Исследование математических моделей физических систем в Грид средах на базе технологий MPI и CUDA

Курочка К.С.; Стефановский И.Л.
Кафедра «Информационные технологии» факультета АИС
Гомельский государственный технический университет им. П.О. Сухого
г.Гомель, Республика Беларусь
e-mail: kurochka@gstu.by, igorst@pisem.net

***Аннотация* – исследуется технология моделирования методом конечных элементов физических систем в Грид средах на базе технологии MPI и на графических ускорителях Nvidia с использованием технологии Compute Unified Device Architecture (CUDA) [1].**

Ключевые слова: метод конечных элементов; упругопластичность; вязкоупругость; параллельные вычисления; численное моделирование; графические процессоры; кластер; метод сопряженных градиентов; CUDA; параллельный алгоритм; многопоточный; многопроцессорный; Graphics Processing Unit (GPU) [1] общего назначения

I. Введение

Метод конечных элементов один из наиболее эффективных численных методов решения математических задач, описывающих состояние физических систем, имеющих сложную геометрическую конфигурацию и нерегулярную физическую структуру [2]. Однако в результате его использования, при исследовании математической модели приходиться многократно решать системы линейных алгебраических уравнений (СЛАУ) [3], что является весьма ресурсоемкой процедурой. Поэтому для эффективного решения такого класса задач целесообразно использовать распределенные вычислительные системы.

Для решения основного уравнения МКЭ [2] применялся метод сопряженных градиентов (МСГ) с диагональным предобуславливанием [4].

II. Архитектуры параллельных вычислительных систем

Существуют следующие архитектуры параллельных вычислительных систем [5]:

1. Симметричные мультипроцессорные системы (SMP). Характеризуются наличием общей памяти, к которой имеют доступ все процессоры, что упрощает программирование, но уменьшает масштабируемость системы.

2. Массово-параллельные системы (MPP). Состоят из множества узлов, содержащих процессоры и локальную память, которые соединены коммуникационной средой. Обладают хорошей масштабируемостью, но отсутствие общей памяти снижает скорость межпроцессорного взаимодействия и приводит к необходимости использовать механизм передачи сообщений для взаимодействия между узлами.

3. Системы с неоднородным доступом к памяти (NUMA), в которых локальная память узлов представлена логически как единое адресное пространство, что позволяет упростить разработку программ.

Кластерные системы представляют собой разновидность MPP-систем, построенных на базе широкодоступного оборудования.

III. Инструментальные средства

Наиболее распространенным средством организации обмена сообщениями в кластерной среде является технология MPI (Message Passing Interface – интерфейс передачи сообщений) [6]. Она представляет собой библиотеку, реализующую высокоуровневый обмен сообщениями между процессами.

Графические процессоры (GPU) представляют собой массивно-параллельные процессоры с общей памятью. В отличие от центрального процессора (CPU) с несколькими ядрами, один графический процессор может состоять из нескольких десятков мультипроцессоров, каждый из которых содержит до нескольких десятков вычислительных ядер, которые проводят вычисления параллельно [1]. Таким образом, на одном процессоре может быть запущено несколько десятков тысяч потоков одновременно.

Для использования этих возможностей необходимо применить одну из технологий, предназначенных для организации вычислений на GPU. Для процессоров компании Nvidia наиболее развитой и удобной является технология CUDA [1], основанная на расширении языка Си, которая позволяет получить доступ к набору инструкций GPU и управления его памятью. Это позволяет рассматривать GPU как обычное вычислительное устройство, при реализации параллельных алгоритмов, и избавляет от необходимости использовать более трудоемкие графические API.

IV. Параллельный алгоритм МСГ для MPI

Основными операциями, производимыми при расчете, являются операции умножения матрицы на вектор, скалярного произведения векторов и операция

$$x = x + a * y . \qquad (1)$$

где x, y – векторы; a – коэффициент.

Параллельная реализация этих операций зависит от распределения данных между компьютерами. Наиболее простым, но в то же время, эффективным способом распределения данных для данного алгоритма является ленточная схема [5], при которой к-ый компьютер получает Nk последовательных строк матрицы. Операции выполняются, так же как и на одном компьютере, за исключением того, что до выполнения операций необходимо распределить данные по узлам кластера, а после – собрать результат. Для сбора скалярных данных использовалась операция редукции MPI_AllReduce.

Для совмещения вычислений с обменом данными использовались неблокирующие операции MPI_ISend и MPI_IRecv, которые позволяют производить обмен в фоновом режиме.

Для хранения разреженных матриц существует множество форматов, которые отличаются различной трудоемкостью реализации операций их обработки, и различными накладными расходами при организации хранения. Т.к. при решении указанных задач возникают

диагональные матрицы, для их хранения наиболее подходит формат CDS (Compressed Diagonal Storage) [7], который обеспечивает наибольшую компактность хранения данных, а, следовательно, и наименьшие накладные расходы при распределении матрицы по узлам кластера.

Самой ресурсоемкой операцией является операция умножения матрицы на вектор, что приводит к необходимости тщательного продумывания алгоритма ее реализации

V. Параллельный алгоритм МСГ для CUDA

При умножении вектора на матрицу вектор используется много раз, и для оптимизации времени доступа для хранения вектора использовалась память текстур, которая хотя и находиться в глобальной памяти на видеокарте, но позволяет кэшировать данные.

CUDA позволяет объединять запросы к памяти в один, при запросе к ячейкам памяти расположенным линейно и кратным 32, 64, 128, 256 битам. Для использования этой возможности каждый поток вычисляет скалярное произведение i-ой строки матрицы на вектор, при этом, т.к. потоки выполняются параллельно группами по 32, происходит объединение запросов доступа к глобальной памяти в блоки по 128 бит, что приводит к увеличению пропускной способности. В силу того, что при матрично-векторном умножении меняется только вектор, то матрица передается по сети и загружается в глобальную память видеокарты один раз в начале вычислений. При этом вектор смещений для формата CDS используется много раз и храниться в разделяемой памяти, которая обладает задержкой доступа 1-2 такта.

VI. Результаты расчетов

Рассматривалась задача численного моделирования прогибов вязкоупругопластической тонкой пластинки с отверстиями [8]. Размерность СЛАУ метода конечных элементов для данной задачи составила 15123 элемента. Количество диагоналей равно 29. Эксперименты производились в кластере, построенном на базе процессоров Intel Dual Core E2140 (2 ядра, 1,6 ГГц, 2Гб ОЗУ), видеокарт GeForce 8500 GT (2 потоковых мультипроцессора, 16 ядер с частотой 450 МГц, 512 Мб видеопамяти с пропускной способностью 12,8 Гб/с) и сети Fast Ethernet. Сходимость метода была достигнута за 3201 итерацию. Величина ошибки составила 10^{-6}. Результаты эксперимента приведены в таблице.

Табл. Зависимость времени решения СЛАУ от используемого алгоритма и количества узлов кластера, с

Алгоритм	***Количество узлов кластера***				
	1	2	3	4	5
Последовательный	81,3				
MPI	41,54	21,53	12,74	9,65	16,98
MPI+CUDA	9,79	8,71	9,73	10,87	11,78

VII. Выводы

1. Реализация с использованием MPI обеспечивает ускорение более чем в 8 раз по сравнению с последовательной реализацией.

2. Реализация с использованием технологии CUDA, обеспечивает ускорение примерно в 4 раза по сравнению с технологией MPI на одном узле кластера и более чем в 8 раз по сравнению с последовательной реализацией.

3. Процесс вычислений является итерационным и предполагает существенный обмен данными между итерациями, т.о. производительность метода сопряженных градиентов существенно зависит от возможности обеспечения высокой скорости обмена между узлами. Т.к. при проведении эксперимента использовалось стандартное сетевое оборудование стандарта Fast Ethernet, не способное обеспечить приемлемой скорости обмена данными, а при увеличении количества узлов кластера увеличивается объем передаваемых по сети данных, использование дополнительных узлов кластера увеличивает время обработки. Причем в случае с CUDA максимально возможное ускорение вычислений достигается при меньшем числе узлов кластера, что объясняется более высокой скоростью GPU в сравнении с CPU.

[1] Боресков А.В., А.А. Харламов. Основы работы с технологией CUDA. М.: ДМК Пресс, 2010, с. 232.

[2] О. Зенкевич. Метод конечных элементов в технике. М.: Мир, 1975, с. 541.

[3] К. С. Курочка. Построение математической модели сложной системы неоднородных вязкоупругих дисперсных и сплошных твёрдых тел. Строительная механика инженерных конструкций и сооружений. – 2007. – №4. – с. 18-31.

[4] Д. Ортега. Введение в параллельные и векторные методы решения линейных систем. М.: Мир, 1991, с. 367.

[5] В.П. Гергель, Р.Г. Стронгин. Основы параллельных вычислений для многопроцессорных вычислительных систем. Учеб. пособие. Нижний Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2003, с. 184.

[6] Воеводин В. В., Вл. В. Воеводин. Параллельные вычисления. СПб.: БХВ-Петербург, 2002, с. 608.

[7] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst. Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. Philadelphia: SIAM, 2000.

[8] К. С. Курочка. Численное моделирование прогибов вязкоупругопластической тонкой пластинки с отверстиями. Механика машин, механизмов и материалов, 2007г., № 1(1), с. 60-64.