

МО ВВС ИВМ и МГ СО РАН

Фрагментированное программирование

Чаюк Ксения

Структура лекции

- **Что такое фрагментированное программирование**
- **Введение**
 - Необходимые свойства параллельной программы
- **Качественная картина исполнения фрагментированной программы**
- **Пример фрагментированного алгоритма**

Что такое фрагментированное программирование

Фрагментированное программирование – это технология программирования, при которой программа собирается из фрагментов вычислений.

Фрагмент кода - это некоторая операция (вычисление функции), которая в программе может реализоваться процедурой, подпрограммой и т.п. Для фрагмента кода определены входные и выходные переменные, которые являются его формальными параметрами.

Фрагмент данных – это часть общей структуры данных (расчетной области) для обработки в некотором фрагменте.

Пара <фрагмент кода, фрагмент данных> называется **фрагментом вычислений (фрагментом)**.

Введение

- Математическое моделирование
- Распространенные средства параллельного программирования
- Основные сложности параллельного программирования

Необходимые свойства параллельной программы

1. Параллельная программа представляется как множество параллельно протекающих и взаимодействующих последовательных процессов
2. Недетерминизм исполнения
3. Гибкая настройка на все доступные ресурсы мультимпьютера
4. Динамическая балансировка загрузки
5. Переносимость в классе мультимпьютеров
6. Динамизм поведения

Качественная картина исполнения фрагментированной программы



Умножение матриц

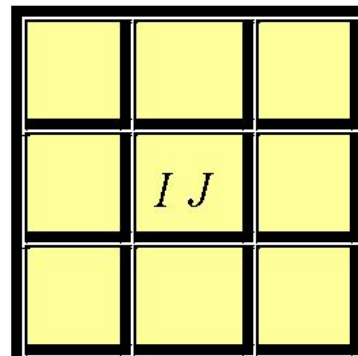
Последовательный алгоритм

Алгоритм умножения квадратных матриц размера $N \times N$ имеет вид:

$$c_{i,j} = \sum_{k=1}^N a_{i,k} \times b_{k,j}, \quad i, j = 1, 2, \dots, N. \quad (1)$$

Фрагментированный алгоритм

Матрицы A , B и C разбивается на фрагменты данных одинакового размера. Под фрагментами данных понимаются блоки матрицы, содержащие M строк и M столбцов.

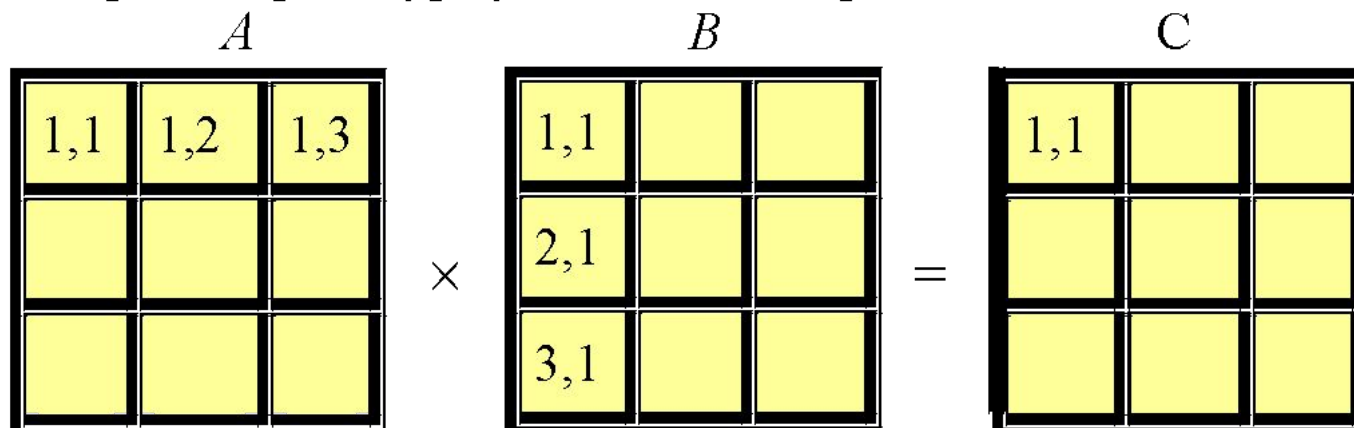


Умножение матриц

Для получения фрагмента данных $C_{I,J}$ используется формула

$$C_{I,J} = \sum_{L=1}^K A_{I,L} \times B_{L,J} \quad (2)$$

где $C_{I,J}, A_{I,L}, B_{L,J}, I, J, L = 1, 2, \dots, K$. Под умножением в ней понимается стандартная процедура умножения матриц.



$$C_{1,1} = A_{1,1} \times B_{1,1} + A_{1,2} \times B_{2,1} + A_{1,3} \times B_{3,1}$$

Реализация фрагментированного алгоритма умножение матриц

В реализованной фрагментированной версии размер фрагмента данных задается пользователем.

В качестве фрагмента кода используется последовательная процедура

умножения матриц вида $c_{i,j} = \sum_{k=1}^N a_{i,k} \times b_{k,j}$, $i, j = 1, 2, \dots, N$ к которой

добавлена процедура суммирования матриц. Суммирование производится сразу после вычисления очередного произведения $A_{i,L} \times B_{L,j}$ в

$$C_{i,j} = \sum_{L=1}^K A_{i,L} \times B_{L,j} .$$

Фрагмент кода применяется к фрагментам данных в соответствии с

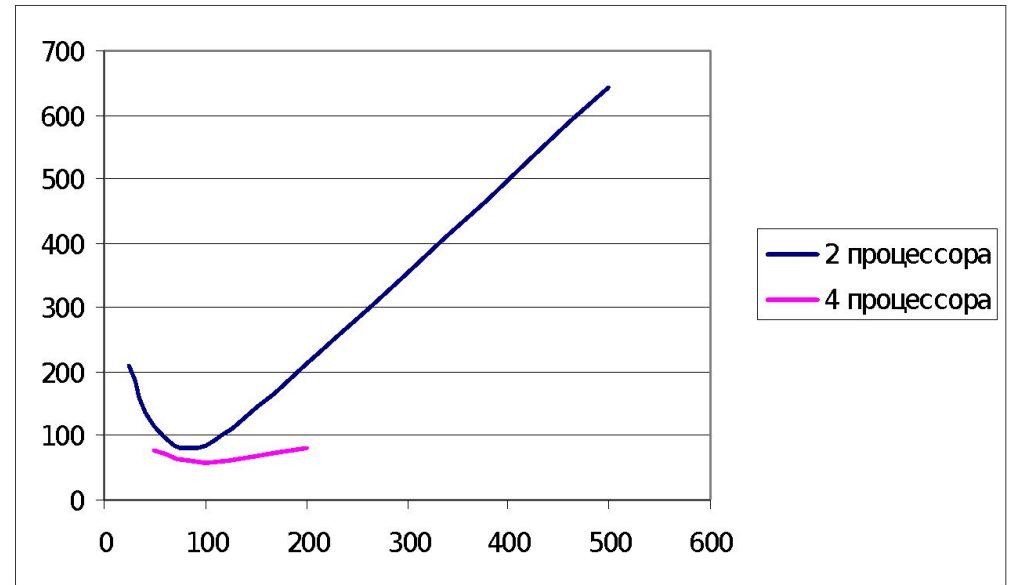
формулой $C_{i,j} = \sum_{L=1}^K A_{i,L} \times B_{L,j}$. Необходимые фрагменты выполняются в

произвольном порядке. Управление отвечает только за то, чтобы они все исполнились. Запуск организован таким образом, чтобы одновременно не исполнялись фрагменты, суммирующие результат в один и тот же блок матрицы C .

После исполнения всех фрагментов кода в матрице C будет получен результат умножения исходных матриц A и B .

Результаты тестирования

Матрица 5000x5000		
Количество потоков	Размер фрагмента	Время
последовательная	5000	658,06
1	100	161,94
2	25	208,09
	50	115,87
	100	85,70
	200	214,25
	500	642,76
4	50	77,99
	100	56,61
	200	81,65



Выводы

- Достигается высокое качество динамического распределения ресурсов и гибкая настройка параллельной программы на все доступные ресурсы, а значит и ее переносимость в классе мультимониторов.
- Фрагментация программы позволяет единообразно решить проблему организации межпроцессных обменов данными на фоне вычислений.
- Фрагментирование позволяет решить и другие проблемы, такие как накопление библиотек стандартных подпрограмм в платформонезависимой форме, масштабируемость программ, использование накопленных библиотек последовательных подпрограмм, оптимизация использования памяти ПЭ, и т.д.

Вопросы

Спасибо за внимание!