

## Параллельное и распределенное программирование, грид-технологии, программирование на графических процессорах

УДК 004.021

**Рыбаков А.А.**

Межведомственный суперкомпьютерный центр Российской академии наук — филиал Федерального государственного учреждения «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук», г. Москва, Россия

### ПАРТИЦИРОВАНИЕ ГРАФА СМЕЖНОСТИ БЛОЧНО-СТРУКТУРИРОВАННОЙ СЕТКИ

#### Аннотация

*В статье рассматривается структура графа смежности блочно-структурированной расчетной сетки. Такие сетки часто используются при численном решении задач газовой динамики. Для эффективного проведения расчетов на суперкомпьютере требуется уметь равномерно распределять вычислительную нагрузку между вычислительными узлами. Данная задача решается с помощью партиципирования графа смежности блочно-структурированной сетки.*

#### Ключевые слова

*Блочно-структурированная сетка; граф смежности блочно-структурированной сетки; партиципирование графа.*

**Rybakov A.A.**

Joint Supercomputer Center of the Russian Academy of Sciences - branch of Scientific Research Institute of System Analysis of the Russian Academy of Sciences, Moscow, Russia

### BLOCK-STRUCTURED GRID ADJACENCY GRAPH PARTITIONING

#### Abstract

*In the article block-structured calculation grid adjacency graph structure is considered. Such grids are often used in the numerical solution of gas dynamics problems. For effective calculations on supercomputer it is needed to be able to distribute computational load between compute nodes. The block-structured grid adjacency graph partitioning is focused on this task solving.*

#### Keywords

*Block-structured grid; block-structured grid adjacency graph; graph partitioning.*

#### Введение

Численное решение задач газовой динамики, как правило, связано с большим объемом вычислений [1]. Для математического моделирования высокоскоростных турбулентных течений требуется использование больших расчетных сеток, состоящих их десятков и сотен миллионов ячеек. Для обработки таких объемов данных часто бывает необходимо прибегать к использованию суперкомпьютеров. При этом остро встает проблема как повышения скорости работы исполняемого кода на вычислителе, так и проблема равномерной загрузки вычислительных

узлов суперкомпьютера.

Одним из наиболее распространенных типов расчетных сеток, используемых в математическом моделировании задач газовой динамики, является блочно-структурированная сетка [2]. Сетка данного типа состоит из отдельных блоков, каждый из которых представляет собой трехмерный массив ячеек, доступ к которым осуществляется по индексам. Блочноструктурированные сетки более выгодны с точки зрения скорости вычислений и объема используемой памяти, однако они гораздо сложнее в плане построения по сравнению с

неструктурированными сетками.

При распределении вычислений, проводимых на блочно-структурированной сетке, между узлами суперкомпьютерного кластера каждый блок сетки распределяется на конкретный вычислительный узел. Обмены данными между соседними блоками, находящимися в разных вычислительных узлах, осуществляются через MPI [3].

Равномерному распределению вычислительной нагрузки между узлами мешают два фактора. Во-первых, это разные размеры блоков сетки, в частности наличие ярко выраженных крупных блоков. Данная проблема решается с помощью алгоритмов дробления сетки. Вторым фактором является большое количество межпроцессных обменов данными, которые могут серьезно затормозить вычисления. Алгоритм, описанный в данной статье, направлен на такое распределение вычислительной нагрузки, которое снижает общий объем межпроцессных обменов данными.

### Граф смежности расчетной сетки

Приведем сначала общее описание блочно-структурированной сетки [4]. Основным объектом блочно-структурированной сетки является блок. Блок состоит из упорядоченного трехмерного массива ячеек, содержащих газодинамические параметры, ассоциированные с центрами масс ячеек. С каждым блоком ассоциирована криволинейная система координат, которая задается тремя линиями координат  $(I, J, K)$ , каждая из которых связывает пары противоположных граней блока.

Объектом, описывающим соприкосновение двух соседних блоков, является интерфейс. Интерфейс является однонаправленным, он сообщает, что у данного блока конкретная прямоугольная часть границы соприкасается с другим блоком. Чтобы определить, с какой частью другого блока граничит рассматриваемый блок, нужно рассмотреть смежный ему интерфейс. Таким образом, полная информация о касании двух блоков описывается парой смежных интерфейсов (рис. 1).

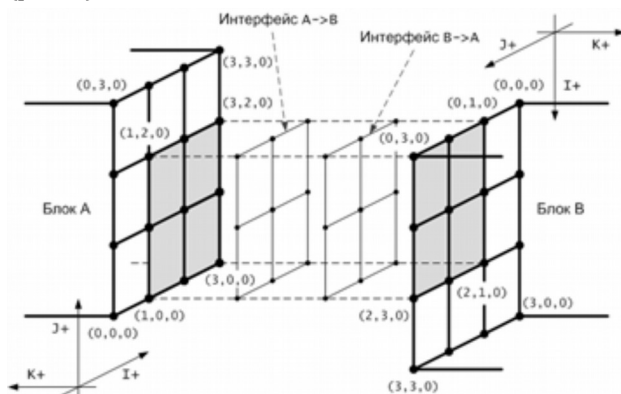


Рис. 1. Задание касания двух блоков парой смежных интерфейсов

Во время проведения расчетов различные блоки сетки обрабатываются независимо друг от друга. Однако некоторые ячейки обрабатываемого блока должны обращаться за данными к ячейкам соседних блоков, с которыми этот блок граничит через интерфейсы. Если два соседних блока обрабатываются на разных узлах суперкомпьютера, то для обмена данными между ними можно использовать MPI-обмены. Если ячейка в процессе проведения вычислений обращается за данными к ячейкам соседних блоков, то будем называть ее интерфейсной ячейкой. Если к тому же она обращается за данными на другой вычислительный узел, то будем называть ее кросс-ячейкой, а соответствующий интерфейс – кросс-интерфейсом.

Пусть нам задана блочно-структурированная сетка, и известно количество вычислительных узлов гомогенной вычислительной системы, то есть системы, состоящей их идентичных вычислительных узлов, для которой нужно организовать расчеты на данной сетке. Сконструируем для данной сетки граф смежности ее блоков (GBAG – grid blocks adjacency graph), в котором вершинами будут блоки сетки, а ребрами – касания соседних блоков, то есть пары интерфейсов. Данный граф является взвешенным. В качестве веса вершины GBAG будем брать количество ячеек соответствующего блока. В качестве веса ребра GBAG положим площадь касания между соответствующими блоками, то есть количество ячеек одного блока, граничащих по грани с ячейками второго блока. Также с каждой вершиной GBAG будем ассоциировать трехмерную точку, являющуюся барицентром всех узлов соответствующего блока.

Задача эффективного распределения вычислительной нагрузки по обработке блочно-структурированной сетки между узлами суперкомпьютера с минимизацией межпроцессных обменов данными сводится к задаче партицирования соответствующего GBAG с условием минимизации суммы весов ребер, связывающих разные партиции. Такие ребра будем называть кросс-ребрами.

Формальная постановка задачи выглядит следующим образом. Пусть дан граф смежности блочно-структурированной сетки  $G = G(V, E)$ , где  $V$  – множество его вершин,  $E$  – множество его ребер. На множествах вершин и ребер данного графа заданы весовые функции:  $w_V: V \rightarrow N$ ,  $w_E: E \rightarrow N$ . Пусть задано множество партиций  $P$ , где  $|P|$  – общее их количество. Распределение блоков расчетной сетки по партициям задается функцией  $\gamma: V \rightarrow P$ . Общий вес партиции  $p \in P$  определяется через сумму весов, входящих в нее вершин:

$$w_p(p) = \sum_{\substack{v \in V \\ \gamma(v)=p}} w_v(v).$$

Равномерность распределения вычислительной нагрузки на узлы суперкомпьютера означает минимизацию веса наиболее тяжелой партии, или минимизацию отклонения веса наиболее тяжелой партии от среднего значения

$$w_p^{\text{sum}} = \sum_{p \in P} w_p(p),$$

$$w_p^{\text{avg}} = \frac{w_p^{\text{sum}}}{|P|},$$

$$w_p^{\text{max}} = \max_{p \in P} w_p(p),$$

$$w_p^{\text{dev}} = \left( \frac{w_p^{\text{max}}}{w_p^{\text{avg}}} - 1 \right) \cdot 100\%.$$

Второе условие эффективности партицирования касается объема межпроцессных обменов. Пусть у нас задана функция  $\gamma$  разделения множества блоков на партии. Определим множество кросс-ребер графа GBAG для одной конкретно взятой партии  $p \in P$

$$E^{\text{cross}}(p) = \{e = \{v, u\} \in E \mid \gamma(v) = p, \gamma(u) \neq p\}.$$

Аналогично определим множество всех внутренних ребер партии  $p \in P$

$$E^{\text{inner}}(p) = \{e = \{v, u\} \in E \mid \gamma(v) = p, \gamma(u) = p\}.$$

Под весом множества ребер будем понимать сумму весов всех ребер, входящих в это множество

$$w_E(M) = \sum_{e \in M} w_E(e).$$

Тогда под минимизацией количества межпроцессных обменов будем понимать следующую величину, отражающую отношение максимального объема межпроцессных обменов для партии к среднему значению всех обменов (обозначим данную величину через  $\chi$ )

$$E(p) = E^{\text{cross}}(p) + E^{\text{inner}}(p),$$

$$\chi = \max_{p \in P} w_E(E^{\text{cross}}(p)) \cdot \left( \frac{\sum_{p \in P} w_E(E(p))}{|P|} \right)^{-1} \cdot 100\%.$$

Далее рассмотрим два подхода к распределению блоков по вычислительным узлам. В первом случае не будем учитывать кросс-ребра вообще (распределение блоков без учета межпроцессного обмена). Во втором случае будем изначально строить партии таким образом, чтобы как можно большее число тяжелых ребер попадали внутрь партии, превращаясь таким образом из кросс-ребер во внутренние ребра партии.

### Распределение блоков сетки без учета межпроцессного обмена

Опишем реализацию простого жадного алгоритма распределения блоков сетки по вычислительным узлам [5]. Для него нужно знать только веса блоков, размеры данных, участвующих в межпроцессных обменах, игнорируются.

Первым пунктом действий занесем все вершины графа в массив необработанных вершин.

Будем обрабатывать данный массив с порядке убывания весов вершин. То есть данный массив нужно отсортировать по весам.

Вторым пунктом алгоритма является обработка очередной вершины из массива необработанных вершин. Если данный массив пуст, то алгоритм заканчивает работу. Если массив не пуст, то возьмем наиболее тяжелую вершину. После этого найдем партию, имеющую наименьший текущий вес, и отнесем рассматриваемую вершину в данную партию. Завершающим действием шага является удаление данной вершины из массива необработанных вершин и возврат в начало пункта.

Данный алгоритм никак не учитывает ни расположение блоков сетки, ни веса ребер GBAG, так что ожидать минимизации объема обменов данными между разными вычислительными узлами не приходится. Будем обозначать данный алгоритм UG (от uniform greedy). Распределение блоков по партициям, получаемое в результате работы данного алгоритма, близко к оптимальному в смысле весов партий. Это справедливо для достаточно большого количества блоков в сетке и отсутствия ярко выраженных крупных блоков.

### Алгоритм партицирования графа смежности

В данном разделе рассматривается алгоритм, который основан на последовательном распределении вершин GBAG по партициям с учетом их пространственного расположения. При этом в начале работы алгоритма каждой партии приписывается изначально определяемая базовая точка (base point), вокруг которой в дальнейшем и строится партия. Базовые точки выбираются таким образом, чтобы они находились в области сетки, и минимальное расстояние между парами этих точек было максимально. Другими словами, базовые точки должны как можно более равномерно заполнять область сетки. Для получения базовых точек можно сначала получить множество случайных точек в области сетки, а затем с помощью последовательных приближений скорректировать их положение для достижения равномерности заполнения области расчетной сетки. Данный алгоритм будем обозначать RVP (random volume points).

Опишем алгоритм RVP распределения вершин по партициям. Сначала введем понятие окрестности партии. Назовем окрестностью партии множество всех не входящих в нее (и ни в какую другую партию) вершин графа, соединенных ребром хотя бы с одной вершиной из этой партии. На каждом шаге работы алгоритма будем выбирать самую легкую партию и добавлять к ней новую еще никуда не распределенную вершину. При этом приоритет в выборе новых вершин для расширения

(пропагации) партии будем отдавать вершинам из ее окрестности. При этом возможны три варианта. В первом варианте вообще нет вершин, которые можно добавить в партию, в этом случае алгоритм заканчивает работу. Во втором случае окрестность партии пуста, тогда нужно добавить вершину, ближайшую к базовой точке партии. И наконец в третьем случае существует непустая окрестность партии и вершину для пропагации нужно выбрать из нее.

алгоритмов использовалась блочно-структурированная сетка, содержащая  $10^8$  ячеек, подготовленная для проведения расчетов на суперкомпьютере, GBAG которой содержит 1000 вершин и более 6000 ребер. Данная сетка подготавливалась с помощью дробления блоков для обработке на суперкомпьютере МВС-10П, находящемся в МСЦ РАН [6].

Первым показателем, по которому будем

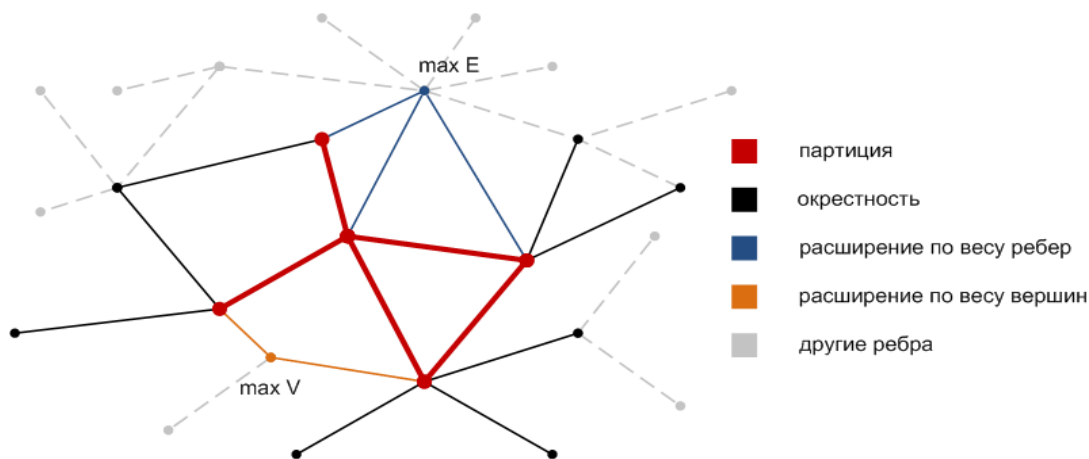


Рис. 2. Пропагация партии в алгоритме RVP

Обозначим текущую рассматриваемую партию через  $p$ , а ее окрестность через  $\delta(p)$ . Для каждой вершины  $v \in \delta(p)$  определим показатель предпочтения  $q(v)$  для добавления в партию

$$q(v) = \alpha (w_V(v))^{1/3} + (1 - \alpha) \left( \sum_{\substack{e=(v,u) \in E \\ \gamma(u)=p}} w_E(e) \right)^{1/2}.$$

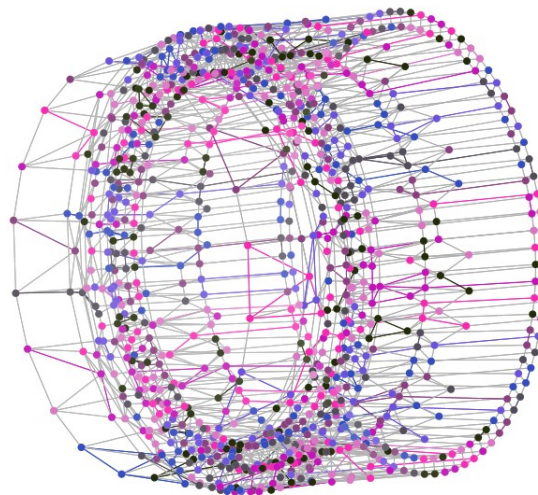
Для добавления в партию выбирается вершина с наибольшим значением показателя предпочтения. В данной формуле  $\alpha$  представляет собой параметр от 0 до 1. Если  $\alpha=1$ , то критерий вырождается в выбор самой тяжелой вершины из окрестности. Если  $\alpha=0$ , то выбирается такая вершина, при добавлении которой максимизируется сумма весов ребер, которые станут внутренними ребрами текущей партии (рис. 2). Таким образом, при значениях  $\alpha$  близких к единице достигается лучшая равномерность распределения весов вершин по партиям. При значениях  $\alpha$  близких к нулю уменьшается объем данных, участвующих в межпроцессном обмене (большее количество кросс-ребер уничтожается, так как они становятся внутренними ребрами своих партий).

Так как в большинстве случаев на каждом шаге пропагации к партии добавляется вершина из окрестности данной партии, то можно говорить о пропагации партии по ребру, что будем отображать в названии алгоритма (RVPEP, random volume points edges propagation).

**Результаты**

Для сравнения эффективности описанных

оценивать эффективность алгоритма партицирования GBAG, является отклонение максимальной вычислительной нагрузки узла от среднего значения, то есть величина  $w_p^{dev}$ . В качестве второго показателя эффективности будем использовать отношение максимального количества межпроцессных обменов партии к среднему показателю объема всех обменов (данная величина вводилась ранее и обозначалась через  $\chi$ ).

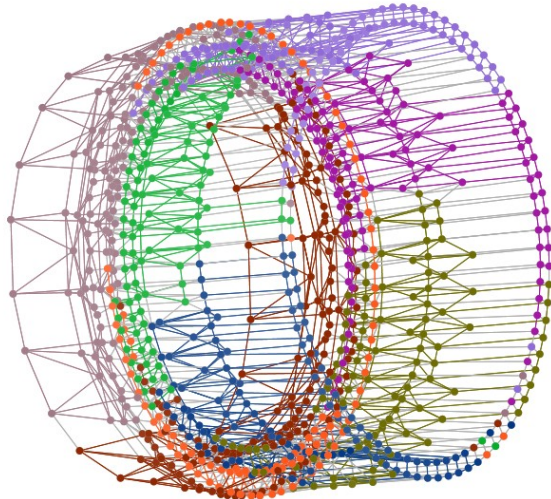


UG : partitions = 8, deviation = 0.0404319482990056, cross\_edges\_f = 83.443951167

Рис. 3. Результат разделения тестовой сетки на 8 партий алгоритмом UG

На рисунках 3 и 4 показана визуальная разница в применении алгоритмов UG и RVPEP для 8 партий. В случае использования алгоритма

RVPEP на рисунке 4 четко различаются локализованные партии (вершины, окрашенные в один цвет), а также показатель  $\chi$  (на рисунках эта величина обозначается  $\text{cross\_edges\_f}$ ) вдвое меньше соответствующего показателя для алгоритма UG, что говорит о существенном сокращении кросс-ребер.



RVPEP/EM : partitions = 8, deviation = 0.263913283000635, cross\_edges\_f = 41.8422

Рис. 4. Результат разделения тестовой сетки на 8 партий алгоритмом RVPEP

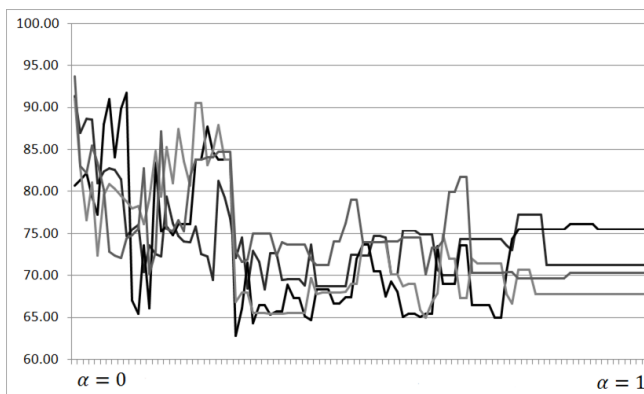


Рис. 5. Зависимость показателя  $\chi$  (значения в процентах отложены по вертикальной оси) от параметра  $\alpha$  для алгоритма RVPEP

Таким образом, для сокращения межпроцессных обменов целесообразно использовать алгоритм RVPEP. Осталось показать, какое влияние на эффективность оказывает

параметр  $\alpha$ , определяющий баланс между равномерностью загрузки вычислительных узлов и уменьшением количества межпроцессных обменов. На рисунках 5 и 6 приведены данные зависимости при распределении вычислительной нагрузки на 64 партии.

На рис. 5 приведена зависимость величины  $\chi$  от параметра  $\alpha$ , а на рис. 6 — зависимость показателя  $w_p^{dev}$  от того же значения  $\alpha$ . Из данных рисунков видно, что использование значения параметра  $\alpha$  примерно от 0.3 и выше уже существенно сокращает количество кросс-ребер в графе, тогда как равномерность распределения весов блоков по партициям изменяется незначительно.

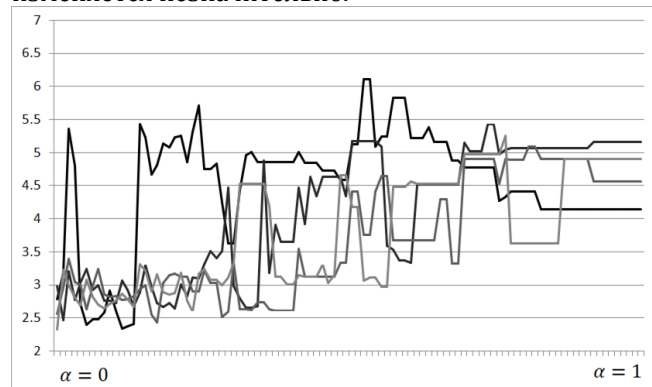


Рис. 6. Зависимость показателя  $w_p^{dev}$  (значения в процентах отложены по вертикальной оси) от параметра  $\alpha$  для алгоритма RVPEP

### Заключение

Проведенные исследования показали, что использование алгоритма RVPEP партиционирования графа смежности блочно-структурированной сетки позволяет добиться равномерного распределения вычислительной нагрузки на узлы суперкомпьютера. При этом варьирование параметра алгоритма, позволяющего выбирать подходящую стратегию пропагации партии, позволяет существенно сократить количество кросс-ребер в графе, тем самым уменьшая объем межпроцессных обменов данными.

Также заметим, что алгоритмы UG и RVPEP, описанные в статье, могут быть без труда расширены на случай неоднородных партий, что делает возможным их использование для гетерогенных вычислительных систем.

### Литература

1. Blazek J. Computational fluid dynamics: Principles and applications. – Elsevier, 2001.
2. Farrashkhalvat M., Miles J.P. Basic structured grid generation, with an introduction to unstructured grid generation. - Butterworth-Heinemann, 2003.
3. Queen M. Parallel programming in C with MPI and OpenMP. - Mc-Grow Hill, 2004.
4. Рыбаков А.А. Внутреннее представление и механизм межпроцессного обмена для блочно-структурированной сетки при выполнении расчетов на суперкомпьютере. // Программы системы: Теория и приложения, №1 (32), 2017, с. 121-134, ISSN 2079-3316.
5. Рыбаков А.А. Распределение вычислительной нагрузки между узлами суперкомпьютерного кластера при расчетах задач газовой динамики с дроблением расчетной сетки. // Международный научный журнал «Современные информационные технологии и ИТ-образование», Том 12, номер 2, 2016, с. 101-107.

6. Описание интерфейса пользователя, предназначенного для работы с интеловской гибридной архитектурой суперЭВМ (СК), где вместе с процессорами Intel Xeon используются сопроцессоры Intel Xeon Phi. URL <http://www.jscs.ru/informat/MVS-10PInter.pdf>.

## References

1. Blazek J. Computational fluid dynamics: Principles and applications. – Elsevier, 2001.
2. Farrashkhalvat M., Miles J.P. Basic structured grid generation, with an introduction to unstructured grid generation. - Butterworth-Heinemann, 2003.
3. Queen M. Parallel programming in C with MPI and OpenMP. - Mc-Grow Hill, 2004.
4. Rybakov A.A. Vnutrennee predstavlenie i mekhanizm mezhprotsessnogo obmena dlya blochno-strukturirovannoi setki pri vypolnenii raschetov na superkomp'yutere. // Programmy sistemy: Teoriya i prilozheniya, №1 (32), 2017, s. 121-134, ISSN 2079-3316.
5. Rybakov A.A. Raspredelenie vychislitel'noi nagruzki mezhdu uzlami superkomp'yuternogo klastera pri raschetakh zadach gazovoi dinamiki s drobleniem raschetnoi setki. // Mezhdunarodnyi nauchnyi zhurnal «Sovremennyye informatsionnyye tekhnologii i IT-obrazovanie», Tom 12, nomer 2, 2016, s. 101-107.
6. Opisanie interfeysa pol'zovatelya, prednaznachennogo dlya raboty s intelovskoy gibridnoy arkhitekturoy superEVM (SK), gde vmeste s protsessorami Intel Xeon ispol'zuyutsya soprotsessory Intel Xeon Phi. URL <http://www.jscs.ru/informat/MVS-10PInter.pdf>.

Поступила: 21.03.2017

### Об авторе:

**Рыбаков Алексей Анатольевич**, кандидат физико-математических наук, ведущий научный сотрудник, Межведомственный суперкомпьютерный центр Российской академии наук - филиал Федерального государственного учреждения «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук, [rybakov@jscs.ru](mailto:rybakov@jscs.ru), [rybakov.aax@gmail.com](mailto:rybakov.aax@gmail.com)

### Note on the author:

**Rybakov Alexey**, Candidate of Physical and Mathematical Sciences, Senior research fellow Joint Supercomputer Center of the Russian Academy of Sciences — branch of Scientific Research Institute of System Analysis of the Russian Academy of Sciences, [rybakov@jscs.ru](mailto:rybakov@jscs.ru), [rybakov.aax@gmail.com](mailto:rybakov.aax@gmail.com)