

# **Лекция 15**

## **Параллельное программирование**

**Ефимов Александр Владимирович**  
**E-mail: [alexandr.v.efimov@sibguti.ru](mailto:alexandr.v.efimov@sibguti.ru)**

**Курс «Архитектура вычислительных систем»**  
**СибГУТИ, 2018**

# Понятия параллельного программирования

## ➤ Параллельный алгоритм

Описание процесса обработки информации, ориентированное на реализацию в коллективе вычислителей

## ➤ Параллельная программа

Запись параллельного алгоритма на языке программирования, доступном коллективу вычислителей

## ➤ Распараллеливание

Процесс «приспособления» методов к реализации на коллективе вычислителей или процесс «расщепления» последовательных алгоритмов решения сложных задач

## ➤ Параллельное программирование

Теоретическая и практическая деятельность по созданию параллельных алгоритмов и программ обработки информации

# Методика распараллеливания сложных задач

## ✓ Локальное распараллеливание

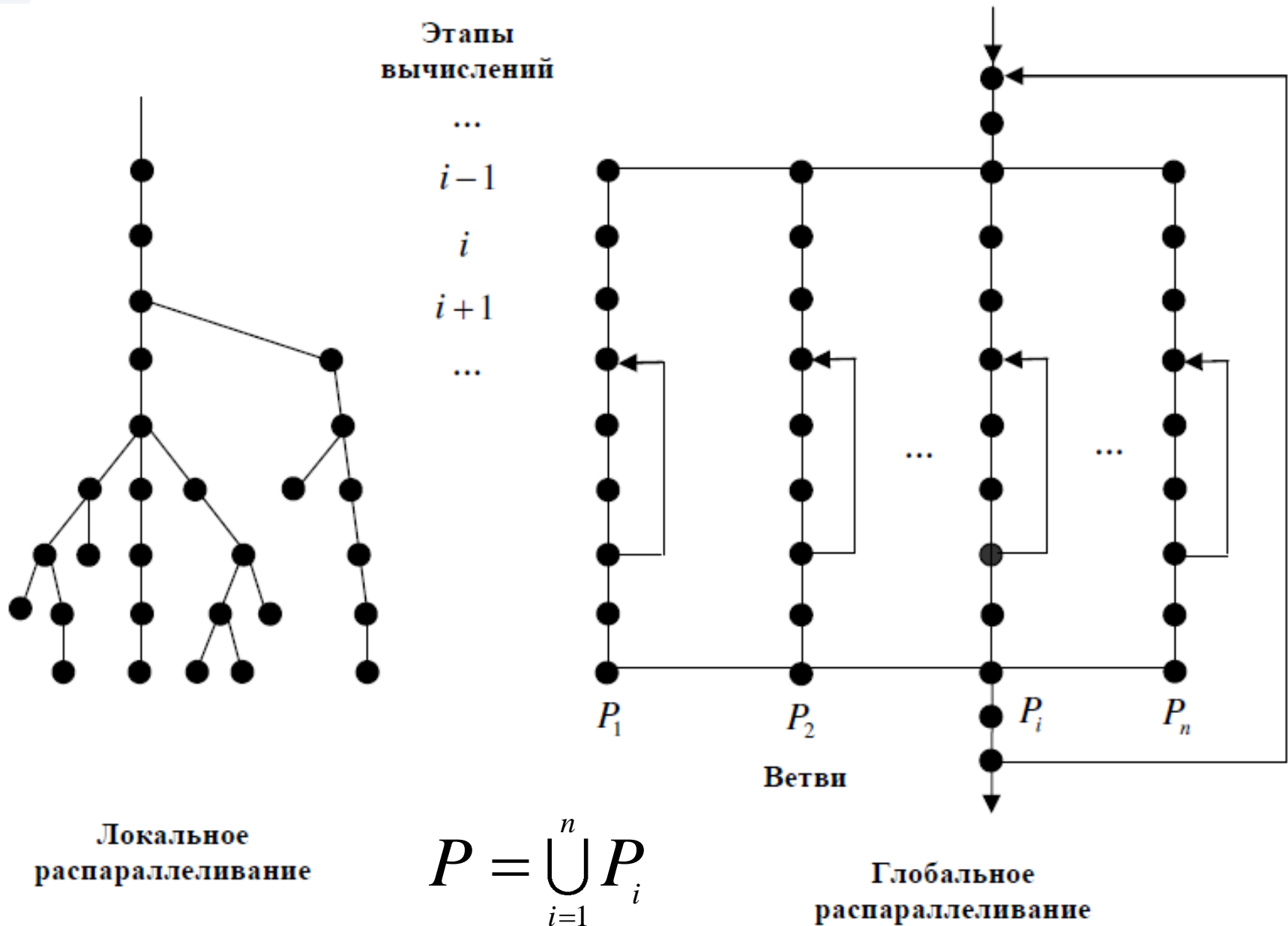
Расщепление алгоритма решения сложной задачи на предельно простые блоки (операции или операторы) и выделение для каждого этапа вычислений максимально возможного количества одновременно выполняемых блоков

## ✓ Глобальное (крупноблочное) распараллеливание

Разбиение сложной задачи на крупные блоки-подзадачи, между которыми существует слабая связность

Такие подзадачи называют *ветвями параллельного алгоритма*, а соответствующие им программы – *ветвями параллельной программы*

# Схемы параллельных алгоритмов



# Параллельный алгоритм умножения матриц

$$A[1:N, 1:K] \times B[1:K, 1:M] = C[1:N, 1:M]$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1h} & \cdots & a_{1K} \\ a_{21} & a_{22} & \cdots & a_{2h} & \cdots & a_{2K} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i1} & a_{i2} & \cdots & a_{ih} & \cdots & a_{iK} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{N1} & a_{N2} & \cdots & a_{Nh} & \cdots & a_{NK} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1j} & \cdots & b_{1M} \\ b_{21} & b_{22} & \cdots & b_{2j} & \cdots & b_{2M} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{h1} & b_{h2} & \cdots & b_{hj} & \cdots & b_{hM} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{K1} & b_{K2} & \cdots & b_{Kj} & \cdots & b_{KM} \end{pmatrix} =$$

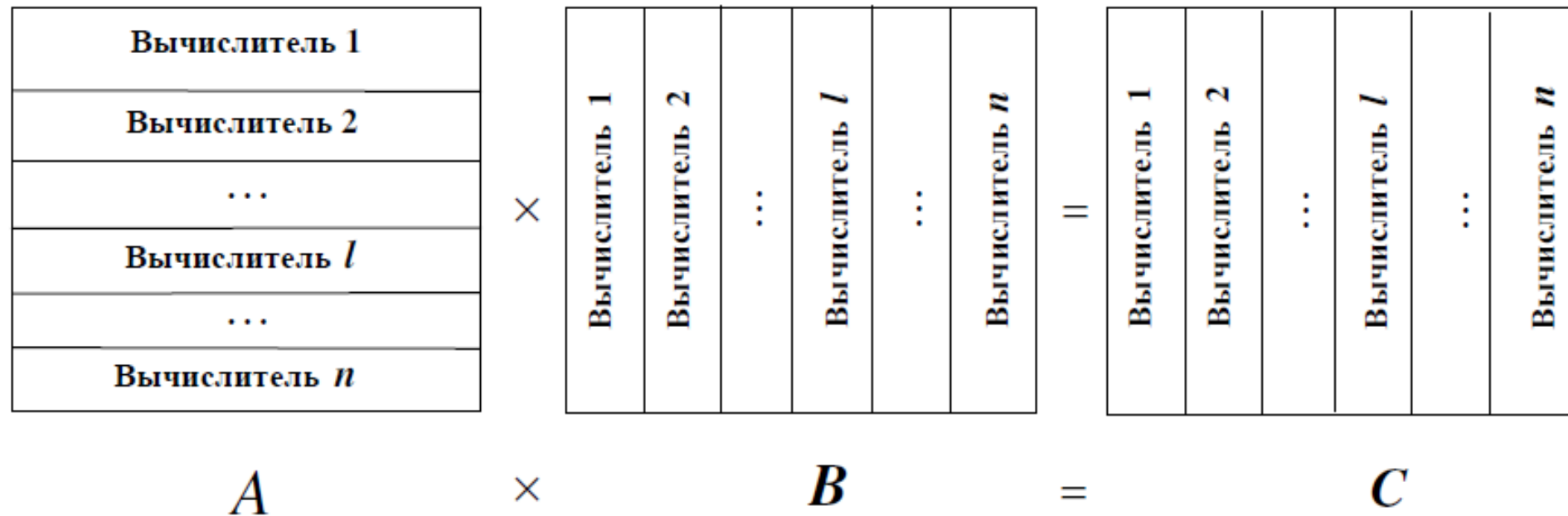
$$= \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1j} & \cdots & c_{1M} \\ c_{21} & c_{22} & \cdots & c_{2j} & \cdots & c_{2M} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{i1} & c_{i2} & \cdots & c_{ij} & \cdots & c_{iM} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{N1} & c_{N2} & \cdots & c_{Nj} & \cdots & c_{NM} \end{pmatrix},$$

$$c_{ij} = \sum_{h=1}^K a_{ih} b_{hj}, \quad i = \overline{1, N}, \quad j = \overline{1, M}.$$

# Последовательный алгоритм умножения матриц

```
1. procedure MAT_MULT (A, B, C)
2. begin
3.   for i:= 1 to N do
4.     for j:= 1 to M do
5.       begin
6.         C[i,j] := 0;
7.         for h:= 1 to K do
8.           C[i,j] := C[i,j] + A[i,h] * B[h,j];
9.         endfor;
10. end MAT_MULT
```

# Параллельный алгоритм умножения матриц



# Параллельный алгоритм умножения матриц

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{pmatrix} \quad \begin{array}{l} N \times K = 4 \times 4 \\ \text{Вычислитель} \\ 1 \\ \text{Вычислитель} \\ 2 \end{array}$$

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix} \quad \begin{array}{l} K \times M = 4 \times 4 \\ \text{Выч.} \\ 1 \end{array} \quad \begin{array}{l} \text{Выч.} \\ 2 \end{array}$$

Количество вычислителей  $n = 2$



# Параллельный алгоритм умножения матриц

## Вычислитель 1:

$$\begin{array}{cc} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} + a_{24}b_{41} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42} \end{array}$$



$$\begin{array}{cc} c_{11} & c_{12} \\ c_{21} & c_{22} \end{array}$$

# Параллельный алгоритм умножения матриц

## Вычислитель 2:

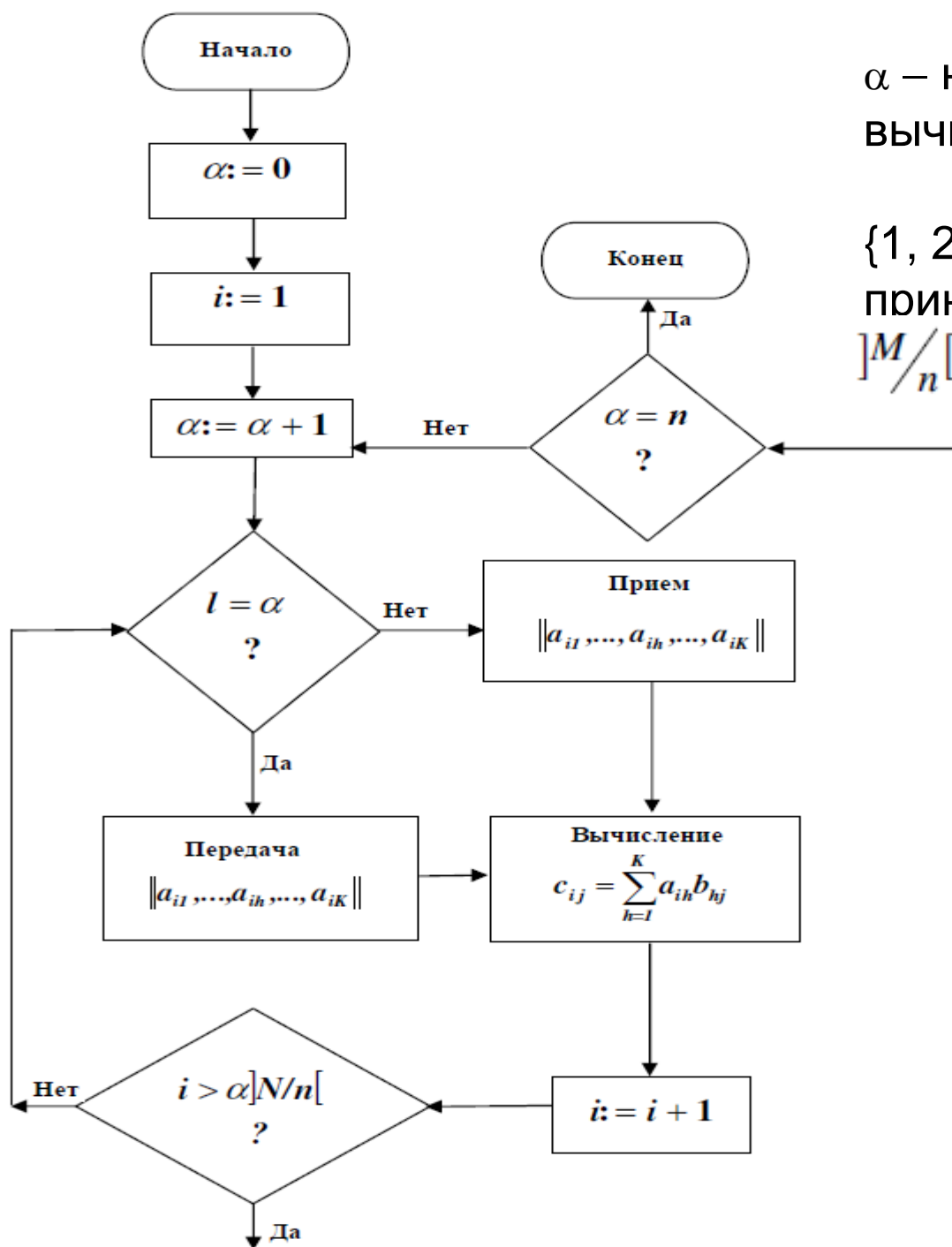
$$\begin{array}{ll} a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} + a_{34}b_{43} & a_{31}b_{14} + a_{32}b_{24} + a_{33}b_{34} + a_{34}b_{44} \\ a_{41}b_{13} + a_{42}b_{23} + a_{43}b_{33} + a_{44}b_{43} & a_{41}b_{14} + a_{42}b_{24} + a_{43}b_{34} + a_{44}b_{44} \end{array}$$



$$\begin{array}{ll} c_{33} & c_{34} \\ c_{43} & c_{44} \end{array}$$

# Параллельный алгоритм умножения матриц

$$C = \begin{pmatrix} \begin{matrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{matrix} & \begin{matrix} ? \\ ? \end{matrix} \\ \begin{matrix} ? \\ ? \end{matrix} & \begin{matrix} c_{33} & c_{34} \\ c_{43} & c_{44} \end{matrix} \end{pmatrix}$$



$\alpha$  – номер передающего  
вычислителя

$\{1, 2, \dots, \alpha-1, \alpha+1, \dots, n\}$  – номера  
принимающих вычислителей

$$\left\lceil \frac{M}{n} \right\rceil \cdot (l-1) < j \leq \left\lceil \frac{M}{n} \right\rceil \cdot l$$

# Показатели эффективности параллельных алгоритмов

## ✓ Коэффициент накладных расходов

$$\varepsilon = t / T,$$

$t$  – время, расходуемое вычислительной системой на организацию и реализацию всех обменов информацией между ветвями,

$T$  – время, требующееся на выполнение арифметических и логических операций при реализации алгоритма

## ✓ Коэффициент ускорения

$$\chi = \tau_1 / \tau_n,$$

$\tau_1$  – время выполнения алгоритма (решения задачи) на одном вычислителе,

$\tau_n$  – время выполнения соответствующего  $P$  -алгоритма в системе из  $n$ -вычислений

# Показатели эффективности параллельных алгоритмов

- ✓ *Коэффициент ускорения параллельного алгоритма по сравнению с наилучшим последовательным алгоритмом*

$$\chi' = \tau'_1 / \tau_n,$$

$\tau'_1$  – время реализации самого быстрого последовательного алгоритма на одном вычислителе.

- ✓ *Закон Амдала*

$$\chi^* \leq \frac{\kappa}{\delta + (1 - \delta) / n},$$

$n$  – количество вычислителей, образующих ВС;

$\delta$  – относительная доля операций Р-программы, выполняемых последовательно;

$\kappa$  – корректирующий коэффициент,  $0 \leq \kappa \leq 1$ .

# Показатели эффективности параллельных алгоритмов

- ✓ *Коэффициент эффективности P-алгоритма*

$$E = \chi / n$$

- ✓ *Коэффициент эффективности P-алгоритма по отношению к наилучшему последовательному алгоритму*

$$E' = \chi' / n$$

# Показатель эффективности для алгоритма умножения матриц

$$\varepsilon = \frac{t_n}{\rho(t_y + t_c)} = \frac{e}{\rho}, \quad e = \frac{t_n}{t_y + t_c}$$

$t_n$  – время пересылки одного слова (элемента матрицы);

$t_y$  и  $t_c$  – время выполнения операций умножения и сложения соответственно

$$\rho = \left] \frac{M}{n} \right[$$

$$\varepsilon \rightarrow 0 \text{ при } \rho \rightarrow \infty$$



# Пример

Определить максимум коэффициента накладных расходов при реализации  $p$ -алгоритма на ВС, имеющей следующие параметры:

- разрядность  $l = 32$ ;
- полосу пропускания канала между машинами  $\nu = 1$  Гигабод;
- время выполнения операции сложения  $0,1$  мкс;
- время выполнения операции умножения  $10$  нс.

$$e = \frac{t_n}{t_c + t_y} = \frac{32 \cdot 10^{-9}}{0,1 \cdot 10^{-6} + 10 \cdot 10^{-9}} = \frac{0,000000032}{0,0000001 + 0,00000001} = 0,29$$

1 бод = 1 бит/с  
1 Гигабод =  $10^9$  бод

# Факторы, ограничивающие ускорение

- Время, расходуемое на синхронизацию параллельных ветвей и на обмены информацией между ними
- Несбалансированность (неоднородность) нагрузки вычислителей и (или) невозможность построения Р-алгоритма с числом ветвей, равным числу вычислителей в ВС

# Парадокс параллелизма

- Достижение ускорения и эффективности параллельного алгоритма, превышающих значения  $n$  и 1, соответственно

$$\chi > n, \quad E > 1$$

«Парадокс параллелизма» - это более чем линейный рост производительности параллельной ВС с увеличением числа  $n$  её вычислителей

- Факторы, объясняющие этот парадокс
  - эффект памяти
  - возможность применения (априори параллельных) методов решения задач, реализация которых затруднительна на последовательной ЭВМ (например, снова из-за оперативной памяти, точнее ее ограниченного объема).

# Понятие о сложных задачах

$$\varepsilon(V, n) = t(V, n) / T(V, n)$$

$V$  – количество операций, которые необходимо выполнить при решении задачи на ВС;

$n$  – число параллельных ветвей или число вычислителей, на которых решается задача,  $n \geq 2$ ;

$t(V, n)$  – время, затрачиваемое на синхронизацию параллельных ветвей алгоритма, на настройку (программирование структуры) системы, на реализацию обменов информацией между ветвями (вычислителями);

$T(V, n)$  – время, расходуемое системой на вычисления.

$$\varepsilon(V, n) \rightarrow 0 \text{ при } V \rightarrow \infty$$

$$V \geq n \cdot 10^k,$$

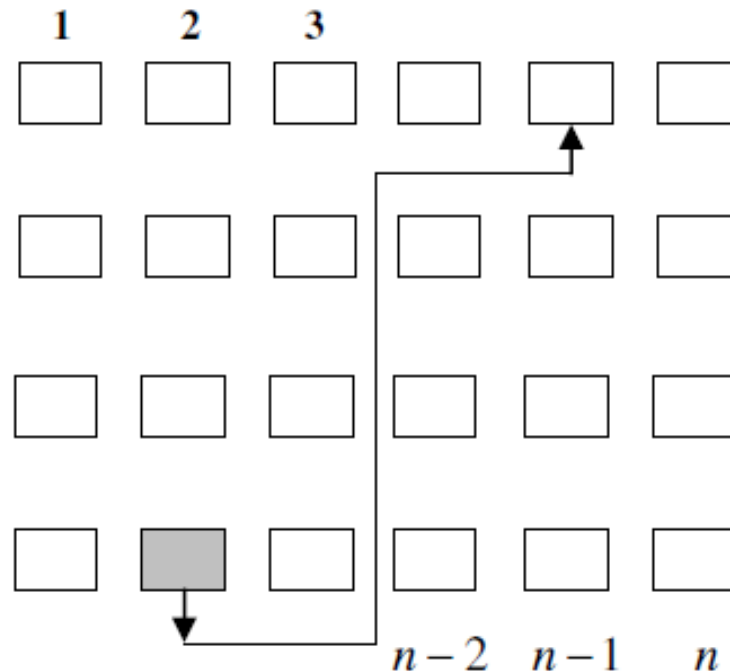
$k$  – эмпирический коэффициент,  $k \geq 1$ .

$k \rightarrow 1$  при  $V \rightarrow V^*$ , где  $1/V^*$  – время обращения к локальной памяти в вычислителе

# Схемы обмена информацией между ветвями параллельных алгоритмов

## ➤ Дифференцированный обмен (ДО, Point-to-Point)

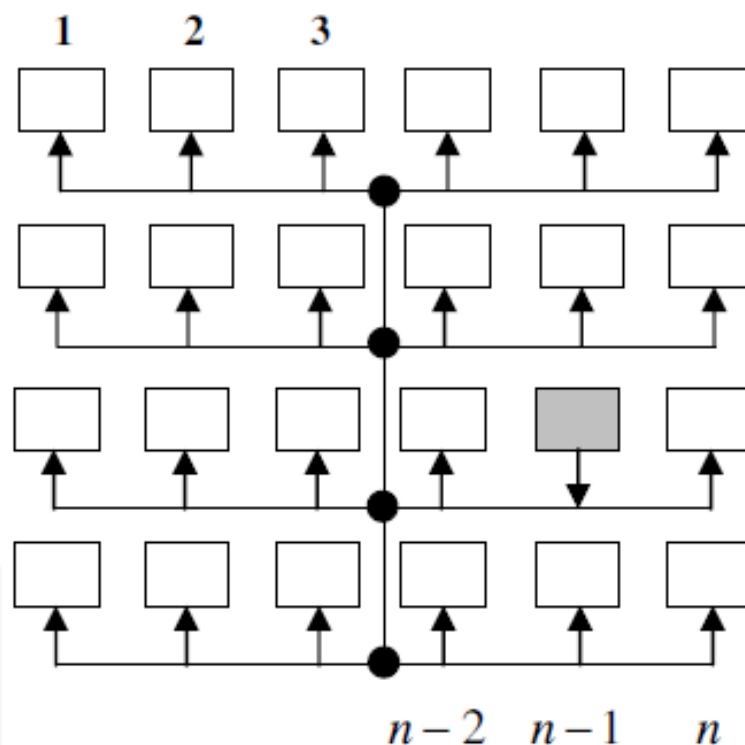
Передача информации из одной ветви в любую другую ветвь или из одного вычислителя (передатчика) к другому вычислителю (приемнику)



# Схемы обмена информацией между ветвями параллельных алгоритмов

## ➤ Трансляционный обмен (ТО, One-to-All Broadcast)

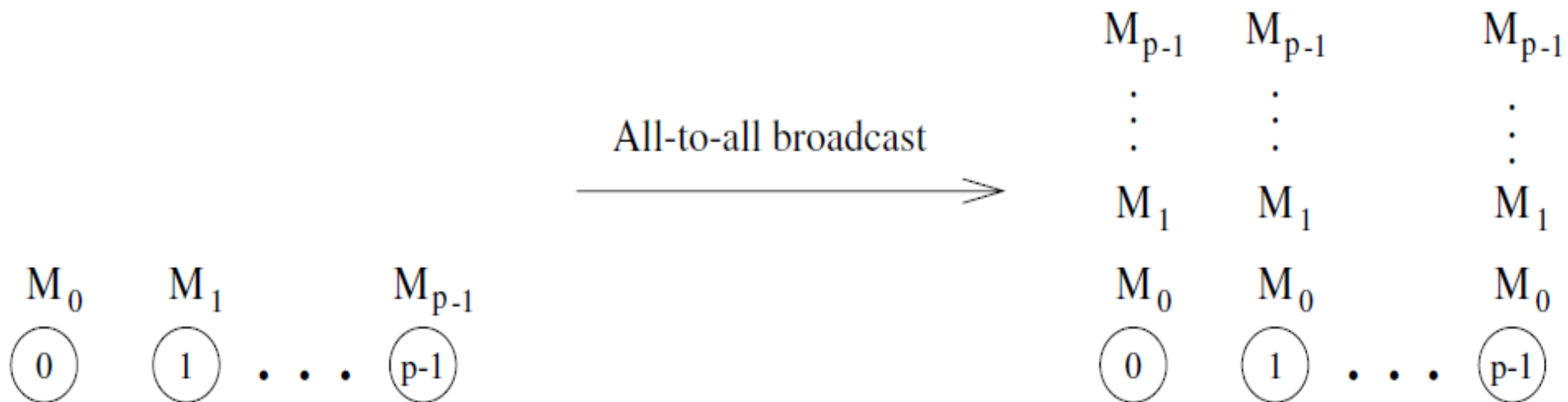
Передача одной и той же информации из одной (любой) ветви одновременно во все остальные ветви параллельного алгоритма



# Схемы обмена информацией между ветвями параллельных алгоритмов

- Трансляционно-циклический обмен (ТЦО, All-to-all Broadcast)

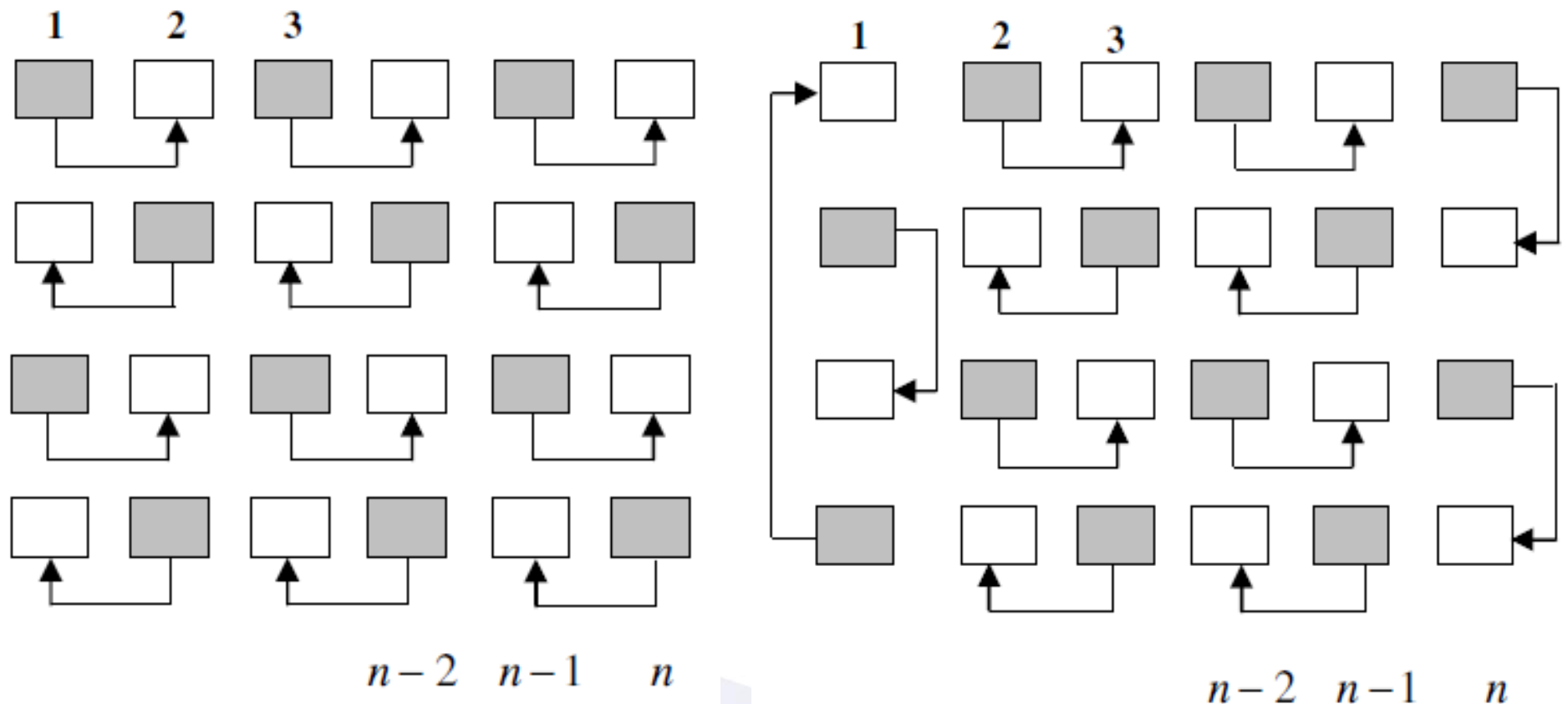
Трансляция информации из каждой ветви во все



# Схемы обмена информацией между ветвями параллельных алгоритмов

## ➤ Конвейерно-параллельный обмен (КПО)

Передача информации между соседними ветвями

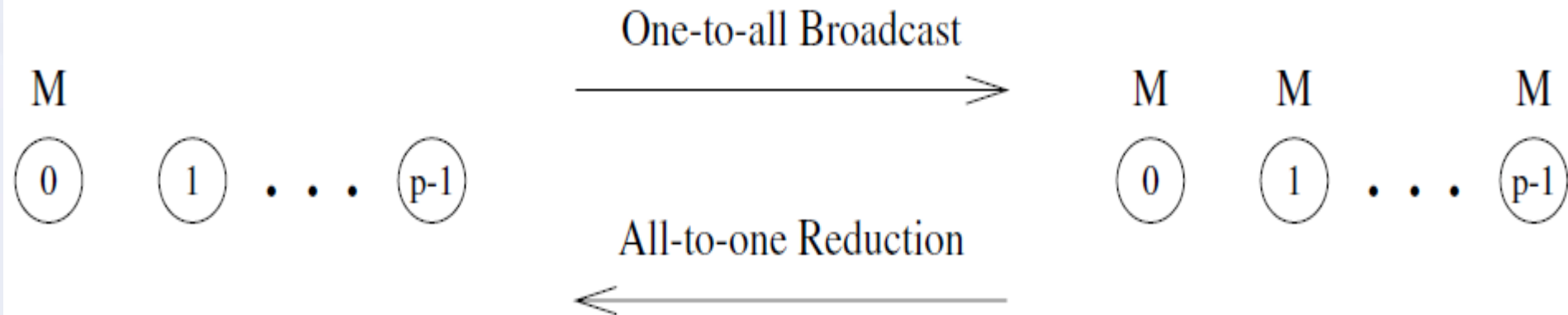




# Схемы обмена информацией между ветвями параллельных алгоритмов

## ➤ Коллекторный обмен (КО)

Инвертированный трансляционный обмен



# Схемы обмена информацией между ветвями параллельных алгоритмов

| Тип обмена            | ДО  | ТО  | ТЦО | КПО | КО |
|-----------------------|-----|-----|-----|-----|----|
| Частота использования | 2 % | 17% | 40% | 34% | 7% |

**Групповые схемы обмена информацией между ветвями параллельных алгоритмов или программ составляют более 90% от общего числа обменов**

# Опыт применения методики крупноблочного распараллеливания сложных задач

1. Сложные задачи допускают построение параллельных алгоритмов, состоящих из идентичных ветвей.
2. При распределении исходных данных между ветвями параллельного алгоритма эффективным является принцип однородного расчленения массивов информации.
3. Для построения параллельных алгоритмов достаточно использовать пять схем обмена информацией между ветвями: дифференцированную, трансляционную, трансляционно-циклическую, конвейерно-параллельную и коллекторную схемы.
4. Простота организации обменов по рассмотренным схемам позволяет использовать в основном регулярные (однородные) структуры вычислительных систем.

# Опыт применения методики крупноблочного распараллеливания сложных задач

5. Для записи параллельных алгоритмов решения сложных задач эффективны версии распространенных языков высокого уровня
6. Простота схем обмена и распределения данных по машинам ведет к простоте записи и реализации параллельных программ
7. Трансляционная, трансляционно-циклическая и конвейерно-параллельные схемы обмена информацией обеспечивают одновременную работу вычислителей системы и составляют не менее 90% всех схем, реализуемых в процессе выполнения параллельных программ сложных задач

# Литература

Хорошевский В.Г. Архитектура вычислительных систем. Учебное пособие. – М.: МГТУ им. Н.Э. Баумана, 2005; 2-е издание, 2008.

Хорошевский В.Г. Инженерные анализ функционирования вычислительных машин и систем. – М.: “Радио и связь”, 1987.