ОПТИМАЛЬНЫЕ МЕТОДЫ ВЫЧИСЛЕНИЯ МАТРИЦ ПРОИЗВОЛЬНОГО ПОРЯДКА СРЕДСТВАМИ ЯЗЫКА СИ С ПРИМЕНЕНИЕМ MPI

П. В. Гончаров

Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого», Беларусь

Научный руководитель А. В. Цитринов

Введение. Ускорение времени работы алгоритмов, работающих с большим объемом данных, – это важнейшая задача для программирования и современного мира в целом. Решив данную задачу, можно многократно повысить производительность и уменьшить время вычислений.

Вычисление определителя – одна из самых распространенных задач линейной алгебры. Данная задача имеет широкий спектр применения в таких проблемах, как вычисление обратной матрицы, определение ранга матрицы в ходе решения СЛАУ (метод Крамера), а также для подсчета векторных произведений.

Прямые методы вычисления определителя могут быть основаны непосредственно на его определении, как суммы по перестановкам, или на разложении Лапласа по определителям меньшего порядка. Однако такие методы очень неэффективны, так как требуют $O(n!)$ операций для вычисления определителя n-го порядка. Один из более быстрых методов заключается в простой модификации метода Гаусса. Следуя методу Гаусса, произвольную матрицу А можно привести к треугольному виду, используя лишь две следующие операции над матрицей – перестановку двух строк и добавление к одной из строк матрицы другой строки, умноженной на произвольное число. Сложность этого метода, как и метода Гаусса, составляет $O(n^3)$ [1].

Все, что связано с большими компьютерами и большими задачами, сопровождается характерным словом «параллельный». В широкое употребление этот термин вошел после осознания того факта, что созданные компьютеры не в состоянии решить за приемлемое время многие задачи. Но если один компьютер не справляется, то попробуем взять два, три, десять и заставим их работать одновременно над разными частями задачи, для ускорения расчетов [2]. Наиболее распространенной технологией программирования для параллельных вычислений с распределенной памятью в настоящее время является технология *MPI* (от англ. *MessagePassingInterface*) [3].

Описание метода и технологий. Для приведения матрицы к треугольному виду воспользуемся некоторыми свойствами определителя: определитель матрицы не изменится, если к какой-то его строке (столбцу) прибавить другую строку (столбец), умноженную на некоторое число; если поменять местами две строки (столбца) матрицы, то определитель матрицы поменяет знак. Таким образом, суть метода заключается в последовательном исключении элементов. Для начала нужно найти в первой строке ведущий элемент и разделить единицу на ведущий элемент – это нужно, чтобы потом многократно не выполнять «дорогостоящую» операцию деления. После того как рассчитан «разрешающий» элемент, нужно пройтись по всем строкам $i + 1$ и всем столбцам, отнимая от каждого элемента произведение:

$$A[k, j] = A[k, j] - A[i, j] \cdot divideEl \cdot A[k, i],$$

где A – это исходная матрица; i, j, k – номера соответствующих строк и столбцов ($k = i + 1$), а $divideEl$ – это разрешающий элемент, который мы нашли по формуле

$$divideEl = \frac{1{,}0}{A[i, i]}.$$

После того как исходная матрица приняла ступенчатый вид, определитель матрицы будет равен произведению элементов, стоящих на главной диагонали. В случае если элемент на главной диагонали равен 0, то следует поменять строки или столбцы, изменив при этом знак определителя. Такой же подход применим в случае метода Гаусса с поиском ведущего элемента, когда ищется наибольший по модулю элемент и меняется с элементом главной диагонали. Это необходимо для того, чтобы снизить погрешности вычислений при округлениях.

Основным способом взаимодействия параллельных процессов в распределенных системах является передача сообщений друг другу. Это и отражено в названии технологии – *MessagePassingInterface* (интерфейс передачи сообщений). Интерфейс *MPI* поддерживает создание параллельных программ в стиле *MMID* (*MultipleInstructionMultipleData*), что подразумевает объединение процессов с различными исходными текстами. Однако разрабатывать и отлаживать такие программы очень сложно, поэтому на практике программисты чаще используют *SPMD*-модель (*SingleProgramMultipleData*) параллельного программирования, в рамках которой для всех параллельных процессов используется один и тот же программный код [3]. Далее в представленной работе использована *SMPD*-модель.

Сравнительные данные по результатам измерений. Для тестирования программы распределенных вычислений была использована аппаратная платформа на базе двухъядерного процессора *IntelCorei*3-4500*U* и тактовой частотой 1,70 ГГц. Для оптимизации количества процессов, при котором эффективность работы приложения будет максимальной, проведено тестирование и измерение скорости выполнения программы в зависимости от количества процессов. Результаты анализа представлены на рис. 1.

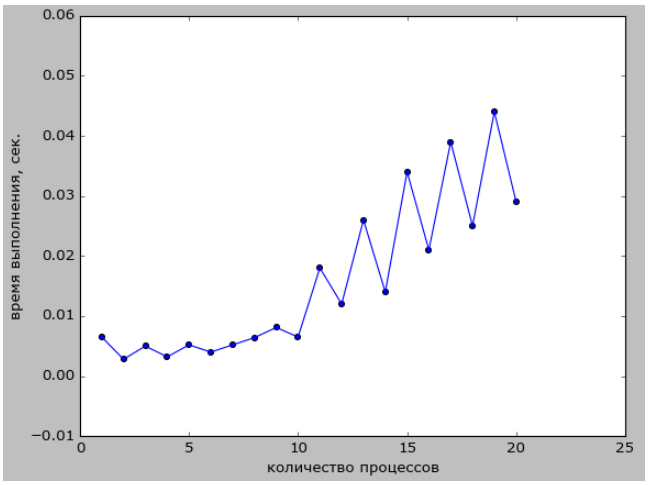

Рис. 1. Зависимость времени выполнения программы от количества процессов (размерность матрицы равна 100)

Как видно из рис. 1, при двух процессах достигнута максимальная эффективность выполнения. Также можно заметить, что на графике присутствуют «скачки» – это можно объяснить тем, что так как количество ядер процессора имеет четное число, то программа будет работать быстрее для четного количества процессов. Причем при увеличении числа процессов до 6 и выше скорость выполнения приложения заметно падает.

После того как эмпирическим путем было выбрано оптимальное число процессов для организации параллельных вычислений, приступим к сравнению скорости работы программного комплекса на разных платформах в однопоточном режиме, а также с использованием технологии MPI. Измерение времени работы программы для нераспределенных программ на Си и *C#*, а также программы с использованием технологии *MPI* представлено в виде графика на рис. 2.

На рис. 2 отлично видно, что реализация алгоритма на языке C# работает примерно в 4 раза медленнее, чем однопоточный вариант на языке Си. В свою очередь, используя многопоточный вариант программы с применением технологии *MPI*, мож-

но повысить производительность в 2 раза. Однако это утверждение верно только для матриц, размерность которых больше 40. Для матриц меньшей размерности применение технологий параллельных вычислений практически не имеет смысла, а для матриц малых размерностей (5 × 5) наоборот проигрывает в скорости однопоточной реализации, за счет слишком больших расходов на создание нескольких потоков.

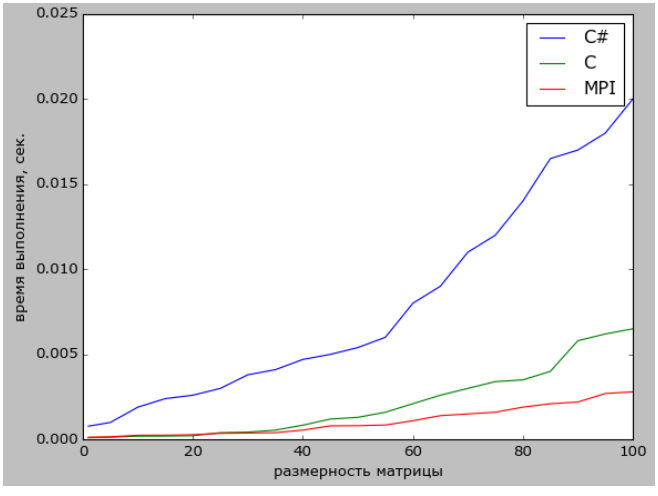

Рис. 2. Сравнение времени работы программы на разных платформах

Заключение. В современном мире с каждым днем данных становится все больше и вычисления занимают все больше времени. Применение технологий распределенных вычислений позволяет достичь максимальной эффективности в работе и задействует наибольшее количество ресурсов процессора, для ускорения выполнения программы. В ходе данного исследования было разработано три программы на разных платформах в однопоточном варианте и многопоточный вариант на языке Си; проведен сравнительный анализ скорости выполнения программного комплекса в зависимости от языка программирования и количества потоков. Многопоточное (2 потока) приложение на языке Си работает быстрее других вариантов, при условии, что размерность матрицы больше 40.

Литература

1. Ильин, В. А. Линейная алгебра / В. А. Ильин, Э. Г. Позняк. – М. : Наука – Физматлит, 1999.
2. Воеводин, В. В. Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. – СПб., 2002.
3. Антонов, А. С. Параллельное программирование с использованием технологии MPI : учеб. пособие / А. С. Антонов. – М. : Изд-во МГУ, 2004.