

Библиотека MPI: Основные понятия и функции

Лекция 4

Основные свойства

- MPI - *Message Passing Interface*, интерфейс передачи сообщений
- Стандарт MPI 4.0
- Языки программирования:
 - FORTRAN/Matlab
 - C/C++
- Более 120 функций
- SPMD-модель параллельного программирования

Основные свойства

- Наличие групп процессов (безопасность сообщений), топологий процессов
- Структурирование передаваемого сообщения, типизация, гетерогенность
- Режимы: normal (blocking and non-blocking), synchronous, buffered
- Большое разнообразие коллективных операций

Чего нет в MPI

- Функции управления процессами
- Работа с удаленной памятью
- Разделяемая общая память
- Потoki

Основные понятия

- MPI-программа – программа запускаемая одновременно на нескольких процессорах
- Каждая копия программы выполняется как отдельный процесс

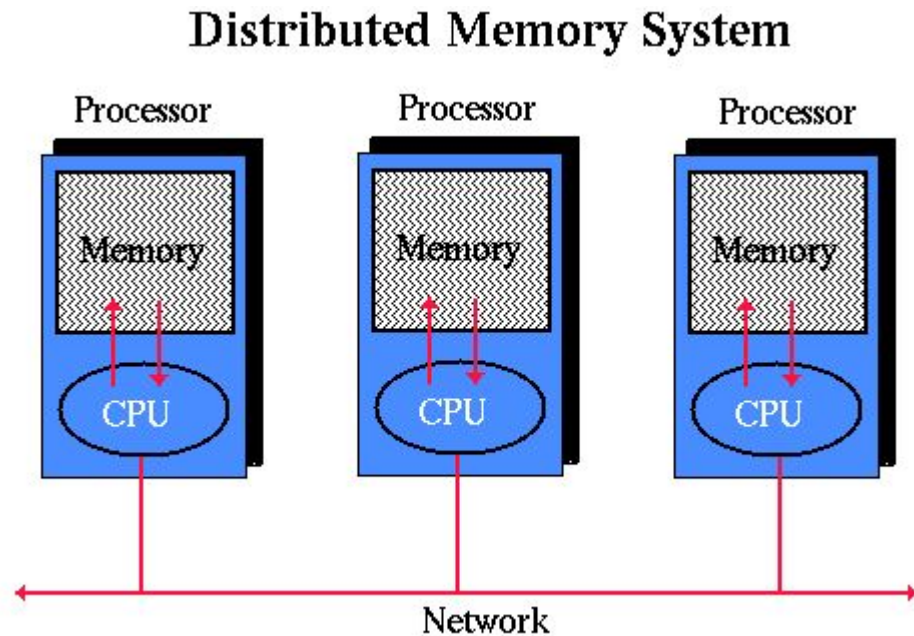


Рис. 1. Модель архитектуры

Основные понятия

- Код программы параметризуется номером процесса.
- С помощью служебных функций MPI процесс может получить информацию о количестве одновременно запущенных процессов и свой номер.

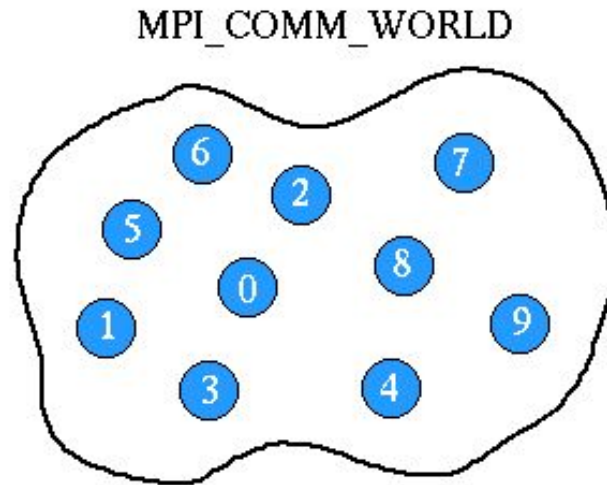


Рис. 2. Пример области связности

Основные понятия

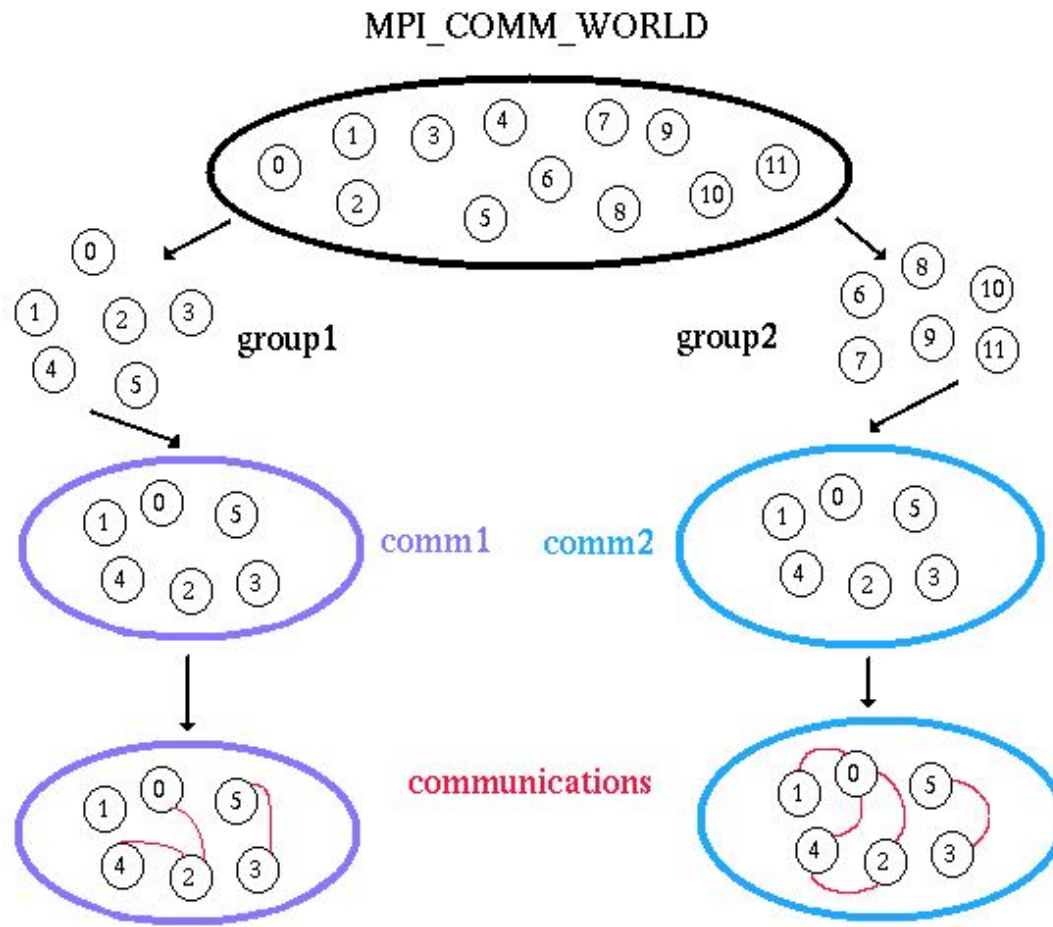


Рис. 3. Коммуникаторы и группы в MPI

Основные понятия

- Коммуникационные функции MPI предоставляют процессам MPI-программы различные способы взаимодействия: индивидуальные, групповые.

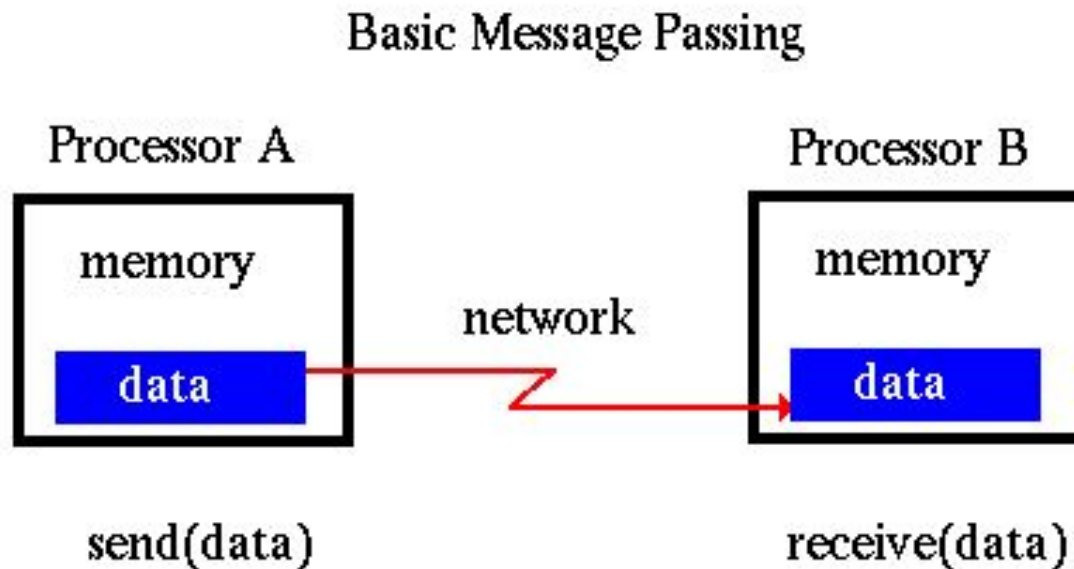


Рис. 4. Схема коммуникаций в MPI

Коммуникации точка-точка

- Блокируемые / неблокируемые
- Синхронные / асинхронные
- Буферизованные
- Пересылка по ГОТОВНОСТИ

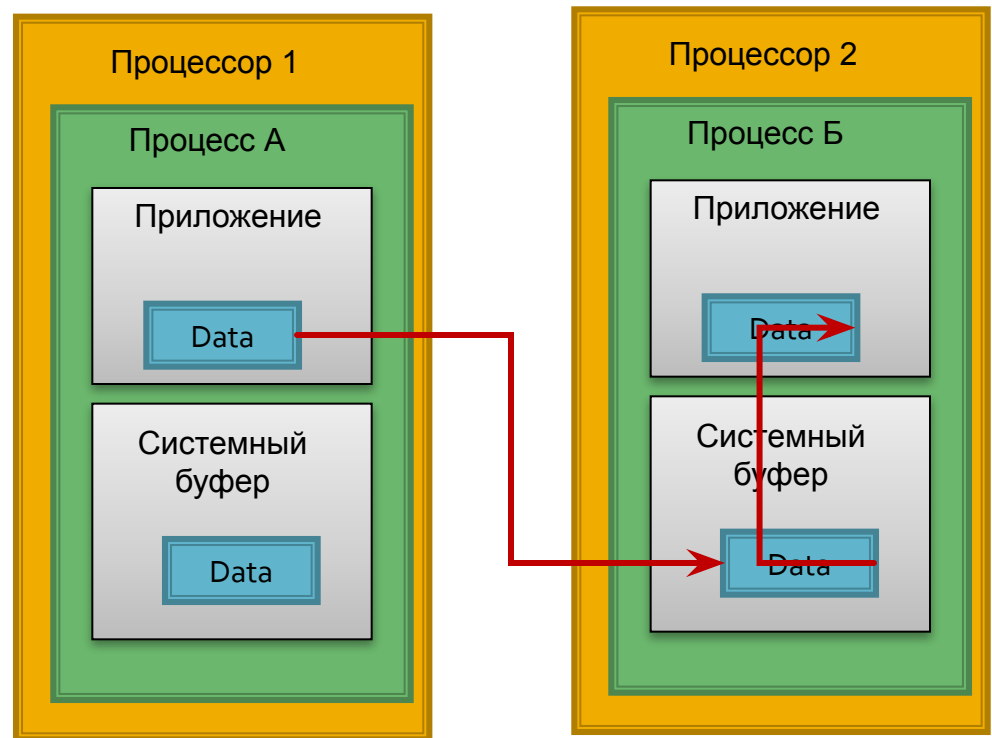


Рис. 5. Схема передачи сообщений “точка-точка” в MPI

Основные соглашения

- Регистр символов существенен в С, и не играет роли в Фортране.
- Все идентификаторы начинаются с префикса **MPI_**. Префиксы **MPID_**, **MPIR_** и **PMPI_** применяются в служебных целях.
- Имена констант записываются заглавными буквами: **MPI_COMM_WORLD**, **MPI_FLOAT**.
- В именах функций только первая за префиксом буква – заглавная: **MPI_Send**, **MPI_Comm_size**.
- Определение всех именованных констант, прототипов функций и определение типов выполняется в языке С подключением файла **mpi.h**, а в Фортране – **mpif.h**.

Соответствие между MPI-типами и типами языка Си

тип MPI	тип языка Си
<code>MPI_CHAR</code>	<code>signed char</code>
<code>MPI_SHORT</code>	<code>signed short int</code>
<code>MPI_INT</code>	<code>signed int</code>
<code>MPI_LONG</code>	<code>signed long int</code>
<code>MPI_UNSIGNED_CHAR</code>	<code>unsigned char</code>
<code>MPI_UNSIGNED_SHORT</code>	<code>unsigned short int</code>
<code>MPI_UNSIGNED</code>	<code>unsigned int</code>
<code>MPI_UNSIGNED_LONG</code>	<code>unsigned long int</code>
<code>MPI_FLOAT</code>	<code>float</code>
<code>MPI_DOUBLE</code>	<code>double</code>
<code>MPI_LONG_DOUBLE</code>	<code>long double</code>
<code>MPI_BYTE, MPI_PACKED</code>	

Первая программа

```
#include <stdio.h>
#include "mpi.h"
int main( argc, argv )
int  argc; char **argv;
{
    int rank, size;
    MPI_Init( &argc, &argv );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    printf( "Hello from process %d of %d\n",
           rank, size );
    MPI_Finalize();
    return 0;
}
```

Основные функции

- `int MPI_Init(int *argc, char ***argv)`
- `int MPI_Finalize(void)`
- `int MPI_Comm_size(MPI_Comm comm, int *size)`
- `int MPI_Comm_rank(MPI_Comm comm, int *rank)`
- `MPI_Get_processor_name (*name, *resultlength)`

Компиляция и запуск

```
% mpicc -o helloworld helloworld.c  
% mpirun -np 4 helloworld
```

```
Hello world from process 0 of 4  
Hello world from process 3 of 4  
Hello world from process 1 of 4  
Hello world from process 2 of 4
```

Передача «точка-точка»

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

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

- MPI_ANY_SOURCE
- MPI_ANY_TAG

Функция MPI_Test

```
MPI_Test(*request, *flag, *status)
```

```
MPI_Testany (count, *array_of_requests,  
             *index, *flag, *status)
```

```
MPI_Testall (count, *array_of_requests,  
            *flag, *array_of_statuses)
```

```
MPI_Testsome (incount, *array_of_requests,  
             *outcount, *array_of_offsets,  
             *array_of_statuses)
```


Функция MPI_Wait

`MPI_Wait (*request, *status)`

`MPI_Waitany (count, *array_of_requests, *index,
*status)`

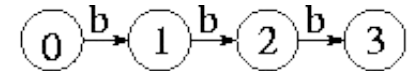
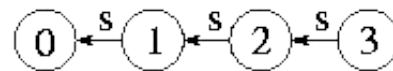
`MPI_Waitall (count, *array_of_requests,
*array_of_statuses)`

`MPI_Waitsome (incount, *array_of_requests,
*outcount,
*array_of_offsets,
*array_of_statuses)`

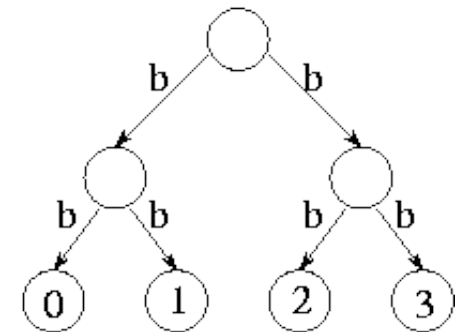
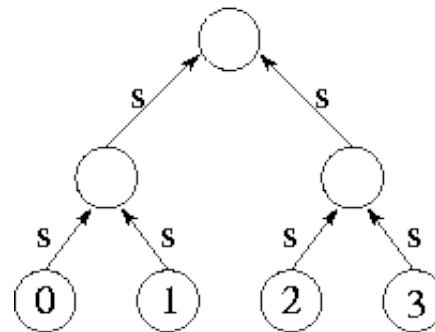
Редукция с последующей общей рассылкой

Варианты:

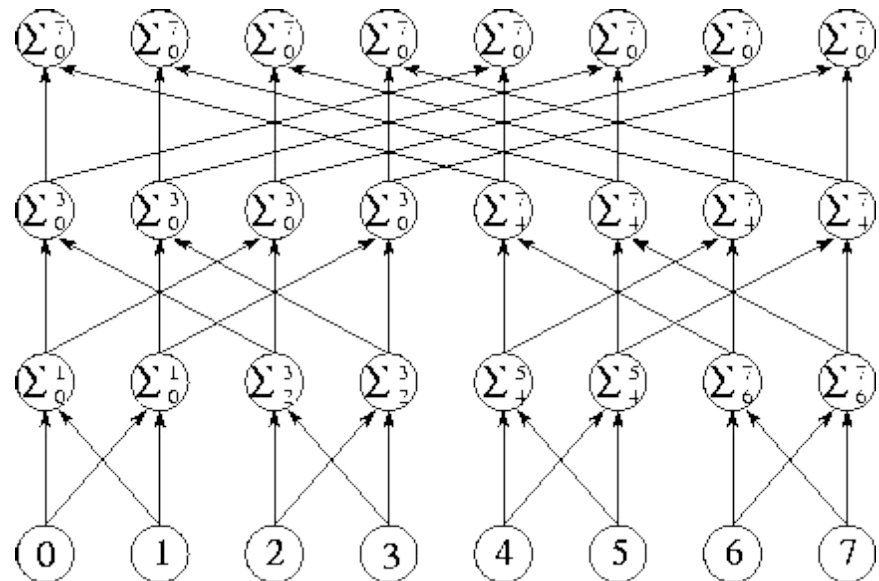
1



2



3



Количество
процессов = 8

Последовательный вариант редукции

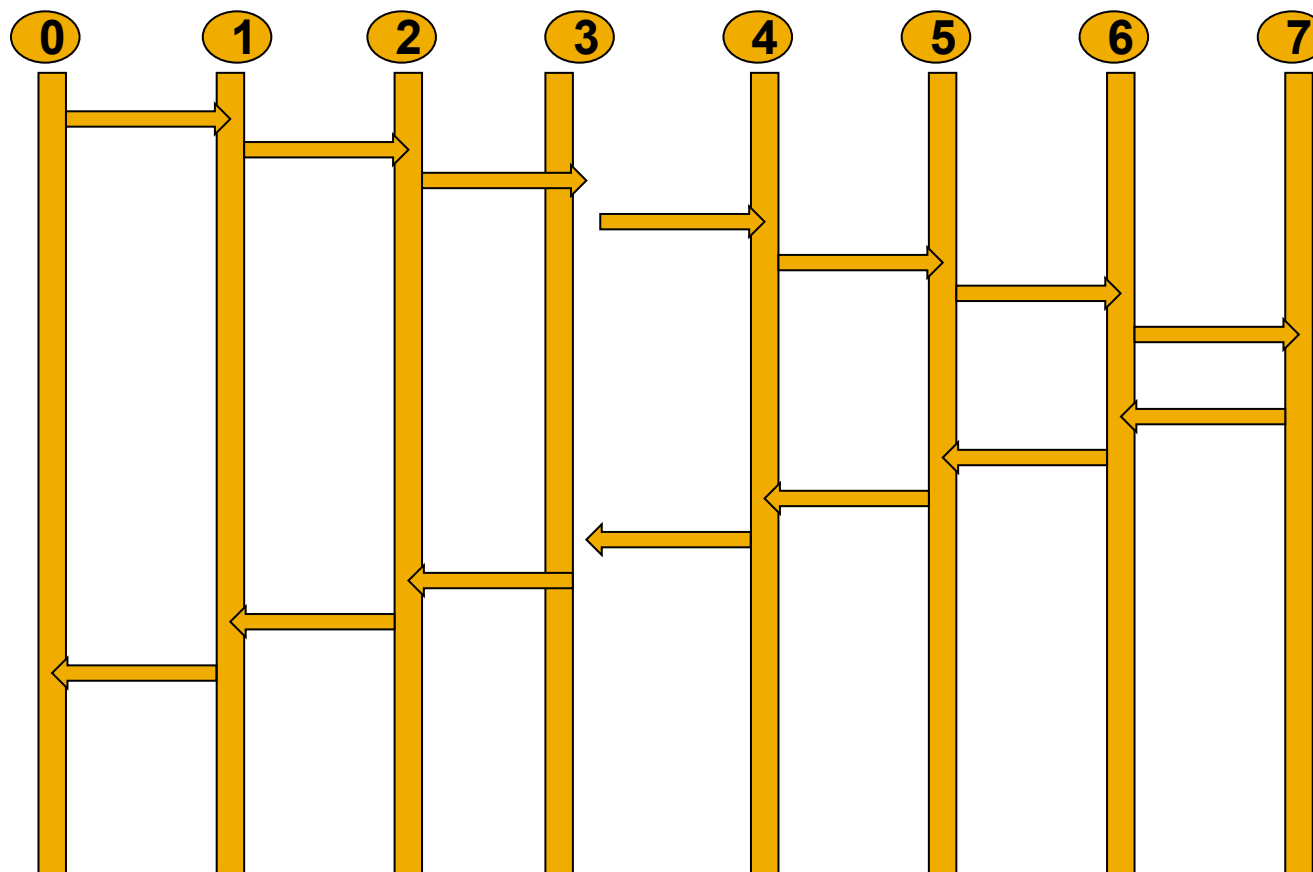


Рис. 6. Схема редукции

Вариант редукции: пирамида

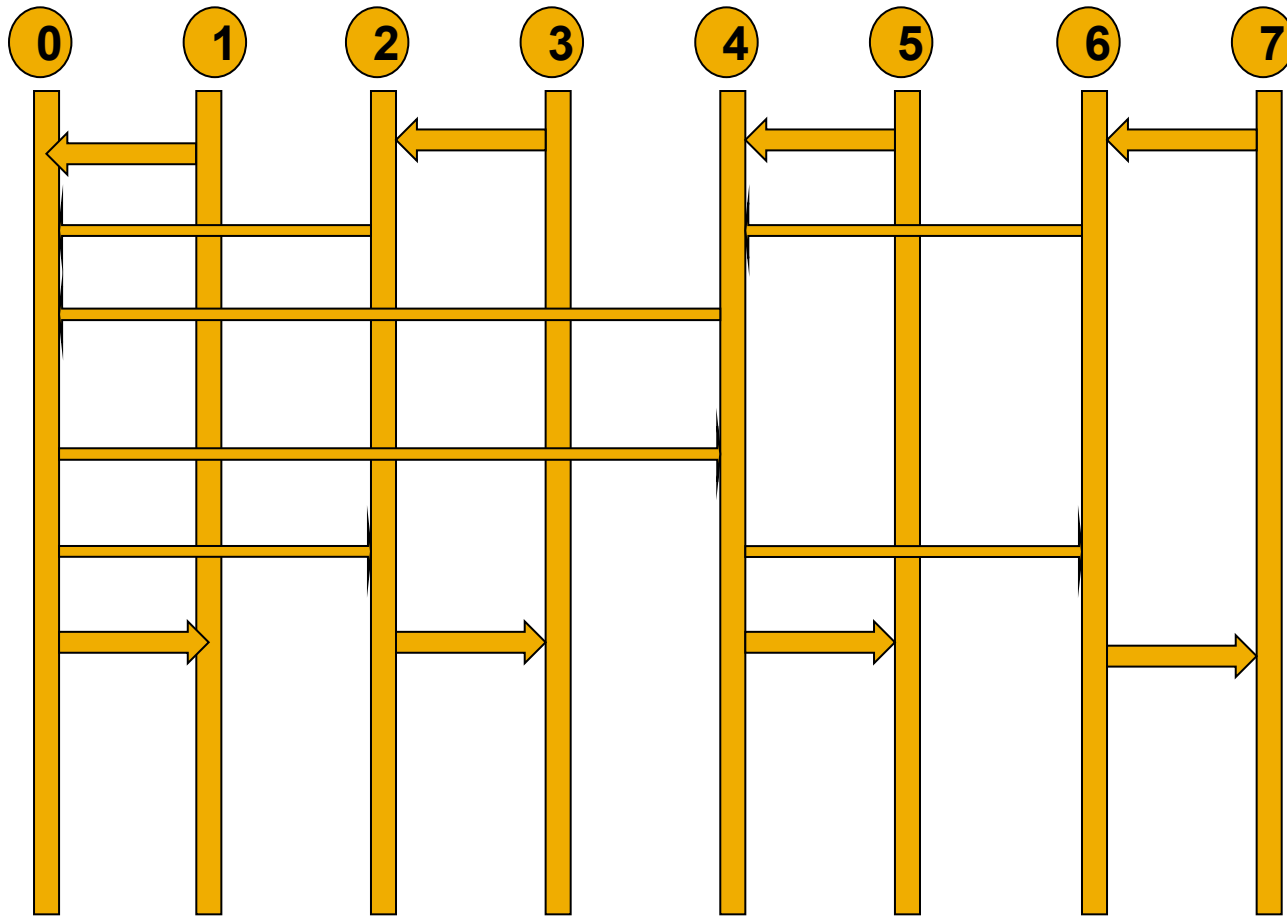


Рис. 7. Схема редукции

Вариант редукции: бабочка

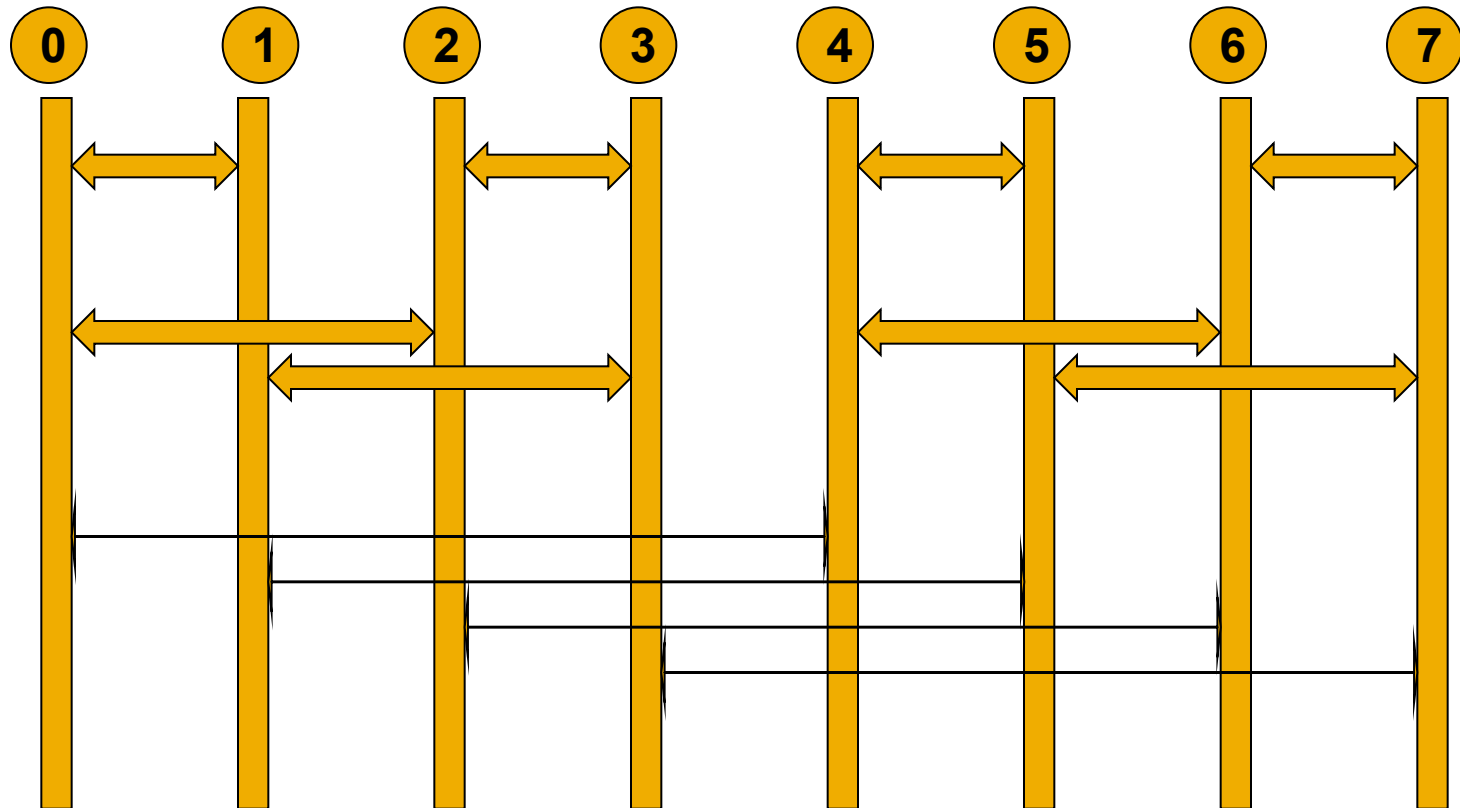


Рис. 8. Схема редукции

Виды операций пересылки сообщений «точка-точка»

Blocking sends	<code>MPI_Send(buffer, count, type, dest, tag, comm)</code>
Non-blocking sends	<code>MPI_Isend(buffer, count, type, dest, Tag, comm, request)</code>
Blocking receive	<code>MPI_Recv(buffer, count, type, source, tag, comm, status)</code>
Non-blocking receive	<code>MPI_Irecv(buffer, count, type, source, tag, comm, request)</code>

Функция MPI_Send

```
MPI_Send(*buf,  
count,datatype,  
dest,tag,comm)
```

```
MPI_SSEND(buf,  
count,datatype,  
dest,tag,comm,  
ierr)
```

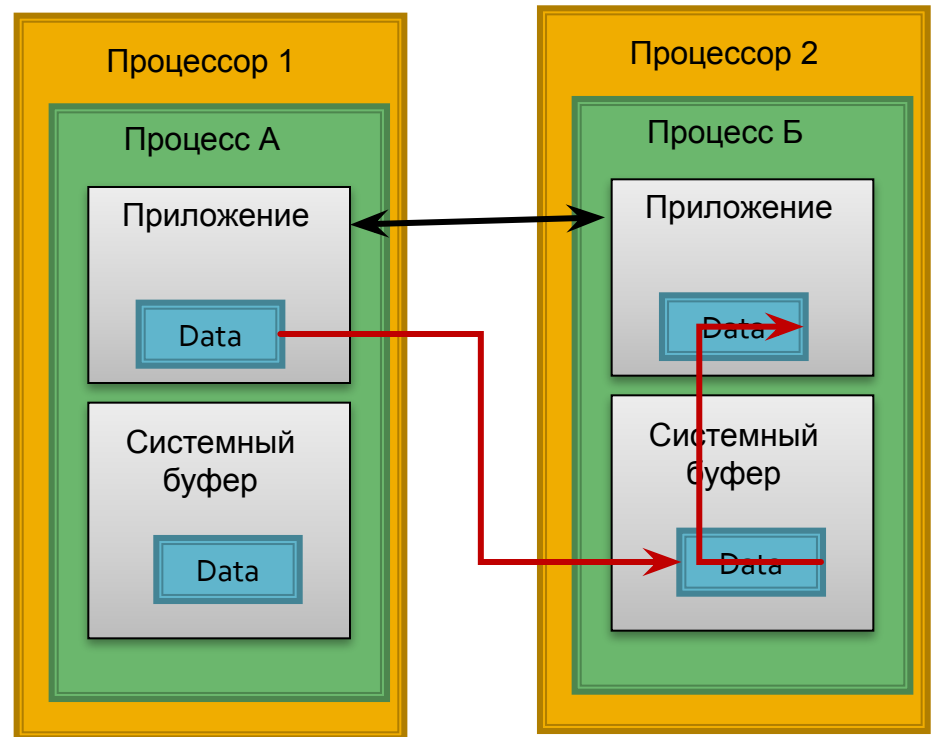


Рис. 9. Схема передачи сообщений "точка-точка" в MPI

Функция MPI_Bsend

```
MPI_Bsend(*buf,  
count,datatype,  
dest,tag,comm)
```

```
MPI_Buffer_attach  
(*buffer,size)
```

```
MPI_Buffer_detach  
(*buffer,size)
```

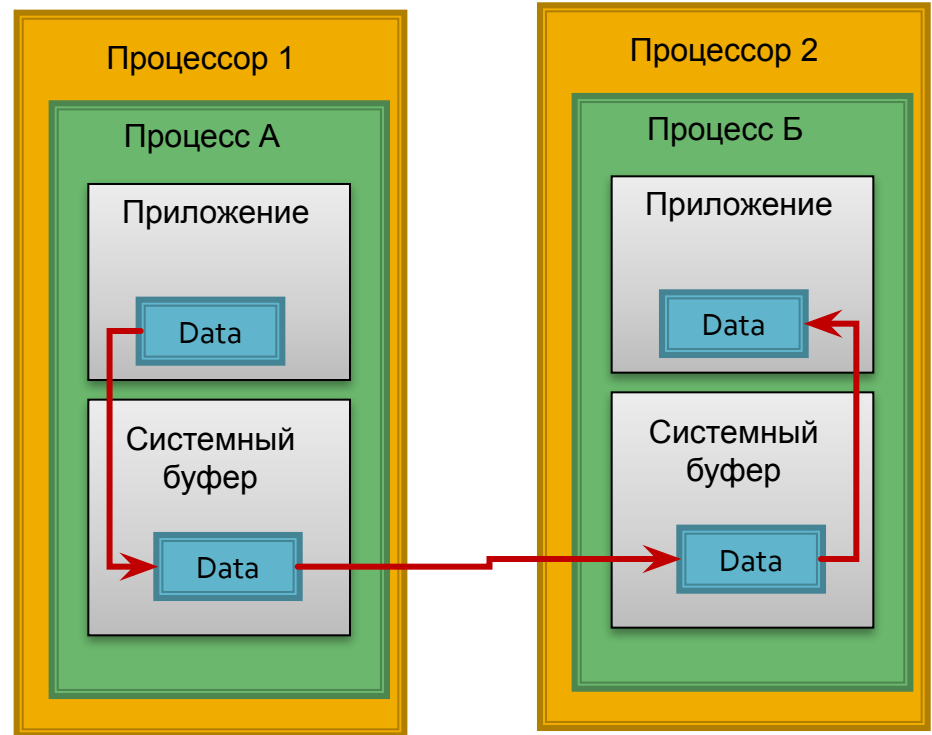


Рис. 11. Схема передачи сообщений “точка-точка” в MPI

Функция MPI_Rsend

```
MPI_Rsend(*buf,  
count,datatype,  
dest,tag,comm)
```

```
MPI_Probe  
(source,tag,  
comm,*status)
```

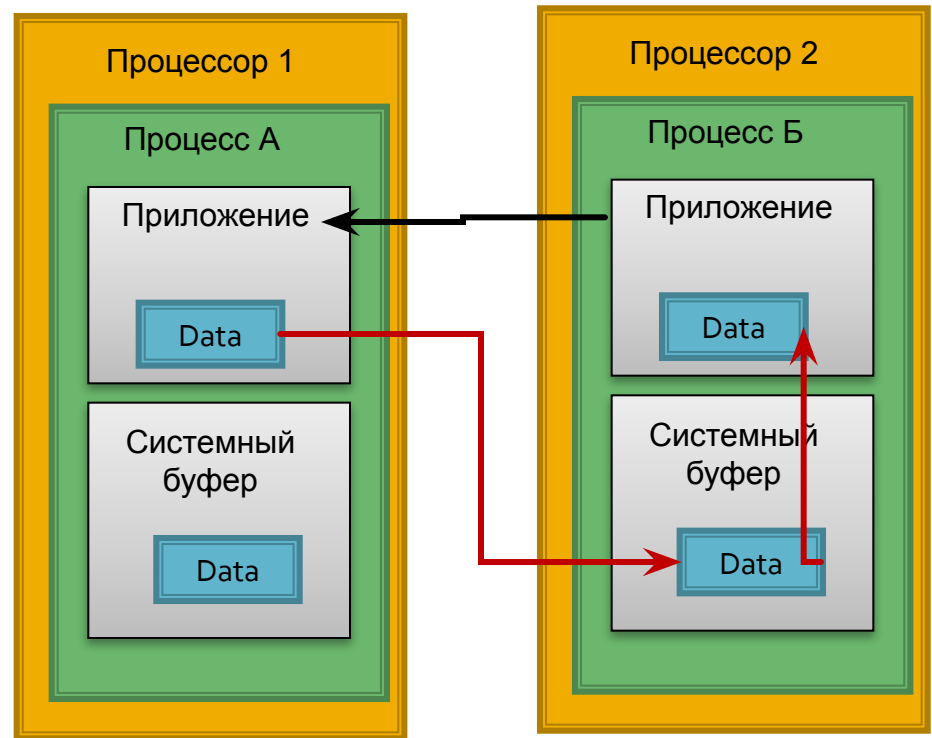


Рис. 12. Схема передачи сообщений “точка-точка” в MPI

Функция MPI_Sendrecv

```
MPI_Sendrecv (*sendbuf, sendcount, sendtype,  
              dest, sendtag,  
              *recvbuf, recvcount, recvtype,  
              source, recvtag, comm, *status)
```

Коллективные операции

- Необходимы для общего взаимодействия и синхронизации процессов.
- Функция общей синхронизации:

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

Широковещательная рассылка данных

```
int MPI_Bcast(void* buffer, int count, MPI_Datatype  
datatype, int root, MPI_Comm comm)
```

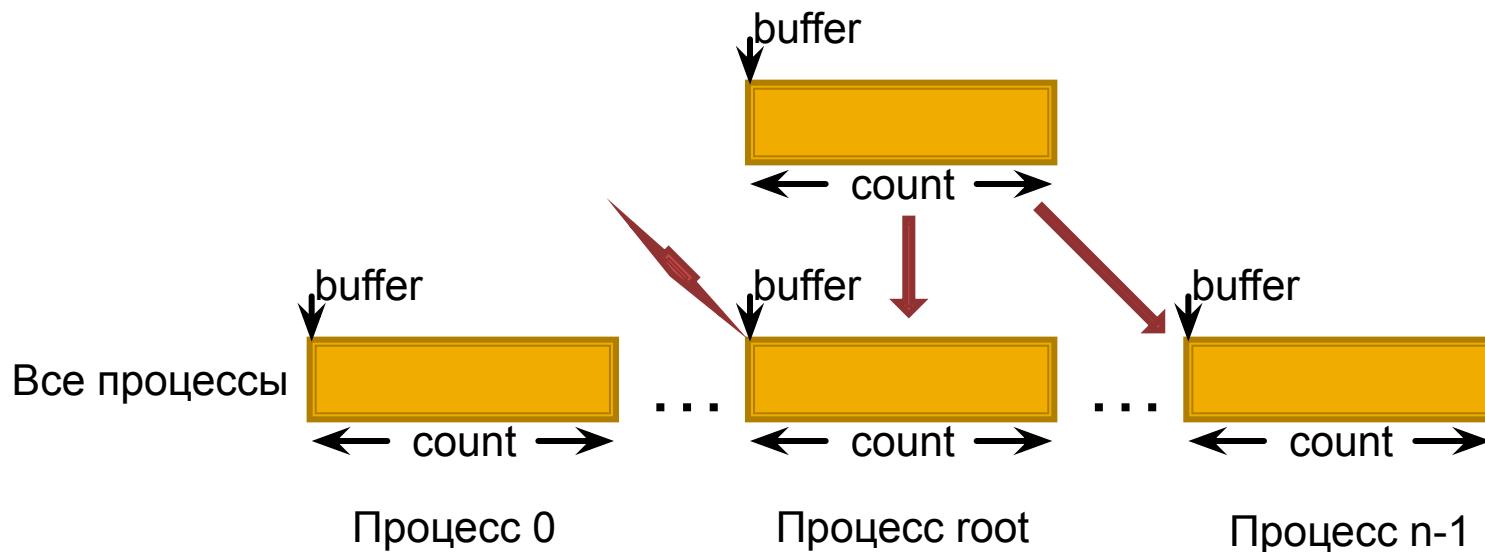


Рис. 1. Схема взаимодействия

Функции сбора блоков данных от всех процессов группы

```
int MPI_Gather(void* sendbuf, int sendcount, MPI_Datatype  
sendtype, void* recvbuf, int recvcount, MPI_Datatype  
recvtype, int root, MPI_Comm comm)
```

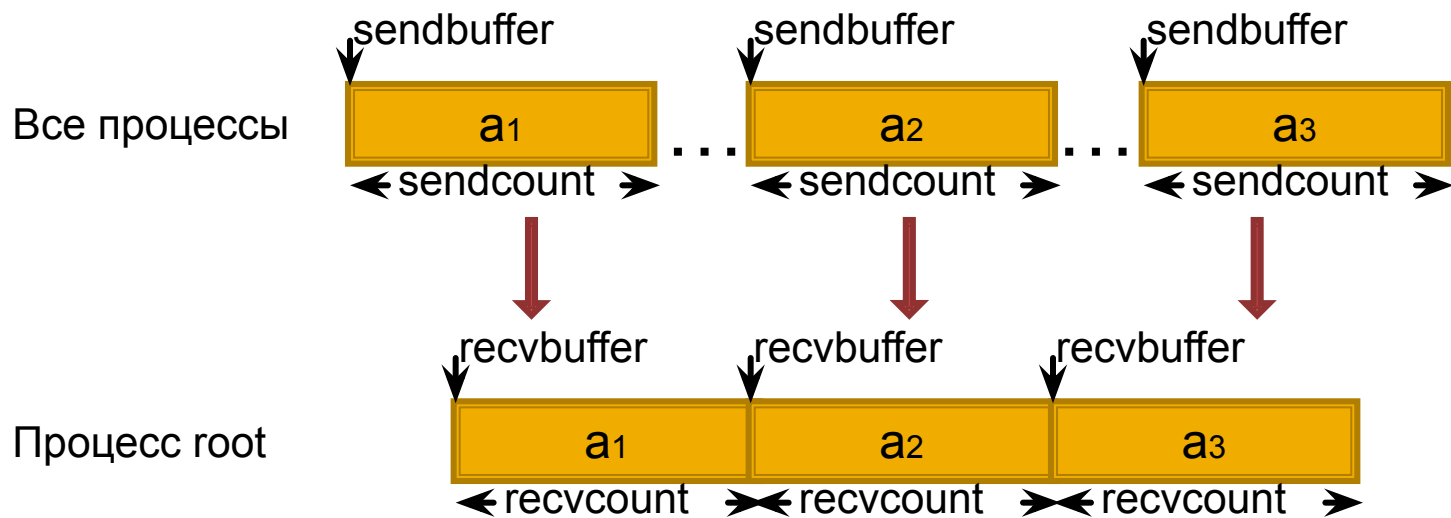


Рис. 2. Схема взаимодействия

Функции сбора блоков данных от всех процессов группы

```
int MPI_Allgather(void* sendbuf, int sendcount, MPI_Datatype  
sendtype, void* recvbuf, int recvcount, MPI_Datatype  
recvtype, MPI_Comm comm)
```

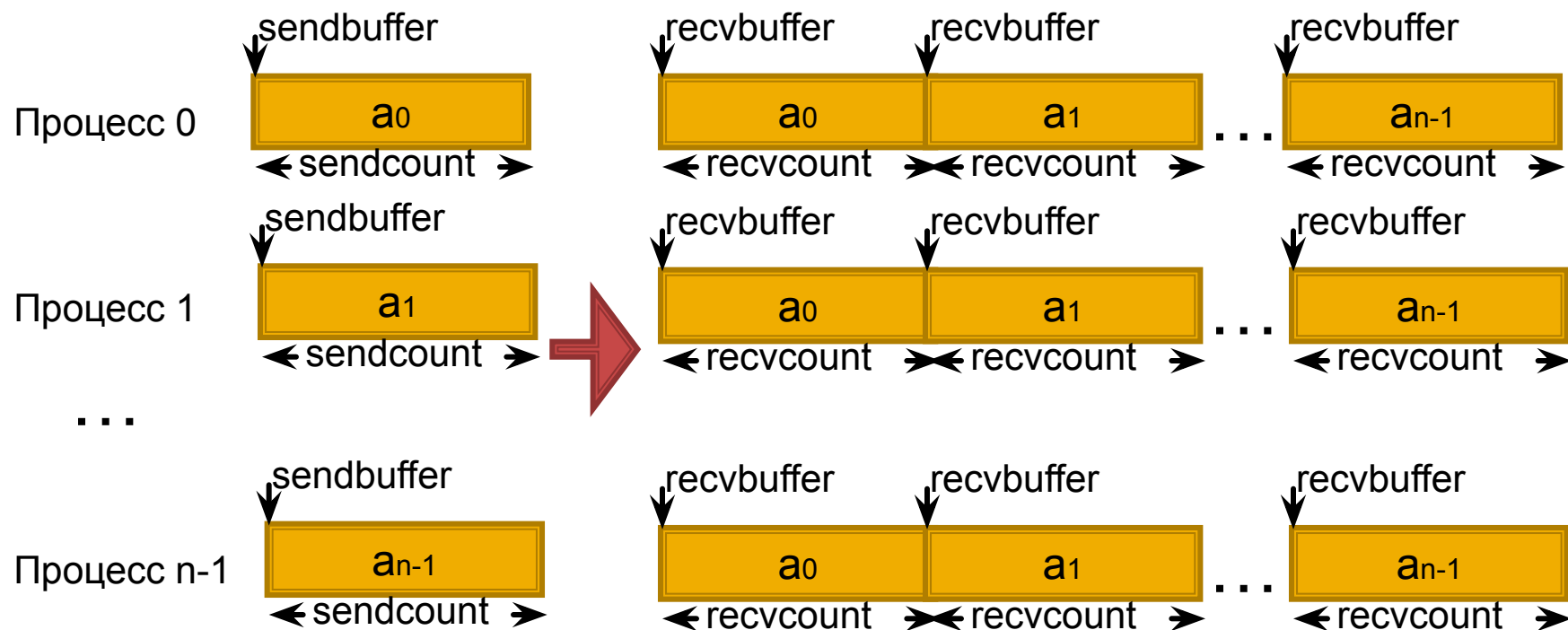


Рис. 3. Схема взаимодействия

Функции сбора блоков данных от всех процессов группы

```
int MPI_Gatherv(void* sendbuf, int sendcount, MPI_Datatype  
sendtype, void* rbuf, int *recvcounts, int *displs,  
MPI_Datatype recvtype, int root, MPI_Comm comm)
```

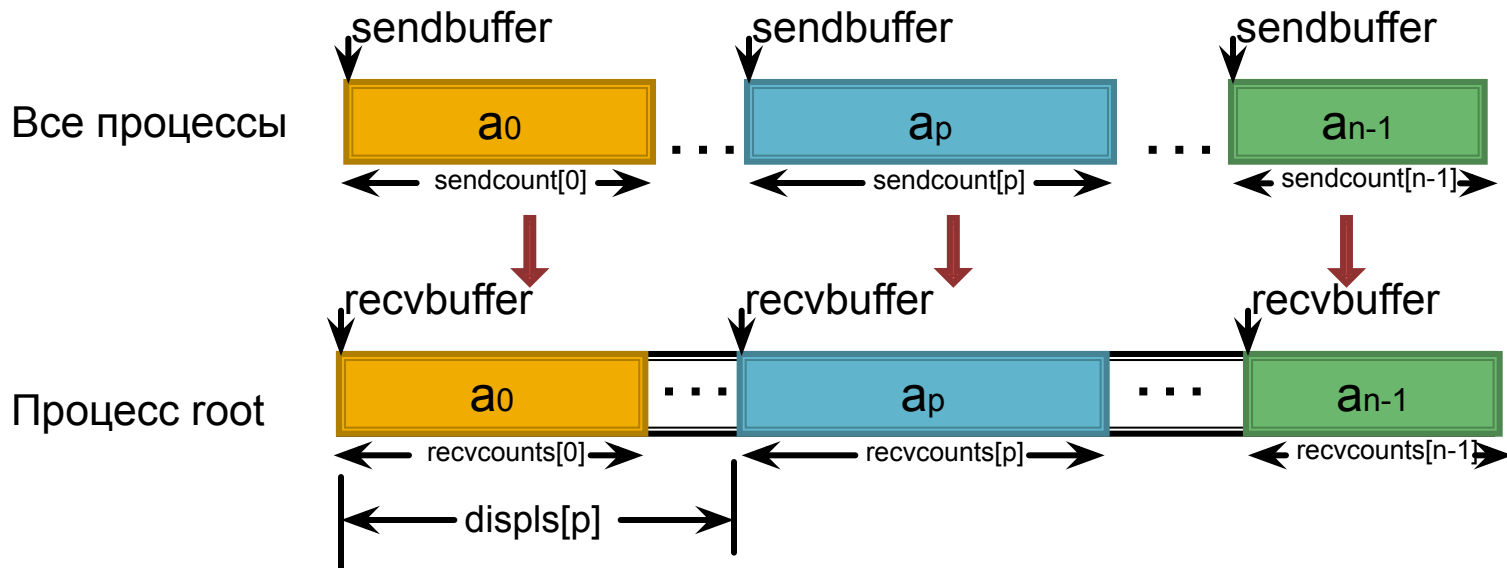


Рис. 4. Схема взаимодействия

Функции распределения блоков данных по всем процессам группы

```
int MPI_Scatter(void* sendbuf, int sendcount, MPI_Datatype  
sendtype, void* recvbuf, int recvcount, MPI_Datatype  
recvtype, int root, MPI_Comm comm)
```

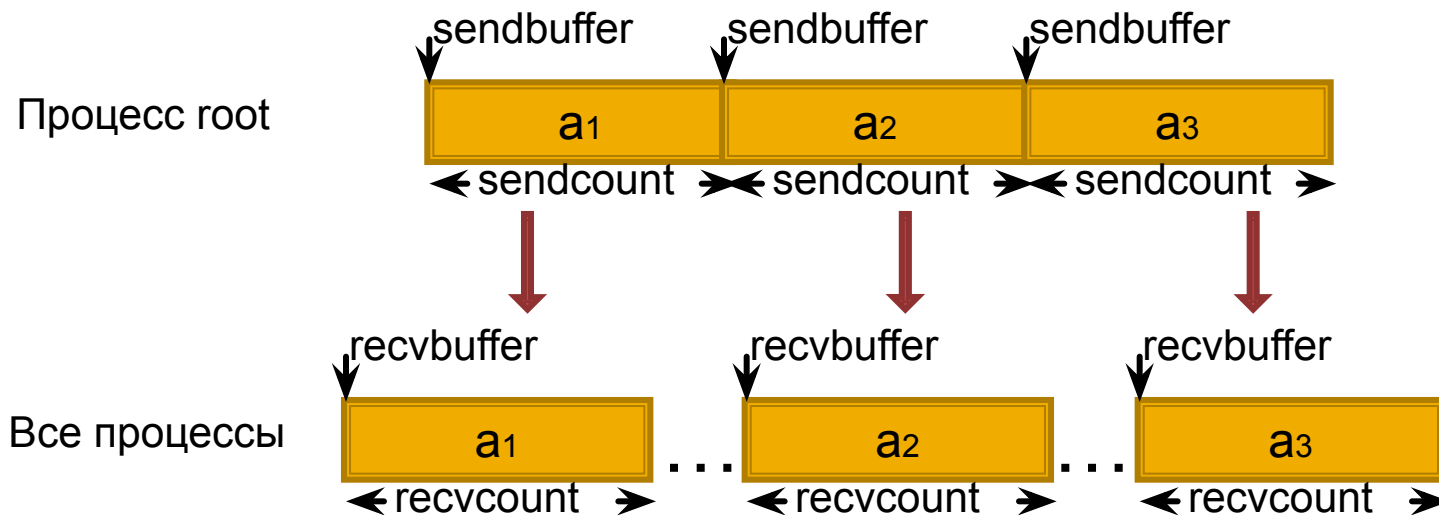


Рис. 5. Схема взаимодействия

Функции распределения блоков данных по всем процессам группы

```
int MPI_Scatterv(void* sendbuf, int *sendcounts, int *displs, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
```

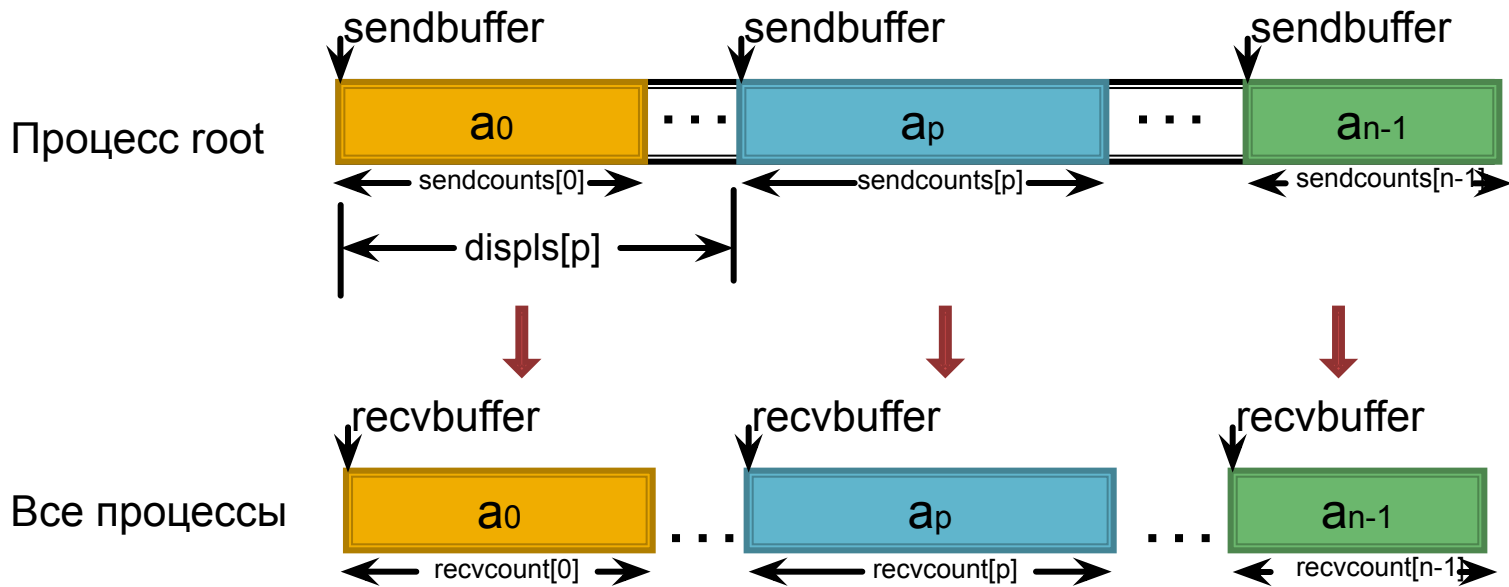


Рис. 6. Схема взаимодействия

Совмещенные коллективные операции

```
int MPI_Alltoall(void* sendbuf, int sendcount, MPI_Datatype  
sendtype, void* recvbuf, int recvcount, MPI_Datatype  
recvtype, MPI_Comm comm)
```

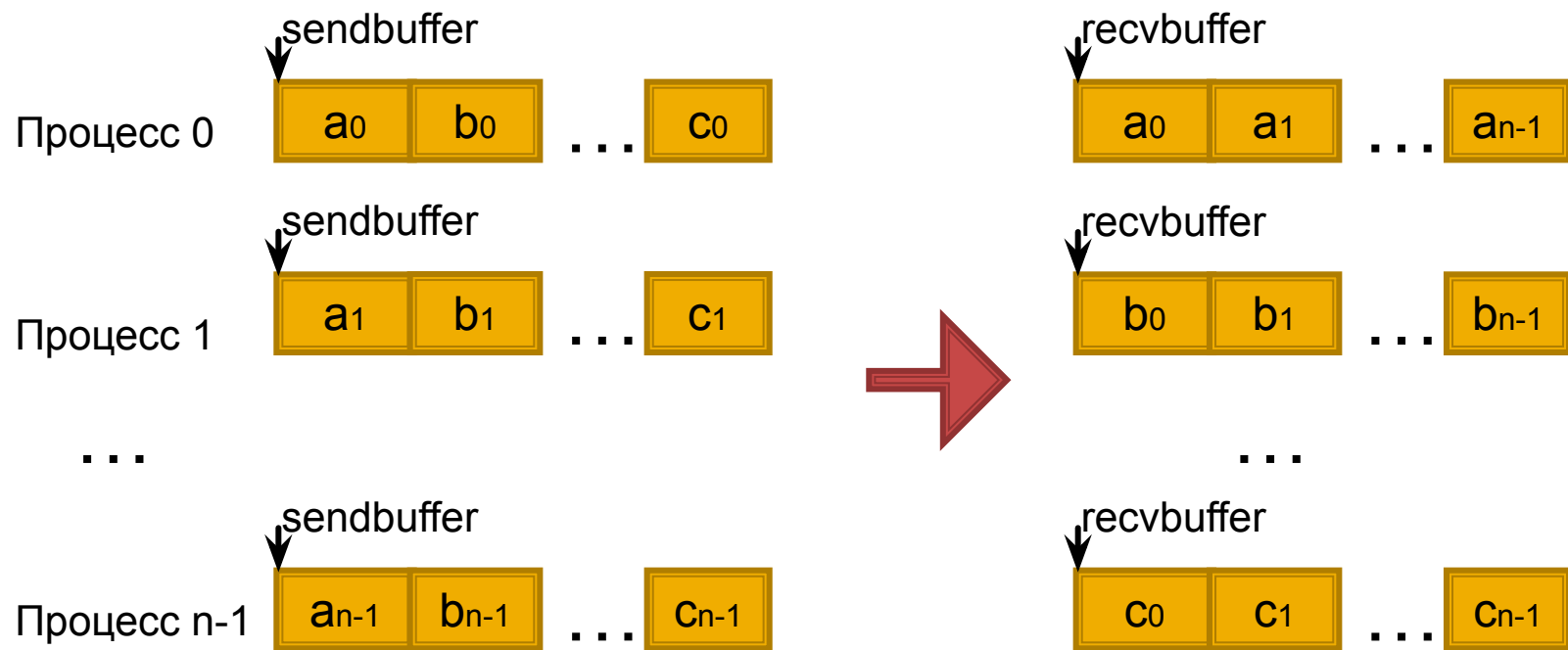


Рис. 7. Схема взаимодействия

Глобальные вычислительные операции над распределенными данными

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

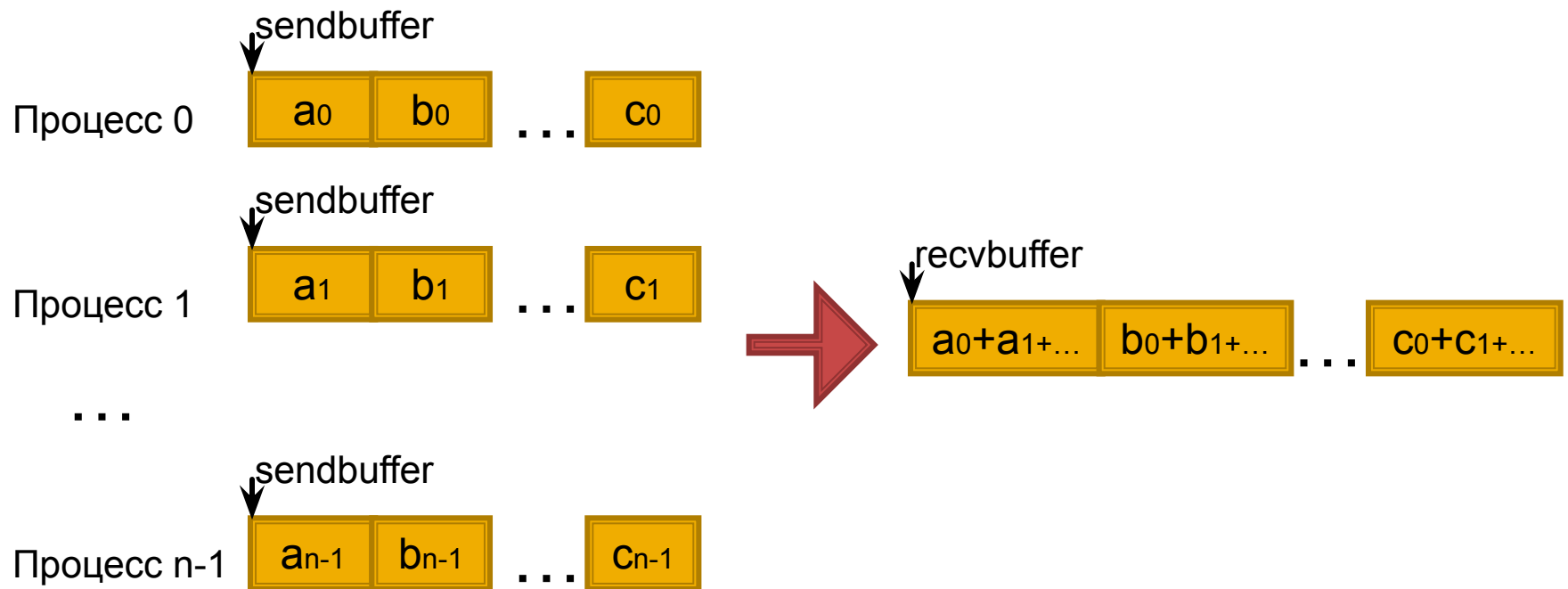


Рис. 8. Схема взаимодействия

Глобальные вычислительные операции над распределенными данными

Название	Операция
MPI_MAX	Максимум
MPI_MIN	Минимум
MPI_SUM	Сумма
MPI_PROD	Произведение
MPI_BAND	Логическое И
MPI_BOR	Логическое ИЛИ
MPI_BXOR	Логическое исключающее ИЛИ
MPI_MAXLOC	Максимальное значение и его индекс
MPI_MINLOC	Максимальное значение и его индекс

Глобальные вычислительные операции над распределенными данными

```
int MPI_Allreduce(void* sendbuf, void* recvbuf, int count,  
MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
```

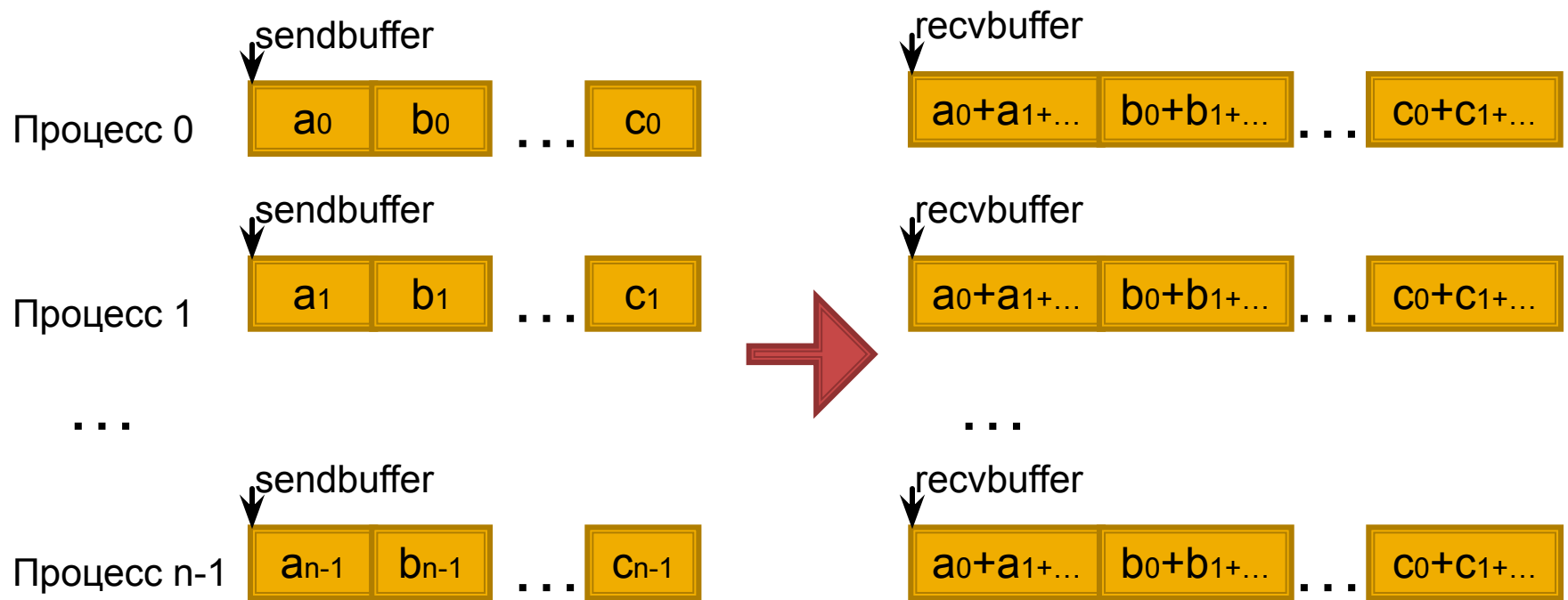


Рис. 9. Схема взаимодействия

Глобальные вычислительные операции над распределенными данными

```
int MPI_Reduce_scatter(void* sendbuf, void* recvbuf, int  
*recvcounts, MPI_Datatype datatype, MPI_Op op,  
MPI_Comm comm)
```

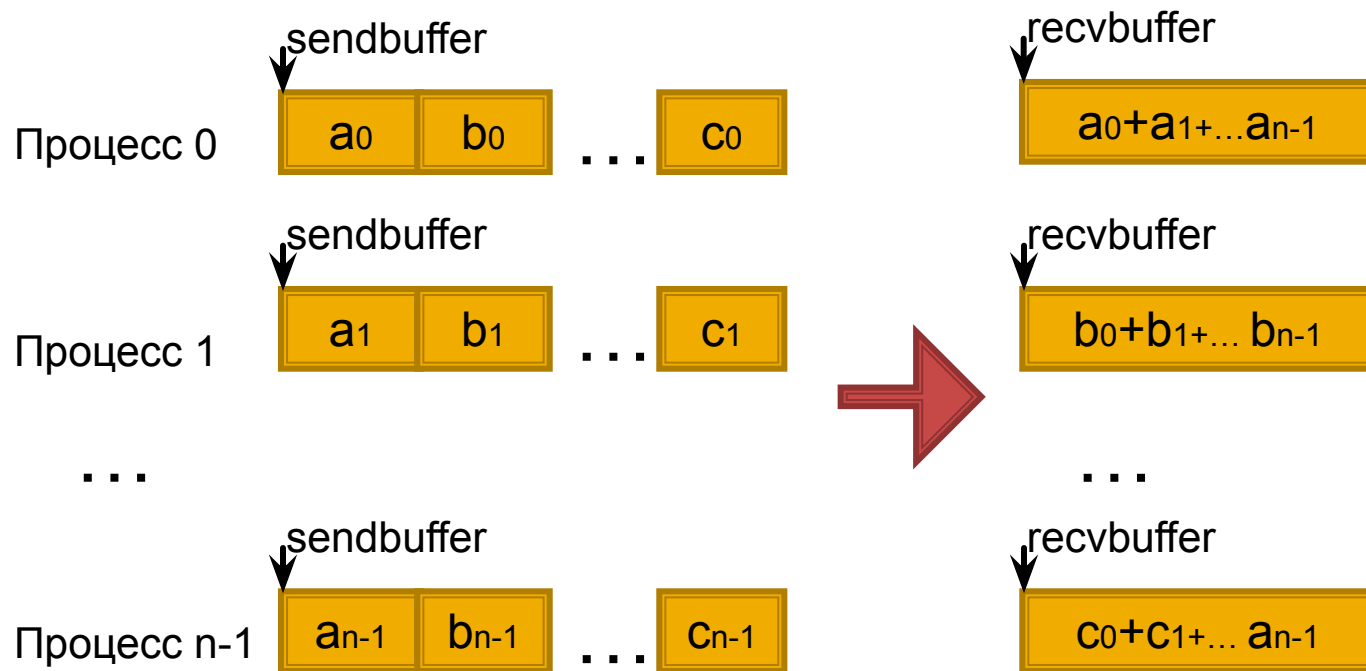


Рис. 10. Схема взаимодействия

Глобальные вычислительные операции над распределенными данными

```
int MPI_Scan(void* sendbuf, void* recvbuf, int count,  
            MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
```

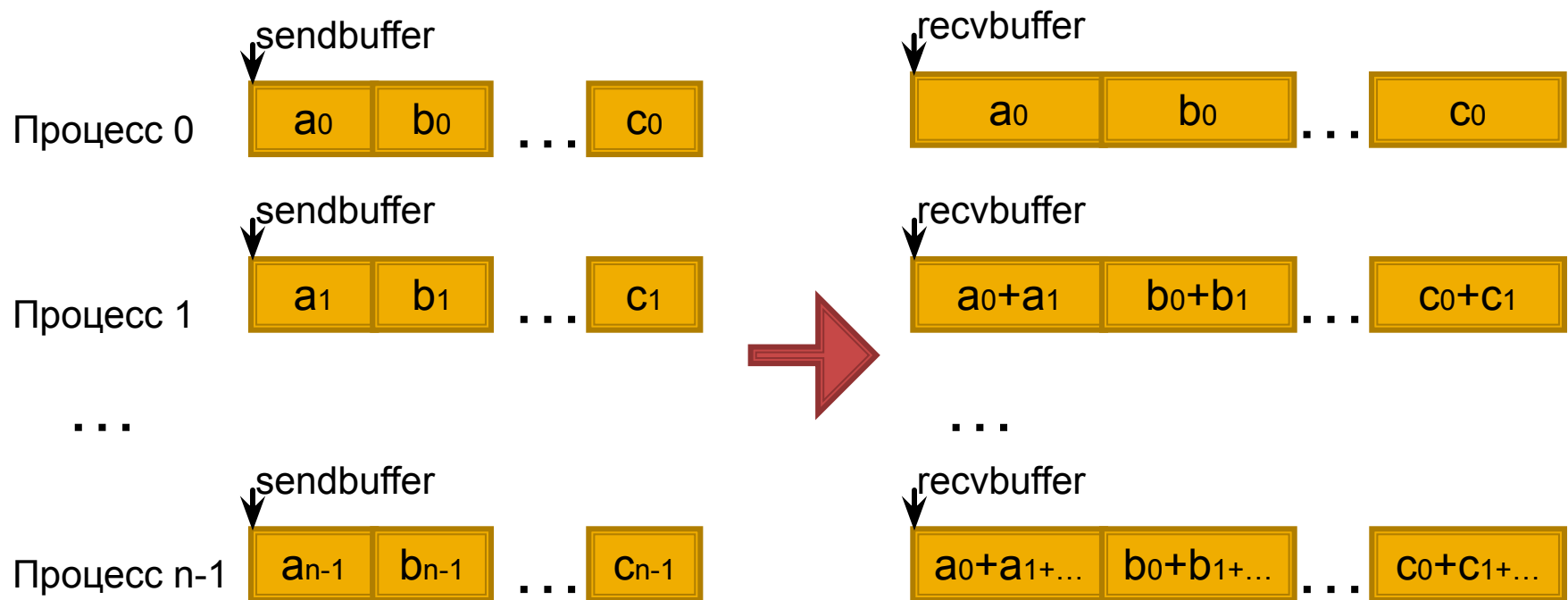


Рис. 11. Схема взаимодействия