Проектирование параллельных алгоритмов.

Лекция 3

Методология проектирования

- Разделение
- Установление связей
- Агрегирование
- Привязка к конкретной ЭВМ

1 Разбиение

- Исходная задача разбивается на маленькие задачи. При этом число процессоров в конкретном компьютере игнорируется, внимание фокусируется на распознавании возможностей параллельных вычислений

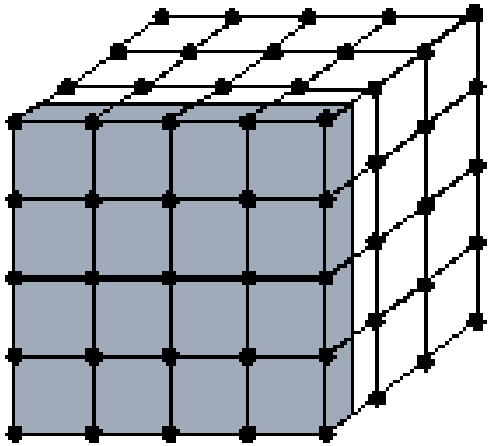
1.1 Способы разбиения

- Разбиение по данным
- Функциональная декомпозиция

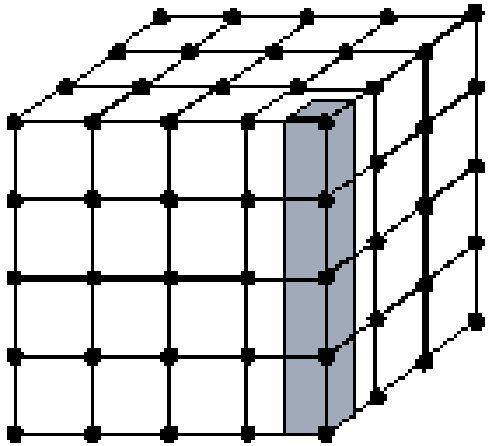
1.2 Разбиение по данным

- разбивают данные, сопоставленные с проблемой, на части приблизительно одинакового размера
- разделяют вычисления, которые должны быть выполнены на этих данных

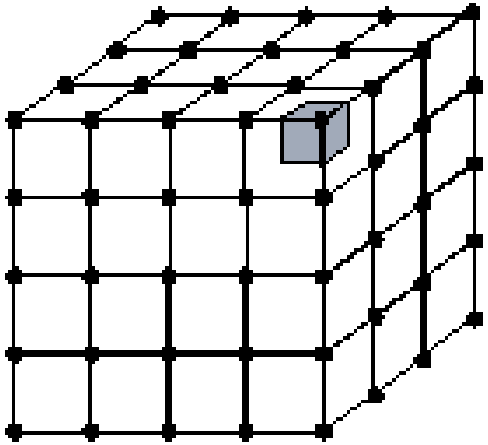
1.2 РАЗБИЕНИЕ ПО ДАННЫМ



1-D



2-D



3-D

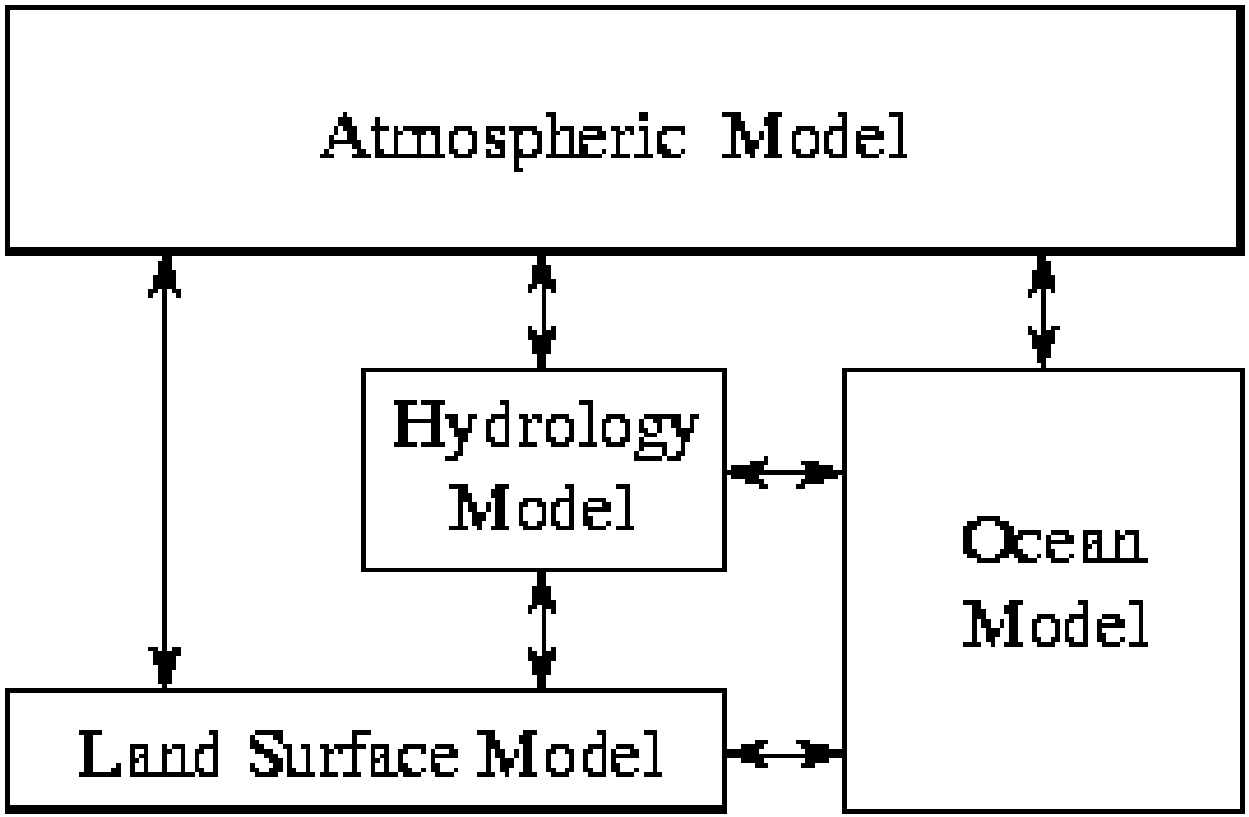
1.2 Разбиение по данным

является основой
для большинства
параллельных алгоритмов

1.3 Функциональная декомпозиция

- фокус направлен на вычисления которые должны быть выполнены, а не данные, которыми манипулируют
- аналог конвейрного параллелизма

1.3 Функциональная декомпозиция



1.3 Функциональная декомпозиция

- в основном используется в комбинации с разбиением по данным

2 ПРОЕКТИРОВАНИЕ СВЯЗЕЙ

Связь между двумя задачами можно рассматривать как канал, в который одна задача посылает сообщение и из которого другая задача принимает сообщение

2 КЛАССИФИКАЦИЯ СВЯЗЕЙ

- Локальные / глобальные
- Структурированные / неструктурированные
- Статические / динамические
- Синхронные / асинхронные

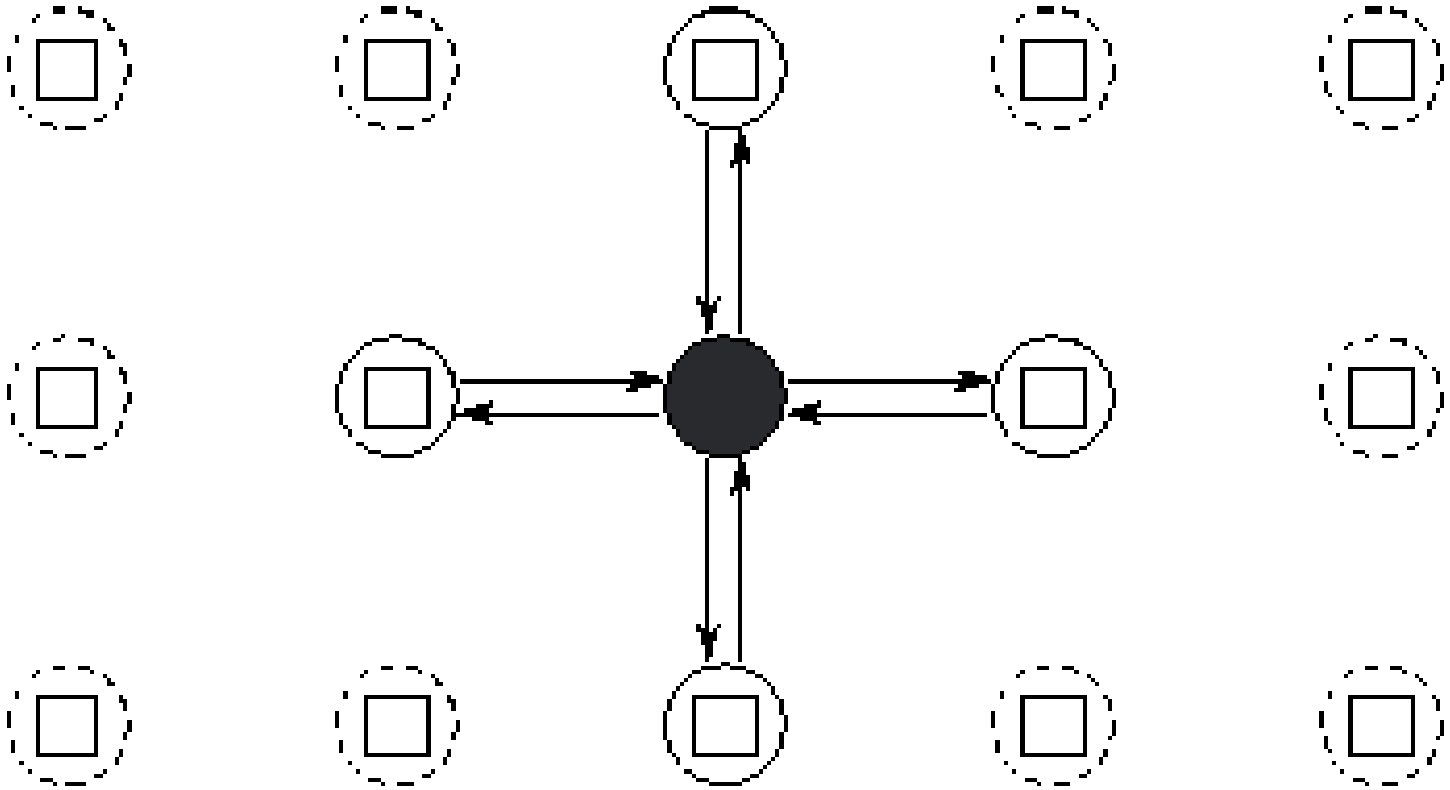
2.2 Локальные связи

Метод Якоби - конечные разности

$$X_{i,j}^{(t+1)} = \frac{4X_{i,j}^{(t)} + X_{i-1,j}^{(t)} + X_{i+1,j}^{(t)} + X_{i,j-1}^{(t)} + X_{i,j+1}^{(t)}}{8}$$

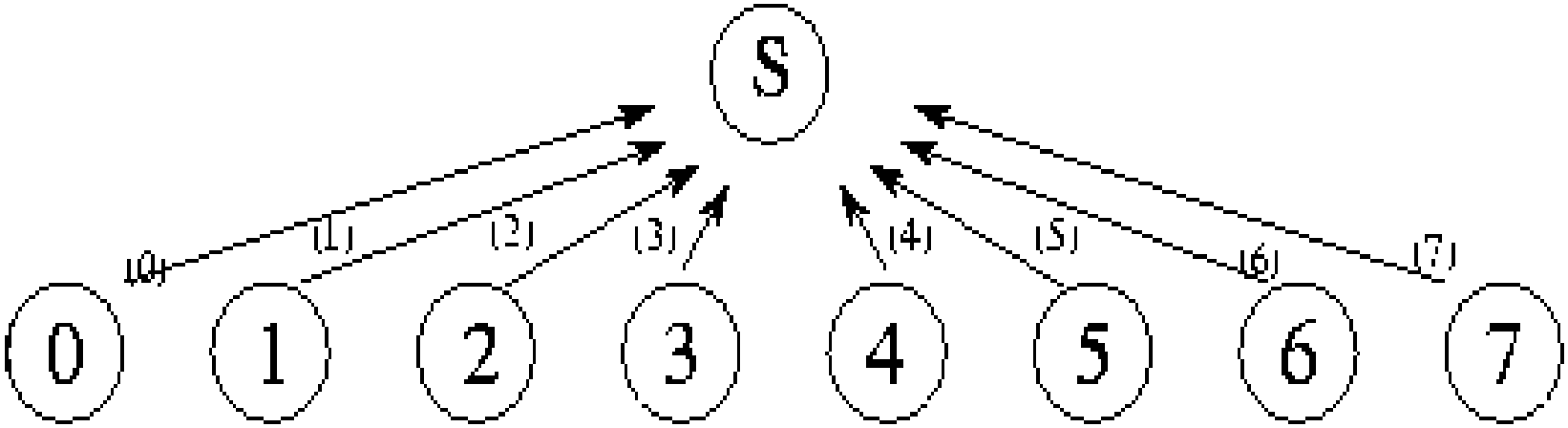
- пятиточечный шаблон
- 2-мерная сетка

2.2 Локальные связи



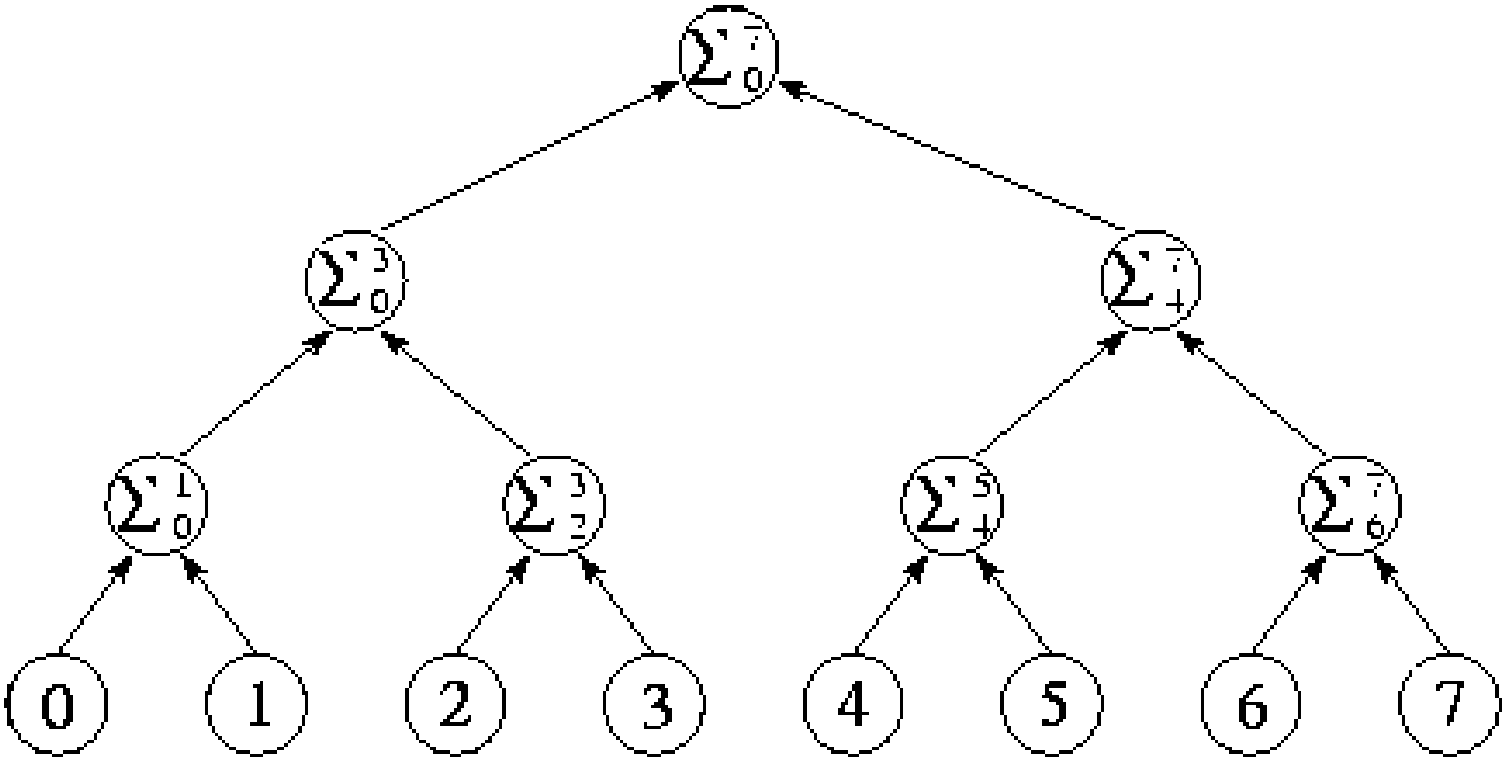
2.3 Глобальные связи

- Сумма N чисел



2.4 Сведение глобальных связей к локальным

- стратегия «разделяй и властвуй»



3 + 2 = АБСТРАКТНЫЙ АЛГОРИТМ

3 АГРЕГИРОВАНИЕ

- Оценка затрат на выполнение спроектированного алгоритма
- Группировка задач

3 АГРЕГИРОВАНИЕ

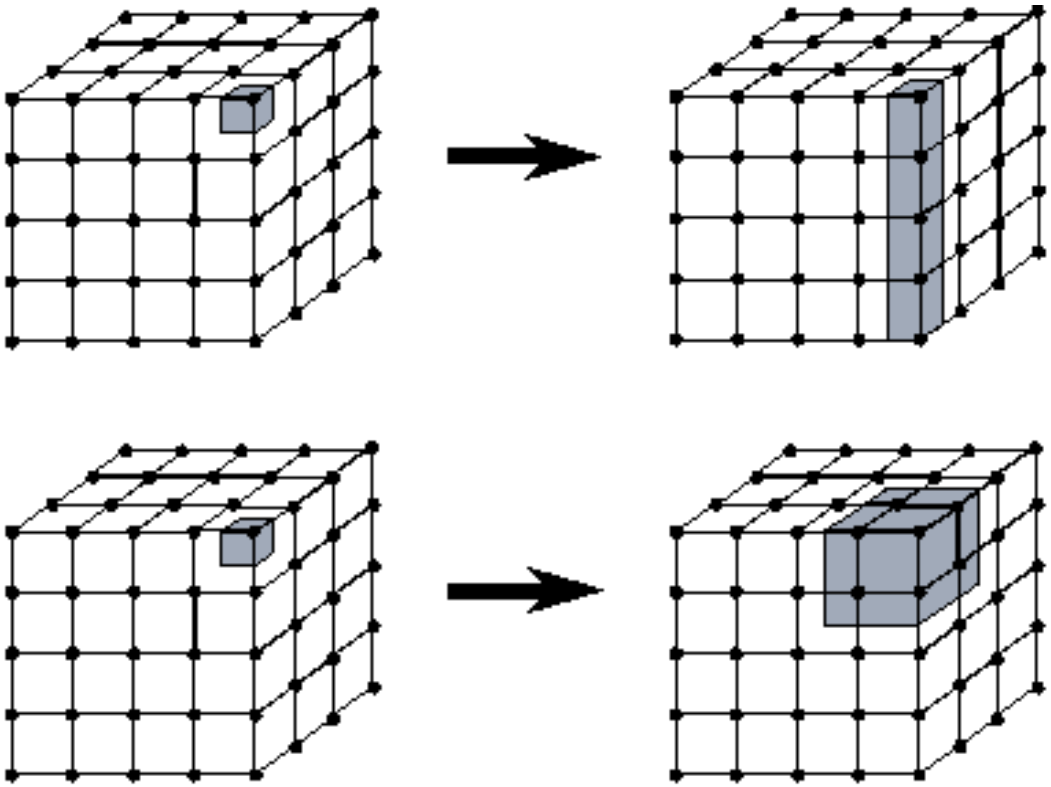
Цель:

- увеличение вычислительной сложности подзадач

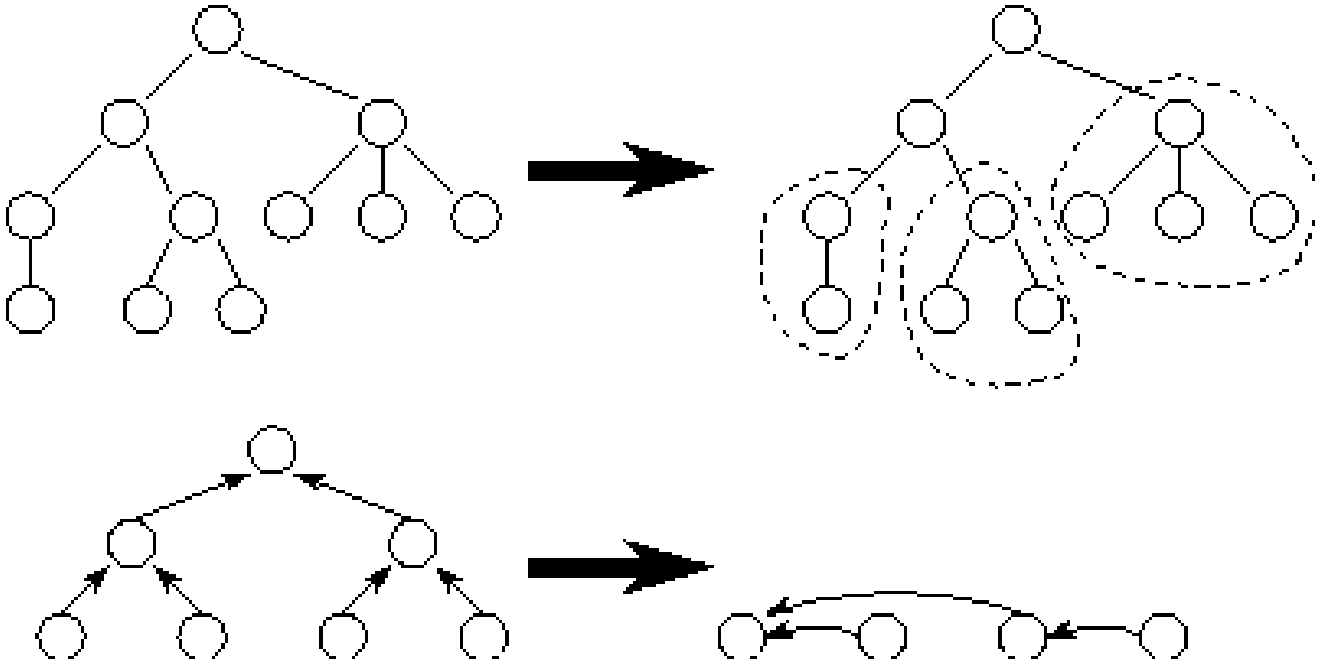
+

- уменьшение затрат на коммуникации

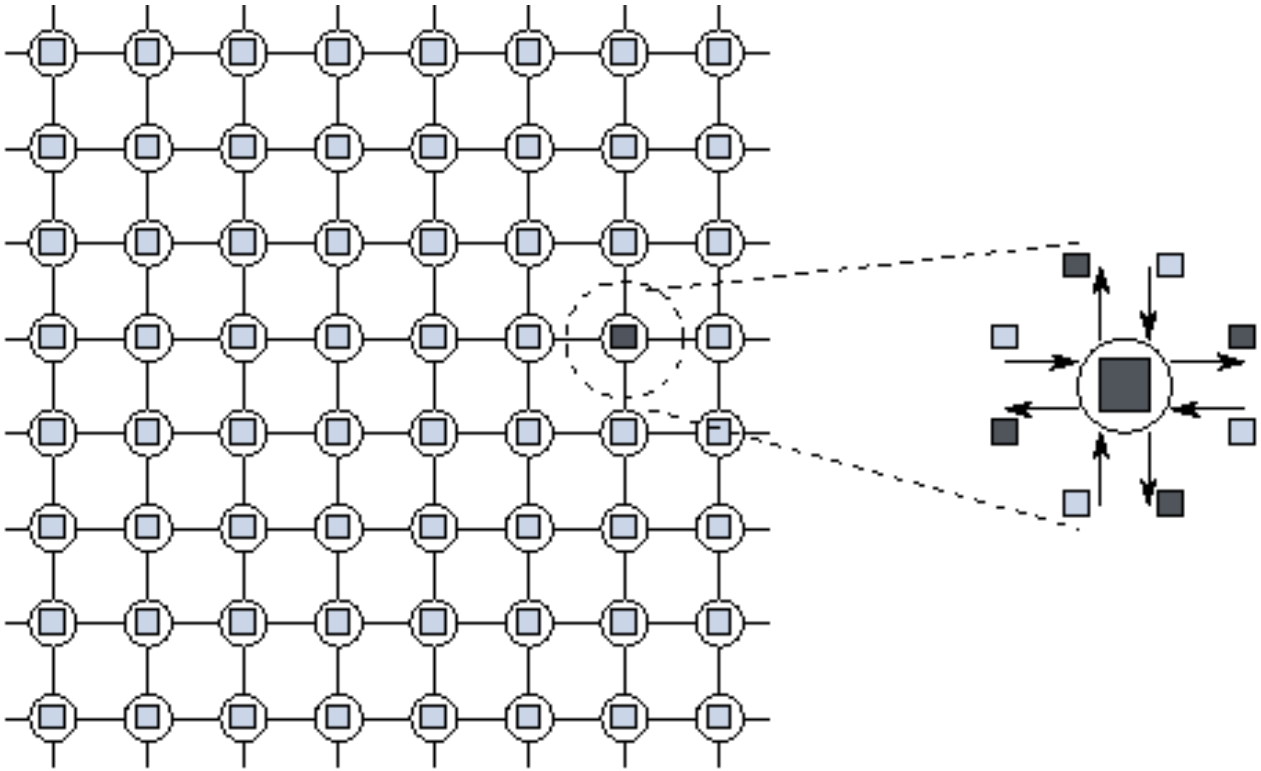
3 АГРЕГИРОВАНИЕ



3 АГРЕГИРОВАНИЕ

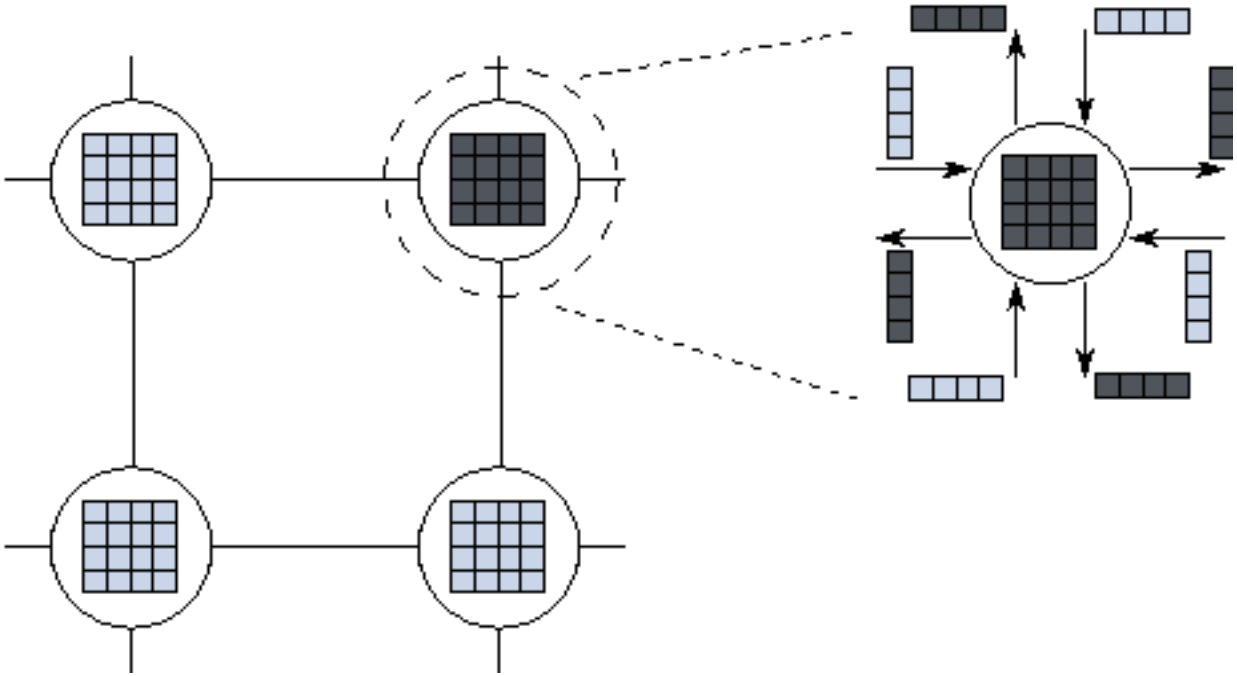


3 АГРЕГИРОВАНИЕ



64 задачи, 256 связей

3 АГРЕГИРОВАНИЕ



4 задачи, 16 связей

4 ПРИВЯЗКА

назначение задач на процессоры вычислительной системы

4 способы назначения

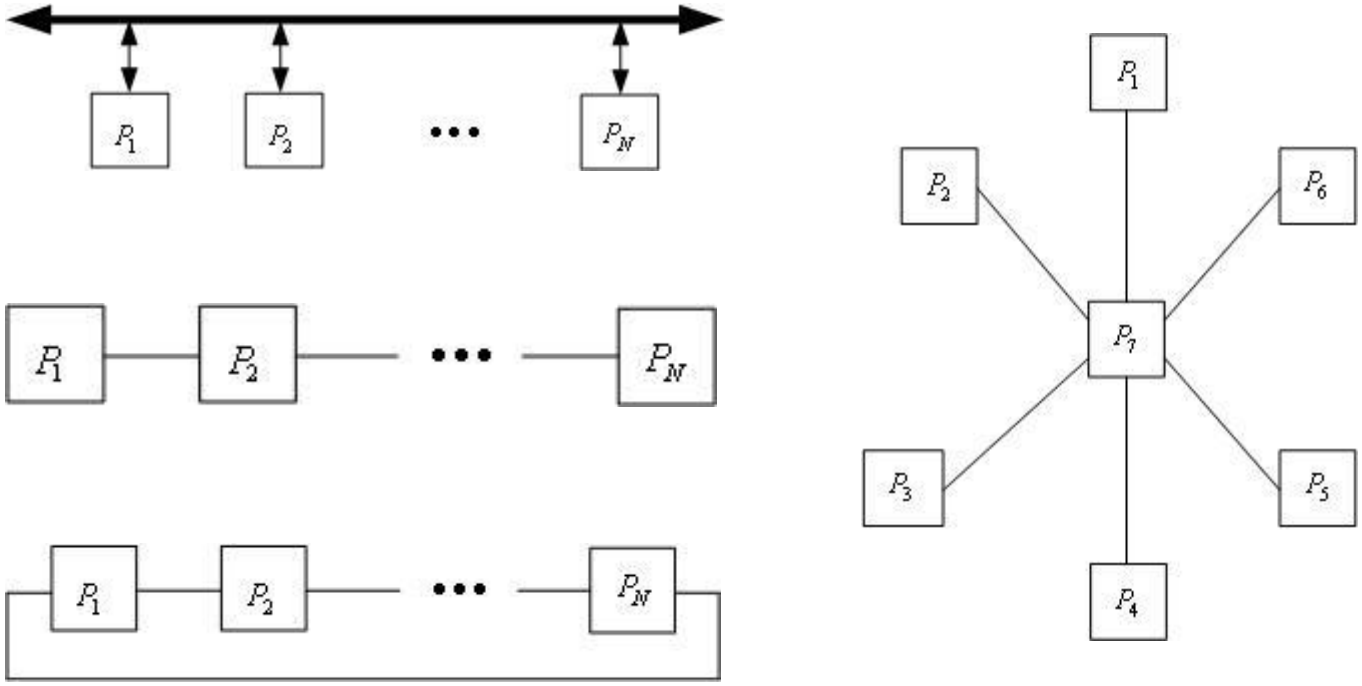
- индивидуальное => вручную, без масштабирования привязки
- программно, в соответствии с некоторым алгоритмом

4.2 Алгоритм привязки

- определяет отображение структуры множества задач на топологию вычислительной системы

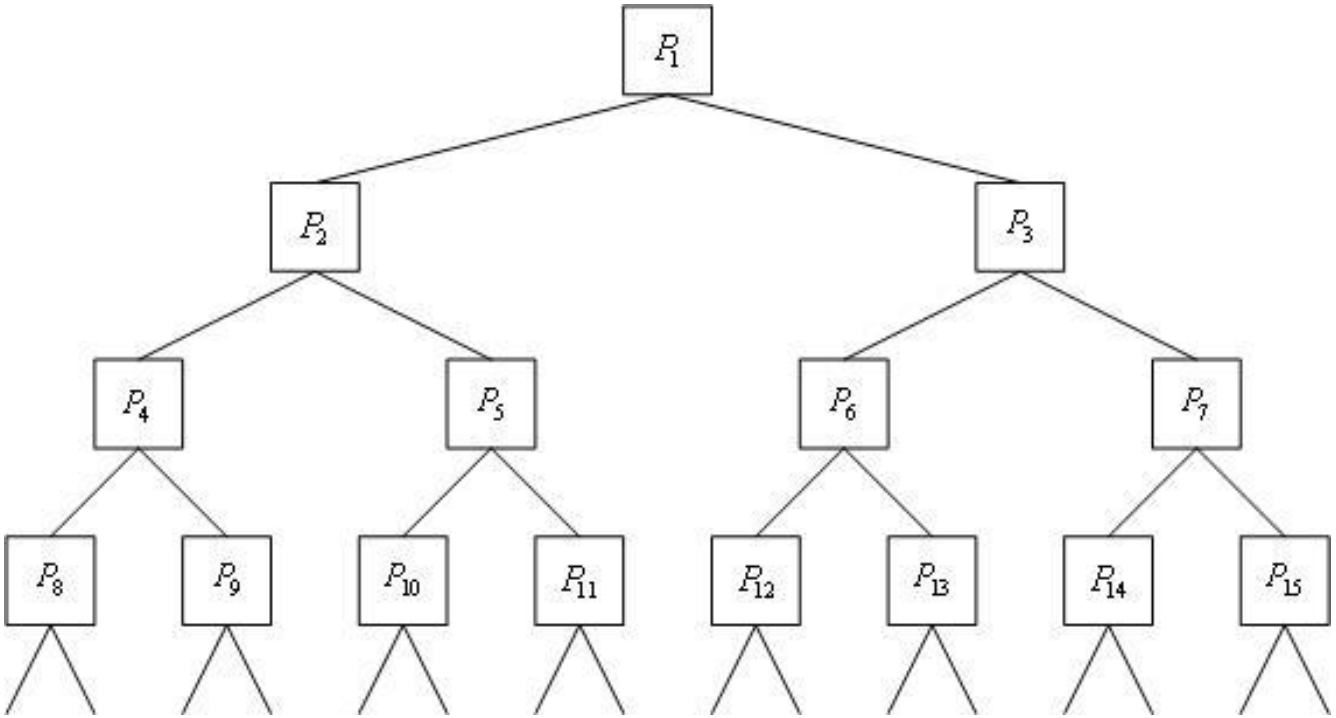
4.2 ТОПОЛОГИИ МНОГОПРОЦЕССОРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

регулярные сети с фиксированной топологией



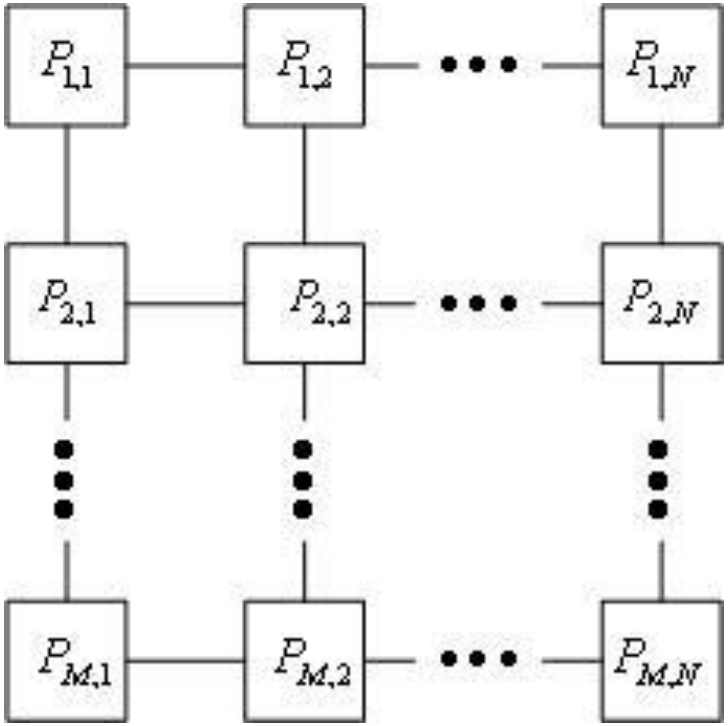
слабосвязанные топологии

4.2 ТОПОЛОГИИ МНОГОПРОЦЕССОРНЫХ ВС



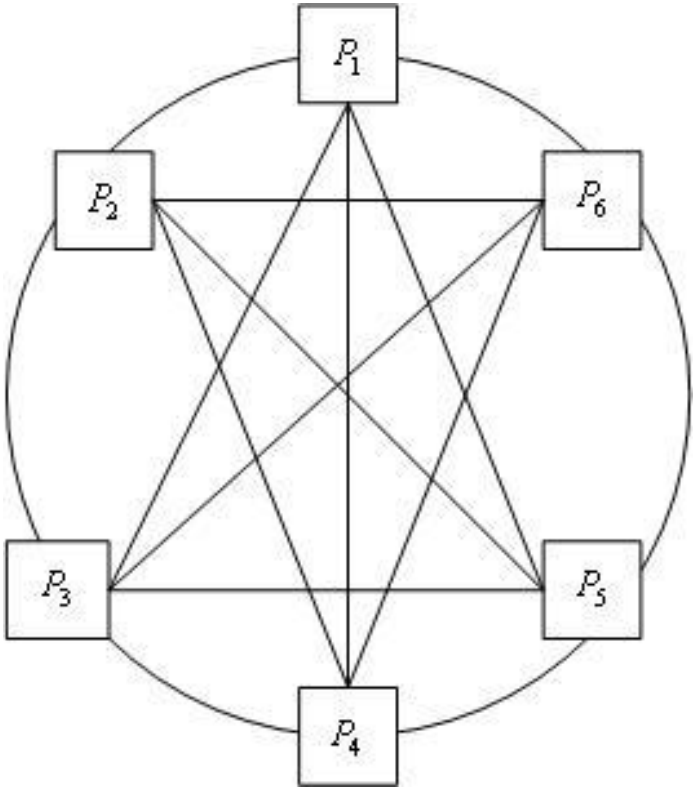
Бинарное дерево, d ≈ log N

4.2 ТОПОЛОГИИ МНОГОПРОЦЕССОРНЫХ ВС



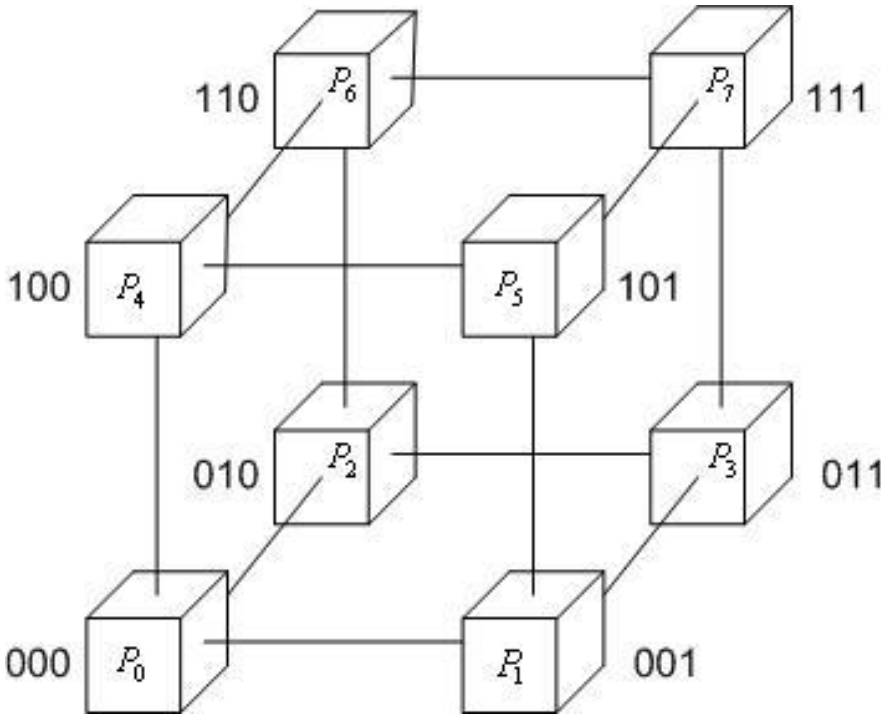
Двумерная решетка, d = $2\left(\sqrt{MN}-1\right)$

4.2 ТОПОЛОГИИ МНОГОПРОЦЕССОРНЫХ ВС



Клика, d = 1

4.2 ТОПОЛОГИИ МНОГОПРОЦЕССОРНЫХ ВС



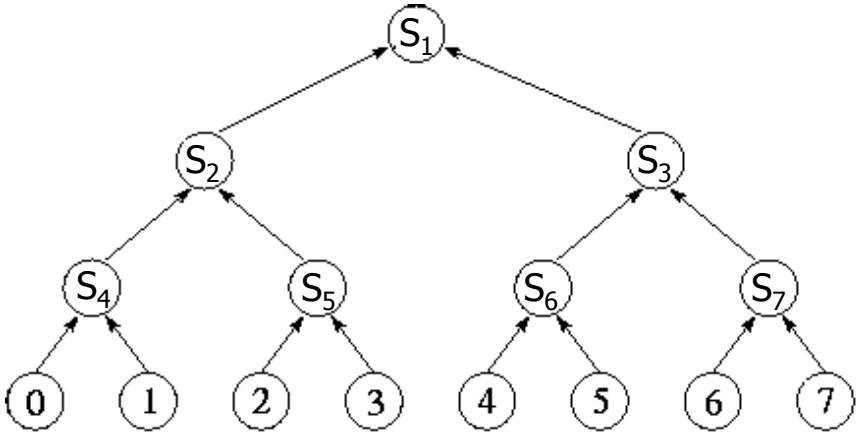
Трехмерный гиперкуб, d = $\sqrt[3]{N}$

4.3 Принципы назначения

- максимизировать использование процессоров
- минимизировать затраты на обмены

4.4 ПРИМЕРЫ НАЗНАЧЕНИЯ

Задача:



Архитектура:

4.4 Примеры назначения

- S_i -> P_i
- S_1 -> $P_{n/2}$, S_2 -> $P_{n/2-1}$, S_3 -> $P_{n/2+1}$, …

5 ПРИМЕР ПРОЕКТИРОВАНИЯ

- *A*-матрица , *x*, *b* – векторы
- вычисление произведения *b* =*Ax* =

$$\begin{pmatrix} a_1, x \\ a_2, x \\ \dots \\ a_m, x \end{pmatrix}$$

где a_i – *i* -я строка матрицы ,
(a_i, x) - скалярное произведение *i* -й строки матрицы на вектор

5 ПРИМЕР ПРОЕКТИРОВАНИЯ

- Разбиение: *m* задач $S_i = (a_i, x)$
- Установление связей: глобальные рассылка (по 1 строке и вектор – всем) и сбор результатов
- Агрегирование: *k* задач, $S_j = \begin{pmatrix} a_{j_1}, x \\ \dots \\ a_{j_s}, x \end{pmatrix}$
- Привязка : S_i -> P_i

5 ПРИМЕР ПРОЕКТИРОВАНИЯ