MPI

Измерение времени:

```
double MPI_Wtime(void);
```

Измерение точности измерения времени:

```
double MPI_Wtick(void);
```

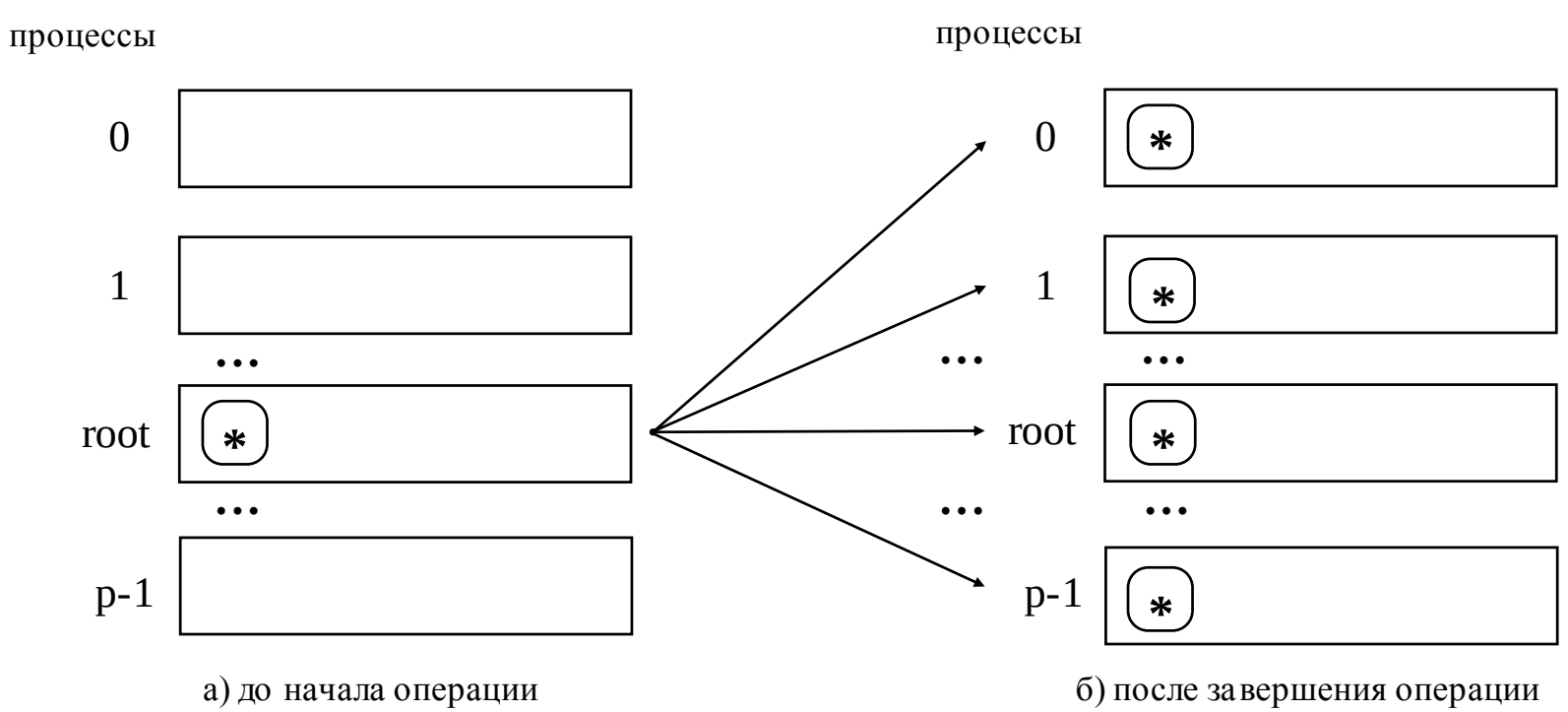
Синхронизация процессов:

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

```c
#include <stdio.h>
#include <mpi.h>

int main(int argc, char** argv)
{
  int rank = 0, size = 0;
  double start_time = 0, end_time = 0;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  MPI_Barrier(MPI_COMM_WORLD);
  if (rank == 0)
    start_time = MPI_Wtime();

  /// Алгоритм,врямя которого измеряем

  MPI_Barrier(MPI_COMM_WORLD);
  if (rank == 0)
    end_time = MPI_Wtime();

  if (rank == 0)
    printf("[TIME] %lf\n", end_time - start_time);

  MPI_Finalize();
  return 0;
}
```

MPI

Широковещательная рассылка данных




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

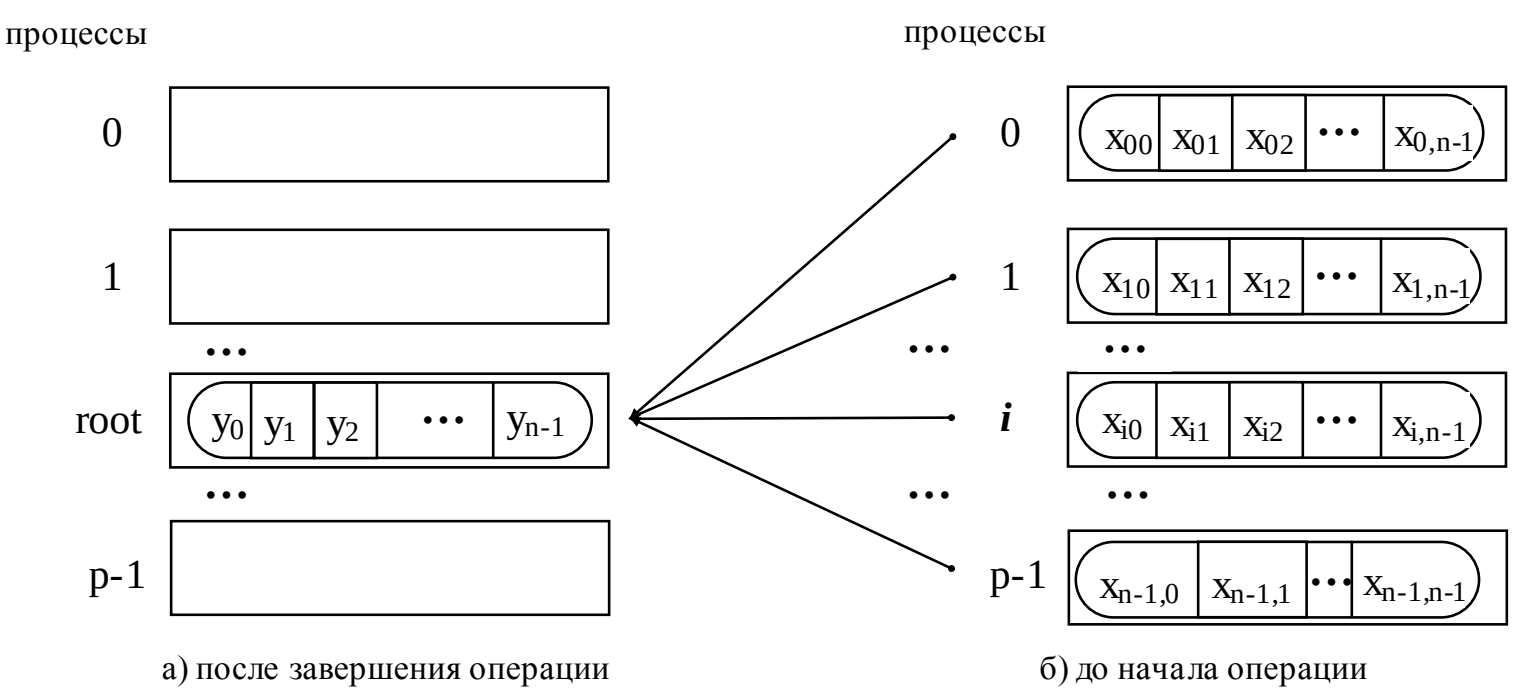
buf – буфер памяти с отправляемым сообщением (для процесса с рангом 0), и для приема сообщений для всех остальных процессов

root – ранг процесса, выполняющего рассылку данных

MPI

Сбор данных с операцией

$$y_j = \bigotimes_{i=0}^{n-1} x_{ij}, 0 \le j < n$$



а) после завершения операции

б) до начала операции

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

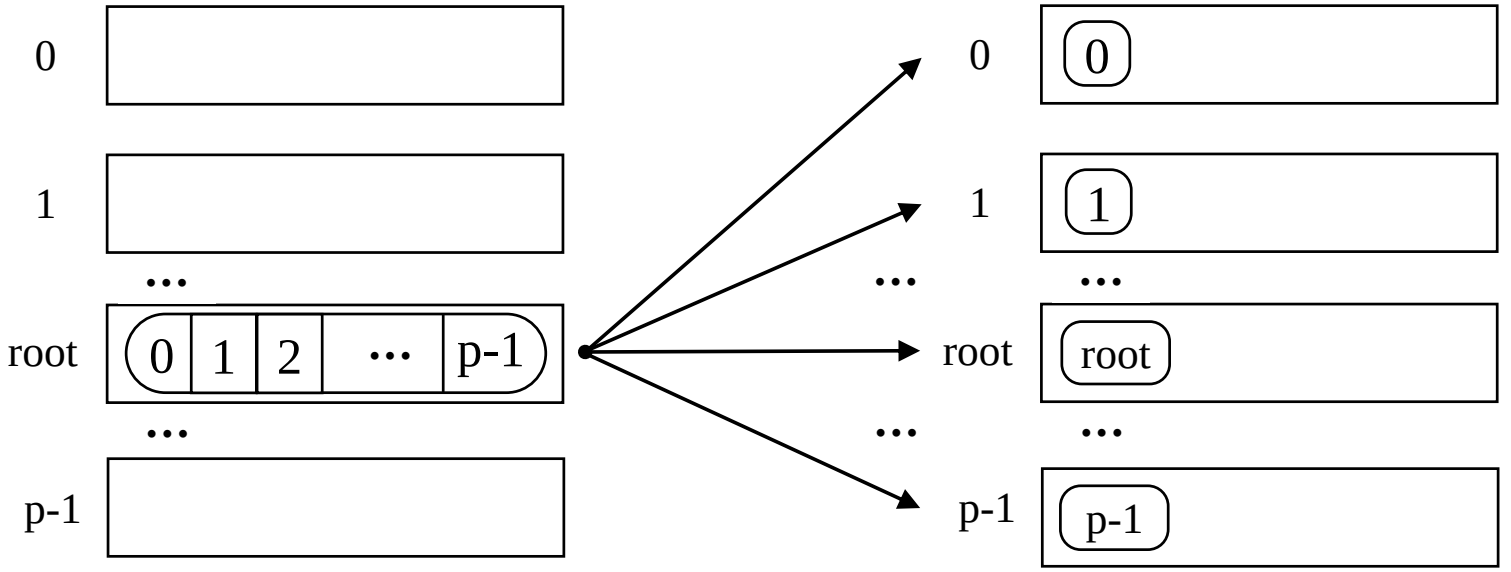
- **sendbuf** – буфер памяти с отправляемым сообщением
- **recvbuf** – буфер памяти для результирующего сообщения (только для процесса с рангом `root`)
- **op** – операция, которая должна быть выполнена над данными
- **root** – ранг процесса, на котором должен быть получен результат

MPI

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

MPI

Обобщенная передача данных от одного всем



а) до начала операции

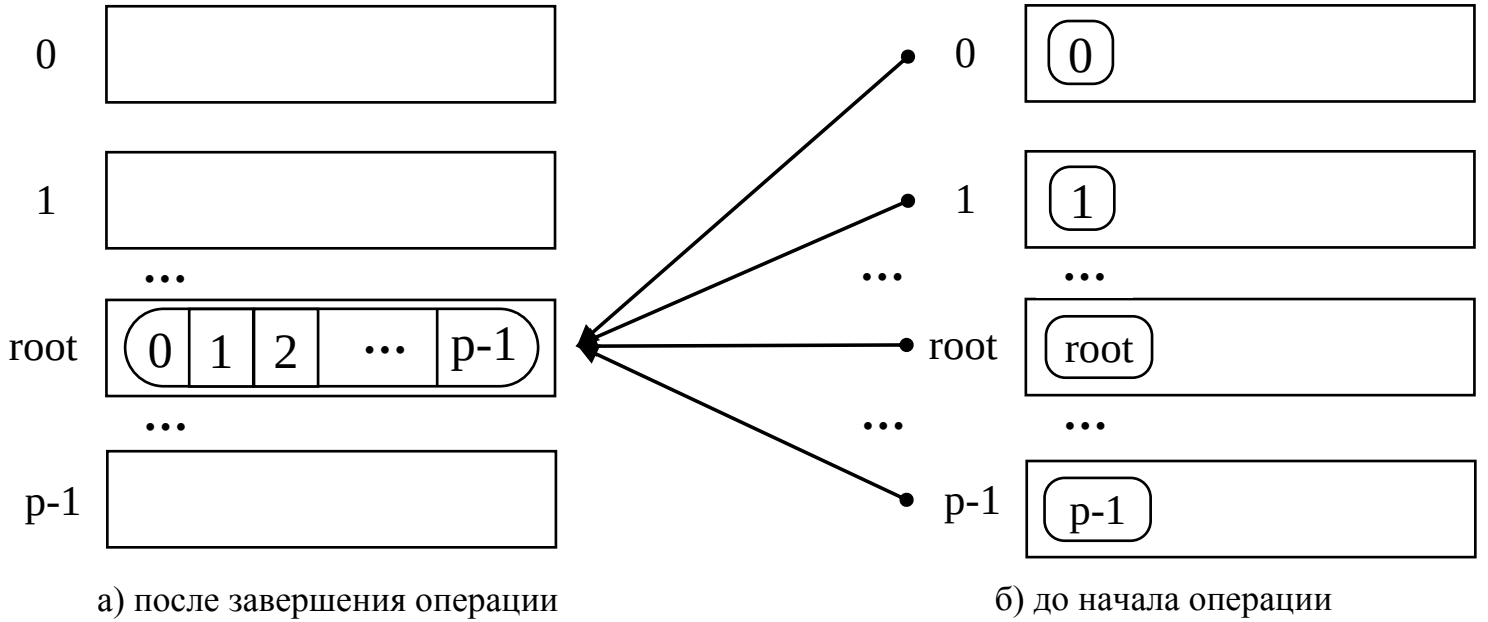
б) после завершения операции

```
int MPI_Scatter(void *sbuf,int scount,MPI_Datatype
stype, void *rbuf,int rcount,MPI_Datatype rtype, int
root, MPI_Comm comm)
```

- **sbuf, scount, stype** - параметры передаваемого сообщения (**scount** определяет количество элементов, передаваемых на каждый процесс)
- **rbuf, rcount, rtype** – параметры сообщения, принимаемого в процессах
- **root** – ранг процесса, выполняющего рассылку данных

MPI

Обобщенная передача данных от всех одному



а) после завершения операции

б) до начала операции

```
int MPI_Gather(void *sbuf,int scount,MPI_Datatype
stype, void *rbuf,int rcount,MPI_Datatype rtype, int
root, MPI_Comm comm)
```

- **sbuf**, **scount**, **stype** - параметры передаваемого сообщения,
- **rbuf**, **rcount**, **rtype** - параметры принимаемого сообщения,
- **root** – ранг процесса, выполняющего сбор данных,

MPI

MPI_All* - результат дублируется во всех процессах (MPI_Allreduce, MPI_Allgather …)

MPI_*v – размеры передаваемых процессами сообщений могут быть различны (MPI_Scatterv, MPI_Allgatherv …)

MPI

Режимы передачи данных:

MPI_Send – стандартный режим.

MPI_Ssend – синхронный режим (ждет ответ о приёме сообщения)

MPI_Rsend – режим передачи по готовности (сработает только при запущенном MPI_Recv)

MPI_Bsend – буферизированный режим (использует дополнительный буфер для отправки)

MPI

Блокирующие функции приостанавливают выполнение процессов до момента завершения работы. (MPI_*Send, MPI_Recv)

Неблокирующие функции обмена данными выполняются без блокировки. (MPI_I*send, MPI_Irecv)

```
int MPI_I*(…, MPI_Request *request);
```

Проверка состояния неблокирующей функции:

```
int MPI_Test( MPI_Request *request, int *flag,
MPI_status *status)
```

-request - дескриптор операции, определенный при вызове неблокирующей функции

-flag - результат проверки (=true, если операция завершена)

-status - результат выполнения операции обмена (только для завершенной операции)

MPI

MPI_Testall - проверка завершения всех перечисленных операций обмена

MPI_Waitall – ожидание завершения всех операций обмена

MPI_Testany - проверка завершения хотя бы одной из перечисленных операций обмена

MPI_Waitany – ожидание завершения любой из перечисленных операций обмена

MPI_Testsome - проверка завершения каждой из перечисленных операций обмена

MPI_Waitsome - ожидание завершения хотя бы одной из перечисленных операций обмена и оценка состояния по всем