

СОВМЕЩЁННАЯ ПРОЦЕДУРА СБОРКИ МАТРИЦЫ ЖЁСТКОСТИ ПРИ РЕШЕНИИ УПРУГОПЛАСТИЧЕСКОЙ ЗАДАЧИ НА КЛАСТЕРНОЙ СИСТЕМЕ

INTERLEAVED STIFFNESS MATRIX ASSEMBLY WITHIN A NUMERICAL SOLUTION OF ELASTIC-PLASTIC PROBLEM ON COMPUTER CLUSTER

А.В. Толмачев, Ю.В. Халевицкий, А.В. Коновалов

Федеральное государственное бюджетное учреждение науки
«Институт машиноведения Уральского отделения
Российской академии наук», г.Екатеринбург, ул. Комсомольская, 34
me@dijkstra.ru

Abstract

Finite element method is widely used in metal forming process simulation. Parallel implementation of this method on computer cluster raises an issue of parallel stiffness matrix assembly. This paper describes a novel approach to data transmission handling during assembly steps. Advanced techniques such as computation and communication interleaving and implicit synchronization are used. Authors introduce additional bufferization to hide MPI latency and reduce memory usage.

Разработка современных технологий ОМД невозможна без использования сложных программно-аппаратных комплексов, позволяющих прогнозировать изменение формы и напряжённого состояния в металлах [1]. Большие пластические деформации, возникающие в ходе обработки металла, приводят к высокой вычислительной сложности моделирования процесса вследствие физической и геометрической нелинейности задачи [2].

Для обеспечения комфортной работы инженера-технолога целесообразно использовать кластерные вычислительные системы с параллельной архитектурой, которые позволяют значительно снизить время вычислений, а следовательно и время отклика программно-аппаратного комплекса.

Кластерные системы

Подробное описание кластерных систем можно найти в книге [3].

Кластерная система состоит из большого числа узлов, соединённых высокопроизводительной вычислительной сетью. Каждый узел является самостоятельной вычислительной системой и обладает собственными центральными процессорами, оперативной памятью и, возможно, дополнительными вычислительными ускорителями. Логическое разделение вычислителей при решении задач может повторять, а может и не повторять топологию высокопроизводительной сети кластерной системы. Для устранения неоднозначности в дальнейшем будет использоваться термин «вычислительный элемент», который соответствует единичному вычислителю. В случае использования терминологии стандарта Message Passing Interface (MPI) вычислительному элементу может соответствовать процесс MPI. Один узел кластерной системы может одновременно поддерживать несколько вычислительных элементов.

Взаимодействие между вычислительными элементами осуществляется по протоколу обмена сообщениями. В обмене сообщениями в общем случае участвуют два вычислительных элемента. Вычислительный элемент-отправитель сообщает программному обеспечению кластерной системы о том, что он готов отправить сообщение, передавая номер вычислительного процесса-адресата. Последний сообщает программному обеспечению о готовности принять сообщение.

Программное обеспечение скрывает от программиста конкретный способ обмена данными между вычислительными элементами: обмен данными внутри одного узла кластерной системы происходит с помощью копирования областей разделяемой памяти, тогда как для связи с соседними узлами используется высокопроизводительная локальная вычислительная сеть. Время операции пересылки сообщения зависит от размера сообщения и конкретного способа обмена данными, однако каждое сообщение требует некоторого дополнительного времени на пересылку независимо от его объёма.

Обмен данными является важнейшей операцией, так как обеспечивает взаимодействие вычислительных элементов. Современное аппаратно-программное обеспечение позволяет гибко управлять обменом данными, в том числе совмещать вычисления с передачей информации, что значительно снижает время решения задачи.

Структура упругопластической задачи

Как правило, для моделирования процесса деформации металла используется метод конечных элементов. Вычисление состоит из большого количества (порядка тысяч) шагов по времени, каждый из которых осуществляется в три этапа [4]:

1. Сборка матрицы жёсткости.
2. Решение системы линейных алгебраических уравнений (СЛАУ).
3. Пересчёт напряжённо-деформированного состояния.

Как правило, этапы 2 и 3 повторяются от 10 до 15 раз для удовлетворения условия текучести Мизеса и должны выполняться строго последовательно. Программные компоненты для решения СЛАУ широко распространены и, как правило, поставляются в виде библиотек, таких, как ScaLAPACK[5]. Опыт применения этих программных компонентов и используемых в них алгоритмов применительно к упругопластической задаче можно найти в других работах, как [6]. Данная работа рассматривает этапы 1 и 3.

Этапы 1 и 3 имеют сходную структуру. Для каждого из конечных элементов строится локальная матрица жёсткости и локальная порция вектора правой части. Затем из полученных данных формируется порция глобальной матрицы жёсткости и порция глобального вектора правой части, которые используются в решении СЛАУ. Глобальная матрица жёсткости строится, как сумма локальных матриц жёсткости конечных элементов, отображённых на глобальную матрицу.

Для выполнения вычислений на всех этапах задачи используется один и тот же набор вычислительных элементов. Если для проведения следующего этапа вычислений требуются данные, полученные на другом вычислительном элементе, то происходит передача информации. Сама передача информации требует значительных затрат времени, кроме того, следующий этап не может быть начат до тех пор, пока не завершена передача данных.

Исходя из структуры задачи, возникает два логических распределения данных по вычислительным элементам: каждому вычислительному элементу соответствует некоторое количество конечных элементов, а также некоторое количество элементов глобальной матрицы жёсткости. В дальнейшем под выражением «Конечный элемент А распределён на вычислительный элемент В» имеется в виду, что информация о конечном элементе А хранится в памяти вычислительного элемента В, и вычислительный элемент А проводит вычисление соответствующей локальной матрицы жёсткости.

Для того чтобы уменьшить объём передаваемой информации между вычислительными элементами необходимо, чтобы результирующая порция глобальной матрицы жёсткости соответствовала тому же вычислительному элементу, что и конечные элементы, вычисления над которыми её порождают. Такого распределения несложно достичь для регулярных конечно-элементных сеток: каждому вычислительному элементу ставится в соответствие слой конечных элементов сетки. Помимо невозможности использования с нерегулярными сетками, такой способ распределения имеет ряд существенных недостатков. Во-первых, для эффективной работы некоторых прямых методов решения СЛАУ необходимо, чтобы на каждый вычислительный элемент было распределено два слоя конечных элементов. При решении

двухмерных задач это ограничение не является значительным, так как количество элементов в слое относительно мало. Однако при решении трёхмерных задач слой может содержать значительное количество элементов. Так, при решении трёхмерной упругопластической задачи на регулярной сетке размера $50 \times 50 \times 50$ каждый слой содержит 2500 конечных элементов и невозможно решение задачи с использованием более 25 вычислительных элементов. Во-вторых, подобное распределение не позволяет гибко производить балансировку нагрузки на вычислительные элементы, время вычисления на которых может существенно отличаться в силу различий в характеристиках узла кластерной системы или специфики задачи, поскольку количество распределённых на вычислительный элемент конечных элементов всегда должно быть кратно количеству конечных элементов в слое.

В данной работе предлагается новый способ построения матрицы жёсткости, который не накладывает никаких принципиальных ограничений на распределение конечных элементов по вычислительным элементам и использующий особенности кластерных систем.

Прямая и обратная процедура обмена данными

Непосредственное вычисление локальных матриц жёсткости происходит на вычислительных элементах, на которые распределены соответствующие конечные элементы. Перед началом решения СЛАУ локальные матрицы жёсткости передаются на вычислительные элементы, которые будут обрабатывать соответствующую часть глобальной матрицы жёсткости. Передача данных возникает в двух случаях: непосредственно после вычисления локальных матриц жёсткости и перед ним, в случае, если до этого уже происходило решение СЛАУ. Во втором случае необходимо передать на узлы с конечными элементами фрагменты вектора решений.

Передачу локальных матриц жёсткости перед началом решения СЛАУ в дальнейшем будем называть прямой процедурой. Передачу фрагментов вектора решения в дальнейшем будем называть обратной процедурой.

В ходе прямой процедуры каждый вычислительный элемент строит для каждого распределённого на него конечного элемента локальную матрицу жёсткости. Получившиеся элементы отображаются на глобальную матрицу жёсткости и накапливаются в специальных буферах. Заполненный буфер передаётся программному обеспечению кластерной системы для последующей отправки. При этом построение локальной матрицы жёсткости и передача информации может происходить одновременно. Использование буферов позволяет минимизировать накладные расходы, вызванные передачей сообщений. Фиксированный размер буфера также позволяет упростить работу вычислительного

элемента-адресата, который заранее выделяет участки памяти соответствующего размера для приёма сообщений.

Такой подход позволяет значительно упростить обмен данными и сократить расходы памяти, однако ставит перед разработчиками вычислительной процедуры новую задачу: требуется вовремя прекратить приём сообщений и начать вычисления. Очевидным решением было бы введение дополнительного сообщения, которое отмечало бы завершение построения локальных матриц жёсткости. Однако одновременная отправка этого сообщения, небольшого по размеру, всем вычислительным элементам нерациональна по ряду причин. Во-первых, при значительном количестве узлов придётся отправить значительное количество сообщений — квадрат количества вычислительных элементов, что создаст значительную нагрузку на коммутаторы высокопроизводительной сети кластерной системы; во-вторых, это откладывает начало вычислений до момента получения всех сообщений об окончании расчётов на предыдущем этапе.

К рациональному решению возникшей задачи приводит возможность заранее вычислить передаваемый объём информации. Используя эти сведения, вычислительный элемент может принять решение о том, какое количество информации необходимо получить и вовремя прекратить приём информации без использования каких-либо дополнительных сообщений. При большом числе вычислительных элементов каждый из них отправляет данные лишь небольшому количеству адресатов.

Для организации процесса используется квадратная матрица передачи данных $T = \{t_{ij}\}$ размерности p , где p — количество вычислительных элементов. Число k на пересечении i -го столбца и j -й строки матрицы означает, что i -му вычислительному элементу необходимо получить от j -го элемента количество информации. Числа на главной диагонали обозначают количество элементов, которые будут использованы при решении СЛАУ на том же узле, где они были получены.

При большом количестве вычислительных элементов матрица передачи данных получается разреженной, так как количество элементов в каждой её строке невелико, поэтому целесообразно использовать формат хранения разреженных матриц, который позволяет экономно использовать память при сохранении высокой скорости операции вставки нового элемента. Такой формат, предложенный в работе [7], может представлять собой матрицу в координатном формате, упакованную в массив стеков согласно соответствующей элементу строке матрицы, однако хороших результатов позволяет добиться и сбалансированное двоичное дерево поиска, где для сравнения элементов используется составной предикат.

В ходе обратной процедуры происходит передача порций вектора решения с узлов, на

которых происходило решение СЛАУ на узлы, которые будут выполнять пересчёт напряжённо-деформированного состояния. В отличие от прямой процедуры, в обратной процедуре невозможно совмещение передачи данных с вычислениями, так как вычислительные элементы, занятые решением СЛАУ заканчивают работу практически одновременно. Кроме того, при обратной процедуре возможна ситуация, когда необходимо отправить одну и ту же информацию сразу на несколько узлов. Для выполнения обратной процедуры каждый из вычислительных элементов строит *отображение обратной передачи данных* — функцию, которая ставит в соответствие номеру вычислительного элемента список элементов вектора решения. После окончания работы программы решения СЛАУ соответствующие элементы упаковываются в буферы и отсылаются вычислительным элементам-адресатам.

Динамическая балансировка нагрузки

По причине высокой гетерогенности современных кластерных систем производительность вычислительных элементов может оказаться не одинаковой: разные узлы могут быть оборудованы различными центральными процессорами и ускорителями, а также выполнять задачи, запущенные другими пользователями. Это влияет на время вычисления локальной матрицы жёсткости. Также на него влияет текущее состояние конечного элемента.

Различное время работы вычислительных элементов может приводить к общему увеличению времени вычисления за счёт возникающих синхронизаций. Для компенсации разницы между временем расчёта вычислительных элементов может осуществляться динамическая балансировка нагрузки. Это означает, что конечные элементы должны быть перераспределены с наиболее загруженных узлов на простаивающие. В процессе динамической балансировки каждый вычислительный элемент выбирает конечные элементы для передачи и строит *обновление матрицы передачи данных*.

Заключение

При решении упругопластической задачи на кластерной системе для достижения наибольшей производительности следует учитывать как специфику задачи, так и особенности компонентов самой системы.

На этапе построения матрицы жёсткости наиболее пристальное внимание следует уделить межузловому обмену и оптимизации синхронизации между вычислительными элементами. В частности, следует совмещать вычисление локальных матриц жёсткости с обменом данными, обеспечить дополнительную буферизацию данных и избегать явной синхронизации работы вычислительных элементов при помощи сообщений.

Предложенные улучшения позволяют повысить эффективность работы высокопроизводительной сети кластерной системы,

и тем самым уменьшить время решения упругопластической задачи.

Список литературы

1. Сметанин С.В., Большаков Н.С. Возможности современных расчётных модулей по трёхмерному моделированию процессов обработки металлов давлением //Машиностроение. 2007. –№ 17. –С. 97-102.
2. Толмачев А.В. Коновалов А.В. Партин А.С. Использование усечённого варианта алгоритма SPIKE из библиотеки IntelAdaptive SPIKE-BasedSolver для решения упругопластических задач // Вычислительные методы и программирование: новые вычислительные технологии. 2011. –Т. 12. –№ 1. –С. 154-159.
3. Воеводин В. В., Воеводин В. В. Параллельные вычисления. – СПб : БХВ-Петербург, 2004. – ISBN 5-94157-160-7.
4. Поздеев А.А., Трусов П.В., Няшин Ю.И. Большие упруго-пластические деформации. М.: Наука, 1986.
5. Blackford L.S., Choi J., Cleary A., D'Azevedo E. et al. ScaLAPACK User's Guide. Philadelphia, USA, Society for Industrial and Applied Mathematics, 1997. – URL: <http://www.netlib.org/scalapack/slug> (датаобращения: 26.09.2013).
6. Толмачев А.В., Коновалов А.В., Партин А.С. Анализ эффективности ряда параллельных итерационных методов решения СЛАУ в упругопластической задаче на кластерной системе // Параллельные вычислительные технологии (ПаВТ'2013): труды международной научной конференции (1–5 апреля 2013 г., г. Челябинск). Челябинск: Издательский центр ЮУрГУ, 2013. 637 с. С. 545-550.
7. N. Jansson, Optimizing sparse matrix assembly in finite element solverswith one-sided communication \ High Performance Computing for ComputationalScience. VECPAR 2012SpringerBerlinHeidelberg, 2013.C. 128-139.

Работа выполнена при финансовой поддержке для молодёжных исследовательских проектов УрО РАН на 2013 год, проект 13-1-НП-657 и программы фундаментальных исследований Президиума РАН №15, проект 12-П-1-1025