УДК 519.683.4

БАЛАНСИРОВКА ЗАГРУЖЕННОСТИ УЗЛОВ КЛАСТЕРА ПРИ РАСЧЕТЕ ЗАДАЧИ ФИЛЬТРАЦИИ

К. Ю. Богачев[1], А. А. Климовский[1], А. Р. Миргасимов[1], А. Е. Семенко[1]

Рассматривается задача балансировки загруженности узлов системы с распределенной памятью при расчете задачи фильтрации вязкой сжимаемой жидкости. Традиционным разделением исходных данных между узлами является разбиение их на равные части. В силу специфики задачи (геологическое строение месторождения) такое разделение не является равномерным с точки зрения загруженности узлов при расчете задачи. Обсуждается подход, в котором после загрузки исходных данных и определения геологического строения месторождения осуществляется повторное разделение расчетной области на основании уже полученной информации для обеспечения равномерного использования вычислительных ресурсов на всех этапах работы программы.

Ключевые слова: высокопроизводительные вычисления, балансировка загруженности, задача фильтрации, Message Passing Interface (MPI).

1. Введение. Для построения аппроксимации системы дифференциальных уравнений фильтрации [1] вводится сетка, учитывающая геологическое строение месторождения (разломы, выклинивания). Далее будем называть упомянутую сетку исходной. Для решения задачи используются только данные для ячеек сетки, имеющих объем, который может занимать жидкость (так называемый пóровый объем), больше определенного порогового значения. Такие ячейки сетки называются активными. Назовем расчетной сеткой совокупность активных ячеек исходной сетки.

Стандартный подход к разделению задачи между узлами кластера состоит в геометрическом разрезании области на части по числу вычислительных узлов. Этому разрезанию области соответствует разделение исходной и расчетной сеток. В каждом MPI-процессе загружаются данные и выполняется вычислительная работа для ячеек сетки, соответствующих его части области. Из-за особенностей геологического строения месторождения часто невозможно разделить область на части так, чтобы в каждой части количество ячеек исходной и расчетной сеток было одинаковым. Иногда при таком геометрическом разделении области в некоторых ее частях отсутствуют блоки расчетной сетки; в этом случае большинство программных пакетов решения задачи фильтрации останавливают свою работу.

Целью настоящей статьи является построение алгоритма решения задачи фильтрации, который обладает достоинствами стандартного алгоритма при загрузке данных и эффективно разделяет вычислительную работу между узлами кластера при расчете.

2. Загрузка данных и построение расчетной сетки. Предлагается разделить загрузку данных исходной сетки и построение распределенной по узлам расчетной сетки на три этапа.

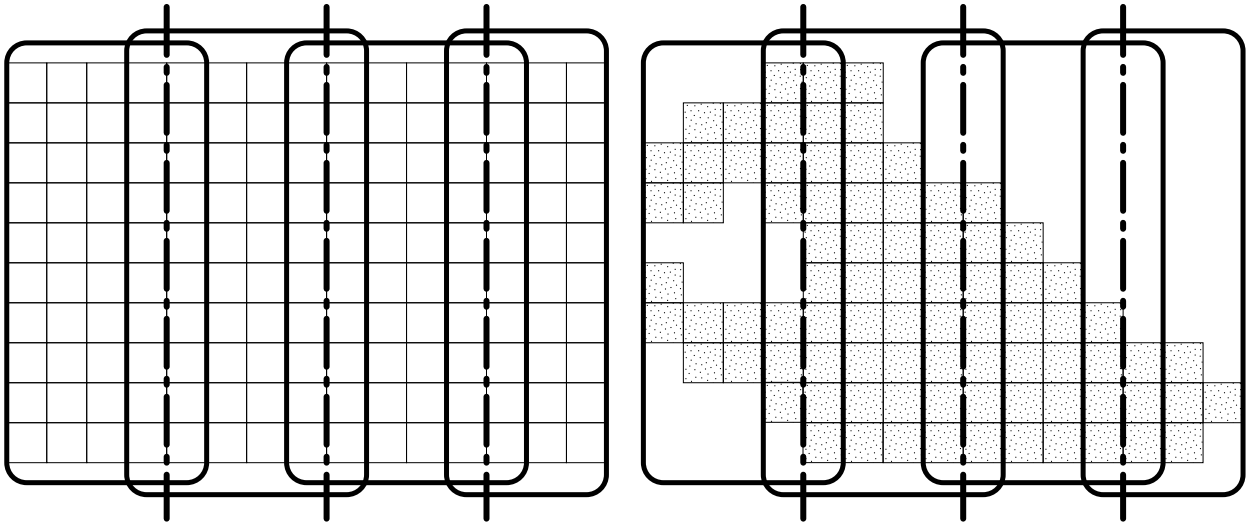
1) Исходная сетка делится равномерно между узлами с перекрытиями (как в традиционном подходе). На каждом узле осуществляется построение части расчетной сетки с вычислением графа связей с ячейками в других MPI-процессах. Схематически разделение показано на рис. 1а, где маленьким квадратам соответствуют ячейки исходной сетки, пунктирные линии показывают разделение между процессами, а жирные линии объединяют ячейки, распределенные на каждый узел.

2) Осуществляется сборка карты соответствия ячеек расчетной и исходной сеток на каждом узле, и на основании этой информации осуществляется вычисление номеров ячеек, которые попадут на каждый из узлов на следующем этапе.

3) Загрузка данных исходной сетки с сохранением на каждом узле кластера данных только для ячеек расчетной сетки, распределенных на данный узел.

Из-за слоистой структуры месторождения обычно имеется сильное различие свойств по вертикали и в горизонтальной плоскости. Для эффективного построения графа связей желательно распределить

[1] Московский государственный университет им. М. В. Ломоносова, механико-математический факультет, Ленинские горы, 119899, Москва; К. Ю. Богачев, доцент, e-mail: bogachev@mech.math.msu.su; А. А. Климовский, студент, e-mail: arseny@klimovsky.ru; А. Р. Миргасимов, аспирант, e-mail: mirgasimov.almaz@gmail.com; А. Е. Семенко, студент, e-mail: asemenko@gmail.com



а) б)

Рис. 1. Стандартное разделение исходной и расчетной сеток: а) разделение исходной сетки; б) разделение расчетной сетки

ячейки с одинаковыми номерами по горизонтальным координатам x и y ("столбики") на один вычислительный процесс, поэтому равномерно разделим область в ее проекции на плоскость xy. Для построения графа связей разделение осуществляется с перекрытием, чтобы ячейки, между которыми теоретически возможна связь, оказались на одном узле. Результат работы изображен на рис. 2, для сравнения на рис. 1б показано разделение, полученное при стандартном разрезании. На рисунках маленьким квадратам соответствуют ячейки расчетной сетки, пунктирные линии показывают разделение между процессами, а жирные линии объединяют ячейки, распределенные на каждый узел, на рис. 2 они не отображены, чтобы не ухудшить читаемость рисунка.

После выполнения первого этапа на основе загруженных данных можно определить, какие ячейки из распределенной на узел части исходной сетки попадают в расчетную сетку, и построить граф связей. Этой информацией обмениваются все процессы, и в каждом из них строится список ячеек расчетной сетки, который сохраняется в текстовом файле, чтобы исключить первый этап вообще, если рассчитывается несколько задач с одной и той же сеткой.

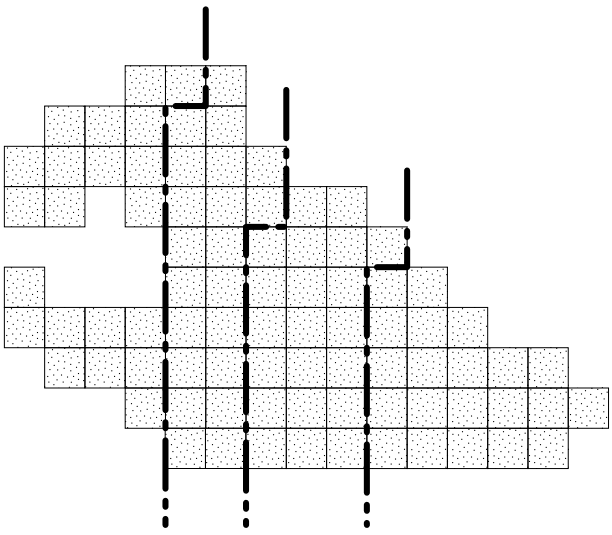


Рис. 2. Рассмотренное разделение расчетной сетки

Третий этап состоит в построении необходимых для параллельного расчета структур. Перед чтением данных исходной сетки на основе полученной информации строится разделение ячеек между узлами кластера с учетом перекрытия между расчетными областями, необходимого для вычисления свойств фильтрационных течений между ячейками в разных MPI-процессах, и необходимые отображения для индексации ячеек расчетной сетки. Нумерация ячеек производится так, чтобы соединенные ячейки имели близкие номера для уменьшения ширины ленты решаемых в последующем систем линейных уравнений [2]. После нумерации ячеек разделение ячеек производится равномерно по номерам: каждому узлу соответствует некоторый отрезок номеров ячеек.

Чтение данных происходит по общей схеме на всех этапах. На каждом узле читаются все входные данные, а сохраняется в памяти только та часть, которая соответствует ячейкам этого узла на данном этапе алгоритма. Это позволяет использовать один и тот же код на всех узлах кластера.

В предложенном алгоритме требуется производить чтение входных данных дважды, что позволяет иметь общий исходный код для первого и третьего этапов и избежать больших требований к оперативной

памяти на каждом узле.

Отметим, что на всех этапах на каждом узле используется объем памяти, достаточный для сохранения геометрических параметров сетки и свойств среды только в ячейках сетки, распределенных на данный узел с учетом перекрытия.

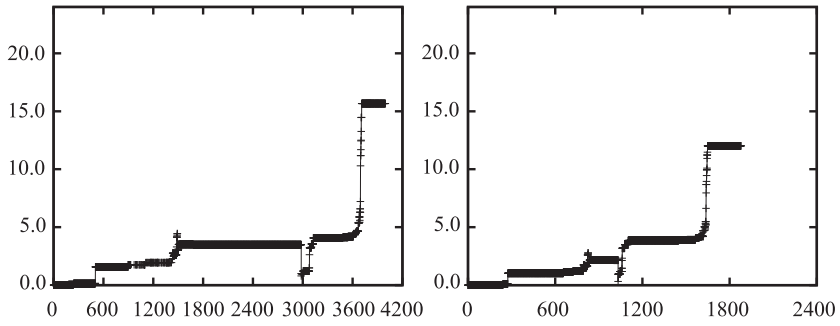


Рис. 3. Графики использования памяти на каждом узле; по оси абсцисс — время в секундах от начала расчета, по оси ординат — используемая оперативная память в гигабайтах: а) расчет на 10 узлах; б) расчет на 20 узлах

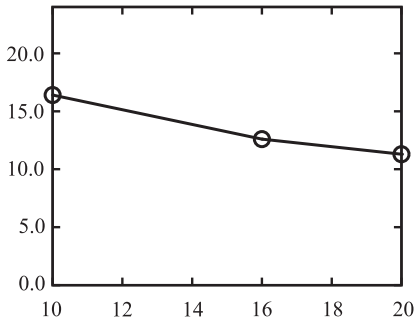


Рис. 4. График зависимости пикового потребления оперативной памяти (в гигабайтах) на узле от их числа

3. Результаты тестирования. Описанный метод был реализован в составе Российского промышленного гидродинамического симулятора tNavigator и протестирован на кластере из 20 параллельных ЭВМ с двумя процессорами Intel Xeon. Для тестирования была выбрана гидродинамическая модель Самотлорского месторождения на геологической сетке, содержащей 232 миллиона ячеек исходной сетки.

Тестовая вычислительная система представляет собой 20 узлов, на каждом из которых два шестиядерных процессора Intel Xeon X5650 с частотой 2.67 GHz и 24 Gb оперативной памяти DDR-III 1333 MHz; соединение между узлами осуществляется Infiniband QDR с пропускной способностью 40 Gb/s.

Приложение было запущено на исполнение в гибридном режиме, оптимизировано для многопроцессорных систем с неоднородной памятью [3]; на каждом узле работал один MPI-процесс, в котором было 12 потоков исполнения.

Чем меньшее количество узлов используется для расчета задачи, тем больший объем оперативной памяти требуется для этого на каждом из узлов (см. рис. 4, где показан график зависимости ее пикового потребления на узле от их числа). Например, при запуске 10 MPI-процессов максимальный объем выделенной оперативной памяти на узле составляет 16,4 ГБ, а при 20 — 11,3 ГБ. Это приводит к тому, что размер рассматриваемой задачи позволяет ее рассчитывать только на 10 и более узлах.

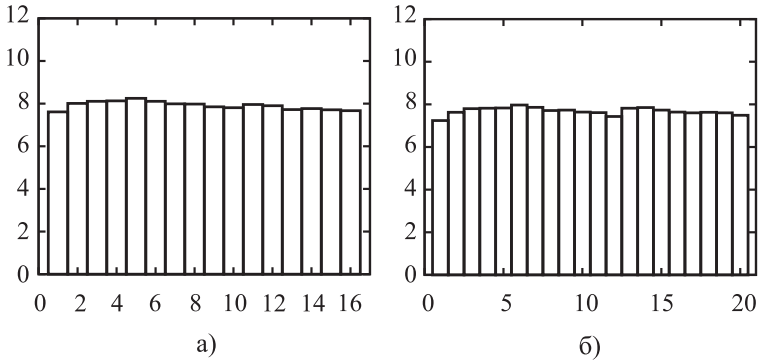


Рис. 5. Графики загруженности узлов; по оси абсцисс — номер узла, по оси ординат — средняя загруженность: а) расчет на 16 узлах, б) расчет на 20 узлах

Объем оперативной памяти, использованной программой на каждом из узлов, оказался практически одинаковым в каждый момент времени как на этапе загрузки данных, так и на этапе расчета. Диаграмма ее использования в зависимости от времени, прошедшего с начала расчета, показана на рис. 3а при расчете на 10 узлах и на рис. 3б при расчете на 20 узлах.

Загруженность процессоров, измеряемая стандартно для UNIX-систем, на этапе чтения на всех узлах — около единицы, потому что, как описывалось выше, в каждом MPI-процессе независимо происходит чтение входных данных. Диаграмма загруженности процессоров в начале расчета представлена на рис. 5а и 5б для запуска на 16 и 20 узлах соответственно. Максимально достижимое значение равно 12, поскольку всего имеется 12 логических процессоров. Из диаграммы видно, что в среднем 8 процессоров на каждом из узлов постоянно занято вычислительной работой.

4. Заключение. Предложенный алгоритм балансировки загруженности позволяет эффективно рассчитывать очень большие задачи фильтрации на кластерных системах, обеспечивая равномерность как выделенной оперативной памяти, так и вычислительной нагрузки даже при сложном взаимном расположении исходной и расчетной сеток. Алгоритм использует из входных данных только геометрическую информацию, что позволяет распределить данные на ранних этапах их загрузки.

СПИСОК ЛИТЕРАТУРЫ

1. *Богачев К.Ю., Мельниченко Н.С.* О пространственной аппроксимации методом подсеток для задачи фильтрации вязкой сжимаемой жидкости в пористой среде // Вычислительные методы и программирование. 2008. **9**, № 2. 42–50.
2. *Богачев К.Ю., Жабицкий Я.В.* Блочные предобусловливатели класса ILU для задач фильтрации многокомпонентной смеси в пористой среде // Вестн. Моск. ун-та. Матем. Механ. 2009. № 5. 19–25.
3. *Богачев К.Ю., Миргасимов А.Р.* Об оптимизации вычислительных приложений для многопроцессорных систем с общей неоднородной памятью // Вычислительные методы и программирование. 2010. **11**, № 2. 40–44.

Поступила в редакцию
22.12.2010