ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ №1(32), 2017, с. 121–134

УДК 004.4

А. А. Рыбаков

Внутреннее представление и механизм межпроцессного обмена для блочно-структурированной сетки при выполнении расчетов на суперкомпьютере

Аннотация. В статье рассматривается внутреннее представление блочно-структурированной сетки, описание основных ее объектов и взаимодействие их между собой. Также рассматривается механизм организации межпроцессного обмена данными, при выполнении расчетов на суперкомпьютере.

Ключевые слова и фразы: суперкомпьютер; математическое моделирование; блочно-структурированная расчетная сетка; внутреннее представление; межпроцессный обмен.

Введение

Численные расчеты задач газовой динамики [1, 2] как правило связаны с большим объемом вычислений, поэтому при использовании суперкомпьютеров важную роль имеет как скорость вычислений, так и оптимальность загрузки вычислительных мощностей. Одним из наиболее распространенных типов расчетных сеток, используемых для численного решения задач газовой динамики, является блочно-структурированная сетка, состоящая из нескольких блоков, некоторые из которых имеют общие границы. Такая структура позволяет выполнять вычисления для различных блоков независимо друг от друга, обмениваясь общими данными на границах по мере необходимости. Для эффективного проведения вычислений на блочно-структурированных сетках с использованием суперкомпьютера важную роль имеет реализация расчетной сетки в виде внутреннего представления и организация механизма межпроцессного обмена данными между соседними блоками сетки.

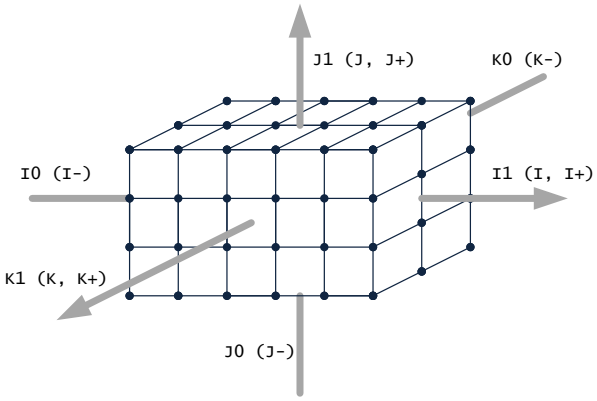


Рис. 1. Блок сетки

1. Структура сетки

Блочно-структурированные сетки состоят из отдельных блоков, каждый из которых представляет собой упорядоченный трехмерный массив условно кубических ячеек. Упорядоченность размещения ячеек позволяет быстрее производить вычисления и снижает требования к объему памяти, однако строить блочноструктурированные сетки гораздо сложнее, чем неструктурированные [4, 5].

Блоки блочно-структурированной сетки могут иметь сложную форму. Для доступа к отдельным ячейкам блока вводятся три индекса, связанные с криволинейной системой координат блока. Система координат задается тремя линиями координат (I, J, K), каждая из которых связывает пары противоположных граней блока. Для наглядности будем изображать блоки сетки в виде прямоугольных параллелепипедов, как показано на рис. 1.

Блоки могут граничить между собой. Граница между блоками описывается интерфейсом. Один интерфейс описывает соприкосновение двух блоков прямоугольными подобластями своих граней. При этом системы координат двух соседних блоков не обязаны согласовываться между собой. Пример касания двух блоков приведен на рис. 2.

Расчет газодинамической задачи носит итерационный по времени характер. Для каждой итерации по времени выполняется пересчет данных ячеек согласно той или иной вычислительной схеме. Во время

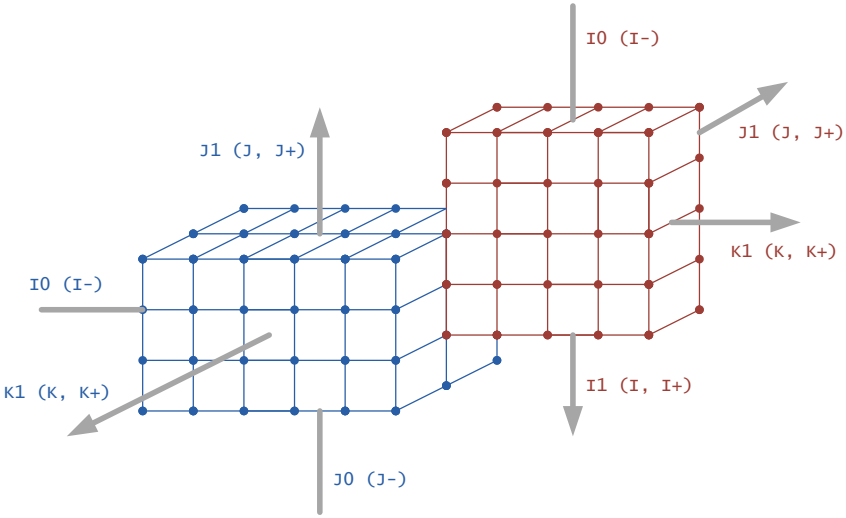


Рис. 2. Касание блоков сетки

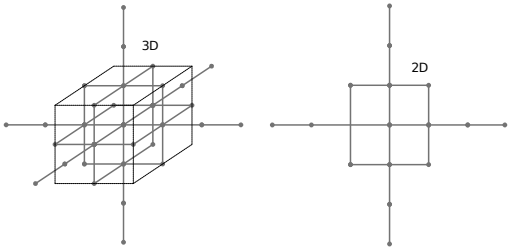


Рис. 3. Примеры вычислительных окрестностей ячеек сетки

обработки одной ячейки возникает потребность обращаться за данными к ячейкам ее окрестности (в простейшем случае это просто данные соседних по граням ячеек, но могут использоваться и используются более сложные структуры окрестностей). На рис. 3 показаны примеры вычислительных окрестностей ячейки.

В дальнейшем для большей наглядности будем изображать блоки в двумерном виде (с системой координат IJ).

Во время произведения расчетов газодинамической задачи различные блоки сетки обрабатываются независимо друг от друга. При этом некоторые ячейки обрабатываемого блока должны обращаться

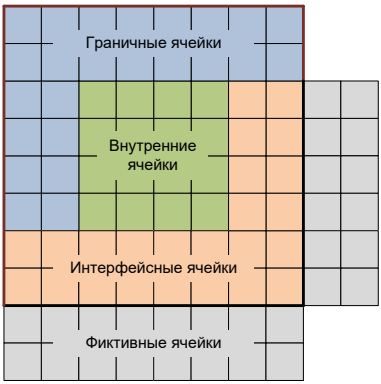


Рис. 4. Ячейки разных типов

за данными к ячейкам соседних блоков. Это происходит в том случае, если окрестность ячейки расположена сразу в нескольких блоках. Для эффективного счета необходимо обеспечить функционал быстрого обмена данными между блоками. Если два соседних блока обрабатываются на разных узлах суперкомпьютерного кластера, то для обмена данными между ними нужно использовать MPI-обмены [**6**, **7**]. Приведем классификацию ячеек внутри блока сетки (рис. 4).

Внутренними ячейками (Inner Cells) будем называть те ячейки блока, вся окрестность которых лежит внутри того же блока. Остальные ячейки блока будем называть граничными (Border Cells), так как окрестность этих ячеек пересекает границу блока. Если окрестность ячейки блока частично накрывает один из соседних блоков, то такую ячейку будем относить к интерфейсным ячейкам (Interface Cells), так как ее окрестность пересекает интерфейс блока.

Если расчет задачи выполняется на суперкомпьютерном кластере, окрестность ячейки может частично накрыть соседний блок, который обрабатывается в другом узле суперкомпьютера (рис. 5). В этом случае интерфейсную ячейку будем называть ячейкой MPI обмена (MPI Cells), так как для получения данных этой окрестности нужно использовать средства межпроцессного обмена.

При обработке ячеек MPI обмена мы должны обращаться за данными к другому процессу. Если делать это только по мере необходимости, то придется совершать большое количество обменов, что негативно скажется на производительности. Поэтому для повышения

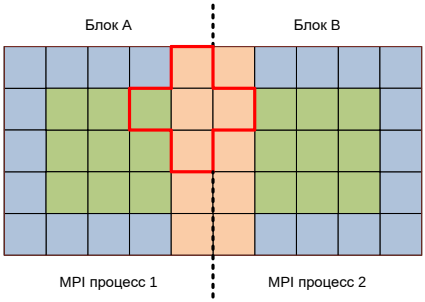


Рис. 5. Пересечение окрестности ячейки и соседнего блока

быстродействия вместо этого для блока заводится специальный слой внешних ячеек, куда копируются все необходимые данные из соседних процессов, которые могут понадобиться при обработке блока. Такие ячейки будем называть фиктивными. Ширина слоя (называемого также теневым слоем) определяется размером вычислительного шаблона обработки ячейки. Они могут не являться ячейками в физическом смысле, так как для проведения вычислений необходимы не все данные фиктивных ячеек, но на логическом уровне будем изображать их отдельными ячейками. Таким образом, объединение всех ячеек блока с множеством фиктивных ячеек является замкнутой структурой, которая более не требует для своего обсчета обращений к другим MPI процессам.

Далее опишем основные объекты, из которых состоит блочно-структурированная сетка, способы описания этих объектов и внутренние структуры, позволяющие эффективно оперировать данными объектами и производить вычисления.

2. Основные объекты сетки

Основным объектом блочно структурированной сетки является блок (Block) (рис. 6). Блок состоит из трехмерного массива ячеек размера ISize×JSize×KSize. Каждая ячейка содержит набор газодинамических параметров, ассоциированных с центром масс ячейки и другие вспомогательные данные. Также блок содержит данные о всех вершинах своих ячеек, эти данные хранятся в трехмерном массиве размера (ISize + 1)× (JSize + 1)×(KSize + 1).

Для определения геометрии блока необходимо знать его линейные размеры и координаты (X, Y, Z) всех его узлов.

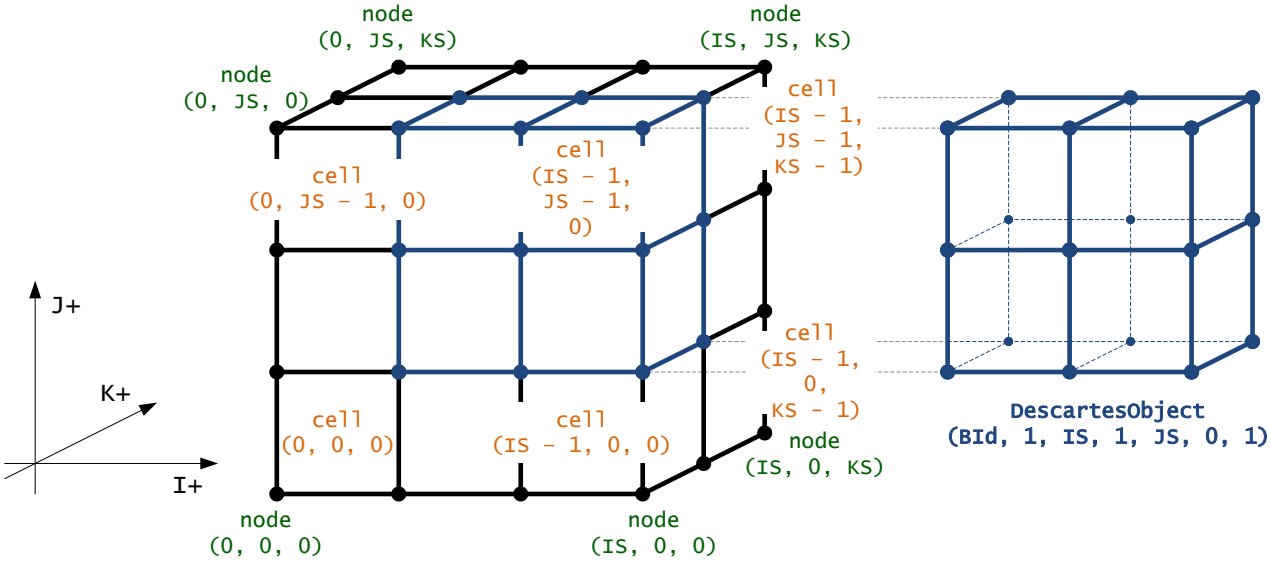


Рис. 6. Блок сетки и координаты его узлов

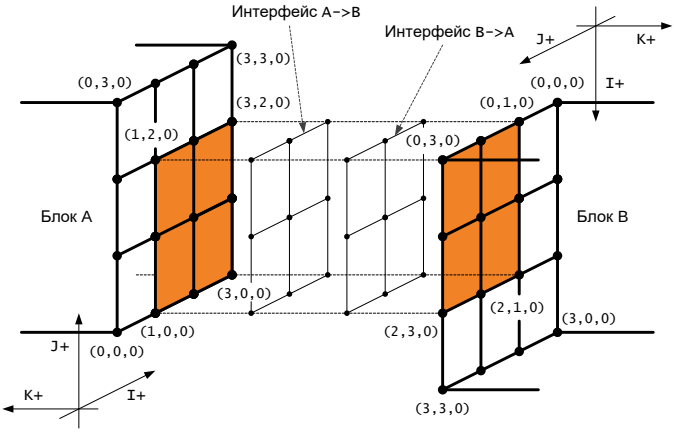


Рис. 7. Структура интерфейса касания блоков

Объектом, описывающим соприкосновение двух соседних блоков, является интерфейс (Iface) (рис. 7). 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 соседних блоков описывается парой смежных интерфейсов.

Интерфейс описывает только касание по ненулевой площади (касание по узлу или ребрам ячеек не рассматривается). Также возможно обоюдное касание друг друга разных граней одного и того же блока, хотя такие случаи нужно рассматривать отдельно, так как самокасание может потребовать специальной обработки блоков при управлении сеткой. В общем случае самокасания блоков рекомендуется избегать.

Интерфейс описывается в системе координат того блока, к которому он относится. Координаты задают размер области соприкосновения по всем трем направлениям.

В данном примере два смежным интерфейса, описывающих касание блоков A и B задаются следующим образом:

$$\mathrm{A}(1,3,0,2,0,0) \to \mathrm{B},$$
$$\mathrm{B}(0,2,1,3,0,0) \to \mathrm{A}.$$

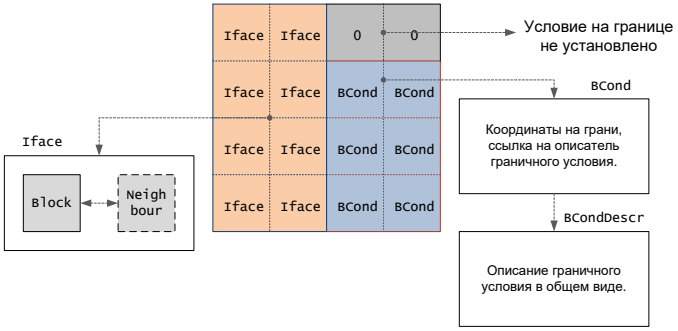


Рис. 8. Структура грани блока

То есть для одного интерфейса задается блок, которому этот интерфейс принадлежит, координаты интерфейса в данном блоке и соседний блок. Данной информации достаточно чтобы полностью определить геометрию касания блоков. Направление интерфейса можно определить по той координате, протяженность интерфейса по которой равна нулю (в данном случае это координата K). Остается определить только каким образом системы координат двух соседних блоков соотносятся друг с другом. Это можно сделать, зная координаты узлов соответствующих блоков.

На границе блока кроме другого блока может находиться и вовсе граница расчетной области. В этом случае на границе блока необходимо задать граничные условия (BCond). Граничные условия задаются аналогично интерфейсу (с помощью системы координат блока), однако вместо соседнего блока подается ссылка на описатель граничных условий. Таким образом и интерфейс и граничное условие являются частным случаем границы блока (Border).

В процессе вычислений необходимо иметь возможность быстро узнать для конкретной граничной ячейки, какого типа граница ей соответствует и быстро найти соответствующий интерфейс или граничное условие. Для этой цели служат грани блока (Facet). Каждый блок имеет 6 граней (для каждого из направлений I-, I+, J-, J+, K-, K+). Каждая грань является двумерным массивом, содержащим для каждой ячейки, граничащей с гранью, ссылку на интерфейс или на граничное условие (рис. 8).

Важным объектом сетки является область блока (Scope). Области нужны для описания начальных условий. Область блока в целом

аналогична граничному условию за исключением того, что является трехмерным объектом. Область блока задается в системе координат блока, имеет ненулевую протяженность по каждому направлению и ссылается на описатель начальных условий. Структура описателя начальных условий полностью аналогична описателю граничных условий.

Таким образом все основные рассмотренные объекты сетки являются двумерными или трехмерными декартовыми объектами, то есть такими объектами, чья геометрия задается двумерными или трехмерными массивами узлов блоков сетки. Также заметим, что граничные условия и области являются именованными объектами, так как им присваиваются имена для определения механизмов задания граничных и начальных условий расчета. Исходя из этого, получаем следующую иерархию классов, реализующих расчетную сетку (рис. 9).

3. Организация межпроцессных обменов данными между блоками сетки

Рассмотрим механизм обмена данными между блоками сетки. Как уже упоминалось, если два соседние блока, между которыми должен быть осуществлен обмен данными, обрабатываются в разных процессах при выполнении расчетов на суперкомпьютере, то обмен данными может быть осуществлен с помощью MPI пересылок. При этом каждый блок должен отправить своим соседям только данные, находящиеся в его интерфейсных ячейках (могут существовать интерфейсные ячейки, данные из которых требуется отправить сразу двум или трем блокам). Получить же должен блок те данные, которые соответствуют ячейкам его теневого слоя.

Логика обменов устроена следующим образом. Так как пара смежных интерфейсов определяет факт касания двух блоков, между которыми должен произойти обмен, то на каждом из этих интерфейсов заводится буфер для данных обмена, после чего происходит обмен данными буферами (рис. 10). Обмены данными происходят одновременно по всем интерфейсам сетки с помощью асинхронных функций MPI_Isend, MPI_Irecv. Рассмотрим механизм обмена данными между соседними блоками, находящимися в разных процессах, более подробно.

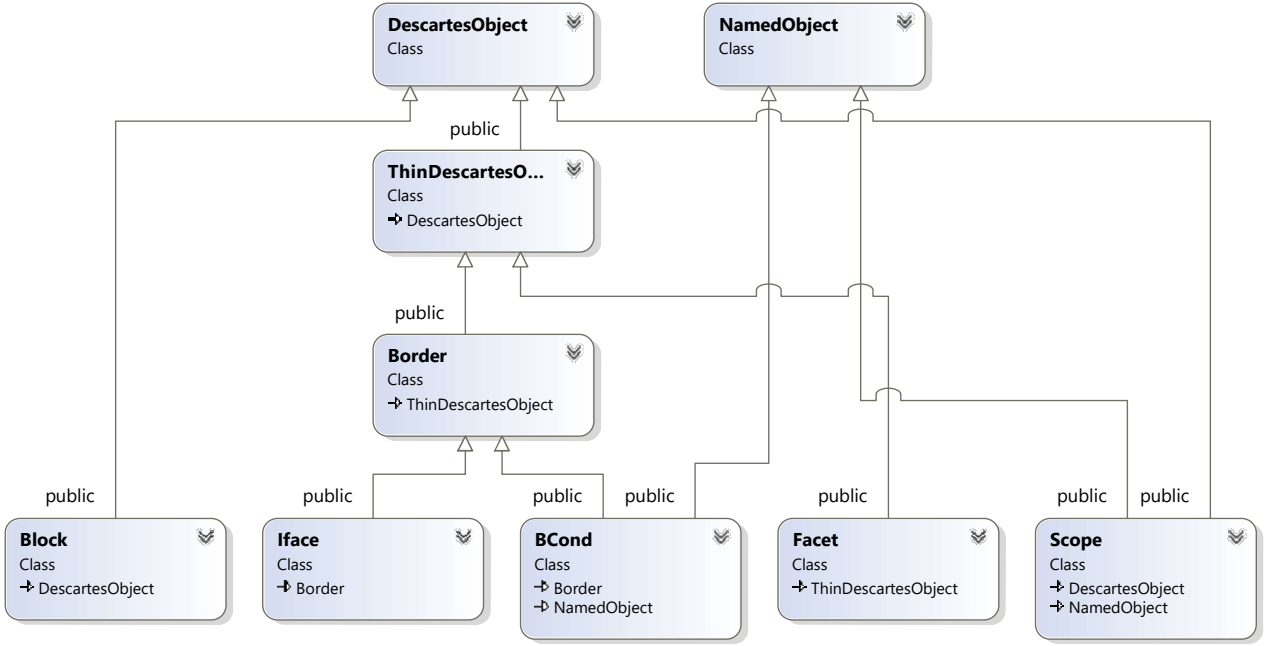


Рис. 9. Иерархия объектов внутреннего представления сетки

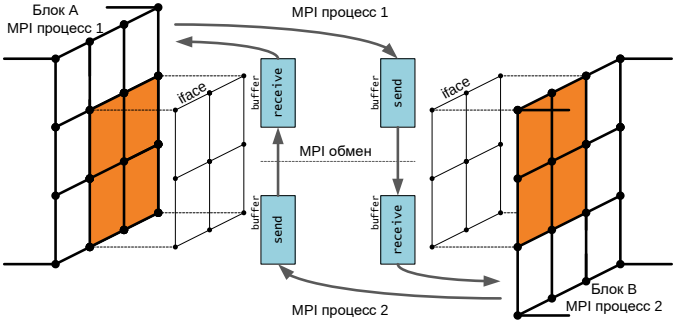


Рис. 10. Обмен данными между блоками сетки

Пусть Блок A обрабатывается в процессе 1, а блок B — в процессе 2, как показано на рис. 10. Каждый из двух смежных интерфейсов, описывающих касание этих двух блоков имеет свой буфер обмена данными. Так как в процессе 1 активным является блок A, то буфер его интерфейса работает на прием данных, а буфер интерфейса со стороны блока B — на отправку (данных блока A). В процессе 2 все происходит ровно наоборот: активным является блок B, буфер интерфейса блока A работает на отправку данных блока B, буфер интерфейса со стороны блока B работает на прием данных. Для всех интерфейсов происходит асинхронный запуск всех вызовов функций обмена, после чего происходит ожидание завершения всех операций. После этого каждый блок может извлечь данный из буфера интерфейса, в котором он является активным, и поместить эти данные в свою теневую зону, после чего начинается следующая итерация расчетов.

После проведения каждой итерации счета задачи необходимо выполнять обмен данными, чтобы состояния граничных ячеек блоков были актуальными. При этом начинать обсчет следующей итерации нельзя до завершения всех обменов. Это может привести к возникновению задержек (рис. 11). Чтобы избежать такого рода потери вычислительного времени можно разделить обработку ячеек блока на отдельную обработку всех внутренних ячеек и обработку граничных ячеек. Сначала нужно произвести обработку всех граничных ячеек блока, затем инициировать асинхронные обмена данными (в которых участвуют только граничные ячейки), а затем сразу начать обработку внутренних ячеек, для которых получение данных фиктивных ячеек не требуется. Такой подход позволяет полностью скрыть издержки на коммуникацию за полезными вычислениями (рис. 12).

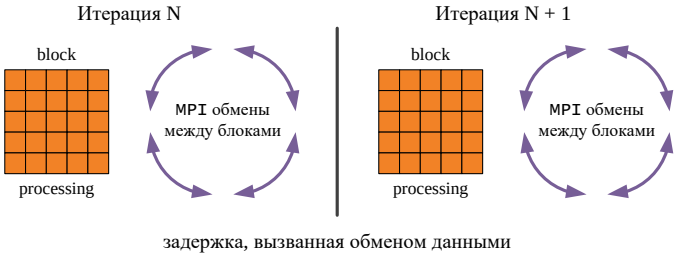


Рис. 11. Возникновение задержки, вызванной mpi обменами

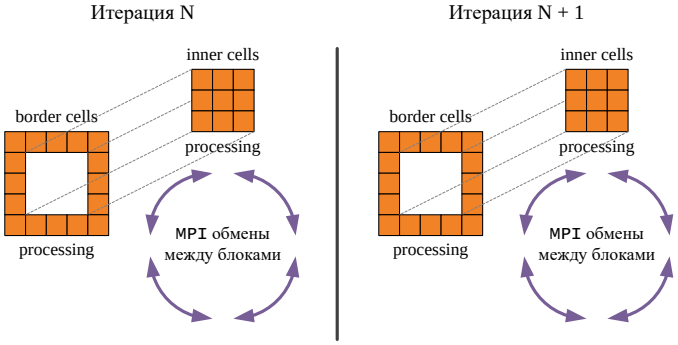


Рис. 12. Избавление от задержки путем разделения вычислений для внутренних и граничных ячеек

Заключение

Особенности реализации внутреннего представления блочно-структурированной сетки имеют большое значение для выполнения расчетов на этой сетке. Правильный выбор объектной модели сетки, способы расположения данных в памяти и другие факторы напрямую влияют на эффективности выполнения программы.

Важную роль в скорости выполнения расчетов играет реализация механизма обмена данными в случае запуска вычисления на суперкомпьютере. Так как обмен данными должен происходить на каждой итерации вычислений, то это является узким местом для повышения эффективности расчетов.

Список литературы

[1] Н. Г. Бураго. *Вычислительная механика*, Конспект лекций, М., 2012, 275 с. ↑ 121

[2] J. Blazek. *Computational fluid dynamics: Principles and applications*, Elsevier, 2001. ↑ 121

[3] M. Farrashkhalvat, J. P. Miles. *Basic structured grid generation, With an introduction to unstructured grid generation*, Butterworth-Heinemann, 2003. ↑

[4] V. Liseikin. *Grid generation methods*, Springer, 2010. ↑ 122

[5] А. С. Лебедев, В. Д. Лисейкин, Г. С. Хакимзянов. «Разработка методов построения адаптивных сеток», *Вычислительные технологии*, **7**:3 (2002), с. 29–43. ↑ 122

[6] К. Хьюз, Т. Хьюз. *Параллельное и распределенное программирование с использованием C++*, Вильямс, М., 2004, 672 с. ↑ 124

[7] M. Queen. *Parallel programming in C with MPI and OpenMP*, Mc-Grow Hill, 2004. ↑ 124

Рекомендовал к публикации *Программный комитет*

Пятого национального суперкомпьютерного форума ***НСКФ-2016***

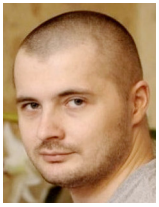
Пример ссылки на эту публикацию:

А. А. Рыбаков. «Внутреннее представление и механизм межпроцессного обмена для блочно-структурированной сетки при выполнении расчетов на суперкомпьютере», *Программные системы: теория и приложения*, 2017, **8**:1(32), с. 121–134.

URL: http://psta.psiras.ru/read/psta2017_1_121-134.pdf

Об авторе:

Алексей Анатольевич Рыбаков



Рыбаков Алексей Анатольевич - к ф.-м. н., внс МСЦ РАН - филиала ФГУ ФНЦ НИИСИ РАН. Области научных интересов - математическое моделирование задач газовой динамики с использованием суперкомпьютеров, методы построения и управления расчетными сетками, дискретная математика, теория графов, модели случайных графов, параллельное программирование, функциональное программирование.

e-mail: rybakov@jscc.ru

A. Rybakov. *Inner representation and crossprocess exchange mechanism for block-structured grid for supercomputer calculations.*

Abstract. In the article block-structured grid inner representation, its basic objects description and their cooperation are considered. Also crossprocess data exchange mechanism for supercomputer calculations is described. (*In Russian*).

Key words and phrases: supercomputer, mathematical modeling, block-structured calculation grid, inner representation, crossprocess exchange.

References

[1] N. G. Burago. *Vychislitel'naya mekhanika*, Konspekt lektsiy, M., 2012 (in Russian), 275 p.

[2] J. Blazek. *Computational fluid dynamics: Principles and applications*, Elsevier, 2001.

[3] M. Farrashkhalvat, J. P. Miles. *Basic structured grid generation, With an introduction to unstructured grid generation*, Butterworth-Heinemann, 2003.

[4] V. Liseikin. *Grid generation methods*, Springer, 2010.

[5] A. S. Lebedev, V. D. Liseykin, G. S. Khakimzyanov. "Development of Methods for Generating Adaptive Grids", *Vychislitel'nyye tekhnologii*, **7**:3 (2002), pp. 29–43 (in Russian).

[6] C. Hughes, T. Hughes. *Parallel and Distributed Programming Using C++*, 1 edition, Addison-Wesley Professional, 2003, 720 p.

[7] M. Queen. *Parallel programming in C with MPI and OpenMP*, Mc-Grow Hill, 2004.

Sample citation of this publication:

A. Rybakov. "Inner representation and crossprocess exchange mechanism for block-structured grid for supercomputer calculations", *Program systems: Theory and applications*, 2017, **8**:1(32), pp. 121–134. (*In Russian*).

URL: http://psta.psiras.ru/read/psta2017_1_121-134.pdf