УДК 004.021

КОМПЛЕКС ПРОГРАММ ПАРАЛЛЕЛЬНОЙ ДЕКОМПОЗИЦИИ СЕТОК

Е. Н. Головченко[1]

Численное решение задач математической физики на распределенных вычислительных системах зачастую предполагает использование геометрического параллелизма. В результате возникает задача сбалансированного распределения сетки между процессорами, сводящаяся к задаче разбиения графа на домены. Целью исследования настоящей статьи является параллельная декомпозиция треугольных и тетраэдральных сеток большого размера. На основе последовательного инкрементного алгоритма декомпозиции графов, обеспечивающего формирование компактных доменов, и алгоритма рекурсивной координатной бисекции создан комплекс программ параллельной декомпозиции сеток. Работа выполнена при финансовой поддержке РФФИ (коды проектов 05–01–00750-а, 08–07–00458-а, 09–01–12022–офи_м). Статья рекомендована к печати программным комитетом Международной научной конференции “Научный сервис в сети Интернет: суперкомпьютерные центры и задачи” (http://agora.guru.ru/abrau).

Ключевые слова: разбиение графов, декомпозиция сеток, параллельные вычисления.

1. Введение. Численное решение задач математической физики на распределенных вычислительных системах зачастую предполагает использование геометрического параллелизма, при котором сетка, аппроксимирующая расчетную область, распределяется между процессорами по геометрическому признаку. В дальнейшем каждый процессор обрабатывает свою часть сетки. Сбалансированность загрузки процессоров определяется равномерностью распределения сетки по процессорам и затратами на передачу данных между процессорами. Объем передач данных между процессорами зависит от числа связей между доменами (частями сеток), распределенными по процессорам.

Задача сбалансированного разбиения сетки на домены сводится к более общей задаче разбиения графа на домены, при этом обрабатывается граф, соответствующий коммуникационным нагрузкам сетки. Вопросы построения такого графа выходят за рамки данной статьи и в ней не рассматриваются. Существует несколько моделей декомпозиции графов [1], отличающиеся построением графа и критериями сбалансированного разбиения. В случае разбиения сеток хорошо себя зарекомендовал наиболее распространенный подход, при котором сетка аппроксимируется неориентированным графом $G = (V, E)$, где V — множество вершин и E — множество ребер, причем и вершины, и ребра имеют вес. Оптимальным считается разбиение на домены, при котором выравнен суммарный вес вершин в доменах и минимизирован суммарный вес разрезанных ребер между доменами. В этой модели суммарный вес вершин в доменах отвечает за равномерность разбиения сетки на домены (в дальнейшем — за равномерность распределения по процессорам, которые будут обрабатывать эти домены), а суммарный вес разрезанных ребер — за коммуникационную нагрузку между процессорами.

Как известно, указанная задача декомпозиции графа является NP-полной, поэтому для ее решения используются различные эвристические методы: алгоритмы рекурсивных координатной бисекции, инерциальной бисекции, спектральной бисекции, алгоритм Kernighan–Lin (KL) и Fiduccia–Mattheyses (FM), “жадный” алгоритм (greedy method), иерархические алгоритмы. Эти алгоритмы реализованы в следующих пакетах декомпозиции графов: METIS, CHACO, JOSTLE и SCHOTCH. Однако параллельными из них являются только PARMETIS (параллельная версия пакета METIS) и JOSTLE.

В нашем исследовании сделан акцент на декомпозиции больших графов, которые невозможно разместить в памяти одного процессора, следовательно, нужен именно параллельный алгоритм. Методы пакетов PARMETIS и JOSTLE основываются на иерархических алгоритмах [2, 3], состоящих из следующих частей: поэтапное огрубление графа, декомпозиция самого маленького из полученных графов и отображение разбиения на предыдущие графы с периодическим локальным уточнением границ доменов. Недостатком таких алгоритмов является образование некомпактных доменов. Зачастую некомпактные домены связаны с большим числом соседних доменов, что ведет к большему объему передач данных на процессорах, обрабатывающих эти домены.

[1] Институт прикладной математики им. М. В. Келдыша РАН, Миусская пл., д. 4а, 125047, Москва; мл. науч. сотр., e-mail: ge03@imamod.ru

Инкрементный алгоритм декомпозиции графов [4], созданный и реализованный М. В. Якобовским, обеспечивает формирование связных доменов. На его основе разработан параллельный инкрементный алгоритм декомпозиции графов. Алгоритм нацелен на декомпозицию больших графов, не помещающихся на одном процессоре (до 10^9 вершин), на компактные домены. Область разбиваемых графов сужена до графов, построенных на треугольных и тетраэдральных сетках.

Разработанный параллельный инкрементный алгоритм декомпозиции графов оказался чувствительным к первоначальному распределению вершин по процессорам. Поэтому для выполнения начального распределения вершин разработан алгоритм геометрической декомпозиции сеточных данных.

На основе рассмотренных алгоритмов создан комплекс программ параллельной декомпозиции сеток.

2. Инкрементный алгоритм декомпозиции графов. В начале выполнения инкрементного алгоритма декомпозиции графов [4] все вершины считаются свободными, т.е. не распределенными по доменам. Затем для каждого домена случайным образом выбирается вершина, которая помещается в этот домен и становится его инициализирующей вершиной. Далее разбиение производится посредством итерационного процесса, на каждом шаге которого выполняются следующие действия.

1. Инкрементный рост доменов (захват вершин). На этом этапе происходит постепенное присоединение к доменам прилегающих к ним свободных вершин и диффузное перераспределение вершин на границах между доменами с целью выравнивания суммарного веса вершин в доменах. В конце этапа все вершины должны быть распределены по доменам.

2. Локальное уточнение доменов. Выполняется алгоритм KL/FM локального уточнения [5].

3. Проверка качества доменов. Если качество доменов соответствует заданному, разбиение считается найденным и происходит выход из цикла, иначе — переход к следующему этапу.

4. Освобождение части вершин плохих доменов и переход к первому этапу. Плохими доменами считаются домены, качество которых не соответствует заданному, и их соседи. В плохих доменах часть вершин освобождается, т.е. вновь считается нераспределенной.

Качество доменов проверяется следующим образом. Все вершины, расположенные на геометрической границе графа и на границах между доменами, считаются принадлежащими первой оболочке своего домена. В каждом домене вершинами второй оболочки считаются соседи вершин из первой оболочки данного домена, которые также принадлежат этому домену и не попали в первую оболочку. Остальные оболочки вычисляются аналогично. Проверяется связность оболочек каждого домена и вычисляется номер наименьшей несвязной оболочки в каждом домене. Хорошими считаются те домены, номер наименьшей несвязной оболочки в которых больше либо равен пороговому значению.

3. Параллельный инкрементный алгоритм декомпозиции графов. Прямое распараллеливание всех этапов инкрементного алгоритма декомпозиции графов невозможно, поскольку этапы инкрементного роста и локального уточнения доменов последовательны по своей природе. В них требуется периодическое обновление информации, которое в параллельном варианте приведет к большим коммуникационным издержкам. Поэтому выбрана следующая стратегия распараллеливания. Каждый процессор обрабатывает свой блок вершин локально инкрементным алгоритмом декомпозиции графов, что позволяет избежать передач данных между процессорами. Недостатком локального разбиения является то, что группы не могут расширяться за пределы процессора, что ограничивает число просматриваемых хороших вариантов разбиения. В результате на границах между процессорами могут образоваться плохие группы доменов. Для устранения этого недостатка впоследствии плохие группы доменов перераспределяются между процессорами так, чтобы каждая группа плохих доменов оказалась собранной на одном процессоре. Затем происходит локальное переразбиение плохих групп доменов.

В рассматриваемой модели предполагается, что после локального разбиения инкрементным алгоритмом декомпозиции графов плохими окажутся не все домены на процессорах, а только небольшая их часть на границах между процессорами. Это утверждение выполняется, когда на процессорах образовывается достаточно большое количество доменов. На практике число процессоров, на которых будет считаться задача, заранее не известно, и выгоднее сразу разбить граф на большое количество микродоменов, чтобы в дальнейшем распределять эти домены по нужному числу процессоров. Количество микродоменов на несколько порядков меньше числа вершин, поэтому многократное распределение микродоменов по нужному числу процессоров быстрее многократного разбиения всего графа. Таким образом, предположение о том, что количество доменов будет достаточно велико, является оправданным.

Существует несколько подходов к перемещению вершин между доменами. Именно в них состоит основное отличие между иерархическими алгоритмами PARMETIS и JOSTLE. В PARMETIS используется виртуальное перемещение вершин между доменами, при котором вершины остаются на том же процессоре, меняется только значение поля, отвечающего за домен, которому принадлежит вершина. В JOSTLE

каждый процессор отвечает за один домен, и если вершина из этого домена переносится в другой домен, то она передается на другой процессор, отвечающий за новый домен. Поскольку реальная передача вершин между доменами предполагает коммуникационные издержки и не позволяет использовать алгоритм локально, в параллельном инкрементном алгоритме декомпозиции графов используется виртуальная передача вершин.

Как и в любом параллельном алгоритме декомпозиции графов, предполагается, что вершины изначально неким образом распределены по процессорам. Здесь возникает задача в задаче, когда для нахождения разбиения следует изначально разбить граф. В качестве начального разбиения обычно берут более простое разбиение, например разбиение по номерам вершин или геометрическое разбиение. Ограничение на расширение доменов за пределы процессоров позволило исключить синхронизацию между процессорами. Однако в результате наложения данного ограничения возникла зависимость локального алгоритма инкрементного роста доменов от начального распределения вершин по процессорам. Если при начальном распределении на процессор попадет множество мелких групп вершин, то после выполнения алгоритма инкрементного роста некоторые группы вершин окажутся не распределенными по доменам. Поэтому для получения относительно компактного начального распределения вершин по процессорам целесообразно использовать геометрическое разбиение. Подробнее о геометрическом разбиении написано ниже.

После геометрического разбиения на процессор должен попасть такой общий вес вершин, какой будет в образуемых на нем доменах, что позволит сформировать домены одинакового веса.

Однако геометрическое разбиение не гарантирует, что на процессор вообще не попадут мелкие группы вершин. Например, если граф описывает сетку, содержащую в себе отверстия, то при геометрическом разбиении на процессор может попасть основной блок вершин с одной стороны отверстия и небольшая группа вершин с другой стороны. Таким образом, даже после геометрической декомпозиции необходимо предварительное перераспределение малых блоков вершин между процессорами.

Учитывая все сказанное, параллельный инкрементный алгоритм декомпозиции графов выглядит следующим образом.

1. Распределение вершин по процессорам в соответствии с результатами геометрической декомпозиции. Общий вес вершин на процессорах определяется количеством будущих доменов.

2. Присоединение малых блоков вершин к большим и восстановление исходного веса вершин на процессорах.

3. Инициализация доменов. Поскольку на процессоре может оказаться несколько больших блоков вершин, сначала оценивается вес вершин в блоках, в соответствии с ним определяется, сколько доменов будет образовано в каждом из блоков, а затем случайным образом выбирается нужное количество инициализирующих вершин в каждом из блоков.

4. Локальное разбиение вершин на домены инкрементным алгоритмом декомпозиции графов.

5. Перераспределение плохих групп доменов. Сбор каждой группы плохих доменов на одном процессоре.

6. Локальное переразбиение плохих групп доменов инкрементным алгоритмом декомпозиции графов.

В соответствии с данным алгоритмом был создан комплекс программ, осуществляющий параллельную инкрементную декомпозицию графов.

При считывании данных использовалась библиотека распределенного хранения и обработки тетраэдральных сеток (Суков С.А., ИПМ им. М.В. Келдыша РАН).

4. Алгоритм геометрической декомпозиции сеточных данных. Основным пакетом в свободном доступе, используемым для параллельной декомпозиции графов, является пакет PARMETIS. Данный пакет обладает следующими недостатками: первое — при разбиении графа на большое число частей образуются пустые домены, второе — при разбиении на равные части формируются домены с количеством узлов, отличающимся от среднего арифметического в несколько раз. Поскольку после предварительной декомпозиции графа на процессоре должно оказаться столько вершин, сколько вершин будет в образуемых на нем доменах (если считать веса вершин, равными единице), вторая проблема является критичной. Поэтому был разработан алгоритм и создан комплекс программ, осуществляющий геометрическую декомпозицию тетраэдральных и треугольных сеток.

Алгоритм геометрической декомпозиции основывается на методе рекурсивной координатной бисекции. Алгоритм работает только с координатами вершин и не учитывает связи между ними, что делает его экономичным по памяти.

При начальном распределении узлов на процессор попадает столько узлов, сколько будет в образуемых на нем доменах. Начальное распределение может быть любым, например в соответствии с порядковыми номерами узлов. Затем узлы сортируются по одной из координат. Локально на каждом про-

цессоре данные сортируются либо сортировкой слиянием, либо пирамидальной, либо оптимизированной сортировкой (гибрид сортировки слиянием и пирамидальной сортировки) [6, 7]. В качестве параллельной сортировки была взята битонная сортировка Бэтчера [6, 7]. Для сортировки данных использовалась библиотека параллельной сортировки из http://lira.imamod.ru/FondProgramm/Sort/ (Якобовский М.В., ИПМ им. М.В. Келдыша РАН). В процессе сортировки, а также после нее число узлов на процессорах сохраняется. В результате блок вершин оказывается разбитым на две части по одной из координат. Далее группа процессоров делится на две, каждая из групп вновь проводит параллельную сортировку по одной из координат и делит вершины на две группы. Рекурсивная параллельная сортировка увеличивает время работы алгоритма, но улучшает качество распределения блоков вершин по процессорам в сравнении с однократным разделением общего блока вершин между процессорами по одной из координат. Поскольку количество памяти, необходимое для работы данного алгоритма, невелико (сетка из 2×10^8 вершин разбивалась на 40 процессорах, 512 Мб на каждом), такой вариант приемлем.

Далее разбиение на домены производится локально на каждом процессоре. Отличие от обычного метода рекурсивной координатной бисекции состоит в следующем. После каждого разбиения производится разделение данных по аналогии с обменной сортировкой с разделением по Хоару (быстрая сортировка) [6], благодаря чему на каждом шаге просматриваются только узлы из разбиваемой области. Выполняется сортировка не всего разбиваемого блока вершин, а только заранее определенного интервала, в котором находится медиана. Разбиение производится одновременно по нескольким координатам, вершины сортируются сначала по одной координате, потом внутри нее по следующей координате в циклическом порядке и т.д., что позволяет обрабатывать ситуации наличия нескольких узлов с одним значением координаты.

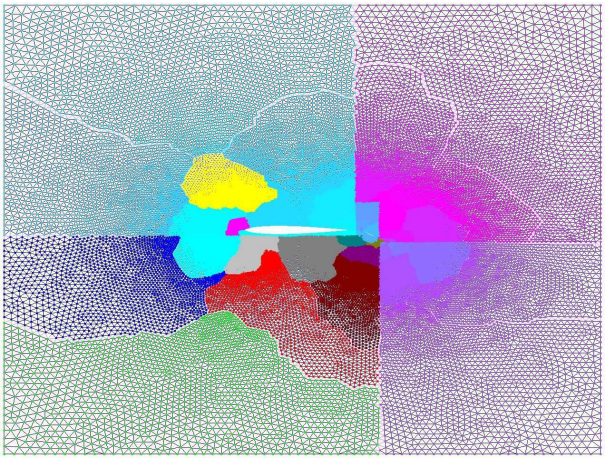

Рис. 1. Результат локального разбиения сетки из 75 790 вершин на 50 доменов на 5 процессорах

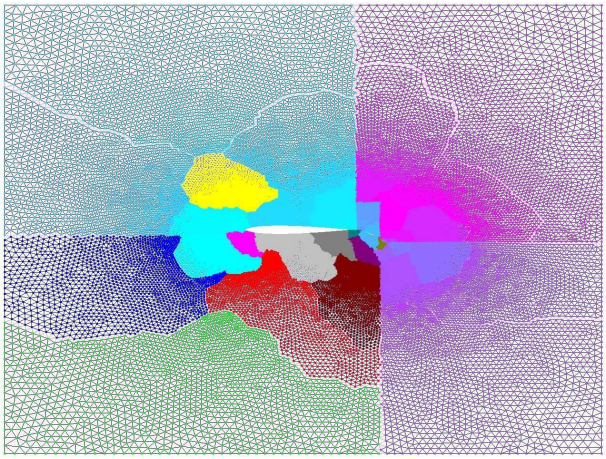

Рис. 2. Результат сбора плохих групп доменов и их повторного разбиения

5. Результаты геометрической декомпозиции. Вычисления производились на кластере МВС-100К (990 модулей по два четырехъядерных процессора, 95 TFlop/s).

На разбиение тетраэдральной сетки, содержащей 209 028 730 узлов, на 80 000 доменов на 40 процессорах потребовалось 69 секунд (без учета ввода-вывода). Из них 67 секунд — предварительная рекурсивная сортировка, 2 секунды — локальное разбиение. Результирующее разбиение содержало домены из 2612 и 2613 узлов, среднее число связей с соседними доменами — 14, количество некомпактных областей — 229.

Та же самая сетка была разбита пакетом PARMETIS на 20 000 доменов. Разбиение данным пакетом на большее число доменов не имеет смысла из-за формирования пустых доменов. Время разбиения без учета ввода-вывода — 10 секунд. В результирующем разбиении присутствовали домены с числом узлов 10 932 и 2 328, что отличается от среднего арифметического в 4 раза. Среднее число связей с соседними доменами — 14, число некомпактных областей — 364.

Большое число доменов использовалось для того, чтобы показать, что алгоритм геометрической декомпозиции и сам по себе может производить довольно качественное разбиение за разумное время.

6. Результаты параллельного инкрементного алгоритма декомпозиции графов. Результаты параллельного инкрементного алгоритма декомпозиции графов получены на двух небольших сетках (10^4–10^5 вершин) и на сетке 2×10^8 вершин. Вычисления на первых двух сетках производились на кластере СКИФ МГУ (1250 четырехъядерных процессоров, 60 TFlop/s), а на последней сетке — на кластере МВС-100К (990 модулей по два четырехъядерных процессора, 95 TFlop/s).

На рис. 1 представлено локальное разбиение треугольной сетки, состоящей из 75790 вершин, на 50 доменов на 5 процессорах, а на рис. 2 — результат сбора плохих групп доменов и их повторного разбиения. Время разбиения 16 секунд (без учета ввода-вывода). Достигнута связность до седьмой оболочки включительно. Минимальное и максимальное число вершин в доменах — 1511 и 1519 соответственно. Число разрезанных ребер 3491.

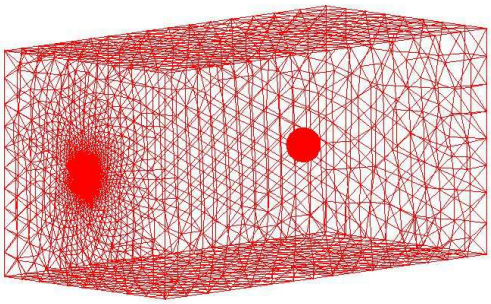

Рис. 3. Тетраэдральная сетка из 546 266 вершин

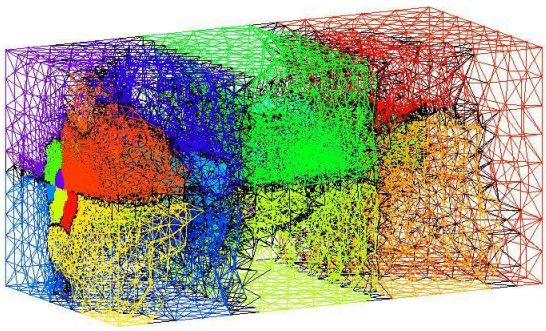

Рис. 4. Результат локального разбиения тетраэдральной сетки из 546 266 вершин на 100 доменов на 5 процессорах

На рис. 3 представлена тетраэдральная сетка, состоящая из 546 266 вершин и 3 214 552 тетраэдров. Изображены только вершины на границах сетки и связи между ними. На рис. 4 отображен результат ее локального разбиения на 100 доменов на 5 процессорах. Изображены только вершины на границе сетки и между доменами и связи между ними. Сбор плохих доменов не понадобился. Время разбиения 28 секунд. Достигнута связность до второй оболочки включительно. Минимальное и максимальное число вершин в доменах — 5458 и 5467 соответственно. Число разрезанных ребер равно 161 598.

Тетраэдральная сетка, состоящая из 209 028 730 вершин и 1 244 316 672 тетраэдров, была разбита на 40 000 доменов на 700 процессорах. Время разбиения 2970 секунд (без учета ввода-вывода). Минимальное и максимальное число вершин в доменах — 4980 и 5356 соответственно. Число разрезанных ребер равно 135 863 627. Число доменов, в которых не достигнута связность до третьей оболочки включительно, равно 10, несвязный домен только один.

Время разбиения сетки из 2×10^8 вершин довольно большое. Основные затраты идут на алгоритм локального уточнения. Время его выполнения $O\big((s-1)m\big)$ [5], где s — число доменов и m — число ребер в графе. В рассматриваемом случае на каждом процессоре располагалось около 3×10^5 вершин и 57 доменов. В дальнейшем предполагается ускорение алгоритма локального уточнения путем разбиения доменов на группы и поочередного уточнения доменов в рамках каждой группы.

7. Заключение. Разработан алгоритм и создан комплекс программ, осуществляющий геометрическую декомпозицию тетраэдральных и треугольных сеток большого размера. Результирующее разбиение тетраэдральной сетки, содержащей 2×10^8 узлов, соответствует равномерному распределению вершин по доменам (разница в один узел) по сравнению с разбиением, полученным пакетом PARMETIS, где количество узлов в некоторых доменах отличалось в 4 раза от среднего арифметического. Среднее число связей с соседними доменами такое же, как в разбиении пакетом PARMETIS. Число некомпактных областей — 229 из 80 000, что меньше 364 из 20 000, полученных PARMETIS. Эти результаты позволяют сделать вывод, что созданный комплекс программ выполняет за разумное время разбиение лучшего качества, чем пакет PARMETIS, и может быть использован как в качестве средства предварительного разбиения, так и в качестве самостоятельного пакета декомпозиции сеток.

На основе последовательного инкрементного алгоритма декомпозиции графов [4] разработан параллельный инкрементный алгоритм декомпозиции графов, ставящий своей целью формирование компактных доменов при декомпозиции треугольных и тетраэдральных сеток большого размера (до 10^9 вершин). На его основе создан комплекс программ, который протестирован на нескольких треугольных и тетраэдральных сетках. Как показали результаты, алгоритм производит достаточно сбалансированное разбиение: при разбиении тетраэдральной сетки, состоящей из 2×10^8 вершин, на 40 000 доменов минимальное число вершин в доменах равно 4980, максимальное — 5356. Разбиение обладает хорошим качеством относительно связности доменов: только один некомпактный домен на 40 000 доменов. Таким образом, созданный комплекс программ осуществляет сбалансированное разбиение лучшего качества, чем пакет PARMETIS,

но пока значительно проигрывает ему по времени.

СПИСОК ЛИТЕРАТУРЫ

1. *Hendrickson B., Kolda T.G.* Graph partitioning models for parallel computing // Parallel Computing. 2000. **26**. 1519–1534.
2. *Karypis G., Kumar V.* A Parallel algorithm for multilevel graph partitioning and sparse matrix ordering // J. of Parallel and Distributed Computing. 1998. **48**. 71–95.
3. *Walshaw C., Cross M.* Parallel optimization algorithms for multilevel mesh partitioning // Parallel Computing. 2000. **26**. 1635–1660.
4. *Якобовский М.В.* Инкрементный алгоритм декомпозиции графов // Вестник Нижегородского университета им. Н.И. Лобачевского. Серия “Математическое моделирование и оптимальное управление”. Вып. 1(28). Нижний Новгород: Издательство ННГУ, 2005. 243–250.
5. *Hendrickson B., Leland R.* A multilevel algorithm for partitioning graphs // Technical Report SAND93-1301. Sandia National Laboratories. Albuquerque, 1993.
6. *Кнут Д.Э.* Искусство программирования. Сортировка и поиск. Т. 3. М.: Издательский дом “Вильямс”, 2001.
7. *Якобовский М.В.* Параллельные алгоритмы сортировки больших объемов данных // Фундаментальные физико-математические проблемы и моделирование технико-технологических систем. Вып. 7. М.: Янус-К, 2004. 235–249.

Поступила в редакцию
20.10.2010