Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«Московский физико-технический институт (государственный университет)»

Факультет управления и прикладной математики
Кафедра информатики и вычислительной математики

Направление подготовки: 03.04.01 Прикладные математика и физика
Профиль подготовки: Математические и информационные технологии
Форма обучения: очная

Допущен к ГИА
Руководитель учебного подразделения
_____/_____
(подпись) *расшифровка подписи*
«___» _____20__г.

РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМОВ КОНТАКТНЫХ УСЛОВИЙ В СЛУЧАЕ СЛОЖНОГО КОНТАКТА МНОГИХ СЕТОК

магистерская диссертация

Обучающийся: Кассирова Ольга Игоревна
_____
(подпись обучающегося)
Научный руководитель: Хохлов Николай Игоревич, кандидат физ.-мат. наук
_____
(подпись научного руководителя)

Москва 2018

Аннотация

Целями данной работы являлись разработка параллельного алгоритма расчета контактных поверхностей в многоблочной сетке, разработка алгоритма декомпозиции многоблочных структурных сеток, реализация полученных алгоритмов и проведение их тестирования.

В ходе работы данные алгоритмы были реализованы в программном комплексе лаборатории прикладной вычислительной геофизики МФТИ и протестированы на задачах математического моделирования. В работе приводится описание полученных алгоритмов и результаты тестирования.

Содержание

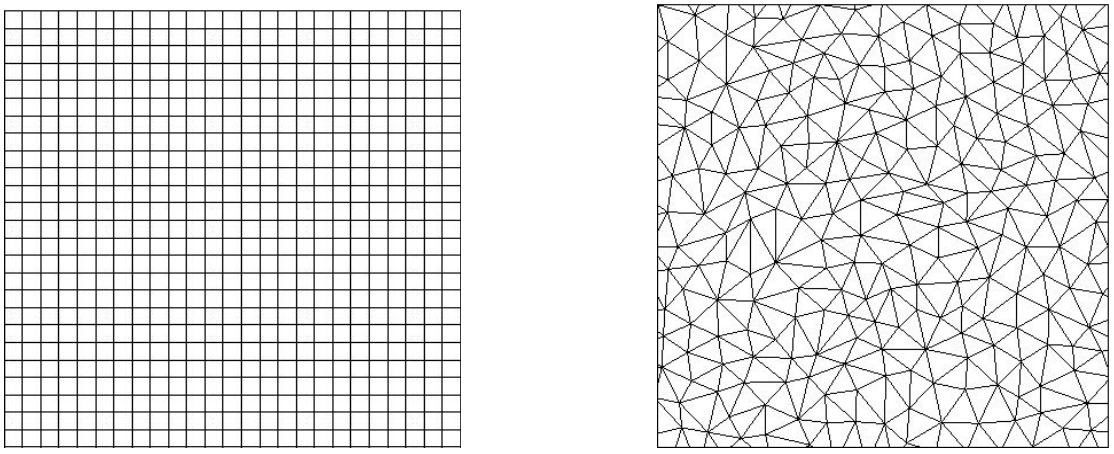
Глава 1. Введение **4**
1.1 Обзор предметной области и существующих решений 4
1.2 Актуальность 8
1.3 Новизна 10
Глава 2. Декомпозиция блочных сеток между процессами **11**
2.1 Описание алгоритма 11
2.2 Теоретическая оценка 13
2.3 Описание тестовой задачи 15
2.4 Результаты тестирования 16
Глава 3. Нахождение процессами своих соседей и обмен узлами **23**
3.1 Архитектурные ограничения 23
3.2 Описание алгоритма 25
3.3 Описание тестовой задачи 26
3.4 Результаты тестирования 27
Заключение **31**
Список литературы **32**

Глава 1. Введение

1.1 Обзор предметной области и существующих решений

Одним из способов дискретизации распределения физических величин при численном решении дифференциальных и интегральных уравнений является расчетная сетка. Расчетная сетка представляет собой совокупность узлов (ячеек), в которых хранятся значения сеточной функции. Качество используемой расчетной сетки в значительной степени влияет на точность численного решения уравнения.

Выделяют два основных класса расчетных сеток: структурные и неструктурные. Примеры структурных и неструктурных сеток приведены на рисунке 1. Сетку называют структурной, если множество ее узлов является упорядоченным. Каждая вершина структурной сетки имеет одинаковое число соседей. При программной реализации структурная сетка может быть представлена в виде многомерного массива с индексацией вида *(i, j, k)* (в трехмерном случае). К структурным сеткам относятся, например, прямоугольные и криволинейные.



а) б)

Рис. 1. Примеры расчетных сеток: а) структурная прямоугольная; б) неструктурная треугольная.

Элементы неструктурных сеток не являются упорядоченными, каждая вершина может иметь произвольное число соседей. В программной реализации неструктурные сетки могут быть реализованы в виде многосвязных списков.

При численном моделировании на параллельных вычислительных системах с распределенной памятью задач механики сплошных сред, электродинамики и других задач, при решении которых используются явные численные методы, возникает проблема рациональной декомпозиции расчетных сеток. В зависимости от типа сеток, их размера, используемого численного метода и других факторов применяют тот или иной алгоритм.

Декомпозиция может быть как статической, так и динамической [1, с. 228-229]. Последнюю имеет смысл использовать в задачах, где время счета отдельного узла непостоянно, а нагрузка на процессы при статической декомпозиции менялась бы с течением времени. Методы динамической балансировки реализованы, например, в пакетах DRAMA [2] и Zoltan [3]. Алгоритм, который будет описан в этой статье, реализует статическую декомпозицию, поэтому остановимся подробнее на ней.

Задача декомпозиции неструктурной сетки обычно сводится к задаче разбиения графа. В этом случае рассматривается граф, который соответствует коммуникационным нагрузкам сетки. При декомпозиции неструктурных сеток может быть использован спектральный алгоритм, основанный на вычислении собственных векторов лапласиана графа сетки [4, с. 83]. Получаемое в результате его применения разбиение часто лучше разбиений, получаемых с помощью других методов, однако применение данного алгоритма может быть затруднительно для больших сеток в силу вычислительной сложности.

Существуют также иерархические алгоритмы, состоящие из огрубления графа, декомпозиции самого маленького из полученных графов и отображения разбиения на предыдущие графы [5, с. 49]. Используется и KL/FM алгоритм [6],

который основывается на понятии о cell gain (выгоды от перемещения вершины из одного блока в другой), инкрементный алгоритм [7], предложенный М. В. Якобовским, и другие. Многие методы реализованы в пакетах PARTY [8], Jostle [9], METIS [10], Chaco [11] и GridSpiderPar [12].

Декомпозиция структурных сеток значительно проще декомпозиции неструктурных. Алгоритмы декомпозиции, о которых было сказано выше, можно применить и к структурным сеткам, но результатом их работы будет набор неструктурных сеток. С точки зрения программной реализации это не всегда удобно, так как не позволяет использовать преимущества структурного разбиения при реализации численного метода и при обмене данными между блоками.

При наличии одной сетки часто применяется алгоритм простой геометрической декомпозиции: расчетная область разбивается на равные (или почти равные) части по нескольким осям [13, 14, 15]. Это удобно в том числе за счет того, что в MPI реализована функция определения оптимальной конфигурации сетки *MPI_Dims_create()*, основанная на минимизации размера данных, пересылаемых между блоками.

Часто расчетная область бывает такова, что для ее моделирования удобно использовать несколько структурных сеток. Одним из подходов к распараллеливанию таких задач является генерирование определенного количества сеток, соответствующего количеству процессов, на которых будут производиться вычисления [16, 17]. В приведенных работах также используется технология OpenMP в качестве следующего уровня параллелизма.

Если декомпозиция проводится независимо от генерации сеток, то задача усложняется. В работах M. J. Djomehri и R. Biswas [18, 19, 20] рассмотрен алгоритм, состоящий из деления наибольших по размеру сеток на блоки и дальнейшей их группировки. Для определения сеток, которые нужно дополнительно разделить, пользователем задаются параметры *maxnb* и *maxgrd*. При группировке блоков используется информация о контактах между сетками:

блок присоединяется к наименьшей группе, в которой уже есть хотя бы один контактирующий с ним блок. В [20] рассмотрены также методы группировки блоков с помощью пакетов CHACO и EVAH, предназначенных для разбиения графов.

В данной работе будет рассматриваться алгоритм статической декомпозиции многоблочных структурных сеток при условии, что декомпозиция проводится независимо от генерации сеток.

После того, как множество сеток будет декомпозировано между процессами параллельной вычислительной системы, необходимо выстроить такие коммуникации между процессами, которые позволяли бы двум контактирующим сеткам обмениваться узлами. Алгоритм, который будет предложен в данной работе, жестко связан с архитектурой проекта, в котором он был реализован.

1.2 Актуальность

Данная работа была выполнена в рамках работы над проектом лаборатории прикладной вычислительной геофизики МФТИ. Алгоритмы, описанные в этой работе, реализованы и используются в программном комплексе RECT, предназначенном для решения задач сейсмики и сейсморазведки, используя сеточно-характеристический метод [21, 22].

Специфика алгоритма декомпозиции состоит в том, что он применим только к многоблочным сеткам. Ранее уже приводилось утверждение, что для ее моделирования некоторых расчетных областей удобно использовать несколько структурных сеток. Покажем это на примере модели шара.

Если пытаться моделировать шар с помощью одной криволинейной структурной сетки путем “скругления” ее углов, то пространственный шаг между узлами сетки в этих углах будет стремительно уменьшаться, что приведет к увеличению числа Куранта при том же временном шаге. То есть по мере приближения сетки к форме шара для устойчивости численного решения необходимо будет уменьшать временной шаг. Это ведет к увеличению числа временных шагов и, следовательно, к увеличению времени расчета задачи.

Куда более оптимальным будет моделирование шара с помощью нескольких контактирующих криволинейных сеток. На рисунке 2 приведен пример такой модели с использованием семи сеток: центрального кубика и шести секторов.

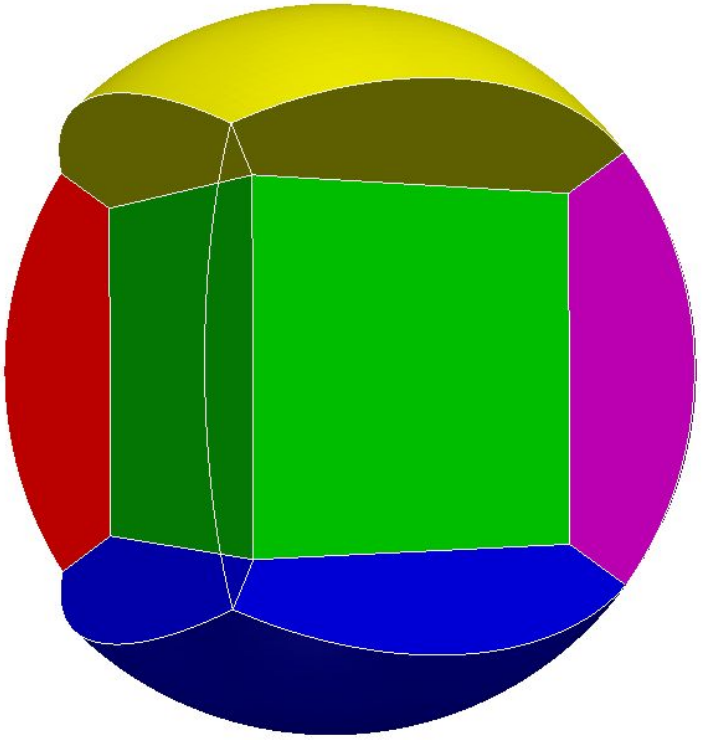


Рис. 2. Пример модели шара, состоящей из нескольких криволинейных сеток.

С помощью многоблочных сеток построена, например, модель дома в работе Григорьевых Д.П. [23].

1.3 Новизна

Из приведенного выше обзора существующих алгоритмов декомпозиции можно сделать вывод, что алгоритмов декомпозиции для многоблочных сеток описано не так много.

В работах Luong P. многоблочная сетка генерируется под заранее заданное количество процессов, на которых будет производиться расчет. Такая автоматическая генерация, строго говоря, декомпозицией не является. Это совершенно другой подход к моделированию с помощью многоблочных сеток, который устраняет саму необходимость в декомпозиции.

Предлагаемый алгоритм декомпозиции основан на делении крупных сеток и группировке малых, что делает его похожим на алгоритм, предложенный M. J. Djomehri и R. Biswas. Но в отличие от него в предложенном алгоритме блоки, полученные делением крупных сеток, не подлежат группировке, а для самого деления пользователю не требуется задавать дополнительные параметры. Входными данными служат размеры сеток и число процессов, на котором планируется проводить вычисления. Связность сеток в данном методе не учитывается. Это упрощает логику алгоритма и делает его применимым для тех архитектур, где контакты сеток неизвестны заранее и рассчитываются на более поздних шагах решения.

Глава 2. Декомпозиция блочных сеток между процессами

2.1 Описание алгоритма

Входными данными для предлагаемого алгоритма служат размеры сеток и число процессов MPI, на котором планируется проводить вычисления.

Пусть задача состоит из N сеток, в i-ой сетке M_i узлов. Расчет будет происходить на P процессах. Рассмотрим следующий алгоритм:

1. Вычисляется количество узлов, которые приходятся на один процесс. Это будет оптимальный размер блока для одного процесса.

$$M_{opt} = \frac{\sum_{i=1}^{N} M_i}{P}$$

2. Все сетки разделяются на две категории: большие и маленькие. Сетки размером менее оптимального размера блока M_{opt} относятся к маленьким, а остальные − к большим.
3. Для каждой большой сетки вычисляется оптимальное количество процессов для ее расчета P_i − частное размера сетки и оптимального размера, округленное вниз. Полученное число процессов сразу выделяется сеткам.

$$P_i = \left[\frac{M_i}{M_{opt}}\right]$$

4. Маленькие сетки необходимо сгруппировать по числу процессов, оставшихся незанятыми. Для этого будем использовать жадный алгоритм. Создаются группы, изначально по числу маленьких сеток. Шаг за шагом две самые маленькие (по количеству узлов) объединяются, пока групп не становится поровну с незанятыми процессами. Процессы распределяются по группам.

5. Если после предыдущего шага остались свободные процессы (например, маленьких сеток просто не было изначально), то они по одному раздаются большим сеткам с максимальной загрузкой процессов $Load_i$. Загрузкой процессов сетки назовем размер сетки, деленный на количество процессов, выделенных ей ранее.

$$Load_i = \frac{M_i}{P_i}$$

6. Все большие сетки делятся между выделенными им процессами методом простой геометрической декомпозиции.

2.2 Теоретическая оценка

При распараллеливании задач, состоящих из множества структурных сеток, самый очевидный способ декомпозиции − это деление каждой сетки между всеми процессами. Данный метод реализует самое оптимальное распределение вычислительной нагрузки. Однако при большом количестве сеток этот алгоритм может быть неэффективен из-за образования большого количества блоков и, следовательно, большого числа граничных узлов, требующих обмена данными с другими процессами. Далее будем называть этот метод простым алгоритмом декомпозиции.

Для того, чтобы дать оценку работе данного алгоритма, проведем сравнение с простой декомпозицией.

Рассмотрим трехмерную задачу, состоящую из N сеток и рассчитывающуюся на P процессах. Будем использовать следующие приближения:

1. Число процессов много больше количества расчетных сеток ($P >> N$).
2. Сетки имеют одинаковый размер. Каждая из N сеток состоит из M узлов.
3. Сетки представляют собой кубики − каждая из сторон имеет длину $M^{1/3}$ узлов.
4. Условия таковы, что P и $\frac{P}{N}$ − полные кубы, т. е. простая декомпозиция разделит сетки между таким числом процессов одинаково по трем осям.
5. Отсутствием пересылок на границах расчетной области можно пренебречь.

Для простой декомпозиции:

Рассмотрим отдельную сетку. В ней каждый процесс отправляет и принимает по 6 сообщений за шаг (по числу граней кубика). Всего в сетке происходит $6P$ обменов. Длина одного такого сообщения − площадь грани кубика − $\left(\frac{M}{P}\right)^{2/3}$. Размер пересылаемой информации в сетке

$2 \cdot 6P \cdot \left(\frac{M}{P}\right)^{2/3} = 12P^{1/3}M^{2/3}$. В N сетках обменов сообщениями $6PN$. Общий размер пересылаемой информации − $12NP^{1/3}M^{2/3}$.

Для предложенного алгоритма декомпозиции:

На одну сетку приходится $\frac{P}{N}$ процессов и, соответственно, $6\frac{P}{N}$ обменов сообщениями. Размер одного сообщения − $\left(\frac{MN}{P}\right)^{2/3}$. Размер всех сообщений в сетке − $2 \cdot 6\frac{P}{N} \cdot \left(\frac{M}{P}\right)^{2/3} = 12\left(\frac{M^2N}{P}\right)^{1/3}$. В N сетках $6P$ обменов. Общий размер пересылаемой информации − $N \cdot 12\left(\frac{M^2N}{P}\right)^{1/3} = 12P^{1/3}(MN)^{2/3}$.

Таким образом, количество обменов сообщениями уменьшилось в N раз, а общий объем пересылаемой информации сократился в $N^{1/3}$ раз. Размер отдельного сообщения ожидаемо вырос в $N^{2/3}$ раз. Так как время, которое тратится на передачу одного сообщения, растет нелинейно в зависимости от размера сообщения, то следует ожидать ускорения коммуникаций от использования данного метода более, чем в $N^{1/3}$ раз.

Следует отметить, что при использовании этого алгоритма для решения реальных задач балансировка нагрузки на процессы может оказаться неоптимальной. Его применение целесообразно в предположении, что потерянное время компенсируется ускорением обмена информацией.

2.3 Описание тестовой задачи

Для практической проверки эффективности приведенного алгоритма декомпозиции была выбрана задача прохождения волн через наземный объект [23]. В этой работе была использована модель здания, состоящая из 41 прямоугольной сетки.

Задача решается сеточно-характеристическим методом. Вначале применяется метод расщепления по пространственным координатам, в результате имеем три системы:

$$\frac{\partial u}{\partial t} = A_j \frac{\partial u}{\partial x_j},\ j = 1,\ 2,\ 3$$

Каждая из этих систем является гиперболической, поэтому их можно переписать в виде:

$$\frac{\partial u}{\partial t} = \Omega_j^{-1} \Lambda_j \Omega_j \frac{\partial u}{\partial x_j}$$

где матрица Ω_j - матрица составленная из собственных векторов, Λ_j − диагональная матрица, элементами которой являются собственные значения.

Для всех координат матрица Λ имеет вид:

$$\Lambda = diag\{c_1,\ -c_1,\ c_2,\ -c_2,\ c_2,\ -c_2,\ 0,\ 0,\ 0\}$$

где $c_1 = \sqrt{(\lambda + 2\mu)/\rho}$ − продольная скорость звука в среде, $c_2 = \sqrt{\mu/\rho}$ − поперечная скорость звука.

После замены переменных $\nu = \Omega u$ каждая из систем распадается на девять независимых скалярных уравнений переноса:

$$\frac{\partial \nu}{\partial t} + \Lambda \frac{\partial \nu}{\partial x} = 0$$

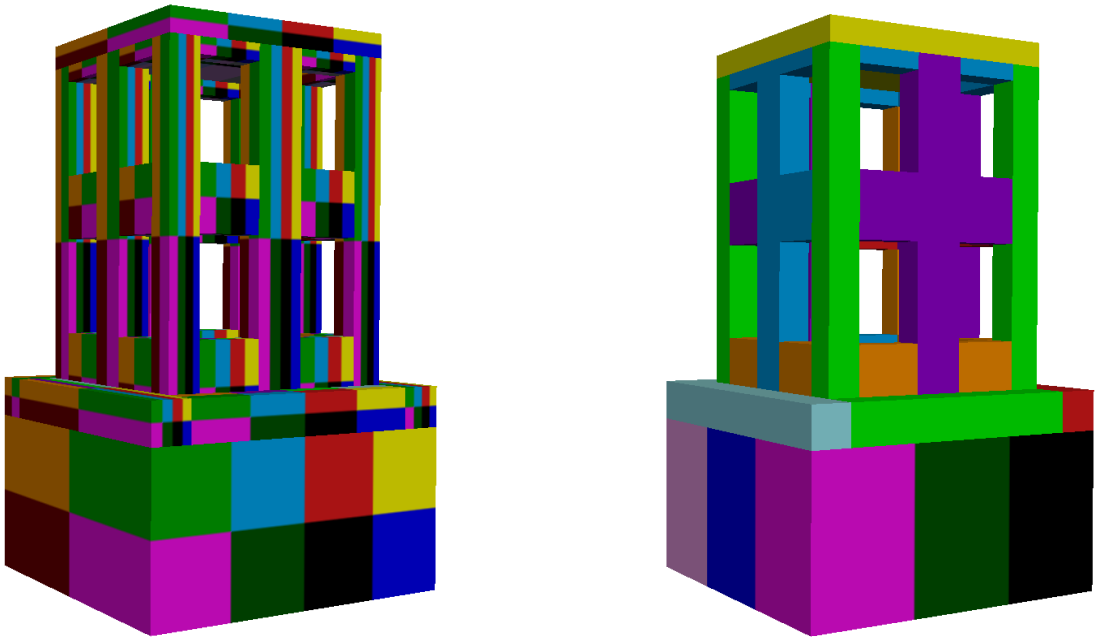
Одномерные уравнения переноса решаются с помощью TVD-схемы второго порядка точности с ограничителем superbee [24]. После того, как все компоненты ν перенесены, восстанавливается само решение:

$$u^{n+1} = \Omega^{-1} \nu^{n+1}$$

2.4 Результаты тестирования

Результаты тестирования алгоритмов декомпозиции на многопроцессорной вычислительной системе приведены в таблице 1. Результаты были получены с использованием вычислительных ресурсов МВК НИЦ «Курчатовский институт» (http://computing.kiae.ru/).

Для наглядности результат декомпозиции между 16 процессами представлен на рисунке 3. На рисунке видно, что общая площадь контактных поверхностей при использовании нового алгоритма декомпозиции существенно уменьшилась по сравнению с простым алгоритмом.



а) б)

Рис. 3. Декомпозиция модели между 16 процессами: а) предложенный алгоритм декомпозиции; б) алгоритм простой геометрической декомпозиции.

Табл. 1. Результаты тестирования предложенного алгоритма декомпозиции (новый алгоритм) и алгоритма простой геометрической декомпозиции (простой алгоритм).

Количество процессов	Простой алгоритм			Новый алгоритм		
	Время, с	Ускорение	Эффективн.	Время, с	Ускорение	Эффективн.
1	29308,5	1	1	29355,3	1	1
2	15443,4	1,8978010	0,94890050	16165,1	1,8159677	0,90798386
4	8344,19	3,5124439	0,87811099	8538,47	3,4380047	0,85950117
8	5260,09	5,5718628	0,69648285	5253,62	5,5876329	0,69845411
16	3041,53	9,6361041	0,60225651	3130,21	9,3780608	0,58612880
32	1912,58	15,324064	0,47887702	1827,32	16,064673	0,50202106
64	1079,98	27,138002	0,42403128	845,112	34,735396	0,54274056
128	751,655	38,991957	0,30462466	914,51	32,099484	0,25077722
192	588,439	49,807201	0,25941250	403,151	72,814652	0,37924298
256	526,62	55,653981	0,21739836	362,012	81,089300	0,31675508
320	486,405	60,255342	0,18829794	241,744	121,43134	0,37947296
384	453,164	64,675261	0,16842515	292,943	100,20823	0,26095893
448	418,568	70,020880	0,15629660	203,273	144,41317	0,32235084
512	388,889	75,364692	0,14719666	211,143	139,03042	0,27154378
640	484,496	60,492759	0,09451993	284,825	103,06433	0,16103802
768	405,554	72,267811	0,09409871	171,198	171,46987	0,22326806
1024	392,646	74,643572	0,07289411	164,562	178,38443	0,17420355

Также на рисунках 4, 5 и 6 приведены графики зависимости времени, ускорения и эффективности от числа процессов для двух алгоритмов декомпозиции. Шкала времени взята в логарифмическом масштабе для большей наглядности.

Ускорение и эффективность вычислялись с помощью известных формул:

$$Acceleration_P = \frac{Time(1)}{Time(P)}$$

$$Efficiency_P = \frac{Time(1)}{P \cdot Time(P)}$$

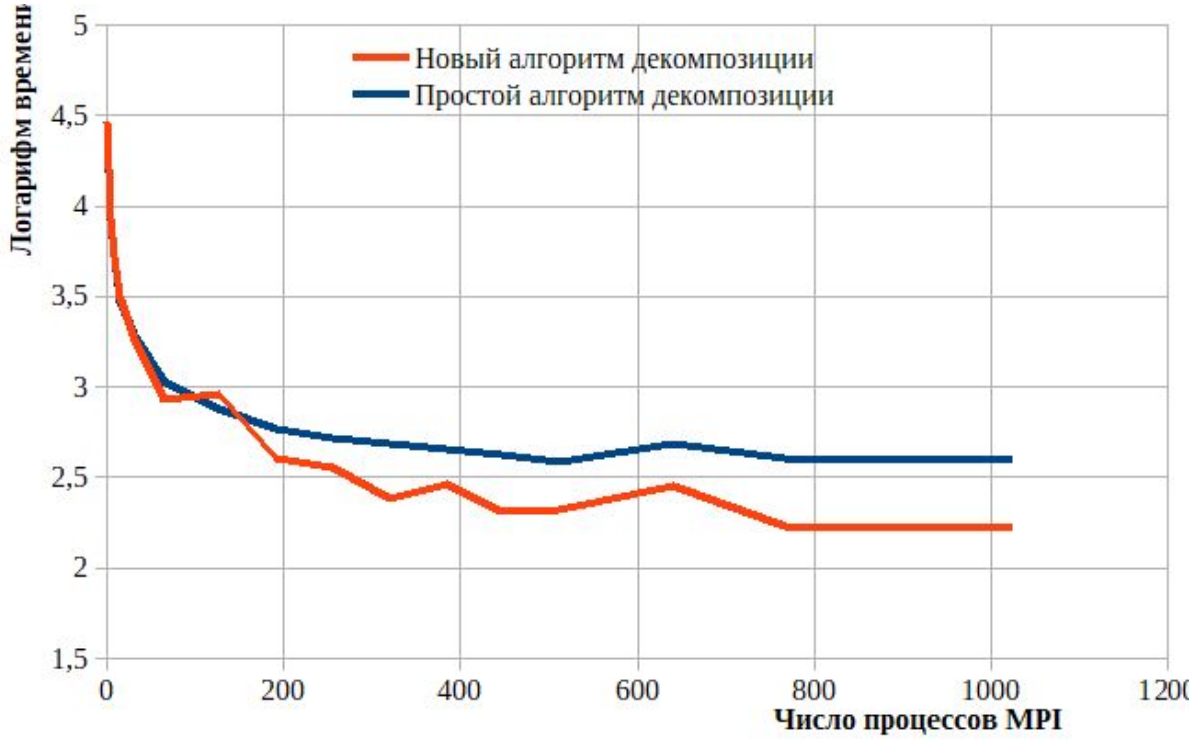


Рис. 4. График зависимости логарифма времени расчета задачи от числа процессов.

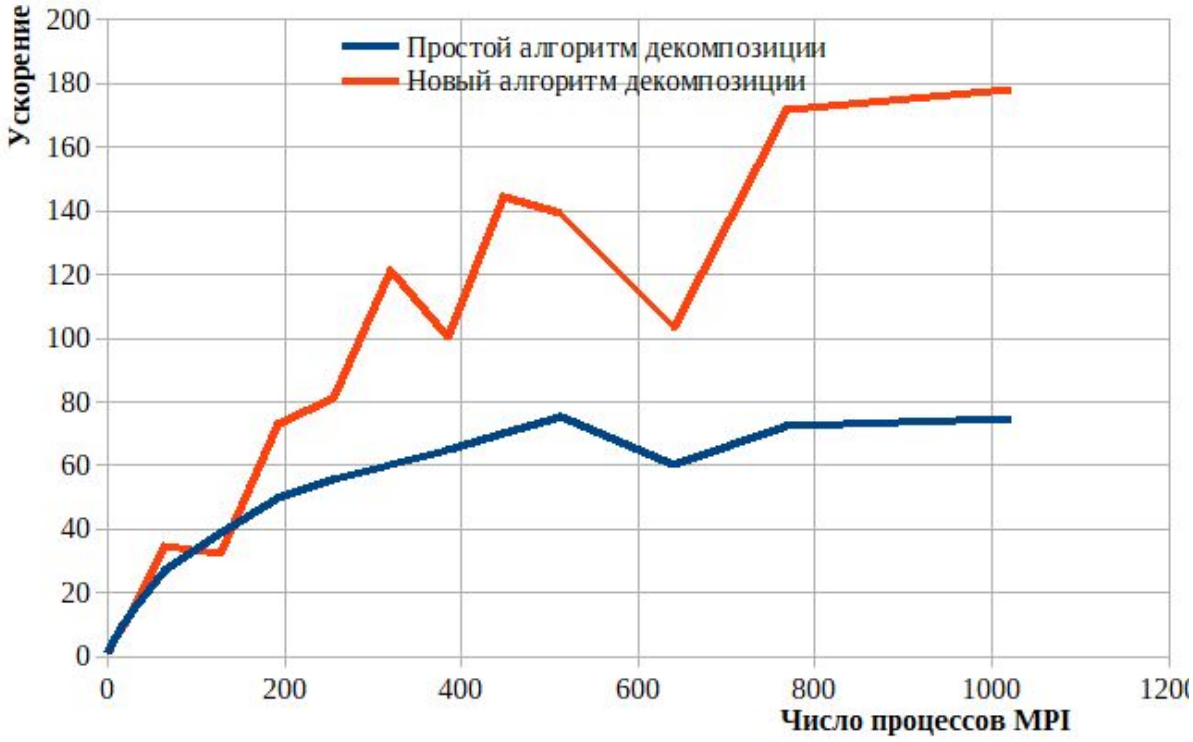


Рис. 5. График зависимости ускорения от числа процессов.

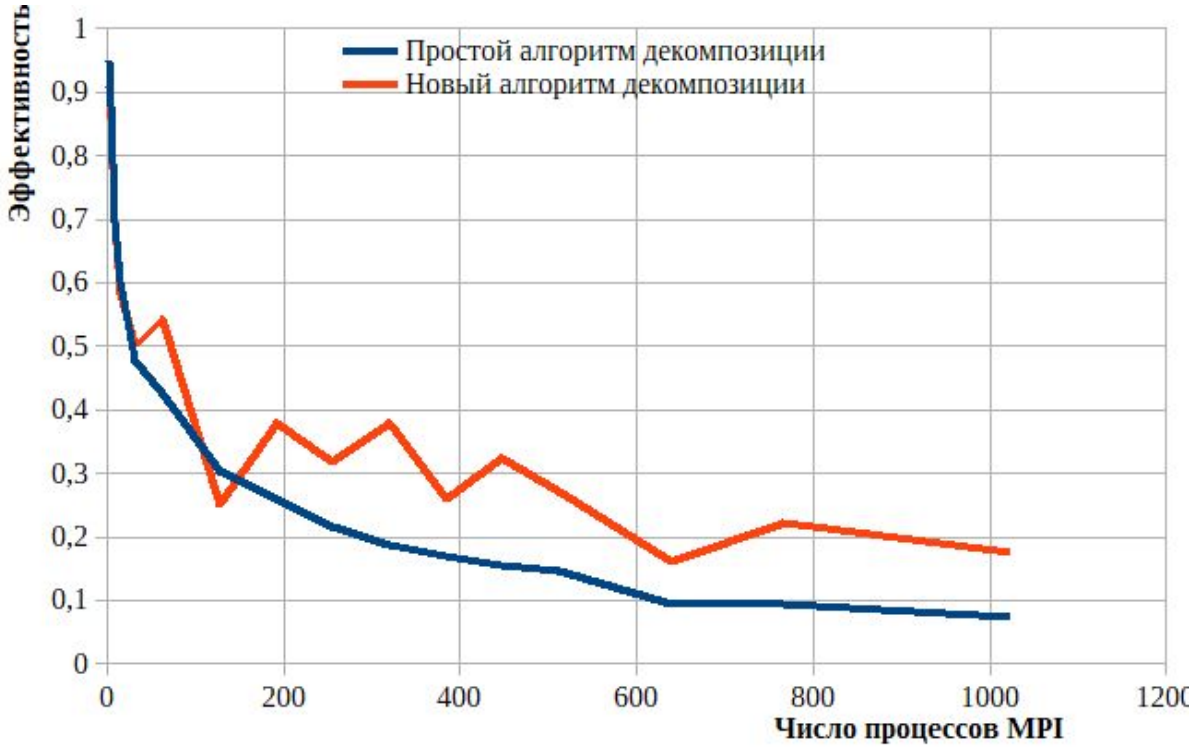


Рис. 6. График зависимости эффективности от числа процессов.

Видно, что предложенный алгоритм декомпозиции ведет себя несколько “непредсказуемо”: время расчета задачи на большем числе процессов может быть больше, чем время расчета на меньшем числе процессов. Это объясняется неравномерным распределением работы между процессами.

Также можно заметить, что с ростом числа процессов растет и выигрыш от использования описанного метода декомпозиции − времена счета на 1024 процессах отличаются более чем в 2 раза. Ускорение коммуникаций, очевидно, превосходит ускорение расчета всей задачи. На рисунках 7, 8, 9, и 10 представлены графики, отражающие ускорение коммуникаций при использовании нового алгоритма декомпозиции по сравнению с простым.

Рис. 7. График зависимости общего размера пересылаемой информации от числа процессов.

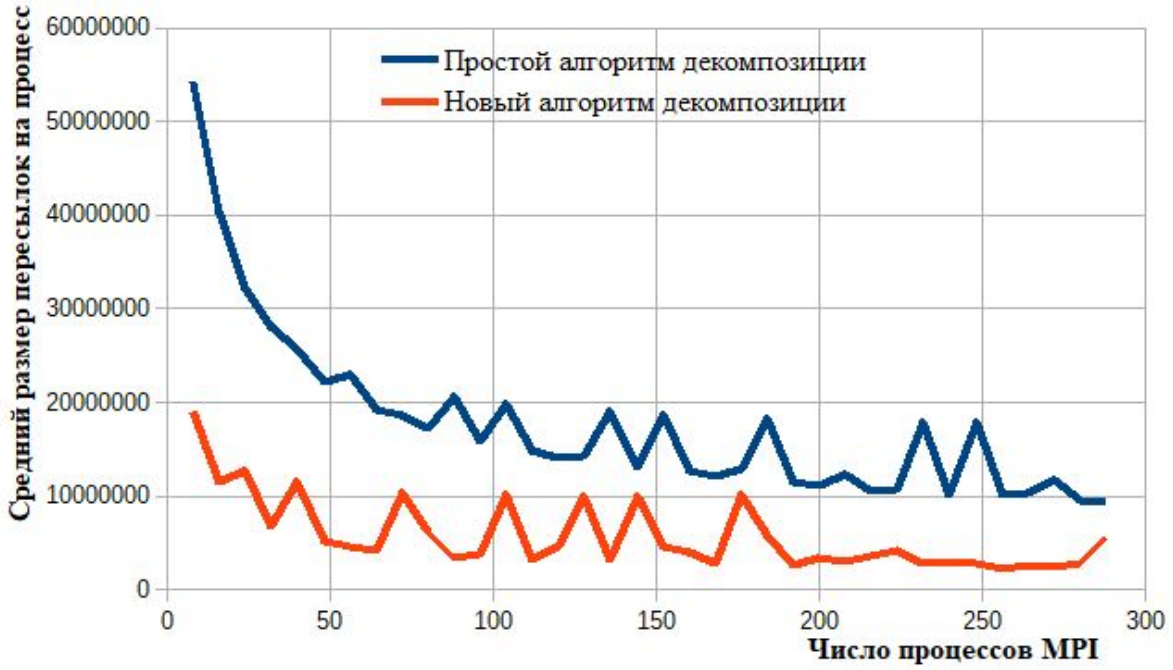


Рис. 8. График зависимости среднего размера пересылаемой информации, приходящейся на один процесс, от числа процессов.

Рис. 9. График зависимости общего числа пересылок информации от числа процессов.

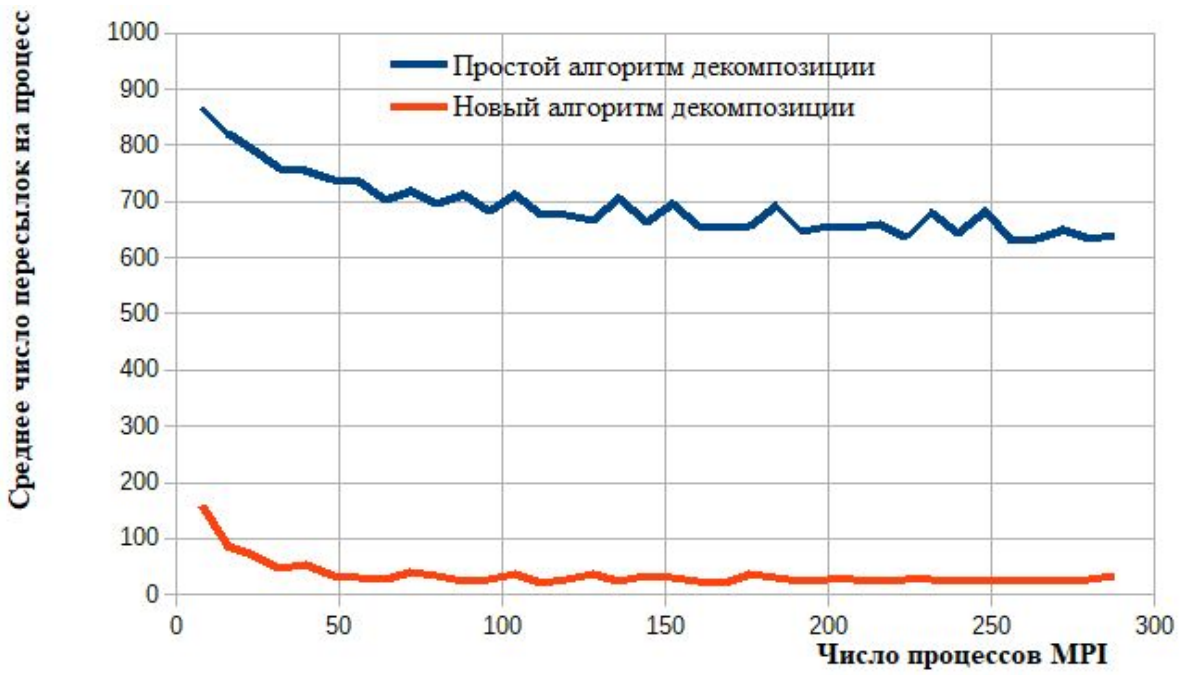


Рис. 10. График зависимости среднего числа пересылок информации, приходящихся на один процесс, от числа процессов.

Из графиков видно, что благодаря новому алгоритму декомпозиции число пересылок между процессами резко снизилось, что как раз являлось целью создания этого алгоритма. Размер пересылаемой информации также снизился, но не так сильно. Следовательно, средний размер отдельной пересылки возрос, что связано с увеличением средней площади отдельной контактной поверхности в многоблочной сетке. Рост среднего размера отдельной пересылки также является положительным эффектом данного алгоритма, так как в MPI время пересылки сообщения в зависимости от размера сообщения растет медленнее чем линейно.

Глава 3. Нахождение процессами своих соседей и обмен узлами

3.1 Архитектурные ограничения

Ниже перечислены некоторые особенности архитектуры программного комплекса RECT, которые оказали влияние на алгоритм поиска процессами соседей:

1. Наличие контакта между двумя сетками задается в конфигурационном файле, по которому строится модель области вычислений.
2. При декомпозиции сетка может быть поделена на блоки, которые будут распределены между процессами. Блоки двух разных контактирующих сеток могут быть распределены одному процессу. Например, это происходит при использовании простого алгоритма декомпозиции. Поэтому процесс может обмениваться узлами сам с собой.
3. После декомпозиции каждый процесс знает о существовании только тех сеток, которые были распределены к нему. Процесс может узнать, является ли внешняя граница его блока внешней границей для всей сетки.
4. Вызов функции поиска контактов и функции обмена контактирующими узлами происходит для каждого контакта каждого блока сетки в каждом процессе.
5. Функции поиска контактов и функции обмена контактирующими узлами занимаются только контактами между двумя разными сетками. Контакты между блоками одной сетки, распределенными к разным процессам, обсчитывают другие функции.
6. Судить о соседстве двух узлов разных сеток можно по координатам узлов − они должны совпадать.

7. В RECT используется MPI, а в MPI длина передаваемого сообщения должна быть известна заранее.

3.2 Описание алгоритма

Алгоритм поиска соседних процессов и обмена узлами с ними состоит из следующих шагов:

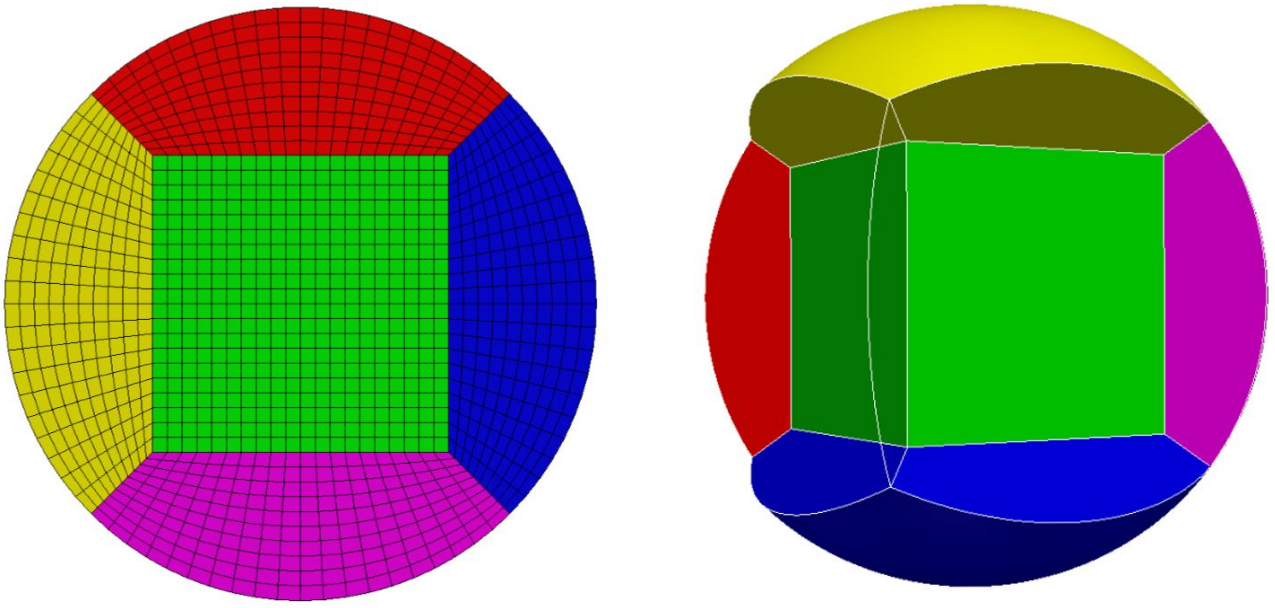
1. Процесс рассчитывает количество граничных узлов, у которых возможен контакт с другими сетками. Это число передается и принимается каждым процессом с помощью функции *MPI_Bcast()*.
2. Процесс выделяет память в соответствии с полученными данными, чтобы суметь принять нужное количество координат узлов.
3. Процесс создает буфер и заполняет его координатами граничных узлов, у которых возможен контакт с другими сетками.
4. Процесс передает и принимает созданные буферы с помощью функции *MPI_Bcast()*. Из-за особенности 7, которая была описана в предыдущем разделе, сразу этого сделать было нельзя.
5. Процесс производит поиск контактов, сопоставляя полученные от других процессов координаты своим. Под найденные контакты сразу создается буфер и выделяется память для хранения узлов соседей.
6. Далее на каждом шаге вычислений будет производиться обмен контактирующими узлами соседствующими процессами.

Все шаги кроме последнего выполняются в рамках функции поиска контактов. Последний шаг выполняется с помощью вызова функции обмена узлами на каждом шаге вычислений.

3.3 Описание тестовой задачи

Для тестирования была выбран расчет Martin Käser и Michael Dumbser [25]: распространение сейсмических волн от взрыва на глубине 171 км от поверхности Земли. Использовалась модель внутреннего строения Земли PREM.

Расчет был выполнен с помощью TVD-схемы второго порядка точности с ограничителем MC. Расчет был выполнен отдельно и для двумерной, и для трехмерной моделей Земли. Расчетная область была задана с помощью нескольких структурных криволинейных сеток, как показано на рисунке 11.



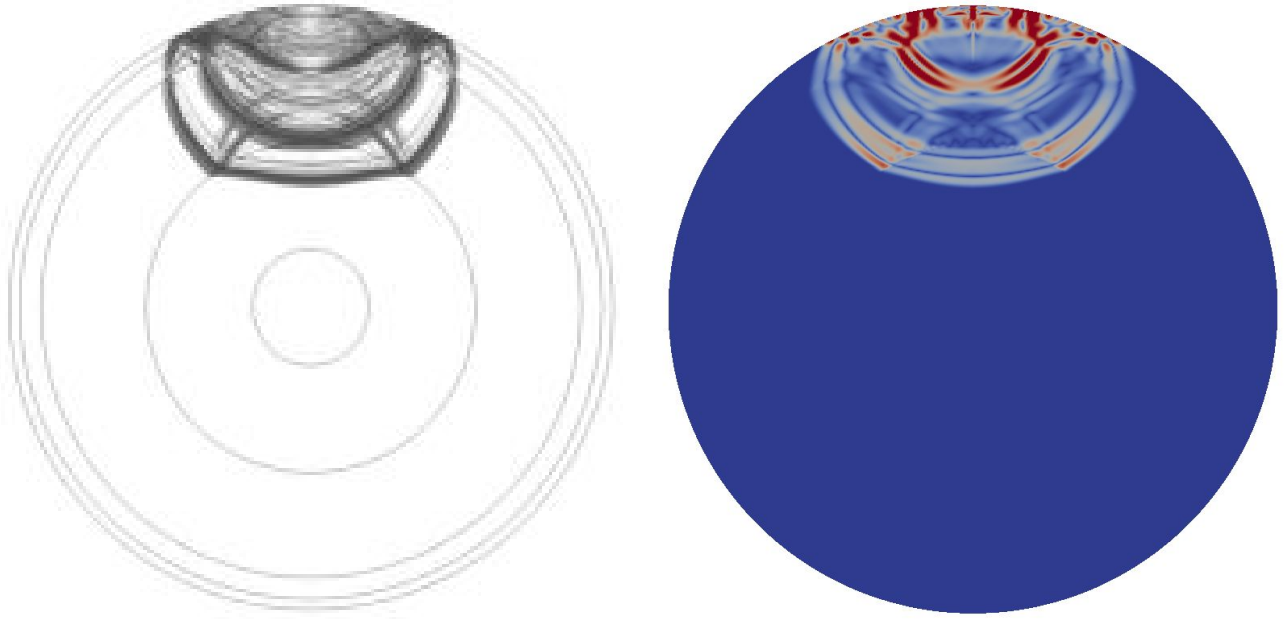
а) б)

Рис. 11. Структура расчетной области для повторения расчета Martin Käser и Michael Dumbser: а) для двумерного случая; б) для трехмерного случая.

3.4 Результаты тестирования

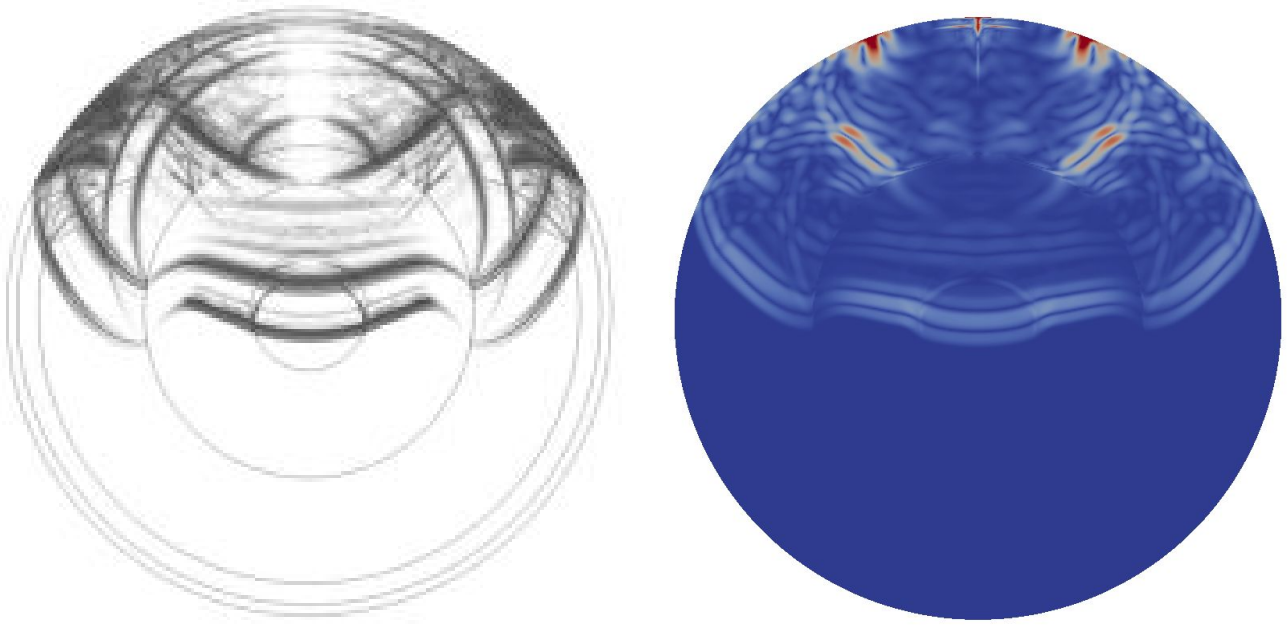
Требования к производительности к алгоритму поиска не выдвигались, так как он используется один раз на стадии подготовки к расчету и не является “бутылочным горлышком”. Оптимизация этого алгоритма не даст существенного выигрыша во времени расчета всей задачи. Поэтому критерий качества алгоритма был бинарный: работает алгоритм или не работает.

Для демонстрации работы алгоритма будем сравнивать полученные с помощью него результаты повторного расчета с результатами, полученными Martin Käser и Michael Dumbser в оригинальной работе. Результаты представлены на рисунках 12 и 13.



а) б)

Рис. 12. Распределение вертикальных скоростей в сейсмическом волновом поле при t = 300 с: а) в оригинальной работе; б) в расчете с применением нового алгоритма.



а) б)

Рис. 13. Распределение вертикальных скоростей в сейсмическом волновом поле при t = 600 с: а) в оригинальной работе; б) в расчете с применением нового алгоритма.

Картина распространения волн немного отличается от картины, полученной в оригинальной работе, из-за разных численных методов. В оригинальной работе использовался метод Галеркина для неструктурных сеток, а в тестовом расчете использовался сеточно-характеристический метод на криволинейных сетках. Каких-либо артефактов на границах контактных поверхностей получено не было, что доказывает корректность работы алгоритма.

Результаты рендера расчета задачи для трехмерной модели Земли представлены на рисунке 14. Рендер сделан с помощью модуля ParaView для Python.

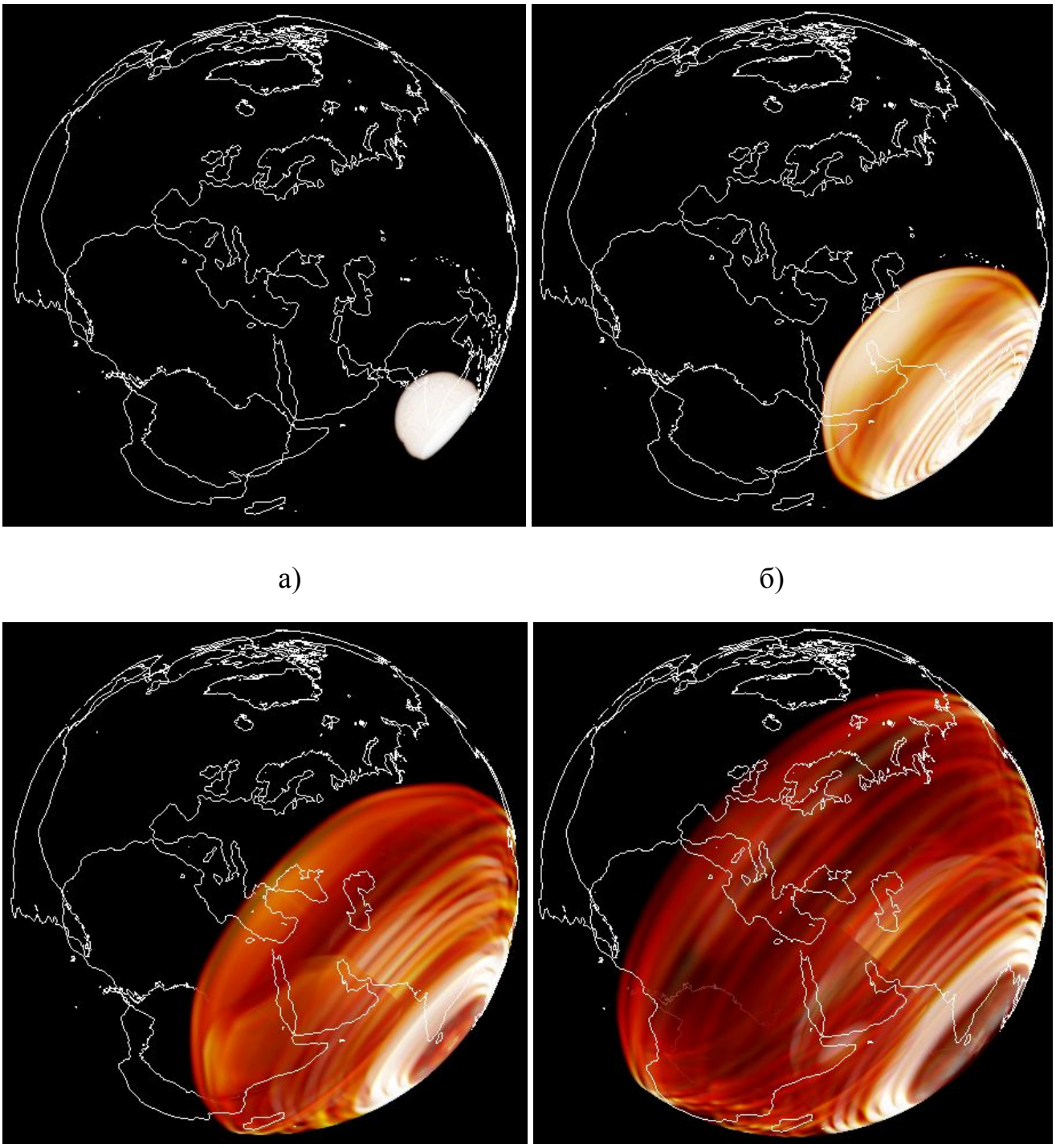


Рис. 14. Результаты рендера расчета Martin Käser и Michael Dumbser для трехмерной модели Земли: а) при t = 100 с; б) при t = 300 с; в) при t = 500 с; г) при t = 700 с.

На рисунках 14в и 14г можно заметить некоторую асимметрию волновой картины, которой не должно наблюдаться в данном расчете. Эта асимметрия является артефактом рендера, и не связана с неправильной работой алгоритма обмена контактами в трехмерном случае. Предположительно это произошло из-за неправильного порядка рендера сеток: сверху сетка, которая находится дальше от наблюдателя, рендерилась первой, а снизу наоборот. В GUI самого ParaView этих артефактов не возникает, что можно видеть на рисунке 15.

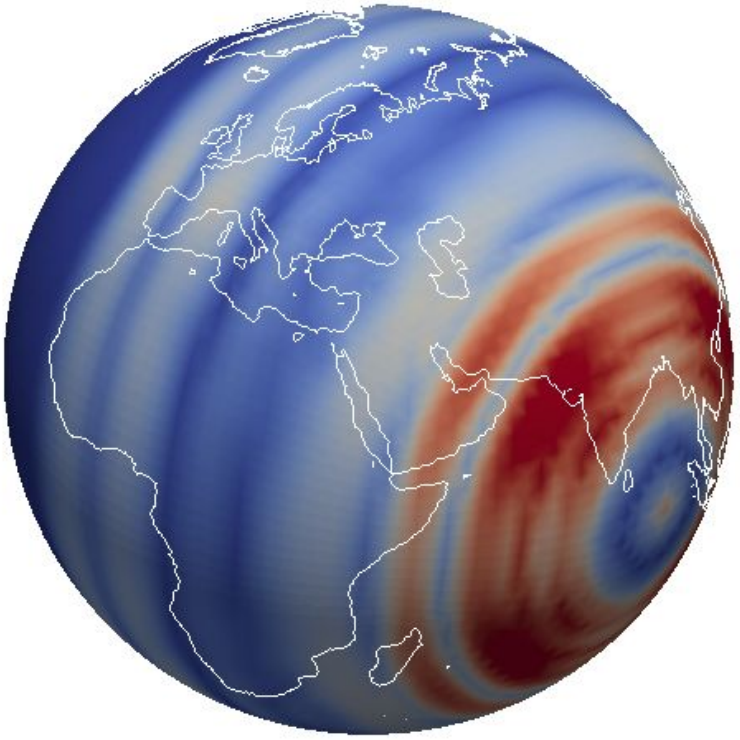


Рис. 15. Результаты расчета Martin Käser и Michael Dumbser для трехмерной модели Земли в GUI ParaView.

Заключение

В работе предложен алгоритм декомпозиции многоблочных структурных сеток для расчета на многопроцессорных вычислительных системах с распределенной памятью с применением технологии MPI. Его особенностью является сохранение структурного разбиения в рамках каждого блока. Проведена аналитическая оценка эффективности предложенного алгоритма. В качестве примера рассматривается задача прохождения динамических волновых возмущений через здание, которое представлено набором структурных сеток. Приведены результаты тестирования алгоритма на данной задаче. Представленный алгоритм позволяет сохранить преимущества структурной декомпозиции, уменьшив количество блоков. В дальнейшем планируется использовать более эффективный алгоритм группировки маленьких сеток.

Также предложен алгоритм поиска процессами своих соседей, имеющих общие границы расчетных сеток, и дальнейшего взаимодействия с соседями. Приведены результаты работы алгоритма на примере расчета Martin Käser и Michael Dumbser.

Список литературы

1. Четверушкин Б. Н. Кинетические схемы и квазигазодинамическая система уравнений. – МАКС Пресс, 2004. – 328 с.
2. Basermann A., Clinckemaillie J., Coupez T. et al. Dynamic load-balancing of finite element applications with the DRAMA library // Applied Mathematical Modelling. — 2000. — Vol. 25, no. 2. — P. 83–98.
3. Devine K., Boman E., Heaphy R. et al. Zoltan data management services for parallel dynamic applications // Computing in Science & Engineering. — 2002. — Vol. 4, no. 2. — P. 90–96.
4. Якобовский М. В. Распределенные системы и сети //М.: Изд-во МГТУ «Станкин». – 2000.
5. Головченко Е. Н. Декомпозиция расчетных сеток для решения задач механики сплошных сред на высокопроизводительных вычислительных системах: дисс. … канд. физ.-мат. наук: 05.13.18 – М., 2014. – 165 с.
6. Fiduccia C. M., Mattheyses R. M. A linear-time heuristic for improving network partitions // Design Automation, 1982. 19th Conference on / IEEE. — 1982. — P. 175–181.
7. Якобовский М. В. Инкрементный алгоритм декомпозиции графов // Вестник Нижегородского университета им. Н.И. Лобачевского. Серия: Математическое моделирование и оптимальное управление. – 2005. – №. 1. – С. 243.
8. Preis R., Diekmann R. PARTY – a software library for graph partitioning // Advances in Computational Mechanics with Parallel and Distributed Processing. — 1997. — P. 63–71.

9. Walshaw C., Cross M. JOSTLE: parallel multilevel graph-partitioning software – an overview // Mesh partitioning techniques and domain decomposition techniques. — 2007. — P. 27–58.
10. Karypis G. — A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Version 5.1.0. — Department of Computer Science and Engineering, University of Minnesota, Minneapolis, 2013. — March 30.
11. Hendrickson B., Leland R. — The Chaco user's guide: Version 2.0. — Sandia National Laboratories, 1995.
12. Головченко Е. Н., Якобовский М. В. Пакет параллельной декомпозиции больших сеток GridSpiderPar //вычислительные методы и программирование. – 2015. – Т. 16. – №. 4. – С. 507-517.
13. Беликов Д. А., Старченко А. В. Распараллеливание пространственной модели переноса примеси на системах с распределенной памятью с помощью одноИ двумерной декомпозиции области // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. – 2009. – №. 17 (150).
14. Бурцев А. П., Попова Н. Н., Курин Е. А. Моделирование волновых процессов в задачах сейсморазведки на суперкомпьютере «Ломоносов» с использованием GPU //Параллельные вычислительные технологии (ПАВТ'2014). – 2014. – С. 225-228.
15. Бастраков С. И. и др. Исследование и поиск наиболее эффективных подходов к параллельному моделированию плазмы методом частиц в ячейках на кластерных системах //Тр. Междунар. конф.“Параллельные вычислительные технологии. – 2011. – С. 411-417.
16. Luong P., Breshears C. P., Ly L. N. Coastal ocean modeling of the US west coast with multiblock grid and dual-level parallelism // Proceedings of the 2001 ACM/IEEE conference on Supercomputing / ACM. — 2001. — P. 9.

17. Luong P., Breshears C. P., Ly L. N. Application of multiblock grid and dual-level parallelism in coastal ocean circulation modeling // Journal of Scientific Computing. — 2004. — Vol. 20, no. 2. — P. 257–275.
18. Djomehri M. J., Biswas R. Performance analysis of a hybrid overset multi-block application on multiple architectures // High Performance Computing-HiPC 2003. — Springer, 2003a. — P. 383–392.
19. Djomehri M. J., Biswas R. Performance enhancement strategies for multi-block overset grid CFD applications // Parallel Computing. — 2003b. — Vol. 29, no. 11. — P. 1791–1810.
20. Djomehri M. J., Biswas R., Lopez-Benitez N. Load balancing strategies for multi-block overset grid applications: NAS Technical Report NAS-03-007 / Djomehri M. J., Biswas R., LopezBenitez N.: NASA Ames Research Center, 2003.
21. Голубев В. И., Петров И. Б., Хохлов Н. И., Шульц К. И. Численный расчет волновых процессов в трещиноватых средах на гексаэдральных сетках сеточно-характеристическим методом. – Журнал вычислительной математики и математической физики. – 2015. – Т. 55. – № 3. – С. 512-522.
22. Петров И. Б., Хохлов Н. И. Моделирование задач 3D сейсмики на высокопроизводительных вычислительных системах. – Математическое моделирование. – 2014. – Т. 26. – № 1. – С. 83-95.
23. Григорьевых Д. П., Хохлов Н. И., Петров И. Б. Расчет динамического разрушения в твердых деформируемых телах // Математическое моделирование. – 2017. – Т. 29. – №. 4. – С. 45-58.
24. Roe P. L. Characteristic-based schemes for the Euler equations //Annual review of fluid mechanics. – 1986. – Т. 18. – №. 1. – С. 337-365.
25. Käser M., Dumbser M. An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes—I. The two-dimensional isotropic case

with external source terms //Geophysical Journal International. – 2006. – Т. 166. – №. 2. – С. 855-877.