



Нижегородский государственный университет
им. Н.И.Лобачевского

Факультет Вычислительной математики и кибернетики

Образовательный комплекс

Введение в методы параллельного программирования

Раздел 4 1.

**Параллельное программирование на
основе MPI**



Гергель В.П., профессор, д.т.н.
Кафедра математического
обеспечения ЭВМ

Что означает MPI?

- ❑ MPI - это стандарт, которому должны удовлетворять средства организации передачи сообщений.
- ❑ MPI – это программные средства, которые обеспечивают возможность передачи сообщений и при этом соответствуют всем требованиям стандарта MPI:
 - программные средства должны быть организованы в виде библиотек программных модулей (*библиотеки MPI*),
 - должны быть доступны для наиболее широко используемых алгоритмических языков C и Fortran.



Достоинства MPI

- ❑ MPI позволяет существенно снизить остроту проблемы переносимости параллельных программ между разными компьютерными системами.
- ❑ MPI содействует повышению эффективности параллельных вычислений - практически для каждого типа вычислительных систем существуют реализации библиотек MPI.
- ❑ MPI уменьшает сложность разработки параллельных программ:
 - большая часть основных операций передачи данных предусматривается стандартом MPI,
 - имеется большое количество библиотек параллельных методов, созданных с использованием MPI.



MPI: основные понятия и определения...

В основу MPI положены четыре основные концепции:

- ☐ Тип операции передачи сообщения
- ☐ Тип данных, пересылаемых в сообщении
- ☐ Понятие коммуникатора (*группы процессов*)
- ☐ Понятие виртуальной топологии



MPI: основные понятия и определения...

Операции передачи данных

- ❑ Основу MPI составляют операции передачи сообщений.
- ❑ Среди предусмотренных в составе MPI функций различаются:
 - парные (*point-to-point*) операции между двумя процессами,
 - коллективные (*collective*) коммуникационные действия для одновременного взаимодействия нескольких процессов.



MPI: основные понятия и определения...

Понятие коммутаторов...

- ❑ *Коммутатор* в MPI - специально создаваемый служебный объект, объединяющий в своем составе группу процессов и ряд дополнительных параметров (*контекст*):
 - парные операции передачи данных выполняются для процессов, принадлежащих одному и тому же коммутатору,
 - Коллективные операции применяются одновременно для всех процессов коммутатора.
- ❑ Указание используемого коммутатора является обязательным для операций передачи данных в MPI.



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

□ Инициализация и завершение MPI программ

- *Первой вызываемой функцией MPI должна быть функция:*

```
int MPI_Init ( int *agrc, char ***argv )
```

(служит для инициализации среды выполнения MPI программы; параметрами функции являются количество аргументов в командной строке и текст самой командной строки.)

- *Последней вызываемой функцией MPI обязательно должна являться функция:*

```
int MPI_Finalize (void)
```



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

- Инициализация и завершение MPI программ
 - структура параллельной программы, разработанная с использованием MPI, должна иметь следующий вид:

```
#include "mpi.h"
int main ( int argc, char *argv[] ) {
    <программный код без использования MPI функций>
    MPI_Init ( &argc, &argv );
    <программный код с использованием MPI функций >
    MPI_Finalize( );
    <программный код без использования MPI функций >
    return 0;
}
```



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

- ❑ Определение количества и ранга процессов...
 - Определение *количества процессов* в выполняемой параллельной программе осуществляется при помощи функции:

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

- Для определения *ранга процесса* используется функция:

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



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

- ❑ Определение количества и ранга процессов...
 - Как правило, вызов функций *MPI_Comm_size* и *MPI_Comm_rank* выполняется сразу после *MPI_Init*:

```
#include "mpi.h"
int main ( int argc, char *argv[] ) {
    int ProcNum, ProcRank;
    <программный код без использования MPI функций>
    MPI_Init ( &argc, &argv );
    MPI_Comm_size ( MPI_COMM_WORLD, &ProcNum);
    MPI_Comm_rank ( MPI_COMM_WORLD, &ProcRank);
    <программный код с использованием MPI функций >
    MPI_Finalize();
    <программный код без использования MPI функций >
    return 0;
}
```



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

- ❑ Определение количества и ранга процессов...
 - Коммуникатор *MPI_COMM_WORLD* создается по умолчанию и представляет все процессы выполняемой параллельной программы;
 - Ранг, получаемый при помощи функции *MPI_Comm_rank*, является рангом процесса, выполнившего вызов этой функции, и, тем самым, переменная *ProcRank* будет принимать различные значения в разных процессах.



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

□ Передача сообщений...

- Для *передачи сообщения* процесс-отправитель должен выполнить функцию:

```
int MPI_Send(void *buf, int count, MPI_Datatype type,  
             int dest, int tag, MPI_Comm comm),
```

где

- **buf** – адрес буфера памяти, в котором располагаются данные отправляемого сообщения,
- **count** – количество элементов данных в сообщении,
- **type** – тип элементов данных пересылаемого сообщения,
- **dest** – ранг процесса, которому отправляется сообщение,
- **tag** – значение-тег, используемое для идентификации сообщений,
- **comm** – коммуникатор, в рамках которого выполняется передача данных.



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

- Передача сообщений...

Базовые типы данных
MPI для
алгоритмического
языка C

MPI_Datatype	C Datatype
MPI_BYTE	
MPI_CHAR	signed char
MPI_DOUBLE	Double
MPI_FLOAT	Float
MPI_INT	Int
MPI_LONG	Long
MPI_LONG_DOUBLE	long double
MPI_PACKED	
MPI_SHORT	short
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long
MPI_UNSIGNED_SHORT	unsigned short



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

□ Передача сообщений

- Отправляемое сообщение определяется через указание блока памяти (*буфера*), в котором это сообщение располагается. Используемая для указания буфера триада (*buf, count, type*) входит в состав параметров практически всех функций передачи данных,
- Процессы, между которыми выполняется передача данных, обязательно должны принадлежать коммунитатору, указываемому в функции *MPI_Send*,
- Параметр *tag* используется только при необходимости различения передаваемых сообщений, в противном случае в качестве значения параметра может быть использовано произвольное целое число.



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

□ Прием сообщений...

- Для *приема сообщения* процесс-получатель должен выполнить функцию:

```
int MPI_Recv(void *buf, int count, MPI_Datatype type,  
             int source, int tag, MPI_Comm comm, MPI_Status *status),
```

где

- **buf**, **count**, **type** – буфер памяти для приема сообщения
- **source** – ранг процесса, от которого должен быть выполнен прием сообщения,
- **tag** – тег сообщения, которое должно быть принято для процесса,
- **comm** – коммуникатор, в рамках которого выполняется передача данных,
- **status** – указатель на структуру данных с информацией о результате выполнения операции приема данных.



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

□ Прием сообщений...

- Буфер памяти должен быть достаточным для приема сообщения, а тип элементов передаваемого и принимаемого сообщения должны совпадать; при нехватке памяти часть сообщения будет потеряна и в коде завершения функции будет зафиксирована ошибка переполнения,
- При необходимости приема сообщения от любого процесса-отправителя для параметра *source* может быть указано значение *MPI_ANY_SOURCE*,
- При необходимости приема сообщения с любым тегом для параметра *tag* может быть указано значение *MPI_ANY_TAG*,



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

□ Прием сообщений...

- Параметр *status* позволяет определить ряд характеристик принятого сообщения:

```
-status.MPI_SOURCE – ранг процесса-отправителя принятого сообщения,  
-status.MPI_TAG    – тег принятого сообщения.
```

Функция

```
MPI_Get_count(MPI_Status *status, MPI_Datatype type, int *count )
```

возвращает в переменной *count* количество элементов типа *type* в принятом сообщении.



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

□ Прием сообщений

Функция *MPI_Recv* является *блокирующей* для процесса-получателя, т.е. его выполнение приостанавливается до завершения работы функции. Таким образом, если по каким-то причинам ожидаемое для приема сообщение будет отсутствовать, выполнение параллельной программы будет заблокировано.



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

- ❑ Первая параллельная программа с использованием MPI...
 - Каждый процесс определяет свой ранг, после чего действия в программе разделяются (разные процессы выполняют различные действия),
 - Все процессы, кроме процесса с рангом 0, передают значение своего ранга нулевому процессу,
 - Процесс с рангом 0 сначала печатает значение своего ранга, а далее последовательно принимает сообщения с рангами процессов и также печатает их значения,
 - Возможный вариант результатов печати процесса 0:

```
Hello from process 0  
Hello from process 2  
Hello from process 1  
Hello from process 3
```



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

- ❑ Первая параллельная программа с использованием MPI (замечания)...
 - Порядок приема сообщений заранее не определен и зависит от условий выполнения параллельной программы (более того, этот порядок может изменяться от запуска к запуску). Если это не приводит к потере эффективности, следует обеспечивать однозначность расчетов и при использовании параллельных вычислений:

```
MPI_Recv(&RecvRank, 1, MPI_INT, i, MPI_ANY_TAG,  
          MPI_COMM_WORLD, &Status)
```

Указание ранга процесса-отправителя регламентирует порядок приема сообщений.



Введение в разработку параллельных программ с использованием MPI...

Основы MPI...

- ❑ Первая параллельная программа с использованием MPI (замечания)...
 - Можно рекомендовать при увеличении объема разрабатываемых программ выносить программный код разных процессов в отдельные программные модули (функции). Общая схема MPI программы в этом случае будет иметь вид:

```
MPI_Comm_rank(MPI_COMM_WORLD, &ProcRank);  
if ( ProcRank == 0 ) DoProcess0();  
else if ( ProcRank == 1 ) DoProcess1();  
else if ( ProcRank == 2 ) DoProcess2();
```



Введение в разработку параллельных программ с использованием MPI...

Основы MPI

❑ Первая параллельная программа с использованием MPI (замечания)

- Для контроля правильности выполнения все функции MPI возвращают в качестве своего значения *код завершения*. При успешном выполнении функции возвращаемый код равен *MPI_SUCCESS*. Другие значения кода завершения свидетельствуют об обнаружении тех или иных ошибочных ситуаций в ходе выполнения функций:

- *MPI_ERR_BUFFER* – неправильный указатель на буфер,
 - *MPI_ERR_COMM* – неправильный коммуникатор,
 - *MPI_ERR_RANK* – неправильный ранг процесса
- и др.



Введение в разработку параллельных программ с использованием MPI...

Определение времени выполнения MPI программы

- Необходимо определять время выполнения вычислений для оценки достигаемого ускорения за счет использования параллелизма,
- Получение времени текущего момента выполнения программы обеспечивается при помощи функции:

```
double MPI_Wtime(void)
```

- Точность измерения времени может зависеть от среды выполнения параллельной программы. Для определения текущего значения точности может быть использована функция:

```
double MPI_Wtick(void)
```

(время в секундах между двумя последовательными показателями времени аппаратного таймера используемой системы)



Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

- Будем использовать учебную задачу суммирования элементов вектора x :

$$S = \sum_{i=1}^n x_i$$

- Для решения необходимо разделить данные на равные блоки, передать эти блоки процессам, выполнить в процессах суммирование полученных данных, собрать значения вычисленных частных сумм на одном из процессов и сложить значения частичных сумм для получения общего результата решаемой задачи,
- Для более простого изложения примера процессам программы будут передаваться весь суммируемый вектор, а не отдельные блоки этого вектора.



Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

- ❑ Передача данных от одного процесса всем процессам программы...
 - Необходимо передать значения вектора x всем процессам параллельной программы,
 - Можно воспользоваться рассмотренными ранее функциями парных операций передачи данных:

```
MPI_Comm_size(MPI_COMM_WORLD, &ProcNum) ;  
for (i=1; i<ProcNum; i++)  
    MPI_Send( &x, n, MPI_DOUBLE, i, 0, MPI_COMM_WORLD) ;
```

Повторение операций передачи приводит к суммированию затрат (латентностей) на подготовку передаваемых сообщений,

Данная операция может быть выполнена за меньшее число операций передачи данных.



Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

- ❑ Передача данных от одного процесса всем процессам программы...
 - *Широковещательная рассылка* данных может быть обеспечена при помощи функции MPI:

```
int MPI_Bcast(void *buf,int count,MPI_Datatype type,  
              int root,MPI_Comm comm),
```

где

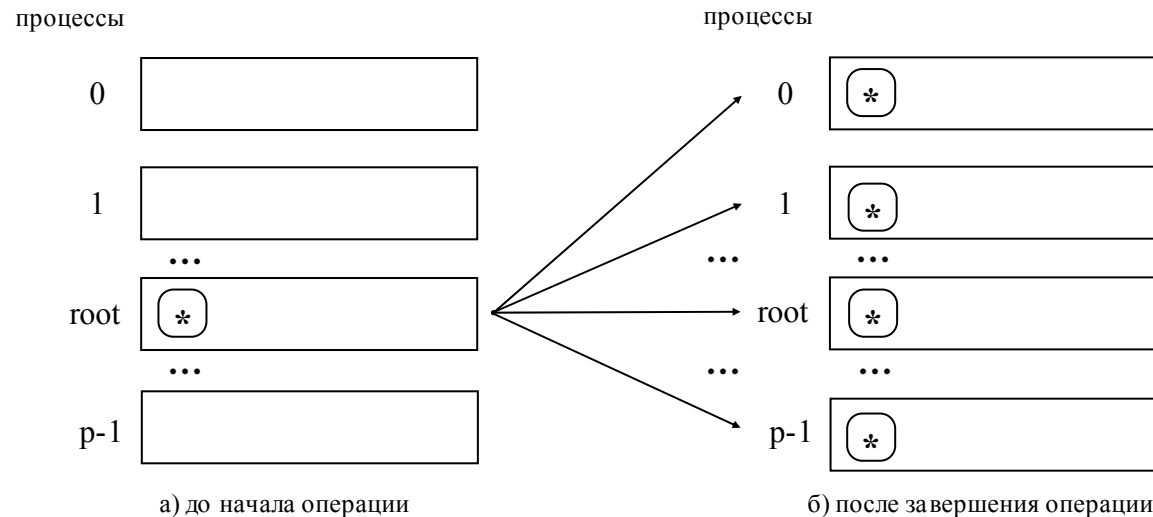
- `buf`, `count`, `type` – буфер памяти с отправляемым сообщением (для процесса с рангом 0), и для приема сообщений для всех остальных процессов,
- `root` – ранг процесса, выполняющего рассылку данных,
- `comm` – коммуникатор, в рамках которого выполняется передача данных.



Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

- ❑ Передача данных от одного процесса всем процессам программы...
 - Функция *MPI_Bcast* осуществляет рассылку данных из буфера *buf*, содержащего *count* элементов типа *type* с процесса, имеющего номер *root*, всем процессам, входящим в коммутатор *comm*



Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

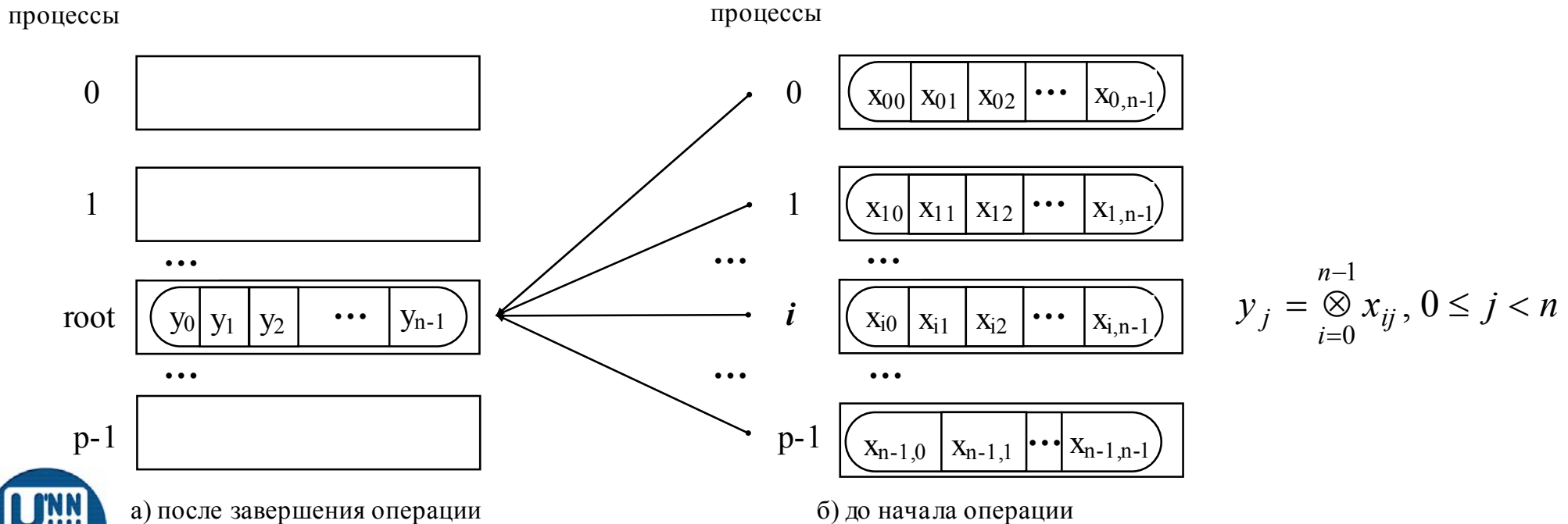
- ❑ Передача данных от одного процесса всем процессам программы
 - Функция *MPI_Bcast* определяет коллективную операцию, вызов функции *MPI_Bcast* должен быть осуществлен всеми процессами указываемого коммутатора,
 - Указываемый в функции *MPI_Bcast* буфер памяти имеет различное назначение в разных процессах:
 - Для процесса с рангом *root*, с которого осуществляется рассылка данных, в этом буфере должно находиться рассылаемое сообщение.
 - Для всех остальных процессов указываемый буфер предназначен для приема передаваемых данных.



Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

- Передача данных от всех процессов одному процессу...
 - Процедура сбора и последующего суммирования данных является примером часто выполняемой коллективной операции *передачи данных от всех процессов одному процессу*, в которой над собираемыми значениями осуществляется та или иная обработка данных.



Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

❑ Передача данных от всех процессов одному процессу...

```
int MPI_Reduce(void *sendbuf, void *recvbuf, int count,  
               MPI_Datatype type, MPI_Op op, int root, MPI_Comm comm),
```

где

- **sendbuf** – буфер памяти с отправляемым сообщением,
- **recvbuf** – буфер памяти для результирующего сообщения (только для процесса с рангом `root`),
- **count** – количество элементов в сообщениях,
- **type** – тип элементов сообщений,
- **op** – операция, которая должна быть выполнена над данными,
- **root** – ранг процесса, на котором должен быть получен результат,
- **comm** – коммуникатор, в рамках которого выполняется операция.



Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

□ Типы операций MPI для функций редукции данных...

Операция	Описание
MPI_MAX	Определение максимального значения
MPI_MIN	Определение минимального значения
MPI_SUM	Определение суммы значений
MPI_PROD	Определение произведения значений
MPI LAND	Выполнение логической операции "И" над значениями сообщений
MPI_BAND	Выполнение битовой операции "И" над значениями сообщений
MPI_LOR	Выполнение логической операции "ИЛИ" над значениями сообщений
MPI BOR	Выполнение битовой операции "ИЛИ" над значениями сообщений
MPI_LXOR	Выполнение логической операции исключающего "ИЛИ" над значениями сообщений
MPI_BXOR	Выполнение битовой операции исключающего "ИЛИ" над значениями сообщений
MPI_MAXLOC	Определение максимальных значений и их индексов
MPI_MINLOC	Определение минимальных значений и их индексов



Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

- Типы операций MPI для функций редукции данных...
 - **MPI_MAX** и **MPI_MIN** ищут поэлементные максимум и минимум;
 - **MPI_SUM** вычисляет поэлементную сумму векторов;
 - **MPI_PROD** вычисляет поэлементное произведение векторов;
 - **MPI_BAND**, **MPI_BOR**, **MPI_LOR**, **MPI_LXOR**, **MPI_BXOR** - логические и двоичные операции И, ИЛИ, исключающее ИЛИ;
 - **MPI_MAXLOC**, **MPI_MINLOC** - поиск индексированного минимума/максимума



Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

- Передача данных от всех процессов одному процессу...
 - Функция *MPI_Reduce* определяет коллективную операцию и, тем самым, вызов функции должен быть выполнен всеми процессами указываемого коммуникатора, все вызовы функции должны содержать одинаковые значения параметров *count*, *type*, *op*, *root*, *comm*,
 - Передача сообщений должна быть выполнена всеми процессами, результат операции будет получен только процессом с рангом *root*,
 - Выполнение операции редукции осуществляется над отдельными элементами передаваемых сообщений.

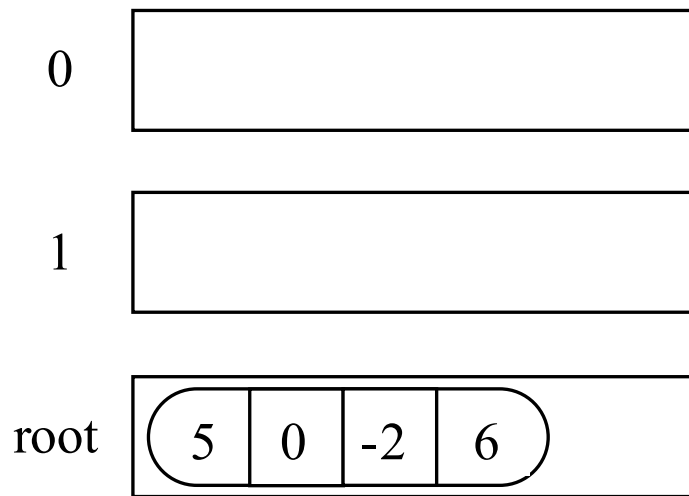


Введение в разработку параллельных программ с использованием MPI...

Начальное знакомство с коллективными операциями передачи данных...

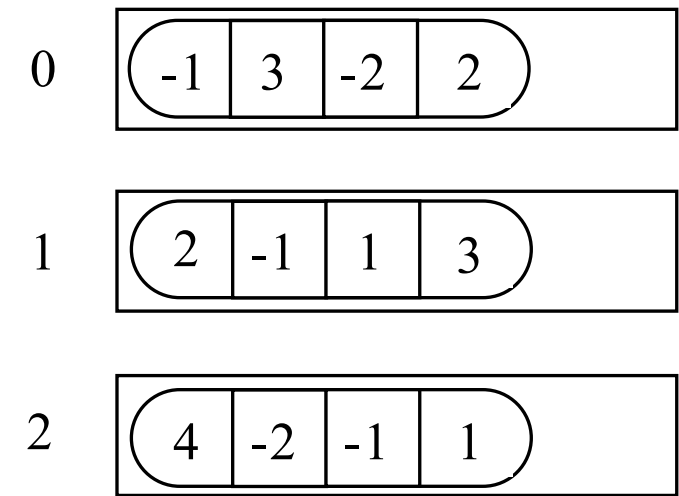
- ❑ Передача данных от всех процессов одному процессу (пример для операции суммирования)

процессы



а) после завершения операции

процессы



б) до начала операции



Введение в разработку параллельных программ с использованием MPI

Начальное знакомство с коллективными операциями передачи данных

□ Синхронизация вычислений

- *Синхронизация* процессов, т.е. одновременное достижение процессами тех или иных точек процесса вычислений, обеспечивается при помощи функции MPI:

```
int MPI_Barrier(MPI_Comm comm);
```

- Функция *MPI_Barrier* определяет коллективную операцию, при использовании должна вызываться всеми процессами коммуникатора.
- Продолжение вычислений любого процесса произойдет только после выполнения функции *MPI_Barrier* всеми процессами коммуникатора.

