ЛАБОРАТОРНАЯ РАБОТА № 4

ПАРАЛЛЕЛЬНОЕ ВЫЧИСЛЕНИЕ ПРОИЗВЕДЕНИЯ ДВУХ МАТРИЦ СРЕДСТВАМИ MPI

1. Цель работы

Научиться использовать принцип геометрической декомпозиции в параллельных алгоритмах и создавать параллельные программы для систем с распределенной памятью с использованием коллективных функций MPI на примере вычисления произведения двух матриц.

2. Теоретическая часть

Пусть требуется найти произведение матриц A $(N \times L)$ и B $(L \times L)$ и вычислить квадрат евклидовой нормы результирующей матрицы:

$$\|C\|^2 = \sum_{i=1}^{N} \sum_{j=1}^{L} c_{ij}^2, \qquad C = AB.$$

Задача матричного умножения

$$c_{ij} = \sum_{k=1}^{L} a_{ik} b_{kj}, \qquad i = 1, \ldots, N;\ j, k = 1, \ldots, L$$

является очень трудоемкой, поскольку требует для решения $2NL^2$ операций. Например, для перемножения двух матриц порядка $10^4 \times 10^4$ компьютер должен выполнить $2 \cdot 10^{12}$ арифметических операций. Во многих практических вычислительных задачах необходимость вычисления матричных произведений возникает очень часто, причем в рамках каждой задачи делать это приходится многократно. Поэтому распараллеливание этой операции часто позволяет довольно существенно сократить продолжительность вычислительного процесса.

Все параллельные матричные алгоритмы строятся на основе принципа геометрической декомпозиции данных, согласно которому исходные матрицы разбиваются на блоки, каждый из которых сопоставляется своему вычислительному процессу. Рассмотрим один из возможных способов декомпозиции для рассматриваемой задачи.

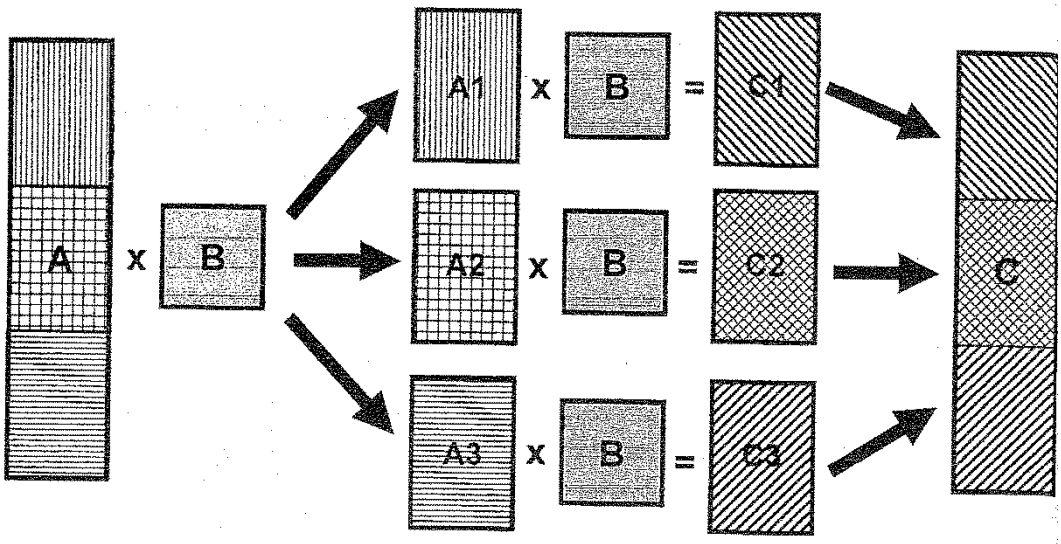

Рис. 2.1. Схема алгоритма со строчной непрерывной декомпозицией

Алгоритм матричного умножения может быть рассмотрен как процесс перемножения N независимых строк матрицы A на матрицу B (строковая декомпозиция матрицы A). Если предположить, что N кратно числу процессоров p, то схематично перемножение можно представить как

$$\begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{pmatrix} \cdot B = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_p \end{pmatrix},$$

где A_l, C_l $(l = 1,\ldots,p)$ – блоки матриц A и C, соответственно. Тогда становится очевидным распределение матриц по процессорам: процессор с номером l получает блок A_l матрицы A и всю матрицу B. Схема работы алгоритма при строчной непрерывной декомпозиции матрицы A показана на рис. 2.1. Так как нашей целью является вычисление квадрата евклидовой нормы, а указанный на схеме последний этап сбора блоков C_l в итоговую матрицу C является весьма трудоемким, можно вычислить на каждом процессоре сумму квадратов элементов своего блока C_l и осуществлять пересылку одного итогового числа.

3. Порядок выполнения работы

1. Написать параллельную программу перемножения двух квадратных матриц с использованием коллективных функций MPI. Размерность L вводится пользователем, размерность N = 10L. Матрицы A и B заполняются случайными элементами вещественного типа.

2. Выполнить отладку параллельной программы на персональном компьютере в режиме эмуляции многопроцессорного режима. Отладку провести сначала на матрицах малой размерности для возможности проверки правильности вычислений.

3. Написать параллельную программу вычисления матричного выражения согласно варианту (номеру в подгруппе):

№ вар.	Матричное выражение	№ вар.	Матричное выражение	№ вар.	Матричное выражение
1	$cABd^T$	7	aBC–d	13	A^2+B^2
2	AB+CD	8	$(A+B)^2$	14	a(BC–D)
3	$(A–BC)d^T$	9	aBC–dF	15	A^2+BC
4	aB^2+c	10	(A+B)(C+D)	16	(A–B)C
5	(A+B)C+D	11	AB–CD	17	$A^2b^T+Cd^T$
6	A(B+C)	12	A^2+B	18	(A+B)(C–D)

Матрицы (N×N) обозначены заглавными буквами, вектора-строки (1×N) – строчными. Размерность N вводится пользователем, матрицы и вектора заполняются случайными элементами вещественного типа.

4. Выполнить отладку параллельной программы на персональном компьютере в режиме эмуляции многопроцессорного режима. Отладку провести сначала на матрицах малой размерности для возможности проверки правильности вычислений.

5. Отлаженные программы запустить при различном числе процессоров. В каждой программе выбрать размерность так, чтобы время выполнения параллельной программы на одном процессоре T_1 составляло порядка 500 с. Последовательно просчитать варианты запуска при числе процессов p = 1, 2, 4, 6, 8. Полученные данные о продолжительности вычислительного процесса занести в таблицу 2.1, аналогичную таблице 1 предыдущей лабораторной работы.

6. Отлаженные программы запустить на Суперкомпьютере согласно инструкции, используя систему очередей. В каждой программе выбрать размерность так, чтобы время выполнения параллельной программы на одном процессоре T_1 составляло порядка 500 с. Последовательно просчитать варианты запуска на 1, 2, 3, 4 узлах при числе процессоров p = 1, 2, 4, 6, 8. Полученные данные о продолжительности вычислительного процесса занести в таблицу 2.2, аналогичную таблице 1 предыдущей лабораторной работы.

7. Вычислить ускорение и эффективность, полученные результаты занести в таблицы, аналогичные таблице 2 предыдущей

лабораторной работы. Построить графики зависимости ускорения и эффективности от числа процессоров для каждой программы. Сравнить случаи использования одинакового общего числа процессоров при разном количестве узлов.

8. Оформить отчет по результатам работы.