Министерство образования и науки Российской Федерации
Государственное образовательное учреждение высшего профессионального образования «Нижегородский государственный университет им. Н.И. Лобачевского»

Институт информационных технологий, математики и механики

Кафедра: программная инженерия

Специальность (направление): Программная инженерия

Отчет

по лабораторной работе
по дисциплине «Параллельное программирование»

тема:
«Вычисление детерминантов для плотной матрицы»

Выполнил(а): студент(ка) группы 0828
Рябинин К.А._____
подпись

Научный руководитель:
Лебедев И.Г. _____
подпись

Нижний Новгород
2016

Содержание

Постановка задачи

Разработать программу, которая будет считать определитель (детерминант) плотной матрицы, разработать параллельный алгоритм подсчета, сравнить результаты последовательного и параллельного алгоритма.

Описание алгоритма

Плотная матрица – это матрица, в которой большая часть элементов матрицы ненулевые.

В основе лежит алгоритм Гаусса. Используем квадратную матрицу A размером NxN. Необходимо вычислить определитель матрицы A. Воспользуемся идеями метода Гаусса решения систем линейных уравнений.

Дана система:
a11 x1 + a12 x2 + ... + a1n xn = b1
a21 x1 + a22 x2 + ... + a2n xn = b2
...
an1 x1 + an2 x2 + ... + ann xn = bn

Выполним следующий алгоритм:

На первом шаге в первом столбце ищется наибольший по модулю элемент, уравнение с этим элементом перемещается на первую строчку (обменяв две соответствующие строки матрицы A и два соответствующих элемента вектора B), а затем отнимается это уравнение от всех остальных, чтобы в первом столбце все элементы (кроме первого) обратились в нуль. Например, при прибавлении ко второй строке будем домножать первую строку на -a21/a11, при добавлении к третьей -на -a31/a11, и т.д.

На втором шаге во втором столбце, начиная со второй строки, ищется наибольший по модулю элемент, поставим уравнение с этим элементом на вторую строчку, и будем отнимать это уравнение от всех остальных (в том числе и от первого), чтобы во втором столбце все элементы (кроме второго) обратились в ноль. Понятно, что эта операция никак не изменит первый столбец ведь от каждой строки мы будем отнимать вторую строку, умноженную на некоторый коэффициент, а во второй строке в первом столбце стоит нуль.

Т.е. на i-ом шаге найдём в i-ом столбце, начиная с i-го элемента, наибольший по модулю элемент, поставим уравнение с этим элементом на i-ю строчку, и будем отнимать это уравнение от всех остальных. Понятно, что это никак не повлияет на все предыдущие столбцы (с первого по (i-1)-ый).

В результате, система приведена к так называемому диагональному виду:
c11 x1 = d1
c22 x2 = d2
...
cnn xn = dn

Замечание 1.

На каждой итерации найдётся хотя бы один ненулевой элемент, иначе система бы имела нулевой определитель, что противоречит условию.

Будем выполнять те же самые действия, что и при решении системы линейных уравнений, исключив только деление текущей строки на a[i][i] (точнее, само деление можно выполнять, но подразумевая, что число выносится за знак определителя). Тогда все операции, которые мы будем производить с матрицей, не будут изменять величину определителя матрицы, за исключением, быть может, знака (мы только обмениваем местами две строки, что меняет знак на противоположный, или прибавляем одну строку к другой, что не меняет величину определителя).

Но матрица, к которой мы приходим после выполнения алгоритма Гаусса, является диагональной, и определитель её равен произведению элементов, стоящих на диагонали. Знак, как уже говорилось, будет определяться количеством обменов строк (если их нечётное, то знак определителя следует изменить на противоположный). Таким образом, мы можем с помощью алгоритма Гаусса вычислять определитель матрицы за O(n^3). Заметим, что если в какой-то момент мы не найдём в текущем столбце ненулевого элемента, то алгоритм следует остановить и вернуть 0.

Для вычисления определителя матрицы методом Гаусса необходимо привести матрицу к треугольному виду.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{2n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

Алгоритм заключается в следующем:

1. Разделим элементы каждой строки на первый элемент соответствующей строки:

$$|A| = \begin{pmatrix} 1 & \frac{a_{12}}{a_{11}} & \dots & \frac{a_{1n}}{a_{11}} \\ 1 & \frac{a_{22}}{a_{21}} & \dots & \frac{a_{2n}}{a_{21}} \\ \dots & \dots & \dots & \dots \\ 1 & \frac{a_{n2}}{a_{n1}} & \dots & \frac{a_{nn}}{a_{n1}} \end{pmatrix}$$

2. Вычтем из элементов всех строк, начиная со второй, элементы первой строки:

$$|A| = \begin{pmatrix} 1 & \frac{a_{12}}{a_{11}} & \dots & \frac{a_{1n}}{a_{11}} \\ 0 & \frac{a_{22}}{a_{21}} - \frac{a_{12}}{a_{11}} & \dots & \frac{a_{2n}}{a_{21}} - \frac{a_{1n}}{a_{11}} \\ \dots & \dots & \dots & \dots \\ 0 & \frac{a_{n2}}{a_{n1}} - \frac{a_{12}}{a_{11}} & \dots & \frac{a_{nn}}{a_{n1}} - \frac{a_{1n}}{a_{11}} \end{pmatrix}$$

3. Повторим данный алгоритм для внутренних строк. Продолжать будем до тех пор, пока размер конечной матрицы не станет размера 1x1:

Пример нахождения определителя методом Гаусса

Дано:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 8 \end{pmatrix}$$

Решение:

Делим 1 строку на 1, получаем:

$$\begin{pmatrix} 1 & 2 \\ 3 & 8 \end{pmatrix}$$

Делим 2 строку на 3, получаем:

$$\begin{pmatrix} 1 & 2 \\ 1 & \frac{8}{3} \end{pmatrix}$$

Вычитаем 1-ю строку из последующих

$$\begin{pmatrix} 1 & 2 \\ 0 & \frac{2}{3} \end{pmatrix}$$

Умножаем первые числа каждой строки (которые исполльзовались при сокращении):

$$det = 1 \cdot 3 \cdot \frac{2}{3} = 2$$

$$|A| = \begin{pmatrix} 1 & 2 \\ 3 & 8 \end{pmatrix} = 2$$

Ответ:

$$|A| = 2$$

Описание программной реализации

При запуске программы, задается размер матрицы и количество процессов, на которых будет выполняться программа. Затем случайным образом генерируется матрица размера N строк на N столбцов. После чего выполняется последовательный и параллельный алгоритмы выполнения программы и вычисляется время работы каждого алгоритма.

Последовательный алгоритм:

При запуске последовательного алгоритма сгенерированная матрица копируется в буфер для ее изменения. После выполнения алгоритма Гаусса, для приведения матрицы к верхне-диагональному виду, перемножаются ее диагональные элементы и определяется знак детерминанта, это произведение является определителем матрицы. В случае, если матрица не приведена к верхне-треугольному виду, делается вывод, что детерминант матрицы равен нулю.

Алгоритм:

- Идем по всем строчкам матрицы:

    1. Ищем строку с максимальным первым элементом
    2. Переставляем эту строку на первое место:
        - Увеличиваем счетчик перестановок на 1
    3. Из всех строк, что ниже, вычитаем самую верхнюю строку, умноженную на отношение первого элемента вычитающей строки к первому элементу вычитаемой строки
    4. Уменьшаем размерность матрицы на 1
    5. Пока размерность матрицы не достигнет единицы:
        - Повторяем предыдущую операцию

- Перемножаем все элементы, лежащие на диагонали
- Умножаем получившееся произведение на минус единицу в степени количества перестановок строк

Параллельный алгоритм:

Нулевой процесс рассылает всем процессам их доли строк исходной матрицы. После чего запускается алгоритм Гаусса. На каждой итерации алгоритма Гаусса (максимум их n - 1) определяется строка из оставшихся с максимальным ведущим элементом и переставляется местами с текущей строкой, при этом если строки, которые нужно обменять местами находятся на разных процессах, происходит обмен строками между процессами. При каждом обмене строками на ведущем процессе считается количество обменов.

После приведения матрицы к верхне-треугольному виду каждый процесс перемножает диагональные элементы своих строк. После этого используя MPI_Reduce с каждого процесса собираются и перемножаются произведения и суммируются значения счетчиков. Затем в зависимости от количества перестановок определяется знак детерминанта.

Алгоритм:

- Создаем карту распределения матрицы по строкам
- Рассылаем каждому процессу свои строки
- Идем по всем строчкам матрицы:
  - Определяем какому процессу принадлежит строка, в которой мы находимся
  - Каждый процесс:
    - Среди своих строк ищет строку с максимальным первым элементом
    - Отправляет эту строку ведущему процессу
  - Ведущий процесс:
    - Принимает строки с максимальными элементами
    - Среди них ищет строку с самым наибольшим первым элементом
    - Если такая строка принадлежит ведущему процессу:
      - Если такая строка –не текущая строка:
        -Увеличить счетчик перестановок на единицу
      - Ставим эту строку на самый верх матрицы
    - Если такая строка принадлежит другому процессу:
      - То принимаем эту строку от процесса, которому она принадлежит
      - Взамен отправляем ему текущую строку
      - Увеличить счетчик перестановок на единицу
      - Строку, которую приняли, ставим на самый верх матрицы
    - Отправляем строку с самым максимальным элементом всем процессам
  - Каждый процесс:
    - Принимает строчку с максимальным элементом
    - Из всех своих строк, которые находятся ниже текущей строки, вычитает самую верхнюю строку, умноженную на отношение первого элемента вычитающей строки к первому элементу вычитаемой строки
  - Уменьшаем размерность матрицы на 1
  - Пока размерность матрицы не достигнет единицы: повторяем всю процедуру
- Каждый процесс:
  - Перемножает все ненулевые элементы своих строк
  - Отправляет это произведение главному процессу

- Главный процесс:
  - Принимает все произведения от всех процессов
  - Перемножает полученные данные
- Итоговое произведение умножает на минус единицу в степени количества перестановок строк

Проверка работоспособности

Для проверки работоспособности параллельного варианта результат нужно сравнить с последовательным вариантом. В программе реализован последовательный и параллельный вариант и сравнение их результатов. При сравнении надо учитывать, что допустима некоторая небольшая погрешность, в результате которой результаты параллельной и последовательной версии могут отличаться на некоторое малое значение. Если результат параллельной версии отличаются от последовательной более допустимой погрешности то параллельный вариант считается не корректным, в ином случае — корректным.

Результаты экспериментов

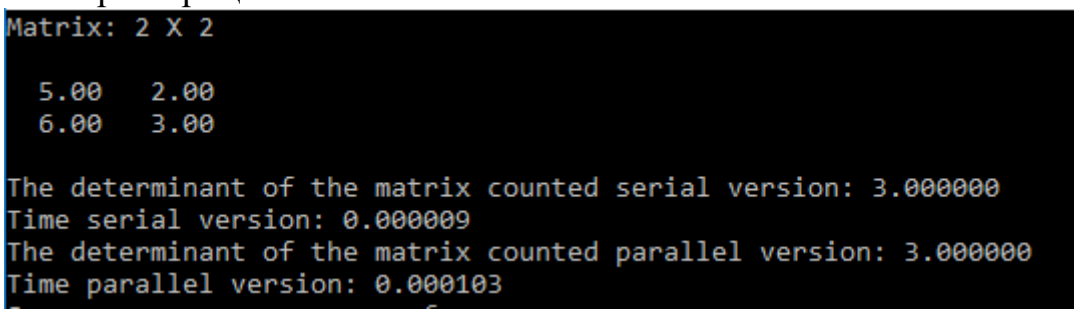
Размерность матрицы NxN	Время работы последовательного алгоритма	Время работы параллельного алгоритма
10x10	0.000010	0.000064
50x50	0.000520	0.000622
100x100	0.004285	0.004048
500x500	0.535937	0.468156
1000x1000	4.300668	3.853454
1500x1500	14.501149	13.274149
3000x3000	118.944722	105.254454

Эксперимент проводился на машине Intel Core i3-M370 2.4GHz 3GB, NVIDIA GeForce GT 415M 1GB OS Windows 10.

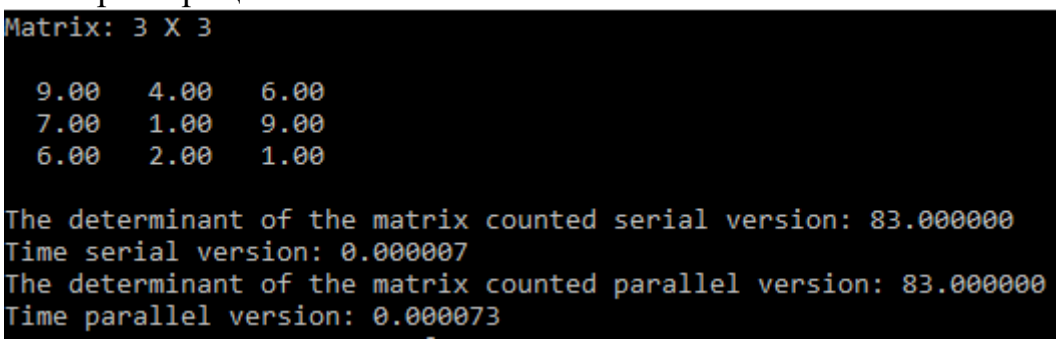
Исходя из результатов, записанных в таблице видно, что последовательный вариант выполняется быстрее параллельного при матрицах небольшого размера, но при увеличении матрицы параллельный вариант начинает ускоряться относительно последовательного. Это объясняется тем, что значительное количество времени тратится на пересылку данных. Для матрицы n x n нужно передать около O(n^2) векторов. Однако параллельная версия сможет посчитать матрицы достаточно большого объема используя распределенную память значительно быстрее, чем параллельный вариант.

Результаты работы программы:

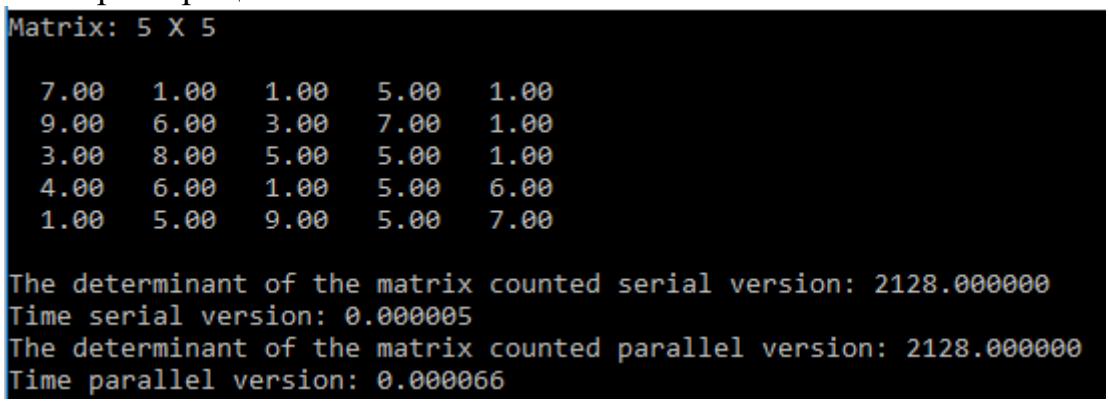
1) Размер матрицы 2x2



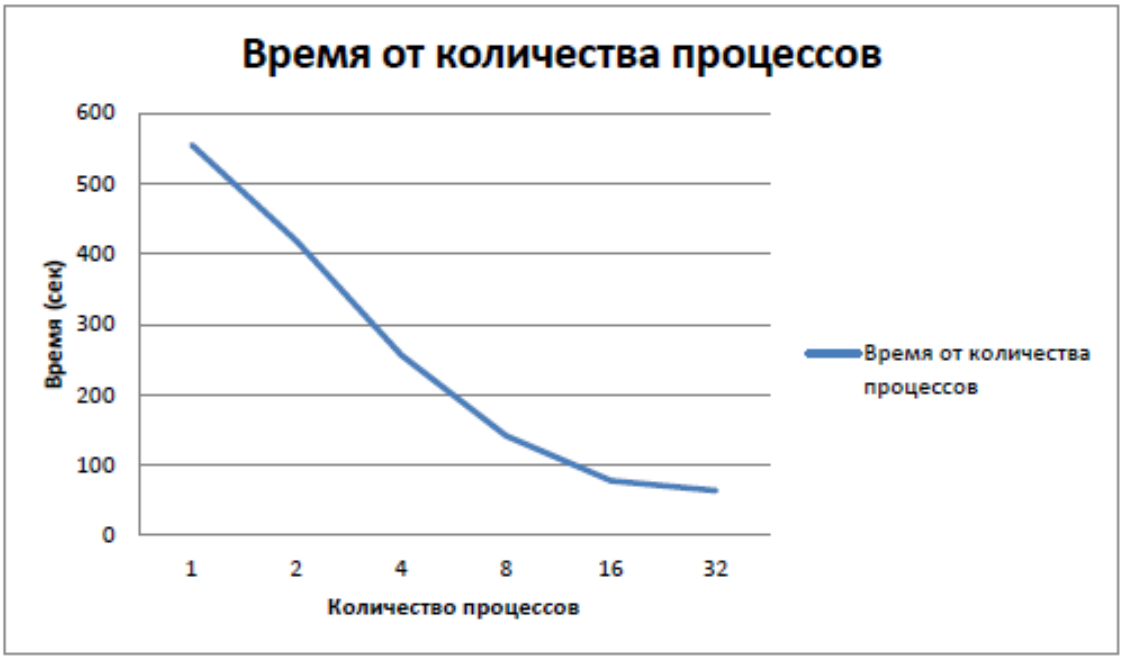
2) Размер матрицы 3x3



3) Размер матрицы 5x5



Результаты экспериментов на кластере



Исходя из полученного графика, мы видим, что с увеличением количества процессов в два раза, время работы уменьшается практически в 1.5-2 раза.

Вывод

В результате выполнение лабораторной работы написана программа, которая позволяет с помощью последовательного и параллельного алгоритмов вычислять детерминант плотной матрицы.

Было выяснено, что время работы параллельной версии больше времени последовательной версии при небольших размерах матрицы. При этом с ростом размера матрицы параллельный вариант будет значительно быстрее последовательного за счет возможности хранить большие матрицы целиком в распределенной памяти, а не на диске, который значительно медленнее памяти. Также было установлено, время выполнения программы уменьшается в 2 раза, при увеличении количества процессов в два раза.