ЛАБОРАТОРНАЯ РАБОТА № 2

ПАРАЛЛЕЛЬНОЕ ВЫЧИСЛЕНИЕ ПРОИЗВЕДЕНИЯ ДВУХ МАТРИЦ СРЕДСТВАМИ OPENMP

1. Цель работы

Приобрести навыки распараллеливания вложенных циклов с использованием директив OpenMP. Исследовать ускорение, эффективность и производительность многопоточных реализаций алгоритмов решения задачи матричного умножения.

2. Постановка задачи и описание алгоритма

Пусть требуется найти произведение двух квадратных матриц A и B

$$C = AB$$

или

$$c_{ij} = \sum_{k=1}^{N} a_{ik} b_{kj}, \quad i, j = 1, \ldots, N. \tag{4.1}$$

Взяв за основу простейший последовательный алгоритм перемножения, в котором элементы результирующей матрицы C вычисляются построчно по формуле (4.1), рассмотрим возможности его параллельной реализации.

Из (4.1) видно, что для получения c_{ij} требуется вычислить сумму, составленную из N произведений $a_{ik} b_{kj}$. Так как все произведения могут быть вычислены независимо, вычисление каждого элемента матрицы C сводится к задаче нахождения суммы числового ряда, параллельный алгоритм решения которой был рассмотрен в лабораторной работе №1. Таким образом, получаем параллельный алгоритм №1, который рассчитан на использование множества вычислительных устройств для определения каждого из элементов результирующей матрицы (см. рис. 4.1).

Следующий уровень параллелизма, заложенный в исходной задаче, заключается в том, что независимо могут быть вычислены отдельные части матрицы C – строки, столбцы, блоки.

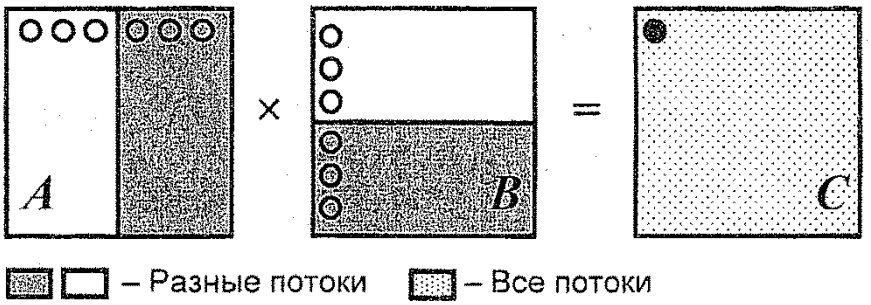


Рис. 4.1. Параллельное умножение матриц (алгоритм №1)

Разбиение результирующей матрицы по одному из измерений на части, которые могут быть найдены независимо, дает еще два возможных варианта параллельного исполнения. Если C делится по столбцам (алгоритм №2), то для вычисления одной части требуются соответствующая часть матрицы B и вся матрица A, как показано на рис. 4.2.

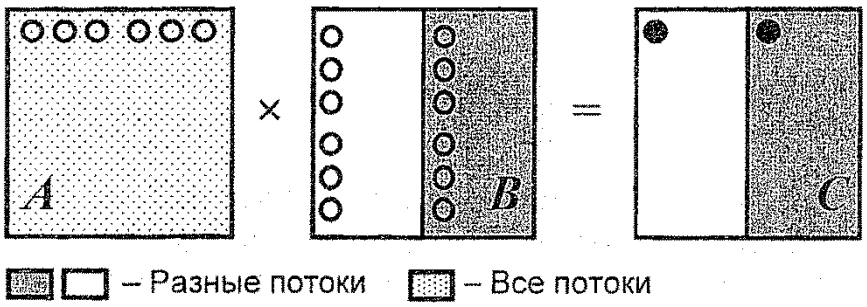


Рис. 4.2. Параллельное умножение матриц (алгоритм №2)

При делении по строкам (алгоритм №3), наоборот, для вычисления части C требуются соответствующая часть матрицы A и полная матрица B (рис. 4.3).

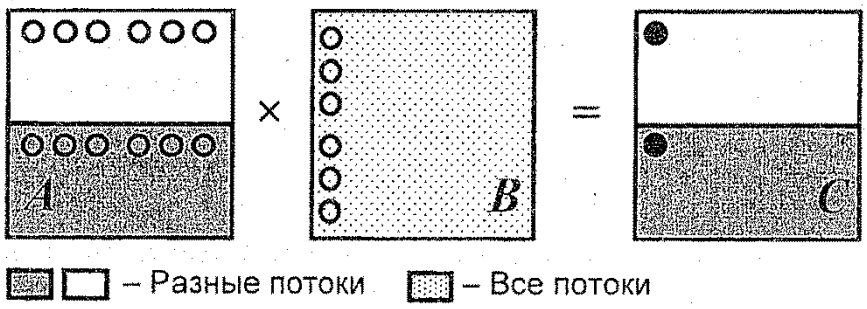


Рис. 4.3. Параллельное умножение матриц (алгоритм №3)

Также возможен вариант разбиения C по двум измерениям на блоки размером $M \times M$ так, что отношение $nb = \frac{N}{M}$ является целым числом. Тогда матрицу C можно рассматривать как nb^2 квадратных блоков, каждый из которых может быть обработан независимо. Для того чтобы вычислить один такой блок, потребуется M строк матрицы A и M столбцов матрицы B.

Для того чтобы определить, насколько эффективно используется вычислительная система при выполнении конкретного алгоритма, рассмотрим понятия реальной и пиковой производительности.

Определение 4.1. Пиковой (теоретической) производительностью называется максимальное количество команд или операций, которое может теоретически выполнять вычислительная система в единицу времени при условии постоянной и полной загрузки всех ее исполнительных устройств.

Пиковая производительность чаще всего измеряется в количестве выполняемых в секунду операций с плавающей точкой – Flops (Floating point operation per second). Формула для расчета пиковой производительности R_{peak} имеет вид

$$R_{peak} = pn\nu, \tag{4.2}$$

где p – количество процессоров или ядер вычислительной системы;

n – теоретическое количество операций с плавающей точкой, которое может выполнять процессор или ядро за 1 такт;

ν – тактовая частота, на которой работает процессор (ядро).

При расчете пиковой производительности обычно учитываются операции с вещественными числами двойной точности, а значение n определяется по числу функциональных устройств в процессоре или ядре, выполняющих элементарные операции (обычно сложение и умножение) с этими числами.

Приведем пример расчета пиковой производительности системы с двуядерным процессором Intel Core 2 Duo 6300:

$p = 2$, $n = 4$, $\nu = 1.87 \cdot 10^9$ 1/с.

$R_{peak} = 2 \times 4 \times 1.87 \cdot 10^9 = 14.96 \cdot 10^9$ Flops.

Таким образом, вычислительная система на базе данного процессора теоретически может выполнять 15 миллиардов операций с вещественными числами двойной точности в секунду.

Определение 4.2. Реальной производительностью называется количество команд или операций, которое выполняет вычислительная система в единицу времени для конкретного алгоритма (программы).

По аналогии с пиковой, реальная производительность также может оцениваться количеством операций, выполняемых с вещественными числами в секунду.

Для оценки реальной производительности в работе определим, сколько операций должно быть выполнено при перемножении квадратных матриц $A(N \times N)$ и $B(N \times N)$. Чтобы вычислить один элемент результирующей матрицы $C(N \times N)$, необходимо выполнить скалярное умножение строки матрицы A на столбец матрицы B – требуется совершить N операций умножения и столько же операций сложения. Таким образом, для вычисления одного элемента C требуется $2N$ вещественных операций, для вычисления полной матрицы C – $2N^3$ вещественных операций.

Поскольку на современных вычислительных системах время, затрачиваемое на умножение матриц при размерности $N = 100$, составляет порядка миллисекунды, а при $N = 10$ – всего порядка микросекунды, для проведения надежных временных замеров необходимо многократно повторить умножение. Количество повторов требуется выбирать индивидуально для каждого N, чтобы время вычислений в программе измерялось секундами. Тогда если за время T осуществляется q матричных умножений, получим, что достигаемую (реальную) производительность R_{real} можно выразить формулами

$$R^1_{real} = 2\frac{qN^3}{T_1} \quad \text{и} \quad R^p_{real} = 2\frac{qN^3}{T_p} \tag{4.3}$$

для последовательного и параллельного алгоритма матричного умножения соответственно.

Чтобы оценить эффективность использования вычислительной системы при выполнении последовательного и параллельного алгоритмов, введем коэффициенты

$$U_1 = \frac{R^1_{real}}{R_{peak}} \quad \text{и} \quad U_p = \frac{R^p_{real}}{R_{peak}}. \tag{4.4}$$

Данные коэффициенты всегда будут принимать значения меньшие единицы, так как пиковая производительность вычислительной системы на практике никогда не достигается.

3. Особенности программной реализации алгоритма с использованием интерфейса OpenMP

Обычно при написании последовательной программы на языке высокого уровня алгоритм умножения матриц реализуется в виде трех вложенных циклов. Используя интерфейс OpenMP можно распределить между потоками итерации любого из этих циклов. Пусть для определенности внешнему циклу соответствует счетчик i, среднему – j, а внутреннему – k (в соответствии с обозначениями (4.1)). Тогда распараллеливание внутреннего цикла будет соответствовать алгоритму 1 п. 2, то есть вычисление суммы по k из (4.1) будет распределено по потокам, в результате чего над вычислением каждого элемента c_{ij} будут параллельно работать все потоки. Распараллеливание среднего цикла (по j) соответствует алгоритму 2 из п. 2, то есть осуществляется столбцевая декомпозиция матрицы C. Наконец, распараллеливание внешнего цикла (по i) приводит к алгоритму 3 п. 2, когда имеет место строковая декомпозиция матрицы C.

В случае, когда параллельный алгоритм рассчитан на решение задач различных размерностей, важно иметь представление о том, при каких размерностях параллельные вычисления могут замедлить работу. В OpenMP программах такой эффект возникает, например, при распараллеливании циклов с небольшим числом итераций, что может быть связано с наличием накладных расходов на поддержку многопоточности, необходимостью синхронизации потоков и т.д. Для того чтобы избежать порождения параллельной области при решении задачи малой размерности, можно использовать спецификатор `if` директив `parallel` и `parallel for`:

```
    double prod = 2;
#pragma omp parallel for if(N>10) reduction(*:pr)
    for (i = 1; i <= N; i++)
        prod *= i / (1. + i);
```

В данном случае, если цикл состоит из более 10 итераций, то произойдет порождение параллельной области и итерации цикла будут распределены между потоками, иначе цикл будет выполняться последовательно.

4. Выполнение работы

1. Выполнить программную реализацию на языке C последовательного алгоритма умножения двух квадратных матриц размера $N \times N$. Предусмотреть:

а) ввод количества повторов умножения q пользователем с клавиатуры;
б) статическое выделение памяти для хранения матриц;
в) заполнение матриц случайными вещественными числами в диапазоне от -0.5 до 0.5;
г) повторение процедуры умножения матриц в цикле q раз;
д) вывод на экран всех (четырех) угловых элементов результирующей матрицы для сравнения результатов работы различных версий программ;
е) вывод на экран времени работы программы.

Время замерять при помощи функции `clock` (см. лаб. раб. №2).

2. Произвести распараллеливание матричного умножения путем добавления директив OpenMP в текст последовательной программы. Реализовать три алгортма параллельного умножения, предложенные выше в части 2. Сравнить написанные программы с приведенными в прил. Е.

3. Отладить написанные программы при небольших размерностях матриц N на многоядерной вычислительной системе. Убедиться, что результат работы параллельных программ совпадает с полученным в последовательной версии и является стабильными при многократных запусках.

4. Запустить все три параллельные версии при размерности матриц $N = 100$ и количестве повторов q таком, что время работы каждой из программ составляло бы порядка 10 секунд. Выбрать наиболее производительную версию для дальнейшей работы.

5. Запустить последовательную программу при различных размерностях матриц $N = 5, 10, 50, 100, 200, 500$, выбирая q так, чтобы время работы программы составляло не менее 10 секунд. Запустить параллельную программу при аналогичных значениях N и q.

6. Время работы каждой программы занести в табл. 4.1.

Таблица 4.1

N	q	T_1	T_p
5			
10			
50			
100			
200			
500			

7. Вычислить ускорение и эффективность по формулам (1.3) и (1.4) (см. лаб. раб. №1) для каждого N, полученные значения занести в табл. 4.2.

Таблица 4.2

N	S	E
5		
10		
50		
100		
200		
500		

8. Построить график зависимости ускорения параллельной программы от размерности умножаемых матриц. Объяснить полученные результаты.

9. Определить пиковую производительность одного ядра и всей многоядерной системы R_p, на которой производятся вычисления.

10. Вычислить количество вещественных операций, выполняемых при матричном умножении для каждого N, реальную производительность, достигнутую в последовательной (R^1_{real}) и

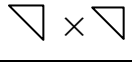
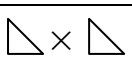
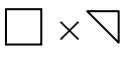
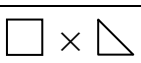
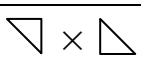
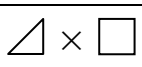
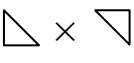
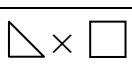
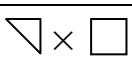
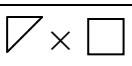
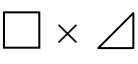
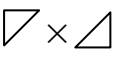
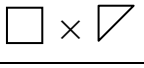
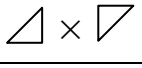
параллельной (R^p_{real}) версиях для каждой размерности по формулам (4.3), и отношения реальной производительности к пиковой U_1 и U_p по формулам (4.4). Полученные значения занести в табл. 4.3.

Таблица 4.3

N	R^1_{real}	R^p_{real}	U_1	U_p
5				
10				
50				
100				
200				
500				

11. Для размерности с максимальным ускорением запустить расчет для числа потоков p = 1, 2, 3, 4, 5, 6. Построить графики зависимости ускорения и эффективности от числа потоков. Объяснить полученные результаты.

12. Модифицировать программу (изменить только порядок и диапазон пробега индексов в циклах) для перемножения матриц специального вида согласно варианту (номеру в подгруппе):

№ вар.	Произведение матриц	№ вар.	Произведение матриц
1		10	 ширина ленты N/2
2			
3		11	 ширина ленты N/2
4			
5		12	
6		13	 ширина ленты N/2
7			
8		14	
9		15	
10		16	

13. Повторить расчеты п.5,7,11 для модифицированной программы. Построить графики и сравнить с полученными ранее для классического случая. Объяснить полученные результаты.

14. Оформить отчет по результатам работы.