Оптимизация работы MPI-программ с учётом особенностей топологии кластеров, использующих коммуникационную сеть Ангара

М.Р. Халилов[2], А.В. Тимофеев[1, 2]

Объединенный Институт Высоких температур Российской Академии Наук[1],
Национальный Исследовательский Университет Высшая Школа Экономики[2]

В данной работе предложена процедура оптимизации работы параллельных MPI-программ на вычислительных кластерах, использующих коммуникационную сеть Ангара, и исследована эффективность предложенной оптимизации. На основе анализа работы параллельной программы составляется информационный граф программы, который используется эвристическим алгоритмом для эффективного распределения её процессов по процессорным ядрам с целью минимизации суммарного времени выполнения обменов между ветвями MPI-программы. Приведена схема реализации процедуры для кластеров, использующих коммуникационную сеть Ангара. Показано уменьшение времени выполнения MPI-программ при использовании алгоритма оптимального отображения. Исследована зависимость времени выполнения MPI-программ от параметров кластера.

Ключевые слова: MPI, Ангара, отображение программ, параллельное программирование, вычислительные кластеры, топология кластеров

1. Введение

Современные вычислительные кластеры (ВК) имеют структуру, насчитывающую большое количество вычислительных узлов, соединение которых описывается сложной топологией. Параллельные алгоритмы для таких кластеров разрабатываются в рамках модели передачи сообщений. Для этого наиболее часто используются библиотеки, основанные на стандарте MPI [1].

Для кластеров, использующих коммуникационную сеть Ангара [2, 3], также имеется специальная реализация MPI, представляющая собой адаптированную для данной сети версию MPICH. В иерархической среде таких кластеров можно выделить два уровня: уровень межузлового обмена, непосредственно задействующий сеть Ангара, и уровень обмена внутри каждого узла между ядрами процессора, использующими общую память. Тем не менее, на данный момент на ВК с сетью Ангара MPI-программы запускаются без учёта особенностей их физической топологии, её иерархии и пропускной способности каналов передачи данных, поскольку текущая реализация MPI на таких кластерах предусматривает линейное отображение MPI-процессов на доступные узлы/ядра процессоров.

Данное обстоятельство приводит к увеличению времени выполнения алгоритмов и большим накладным расходам, по сравнению с запуском параллельных программ с использованием оптимального отображения процессов. Исходя из этого, вопрос, связанный с реализацией программных средств для оптимального отображения MPI-программ на ВК с сетью Ангара, становится особенно актуальным.

Методология и алгоритмы эффективного запуска MPI-программ с учётом особенностей ВК на данный момент освещены во множестве работ. Так, в статье [4] рассмотрены пространственно-вычислительные системы (ВС), для которых были разработаны алгоритмы отображения параллельных MPI-программ по заданному информационному графу, учитывая несколько уровней иерархии ВС. Для приближенного решения задачи используется алгоритм Кернигана-Лина, реализованный в библиотеке Scotch [8], для разбиения информационного графа программы.

Исследования в работах [5,9] направлены на создание алгоритмов отображения на кластеры и суперкомпьютеры, построенные на базе InfiniBand с топологией толстого дерева (fat-tree). Большое внимание уделяется разработке программных средств для анализа топологии. В публикации [5] для изучения топологии системы предложена оригинальная реализация алгоритма метода присоединения соседей, заимствованного из биоинформатики.

Алгоритмы отображения параллельных MPI-программ на системы с тороидальной топологией предложены в работах [10, 11]. Авторы работы [6] рассматривают комплексные алгоритмы отображения параллельных программ на топологию, применимые как для сетей с fat-tree топологией, так и для сетей с тороидальной топологией.

Попытки создания средств оптимального отображения MPI-программ были предприняты в рамках разработки OpenMPI. Данная библиотека MPI включает в себя пакет [7], который реализует отображение MPI-программ с учётом физической топологии ВК и его иерархии. Однако использование этого пакета возможно только в ВК, использующих InfiniBand или Ethernet в качестве коммуникационной сети.

Применение известных методов эффективного отображения MPI-программ возможно и для ВК, использующих коммуникационную сеть Ангара. В рамках данной работы предлагается использовать обобщенный подход, применяемый в работах [4, 6] и задействующий эвристические алгоритмы разбиения информационного графа MPI-программ с учётом иерархической структуры ВК.

2. Схема осуществления оптимизации отображения MPI-программ на физическую топологию

Для оптимизации отображения на первом этапе выполняется анализ информационного графа обмена сообщений между процессами данной MPI-программы. Затем выполняется разбиение информационного графа таким образом, что процессы, между которыми происходят наиболее интенсивные обмены, привязываются к узлам/процессорам с наиболее высокой пропускной способностью. Для реализации данного алгоритма была разработана программная библиотека, функциональная схема которой разделена на 4 этапа и приводится на рис.1.

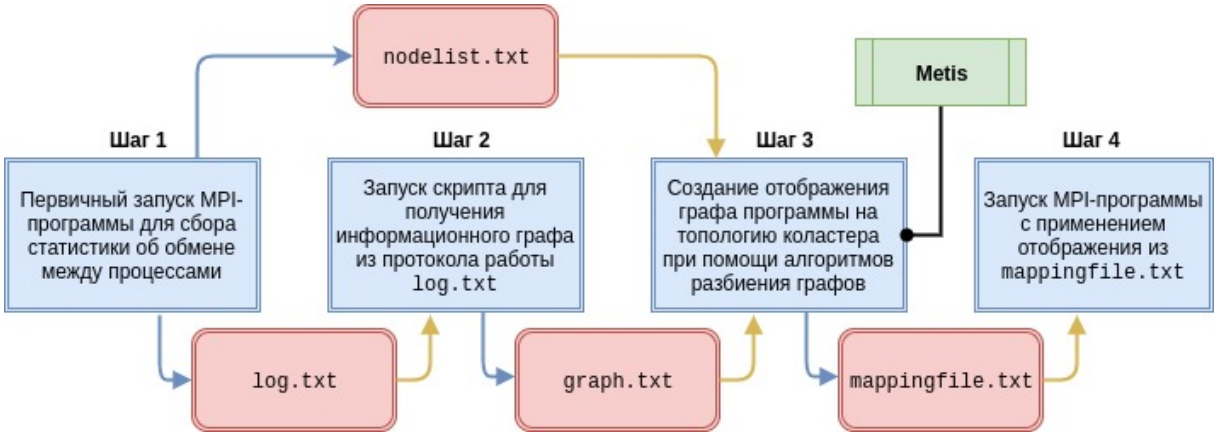

Рис. 1. Функциональная схема работы библиотеки

Шаг 1 Для получения информации об обменах между процессами MPI-программы, их интенсивности и дальнейшего построения на основе данных информационного графа используется система сбора статистики amfch_statistics, встроенная в реализацию MPI Ангара. Для активации сбора статистики в формате матрицы коммуникационного паттерна устанавливается переменная среды AMFCH_STATISTICS_PRINT_MODE=2. Элементы

матрицы коммуникационного паттерна - количество переданных 64-битных слов между источником и адресатам, строки - процессы-источники, столбцы - процессы-адресаты. Для того чтобы система сбора статистики amfch_statistics работала, на этапе установки MPI Ангара администратором кластера должен быть определён макрос AMFCH_ENABLE_STATISTICS=1.

Профилирование MPI-программы выполняется для получения информации об обменах между процессами. На этом этапе пользователь при запуске программы через mpirun выделяет ресурсы, которые данная MPI-программа сможет использовать и для которых будет выполняться оптимизация её работы. Данные о количестве выделенных узлов и процессоров для данной MPI-программы сохраняются в файле nodelist.txt. После первичного запуска вся информация с потока вывода сохраняется в файле log.txt, включающем в себя в т.ч. матрицу коммуникационного паттерна.

Шаг 2 При помощи скрипта, осуществляющего парсинг элементов матрицы, матрица коммуникационного паттерна из файла log.txt преобразуется в формат представления графов CSR (Compressed Sparse Row) и сохраняется в файл graph.txt. Этот файл описывает информационный граф обменов между процессами запускаемой пользователем MPI-программы.

Шаг 3 Информационный граф в файле graph.txt представляет собой неориентированный взвешенный граф. Вершины графа - MPI-процессы, веса заданные на рёбрах - количество 64-битных слов переданных между процессами.

Идея метода оптимизации отображения параллельной программы заключается в разбиении информационного графа программы на непересекающиеся подмножества интенсивно обменивающихся процессов и привязке этих подмножеств к узлам/процессорам, соединённым наиболее быстрыми каналами связи. Разбиение выполняется для минимизации суммы рёбер, соединяющих разные подмножества разбиения. Разбиение рекурсивно выполняется сначала для уровня описывающего обмены между вычислительными узлами, а затем для уровня описывающего обмены внутри каждого из узлов в случае, если внутри узла установлено несколько процессоров. Целью такого разбиения является минимизация времени выполнения программы. Как показано в исследовании [6] задача оптимального отображения процессов MPI-программы является NP-полной, так как её можно свести к задаче разбиения графов. Для её решения целесообразно использовать эвристические алгоритмы дающие решения, близкие к оптимальным.

Организация разбиения информационного графа из файла graph.txt в соответствии со списком узлов/процессоров из файла nodelist.txt осуществляется при помощи API библиотеки разбиения графов METIS [12]. В METIS реализована многоуровневая схема Karypis-Kumar (KK) [12] разбиения графов, в основе которой лежит алгоритм Кернигана-Лина [13]. Выходная информация с отображением каждого из процессов MPI-программы на отдельный процессор, принадлежащий какому-либо узлу, сохраняется в файле mappingfile.txt.

Вычислительная сложность разбиения графа с использованием схемы KK составляет $T = O(|E|log_2(z))$, где E - количество рёбер в графе, а z -число подмножеств разбиения. Для получения субоптимального отображения необходимо $1 + N$ разных разбиений, где N равно числу узлов с ненулевым количеством процессоров.

Шаг 4 Файл mappingfile содержит фактическую информацию об отображении процессов MPI-программы на физическую топологию. Каждому процессу MPI-программы поставлен в соответствие узел/процессор, к которому данный процесс будет привязан при запуске программы. Данный файл используется SLURM с опцией -m arbitrary при запуске программы, для которой проводилась оптимизация отображения на физическую топологию.

3. Аппаратное и программное обеспечение для моделирования работы алгоритмов

В качестве платформы для тестирования работы алгоритмов отображения параллельных MPI-программ использовались вычислительные кластеры "Ангара-К1" НИЦЭВТ и "Десмос" суперкомпьютерного центра ОИВТ РАН.

Кластер "Ангара-К1" состоит из 36 вычислительных узлов двух типов. Топология сети кластера - 3D-тор с размерностью 3 x 3 x 4. Каждый А-узел (24 шт.) оборудован двумя 6-ядерными Intel Xeon CPU E5-2630, 2.30 GHz, 64 GB ОЗУ, сетевым адаптером сети Ангара на базе СБИС ЕС8430. В-узлы (12 шт.) оснащены 8-ядерными Intel Xeon CPU E5-2660, 2.20 GHz, 64 GB RAM и сетевыми адаптерами, аналогичными А-узлам.

Кластер "Десмос"оборудован 32 вычислительными узлами, соединёнными коммуникационной сетью Ангара с топологией 4D-тора размерностью 4 x 2 x 2 x 2. На каждом вычислительном узле установлены 6-ядерный процессор Intel Xeon E5-1650, 3.5 GHz, 8 GB ОЗУ и сетевой адаптер сети Ангара на базе СБИС ЕС8430.

Запуск параллельных MPI-программ осуществлялся с использованием менеджера управления ресурсами SLURM и MPI Ангара. Данная реализация стандарта MPI представляет собой MPICH3, адаптированный для сети Ангара. Программный модуль angara_devinfo, входящий в состав данной реализации, позволяет получить информацию о текущей топологии кластера и узлах, доступных на данный момент.

Схема осуществления отображения MPI-программ на физическую топологию, описанная в разделе 2, была реализована в виде единой программной библиотеки. Данная библиотека написана на языках C++ и C и использует API METIS для разбиения информационного графа MPI-программы. Тестирование библиотеки проводилось при помощи набора тестов производительности NAS Parallel Benchmarks 3 (NPB).

4. Результаты оптимизации работы MPI-программ

Для оценки эффективности работы разработанной библиотеки MPI-программы выполнялись в нескольких режимах:

a) запуск MPI-программ осуществлялся без использования оптимизации отображения (процессы распределялись стандартными средствами SLURM динамически);

b) запуск программ с использованием разработанной библиотеки оптимизации отображения, при этом для работы программ выделялись те же ресурсы, что и на этапе профилирования (информация об этих ресурсах хранится в файле nodelist.txt);

c) запуск параллельных программ аналогично предыдущему режиму, но оптимизация отображения осуществляется для всех доступных на вычислительном кластере узлов.

В ходе экспериментов измерялось суммарное время выполнения тестовых параллельных MPI-программ из пакета NPB. На рис. 2 и рис. 3 приведены результаты, показанные тестами LU и SP (рассматривались задачи класса сложности "C").

Для теста LU при запуске в режимах b) и c) (рис.2) с 6 и 12 MPI-процессами уменьшение времени выполнения по сравнению с режимом a) составило менее 1%, а с ростом количества процессов выигрыш при использовании данных режимов также растёт. В режиме b) для 24 процессов улучшение времени выполнения достигло 7%, для 42 процессов уже 14%. Использование режима c) без ограничения количества узлов ВК, доступных для отображения, даёт наибольший прирост производительности: для 24 процессов уменьшение времени выполнения в сравнении с режимом a) составило 30%, для 42 MPI-процессов - 35%. Наибольший прирост в производительности был достигнут при использовании 120 MPI-процессов для режимов b) и c) - уменьшение времени выполнения тестов составило 25% и 51%, соответственно. Больший прирост эффективности работы MPI-программы при использовании

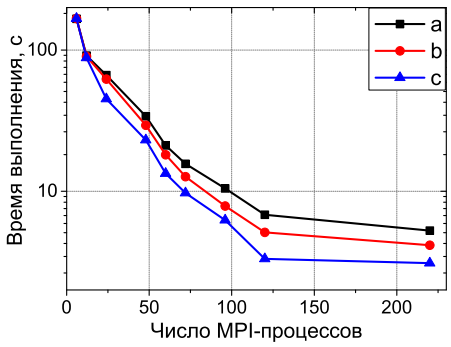

Рис. 2. Зависимость времени выполнения алгоритма LU NAS Parallel Benchmarks: a) без оптимизации, b) оптимизация с использованием тех же ресурсов, что и во время профилирования, c) оптимизация с использованием всех доступных ресурсов ВК

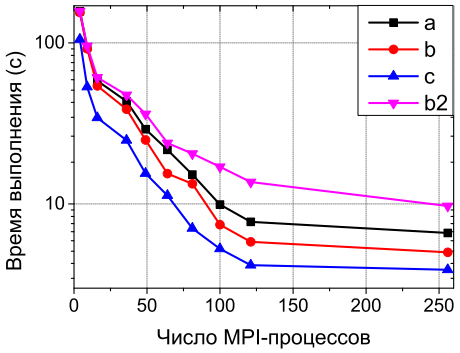

Рис. 3. Зависимость времени выполнения алгоритма SP NAS Parallel Benchmarks: a) без оптимизации, b) оптимизация с использованием тех же ресурсов, что и во время профилирования, b2) ухудшение отображения на базе тех же ресурсов, что и во время профилирования, c) оптимизация с использованием всех доступных ресурсов ВК

всех доступных узлов по сравнению с режимом с ограничениями на количество выделенных для задачи узлов достигается за счёт того, что данный режим позволяет эффективнее использовать коллективные операции передачи данных между MPI-процессами.

Тест SP при использовании режима b) даёт соизмеримые результаты (рис.3) с тестом LU в этом же режиме. Значительное уменьшение времени выполнения (на 32%) теста SP в режиме c) при использовании 4 MPI-процессов объясняется большей (по сравнению с тестом LU) асимметричностью распределения данных между процессами и, следовательно, высокой дифференцируемостью обменов между ними, поэтому суммарное время выполнения сильно зависит от того, насколько оптимально процессы распределены между узлами. Уменьшение времени выполнения теста SP в режиме c) относительно режима a) с 16 процессами - 44%, с 36 процессами - 48%, а с 81 процессом - 54% (рис. 4). Наблюдаемая зависимость показывает, что эффективность отображения улучшается с увеличением количества процессов.

Важно отметить, что оптимизация отображения не всегда даёт оптимальные результаты, т.к. разбиение, полученное при использовании рекурсивной схемы KK, не всегда соответствует лучшему для данной задачи разбиению [14]. Пример такого неудачного отображения для режима b) также приводится на рис.3, а его сравнение с оптимизированным отображением приводится на рис.4.

5. Заключение

В рамках данного исследования была разработана библиотека оптимизации отображения параллельных MPI-программ на вычислительные кластеры, использующие коммуникационную сеть Ангара, с учётом особенностей их физической топологии и иерархической структуры.

Анализ результатов работы библиотеки на тестах SP и LU из набора тестов производительности NAS Parallel Benchmarks и MPI Ангара показал уменьшение времени выполнения на тесте LU от 7% до 51% по сравнению со стандартным распределением процессов при использовании от 24 до 120 MPI-процессов, соответственно. На тесте SP удалось достичь до 54% уменьшения времени выполнения при использовании 81 MPI-процессов. Показана зависимость эффективности построения отображения от количества доступных для парал-

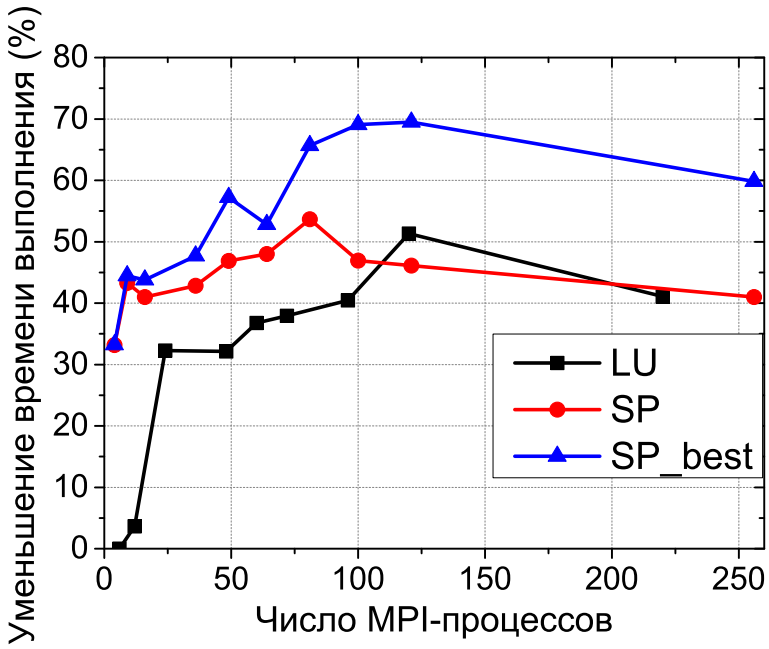

Рис. 4. Зависимость относительного уменьшения времени выполнения алгоритмов LU и SP NAS Parallel Benchmarks при оптимизации отображения от числа процессов. SP_best показывает уменьшение времени работы программы SP при оптимизированном отображении процессов по сравнению с временем работы SP на ухудшенном отображении процессов.

лельной программы MPI-процессов и вычислительных узлов кластера.

Авторы выражают признательность ОАО НИЦЭВТ за предоставленный доступ к вычислительному кластеру "Ангара-К1" и суперкомпьютерный центр ОИВТ РАН за доступ к кластеру "Десмос". Авторы благодарят А.А. Агаркова, А.С. Семёнова и В.В.Стегайлова за продуктивное обсуждение.

Исследование финансировалось в рамках государственной поддержки ведущих университетов Российской Федерации «5-100».

Литература

1. MPI Forum. MPI: A Message-Passing Interface Standard Version 3.1.-[Электронный ресурс] // URL: mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf

2. А.А. Агарков, Т.Ф. Исмагилов, Д.В. Макагон, А.С. Семенов, А.С. Симонов. Результаты оценочного тестирования отечественной высокоскоростной коммуникационной сети Ангара // Суперкомпьютерные дни в России: Труды международной конференции (26-27 сентября 2016 г., г. Москва). – М.: Изд-во МГУ, 2016. – С. 626-639.

3. Симонов А.С., Макагон Д.В., Жабин И.А., Щербак А.Н., Сыромятников Е.Л., Поляков Д.А. Первое поколение высокоскоростной коммуникационной сети «Ангара» // Наукоемкие технологии. — 2014. — Т. 15, №1. — С. 21-28.

4. Пазников А. А., Курносов М. Г., Куприянов М.С. Многоуровневые алгоритмы отображения параллельных MPI-программ на вычислительные кластеры. // Проблемы информатики. 2015. Т1, С. 4-17

5. Subramoni H. et al. Design of a scalable InfiniBand topology service to enable network-topology-aware placement of processes. // High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for. – IEEE, 2012. – C. 1-12.

6. Hoefler T., Snir M. Generic topology mapping strategies for large-scale parallel architectures. // Proceedings of the international conference on Supercomputing. – ACM, 2011. – C. 75-84.

7. Broquedis F. et al. hwloc: A generic framework for managing hardware affinities in HPC applications. // Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on. – IEEE, 2010. – C. 180-186.

8. Chevalier C., Pellegrini F. PT-Scotch: A tool for efficient parallel graph ordering. // Parallel computing. – 2008. – Т. 34. – №. 6. – C. 318-331.

9. Abhinav Bhatele, Automating Topology Aware Mapping for Supercomputers. // Ph.D. Dissertation. University of Illinois at Urbana-Champaign, Champaign, IL, USA. Advisor(s) Laxmikant V. Kale., 2010. AAI3425400.

10. Yu H. et al. Topology mapping for Blue Gene/L supercomputer. // Proceedings of the 2006 ACM/IEEE conference on Supercomputing. – ACM, 2006. – C. 116.

11. Balaji P. et al. Mapping communication layouts to network hardware characteristics on massive-scale blue gene systems. // Computer Science-Research and Development. – 2011. – Т. 26. – №. 3-4. – C. 247-256.

12. Karypis, G. and Kumar, V. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. // SIAM Journal on Scientific Computing, 1999, Vol. 20, No. 1, pp. 359 - 392

13. B. W. Kernighan, S. Lin. An efficient heuristic procedure for partitioning graphs. // The Bell System Technical Journal, 1970.

14. H. D. Simon and S.-H. Teng. How good is recursive bisection? // SIAM J. Sci. Comput., September 1997, Vol. 18, pp. 1436–1445.

Topology-aware placement of MPI-programs processes for clusters that use a communication network Angara

M.R. Khalilov[1], A.V. Timofeev[2,1]

National Research University Higher School of Economics, Myasnitskaya 20, Moscow 101000, Russia[1], Joint Institute for High Temperatures of the Russian Academy of Sciences, Izhorskaya 20 bld.2, Moscow 125412, Russia[2]

In this paper we consider a heuristic algorithm to optimize topology-aware mapping of MPI-programs processes on the physical topology to computer clusters using a communication network Angara. The algorithm optimally distributes its processes for processor cores to minimize the total execution time of exchanges between the branches of MPI-program. We design of its implementation for clusters using a communications network Angara. Experimental results show performance improvement of execution time of MPI-programs using topology-aware placement of processes. We demonstrate analysis the dependence of the the MPI-program execution time on cluster parameters and task parameters.

Keywords: MPI, Angara network, process mapping, parallel programming, computer clusters, cluster topology

References

1. MPI Forum. MPI: A Message-Passing Interface Standard Version 3.1.-[Electronic resourse] // URL: mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf
2. A.A. Agarkov, T.F. Ismagilov, D.V. Makagon, A.S. Semyonov, A.S. Simonov. Rezultaty otsenochnogo testirovanija otechestvennoj vysokoskorostnoj kommunikatsyonnoj seti Angara // Russian Supercomputing Days: Proceedings of the International Scientific Conference (26 - September,27, 2016, Moscow). – Moscow, MSU Publishing, 2016. – P. 626-639.
3. A.S. Simonov, D.V. Makagon, Zhabin I.A., Sherbak A.N., Syromiatnikov E.L., Polyakov D.A. Pervoje pokolenije vysokoskorostnoj kommunikatsyonnoy seti «Angara» // Naukoemkie tekhnologii. — 2014. — Vol. 15, NO. 1. — P. 21-28.
4. Paznikov A. A., Kurnosov M. G., Kuprijanov M.S. Mnogourovneviye algoritmy otobrajenija parallelnikh MPI-programm na vicheslitelnije clastery. // Problemy informatiki. 2015. Vol 1, P. 4-17
5. Subramoni H. et al. Design of a scalable InfiniBand topology service to enable network-topology-aware placement of processes. // High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for. – IEEE, 2012. – C. 1-12.
6. Hoefler T., Snir M. Generic topology mapping strategies for large-scale parallel architectures. // Proceedings of the international conference on Supercomputing. – ACM, 2011. – C. 75-84.
7. Broquedis F. et al. hwloc: A generic framework for managing hardware affinities in HPC applications. // Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on. – IEEE, 2010. – C. 180-186.
8. Chevalier C., Pellegrini F. PT-Scotch: A tool for efficient parallel graph ordering. // Parallel computing. – 2008. – Т. 34. – №. 6. – C. 318-331.

9. Abhinav Bhatele, Automating Topology Aware Mapping for Supercomputers. // Ph.D. Dissertation. University of Illinois at Urbana-Champaign, Champaign, IL, USA. Advisor(s) Laxmikant V. Kale., 2010. AAI3425400.

10. Yu H. et al. Topology mapping for Blue Gene/L supercomputer. // Proceedings of the 2006 ACM/IEEE conference on Supercomputing. – ACM, 2006. – С. 116.

11. Balaji P. et al. Mapping communication layouts to network hardware characteristics on massive-scale blue gene systems. // Computer Science-Research and Development. – 2011. – Т. 26. – №. 3-4. – С. 247-256.

12. Karypis, G. and Kumar, V. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. // SIAM Journal on Scientific Computing, 1999, Vol. 20, No. 1, pp. 359 - 392

13. B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. // The Bell System Technical Journal, 1970.

14. H. D. Simon and S.-H. Teng. How good is recursive bisection? // SIAM J. Sci. Comput., September 1997, Vol. 18, pp. 1436–1445.