Параллельные вычисления

Основные понятия
20230228

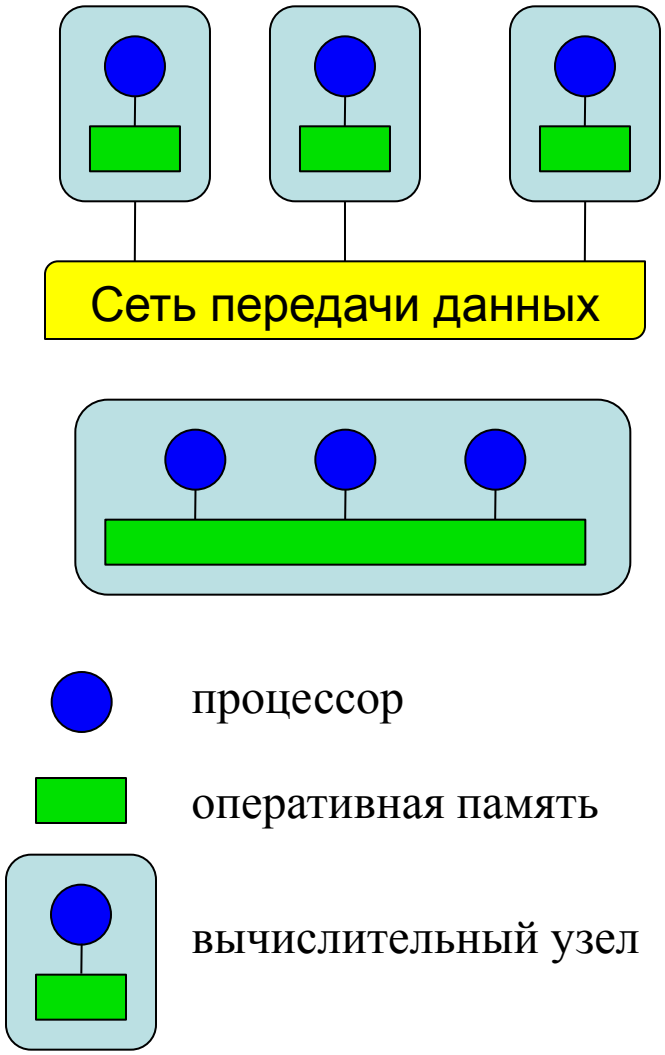
Якобовский Михаил Владимирович

МФТИ, г. Долгопрудный

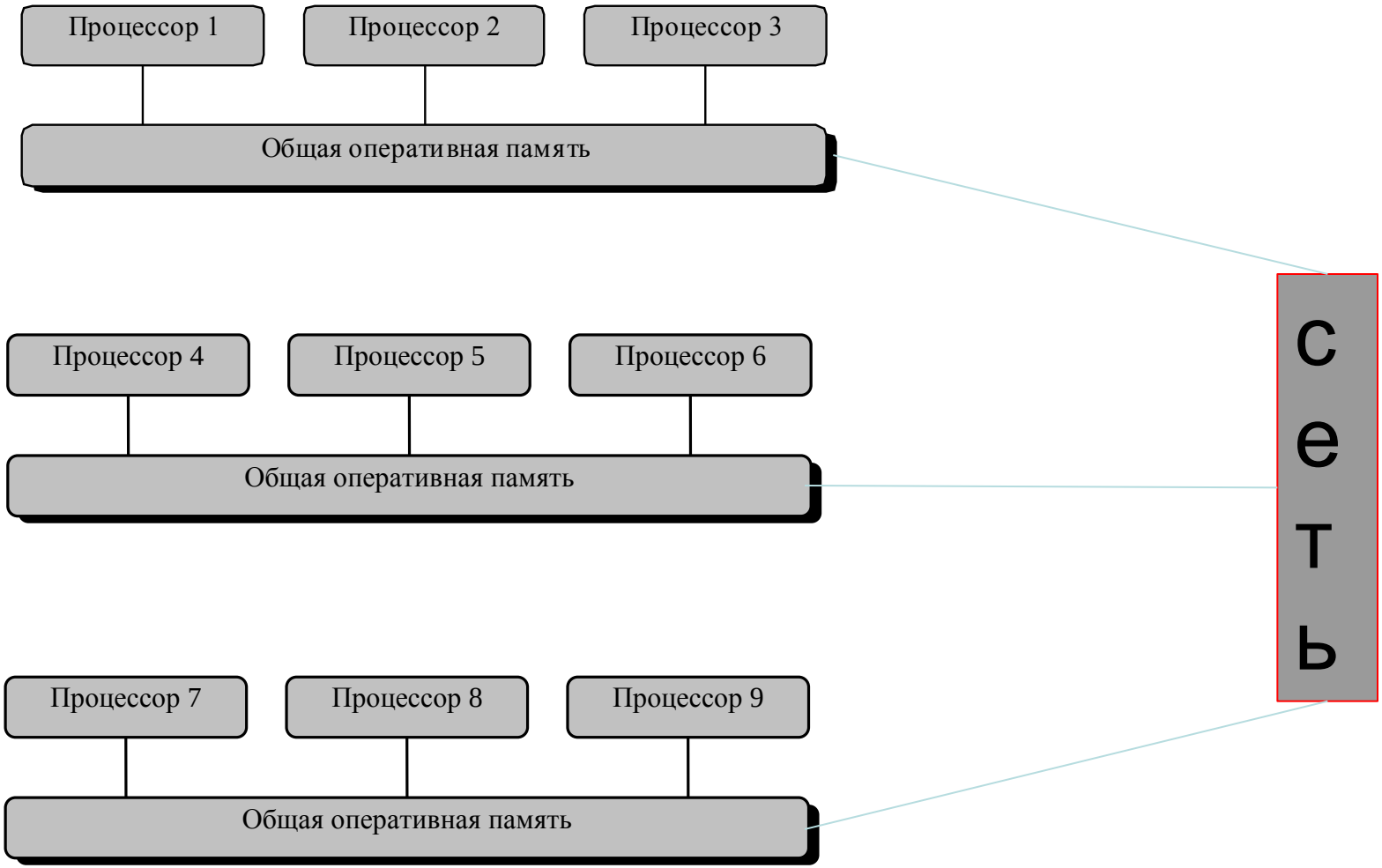
весна, 2023

Рассматриваемые системы

- Системы на основе объединенных сетью типовых вычислительных узлов – системы с распределенной оперативной памятью
- Системы с доступом всех процессоров к общей оперативной памяти



Гибридные вычислительные системы



Методы взаимодействия и синхронизации процессов

- Выполнение программы на вычислительной системе с распределённой оперативной памятью
  - Передача сообщений
- Выполнение программы на вычислительной системе с общей оперативной памятью
  - Семафоры

Модель выполнения программы на общей памяти

- Работа начинается с запуска одного экземпляра программы
- При необходимости программа порождает новые процессы, каждый из которых:
  - Обладает собственными локальными переменными
  - Имеет доступ к глобальным переменным

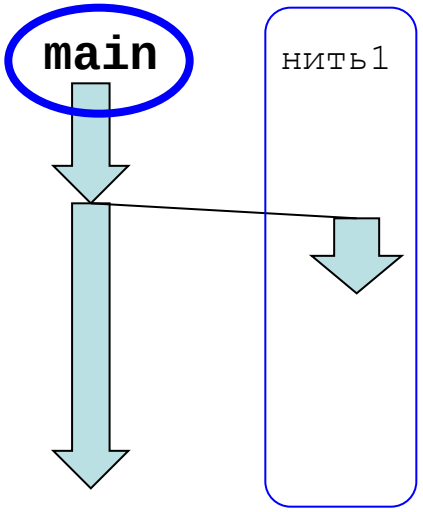
```
int a_global;
main()
{
int b1_local;
Запуск нити(fun())
}
fun()
{
int b2_local;
Запуск нити(…)
}
```



Модель выполнения программы на общей памяти

- Работа начинается с запуска одного экземпляра программы
- При необходимости программа порождает новые процессы, каждый из которых:
  - Обладает собственными локальными переменными
  - Имеет доступ к глобальным переменным

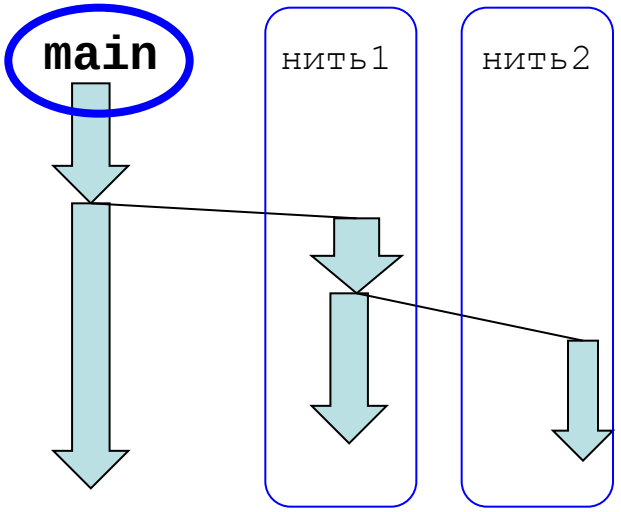
```
int a_global;
main()
{
int b1_local;
Запуск нити(fun())
}
fun()
{
int b2_local;
Запуск нити(…)
}
```



Модель выполнения программы на общей памяти

- Работа начинается с запуска одного экземпляра программы
- При необходимости программа порождает новые процессы, каждый из которых:
  - Обладает собственными локальными переменными
  - Имеет доступ к глобальным переменным

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

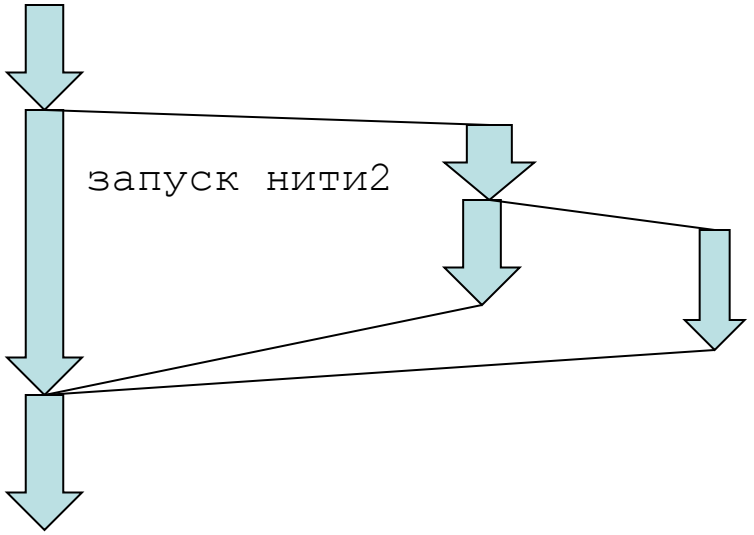


Модель программы на общей памяти

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

```
int a_global;
main()
{
Start_process(fun);
int b1_local;
}

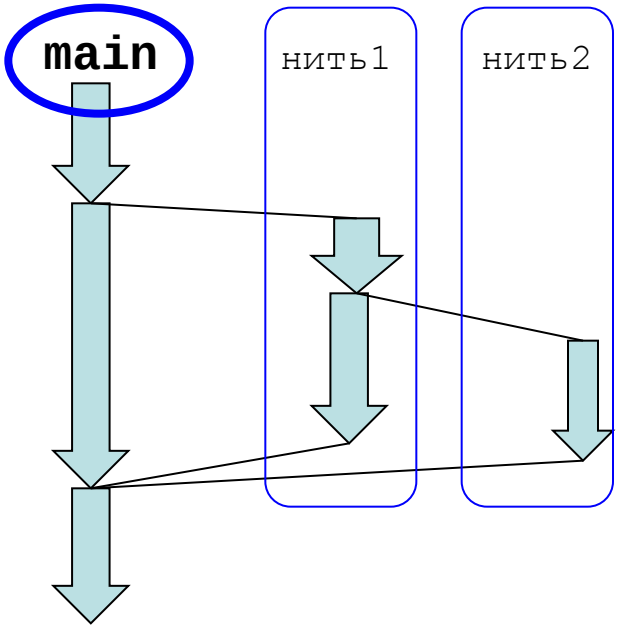
fun()
{
int b2_local;
}
```



Модель выполнения программы на общей памяти

- Работа начинается с запуска одного экземпляра программы
- При необходимости программа порождает новые процессы, каждый из которых:
  - Обладает собственными локальными переменными
  - Имеет доступ к глобальным переменным

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



Что будет напечатано?

```
int a=1;
```

```
Нить1
{
  a=a+2
}
```

```
Нить2
{
  a=a+3
}
```

```
Нить3
{
  print(a)
}
```

a=?

Что будет напечатано?

```
int a=1;
```

```
Нить1
{
   a=a+2
}
```

```
Нить2
{
   a=a+3
}
```

```
Нить3
{
   print(a)
}
```

6

Что будет напечатано?

```
int a=1;
```

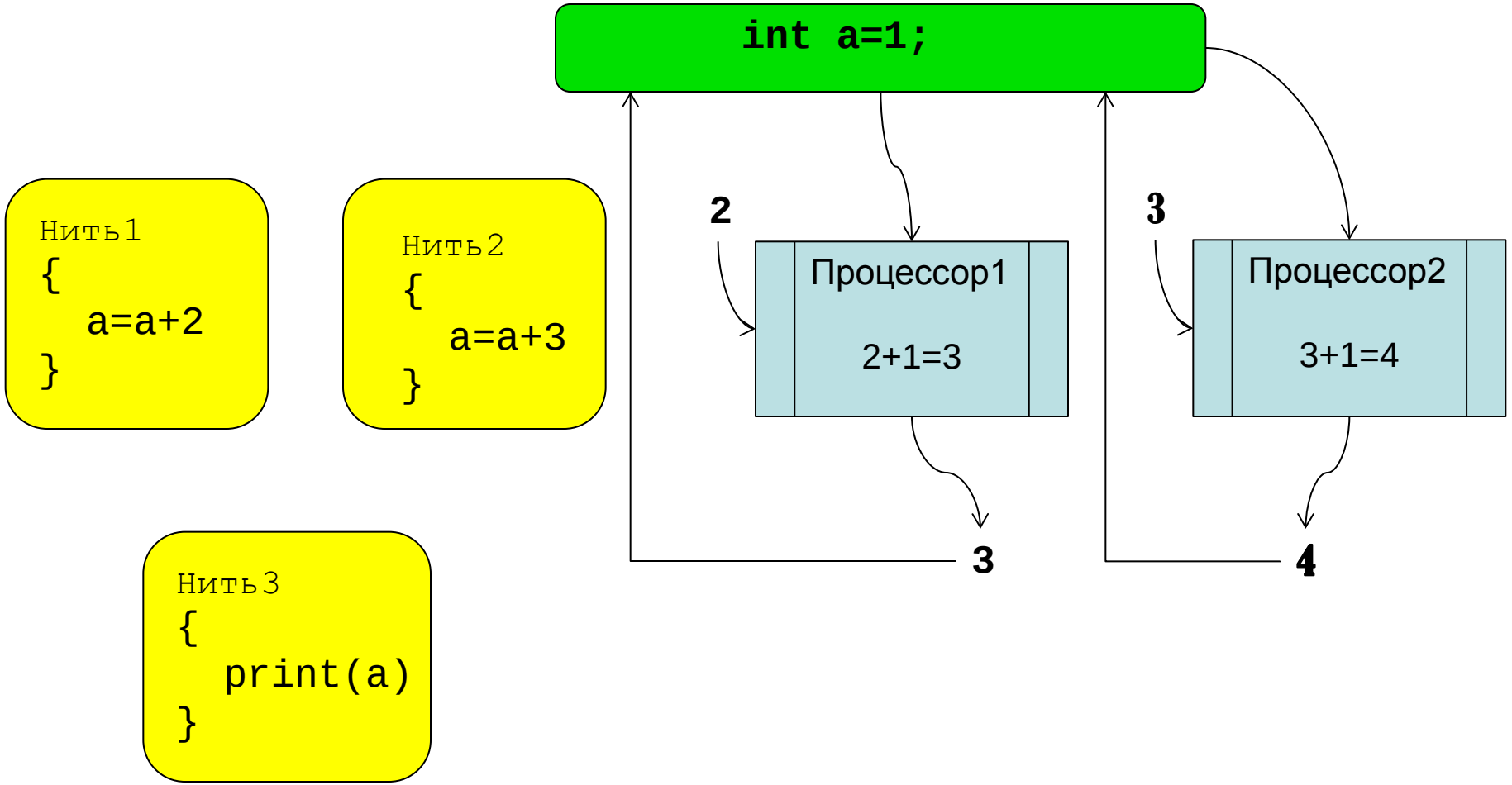
```
Нить1
{
  a=a+2
}
```

```
Нить3
{
  print(a)
}
```

```
Нить2
{
  a=a+3
}
```

3

Что будет напечатано?



6 ? 4 ? 3

a=?

```
int a=0
```

```
// Нить 1:

{  int i

   for(i=0;i<100;i++)
      a += 1
}
```

```
// Нить 2:

{  int i

   for(i=0;i<100;i++)
      a += 1
}
```

```
print a
```

a=?

```
a = 0;
```

```
// Нить 1:

{
  //  a += 1
    mov r1 , a
    inc r1
    mov a , r1
}
```

```
// Нить 2:

{
  //  a += 1
    mov r1 , a
    inc r1
    mov a , r1
}
```

a = 1 или a = 2 ?

Что будет напечатано?

```
int a=1;
```

```
Нить1
{
  a=a+2
}
```

```
Нить2
{
  a=a+3
}
```

```
Нить3
{
  print(a)
}
```

6 ? 4 ? 3 ? ___

Алгоритм Деккера


```
begin integer c1, c2, очередь;
      c1 := 1; c2 := 1; очередь := 1;
      parbegin
      процесс 1: begin A1: c1 := 0;
                     L1: if c2 = 0 then
                         begin if очередь = 1 then goto L1;
                               c1 := 1;
                           B1: if очередь = 2 then goto B1;
                               goto A1
                         end;
                         критический интервал 1;
                         очередь := 2; c1 := 1;
                         остаток цикла 1; goto A1
                 end;
      процесс 2: begin A2: c2 := 0;
                     L2: if c1 = 0 then
                         begin if очередь = 2 then goto L2;
                               c2 = 1;
                           B2: if очередь = 1 then goto B2;
                               goto A2
                         end;
                         критический интервал 2;
                         очередь := 1; c2 := 1;
                         остаток цикла 2; goto A2
                 end;
      parend;
end
```

Алгоритм Деккера

http://lectures.stargeo.ru/add_qst/semaph.htm Староверов В.М.

```
//указывает занятость ресурса, соответственно, первым и вторым процессами
int flag[2]={0,0};
int Num=0;
```

```
//Процесс 1:

void CriticalBegin1()
{
  flag[0]=1;//занять ресурс первым процессом
 while(flag[1]==1)//пока ресурс занят вторым процессом
       if(Num==1)//если надо уступить второму процессу
         {
          flag[0]=0;//освободим ресурс
          while(Num==1); //подождем завершения второго процесса
          flag[0]=1;//займем ресурс
         }
}

void CriticalEnd1();
{
 Num=1;
 flag[0]=0;//освободить ресурс первым процессом
}
```

```
//Процесс 2:

void CriticalBegin2()
{
  flag[1]=1;//занять ресурс вторым процессом
 while(flag[0]==1)//пока ресурс занят первым процессом
     if(Num==0)//если надо уступить первому процессу
       {
        flag[1]=0;//освободим ресурс
        while(Num==0); //подождем завершения первого процесса
        flag[1]=1;//займем ресурс
       }
}

void CriticalEnd2();
{
 Num=0;
 flag[1]=0;//освободить ресурс вторым процессом
}
```

Семафор

- Целочисленная неотрицательная переменная
- Инициализация и две **атомарные** операции

- Операция V(S) – атомарная !
  - Атомарно увеличивает значение S на 1

- Операция P(S) – атомарная !
  - 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/extent/dijkstra/ewd123/index.html

Что будет напечатано?

```
int a=1;
Sem S=1, S1=0, S2=0;
```

```
Нить1
{
  P(S)
  a=a+2
  V(S)
  V(S1)
}
```

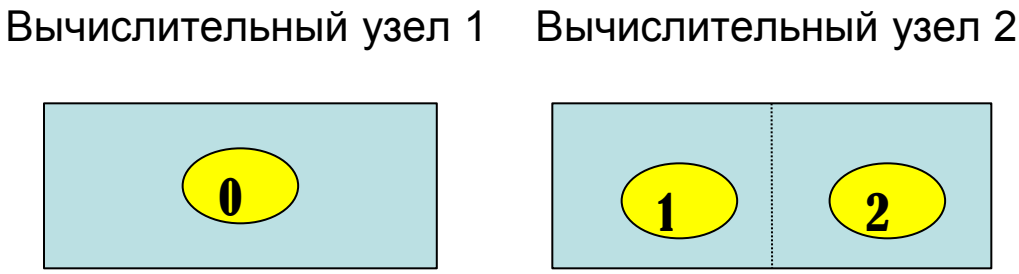
```
Нить2
{
  P(S)
  a=a+3
  V(S)
  V(S2)
}
```

```
Нить3
{
 P(S1)
 P(S2)
 print(a)
}
```

6

Модель выполнения программы на распределенной памяти

- При запуске указывается число требуемых процессоров Np и название программы
- На выделенных для расчета узлах запускается Np копий 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:

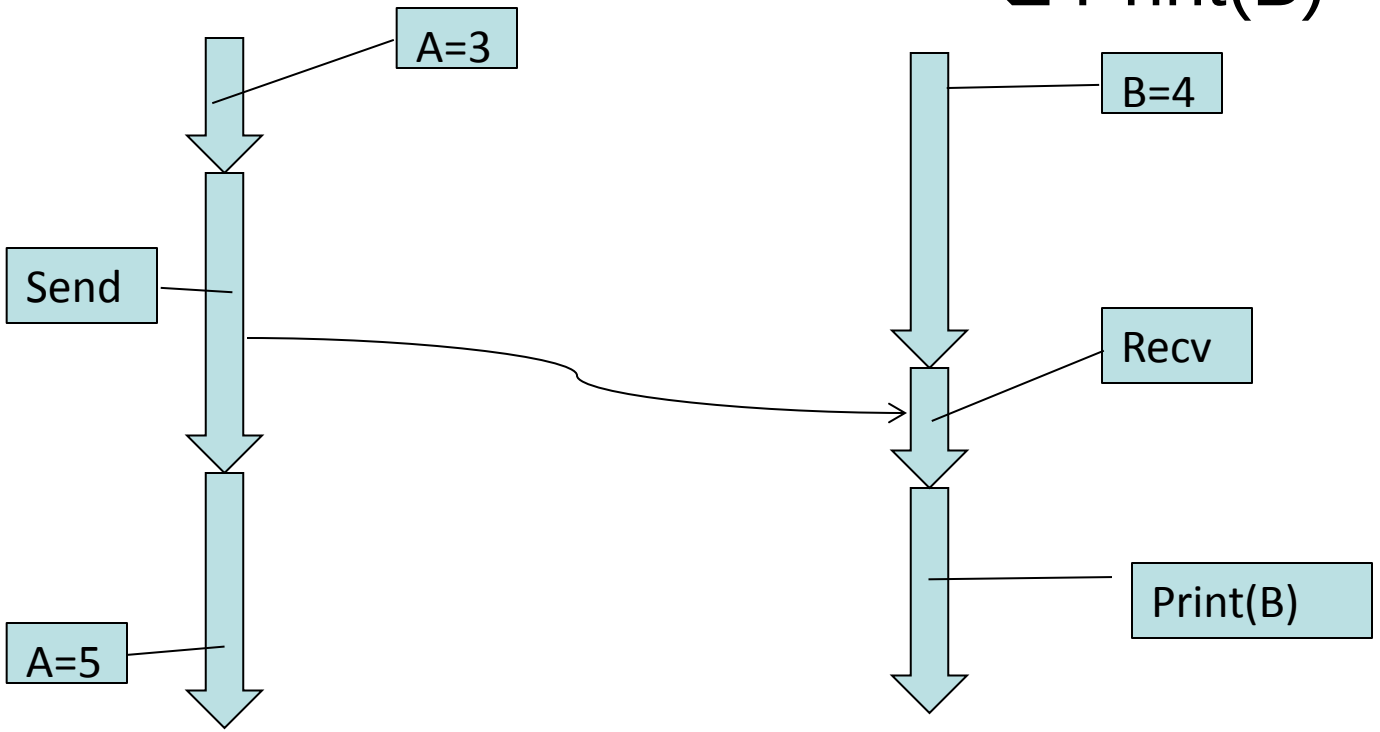


- В каждой копии программы известны значения двух переменных
  - Np – одинаковое во всех копиях – число копий
  - rank из диапазона [0 … Np-1] – уникальный номер копии
- Любые две копии программы могут непосредственно обмениваться данными с помощью функций передачи сообщений Send/Recv

Синхронный метод передачи сообщений

- A=3
- Send(&A)
- A=5

- B=4
- Recv(&B)
- Print(B)



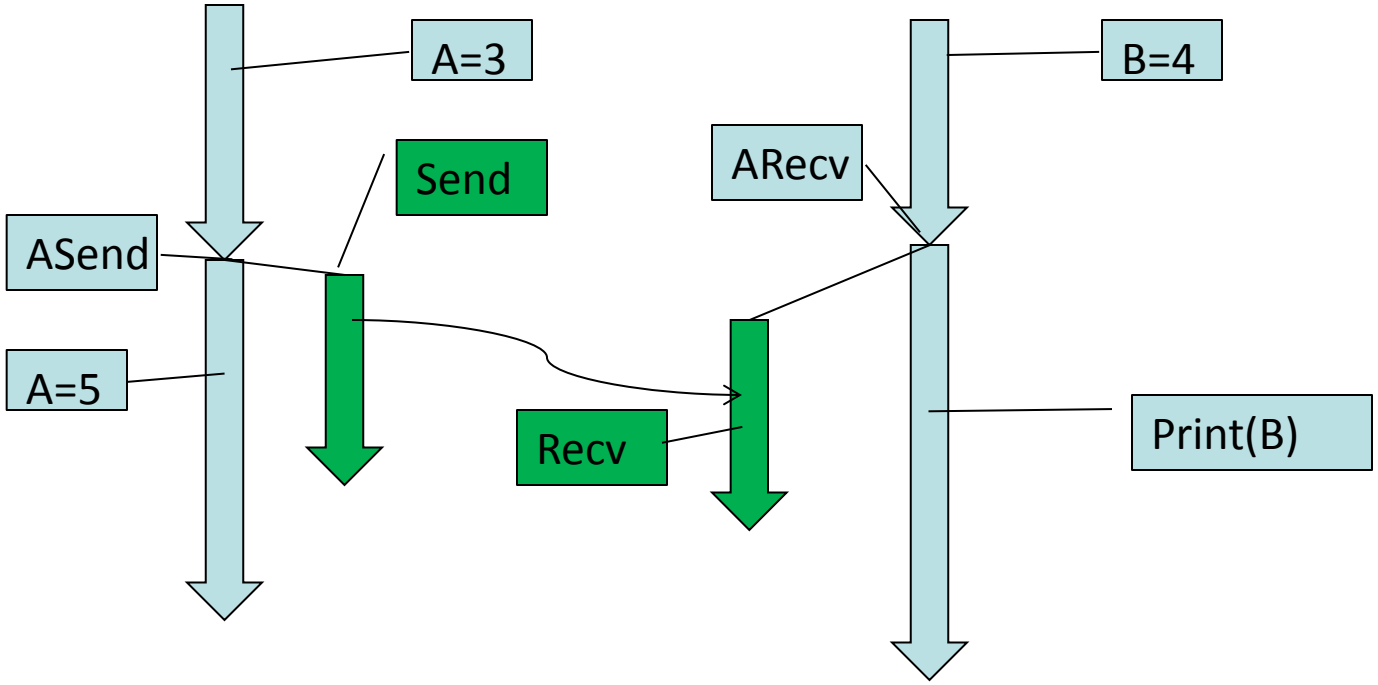
Результат

3

Асинхронные методы передачи сообщений

- A=3
- ASend(&A)
- A=5

- B=4
- ARecv(&B)
- Print(B)



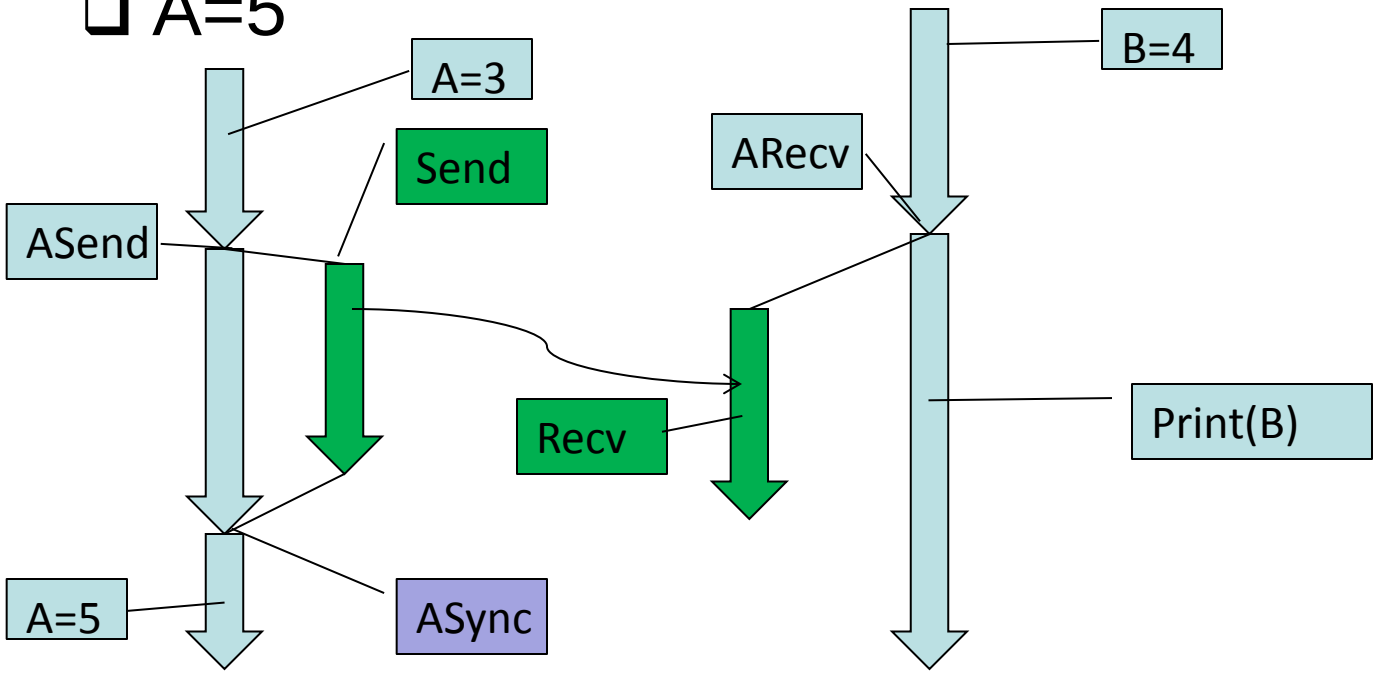
Результат

3 ? 4 ? 5

Асинхронные

- A=3
- ASend(&A)
- Async()
- A=5

- B=4
- ARecv(&B)
- Print(B)



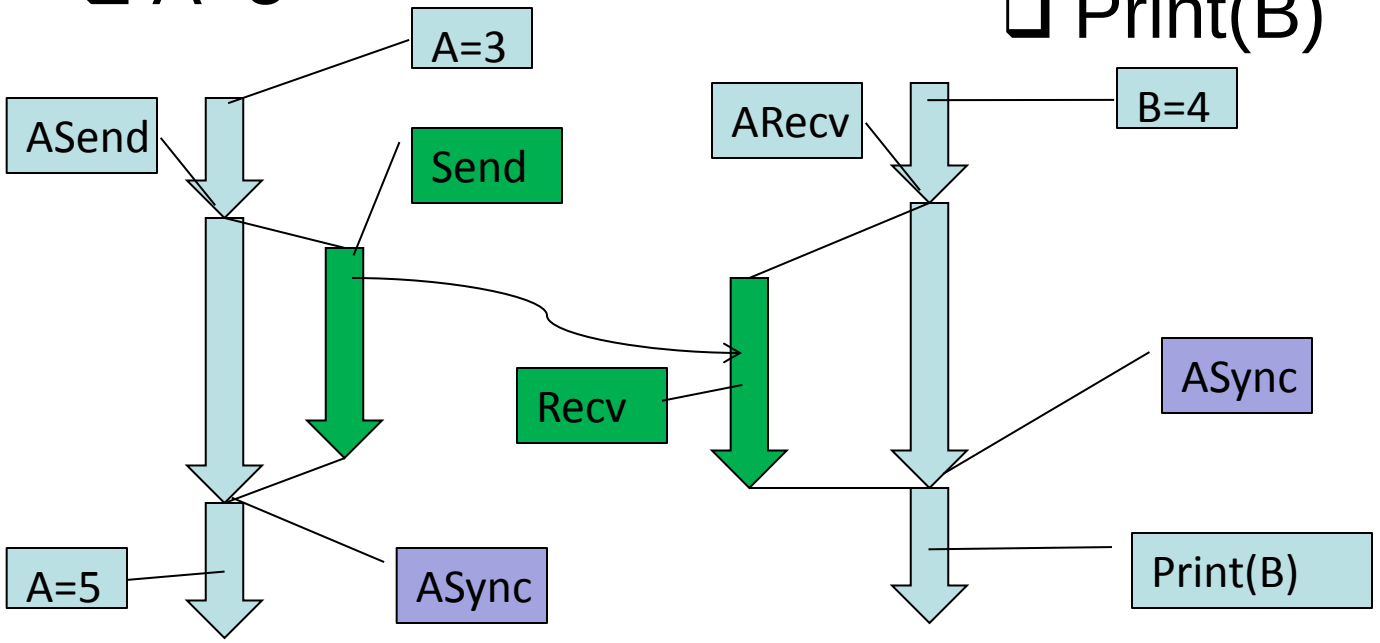
Результат

3 ? 4

Асинхронные

- A=3
- ASend(&A)
- Async()
- A=5

- B=4
- ARecv(&B)
- Async()
- Print(B)



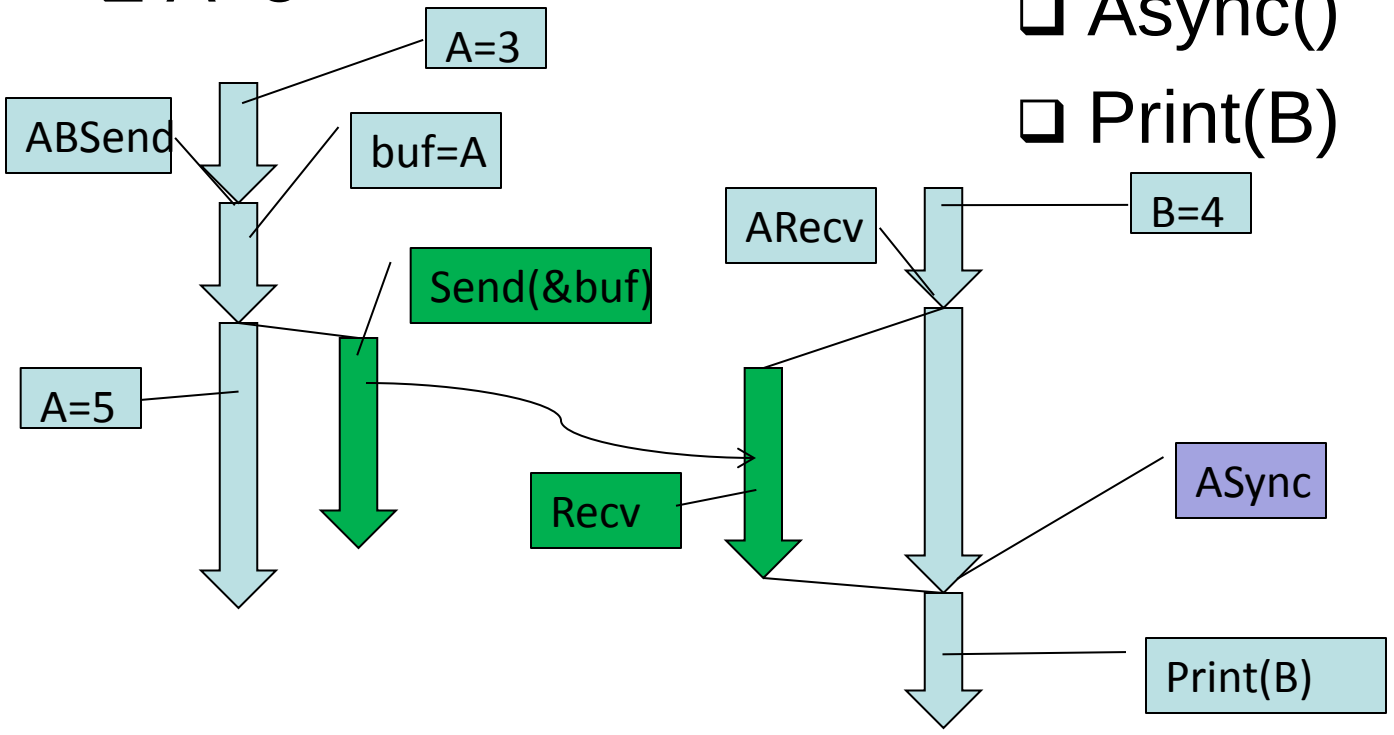
Результат

3

Асинхронные буферизованные

- A=3
- ABSend(&A)
- A=5

- B=4
- ARecv(&B)
- Async()
- Print(B)



Результат

3

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)

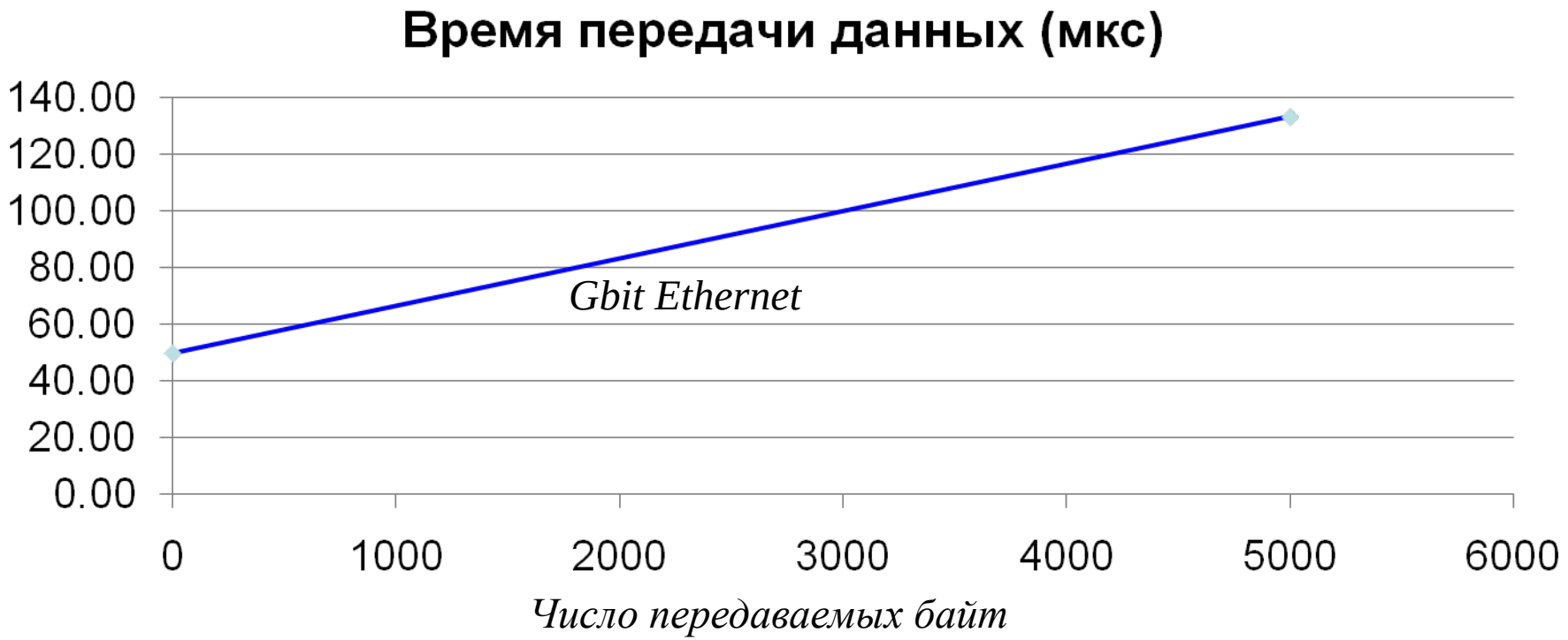
  ASync

  – Буферизованный

  ABSend(адрес данных, размер, номер процессора)

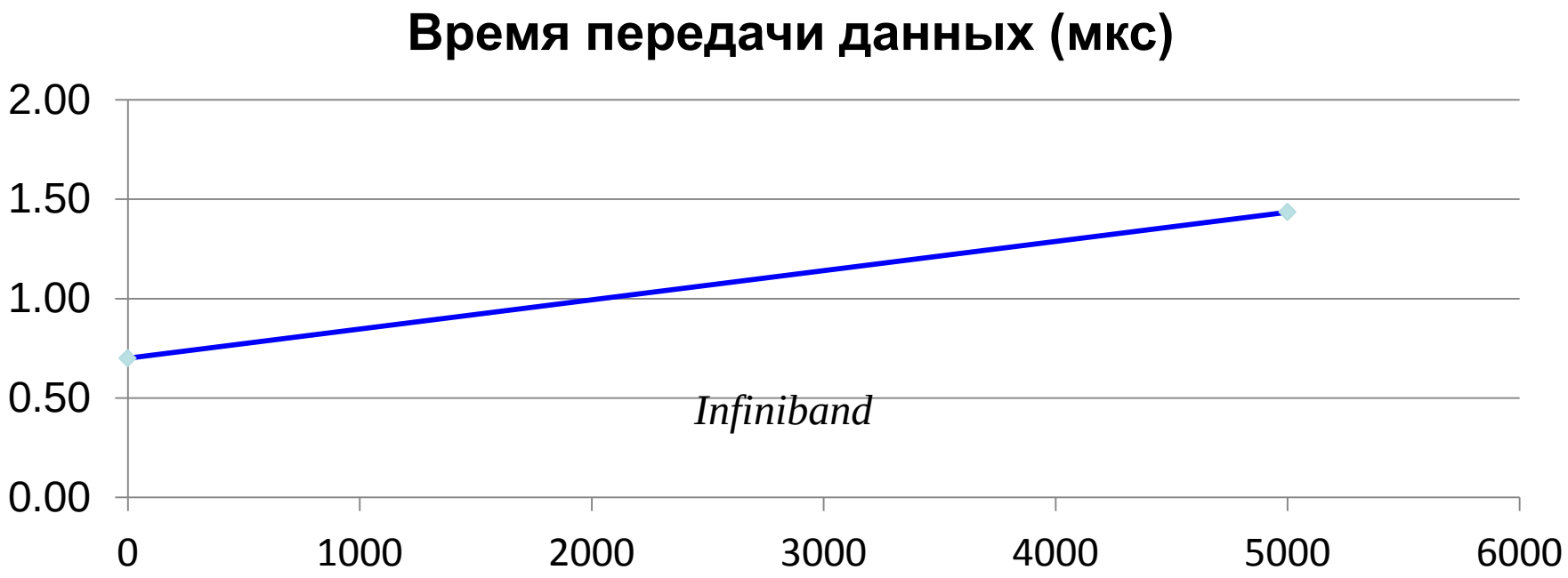
Свойства канала передачи данных

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



Свойства канала передачи данных

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



Число передаваемых байт
6.8 Гбайт/с
За 0.7 мкс передаётся 5 Кбайт

Контакты

Якобовский М.В.

чл.-корр. РАН, проф., д.ф.-м.н.

Заместитель директора по научной работе
Института прикладной математики
им. М.В. Келдыша Российской академии наук

mail: lira@imamod.ru

web: http://lira.imamod.ru