ОРГАНИЗАЦИЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ ДЛЯ ЗАДАЧИ МОДЕЛИРОВАНИЯ МЕТОДОМ КОНЕЧНЫХ ЭЛЕМЕНТОВ

И. Г. Нестереня
Кафедра информационных технологий, Гомельский государственный технический университет им. П. О. Сухого
Гомель, Республика Беларусь
E-mail: igor.nesterenya@gmail.com

Рассмотрен подход организации распределенных вычислений на примере задачи моделирования методом конечных элементов. Проведен анализ эффективности различных способов распределенных вычислений.

Введение

В настоящее время метод конечных элементов все больше используют для решения физических задач. Метод доказал свою эффективность в ряде научных исследований [1,2,3]. Но, как правило, при решении задач с высокой точностью мы получаем очень большие системы уравнений, в результате чего их решение становится ресурсоемкой задачей. Несмотря на то, что есть много способов ускорить такие операции, как умножение матриц (например, рекурсивные алгоритмы [4]), мы не всегда можем получить нужное нам ускорение, используя даже самые быстрые алгоритмы. В результате чего популярность получил другой подход: параллельные и распределенные вычисления. Многие из самых частых операций могут быть вычислены параллельно, и таким образом можно масштабировать скорость вычисления, добавляя больше вычислительных узлов.

I. Постановка задачи

Для ускорения вычислений будет рассмотрена задача нахождения изменения температуры, вызванной деформацией пластины под действием поперечной нагрузки [2].

При решении задачи методом конечных элементов можно выделить несколько стадий:

- построение модели;
- формирование сетки;
- указание материалов;
- формирование глобальной матрицы;
- указание граничных условий (нагрузки и закрепления);
- решение системы уравнений.

Необходимо провести анализ и выявить ресурсоемкие места. Для анализа времени выполнения было проведено профилирование с применением библиотеки Simon [5]. Анализировалось только время выполнения, так как оперативная память более доступный ресурс. Выявлены 3 самых медленых участка: формирование сетки, формирование глобальной матрицы, и решение системы уравнений. При формировании матрицы происходит множество операций умножения матриц и векторов. В настоящее время есть несколько популярных технологий кластерных вычислений: MPICH, OpenMP, Hadoop, Spark. Последние позволяют строить распределенные вычисления, используя концепцию map-reduce, которая отлично подходит для распределенного решения поставленной задачи.

II. Распределенное умножение матриц

Умножение матриц частая операция при решении методом конечных элементов. При использовании 12000 конечных элементов, для формирования глобальной матрицы происходит свыше 50000 умножений матриц. Умножение – классическая задача и есть множество известных подходов распараллеливания. Для распределенного умножения использован алгорим внешнего произведения матриц [6]. Задачу можно разделить на 2 стадии: map и reduce. На первой стадии на каждый узел кластера рассылаются по порядку столбец первой матрицы и соответствующая ему строка второй. Дальше, независимо друг от друга, они перемножаются. Потом следует стадия reduce, на которой нужно объеденить результат со всех узлов. Поскольку у каждого узла в результате предыдущей стадии получилась матрица, нужно сложить их, полученная матрица будет результатом перемножения первой и второй.

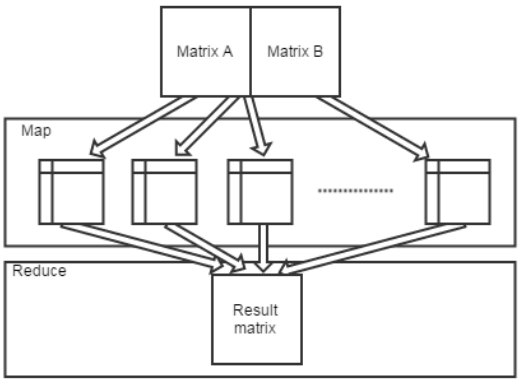

Рис. 1 – Распределенное умножение матриц с применением map-reduce

Для замера скорости умножения матриц, использованы заполненные ненулевыми элементами квадратные матрицы различных размерностей. Сравнивается обычный алгорим умножения матриц и его распределенная версия. Для распределенного умножения было использовано 4 узла.

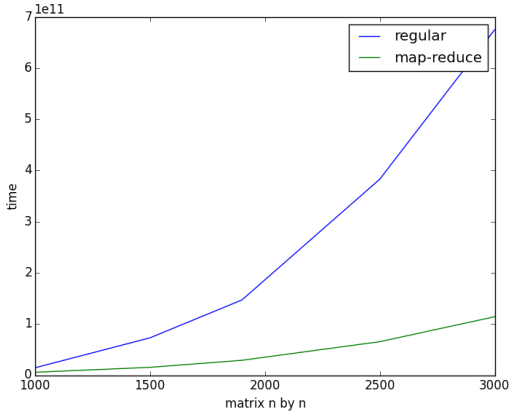

Рис. 2 – Сравнение скорости умножения матриц

По графику видно очевидное преимущество распределенного умножения. Эффективность распределенного варианта наиболее заметна при увеличении размерности умножаемых матриц.

III. Распределенное решение СЛАУ

Несмотря на то, что было полученно значительное ускорение умножения матриц, самой медленной частью решения задачи методом конечных элементов остается решение системы уравнений. Поскольку алгоримтм требует формирования глобальной матрицы [1,3], приходится работать с матрицей очень больших размерностей. При использовании 12000 конечных элементов, глобальная матрица имеет размерность 36000 для рассматриваемой задачи [2].

В данном случае используется решение методом Гауса, с распределением по строкам.

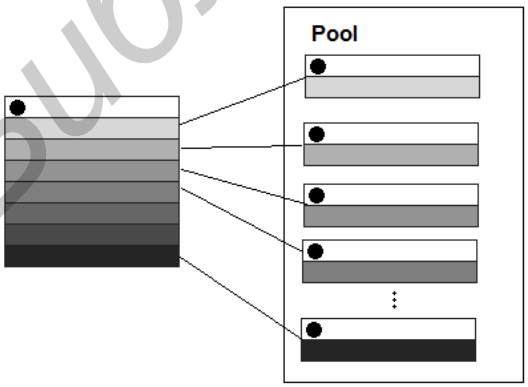

Рис. 3 – Распределенное решение СЛАУ

После проведения замеров времени вычисления, распределенный таким образом расчет не дал нужных результатов, из-за накладных расходов скорость вычисления повышается только на сверхбольших размерах матриц. Поэтому для практического применения данный подход не применим.

IV. Анализ общего прироста эффективности

На данный момент получилось ускорить вычисление умножения матриц. Полученная эффективность зависит от размера матриц и количества узлов в кластере. Чем больше матрицы, тем более ощутимый прирост скорости. Это объясняется дополнительными накладными расходами на разбиение задачи на подзадачи и передачу данных.

Заключение

В результате проведенного исследования, можно совершенно уверенно сказать, что использование распределенных вычислений позволяют ускорить решение задачи даже на одном компьютере. Несмотря на то, что есть большое число библиотек линейной алгебры, они не применимы к распределенным вычислениям, а использование подхода map-reduce позволяет сделать вычисления легко расширяемыми, где основная сложность огранизации передачи и распределения данных между вычислительными узлами ложится на реализацию map-reduce.

1. Zienkiewicz, O.C. The finite element method for solid and structural mechanics. Sixth edition / O.C. Zienkiewicz, R.L. Taylor. – Oxford : Elsivier, 2005. – 631 p.
2. Информационные технологии и системы 2014: материалы междунар. конф. БГУИР, Минск, Беларусь, 29 октября 2014 г. / редкол. Л.Ю Шилин [и др.]. – Минск: БГУИР, 2014. – 336 с.
3. Норри, Д. Введение в метод конечных элементов: учебник / Д. Норри, Ж. де Фриз – Москва: Мир, 1981. – 298 с.
4. Strassen algorithm [Электронный ресурс]/ Wikipedia. – Режим доступа: https://en.wikipedia.org/wiki/Strassen-algorithm/ – Дата доступа: 14.09.2015.
5. Java Simon [Электронный ресурс]/ Github. – Режим доступа: https://github.com/virgo47/javasimon – Дата доступа: 14.09.2015.
6. Outer product [Электронный ресурс]/ Wikipedia. – Режим доступа: https://en.wikipedia.org/wiki/Outer-product – Дата доступа: 14.09.2015.