Многопоточные вычисления на основе технологий MPI и OpenMP: Коллективные операции

Н. И. Хохлов

МФТИ, Долгопрудный

26 октября 2016 г.

Состав

- Синхронизация всех процессов с помощью барьеров (MPI_Barrier).
- Коллективные коммуникационные операции.
- Глобальные вычислительные операции (sum, min, max и др.) над данными, расположенными в адресных пространствах различных процессов.

Коллективные коммуникационные операции

- рассылка информации от одного процесса всем остальным членам некоторой области связи (MPI_Bcast);
- сборка (gather) распределенного по процессам массива в один массив с сохранением его в адресном пространстве выделенного (root) процесса (MPI_Gather, MPI_Gatherv);
- сборка (gather) распределенного массива в один массив с рассылкой его всем процессам некоторой области связи (MPI_Allgather, MPI_Allgatherv);
- разбиение массива и рассылка его фрагментов (scatter) всем процессам области связи (MPI_Scatter, MPI_Scatterv);
- совмещенная операция Scatter/Gather (All-to-All), каждый процесс делит данные из своего буфера передачи и разбрасывает фрагменты всем остальным процессам, одновременно собирая фрагменты, посланные другими процессами в свой буфер приема (MPI_Alltoall, MPI_Alltoallv).

Глобальные вычислительные операции

- с сохранением результата в адресном пространстве одного процесса (MPI_Reduce);
- с рассылкой результата всем процессам (MPI_Allreduce);
- совмещенная операция Reduce/Scatter (MPI_Reduce_scatter);
- префиксная редукция (MPI_Scan).

Простой/векторный вариант

Все коммуникационные подпрограммы, за исключением MPI_Bcast, представлены в двух вариантах:

- простой вариант, когда все части передаваемого сообщения имеют одинаковую длину и занимают смежные области в адресном пространстве процессов;
- "векторный"вариант, который предоставляет более широкие возможности по организации коллективных коммуникаций, снимая ограничения, как в части длин блоков, так и в части размещения данных в адресном пространстве процессов. Векторные варианты отличаются дополнительным символом "v"в конце имени функции.

Особенности

- Коллективные коммуникации не взаимодействуют с коммуникациями типа точка-точка.
- Коллективные коммуникации выполняются в режиме с блокировкой. Возврат из подпрограммы в каждом процессе происходит тогда, когда его участие в коллективной операции завершилось, однако это не означает, что другие процессы завершили операцию.
- Количество получаемых данных должно быть равно количеству посланных данных.
- Типы элементов посылаемых и получаемых сообщений должны совпадать.
- Сообщения не имеют идентификаторов (tag).

Функция синхронизации процессов

- int MPI_Barrier(MPI_Comm comm);
  - comm – комуникатор.

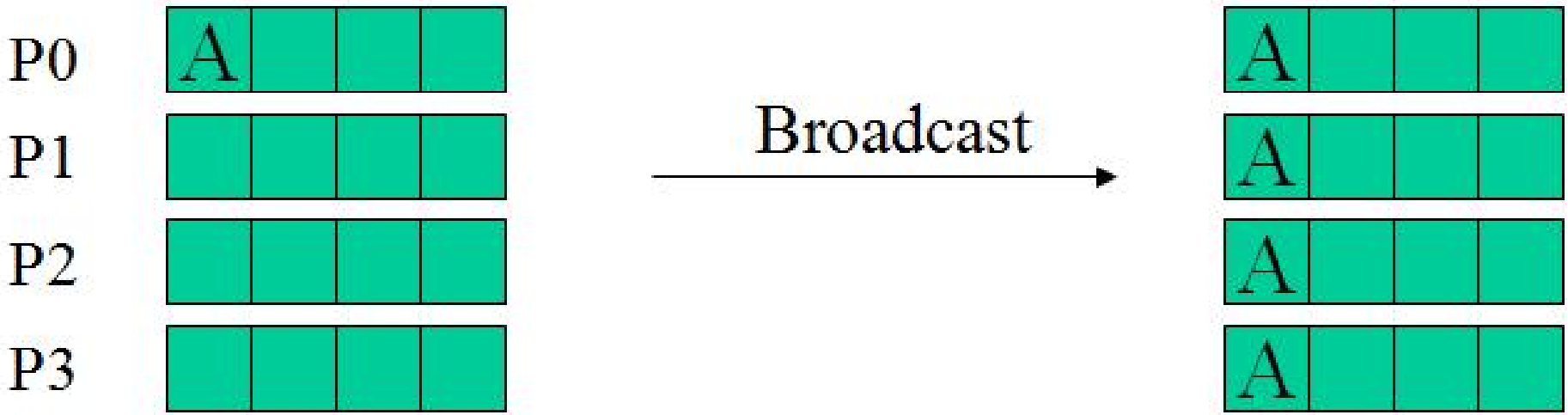
Блокирует работу вызвавшего ее процесса до тех пор, пока все другие процессы группы также не вызовут эту функцию. Завершение работы этой функции возможно только всеми процессами одновременно (все процессы "преодолевают барьер"одновременно). Использование барьера гарантирует, что ни один из процессов не приступит раньше времени к выполнению следующего этапа, пока результат работы предыдущего не будет окончательно сформирован. Неявную синхронизацию процессов выполняет любая коллективная функция.

Широковещательная рассылка данных

- int MPI_Bcast(void* buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm);
  - buffer – адрес начала расположения в памяти рассылаемых данных;
  - count – число посылаемых элементов;
  - datatype – тип посылаемых элементов;
  - root – номер процесса-отправителя;
  - comm – комуникатор.

Процесс с номером root рассылает сообщение из своего буфера передачи всем процессам области связи коммуникатора comm. После завершения подпрограммы каждый процесс в области связи коммуникатора comm, включая и самого отправителя, получит копию сообщения от процесса-отправителя root.

Графическая интерпретация



Пример

```
a = rank;
root = 0;
printf("Before: rank = %d, a = %d\n");

MPI_Bcast(&a, 1, MPI_INT, root, comm);

printf("After: rank = %d, a = %d\n");
```

Функции сбора блоков данных от всех процессов группы

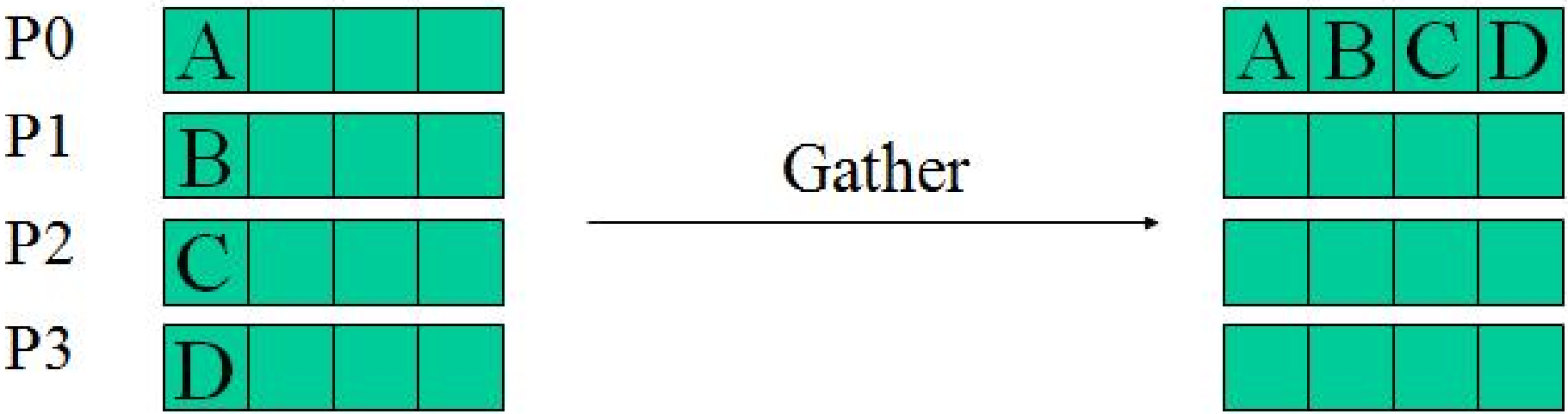
- MPI_Gather.
- MPI_Allgather.
- MPI_Gatherv.
- MPI_Allgatherv.

MPI_Gather

- int MPI_Gather(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm);
  - sendbuf – адрес начала размещения посылаемых данных;
  - sendcount – число посылаемых элементов;
  - sendtype – тип посылаемых элементов;
  - recvbuf – адрес начала буфера приема (используется только в процессе-получателе root);
  - recvcount – число элементов, получаемых от каждого процесса (используется только в процессе-получателе root);
  - recvtype – тип получаемых элементов;
  - root – номер процесса-получателя;
  - comm – комуникатор.

Производит сборку блоков данных, посылаемых всеми процессами группы, в один массив процесса с номером root. Длина блоков одинаковая. Объединение происходит в порядке увеличения номеров процессов-отправителей.

Графическая интерпретация



Пример

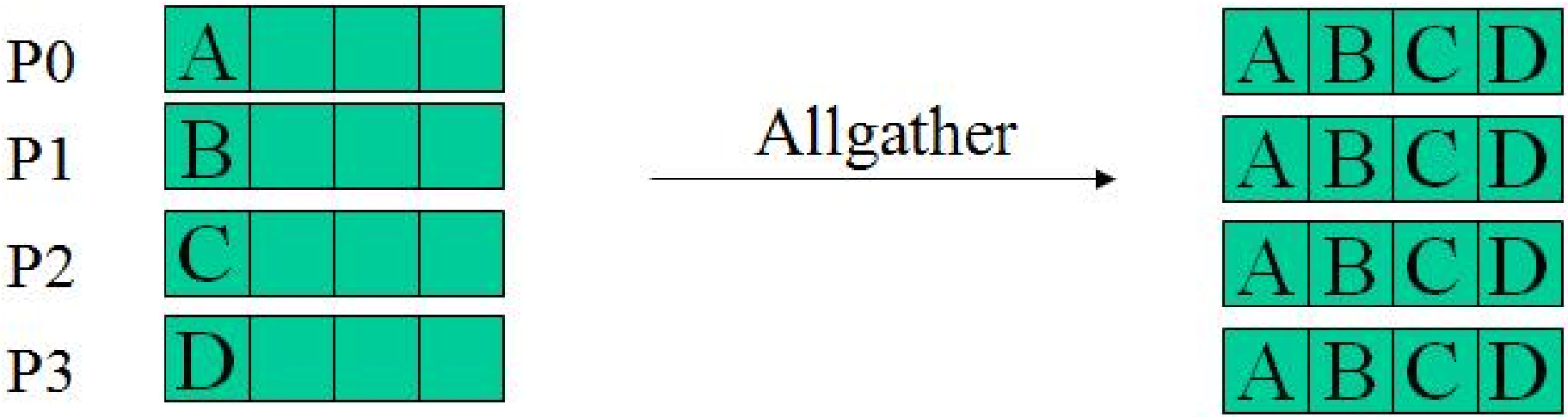
```
MPI_Comm comm;
int array[100];
int root, *rbuf;
. . .
MPI_Comm_rank(comm, &rank);
MPI_Comm_size(comm, &size);
if (root == rank)
    rbuf = (int*)malloc(size * 100 * sizeof(int));
MPI_Gather(array, 100, MPI_INT,
           rbuf, 100, MPI_INT, root, comm);
```

MPI_Allgather

- int MPI_Allgather(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm);
  - sendbuf – адрес начала размещения посылаемых данных;
  - sendcount – число посылаемых элементов;
  - sendtype – тип посылаемых элементов;
  - recvbuf – адрес начала буфера приема (используется только в процессе-получателе root);
  - recvcount – число элементов, получаемых от каждого процесса (используется только в процессе-получателе root);
  - recvtype – тип получаемых элементов;
  - comm – комуникатор.

Выполняется так же, как MPI_Gather, но получателями являются все процессы группы. Данные, посланные процессом i из своего буфера sendbuf, помещаются в i-ю порцию буфера recvbuf каждого процесса. После завершения операции содержимое буферов приема recvbuf у всех процессов одинаково.

Графическая интерпретация



Пример

```
MPI_Comm comm;
int array[100];
int *rbuf;
. . .
MPI_Comm_size(comm, &size);
rbuf = (int*)malloc(size * 100 * sizeof(int));
MPI_Allgather(array, 100, MPI_INT,
              rbuf, 100, MPI_INT, comm);
```

MPI_Gatherv

- int MPI_Gatherv(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* rbuf, int *recvcounts, int *displs, MPI_Datatype recvtype, int root, MPI_Comm comm);
  - sendbuf – адрес начала размещения посылаемых данных;
  - sendcount – число посылаемых элементов;
  - sendtype – тип посылаемых элементов;
  - rbuf – адрес начала буфера приема (используется только в процессе-получателе root);
  - recvcounts – целочисленный массив (размер равен числу процессов в группе), i-й элемент которого определяет число элементов, которое должно быть получено от процесса i;
  - displs – целочисленный массив (размер равен числу процессов в группе), i-ое значение определяет смещение i-го блока данных относительно начала rbuf;
  - recvtype – тип получаемых элементов;
  - root – номер процесса-получателя;
  - comm – комуникатор.

Данные от i-го процесса, размещаются начиная с rbuf + displs[i].

MPI_Allatherv

- int MPI_Allatherv(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* rbuf, int *recvcounts, int *displs, MPI_Datatype recvtype, MPI_Comm comm);
  - sendbuf – адрес начала размещения посылаемых данных;
  - sendcount – число посылаемых элементов;
  - sendtype – тип посылаемых элементов;
  - rbuf – адрес начала буфера приема (используется только в процессе-получателе root);
  - recvcounts – целочисленный массив (размер равен числу процессов в группе), i-й элемент которого определяет число элементов, которое должно быть получено от процесса i;
  - displs – целочисленный массив (размер равен числу процессов в группе), i-ое значение определяет смещение i-го блока данных относительно начала rbuf;
  - recvtype – тип получаемых элементов;
  - comm – комуникатор.

Сборка выполняется всеми процессами группы. Поэтому в списке параметров отсутствует параметр root

Функции распределения блоков данных по всем процессам группы

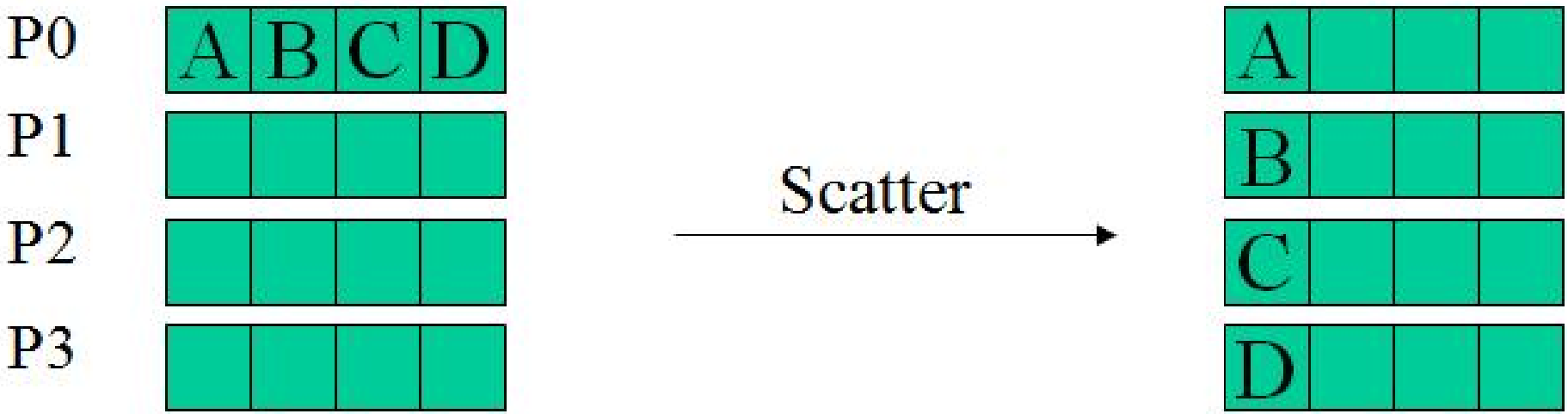
- MPI_Scatter.
- MPI_Scaterv.

MPI_Scatter

- int MPI_Scatter(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm);
  - sendbuf – адрес начала размещения блоков распределяемых данных (используется только в процессе-отправителе root);
  - sendcount – число посылаемых элементов;
  - sendtype – тип посылаемых элементов;
  - recvbuf – адрес начала буфера приема;
  - recvcount – число получаемых элементов;
  - recvtype – тип получаемых элементов;
  - root – номер процесса-отправителя;
  - comm – комуникатор.

Следует обратить внимание, что значение sendcount в вызове из процесса root – это число посылаемых каждому процессу элементов, а не общее их количество. Операция Scatter является обратной по отношению к Gather

Графическая интерпретация



Пример

```
MPI_Comm comm;
int  rbuf[100], size;
int  root, *array;
. . . . . .
MPI_Comm_size(comm, &size);
array = (int*)malloc(size * 100 * sizeof(int));
. . . . . .
MPI_Scatter(array, 100, MPI_INT,
            rbuf, 100, MPI_INT, root, comm);
```

MPI_Scatterv

- int MPI_Scatterv(void* sendbuf, int *sendcounts, int *displs, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm);
  - sendbuf – адрес начала размещения блоков распределяемых данных (используется только в процессе-отправителе root);
  - sendcounts – целочисленный массив (размер равен числу процессов в группе), содержащий число элементов, посылаемых каждому процессу;
  - displs – целочисленный массив (размер равен числу процессов в группе), i-ое значение определяет смещение относительно начала sendbuf для данных, посылаемых процессу i;
  - sendtype – тип посылаемых элементов;
  - recvbuf – адрес начала буфера приема;
  - recvcount – число получаемых элементов;
  - recvtype – тип получаемых элементов;
  - root – номер процесса-отправителя;
  - comm – комуникатор.

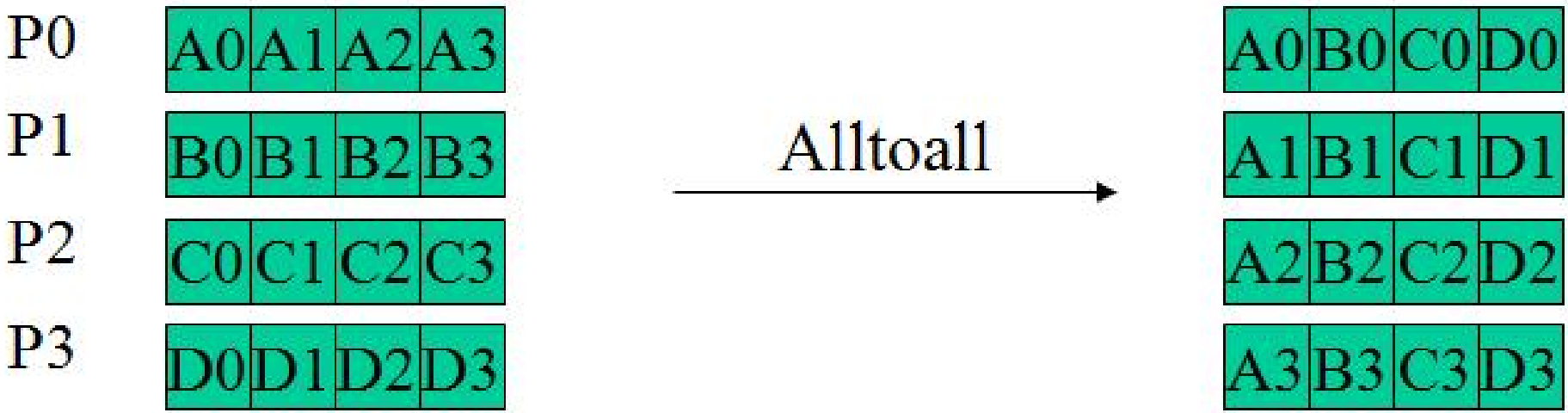
Данные к i-му, в массиве смещений displs, а число элементов – в sendcounts

Совмещенные коллективные операции

- int MPI_Alltoall(void* sendbuf, int sendcount, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm);
  - sendbuf – адрес начала буфера посылки;
  - sendcount – число посылаемых элементов;
  - sendtype – тип посылаемых элементов;
  - recvbuf – адрес начала буфера приема;
  - recvcount – число элементов, получаемых от каждого процесса;
  - recvtype – тип получаемых элементов;
  - comm – комуникатор.

Совмещает в себе операции Scatter и Gather и является по сути дела расширением операции Allgather, когда каждый процесс посылает различные данные разным получателям. Процесс i посылает j-ый блок своего буфера sendbuf процессу j, который помещает его в i-ый блок своего буфера recvbuf. Количество посланных данных должно быть равно количеству полученных данных для каждой пары процессов.

Графическая интерпретация



Глобальные вычислительные операции над распределенными данными

Операция редукции

Операция, аргументом которой является вектор, а результатом – скалярная величина, полученная применением некоторой математической операции ко всем компонентам вектора.

Глобальная (параллельная) операция редукции – компоненты вектора расположены в адресных пространствах процессов, выполняющихся на различных процессорах.

Глобальные операции редукции в MPI

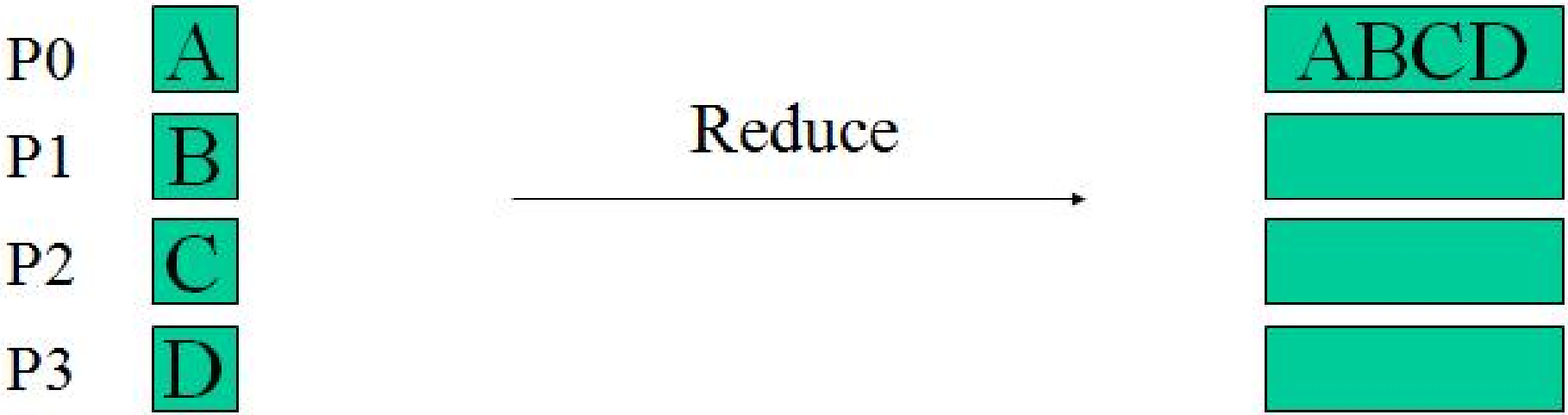
- с сохранением результата в адресном пространстве одного процесса (MPI_Reduce).
- с сохранением результата в адресном пространстве всех процессов (MPI_Allreduce).
- префиксная операция редукции, которая в качестве результата операции возвращает вектор. i-я компонента этого вектора является результатом редукции первых i компонент распределенного вектора (MPI_Scan).
- совмещенная операция Reduce/Scatter (MPI_Reduce_scatter).

MPI_Reduce

- int MPI_Reduce(void* sendbuf, void* recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm);
  - sendbuf – адрес начала входного буфера;
  - recvbuf – адрес начала буфера результатов (используется только в процессе-получателе root);
  - count – число элементов во входном буфере;
  - datatype – тип элементов во входном буфере;
  - op – операция, по которой выполняется редукция;
  - root – номер процесса-получателя результата операции;
  - comm – комуникатор.

Операция глобальной редукции, указанная параметром op, выполняется над первыми элементами входного буфера, и результат посылается в первый элемент буфера приема процесса root. Затем то же самое делается для вторых элементов буфера и т.д.

Графическая интерпретация



Особенности операций

- качестве операции op можно использовать либо одну из предопределенных операций, либо операцию, сконструированную пользователем;
- предопределенные операции являются ассоциативными и коммутативными;
- пользовательская операция, по крайней мере, должна быть ассоциативной;
- порядок редукции определяется номерами процессов в группе;
- datatype элементов должен быть совместим с операцией op.

Предопределенные операции в функциях редукции MPI

Название	Операция	Разрешенные типы
MPI_MAX	Максимум	C integer, Floating poin
MPI_MIN	Минимум	C integer, Floating poin
MPI_SUM	Сумма	C integer, Floating poin
MPI_PROD	Произведение	C integer, Floating poin
MPI_LAND	Логическое AND	C integer
MPI_LOR	Логическое OR	C integer
MPI_LXOR	Логическое исключающее OR	C integer
MPI_BAND	Поразрядное AND	C integer, Byte
MPI_BOR	Поразрядное OR	C integer, Byte
MPI_BXOR	Поразрядное исключающее OR	C integer, Byte
MPI_MAXLOC	Максимальное значение и его индекс	Специальные типы дл функций
MPI_MINLOC	Минимальное значение и его индекс	Специальные типы дл функций

Расшифровка типов

Название	Значения
C integer	MPI_INT, MPI_LONG, MPI_SHORT, MPI_UNSIGNED_SHORT, MPI_UNSIGNED, MPI_UNSIGNED_LONG
Floating point	MPI_FLOAT, MPI_DOUBLE, MPI_LONG_DOUBLE
Byte	MPI_BYTE

Типы для MAXLOC, MINLOC

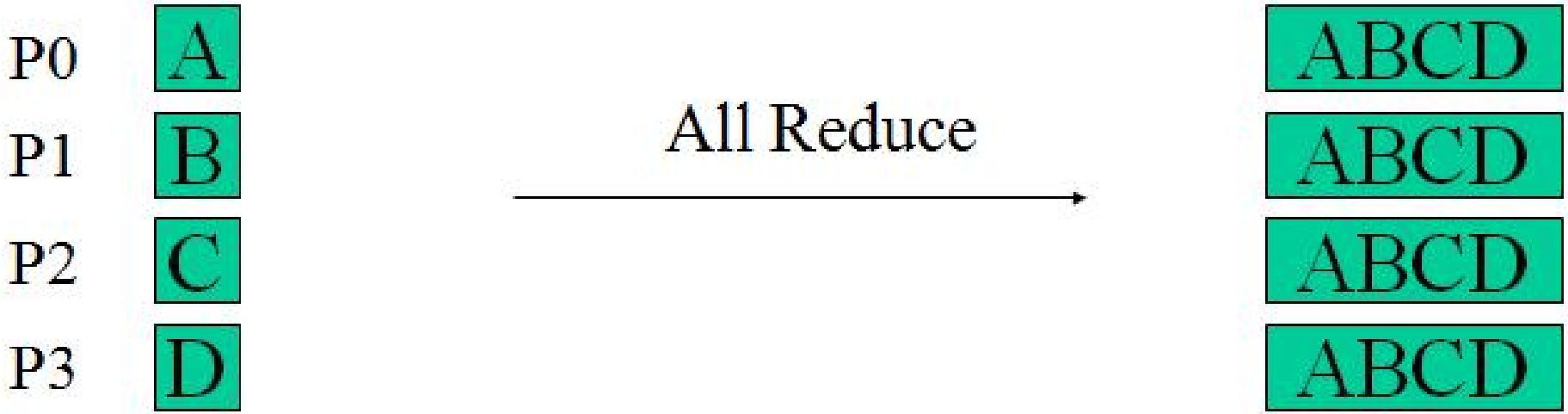
Название	Значения
MPI_FLOAT_INT	float and int
MPI_DOUBLE_INT	double and int
MPI_LONG_INT	long and int
MPI_2INT	int and int
MPI_SHORT_INT	short and int
MPI_LONG_DOUBLE_INT	long double and int

MPI_Allreduce

- int MPI_Allreduce(void* sendbuf, void* recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm);
  - sendbuf – адрес начала входного буфера;
  - recvbuf – адрес начала буфера результатов;
  - count – число элементов во входном буфере;
  - datatype – тип элементов во входном буфере;
  - op – операция, по которой выполняется редукция;
  - comm – комуникатор.

Сохраняет результат редукции в адресном пространстве всех процессов, поэтому в списке параметров функции отсутствует идентификатор корневого процесса root.

Графическая интерпретация



MPI_Reduce_scatter

- int MPI_Reduce_scatter(void* sendbuf, void* recvbuf, int *recvcounts, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm);
  - sendbuf – адрес начала входного буфера;
  - recvbuf – адрес начала буфера результатов;
  - recvcounts – массив, в котором задаются размеры блоков, посылаемых процессам;
  - datatype – тип элементов во входном буфере;
  - op – операция, по которой выполняется редукция;
  - comm – комуникатор.

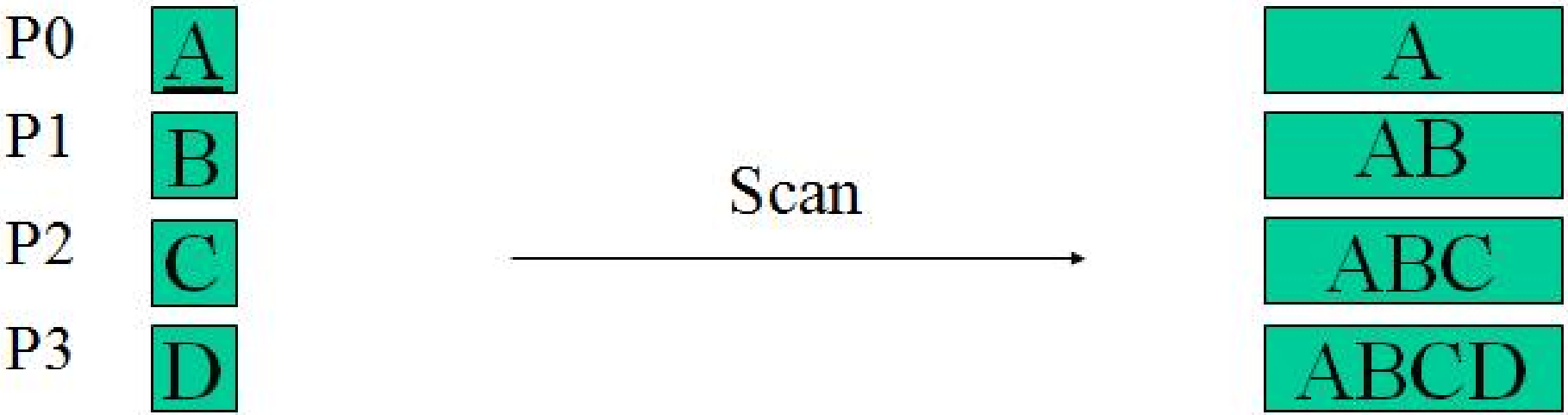
Функция MPI_Reduce_scatter отличается от MPI_Allreduce тем, что результат операции разрезается на непересекающиеся части по числу процессов в группе, i-ая часть посылается i-ому процессу в его буфер приема. Длины этих частей задает третий параметр, являющийся массивом.

MPI_Scan

- int MPI_Scan(void* sendbuf, void* recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm);
  - sendbuf – адрес начала входного буфера;
  - recvbuf – адрес начала буфера результатов;
  - count – число элементов входного буфера;
  - datatype – тип элементов во входном буфере;
  - op – операция, по которой выполняется редукция;
  - comm – комуникатор.

Выполняет префиксную редукцию. Параметры такие же, как в MPI_Allreduce, но получаемые каждым процессом результаты отличаются друг от друга. Операция пересылает в буфер приема i-го процесса редукцию значений из входных буферов процессов с номерами 0, ... i включительно.

Графическая интерпретация



Задание. Игра "Жизнь"

- Реализовать сбор данных используя коллективные операции MPI.
- Отметка 1 балл.

Спасибо за внимание! Вопросы?