

Лекция №6
по курсу
«Проектирование и архитектура вычислительных
систем»

Москва, 2020

MPI

<https://www.mpich.org//static/downloads/1.4.1p1/>

<https://www.microsoft.com/en-us/download/confirmation.aspx?id=36043>

<https://www.microsoft.com/en-us/download/details.aspx?id=12218>

<https://www.microsoft.com/en-us/download/confirmation.aspx?id=57467>

Этапы

Этапы параллельного программирования:

- определение параллелизма: анализ задачи с целью выделить подзадачи, которые могут выполняться одновременно;
- организация параллелизма: изменение структуры задачи таким образом, чтобы можно было эффективно выполнять подзадачи;
- реализация параллелизма: реализация параллельного алгоритма в исходном коде с помощью системы обозначений параллельного программирования.

Реализация параллелизма

1. Параллельные программные пакеты, созданные для решения задач из какой-то области науки
2. Параллельные библиотеки, в которых уже реализован определенный набор алгоритмов и они выполняются параллельно
 - Math Kernel Library (Intel® MKL)
 - PARMETIS и др.
3. Инструменты, которые позволяют нам организовывать параллелизм явно в программах

OpenMp, MPI, Cuda (Nvidea)

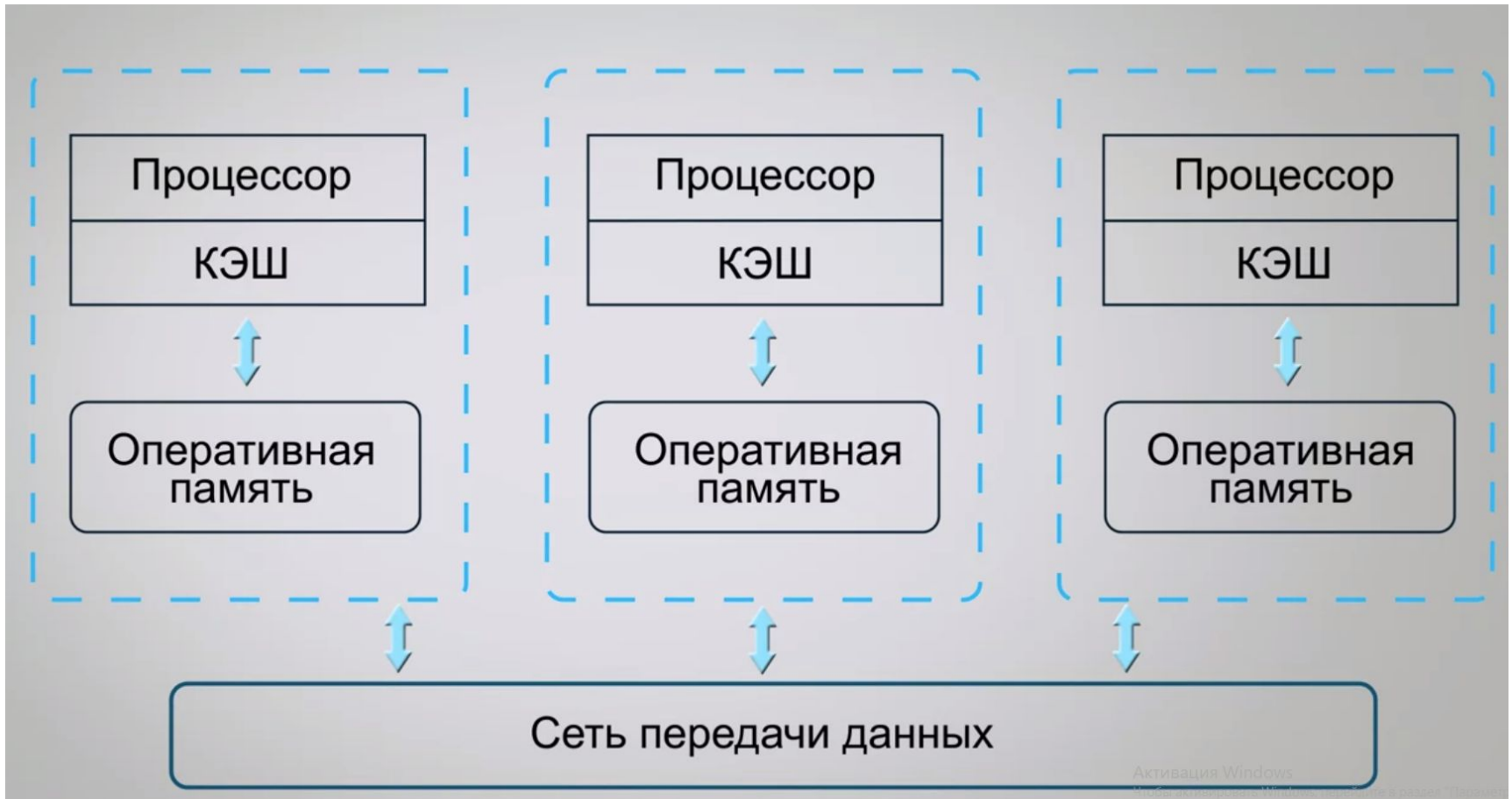
MPI

MPI (Message Passing Interface) — это программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу.

Компьютер с общей памятью

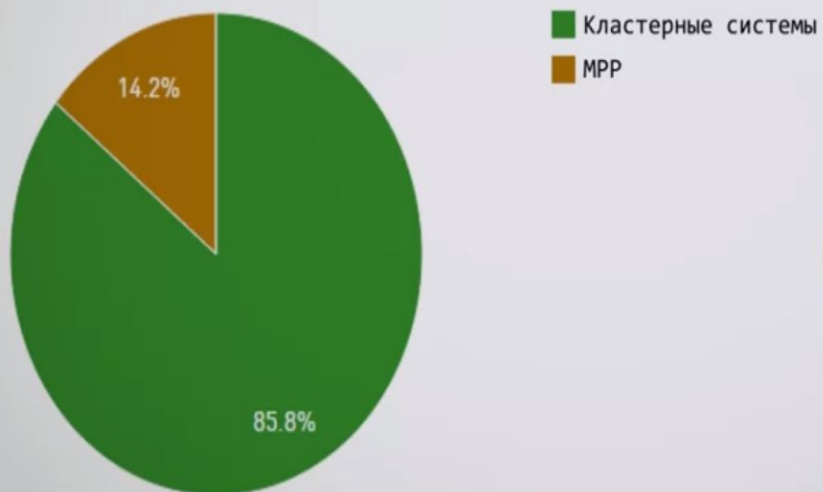


Компьютер с распределенной памятью

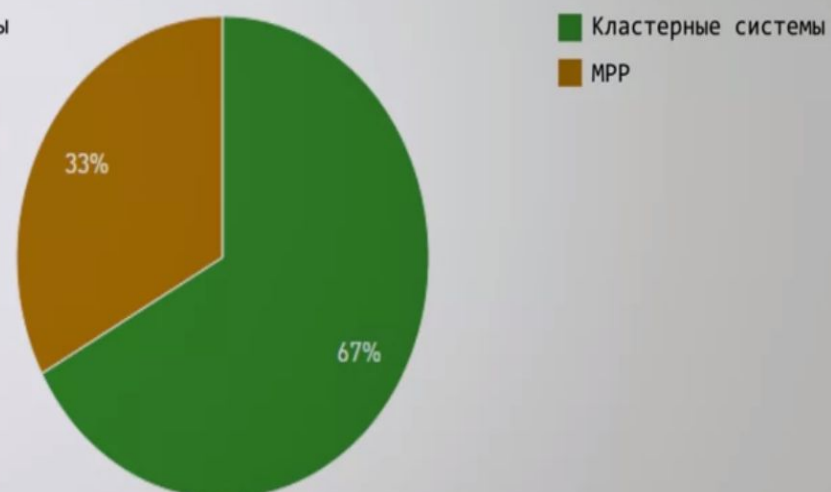


Доля кластерных систем

Распределение количества систем



Распределение суммарной производительности систем



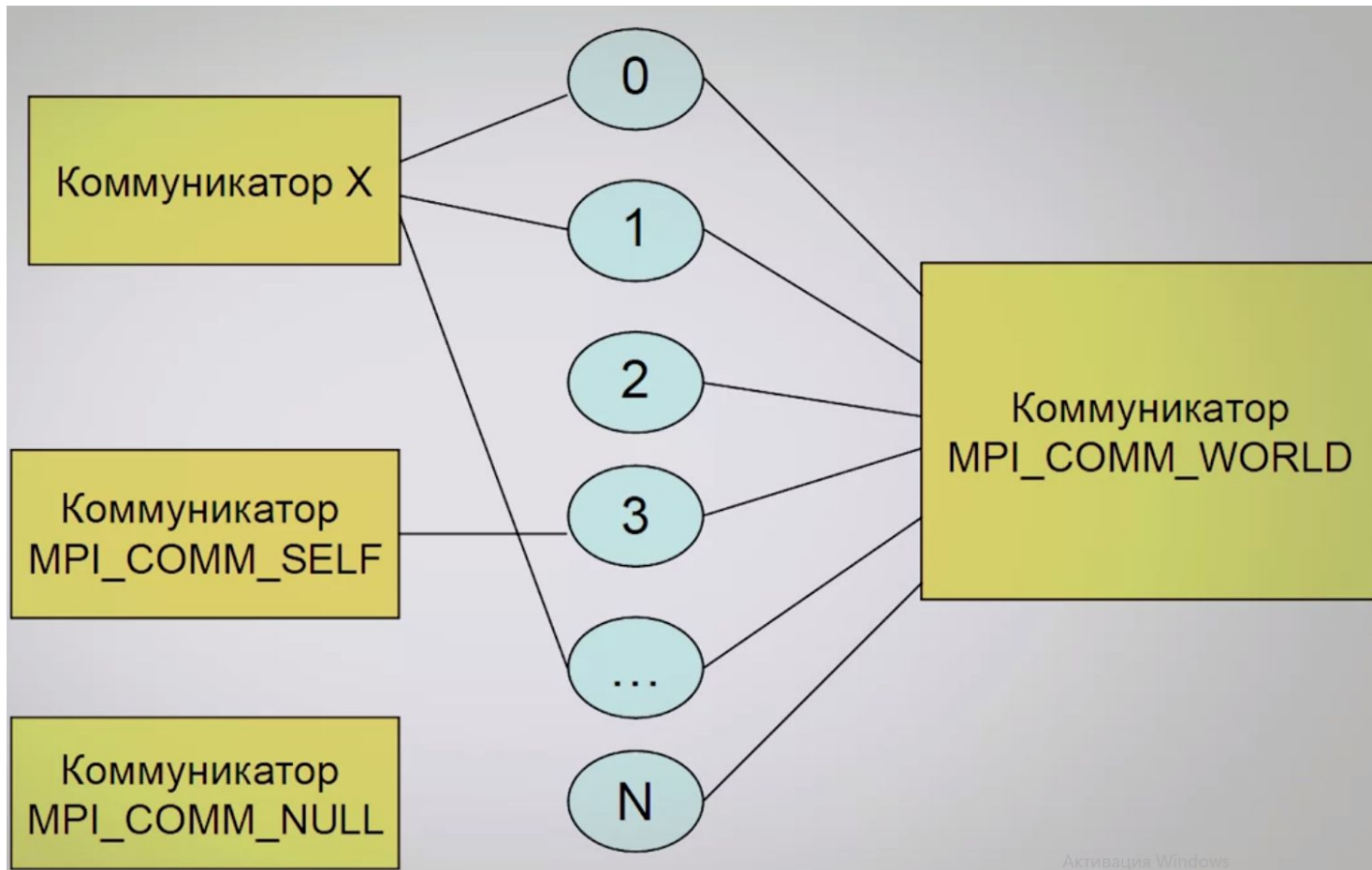
MPI

MPI – это стандарт передачи сообщений, в котором описаны функции для реализации обмена сообщениями, и параметры от которых эти функции вызываются.

Примеры реализации MPI

- MPICH - самая распространённая бесплатная реализация, разработанная в Аргоннской национальной лаборатории
- LAM/MPI - бесплатная реализация для гетерогенных кластеров из UNIX-машин (в настоящее время OpenMPI)

Параллельные процессы



Закон Амдала

MPI поддерживается:

- на классических MPP-системах;
- на кластерах;
- в сетях рабочих станций;
- на SMP-системах.

Префиксы MPI_

все имена

процедур

функций

констант

типов данных

имеют префикс MPI_

MPI

- Функции в C/C++:

```
MPI_Init  
MPI_Comm_size
```

MPI

MPI_SUCCESS	Ошибки нет
MPI_ERR_BUFFER	Неправильный указатель буфера
MPI_ERR_COUNT	Неверное количество аргумента
MPI_ERR_TYPE	Неправильный тип аргумента
MPI_ERR_TAG	Неправильный тэг аргумента
MPI_ERR_COMM	Неправильный коммуникатор
MPI_ERR_RANK	Неправильный номер
MPI_ERR_REQUEST	Неверный запрос (дескриптор)
MPI_ERR_ROOT	Неверный корневой идентификатор
MPI_ERR_GROUP	Неправильная группа
MPI_ERR_OP	Неправильная операция
MPI_ERR_TOPOLOGY	Неверная топология
MPI_ERR_DIMS	Неправильная размерность аргумента
MPI_ERR_ARG	Ошибка аргумента некоторого другого типа
MPI_ERR_UNKNOWN	Неизвестная ошибка
MPI_ERR_TRUNCATE	Неправильное округление
MPI_ERR_OTHER	Известная ошибка не из этого списка
MPI_ERR_INTERN	Внутренняя ошибка реализации MPI
MPI_ERR_IN_STATUS	Неправильный код статуса
MPI_ERR_PENDING	Зависший запрос

MPI

```
int MPI_Init (  
    int *argc, char ***argv);
```

Параметры вызова `argc` и `argv` соответствуют параметрам вызова самой программы.

MPI

Коммуникатор в MPI - специально создаваемый служебный объект объединяющий в своем составе группу процессоров и ряд дополнительных параметров, используемых при передаче данных.

MPI

MPI_Init

MPI_Comm_size

MPI_Comm_rank

*MPI_Comm_size - позволяет определить общее количество процессов в группе

*MPI_Comm_rank - позволяет определить номер каждого процесса

MPI

```
int MPI_Comm_size  
    (MPI_Comm comm, int *size);
```

comm - коммуникатор;

size - количество процессов в коммуникаторе.

MPI

```
int MPI_Comm_rank  
    (MPI_Comm comm, int *rank);
```

comm - коммутатор;

rank - номер процесса в коммутаторе.

rank будет иметь значение из диапазона $[0, \text{size}-1]$

MPI завершение параллельной части программы

```
int MPI_Finalize(void);
```

Сообщение в MPI

данные (пересылаемая информация)

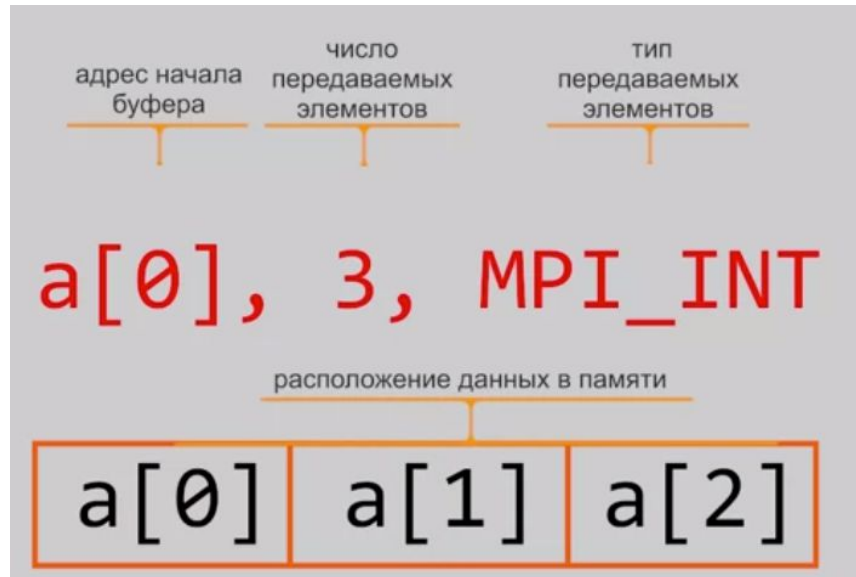
служебные данные

Буфер в вызовах MPI функций – это участок памяти, из которого сообщение должно быть послано или куда будет получено.

Параметры буфера:

- адрес начала буфера;
- число передаваемых элементов;
- тип пересылаемых данных.

MPI



Базовые типы MPI	Типы языка C
MPI_INT	int
MPI_DOUBLE	double

MPI_SEND

Для отправки сообщений
используется функция MPI_Send:

описывают буфер отправки {
определяют, кому
сообщение должно быть
отправлено {

```
int MPI_Send  
(void *buf, указывает на адрес начала буфера отправки  
int count, это количество передаваемых элементов  
MPI_Datatype type, это тип данных пересылаемого  
сообщения  
int dest,  
int tag,  
MPI_Comm comm)
```

Пример вызова функции MPI_Send

```
MPI_Send  
(a[0], 3, MPI_Int,  
1, 1, MPI_Comm_world)
```

MPI_RECEIVE

Для приема сообщений используется функция `MPI_Recv`:

описывают буфер приема

указывают, от процесса с каким номером, сообщение с каким идентификатором и в рамках какого коммуникатора должно быть принято

```
int MPI_Recv
(
  void *buf,           указывает на адрес начала буфера приема
  int count,          это количество принимаемых элементов
  MPI_Datatype type,  это тип данных принимаемого сообщения
  int source,
  int tag,
  MPI_Comm comm,
  MPI_Status *status)  это указатель на структуру данных в которую помещается информация о результате выполнения операции приема данных
```

MPI_RECEIVE

MPI_ANY_SOURCE

MPI_ANY_TAG

MPI_RECEIVE

```
int main(int argc, char **argv) {  
    MPI_Init(&argc, &argv);  
    double Time_work = MPI_Wtime();  
    //Вычислительная часть  
    Time_work =  
        MPI_Wtime() - Time_work;  
    MPI_Finalize();  
}
```