAE) with large sparse matrix arise from numerical simulations in various scientific and engineering applications. SLAE solution is one of the time and memory-consuming stages of modeling, which requires efficient implementation on modern supercomputers. Depending on the properties of the matrix, direct and iterative methods of solving such systems are applied. Direct methods are based on matrix factorization into two triangular matrices and backward solution of these systems. A feature of direct methods is that the number of non-zero elements during the factorization step can significantly increase over the initial matrix. Symmetric row and column reordering is used before numerical factorization to minimize the fill-in. A suitable permutation reduces the memory consumption for storing the factor and the time required for the most time-consuming stage - numerical factorization. Therefore, it is important to develop new parallel reordering algorithms that reduce the total time for solving SLAEs with a sparse matrix, and hence, the time of numerical simulation in applications.

The problem of finding the ordering that minimizes the factor fill-in is NP-hard. In practice, two heuristic approaches are commonly used: the minimum degree and nested dissection algorithms. The nested dissection algorithm is built on the divide-and-conquer principle. At each step of the algorithm, a graph separator is found. The separator vertices divide the graph into two disconnected subgraphs of similar size. Then the separator vertices are numbered and removed from the graph, and the algorithm operates recursively on new subgraphs. The process stops when all vertices are numbered. There are many modifications of the nested dissection method that differ in the algorithm for finding the separator. Since 1993, the modifications of the nested dissection algorithm employing the multilevel approach are used in parallel computations.

Most sparse SLAE solvers have built-in implementations of reordering methods, as well as an interface for using third-party libraries and user permutations. Open source libraries ParMETIS and PT-Scotch are widely used in the academic community. They implement a multilevel nested dissection algorithm for distributed memory systems using MPI technology. In particular, the open academic direct solver MUMPS has an interface for using these libraries. The ParMETIS library also has a version for shared memory systems called mt-metis. The commercial solver Intel MKL PARDISO and its cluster version Intel Cluster PARDISO are examples of tight integration of the solver and ordering routine. They include optimized versions of the METIS library, designed for shared-memory and distributed-memory systems, but its approaches to optimizing and combining shared and distributed memory computations during the reordering process have not been published.

Earlier, we presented the PMORSy and DMORSy reordering libraries that implement the multilevel nested section method for shared- and distributed-memory systems, respectively. The ordering

The study was supported by Ministry of Science and Higher Education, project № 0729-2020-0055

algorithm in PMORSy is based on parallel processing of the task queue. The task is to find the separator in the current subgraph. After calculating the separator, new subgraphs go to the shared task queue, from which they are assigned for execution to free threads. The algorithm is implemented using OpenMP technology. The parallel algorithm in DMORSy is based on parallel processing of a binary tree constructed during the nested dissection method. Let the original graph be distributed among P processes. Then its separator is calculated in parallel by all P processes using a multilevel algorithm similar to that used in PT-Scotch. After the separator is found, new subgraphs are redistributed to P / 2 processes each, and the ordering process is continued independently for new subgraphs. The algorithm is implemented using MPI technology.

In this paper, we propose a hybrid MPI + OpenMP parallel algorithm for distributed memory systems, in which the use of processes and threads within a single computing node is combined. The algorithm is constructed as follows. While the input graph is distributed among $P > 1$ processes, its separator is calculated in parallel by all P processes using a multilevel algorithm from DMORSy. Once the input subgraph is stored on a single process, the ordering is performed using a parallel task queue according to the algorithm from PMORSy. The combination of parallelization schemes includes their advantages: the scalability of the distributed memory algorithm and less factor fill-in obtained by the algorithm on shared memory.

We show the competitiveness of the implementation in comparison with analogs in terms of ordering time and factor fill-in. We test our implementation on 37 symmetric matrices from the SuiteSparse Matrix Collection and LOGOS-FEM Collection. Computational experiments show that the use of a hybrid parallelization scheme in comparison with the pure MPI DMORSy parallelization reduces the run-time of reordering by 5 % on average when working on two computational nodes. In comparison with ParMETIS, hybrid DMORSy works faster on most matrices, the average advantage is 2 times for matrices over 1 million rows and 4.9 times for matrices less than 1 million rows. Moreover, hybrid DMORSy produces orderings of 8 % on average smaller fill-in than ParMETIS for a 2 / 3 of test matrices. Further, we tested the obtained permutations for solving the SLAEs. For this, a series of experiments were carried out with the open source solver MUMPS. The use of the hybrid DMORSy permutations allowed us to reduce the run-time of the solver on 26 matrices out of 37 in comparison with the use of permutations from ParMETIS (on average, by 26 %). For most of these matrices, the lead is obtained both by reducing the ordering time and by the time of numerical factorization.

Key words: nested dissection ordering, sparse matrix reordering, distributed-memory parallel algorithms, sparse matrix solve.

References

1. Karypis G. and Kumar V. ParMetis: Parallel graph partitioning and sparse matrix ordering library. Tech. Rep. TR 97-060, University of Minnesota, Department of Computer Science. 1997.

2. Chevalier C., Pellegrini F. PT-Scotch: A tool for efficient parallel graph ordering // Parallel Computing. 2008. Vol. 34, N 6. P. 318–331. DOI:10.1016/j.parco.2007.12.001.

3. MUltifrontal Massively Parallel Solver (MUMPS). [Electron. Res.]: `http://mumps.enseeiht.fr/index.php?page=home` (date of access: 10.11.2021).

4. LaSalle D. and Karypis G. Efficient Nested Dissection for Multicore Architectures // Euro-Par 2015: Parallel Processing. 2015, Springer Berlin Heidelberg. P. 467–478. DOI:10.1007/978-3-662-48096-0_36.

5. OneMKL PARDISO — Parallel Direct Sparse Solver Interface. Developer Reference. [Electron. Res.]: `https://www.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/sparse-solver-routines/onemkl-pardiso-parallel-direct-sparse-solver-iface.html` (date of access: 10.11.2021).

6. Parallel Direct Sparse Solver for Clusters Interface. Developer Reference. [Electron. Res.]: `https://www.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/sparse-solver-routines/parallel-direct-sp-solver-for-clusters-iface.html` (date of access: 10.11.2021).

7. Pirova A., Meyerov I., Kozinov E., Lebedev S. PMORSy: parallel sparse matrix ordering software for fill-in minimization // Optimization Methods and Software. 2017. Vol. 32. N 2. P. 274–289. DOI: 10.1080/10556788.2016.1193177.

8. Pirova A. Yu., Meyerov I. B., Kozinov E. A. Programmny complex DMORSy dlya pereuporyadocheniya razrezhennyh matric na clasternyh systemah // Trudy Mezdunarodnoy conferencii „Russian Supercomputing Days“ (Moskva, 24–25 Sentybrya 2018 g.). M: Izdatelstvo MGU. 2018. S. 749–757.

9. Pirova A. Yu. Gibridny MPI + OpenMP algoritm pereuporyadocheniya simmetrichnyh razrezhennyh matric dlya system s raspredelennoy pamyatiy // Matematicheskoe Modelirovanie I Supercomputernye Technologii. Trudy XXI Mezdunarodnoy conferencii (N. Novgorod, 22–26 noaybrya 2021 g.). Nizhni Novgorod: izdatelstvo NNGU, 2021. S. 268–273.

10. Suite Sparse Matrix Collection. [Electron. Res.]: `https://sparse.tamu.edu/` (date of access: 10.11.2021).

ГИБРИДНЫЙ MPI + OPENMP АЛГОРИТМ ПЕРЕУПОРЯДОЧЕНИЯ СИММЕТРИЧНЫХ РАЗРЕЖЕННЫХ МАТРИЦ И ЕГО ПРИМЕНЕНИЕ К РЕШЕНИЮ СЛАУ

А. Ю. Пирова

Национальный исследовательский Нижегородский государственный университет им. Н. И. Лобачевского,
603950, Нижний Новгород, Россия

УДК 519.688, 004.428
DOI: 10.24412/2073-0667-2022-1-28-41

В работе рассматривается задача переупорядочения строк и столбцов разреженной матрицы с целью уменьшения заполнения фактора при прямом решении СЛАУ. Предлагается параллельный алгоритм многоуровневого метода вложенных сечений для систем с распределенной памятью, в котором выполняется согласованное использование процессов и потоков в рамках одной вычислительной системы. Приводятся результаты вычислительных экспериментов, показывающие конкурентоспособность реализации в сравнении с аналогами по времени переупорядочения и заполнению фактора матриц. Показано, что применение полученных перестановок позволяет сократить время решения СЛАУ с помощью открытой библиотеки MUMPS на ряде тестовых задач.

Ключевые слова: метод вложенных сечений, переупорядочение разреженных матриц, параллельный алгоритм, решение разреженных СЛАУ.

Введение. При выполнении численного моделирования в различных прикладных областях (аэро- и гидродинамике, моделировании электромагнитных волн, сред, геофизических процессов и др.), как правило, требуется численное решение дифференциальных уравнений в частных производных. Одним из ключевых этапов их решения, требующим значительных временных затрат, является решение систем линейных уравнений, полученных в ходе дискретизации. Как правило, при моделировании двумерных и трехмерных систем, построенные в ходе расчетов матрицы имеют большой порядок (несколько миллионов строк) и малое число ненулевых элементов. В зависимости от свойств матрицы, для решения СЛАУ используются прямые или итерационные методы. В ходе прямого решения СЛАУ выполняется разложение матрицы системы на произведение двух треугольных матриц, называемых *факторами*, при этом число ненулевых элементов фактора в разы больше, чем в исходной матрице. Поэтому выполняется предобработка матрицы — ее *переупорядочение*, то есть нахождение симметричной перестановки строк и столбцов исходной матрицы с целью уменьшения заполнения. Таким образом, прямое решение СЛАУ с разреженной матрицей выполняется в четыре этапа: переупорядочение, символьная факторизация (фаза анализа), численная факторизация, обратный ход метода Гаусса. Нахождение подходящей перестановки позволяет сократить затраты памяти для хранения фактора, время, необходимое для наиболее трудоемкого этапа – численной факторизации, а также

влияет на потенциал параллелизма численной факторизации. Поэтому актуальна разработка новых параллельных алгоритмов переупорядочения, позволяющих сократить общее время решения СЛАУ с разреженной матрицей, а значит, время численного моделирования в прикладных задачах.

Задача нахождения оптимальной перестановки, минимизирующей размер фактора, NP-трудная [1]. На практике для ее решения применяются эвристические методы, позволяющие за относительно небольшое время работы найти перестановку, приемлемую по заполнению для дальнейшей факторизации. Как правило, для минимизации заполнения применяются модификации метода минимальной степени или метода вложенных сечений.

В большинстве решателей разреженных СЛАУ имеются встроенные реализации методов переупорядочения, а также интерфейс для использования сторонних библиотек и пользовательских перестановок. Широкое применение в научном сообществе получили библиотеки ParMETIS [2] и PT-Scotch [3], в которых реализован многоуровневый метод вложенных сечений для систем с распределенной памятью с использованием технологии MPI. В частности, открытый академический решатель MUMPS [4], предназначенный для кластерных систем, имеет интерфейс для использования данных библиотек. Библиотека ParMETIS имеет отдельную версию для систем с общей памятью, mt-metis [5]. Коммерческий решатель Intel MKL PARDISO [6] и его кластерная версия Intel Cluster PARDISO [7] являются примером тесной интеграции решателя и переупорядочивателя. Они включают оптимизированные версии библиотеки METIS, однако примененные там подходы к оптимизации и комбинированию вычислений на общей и распределенной памяти в рамках переупорядочения не опубликованы.

Автором данной работы ранее были представлены библиотеки для переупорядочения, в которых реализован параллельный многоуровневый метод вложенных сечений для систем с общей памятью PMORSy [8] и систем с распределенной памятью DMORSy [9]. В данной работе предлагается гибридная схема распараллеливания метода вложенных сечений, объединяющая ранее описанные подходы. Применение данной схемы позволяет задавать число используемых параллельным алгоритмом процессов и потоков исходя из архитектуры вычислительной системы, а значит, может позволить более гибко настраивать переупорядочиватель для решения СЛАУ.

Работа рекомендована к печати программным комитетом конференции „Математическое моделирование и суперкомпьютерные технологии“ и является расширенным вариантом тезисов [10], опубликованных в материалах конференции. В сравнении с [10] существенно расширена экспериментальная часть работы и ее анализ. Статья построена следующим образом. В разделе 1 дается описание предлагаемого гибридного параллельного алгоритма для переупорядочения разреженных матриц. В разделе 2 приводится анализ производительности гибридного алгоритма в сравнении с предыдущими реализациями и переупорядочивателем ParMETIS. В разделе 3 изучаются результаты применения полученных перестановок в решателе MUMPS, дается сравнение времени работы решателя при использовании перестановок из DMORSy и ParMETIS.

1. Параллельный алгоритм переупорядочения. Алгоритм переупорядочения, рассматриваемый в данной работе, основан на многоуровневом методе вложенных сечений. Для его выполнения по исходной матрице системы строится граф, в котором каждая вершина соответствует строке матрицы, а каждое ребро — ненулевому элементу. Работа метода вложенных сечений основана на принципе „разделяй и властвуй“ и заключается в многократном нахождении вершинных разделителей. В начале работы вычисляется

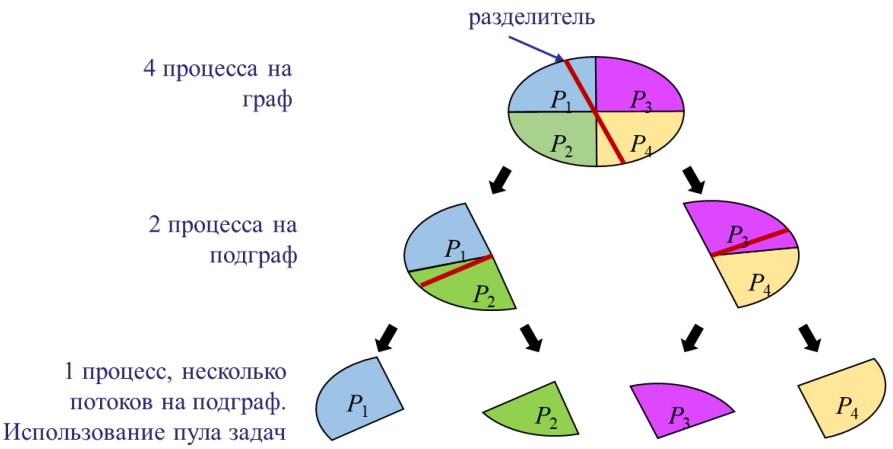

Рис. 1. Гибридный MPI + OpenMP алгоритм переупорядочения. Адаптировано из [9]

вершинный разделитель исходного графа, при этом множество вершин, его составляющих, нумеруются и удаляются из графа. Подходящим считается разделитель, имеющий наименьшее число вершин при том, что после его удаления образуются два близких по размеру подграфа. Затем переупорядочение продолжается аналогично на графах, образовавшихся после удаления разделителя. Перестановка найдена, когда все вершины графа занумерованы.

Гибридный параллельный алгоритм переупорядочения, предлагаемый в работе, основан на комбинации реализованных ранее параллельных алгоритмов для систем с общей и распределенной памятью [8, 9]. Пусть исходный граф хранится распределено на P процессах (рис. 1). Тогда разделитель для него вычисляется параллельно всеми P процессами по многоуровневому алгоритму, описанному в [9]. После того как разделитель найден, новые подграфы перераспределяются на $P/2$ процессов каждый, и нахождение разделителей продолжается независимо процессами, хранящими подграф. Как только на процессе хранится подграф целиком, переупорядочение выполняется с использованием параллельной очереди задач, которая обрабатывается несколькими потоками. Одной задачей, назначаемой потоку, является нахождение разделителя в подграфе. После вычисления разделителя, новые подграфы попадают в очередь задач процесса, из которой захватываются на выполнение свободными потоками. Псевдокод алгоритма представлен на рис. 2.

Реализация алгоритма выполнена с использованием технологий MPI и OpenMP. Комбинирование схем распараллеливания позволило объединить их преимущества: масштабируемость алгоритма для систем с распределенной памятью и меньшее заполнение фактора, получаемое алгоритмом на общей памяти.

2. Анализ гибридной реализации.

2.1. *Методика проведения экспериментов*. Целью первой серии вычислительных экспериментов было сравнение результатов работы гибридного алгоритма DMORSy_hybrid с результатами исходного распределенного алгоритма DMORSy и сторонними библиотеками ParMETIS v.4.0.3 (для распределенной памяти) и mt-metis v. 7.0.2 (для общей памяти). Результаты переупорядочения сравнивались по времени получения перестановок и заполнению факторов матриц. Все указанные пакеты запускались с настройками переупорядочения по умолчанию. Вычислительные эксперименты проводили на узлах кластера МСЦ

```
1  int* NDStepParallel(graph G0, int barrier) {
2      Find separator G0->sep using one thread;
3      Find subgraphs G1, G2, \dots Gk using one thread. Get G0->subgraphs[]
4      for (i = 0; i < k; i++) {
5          if (Gi > barrier)
6          #pragma omp task
7              Gi->sep = NDStepParallel(Gi, barrier);
8          else
9              Gi->sep = NDStepSequential(Gi);
10     }
11     return G0->sep;
12 }
13
14
15 int* PMORSyOrdering(graph G, parameters, int nThreads) {
16     omp_set_num_threads(nThreads);
17     #pragma omp parallel
18     #pragma omp single
19         G->Sep = NDStepParallel(G, barrier);
20     separators = Merge(separators from graph tree with root G);
21     return separators;
22 }
23
24 int* NDStepMPI(graph currGraph, parameters, int nProcs) {
25     Find separator of currGraph currGraph->Sep using all processes;
26     Find two subgraphs subgraph[0], subgraph[1] using all processes;
27     Distribute subgraph[0] to processes 0, 2, \dots, nProcs / 2;
28     Distribute subgraph[1] to processes 1, 3, \dots, nProcs / 2;
29     currGraph->subgraph = subgraph[proc_id
30
31     return currGraph->Sep;
32 }
33
34 int* NestedDissectionMPI(graph G(V, E), int nProcs, int nThreads) {
35     currGraph = G;
36     while ((currGraph != NULL) && (nProcs > 1)) {
37         currGraph->Sep = NDStepMPI(currGraph, parameters, nProcs);
38         currGraph = currGraph->subgraph;
39         nProcs = nProcs / 2;
40 }
41 if (currGraph != NULL) {
42     currGraph->Sep = PMORSyOrdering(currGraph, parameters, nThreads);
43 }
44 iperm = Merge(separators from graph tree with root G);
45 return iperm;
46 }
```

Рис. 2. Псевдокод гибридного MPI + OpenMP алгоритма вложенных сечений

РАН со следующими характеристиками: процессор Intel Xeon Gold 6154 (Skylake), 3,0 GHz, 2 x 18 ядер, память 192 GB, компилятор Intel Parallel Studio XE 2017 Cluster Edition. Всего рассматривалось 27 матриц из коллекции SuiteSparse [11] порядком от $2 \cdot 10^6$ до $1{,}5 \cdot 10^6$

Таблица 1

Параметры тестовых матриц

название	N	nz	nz / n2	коллекция
ecology2	999 999	2 997 995	3,00E-06	SuiteSparse
ecology1	1 000 000	2 998 000	3,00E-06	SuiteSparse
CurlCurl_3	1 219 574	7 382 096	4,96E-06	SuiteSparse
thermal2	1 228 045	4 904 179	3,25E-06	SuiteSparse
Kamaz_gusev	1 429 158	50 191 148	2,46E-05	ЛОГОС
Geo_1438	1 437 960	32 297 325	1,56E-05	SuiteSparse
StocF-1465	1 465 137	11 235 263	5,23E-06	SuiteSparse
Hook_1498	1 498 023	31 207 734	1,39E-05	SuiteSparse
af_shell10	1 508 065	27 090 195	1,19E-05	SuiteSparse
Flan_1565	1 564 794	59 485 419	2,43E-05	SuiteSparse
G3_circuit	1 585 478	4 623 152	1,84E-06	SuiteSparse
lopatka2	2 545 314	88 273 521	1,36E-05	ЛОГОС
49_750	2 615 169	97 081 773	1,42E-05	ЛОГОС
p4_6	4 216 212	144 714 294	8,14E-06	ЛОГОС

и 10 матриц, сгенерированных пакетом „ЛОГОС“ в ходе расчетов задач прочности. В работе приведены результаты для матриц порядка более 1 млн. строк, их характеристики указаны в табл. 1.

2.2. *Анализ производительности и заполнения фактора матриц.* Для анализа результатов работы переупорядочения были произведены запуски на одном и двух вычислительных узлах кластера, а также проведено сравнение с библиотеками ParMETIS и mt-metis.

В рамках одного вычислительного узла были проведены запуски пакетов для общей памяти PMORSy, mt-metis и пакетов для распределенной памяти DMORSy, DMORSy_hybrid и ParMETIS. Запуски производились на 32 вычислительных ядрах. Для гибридной версии было выбрано соотношение 16 процессов по 2 потока для всех тестовых матриц.

Сравнение времени работы (рис. 3) показывает, что DMORSy и DMORSy_hybrid работают быстрее PMORSy за счет параллельного выполнения этапов многоуровневого метода. В сравнении с версией для общей памяти, сокращение времени работы в среднем составляет 2,4 раза. В сравнении с mt-metis и ParMETIS, DMORSy_hybrid работает быстрее на половине тестовых матриц (в среднем, в 1,5 раза относительно mt-metis, в 1,9 раз относительно ParMETIS). Использование гибридной схемы распараллеливания на одном вычислительном узле не дает дополнительного выигрыша по времени по сравнению с базовой MPI версией, однако при использовании нескольких вычислительных узлов будет показано ее преимущество.

Сравнение заполнения факторов матриц (рис. 4) показывает, что результаты, полученные PMORSy, DMORSy и DMORSy_hybrid близки. Для 10 из 14 матриц наименьшее заполнение фактора было получено пакетами PMORSy или DMORSy_hybrid, для оставшихся 4 матриц — пакетом ParMETIS. В сравнении со сторонними пакетами, DMORSy_hybrid позволяет получить лучшее заполнение фактора на половине тестовых матриц, получая до 20 % меньше, чем mt-metis, и до 12 % меньше, чем ParMETIS.

Далее сравним результаты работы DMORSy, DMORSy_hybrid и ParMETIS при работе на 32 ядрах двух вычислительных узлов (рис. 5). Для DMORSy_hybrid для каждой

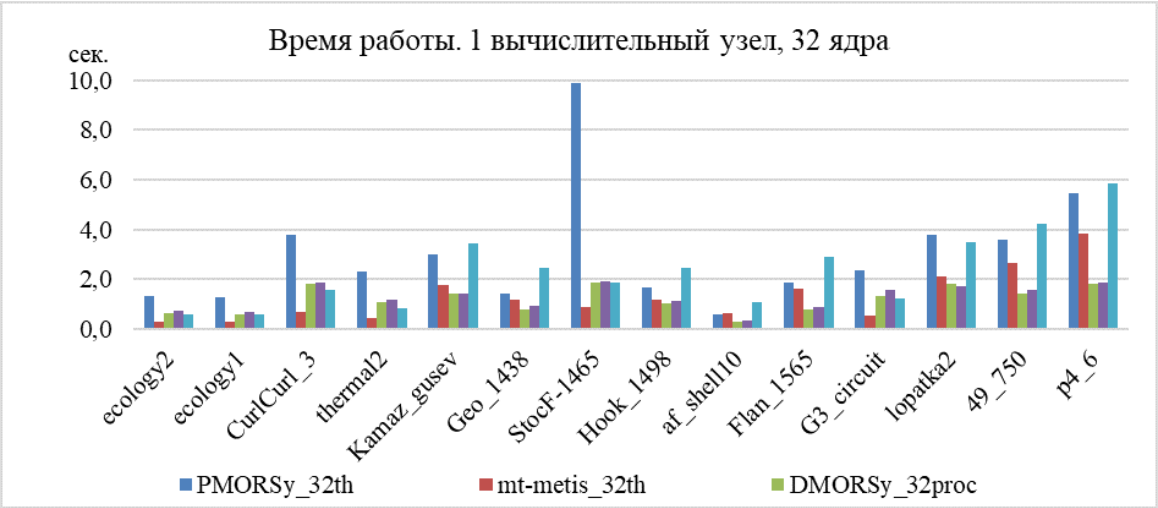

Рис. 3. Сравнение времени работы переупорядочивателей на одном вычислительном узле

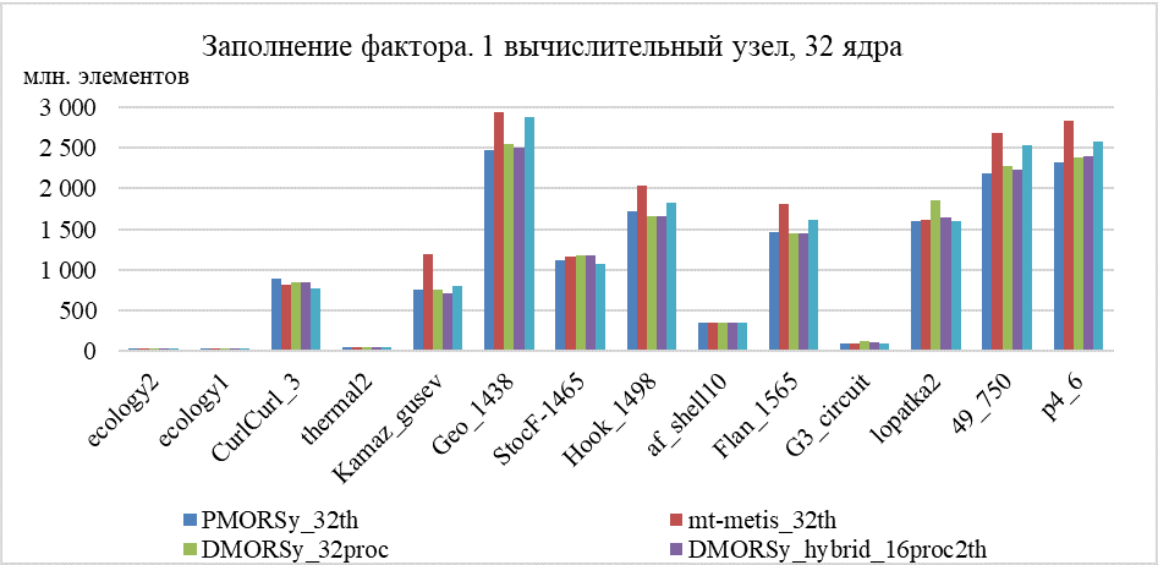

Рис. 4. Сравнение заполнения фактора, полученного переупорядочивателями на одном вычислительном узле

матрицы соотношение числа процессов и потоков выбиралось таким образом, чтобы время работы было наименьшим. Для большинства матриц была выбрана конфигурация 16 процессов по 2 потока, для остальных — 8 процессов по 4 потока или 32 процесса по 1 потоку. Использование гибридной схемы распараллеливания позволило сократить время работы DMORSy для большинства матриц (в среднем, на 5 %, наибольший выигрыш — 12 %). В сравнении с ParMETIS, DMORSy_hybrid работает быстрее на 11 матрицах из 14, в среднем опережая в 2,2 раза (наибольший выигрыш — 3,8 раз).

Сравним заполнение факторов матриц, полученное при работе на двух вычислительных узлах (рис. 6). Также как и при работе на одном вычислительном узле, заполнение факторов, полученное DMORSy и DMORSy_hybrid, отличается незначительно (разница в пределах 2 %). При этом DMORSy_hybrid было получено меньшее заполнение факто-

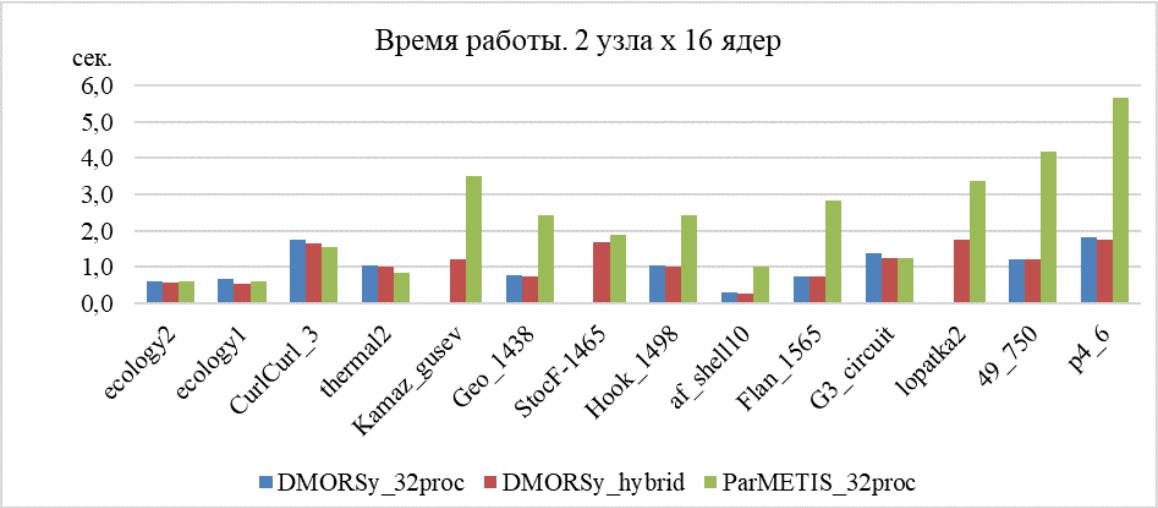

Рис. 5. Сравнение времени работы переупорядочивателей на двух вычислительных узлах

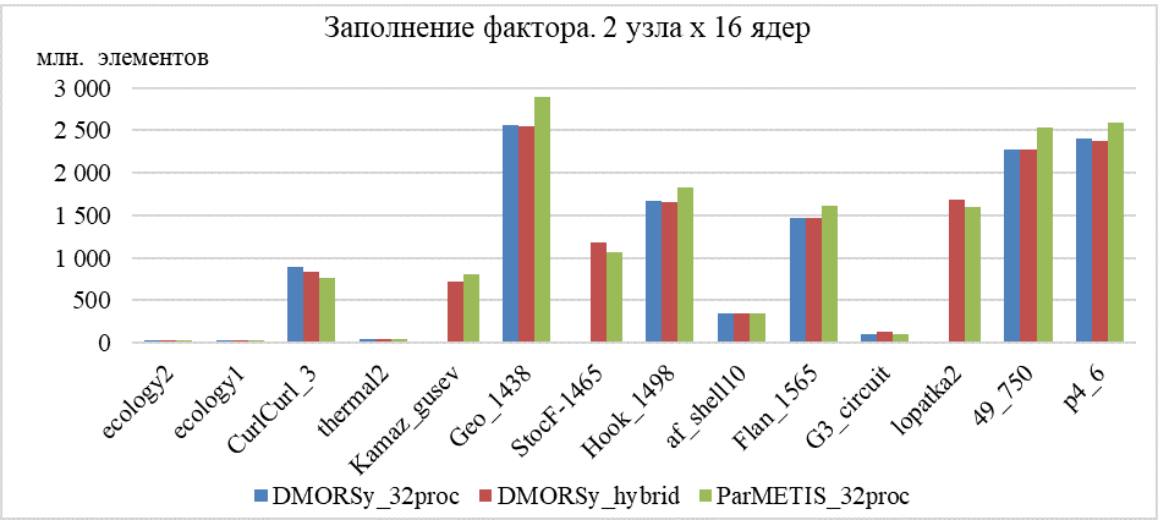

Рис. 6. Заполнение фактора матриц, полученного переупорядочивателями на двух вычислительных узлах

ра, чем ParMETIS, для половины тестовых матриц (среднее опережение 8 %, наибольший выигрыш — 12 %).

Ранее было показано, что DMORSy и DMORSy_hybrid работают быстрее, чем ParMETIS, на матрицах порядка менее 1 млн. строк и матрицах с регулярной структурой (все матрицы из коллекции ЛОГОС и часть матриц из коллекции SuiteSparse). Матрицы с регулярной структурой, в частности, полученные в ходе конечно-элементных расчетов, допускают точное сжатие, т. е. возможность объединить строки матрицы с одинаковым расположением ненулевых элементов. Этот прием позволяет сократить время работы переупорядочения в разы с сохранением размера фактора, полученного в результате перестановки.

3. Применение перестановок для решения разреженных СЛАУ.

3.1 *Методика проведения экспериментов.* Для тестирования возможности использования полученных перестановок для решения СЛАУ была проведена серия экспериментов с открытым решателем СЛАУ MUMPS [4]. Решалась совместная СЛАУ $Ax = b$, где A —

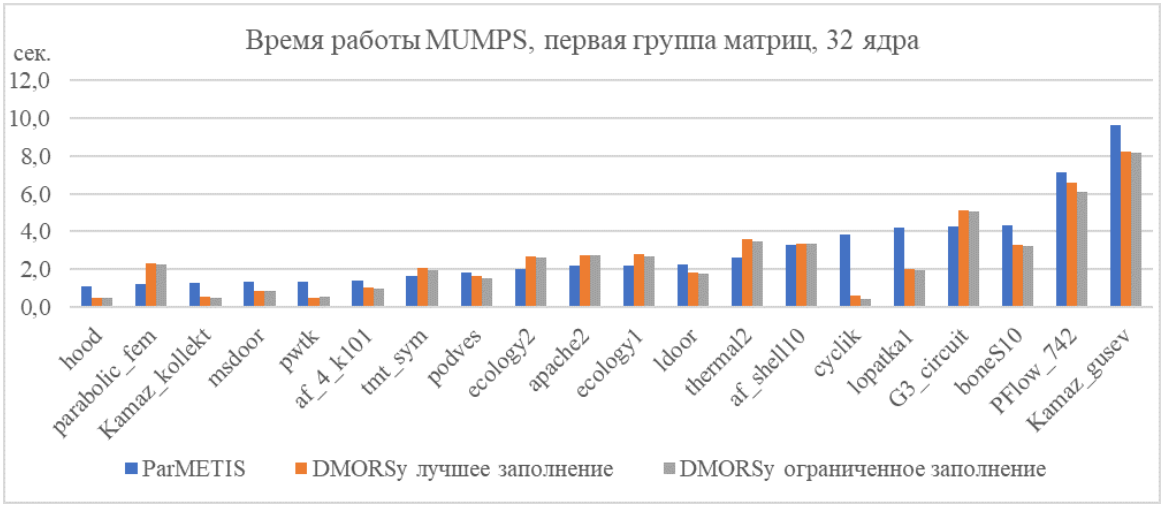

Рис. 7. Сравнение времени работы решателя MUMPS при использовании разных перестановок. Время работы решателя до 10 секунд

матрица из тестового набора, x — заданный вектор неизвестных. Применялось LDL^T разложение матрицы системы с указанием типа матрицы (симметричная положительно определенная или симметричная неопределенная). Решение выполнялось на одном вычислительном узле в 32 процесса, матрица изначально была разделена на равные полосы между процессами.

Проведем сравнение результатов решения СЛАУ при трех вариантах перестановок: 1) перестановка из ParMETIS, ParMETIS запускался через встроенный в MUMPS интерфейс; 2) перестановка из DMORSy, с параметрами, дающими наименьшее заполнение фактора при работе в 32 процесса; 3) перестановка из DMORSy, с параметрами, ограничивающими время работы переупорядочения за счет сокращения отдельных этапов многоуровневого метода вложенных сечений.

Для DMORSy с фиксированными параметрами производились запуски на одном вычислительном узле при разном соотношении числа процессов и потоков, затем были выбраны перестановки, дающие минимальное суммарное время работы переупорядочивателя и решателя. Перестановки DMORSy подавались в MUMPS как пользовательские.

3.2. *Результаты решения СЛАУ с различными перестановками.* Время работы решателя MUMPS при различных перестановках приведено ниже. На рисунках матрицы разделены на две группы — те, на которых время работы MUMPS менее 10 секунд (рис. 7), и те, на которых время работы MUMPS более 10 секунд (рис. 8).

Применение перестановок DMORSy с параметрами, дающими наименьшее заполнение фактора, позволило сократить время работы решателя на 26 матрицах из 37 в сравнении с использованием перестановки из ParMETIS (в среднем, на 26 %). Для большинства из этих матриц опережение получено как за счет сокращения времени переупорядочения, так и за счет времени численной факторизации.

Использование параметров DMORSy с ограниченным заполнением фактора (конфигурация 3) дало перестановки с большим заполнением фактора (в среднем, +3 %), при этом время переупорядочения было сокращено от 20 % до 2х раз. Применение таких перестановок к решению СЛАУ на 18 матрицах позволило дополнительно сократить время работы решателя относительно ParMETIS за счет времени переупорядочения, средний выигрыш

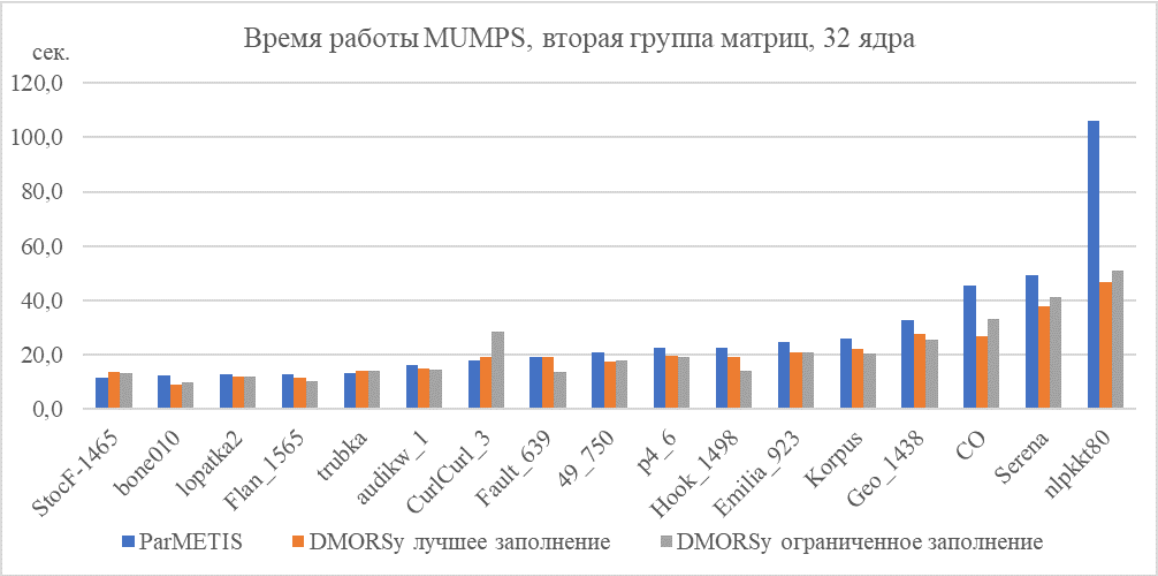

Рис. 8. Сравнение времени работы решателя MUMPS при использовании разных перестановок. Время работы решателя более 10 секунд

общего времени работы составил 29 %. Еще на 8 матрицах время работы решателя было по-прежнему меньше, чем при использовании перестановок ParMETIS, но больше, чем при использовании перестановок DMORSy с параметрами из конфигурации 2.

Из рис. 9 видно, что при близком заполнении матриц после переупорядочения (матрицы StocF-1465, audikw_1, CurlCurl_3) время численной факторизации при использовании разных перестановок отличается незначительно. Если DMORSy было получено заполнение на 10 % меньше, то время численной факторизации при этом сокращалось на 20 % (например, матрицы 49_750, p4_6, Geo_1438). Для 20 матриц из 37 численная факторизация после перестановок DMORSy была быстрее более чем на 13 %.

На тех матрицах, где использование DMORSy замедляет общее решение СЛАУ, отставание в общем времени работы MUMPS является накладными расходами на использование пользовательской перестановки, в частности, последовательной фазой анализа. Преимуществом использования встроенной в решатель перестановки является возможность параллельного выполнения фазы анализа, что дает сокращение времени работы этого этапа в 1,5–3,5 раза. При интеграции переупорядочивателя в решатель возможно использовать часть информации о перестановке для предварительного построения дерева исключения, как это сделано в MUMPS. В некоторых решателях выполняется оценка заполнения фактора в процессе нахождения перестановки, что позволяет ограничить время работы переупорядочения при получении перестановки не с наименьшим возможным размером фактора, но приемлемой для факторизации.

Заключение. В работе рассматривается задача переупорядочения симметричной разреженной матрицы с целью уменьшения заполнения фактора при прямом решении СЛАУ. Как правило, для решения данной задачи используются эвристические методы, основанные на методе вложенных сечений или методе минимальной степени. Ранее автором были предложены параллельные реализации многоуровневого метода вложенных сечений для систем с общей памятью [8] и систем с распределенной памятью [9]. В данной работе

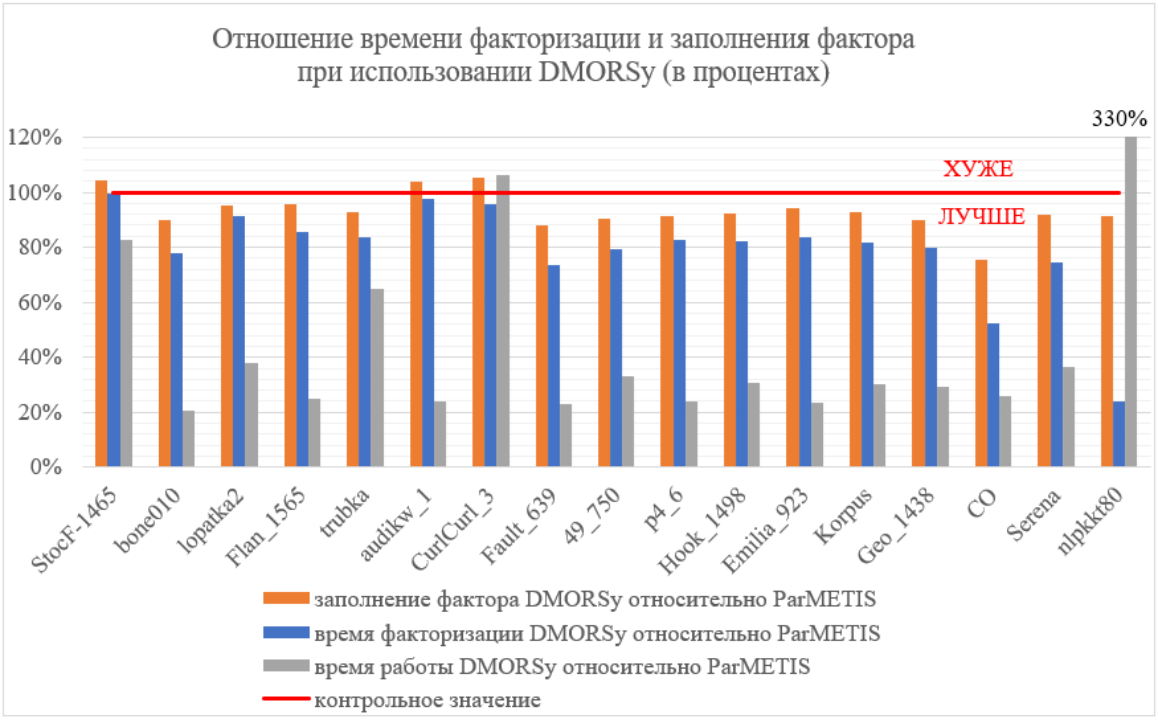

Рис. 9. Сокращение времени численной факторизации и заполнения фактора при использовании перестановок из DMORSy в конфигурации 2 вместо перестановок из ParMETIS, для матриц, на которых время работы решателя более 10 секунд. Опережению соответствуют столбцы высотой менее 100 %, отставанию — более 100 %

предложен новый гибридный MPI + OpenMP параллельный алгоритм метода вложенных сечений, в котором комбинируется параллелизм с использованием процессов и потоков.

Вычислительные эксперименты показали, что использование гибридной схемы распараллеливания в сравнении с реализацией для систем с распределенной памятью позволило сократить время нахождения перестановок и заполнение фактора матриц при работе на двух вычислительных узлах. В сравнении с пакетом ParMETIS [2], широко применяемым в приложениях, в частности, при расчетах в системах с распределенной памятью, для половины тестовых матриц были получены лучшие результаты как по времени работы, так и по заполнению фактора.

В работе приводятся результаты применения полученных перестановок для решения СЛАУ с разреженной матрицей с помощью открытого решателя MUMPS. Показано, что удается сократить общее время решения СЛАУ на большом числе тестовых матриц, в сравнении с использованием переупорядочивателя ParMETIS. Опережение получено как за счет меньшего времени переупорядочения, так и за счет меньшего заполнения фактора, что, в свою очередь, приводит к уменьшению времени численной факторизации. В дальнейшем можно выполнить интеграцию переупорядочивателя в решатель, чтобы иметь возможность задавать настройки параметров переупорядочения, необходимые при решении конкретной задачи, а также параллельно выполнять фазу анализа.

Автор выражает благодарность Меерову И. Б., Бартеневу Ю. Г. и Козинову Е. А. за ценные обсуждения и внимание к работе.

Список литературы

1. Karypis G. and Kumar V. ParMetis: Parallel graph partitioning and sparse matrix ordering library. Tech. Rep. TR 97-060, University of Minnesota, Department of Computer Science. 1997.

2. Chevalier C., Pellegrini F. PT-Scotch: A tool for efficient parallel graph ordering // Parallel Computing. 2008. Vol. 34, N 6. P. 318–331. DOI:10.1016/j.parco.2007.12.001

3. MUltifrontal Massively Parallel Solver (MUMPS). [Electron. Res.]: `http://mumps.enseeiht.fr/index.php?page=home` (дата обращения: 10.11.2021).

4. LaSalle D. and Karypis G. Efficient Nested Dissection for Multicore Architectures // Euro-Par 2015: Parallel Processing. 2015, Springer Berlin Heidelberg. P. 467–478. DOI:10.1007/978-3-662-48096-0_36.

5. OneMKL PARDISO — Parallel Direct Sparse Solver Interface. Developer Reference. [Electron. Res.]: `https://www.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/sparse-solver-routines/onemkl-pardiso-parallel-direct-sparse-solver-iface.html`.

6. Parallel Direct Sparse Solver for Clusters Interface. Developer Reference. [Electron. Res.]: `https://www.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/sparse-solver-routines/parallel-direct-sp-solver-for-clusters-iface.html`.

7. Pirova A., Meyerov I., Kozinov E., Lebedev S. PMORSy: parallel sparse matrix ordering software for fill-in minimization // Optimization Methods and Software. 2017. Vol. 32. N 2. P. 274–289. DOI: 10.1080/10556788.2016.1193177.

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

9. Пирова А. Ю. Гибридный MPI + OpenMP алгоритм переупорядочения симметричных разреженных матриц для систем с распределенной памятью // 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 Нижегородского госуниверситета, 2021. С. 268–273.

10. Suite Sparse Matrix Collection. [Electron. Res.]: `https://sparse.tamu.edu/` (дата обращения: 10.11.2021).

Пирова Анна Юрьевна — преподаватель кафедры математического обеспечения и суперкомпьютерных технологий института информационных технологий, математики и механики ННГУ им. Н. И. Лобачевского, e-mail: `anna.pirova@itmm.unn.ru`. Область научных интересов: алгоритмы на графах, параллельные алгоритмы, разреженная алгебра. Тел. 8-920-060-50-30.

Pirova Anna Yurievna — the lecturer of the department of Mathematical software and Supercomputing Technologies, Institute of Information Technologies, Mathematics and Mechanics, e-mail: `anna.pirova@itmm.unn.ru`. Area of scientific interests: graph algorithms, parallel algorithms, sparse linear algebra. Phone 8-920-060-50-30.

Дата поступления — 01.02.2022