

Интернет Университет Суперкомпьютерных технологий

Учебный курс

Введение в параллельные алгоритмы

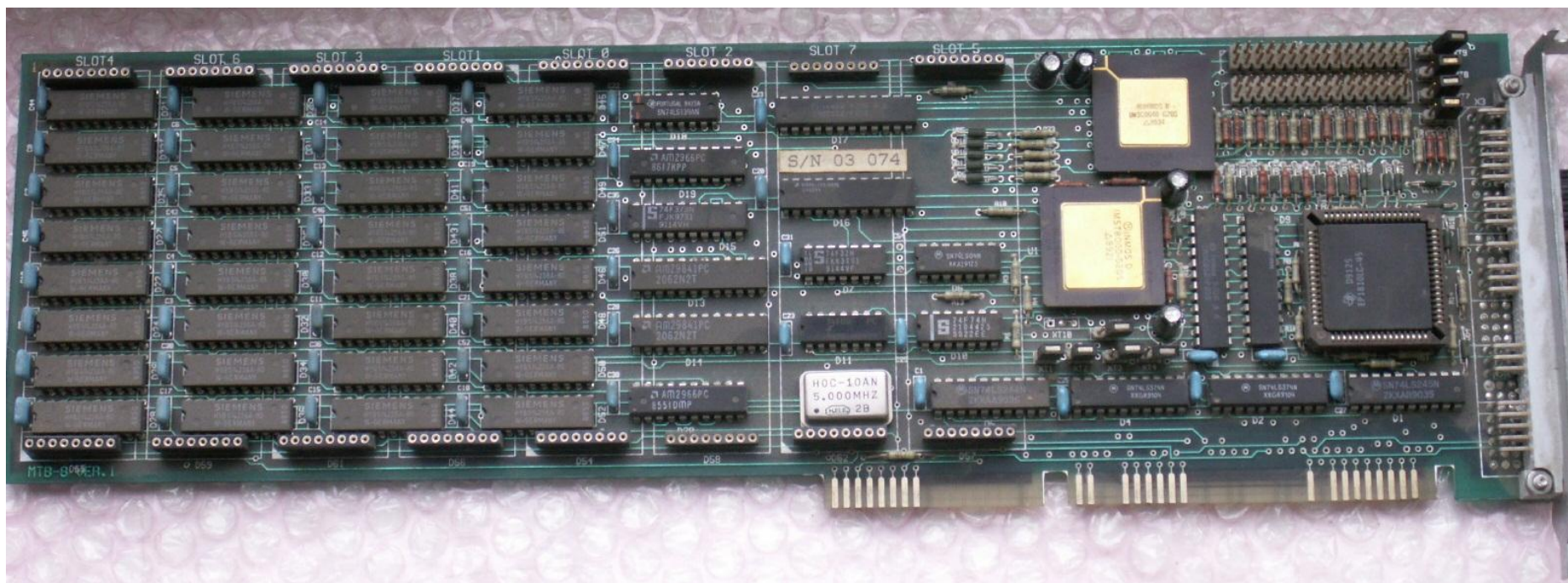
Лекция 1 Основные понятия

Якобовский М.В., проф., д.ф.-м.н.
Институт прикладной
математики им. М.В.Келдыша
РАН, Москва..

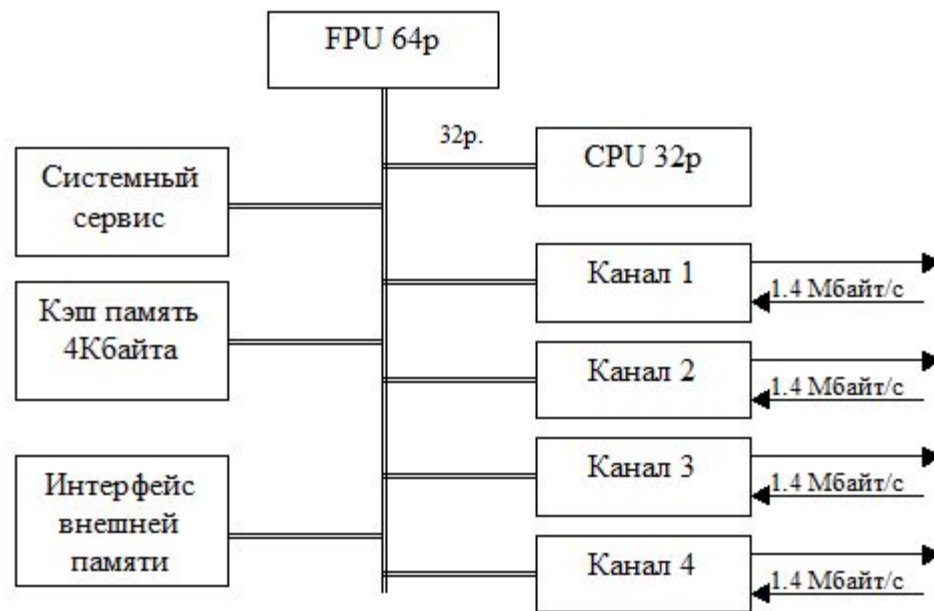
Содержание лекции

- ❑ Многопроцессорные системы
 - с распределенной памятью
 - с общей памятью
 - Гибридные
- ❑ Модель выполнения программ
- ❑ Методы взаимодействия процессов
 - Методы передачи данных
 - Семафоры
- ❑ Ускорение и эффективность параллельных алгоритмов
- ❑ Пример алгоритма низкой эффективности

Транспьютерная материнская плата МТБ-8

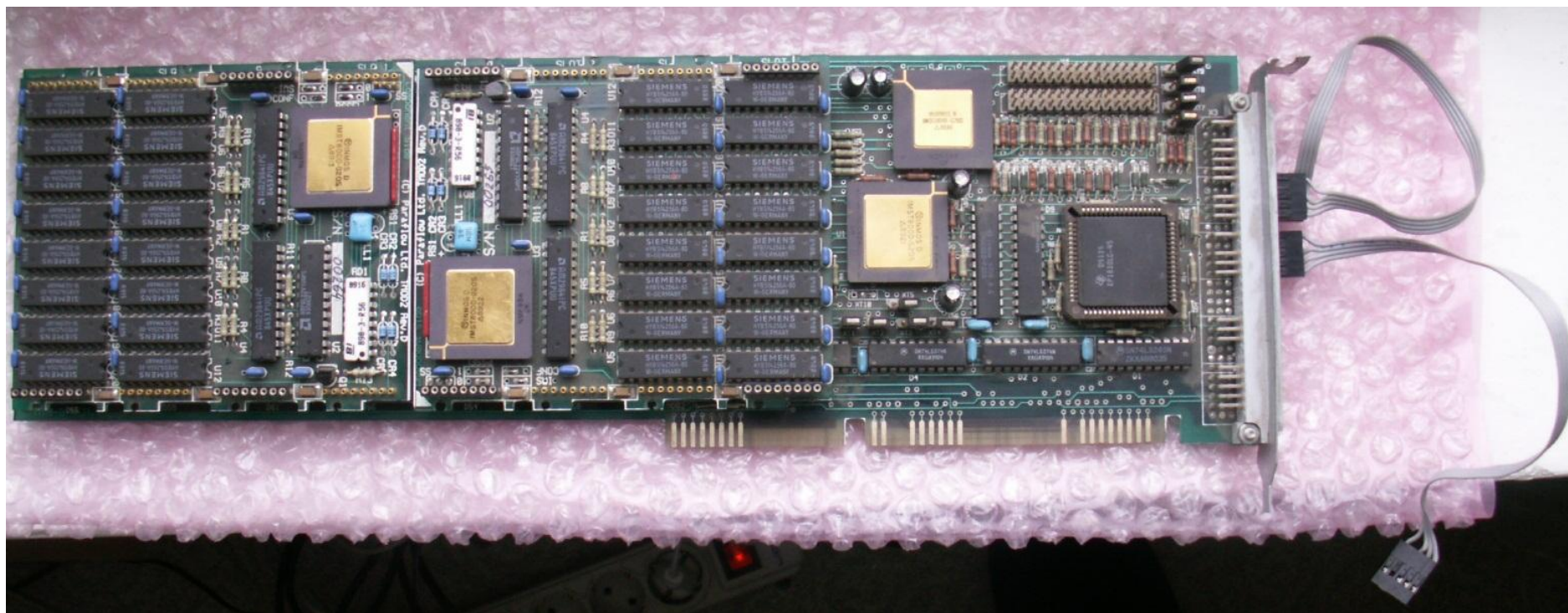


Транспьютер Т-800

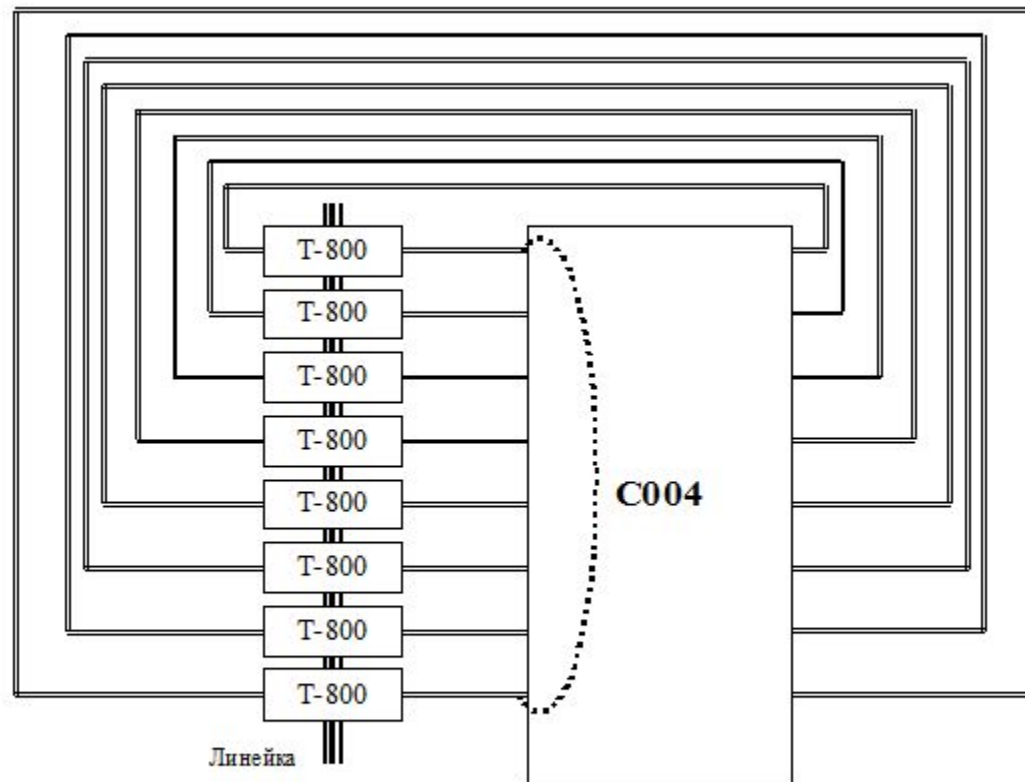


Структура транспьютера Т-800

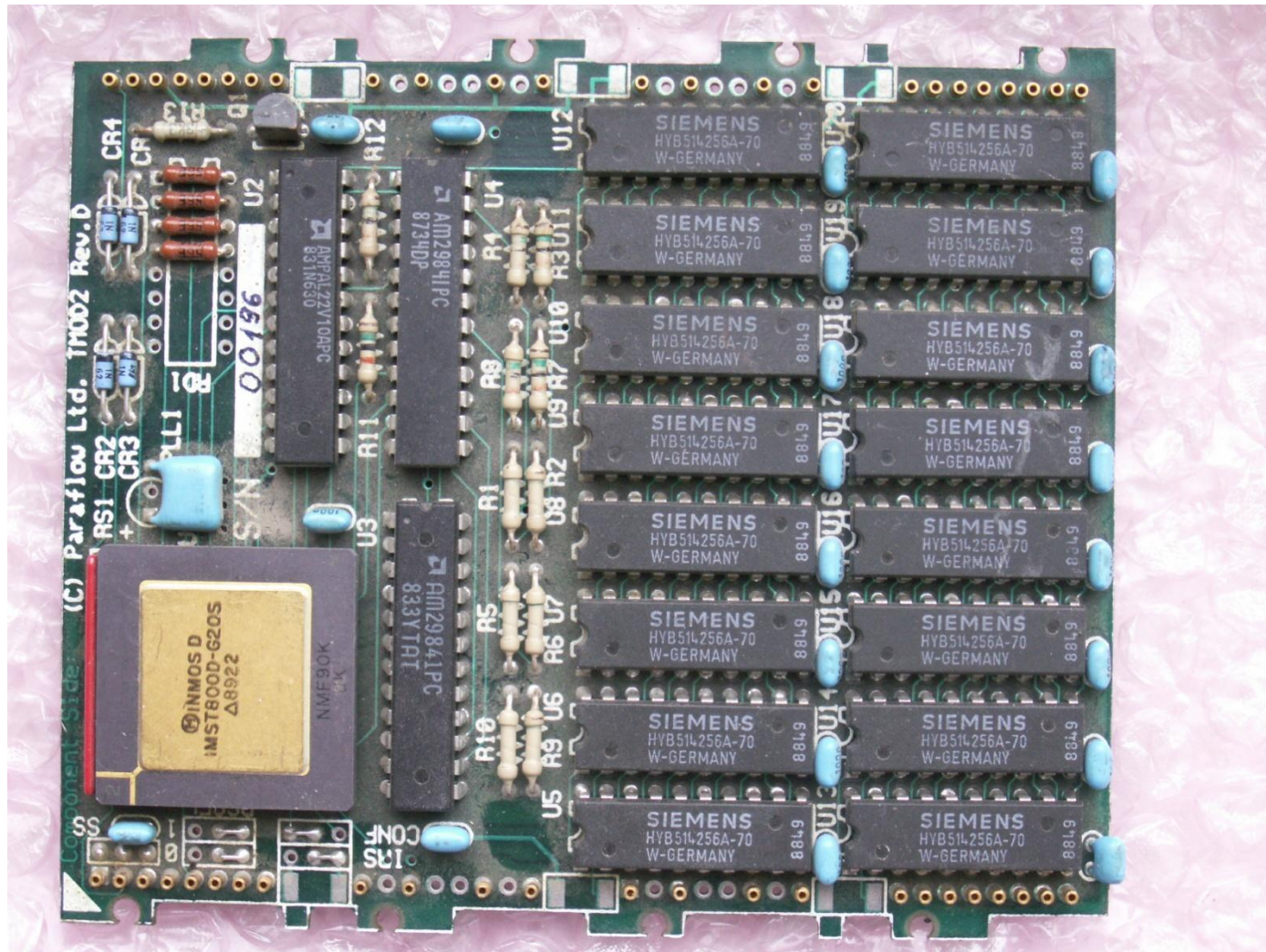
Транспьютерная материнская плата МТБ-8

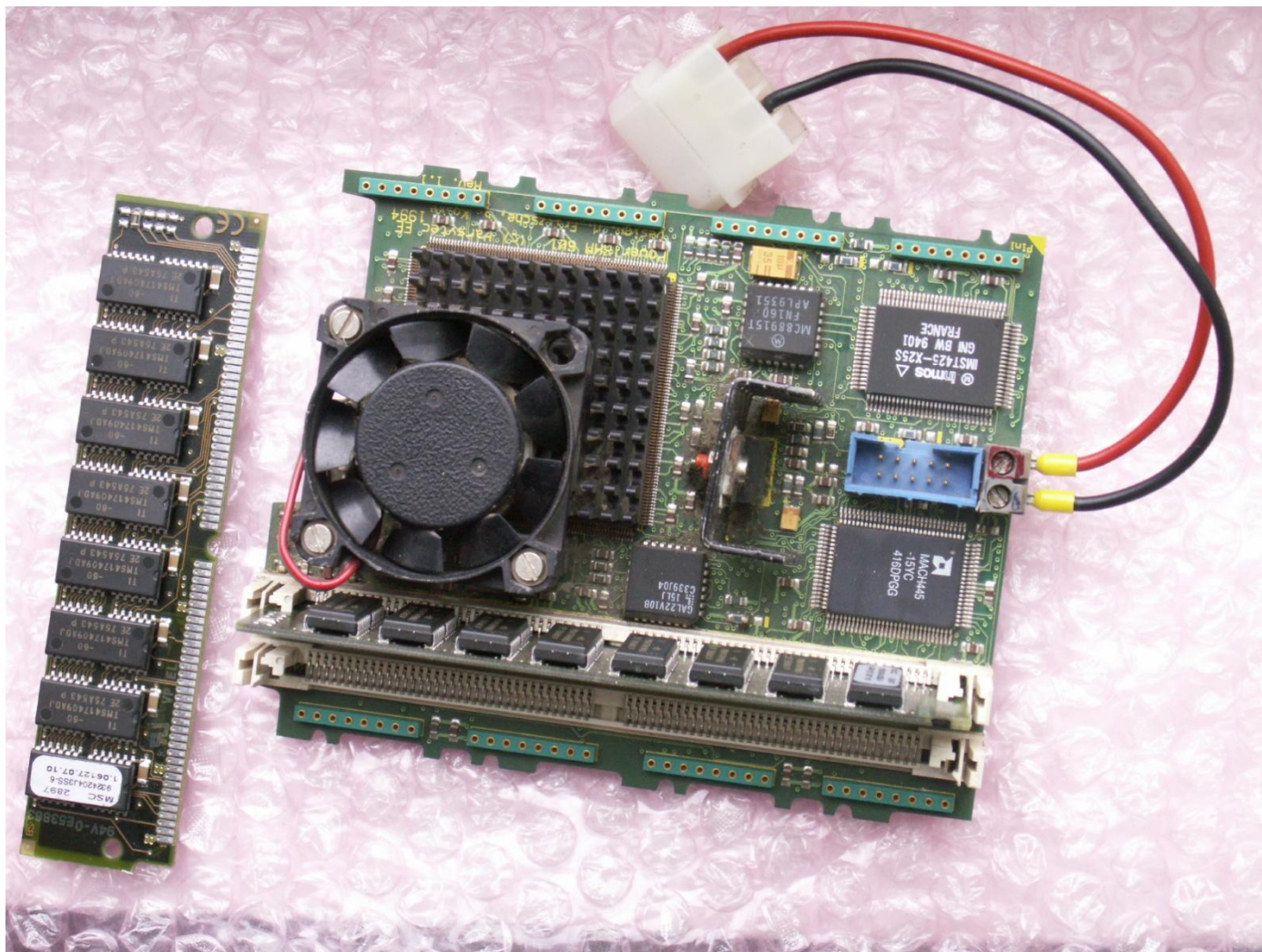


Электронный коммутатор

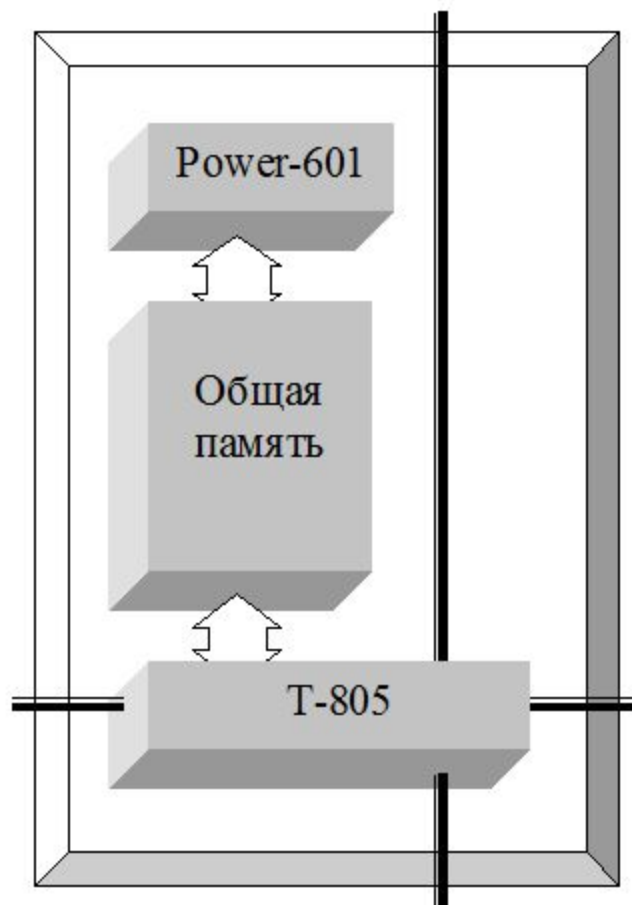


Электронно-реконфигурируемое соединение транспьютеров с помощью коммутатора C004



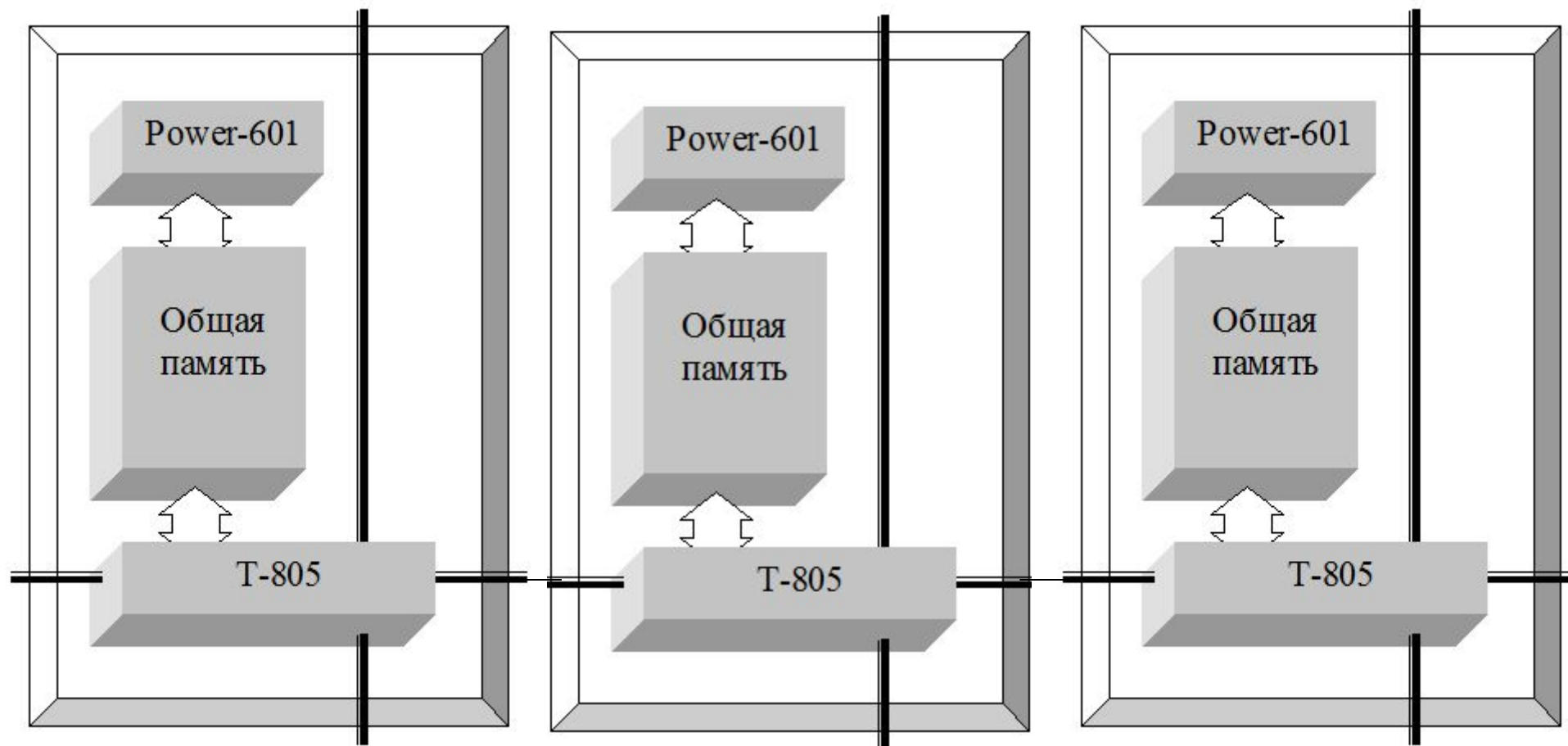


Узел PowerXplorer

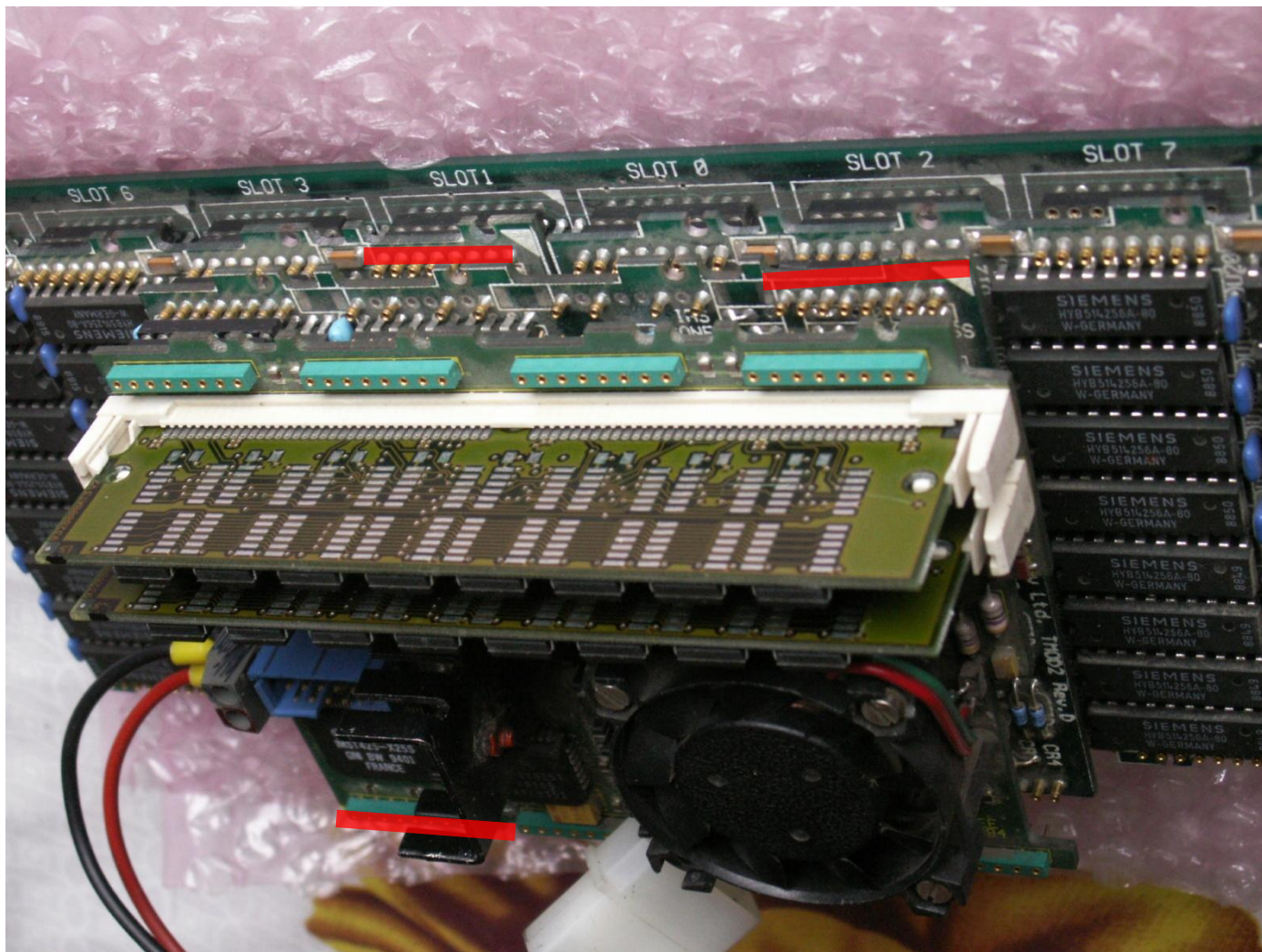


Структура узла PowerXplorer

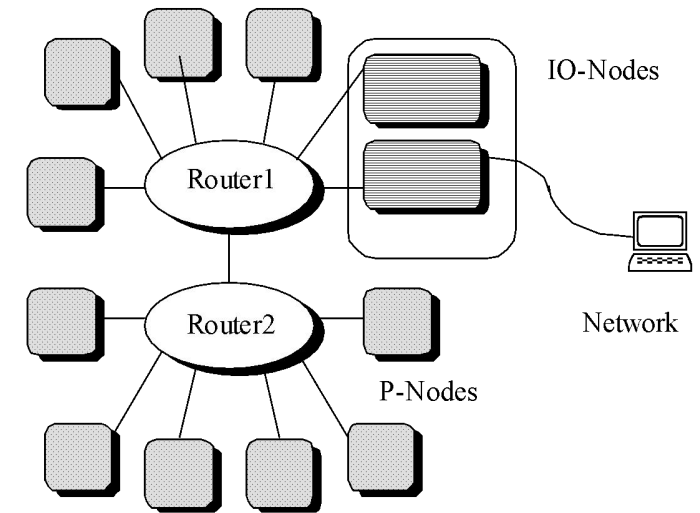
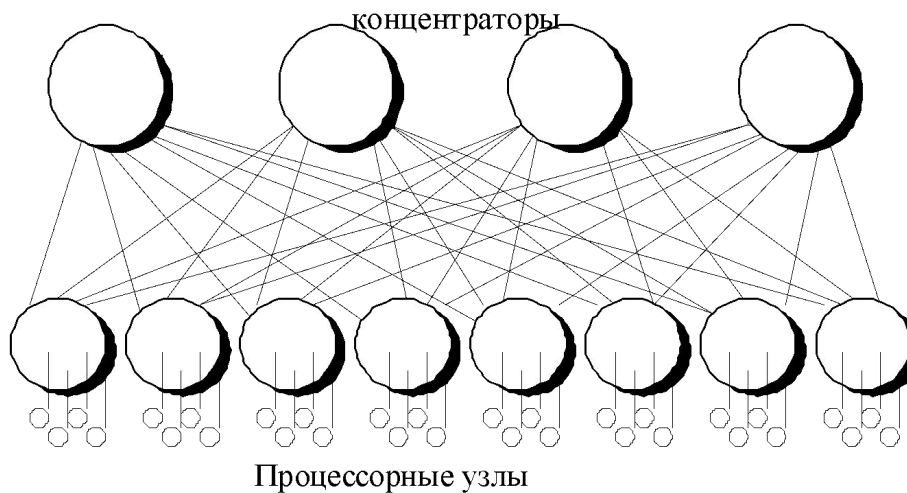
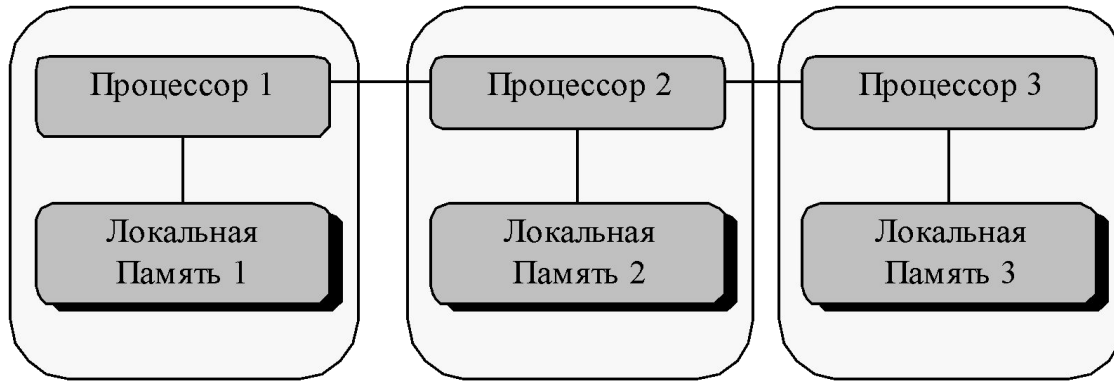
Гибридная система



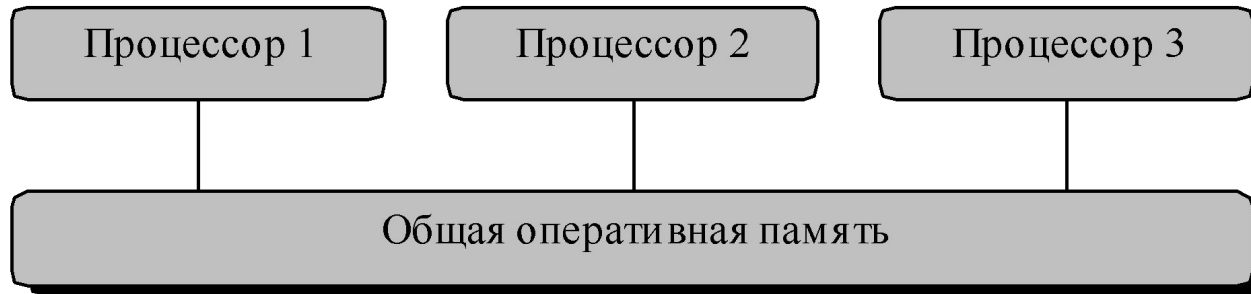
Плата и 4 модуля



Многопроцессорные системы с распределенной памятью

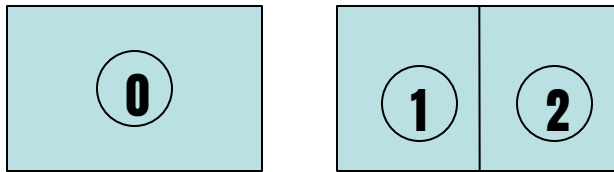


Многопроцессорные системы с общей памятью



Модель программы на распределенной памяти

- ❑ При запуске указываем число требуемых процессоров N_p и название программы
- ❑ На выделенных для расчета узлах запускается N_p копий указанной программы
 - Например, на двух узлах запущены три копии программы. Копия программы с номером 1 не имеет непосредственного доступа к оперативной памяти копий 0 и 2:

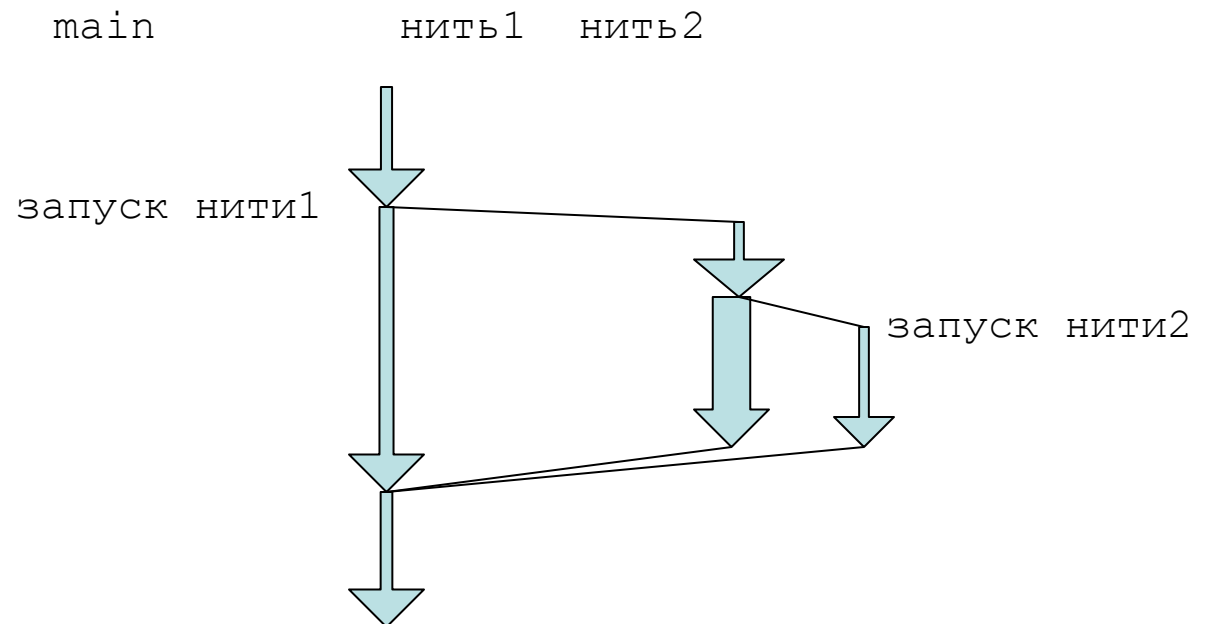


- ❑ Каждая копия программы получает две переменные
 - N_p
 - rank из диапазона $[0 \dots N_p-1]$
- ❑ Любые две копии программы могут непосредственно обмениваться данными с помощью функций передачи сообщений

Модель программы на общей памяти

- ❑ Работа начинается с запуска одной программы
- ❑ При необходимости программа порождает новые процессы, эти процессы:
 - Обладают собственными локальными переменными
 - Имеют доступ к глобальным переменным

```
int a_global;  
main()  
{  
  int b1_local;  
  Запуск нити(fun())  
}  
fun()  
{  
  int b2_local;  
  Запуск нити(...)  
}
```



Методы передачи данных

- ❑ Синхронный метод

Send(адрес данных, размер, номер процессора)

Recv(адрес данных, размер, номер процессора)

- ❑ Асинхронные методы

- Небуферизованный

ASend(адрес данных, размер, номер процессора)

AREcv(адрес данных, размер, номер процессора)

ASync

- Буферизованный

ABSend(адрес данных, размер, номер процессора)

Синхронные

- ❑ A=3

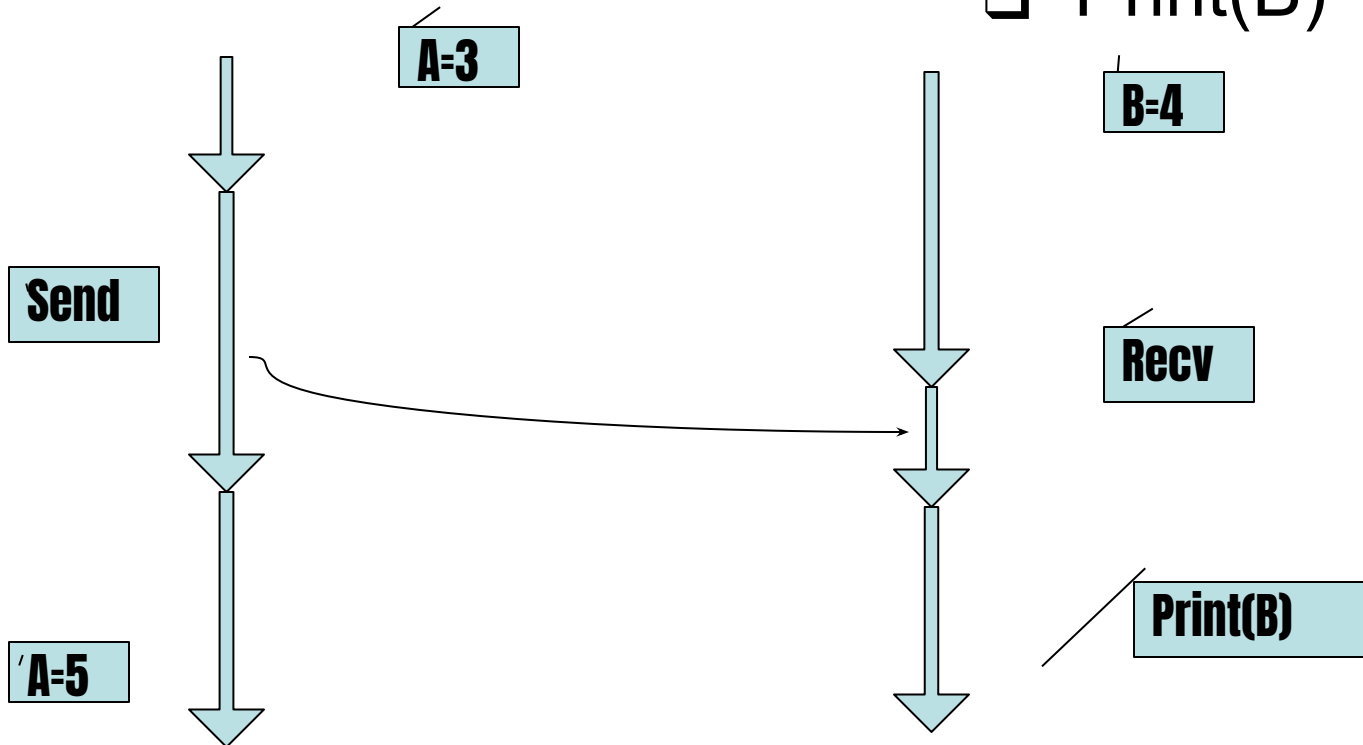
- ❑ Send(&A)

- ❑ A=5

- ❑ B=4

- ❑ Recv(&B)

- ❑ Print(B)



Результат
3

Асинхронные

- ❑ A=3

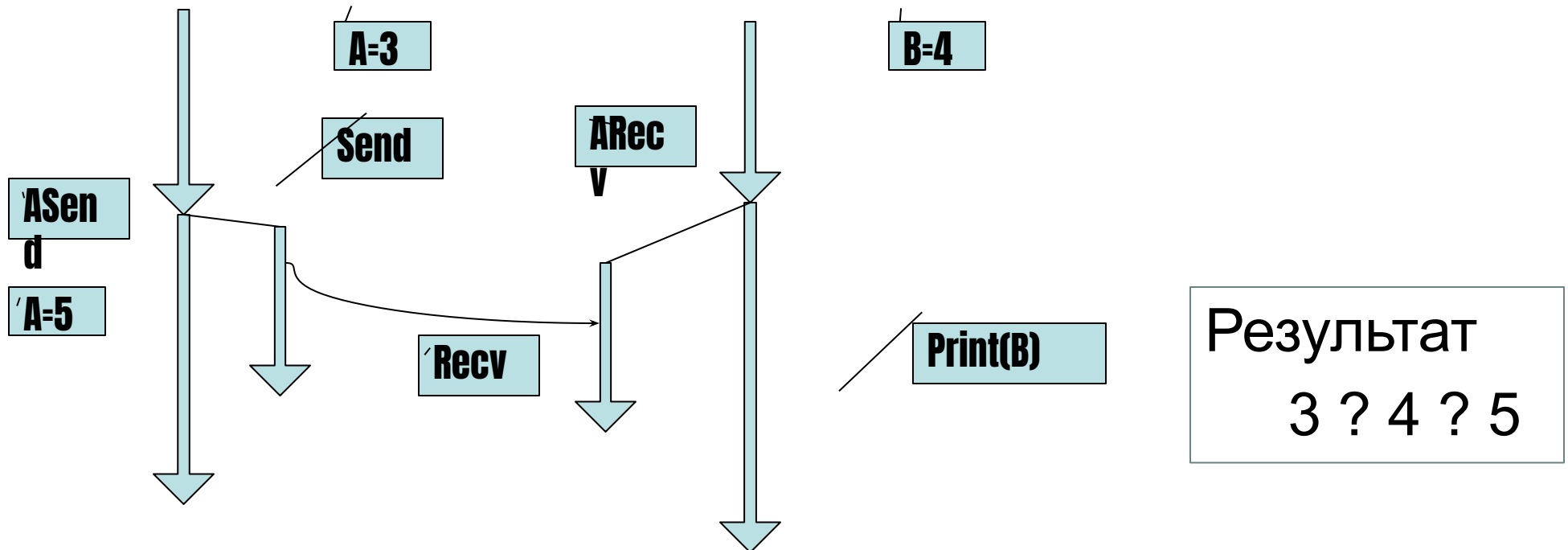
- ❑ ASend(&A)

- ❑ A=5

- ❑ B=4

- ❑ ARecv(&B)

- ❑ Print(B)



Асинхронные

- ❑ A=3

- ❑ ASend(&A)

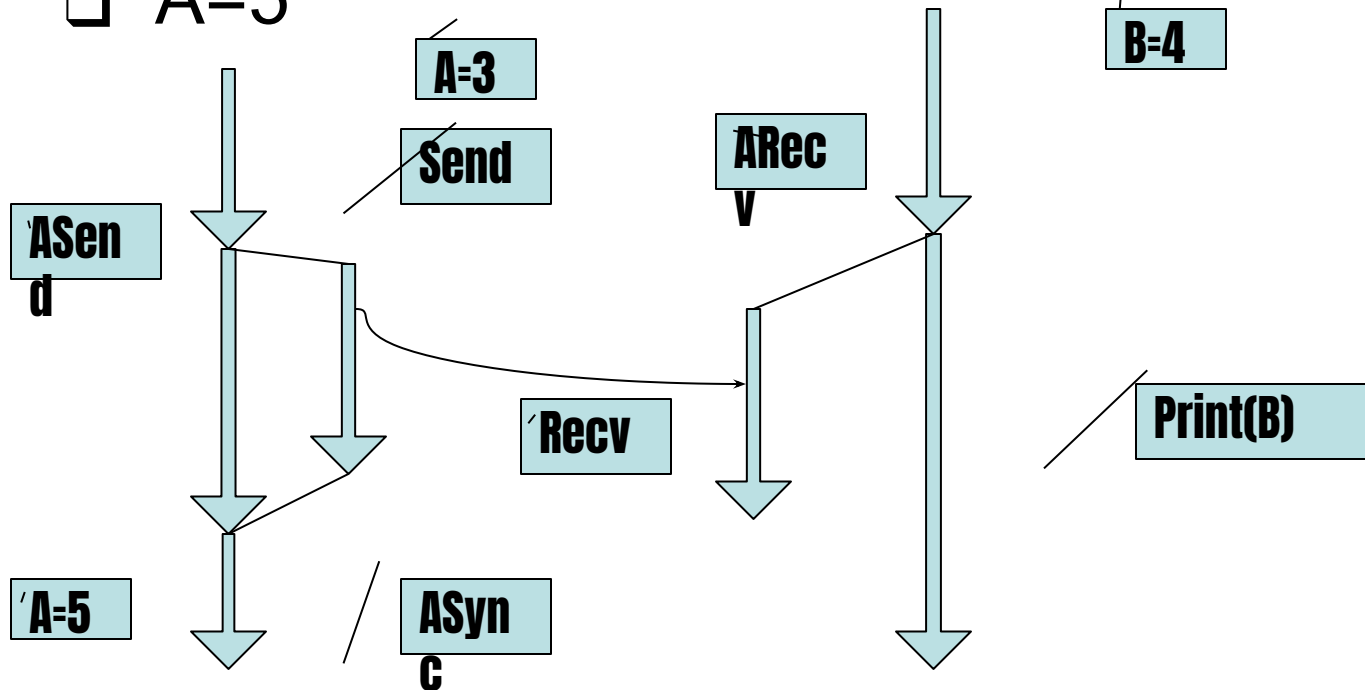
- ❑ Async()

- ❑ A=5

- ❑ B=4

- ❑ ARecv(&B)

- ❑ Print(B)



Результат
3 ? 4

Асинхронные

- ❑ A=3

- ❑ ASend(&A)

- ❑ Async()

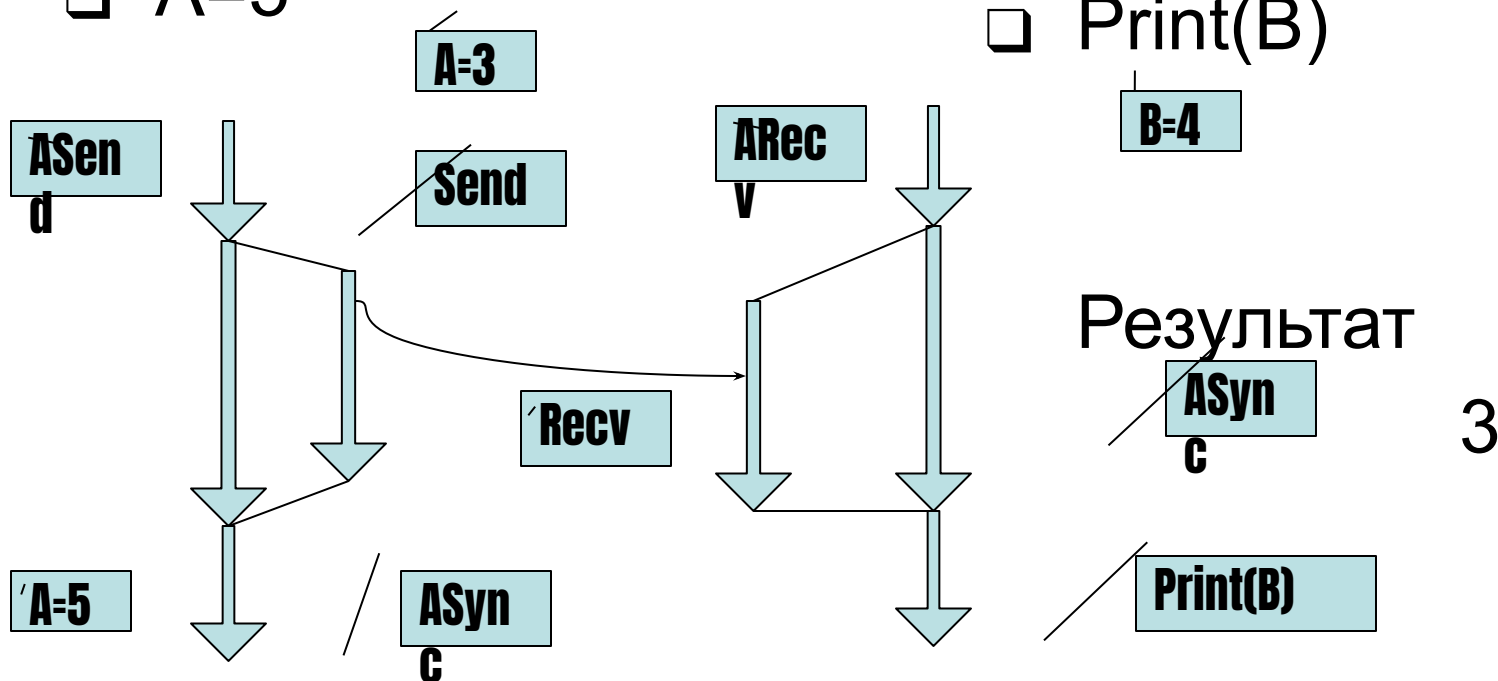
- ❑ A=5

- ❑ B=4

- ❑ ARecv(&B)

- ❑ Async()

- ❑ Print(B)



Асинхронные буферизованные

- ❑ A=3

- ❑ ABSend(&A)

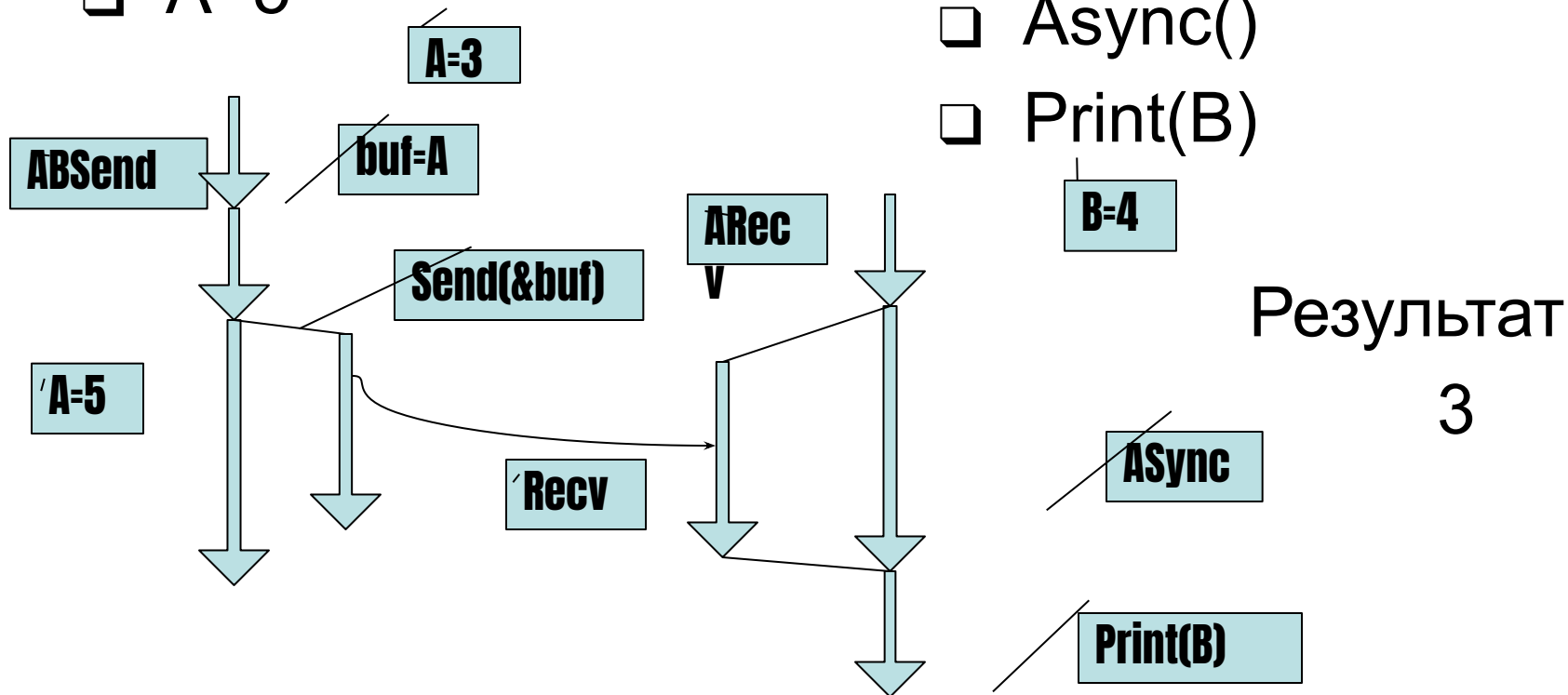
- ❑ A=5

- ❑ B=4

- ❑ ARecv(&B)

- ❑ Async()

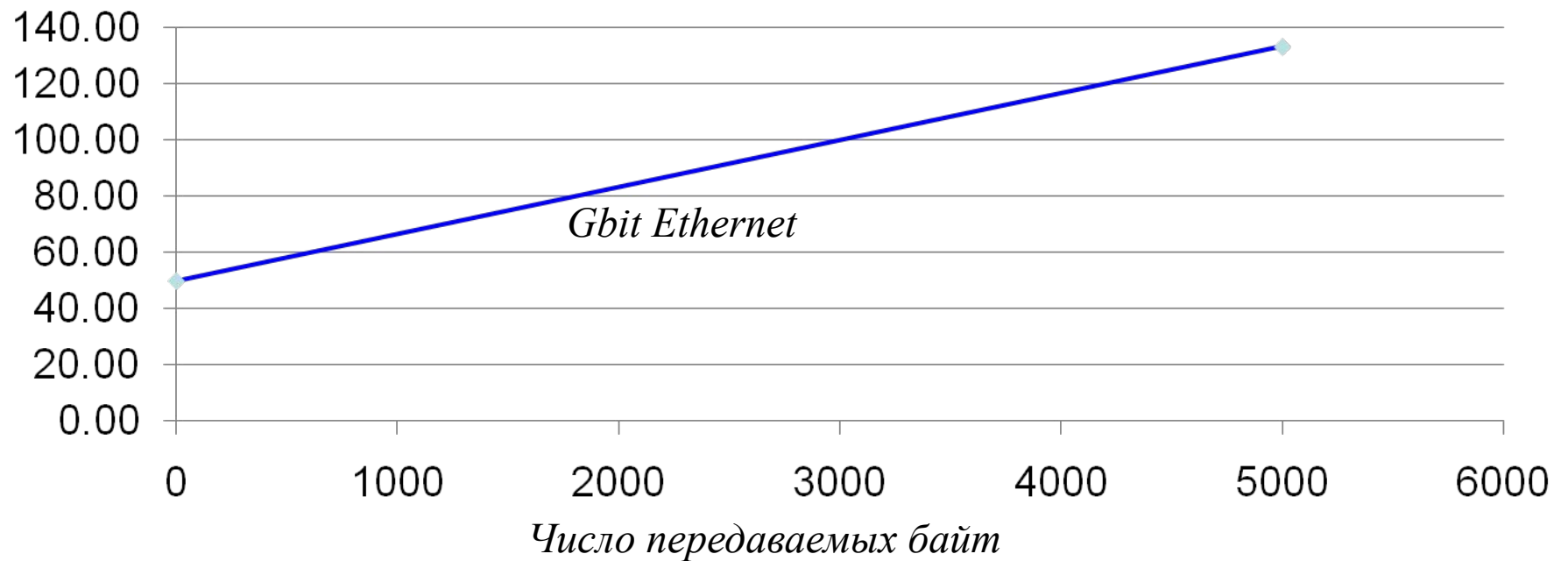
- ❑ Print(B)



Свойства канала передачи данных

$$T(n) = n * T_{\text{передачи байта}} + T_{\text{латентности}}$$

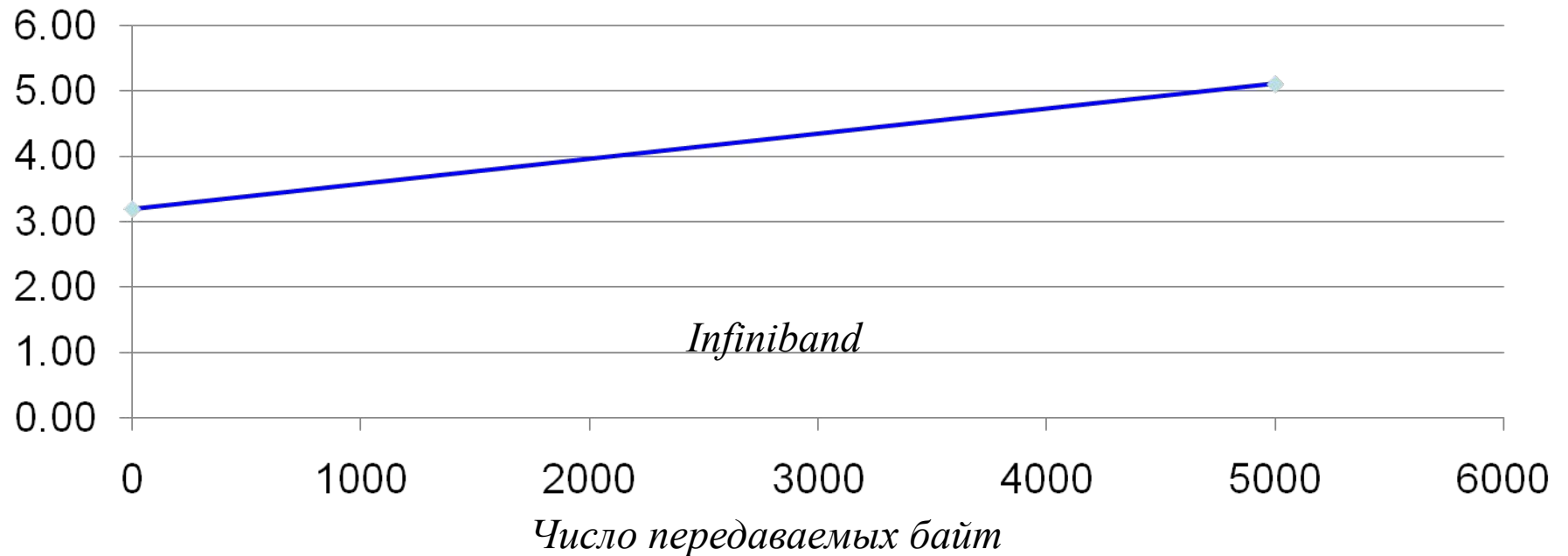
Время передачи данных (мкс)



Свойства канала передачи данных

$$T(n) = n * T_{\text{передачи байта}} + T_{\text{латентности}}$$

Время передачи данных (мкс)



Семафор

- Целочисленная неотрицательная переменная
- Две атомарные операции, не считая инициализации
- $V(S)$
 - Увеличивает значение S на 1
- $P(S)$
 - Если S положительно, то уменьшает S на 1
 - Иначе ждет, пока S не станет больше 0



Языки программирования. Редактор Ф.Женюи. Перевод с англ В.П. Кузнецова. Под ред. В.М.Курочкина. М.: "Мир", 1972 Э. Дейкстра. Взаимодействие последовательных процессов.

<http://khpi-iip.mipk.kharkiv.edu/library/extant/dijkstra/ewd123/index.html>

Ускорение и эффективность параллельных алгоритмов

**ускорение
параллельного
алгоритма**

$$S(p) = T_1 / T(p)$$

**эффективность
использования
процессорной мощности**

$$E(p) = S(p) / p$$

**Ускорение параллельного
алгоритма относительно
наилучшего последовательного**

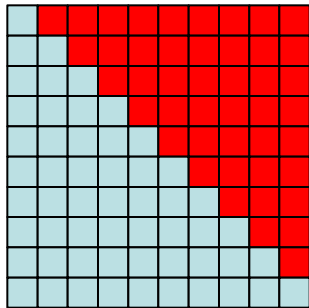
$$S^*(p) = T_1^* / T(p)$$

$$E^*(p) = S^*(p) / p$$

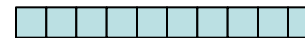
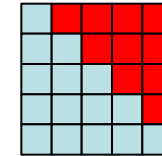
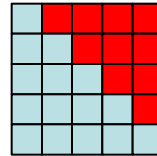
Может ли быть $S(p) > p$?

□ Да:

- Плохой последовательный алгоритм
- Влияние аппаратных особенностей вычислительной системы



$$\frac{n(n-1)}{2} \sim \frac{n^2}{2}$$



$$\frac{\left(\frac{n}{2}\right)^2}{2} + n \sim \frac{1}{4} \frac{n^2}{2}$$

Может ли неэффективный алгоритм работать быстрее эффективного?

- ❑ Да

- Если первый алгоритм позволяет использовать больше процессоров, чем второй.

- ❑ Самый эффективный алгоритм – использующий один (1) процессор.

- Его эффективность по определению равна 100%, но он не всегда самый быстрый.

Определить сумму конечного ряда

$$S = \sum_{i=0}^n \frac{x^i}{i!}$$

- ❑ Все элементарные операции (+ - * /)
выполняются за время T_c
- ❑ Все операции выполняются точно, результат не
зависит от порядка их выполнения
- ❑ Число процессоров неограничено

Последовательный алгоритм

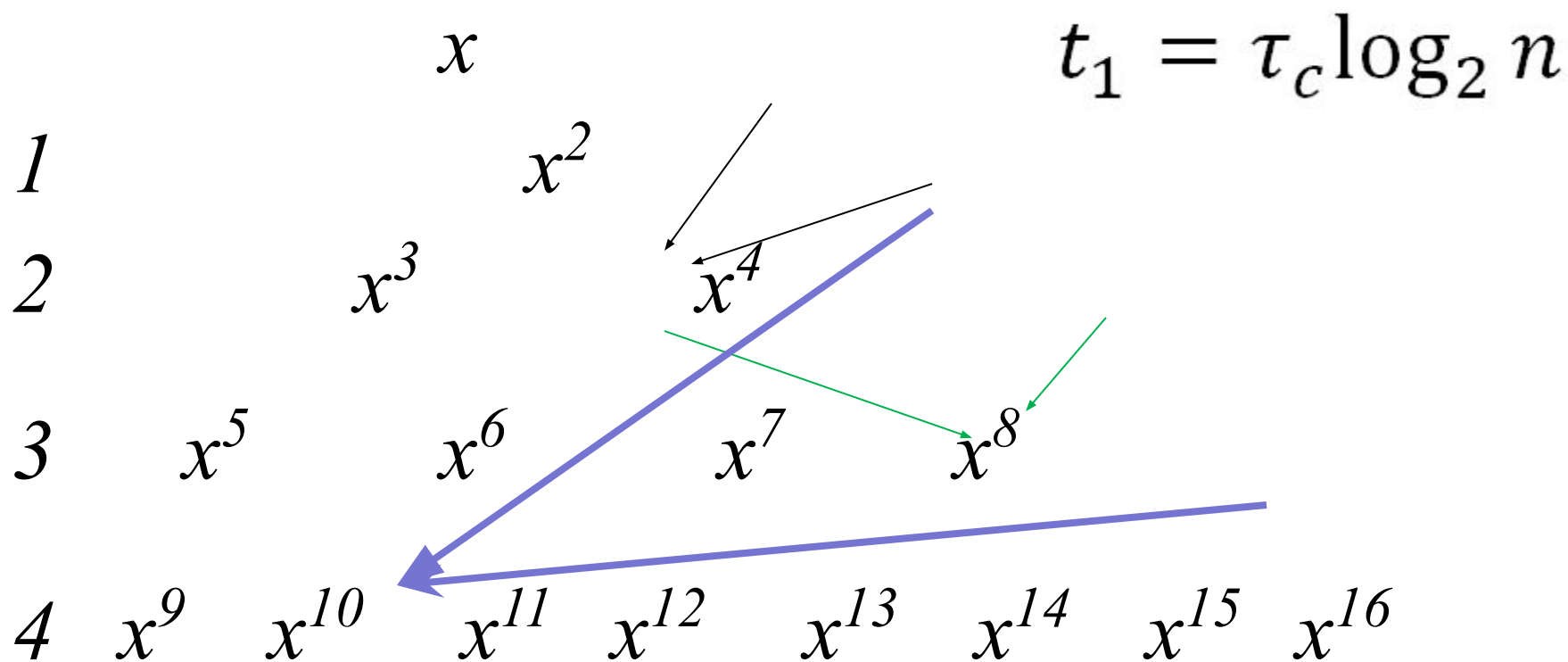
```
S=1;  
a=1;  
for (i=1; i<= n; i++)  
{  
    a=a*x/i;  
    S=S+a;  
}
```

$$T_1 = 3n\tau_c$$

Параллельный алгоритм

- ❑ Вычислить для всех $i = 1, \dots, n$: x^i
- ❑ Вычислить для всех $i = 1, \dots, n$: $i!$
- ❑ Вычислить для всех $i = 1, \dots, n$: $a_i = \frac{x^i}{i!}$
- ❑ Вычислить сумму всех членов a_i

Для вычисления x^i воспользуемся методом
бинарного умножения

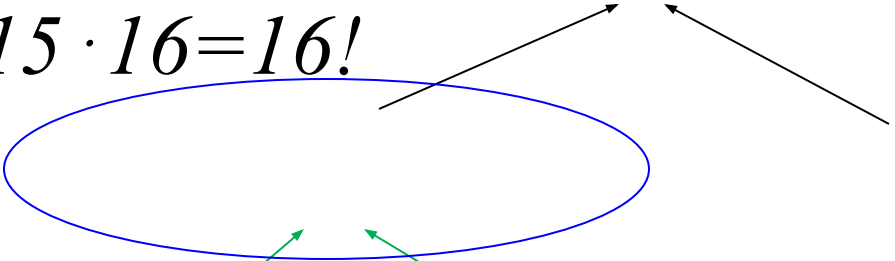


Параллельное вычисление всех требуемых $i!$?

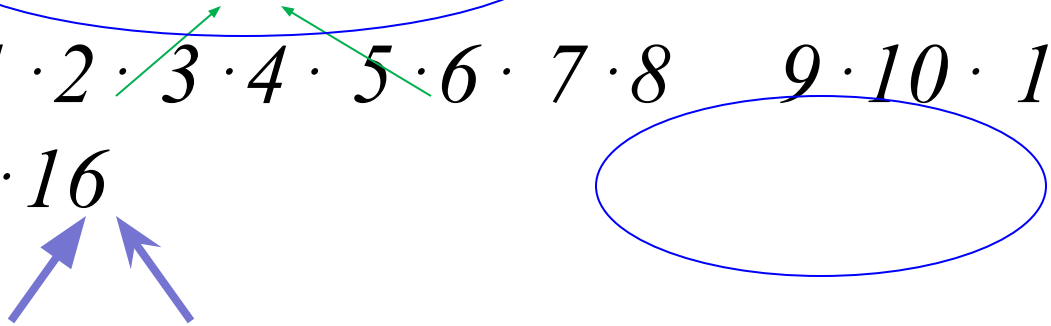
?

$i!$

4 $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11 \cdot 12 \cdot 13 \cdot 14 \cdot$
 $15 \cdot 16 = 16!$



3 $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11 \cdot 12 \cdot 13 \cdot 14 \cdot$
 $15 \cdot 16$



2 $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11 \cdot 12 \cdot 13 \cdot 14 \cdot$

12!

Для вычисления $i!$ воспользуемся
аналогичным методом

$$4 \quad 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11 \cdot 12 = 12!$$

$$3 \quad \underbrace{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot}_{\text{arrow to } 4} 7 \cdot 8 \quad 9 \cdot 10 \cdot 11 \cdot 12 \cdot 13 \cdot 14 \cdot 15 \cdot 16$$

$$2 \quad 1 \cdot 2 \cdot 3 \cdot 4 \quad 5 \cdot 6 \cdot 7 \cdot 8 \quad 9 \cdot 10 \cdot 11 \cdot 12 \quad 13 \cdot 14 \cdot 15 \cdot 16$$

14!

Для вычисления $i!$ воспользуемся
аналогичным методом

$$t_2 = \tau_c \log_2 n$$

$$4 \quad 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11 \cdot 12 \cdot 13 \cdot 14 = 14!$$

$$3 \quad 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \quad 9 \cdot 10 \cdot 11 \cdot 12 \cdot 13 \cdot 14$$

$$2 \quad 1 \cdot 2 \cdot 3 \cdot 4 \quad 5 \cdot 6 \cdot 7 \cdot 8 \quad 9 \cdot 10 \cdot 11 \cdot 12$$

$$1 \quad 1 \cdot 2 \quad 3 \cdot 4 \quad 5 \cdot 6 \quad 7 \cdot 8 \quad 9 \cdot 10 \quad 11 \cdot 12 \quad 13 \cdot 14$$

Новые операции

Вычисление всех факториалов до 8! включительно

<i>Шаг</i>	<i>Процессор 1</i>	<i>Процессор 2</i>	<i>Процессор 3</i>	<i>Процессор 4</i>
<i>1</i>	<i>$1 \cdot 2$</i>			
<i>2</i>	<i>$12 \cdot 3$</i>			
<i>3</i>				

Новые операции

Вычисление всех факториалов до 8! включительно

<i>Шаг</i>	<i>Процессор 1</i>	<i>Процессор 2</i>	<i>Процессор 3</i>	<i>Процессор 4</i>
<i>1</i>	<i>$1 \cdot 2$</i>			
<i>2</i>	<i>$12 \cdot 3$</i>	<i>$12 \cdot 34$</i>		
<i>3</i>				

Новые операции

Вычисление всех факториалов до 8! включительно

<i>Шаг</i>	<i>Процессор 1</i>	<i>Процессор 2</i>	<i>Процессор 3</i>	<i>Процессор 4</i>
<i>1</i>	<i>$1 \cdot 2$</i>	<i>$3 \cdot 4$</i>		
<i>2</i>	<i>$12 \cdot 3$</i>	<i>$12 \cdot 34$</i>		
<i>3</i>				

Новые операции

Вычисление всех факториалов до 8! включительно

<i>Шаг</i>	<i>Процессор 1</i>	<i>Процессор 2</i>	<i>Процессор 3</i>	<i>Процессор 4</i>
<i>1</i>	<i>$1 \cdot 2$</i>	<i>$3 \cdot 4$</i>	<i>$5 \cdot 6$</i>	<i>$7 \cdot 8$</i>
<i>2</i>	<i>$12 \cdot 3$</i>	<i>$12 \cdot 34$</i>	<i>$\textcircled{56} \cdot 7$</i>	<i>$\textcircled{56} \cdot 78$</i>
<i>3</i>	<i>$1234 \cdot 5$</i>	<i>$1234 \cdot \textcircled{56}$</i>	<i>$1234 \cdot 567$</i>	<i>$1234 \cdot 567$</i> <i>8</i>

Новые операции

Вычисление всех факториалов до 8! включительно

Шаг	Процессор 1	Процессор 2	Процессор 3	Процессор 4
1	$1 \cdot 2$	$3 \cdot 4$	$5 \cdot 6$	$7 \cdot 8$
2	$12 \cdot 3$	$12 \cdot 34$	$\textcircled{56} \cdot 7$	$\textcircled{56} \cdot 78$
3	$1234 \cdot 5$	$1234 \cdot \textcircled{56}$	$1234 \cdot 567$	$1234 \cdot 567$ 8

```

F=1;
for(i=2; i <= n; i++)
    F*=i;
    
```

$$T_1(n) = \tau_c (n - 1)$$

$$T_{p=n/2}(n) = \tau_c \log_2 n$$

$$S = \frac{n-1}{\log_2 n} \bigg|_{\substack{n=8 \\ p=4}} = \frac{7}{3} < 4 = p$$

$$E_{p=4}(n=8) = \frac{7}{12}$$

Новые операции

Шаг	Процессор 1	Процессор 2	Процессор 3	Процессор 4
1	1 · 2	3 · 4	5 · 6	7 · 8
2	12 · 3	12 · 34	56 · 7	56 · 78
3	1234 · 5	1234 · 56	1234 · 567	1234 · 567

$$T_{p=n/2}(n) = t_c \log_2 n$$

$$S = \frac{n-1}{\log_2 n} \Big|_{\substack{n=8 \\ p=4}} = \frac{7}{3} < 4 = p$$

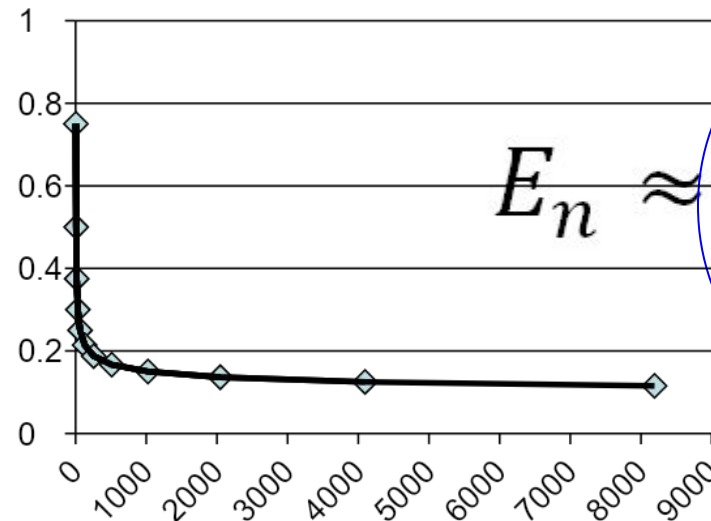
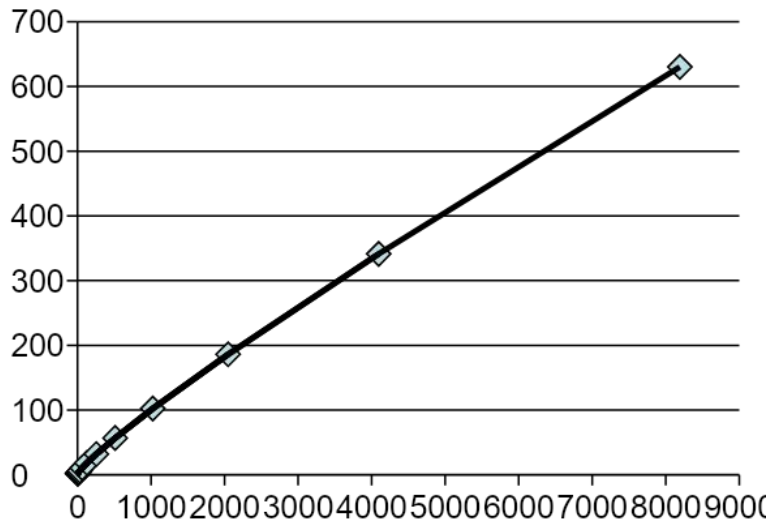
$$E_{p=4}(n=8) = \frac{7}{12}$$

Шаг	Процессор 1	Процессор 2	Процессор 3	Процессор 4
1	1 2!	8 3 · 4	9 5 · 6	10 7 · 8
2	2 3!	3 4!	11 56 · 7	12 56 · 78
3	4 5!	5 6!	6 7!	7 8!

Ускорение и эффективность при $p=n$

$p=n$

$$S_n = \frac{3n\tau_c}{2\tau_c \log_2 n + \tau_c} \approx \frac{n}{\log_2 n} \mathbf{1.5}$$



$$E_n \approx \frac{\mathbf{1.5}}{\log_2 n}$$

Заключение

- ❑ Определены классы рассматриваемых вычислительных систем
- ❑ Представлены модели параллельных программ
- ❑ Представлен ряд способов организации взаимодействия последовательных процессов
- ❑ Представлен алгоритм, обладающий низкой эффективностью, но высоким быстродействием

Вопросы для обсуждения

- ❑ Сколько нужно процессоров для вычисления суммы ряда $s = \sum_{i=0}^n \frac{x^i}{i!}$ за время

$$2 \tau_c \log_2 n + q \tau_c ,$$

где q – константа

- ❑ Какие преимущества и недостатки присущи синхронным и асинхронным методам обмена данными?
- ❑ Приведите примеры алгоритмов обладающих эффективностью больше 100%.
- ❑ Есть ли процессорные инструкции $P(S)$ и $V(S)$?

Якобовский М.В.

д.ф.-м.н., проф.,

зав. сектором

«Программного обеспечения многопроцессорных систем и вычислительных сетей»

Института прикладной математики им. М.В.

Келдыша Российской академии наук

mail: mail: lira@imamod.ru mail: lira@imamod.ru

web: web: <http://lira.imamod.ru>