РАСЧЕТНЫЕ НЕСТРУКТУРИРОВАННЫЕ СЕТКИ ДЛЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ

С.П. Копысов, А.Б. Пономарев, В.Н. Рычков, С.Н. Зубцовский

Институт прикладной механики УрО РАН, г. Ижевск

Введение

Эффективность распараллеливания вычислительных алгоритмов определяется программной реализацией сетки в тех случаях, когда алгоритмы включают шаги, связанные с изменением топологии и расчетных данных сетки в процессе вычислений.

В работе [1] была предложена объектно-ориентированная модель расчетной неструктурированной сетки. Целью данной работы является создание на ее основе распределенной модели расчетной сетки, предназначенной для реализации параллельных алгоритмов. Одним из требований является внесение минимально возможного числа изменений в существующую модель сетки. Модель распределенной расчетной сетки является основой для решения большинства расчетных задач на многопроцессорных системах.

1. Использование расчетных сеток на многопроцессорных системах

Расчетная сетка – это данные, над которыми выполняются операции двух видов: методы модификации сетки и вычислительные алгоритмы, основанные на сеточной аппроксимации. Первые используют сетку как входные и выходные данные и модифицируют ее. Вторые же – абстрагируются от сетки к системе уравнений; их основные действия выполняются над матрицами и векторами, которые формируются в соответствии с сеточной аппроксимацией. В вычислительных алгоритмах с адаптивным перестроением сетки [2, 3] эти виды операций комбинируются.

Если рассматривать расчетные сетки с точки зрения их использования на многопроцессорных системах, то это распределенные данные. Над ними выполняются последовательные и параллельные операции. В общем степень распараллеливания вычислительных алгоритмов не зависит от программной реализации сетки. Но как только в алгоритмах появляются шаги, связанные с модификацией сетки (перестроение, повторное использование после изменения числа расчетных параметров в отдельных элементах, балансировка нагрузки и пр.), так сразу большое значение начинает играть эффективность параллельной реализации операций этого вида.

При расширении исходной (последовательной однопроцессорной) модели сетки необходимо:

– создать модель разделенной сетки и разработать на ее основе распределенную модель;

– провести декомпозицию программного кода операций модификации сетки и реализовать их распределенную версию;

– распараллелить и оптимизировать операции модификации сетки.

Пусть задана грубая сетка, на которой установлены расчетные данные. Рассмотрим следующую последовательность шагов типичного сеточного алгоритма над большими объемами данных, предназначенного для запуска на многопроцессорных системах.

Шаг 1. Разделение сетки – грубая расчетная сетка разделяется на заданное число подсеток, равное, например, числу процессоров. Для реализации этого шага будем использовать исходную объектно-ориентированную модель сетки [1], которую необходимо расширить путем введения нового класса, задающего подсетки. Затем расширенная модель будет наполнена функциональностью, связанной с различными алгоритмами разделения сеток [2].

Шаг 2. Распределение сетки – подсетки размещаются на процессорах. Поддержка распределенных между различными вычислительными узлами объектов невозможна без коммуникационного промежуточного программного обеспечения [4], использование которого в объектно-ориентированном моделировании неизбежно приводит к понятиям клиент–сервер. Серверами в данном случае удобно представить подсетки. Их взаимодействиями управляет объект, задающий разделенную сетку, – клиент. Распараллеливание достигается путем асинхронного вызова со стороны клиента методов удаленных объектов подсеток.

Шаг 3. Дробление сетки – в соответствии с тем или иным алгоритмом подсетки измельчаются для проведения расчетов на полученной в результате сетке. Этот этап может неоднократно выполняться в ходе вычислений с адаптивным перестроением сетки. Алгоритмы дробления уже реализованы в исходной модели расчетной сетки. Перед распределенной реализацией их необходимо записать в терминах разделенной сетки, другими словами, декомпозировать код в соответствии с декомпозицией сетки. После этого алгоритмы распараллеливаются.

Шаг 4. Перераспределение сетки – перемещение сеточных объектов с одной подсетки/процессора на другие с целью выравнивания связанной с ними нагрузки того или иного вычисленного алгоритма. Для эффективного выполнения данного шага применительно к самому широкому классу сеточных алгоритмов требуется наличие программных методов, обеспечивающих различные стратегии размещения сеточных объектов. Для одних численных методов требуется пересылать сеточные объекты и все, связанные с ними расчетные данные, для других целесообразнее пересчитывать те или иные параметры заново, а иногда лучше избегать пересылки объектов, а передавать только некоторые их параметры.

2. Декомпозиция кода в модели разделенной сетки

Характерной особенностью созданной модели является то, что и сетка, и подсетка представляются одним и тем же классом. Данный подход позволяет рассматривать подобласти как самостоятельные сетки, что, в свою очередь, позволяет применять к подобластям те же алгоритмы, использовать их в тех же расчетах, что и исходный класс сетки без необходимости внесения изменений в код.

С помощью данной структуры можно декомпозировать код алгоритмов, реализованных в исходной объектно-ориентированной модели. Это предварительная декомпозиция программы до перехода к параллельным вычислениям на многопроцессорных системах. Декомпозиции подвергаются методы перемещения (используются на 2-м и 4-м шагах) и перестроения (используются на 3-м шаге) сеточных объектов.

3. Размещение сеточных объектов в распределенной модели

Разработанная модель распределенной расчетной сетки является расширением модели разделенной расчетной сетки. Новым здесь является введение объектов клиент–сервер для подсеток и всех видов сеточных объектов. Классы клиент–сервер подсетки обеспечивают удаленный доступ сетки к подсеткам, размещенным на других узлах вычислительной системы. Клиенты-серверы сеточных объектов предназначены для представления границы – это позволяет сохранить целостность данных – клиентские объекты обеспечивают доступ к граничным объектам, которые как серверы существуют в единственном экземпляре.

Основная проблема, решаемая в распределенной модели, – это обработка объектов, принадлежащих двум или более подсеткам (граничными объектами). Если подсетки будут получать копии граничных объектов, возникнет проблема синхронизации этих копий. В больших сетках, количество узлов в которых достигает пяти-шести порядков, синхронизация граничных объектов будет занимать значительное дополнительное время и ресурсы. Кроме того, огромная дополнительная ответственность ложится на программиста, ведь существует большое множество случаев, когда может потребоваться синхронизация копий объектов, и предусмотреть их все крайне сложно. Если же синхронизация будет производиться всякий раз, когда с объектом произошли какие-либо незначительные изменения, это приведет к огромному количеству вызовов от клиента к удаленному серверу и обратно, а также значительным дополнительным расходам памяти.

В модели распределенной сетки были созданы классы клиентов и серверов для граничных объектов (узлов, ребер, граней). Таким образом, исчезает необходимость хранения копий объектов в подсетках и исчезает проблема синхронизации. Экземпляры граничных объектов существуют в

единственном числе и хранятся в остове сетки. Остов содержит все объекты границ разделенной сетки. Взаимодействие с ними организуется посредством CORBA [5] с помощью соответствующих клиентов. Помимо граничных узлов существуют также граничные ребра и грани, для них также созданы соответствующие классы, исполняющие роль клиента и сервера.

Введение цепочки «клиент–сервер» для граничных объектов позволяет избежать перерасхода ресурсов и уменьшить время, затрачиваемое на расчеты. Безусловно, передача данных по сети будет замедлять работу, но потери времени в этом случае несравнимы с потерями во времени при проведении синхронизации между копиями объектов, находящихся в разных сетках.

Важнейшим методом, реализованным на уровне распределенной модели, является размещение сеточных объектов и создание заместителей для них на сетке и подсетках. Этот метод используется при реализации различных стратегий перераспределения объектов. По сути стратегия перераспределения есть решение вопроса, где нужно разместить сам объект, а где следует обращаться к этому объекту удаленно, посредством заместителей. Когда в подсетках граница перестраивается адаптивно к геометрии, сеточные объекты, принадлежащие границе области, целесообразно размещать в подсетках, оставляя заместителей в остове. Если же граница области не изменяется или изменяется несущественно, то размещать серверы граничных объектов можно в остове.

4. Асинхронное выполнение методов на подсетках

Распределенная модель позволяет привлечь большее число ресурсов, например кластеров ЭВМ. Суммарное же время выполнения будет даже больше времени выполнения задачи на отдельной машине за счет сетевых коммуникаций, которые добавляются в последовательный код. Распределенная модель составляет основу для распараллеливания вызовов отдельных методов. При параллельной работе затраты времени на решение задачи будут существенно уменьшаться.

Распараллеливание вызовов организуется с помощью технологии CORBA AMI [5], которая позволяет вызывать методы удаленных объектов асинхронно, не дожидаясь возвращения ими результата. Для превращения распределенной модели в параллельную требуется расширение класса клиента и создание дополнительного класса, который будет обрабатывать ответ, приходящий от сервера после завершения выполнения метода. Изменения необходимо вносить лишь на стороне клиента, что удобно, так как избавляет от необходимости переписывать больше строк кода.

При асинхронных вызовах операция может быть поделена на две части: запрос серверу и ожидание ответа клиентом. При ожидании клиентом результатов происходит приостановка всех обращений к серверу, что дает

ему возможность закончить работу и вернуть результат выполнения метода. Как только сервер готов отдать результат, он связывается с обработчиком ответа, который, в свою очередь, сообщает связанному с ним асинхронному клиенту о том, что операция была завершена, после чего клиент может продолжать работу с другими методами.

На данный момент распараллеливание реализовано для методов загрузки разделенной сетки из набора файлов, переноса объектов на удаленные серверы при делении и сохранение сетки вместе с подсетками, находящимися на удаленных серверах.

5. Перераспределение сеточных объектов

Динамическое перераспределение сеточных объектов применяется при балансировке нагрузки между вычислительными узлами. При перераспределении сеточных объектов, помимо загруженности вычислительных узлов, немаловажную роль играют затраты на выполнение непосредственно пересылки данных сеточных объектов. Выделим три стратегии перераспределения объектов [6]:

1. Перемещение сеточного объекта со всеми свойствами. Эта стратегия является наиболее затратной из всех рассматриваемых, поскольку необходимо переместить все данные, и выбирается в качестве базовой в построенной модели.
2. Перемещение сеточного объекта с созданием некоторых (или всех) из его свойств заново. Применяется в том случае, если восстановить данные свойств проще и быстрее, чем переслать их по коммуникационной сети. Для реализации этой стратегии у класса, реализующего конкретное свойство, в функции перемещения на подобласть необходимо, вместо реального перемещения, указать уничтожение свойства с созданием его на другой стороне и запустить его инициализацию.
3. Удаленный доступ через заместителя без реального перемещения объекта. Данная стратегия является самой экономной по времени перераспределения, но самой затратной в дальнейших вычислениях. Она может использоваться в том случае, когда пересылка данных всего объекта очень длительна и прогнозируется повторное перемещение объекта. В этом случае используется заместитель объекта, который создаётся автоматически при попытке доступа к объекту из другой подсетки.

Одним из методов решения задачи динамического перераспределения является последовательная обработка сеток, т.е. в определенный момент времени расчет и разделение происходят только на одной подсетке. В этом случае худшее, что можно ожидать, – некоторые объекты переместятся дважды, например, будучи граничными между подсетками, они на операции перераспределения одной подсетки станут внутренними, а на операции следующей подсетки – снова граничными, или наоборот. Но число та-

ких объектов будет достаточно небольшое, и по сравнению с общим объемом пересылок практически ничтожно.

Другой метод предназначен для параллельного перераспределения сеток. После того как становится известным расположение ячеек, рассылается информация в минимальном формате по всем подсеткам, тогда при определении расположения граней на каждом вычислительном узле будет находиться вся необходимая информация. После каждого следующего цикла определений расположения (для граней и для ребер) снова происходят рассылки. Когда определятся места расположения узлов, начнется фактическое перемещение объектов.

Заключение

Основная идея данной работы – построение и использование программной модели расчетной сетки в параллельных вычислениях. Для этого применялись следующие методы: декомпозиция методов перемещения и перестроения сеточных объектов исходной модели в соответствии с новой структурой данных, представляющей разделенную сетку в виде остова и набора подсеток; реализация подхода клиент–сервер с помощью промежуточного программного обеспечения CORBA при создании распределенной модели; распараллеливание на основе асинхронного выполнения методов в подсетках; определение методов размещения клиентов и серверов сеточных объектов для реализации различных стратегий перераспределения объектов.

Литература

1. Копысов С.П., Пономарев А.Б., Рычков В.Н. Открытое визуальное окружение для взаимодействия с геометрическими ядрами, генерации/перестроения/разделения сеток и построения расчетных моделей // Прикладная геометрия, построение расчетных сеток и высокопроизводительные вычисления / Ред. Ю.Г. Евтушенко, М.К. Керимов, В.А. Гаранжа. – М.: ВЦ РАН, 2004. – Т. 2. – С. 154–164.
2. Копысов С.П., Новиков А.К. Параллельные алгоритмы адаптивного перестроения и разделения неструктурированных сеток // Математическое моделирование. − 2002. Т. 14, № 9. − С. 91–96.
3. Копысов С.П., Красноперов И.В. Адаптивное решение задач в p-версии метода конечных элементов // Прикладная геометрия, построение расчетных сеток и высокопроизводительные вычисления / Ред. Ю.Г. Евтушенко, М.К. Керимов, В.А. Гаранжа. − М.: ВЦ РАН, 2004. − Т. 2. − С. 85−95.
4. Копысов С.П., Красноперов И.В., Рычков В.Н. Совместное использование систем промежуточного программного обеспечения CORBA и MPI // Программирование. − 2006. − № 4 (принята в печать).

5. Цимбал А. Технология CORBA для профессионалов. − СПб.: Питер, 2001. − 624 с.
6. Schmidt D.C. and Vinoski S. Programming Asynchronous Method Invocations with CORBA Messaging // C++ Report, SIGS. – 1999. − Vol. 11. № 2.
7. Копысов С.П., Красноперов И.В., Пономарев А.Б., Рычков В.Н. Модель расчетных неструктурированных сеток для параллельных распределенных вычислений // Математическое моделирование. − 2006 (принята в печать).