



Высокопроизводительные вычисления. Лекция 6. MPI продолжение

Рыкованов Сергей

доцент Сколковского института науки и технологий



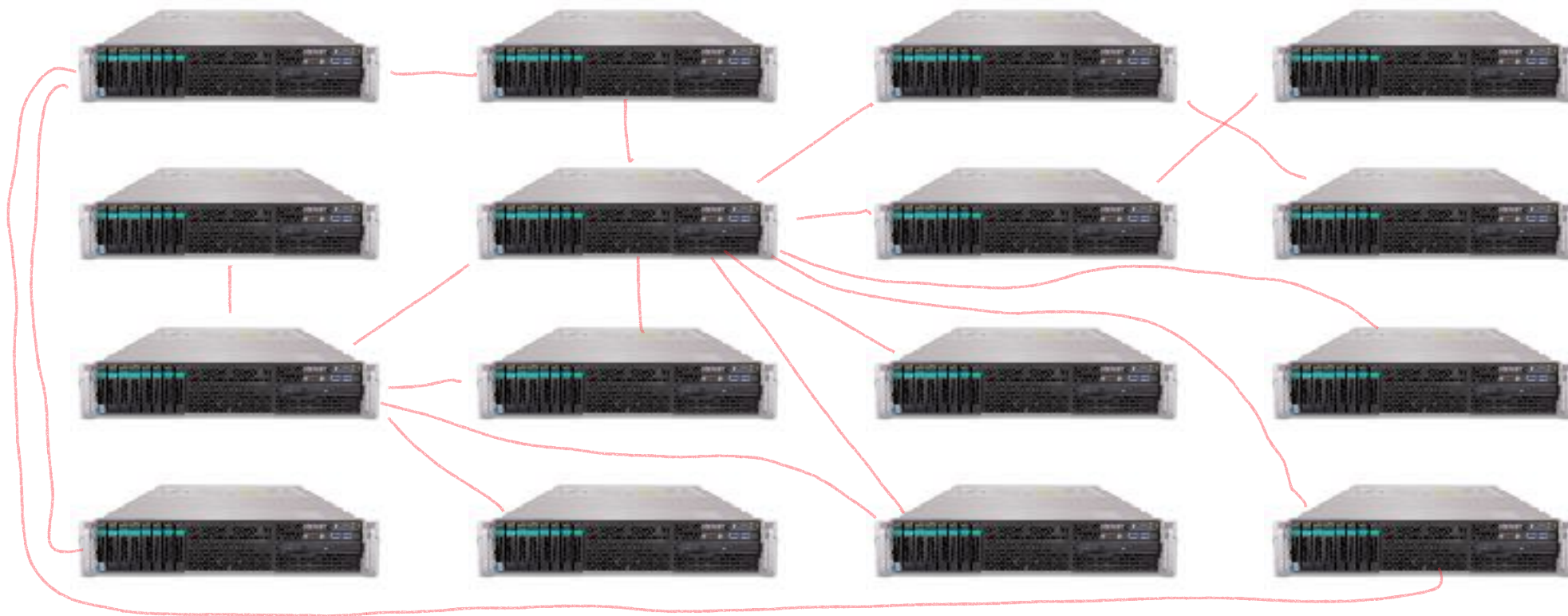
Матвеев Сергей

доцент Сколковского института науки и технологий



Системы с распределенной памятью

Не хватает мощности: покупай больше серверов



+ компьютерная сеть

9. МРІ виртуальна топология

MPI виртуальная топология

Как мы адресуемся к процессам?

Пока что одномерный массив процессов (ранги от 0 до size-1)

Иногда это очень не удобно (1е6 процессов должны разбить на куски многомерный тензор)

Декартова Топология

0 (0, 0)	1 (0, 1)	2 (0, 2)
3 (1, 0)	4 (1, 1)	5 (1, 2)

MPI виртуальная топология

```
MPI_Cart_create(MPI_Comm comm_old,  
                int ndims,  
                int *dims,  
                int *periods,  
                int reorder,  
                MPI_Comm *comm_2D)
```

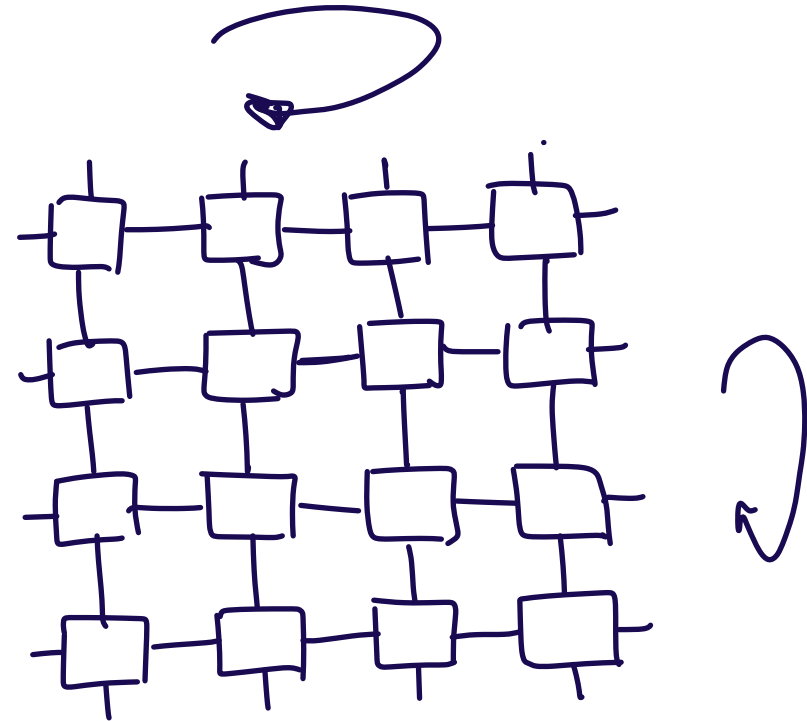
← Попросить MPI оптимизировать под топологию сети

MPI виртуальная топология

```
MPI_Cart_create(MPI_Comm comm_old,  
                int ndims,  
                int *dims,  
                int *periods,  
                int reorder,  
                MPI_Comm *comm_2D)
```

```
MPI_Comm comm_2D;  
int ndim = 2;  
int dims[2], periods[2];  
dims[0] = 4;  
dims[1] = 4;  
periods[0] = 1;  
periods[1] = 1;  
MPI_Cart_create(MPI_COMM_WORLD,  
                ndim, &dims, &periods,  
                0, &comm_2D);
```

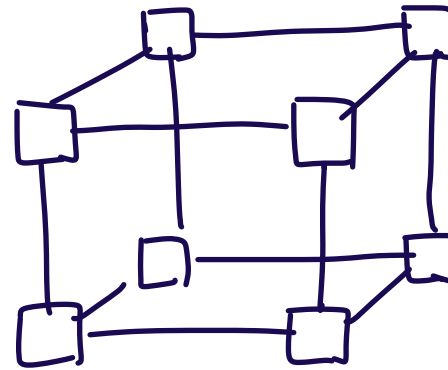
→ top



MPI виртуальная топология

```
MPI_Cart_create(MPI_Comm comm_old,  
                int ndims,  
                int *dims,  
                int *periods,  
                int reorder,  
                MPI_Comm *comm_2D)
```

```
MPI_Comm comm_3D;  
int ndim=3;  
int dims[3], periods[3];  
dims[0]=2; dims[1]=2; dims[2]=0;  
periods[0]=0; periods[1]=0; periods[2]=0;  
MPI_Cart_create(MPI_COMM_WORLD, ndim, &dims, &periods, 0, &comm_3D);
```



к у δ



MPI виртуальная топология. Кольцо

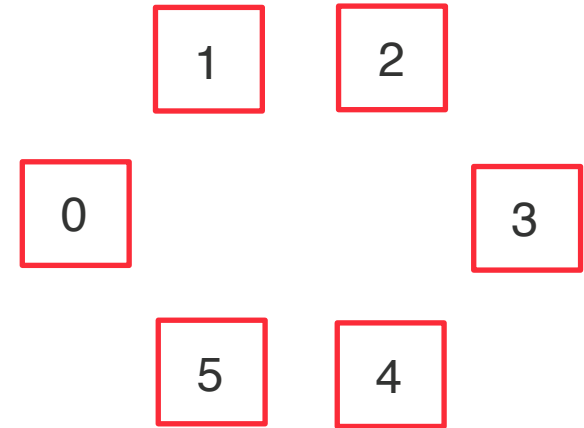


MPI виртуальная топология. Кольцо

```
MPI_Comm ring_1D;  
int ndim = 1;  
int dims, periods;  
dims = 6;  
periods = 1;  
  
MPI_Cart_create(MPI_COMM_WORLD,  
                ndim, &dims, &periods, 0, &ring_1D);
```

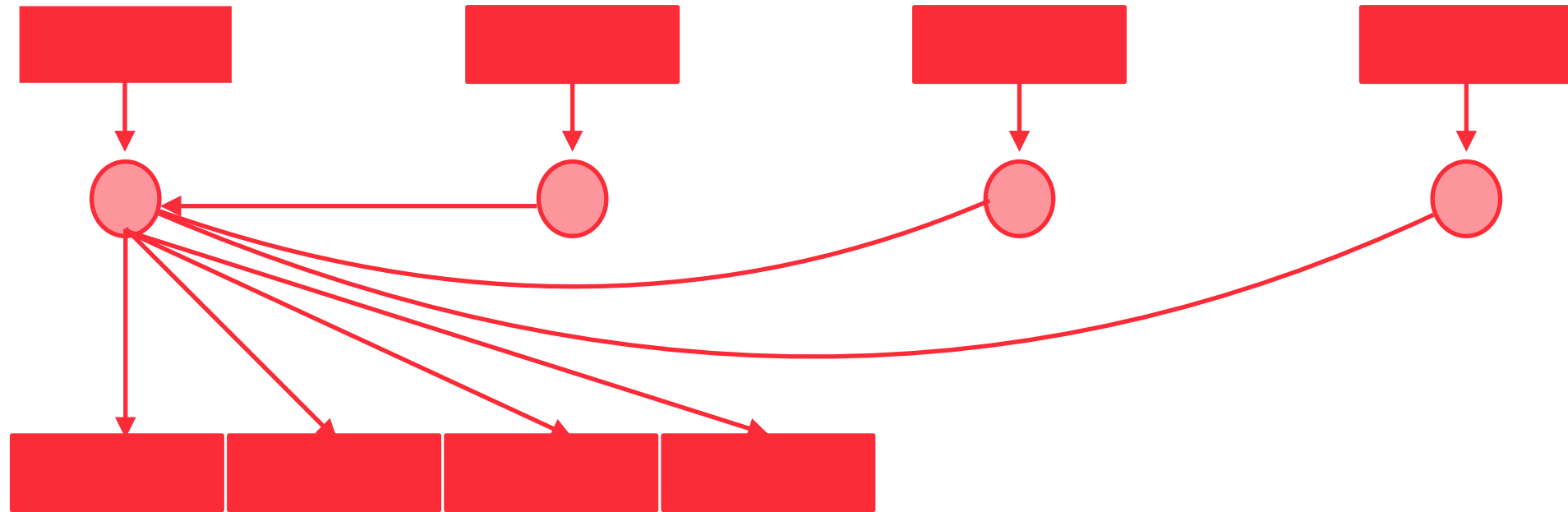
Вспомогательные функции:

```
MPI_Cartdim_get(ring_1D, &ndims);  
MPI_Cart_get(ring_1D, ndims, &dims, &periods, &coords);  
MPI_Comm_rank(ring_1D, &my_ring_rank);  
MPI_Cart_rank(ring_1D, coords, &rank);  
MPI_Cart_coords(ring_1D, rank, &coords);  
MPI_Cart_shift(ring_1D, direction, shift, &source, &dest);
```



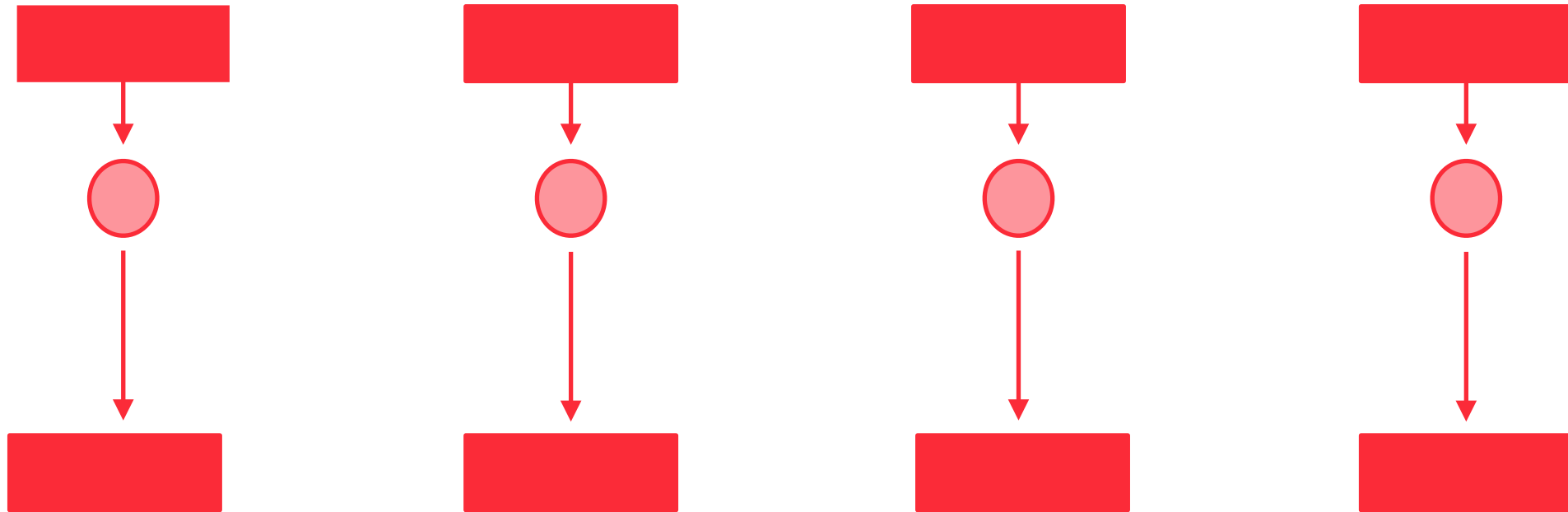
10. MPI средства ввода/ вывода

MPI не параллельная запись (POSIX)



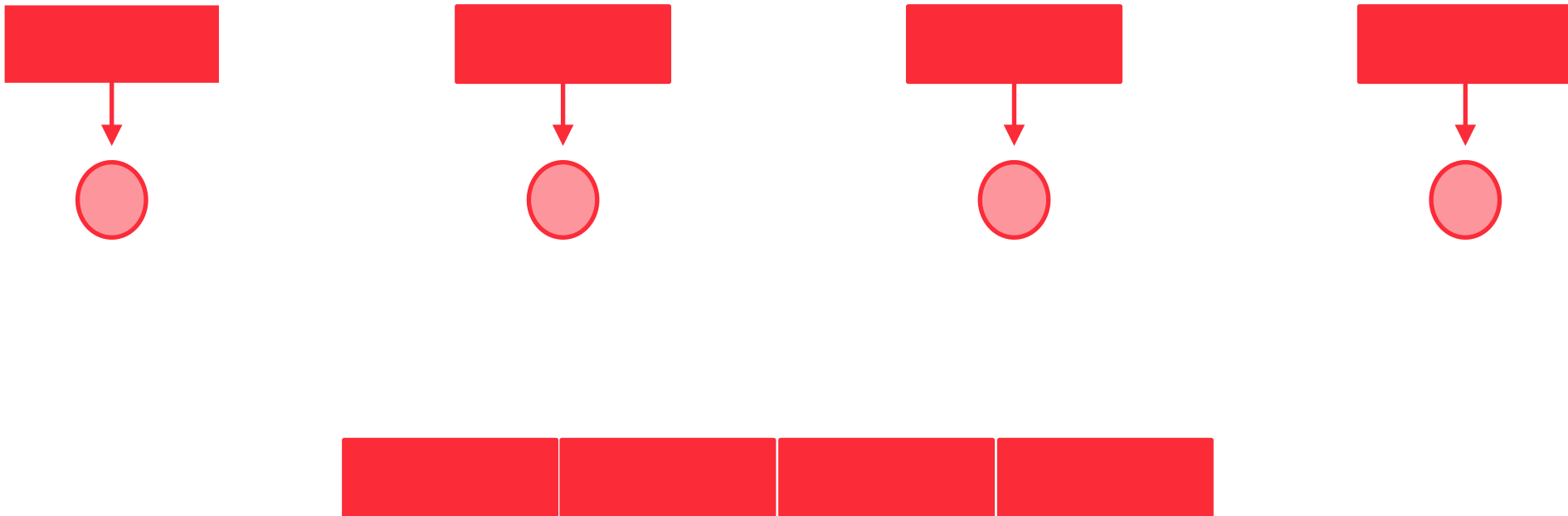
- (-) не параллельная запись
- (-) скорость ниже, чем серийная версия
- (+) один файл, довольно простая имплементация

MPI параллельная запись 1 (POSIX)



- (+) параллельная запись
- (+) быстрее, чем серийная версия
- (-) много файлов, которые потом надо обрабатывать

MPI параллельная запись 2 (MPI IO)



- (+) параллельная запись
- (+) быстрее, чем серийная версия
- (-) надо выучить синтаксис (но это не сложно)

Программный уровень: параллельное чтение и запись в/из файла, библиотека для I/O
Системный уровень: параллельная файловая система (софт и хард)



MPI IO

В POSIX I/O вам надо:

1. Открыть файл
2. Найти место с которого читать/писать (если необходимо)
3. Считать/записать
4. Заккрыть файл

В MPI I/O практически также:

1. Открыть файл: `MPI_File_open`
2. Найти место: `MPI_File_seek` и другие
3. Считать/записать: `MPI_File_read` и `MPI_File_write`
4. Заккрыть файл: `MPI_File_close`



MPI IO основные определения

File handle - структура данных для доступа к файлу

File pointer - указатель на место в файле, откуда читать/куда писать:

- Может быть индивидуальным для процесса
- Может быть общим для всех процессов

File view - часть файла, которая “видна” процессу

Collective/individual IO - обращение с файлом может быть определено пользователем (if rank==0...) или коллективно с помощью MPI

MPI IO открыть/заккрыть файл

```
MPI_File_open(comm, filename, mode, info, fhandle)
```

comm - коммуникатор, внутри которого осуществляется IO

mode - мода открытого файла (можно соединять с помощью 'I')

info - тонкие настройки MPI (размер буфера, сколько времени ждать открытия файла перед тем как выйти с ошибкой и.т.д.), настройки по-умолчанию: MPI_INFO_NULL

fhandle - file handle

```
MPI_File_close(fhandle)
```

MPI_MODE_APPEND

MPI_MODE_CREATE

MPI_MODE_DELETE_ON_CLOSE


MPI_MODE_EXCL

MPI_MODE_RDONLY

MPI_MODE_RDWR

MPI_MODE_WRONLY

другие...



MPI IO запись с помощью индивидуального указателя

```
MPI_File fh;
int buf[1024];
int prank, psize;

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &prank);

MPI_File_open(MPI_COMM_WORLD, "test.dat",
              MPI_MODE_CREATE | MPI_MODE_WRONLY, MPI_INFO_NULL, &fh);

if(prank == 1)
{
    MPI_File_write(fh, buf, 1024, MPI_INT, MPI_STATUS_IGNORE);
}
MPI_File_close(&fh);
```



MPI_File_seek

```
MPI_File_seek(fhandle, disp, whence)
```

disp - displacement in bytes

whence:

MPI_SEEK_SET: установить указатель на disp

MPI_SEEK_CUR: установить указатель на текущий указатель + disp

MPI_SEEK_END: установить указатель на конец файла + disp



Другие функции для работы с файлами

`MPI_File_seek`

`MPI_File_read`

`MPI_File_write`

`MPI_File_read_at`

`MPI_File_write_at`

`MPI_File_write_all`

`MPI_File_read_all`

11. MPI / OpenMP

Гибридный параллелизм MPI / OpenMP



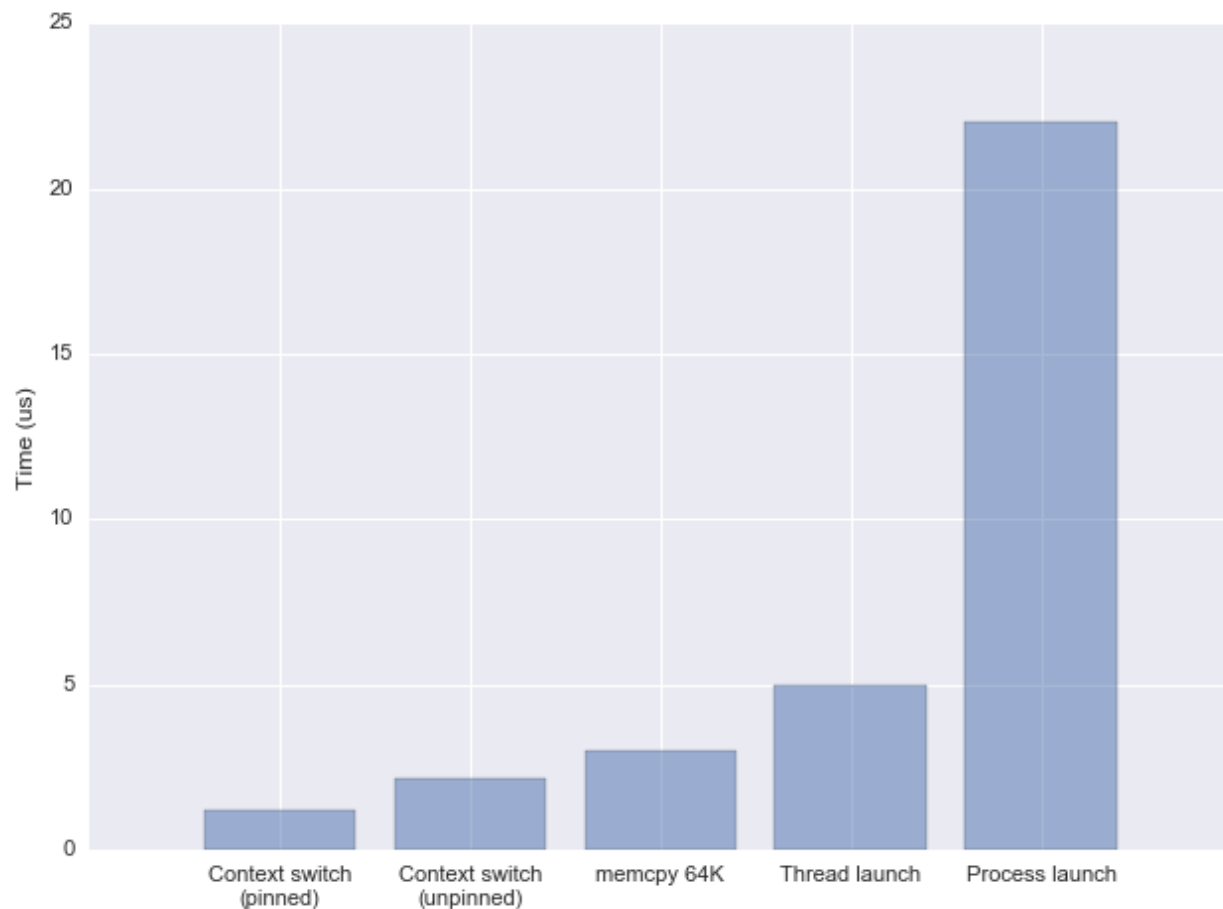
Системная шина

ОЗУ / RAM

I/O



Гибридный параллелизм MPI / OpenMP



OpenMP - нити одного процесса, разделяющие общую память:

Небольшие накладные на создание
Общая память - быстрый доступ к данным
Нет возможности общения между серверами

MPI - независимые процессы:

Большие накладные на создание
Распределенная память
Общение через передачу сообщений
На shared memory - ненужный посредник

Гибридный параллелизм MPI / OpenMP

<http://tiny.cc/u8xy4y>

```
#include <mpi.h>
#include <omp.h>

MPI_Init_thread(&argc, &argv, MPI_THREAD_MULTIPLE, &required);
MPI_Comm_rank(MPI_COMM_WORLD, &prank);
MPI_Comm_size(MPI_COMM_WORLD, &psize);
#pragma omp parallel
{
    int tid = omp_get_thread_num();
    printf("Hello from thread[%d] from process[%d]\n", tid, prank);
}
```

mpicc -fopenmp test.c -o test

12. MPI в Питоне / mpi4py



Mpi4py

Что такое mpi4py:

- Интерфейс MPI для Python
- Основан на MPI-2
- Поддерживает почти все функции MPI
- Поддерживает передачу сериализуемых (pickleable) объектов Python
- Отличная поддержка передачи numpy-объектов

Установка:

- `pip install mpi4py`
- `pip install mpi4py -- user`
- `conda install mpi4py`

На нашем кластере mail.ru:

- `module load mpich-3.3.2`
- `module load python-3.7.1`

На нашем кластере mail.ru доп библиотеки:

- `module load mpich-3.3.2`
- `module load python-3.7.1`
- `pip install libraryname --user`



Mpi4py простейший пример

```
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

print("Hello from rank = ", rank)
```

```
mpirun -n 4 python hello.py
```

Mpi4py коммуникация точка-точка

“send” и “recv” по смыслу такие же как и в C/C++:

- `comm.send(obj, dest, tag=0)`
- `comm.recv(source=MPI.ANY_SOURCE, tag=MPI.ANY_TAG, status=None)`
- “tag” - номер сообщения
- “dest” ранк в коммуникаторе
- “source” либо ранк в коммуникаторе, либо “wildcard” `MPI.ANY_SOURCE`
- блокирующие операции

```
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

if rank == 0:
    msg = 'Hello, world'
    comm.send(msg, dest = 1)
elif rank == 1:
    s = comm.recv()
    print("rank %d: %s" % (rank, s))
```



Мрі4ру коммунікація точка-точка

“send” и “recv” по смыслу такие же как и в C/C++:

- `comm.send(obj, dest, tag=0)`
- `comm.recv(source=MPI.ANY_SOURCE, tag=MPI.ANY_TAG, status=None)`
- “tag” - номер сообщения
- “dest” ранк в коммунікаторе
- “source” либо ранк в коммунікаторе, либо “wildcard” `MPI.ANY_SOURCE`
- блокирующие операции

Если buffer object (например, массив numpy), то:

```
comm.Send([array, count, datatype], dest=0, tag=0)
comm.Recv([array, count, datatype], src=0, tag=0)
```

Мрі4ру коммунікація точка-точка

“send” и “recv” по смыслу такие же как и в C/C++:

- `comm.send(obj, dest, tag=0)`
- `comm.recv(source=MPI_ANY_SOURCE, tag=MPI_ANY_TAG, status=None)`

Аналогично неблокирующие коммуникации:
`comm.isend` и `comm.irecv`

На заметку:

- Для стандартных Python objects используйте `send`, `recv` с прописной буквы!!!
- Для buffer objects (например, numpy arrays) используйте `Send`, `Recv` с заглавной буквы!!!

```
if rank == 0:
    msg = 'Hello, world'
    comm.send(msg, dest = 1)
elif rank == 1:
    s = comm.recv()
    print("rank %d: %s" % (rank, s))
```



Мpi4py коллективные операции

```
comm.bcast(obj, root = 0)  
comm.Bcast(numpyarray, root = 0)
```

```
comm.scatter(obj, root = 0)  
comm.gather(obj, root = 0)
```

```
comm.allgather(obj)
```

```
comm.reduce(obj, op=MPI.SUM, root = 0)
```



академия
больших
данных

made

mail.ru
group

Пояснения к шаблону

В вашем распоряжении есть следующие слайды:

- Титульный слайд
- Слайд №1.1 Стандартный
- Слайд №1.2 Стандартный с подзаголовком
- Слайд №2.1 Разворот справа
- Слайд №2.2 Разворот слева
- Слайд №3.1 Ноутбук справа
- Слайд №3.2 Ноутбук слева
- Слайд №4.1 2 столбца
- Слайд №4.2 2 столбца
- Слайд №5 Контакты
- Слайд №6.1 Отбивка вертикальная
- Слайд №6.2 Пустой слайд

Для акцентов в коде и тексте на слайдах в настройках цвета у вас есть готовая палитра:



HEX #fc2c38
RGB 252, 43, 56
Pantone 032C
CMYK 0, 97, 75, 0



HEX #e5e8e8
RGB 230, 231, 232
Pantone Cool Gray 1C
CMYK 0, 0, 0, 9




HEX #75787b
RGB 117, 120, 123
Pantone Cool Gray 9C
CMYK 0, 0, 0, 73



HEX #030303
RGB 3, 3, 3
Pantone Black
CMYK 75, 67, 67, 89

Слайды всех типов находятся в шаблоне, их можно выбрать из выпадающего меню в окне «Создать слайд»



Используйте для
отбивки между
блоками фотографии
из папки

Текст

Иконки и элементы

Для создания ориентиров на слайде используйте иконки из готового набора или подходящие по цветовой гамме:

