Практика
MPI-групповые операции
Часть 1 (broadcast и reduce)

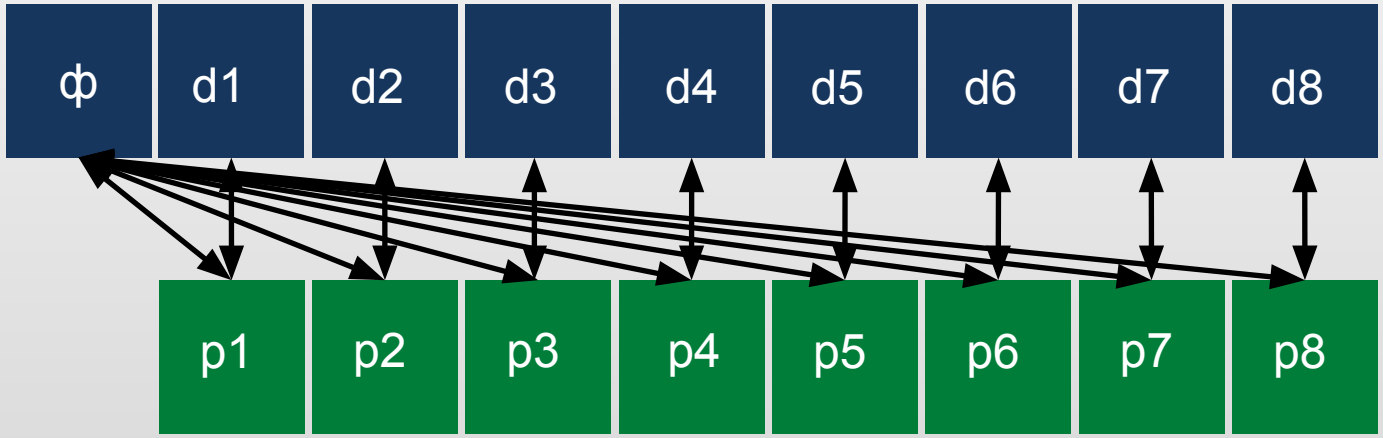
MPI – групповые операции

Операция broadcast

Пусть есть *N* процессов, которые используют одно и то же данное, лежащее в файле.

Для считывания данного им понадобится время Θ(*N*).

Константа перед N – очень большая



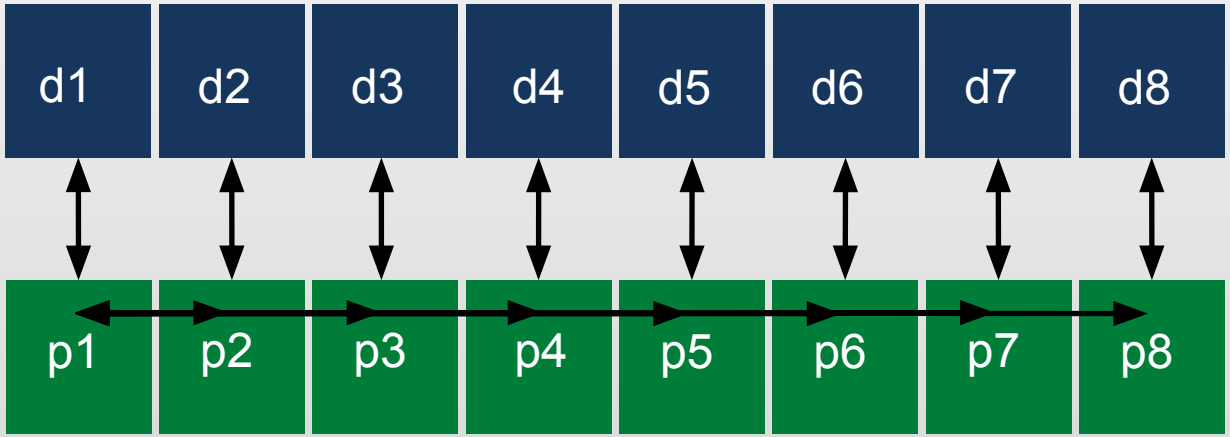
MPI – групповые операции

Операция broadcast

Пусть есть *N* процессов, которые используют одно и то же данное, лежащее в в памяти первого процесса.

Для размножения данного понадобится время Θ(*N*).

Константа перед N – много меньше, чем в первой схеме



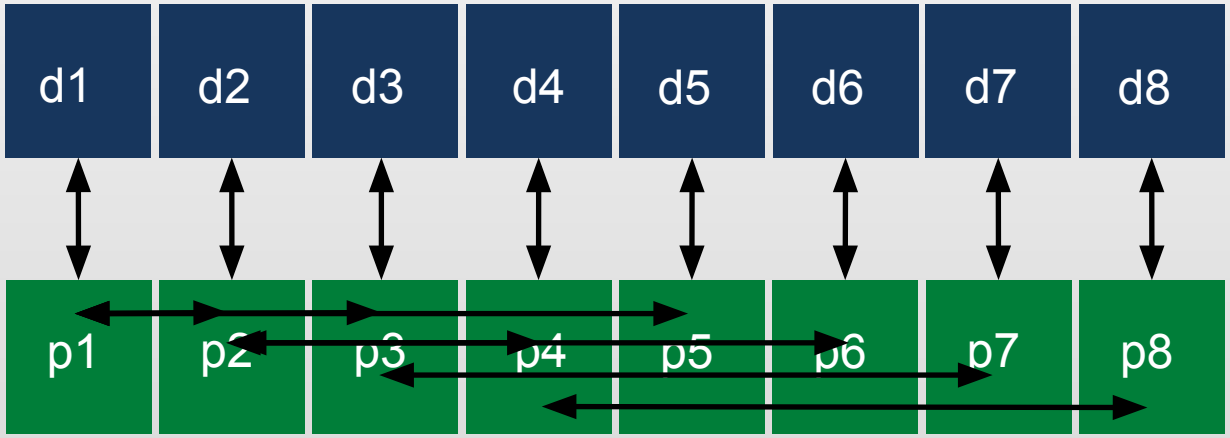
MPI – групповые операции

Операция broadcast

Пусть есть *N* процессов, которые используют одно и то же данное, лежащее в в памяти первого процесса.

Для размножения данного понадобится время $\Theta(\log N)$.

Константа – практически та же, что и во второй схеме



MPI – групповые операции

Операция broadcast

Для логарифмической рассылки данных от одного процесса всем остальным процессам в некоторой группе используется функция MPI_Bcast().

```
#include <mpi.h>

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

.

Возвращаемое значение – стандартное:

MPI_SUCCESS – все хорошо, иначе – что-то пошло не так.

.

MPI – групповые операции

Операция broadcast

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

buffer – для процесса с ранком root – адрес памяти, где лежит рассылаемая информация, для остальных – адрес памяти, по которому нужно расположить принятые данные (у каждого процесса может быть свой)

count – количество данных (элементов типа datatype) в сообщении

datatype – тип данных в сообщении

root – ранк процесса в группе, описываемой коммуникатором comm, на котором лежит рассылаемая информация

comm – коммуникатор группы процессов для групповой операции

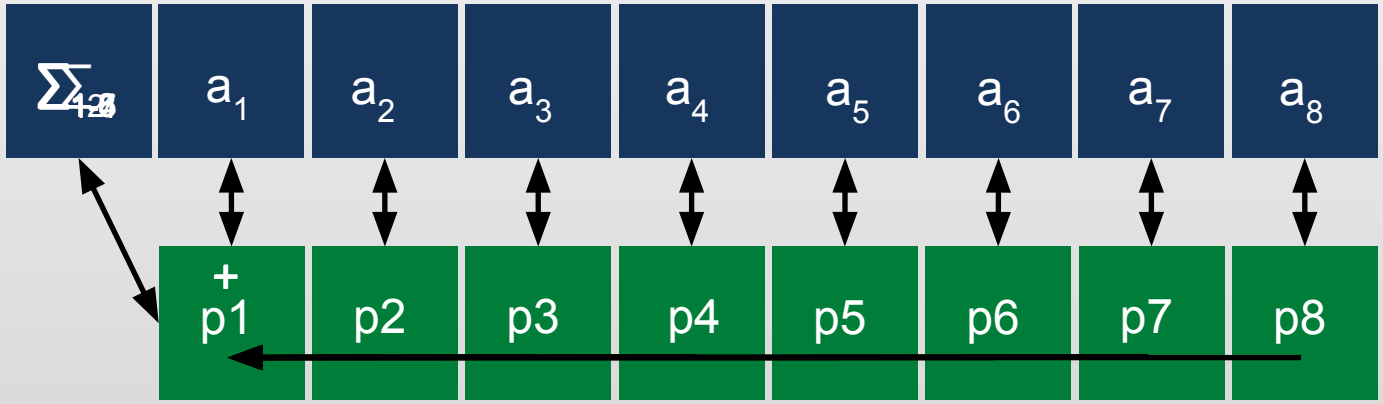
MPI – групповые операции

Редукционные операции

Пусть есть N процессов и N значений – a_1, …, a_N, по одному на каждом из процессов.

Требуется посчитать сумму $a_1 + \ldots + a_N$.

Время на суммирование – $\Theta(N)$.



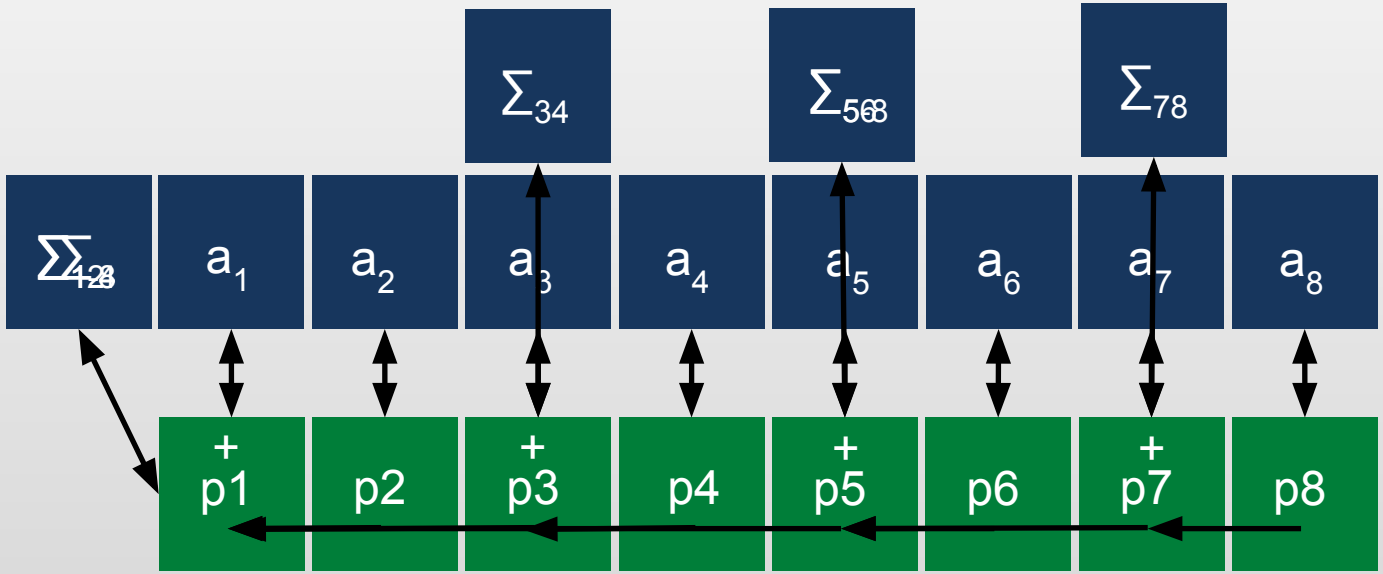
MPI – групповые операции

Редукционные операции

Пусть есть *N* процессов и *N* значений – a_1, …, a_N, по одному на каждом из процессов.

Требуется посчитать сумму $a_1 + \ldots + a_N$.

Время на суммирование – $\Theta(\log N)$.



MPI – групповые операции

Редукционные операции

Редукционные операции – это операции, который обладают свойствами коммутативности (a op b = b op a) и ассоциативности ((a op b) op c = a op (b op c)).

Примеры редукционных операций:

+

*

&

|

&&

||

max

min

MPI – групповые операции

Редукционные операции

Для редукционных операций в некоторой группе процессов используется функция MPI_Reduce().

```
#include <mpi.h>

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

.

Возвращаемое значение – стандартное:

MPI_SUCCESS – все хорошо, иначе – что-то пошло не так.

.

MPI – групповые операции

Редукционные операции

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

op – код операции (какая редукционная операция выполняется)

MPI – групповые операции

Редукционные операции

Операция	Код операции
+	MPI_SUM
*	MPI_PROD
min	MPI_MIN
max	MPI_MAX
&	MPI_BAND
|	MPI_BOR
&&	MPI_LAND
||	MPI_LOR

MPI – групповые операции

Редукционные операции

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

sendbuf – адрес памяти, где у процесса лежат исходные данные (у каждого процесса может быть свой)

recvbuf – адрес памяти, куда будет занесен результат, - реально используется только на процессе с ранком root

op – код операции (какая редукционная операция выполняется)

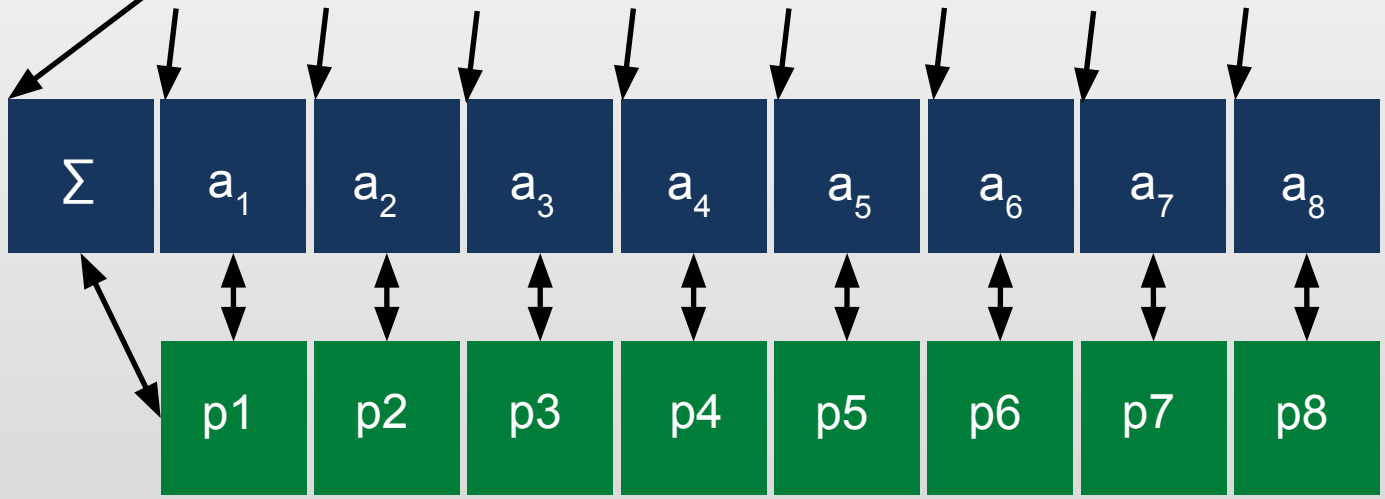
root – ранк процесса в группе, описываемой коммуникатором comm, на который нужно поместить результат

comm – коммуникатор группы процессов для групповой операции

MPI – групповые операции

Редукционные операции

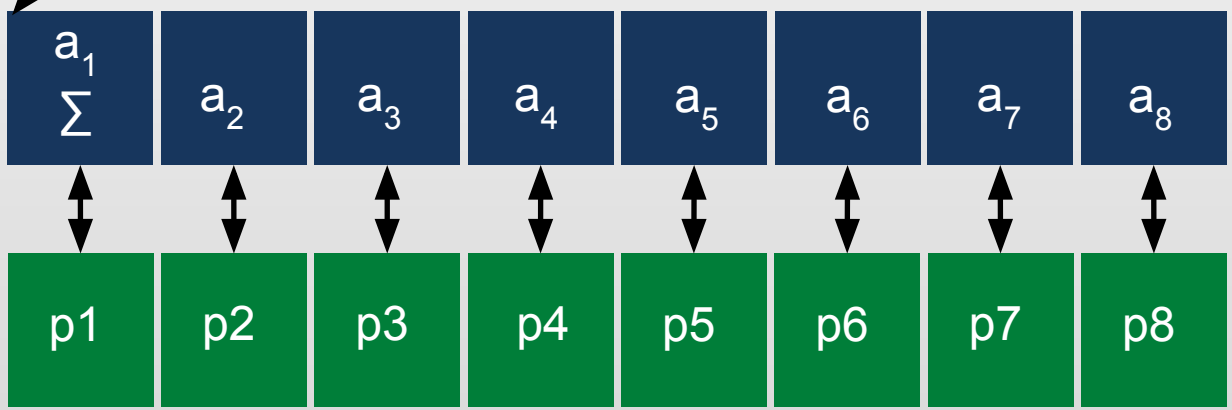
root – в нашем случае процесс с рангом 0
op – у нас код операции MPI_SUM
recvbuf – для процесса root (т.е. у нас для процесса с рангом 0) - это адрес переменной для размещения результата
– для остальных – произвольный адрес – не используется
sendbuf – для каждого процесса это адрес, где лежат исходные данные



MPI – групповые операции

Редукционные операции

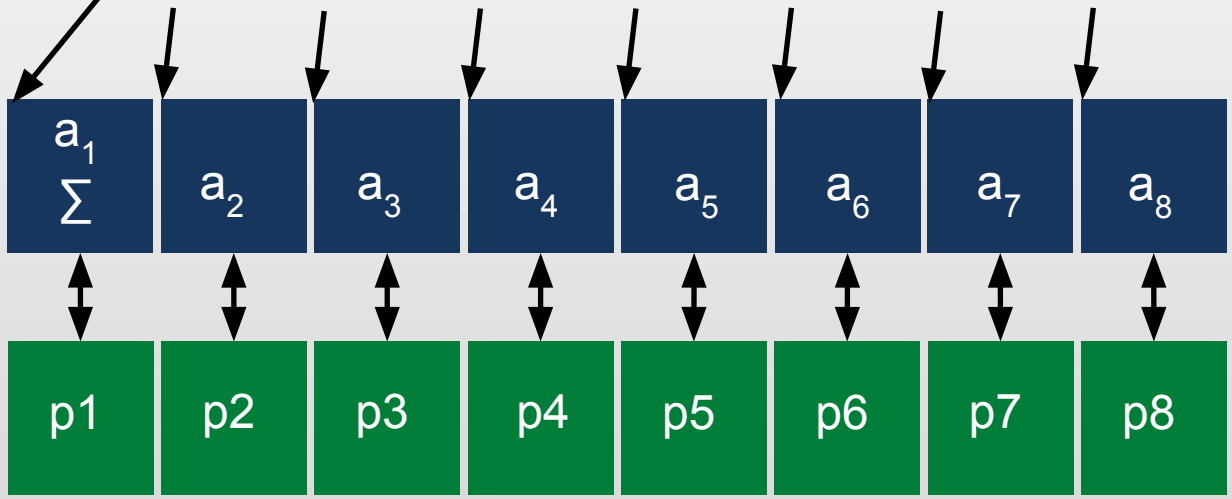
root – в нашем случае процесс с рангом 0
op – у нас код операции MPI_SUM
recvbuf – для процесса root (т.е. у нас для процесса с рангом 0) - это адрес переменной для размещения результата
– для остальных – произвольный адрес – не используется
sendbuf – для каждого процесса это адрес, где лежат исходные данные



MPI – групповые операции

Редукционные операции

root – в нашем случае процесс с рангом 0
op – у нас код операции MPI_SUM
recvbuf – для процесса root (т.е. у нас для процесса с рангом 0) - это адрес переменной для размещения результата
– для остальных – произвольный адрес – не используется
sendbuf – для процесса root – значение MPI_IN_PLACE, для остальных – адрес, где лежат исходные данные



MPI – групповые операции

Редукционные операции

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

sendbuf – адрес памяти, где у процесса лежат исходные данные (у каждого процесса может быть свой)

recvbuf – адрес памяти, куда будет занесен результат, - реально используется только на процессе с ранком root

count – количество данных (элементов типа datatype) для операции

datatype – тип данных, над которыми выполняется операция

op – код операции (какая редукционная операция выполняется)

root – ранк процесса в группе, описываемой коммуникатором comm, на который нужно поместить результат

comm – коммуникатор группы процессов для групповой операции

MPI – групповые операции

Задача

Модифицировать программу расчета значения числа пи с использованием MPI_Bcast и MPI_Reduce

Для рассылки значения N от процесса с рангом 0 использовать MPI_Bcast, для окончательного суммирования использовать MPI_Reduce.