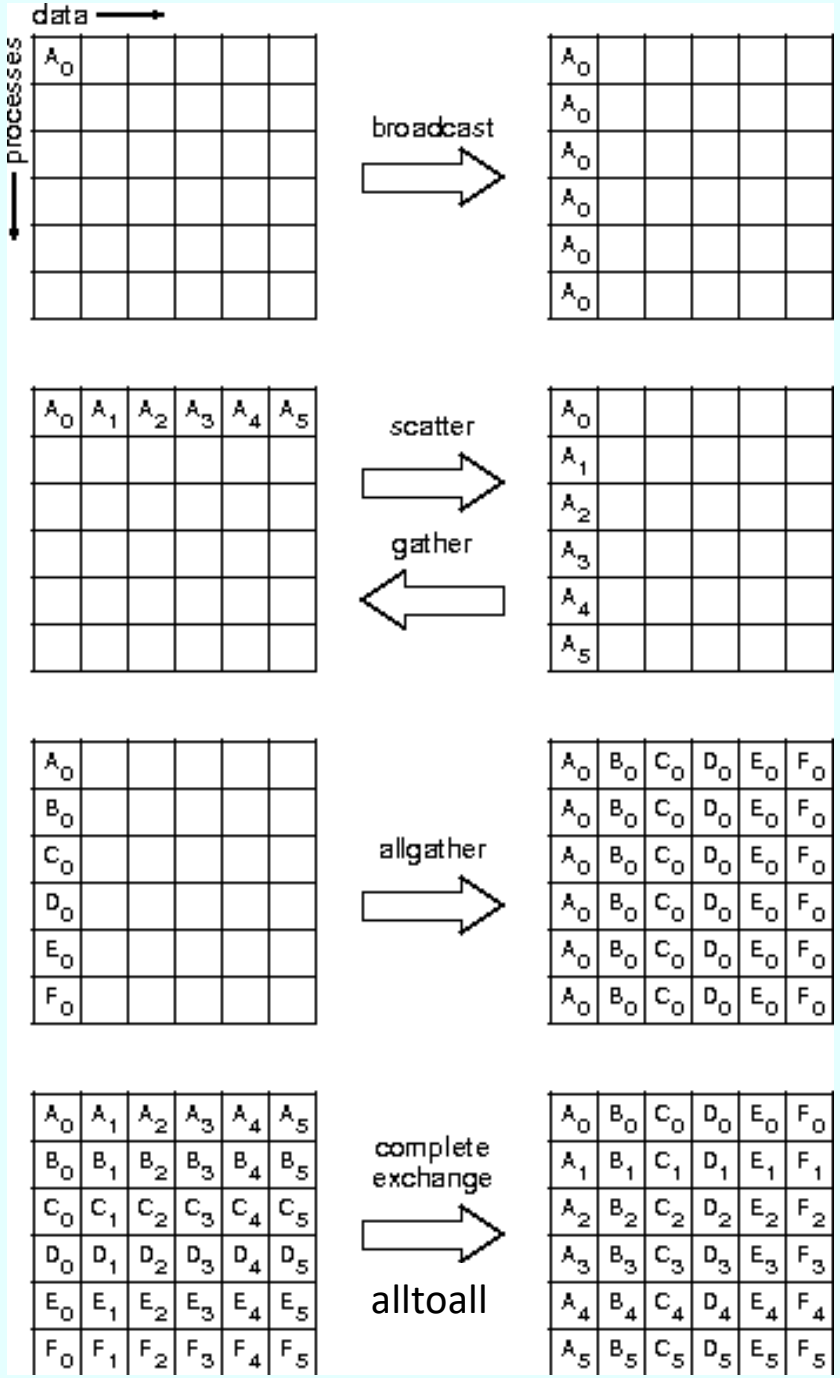
Технологии параллельного программирования на C++

Семинар 9

MPI

Коллективные операции



Сборка и рассылка данных

- **int MPI_Allgather(void *sbuf, int scount, MPI_Datatype stype, void *rbuf, int rcount, MPI_Datatype rtype, MPI_Comm comm)**

  sbuf - адрес начала буфера посылки

  scount - число элементов в посылаемом сообщении

  stype - тип элементов отсылаемого сообщения

  rbuf –адрес начала буфера сборки получаемых данных

  rcount - число элементов в принимаемом сообщении

  rtype - тип элементов принимаемого сообщения

  comm - идентификатор коммуникатора
- Сборка данных со всех процессов в буфере **rbuf** всех процессов. Данные сохраняются в порядке возрастания номеров процессов.

Сборка и рассылка данных

- **int MPI_Alltoall(void* sbuf, int scount, MPI_Datatype stype, void* rbuf, int rcount, MPI_Datatype rtype, MPI_Comm comm)**

  sbuf - адрес начала буфера посылки

  scount - число элементов в посылаемом сообщении

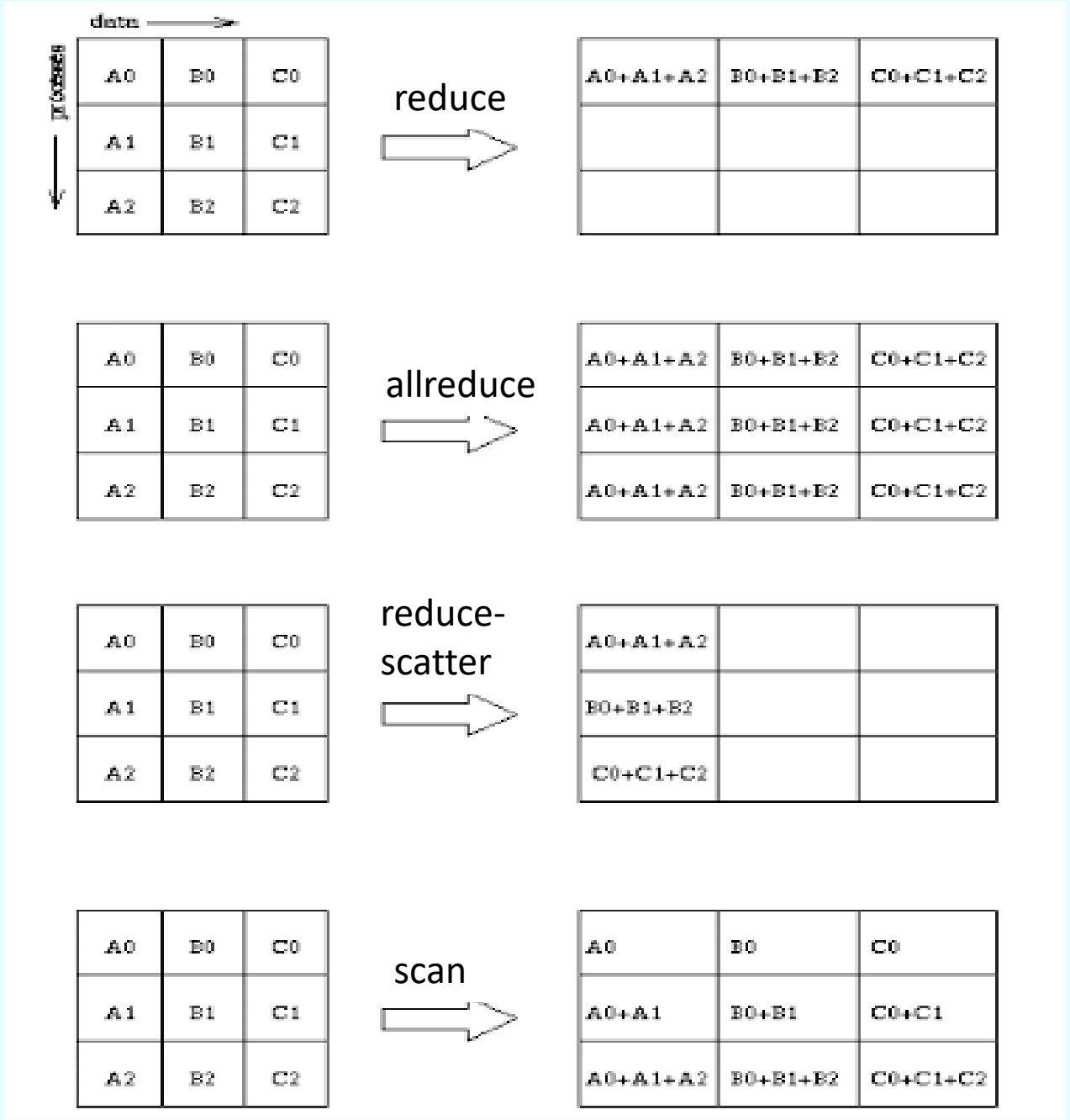
  stype - тип элементов отсылаемого сообщения

  rbuf - адрес начала буфера сборки получаемых данных

  rcount - число элементов в принимаемом сообщении

  rtype - тип элементов принимаемого сообщения

  comm - идентификатор коммуникатора
- Рассылка каждым процессом различных блоков данных всем другим процессам. j-й блок данных i-го процесса попадает в i-й блок j-го процесса.
- Можно использовать **MPI_IN_PLACE** вместо **sbuf** на всех процессах, тогда для пересылки будут использоваться данные из **rbuf**, после чего они будут перезаписаны данными, полученными от соответствующего процесса. При этом **scount** и **stype** игнорируются. При этом **rcount** и **rtype** должны быть одинаковыми.


data
processes
A0 B0 C0
A1 B1 C1
A2 B2 C2
reduce
A0+A1+A2 B0+B1+B2 C0+C1+C2
A0 B0 C0
A1 B1 C1
A2 B2 C2
allreduce
A0+A1+A2 B0+B1+B2 C0+C1+C2
A0+A1+A2 B0+B1+B2 C0+C1+C2
A0+A1+A2 B0+B1+B2 C0+C1+C2
A0 B0 C0
A1 B1 C1
A2 B2 C2
reduce-
scatter
A0+A1+A2
B0+B1+B2
C0+C1+C2
A0 B0 C0
A1 B1 C1
A2 B2 C2
scan
A0 B0 C0
A0+A1 B0+B1 C0+C1
A0+A1+A2 B0+B1+B2 C0+C1+C2

Операции редукции

- **int MPI_Allreduce(void *sbuf, void *rbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)**

  sbuf - адрес начала буфера для аргументов

  rbuf - адрес начала буфера для результата

  count - число аргументов у каждого процесса

  datatype - тип аргументов

  op - идентификатор глобальной операции

  comm - идентификатор коммуникатора

- Выполнение **count** глобальных операций **op** с возвратом **count** результатов во всех процессах в буфере **rbuf**. Операция выполняется независимо над соответствующими аргументами всех процессов. Значения параметров **count**, **datatype** и **comm** у всех процессов должны быть одинаковыми.
- На всех процессах вместо **sbuf** можно использовать **MPI_IN_PLACE**, в этом случае содержимое **sbuf** должно находиться в **rbuf**.

Операции редукции

- **int MPI_Reduce_scatter(void* sbuf, void* rbuf,const int rcounts[], MPI_Datatype datatype, MPI_Op op,MPI_Comm comm)**

  sbuf - адрес начала буфера для аргументов

  rbuf - адрес начала буфера для результата

  rcounts[] - массив, **rcounts[i]** - число результатов глобальной операции, размещаемых в буфере **rbuf** i-гопроцесса

  datatype - тип аргументов

  op - идентификатор глобальной операции

  comm - идентификатор коммуникатора
- Выполнение $\sum_{i=0}^{n-1}$ **rcounts[i]** глобальных операций **op** с возвратом **rcounts[i]** результатов в буфере **rbuf** процесса **i**. Операция выполняется независимо над соответствующими аргументами всех процессов. Значения параметров **rcounts**, **datatype** и **comm** у всех процессов должны быть одинаковыми.
- На всех процессах вместо **sbuf** можно использовать **MPI_IN_PLACE**, в этом случае содержимое **sbuf** должно находиться в **rbuf**. **MPI_IN_PLACE** не нужно использовать на процессе **i** с **rcounts[i]=0** (т.к. место под rbuf можно не выделять).

Операции редукции

- **int MPI_Scan(void* sbuf, void* rbuf, int count,MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)**

  sbuf - адрес начала буфера для аргументов

  rbuf - адрес начала буфера для результата

  count - число аргументов у каждого процесса

  datatype - тип аргументов

  op - идентификатор глобальной операции

  comm - идентификатор коммуникатора
- Выполнение **count** частичных глобальных операций op над соотвествующими элементами буфера **sbuf**. i-й процесс выполняет глобальную операцию над соответствующими элементами массива **sbuf** процессов 0…i и помещает результат в массив **rbuf**. Окончательный результат глобальной операции получается в массиве **rbuf** последнего процесса.
- Значения параметров **count**, **datatype**, **op** и **comm** у всех процессов должны быть одинаковыми. На всех процессах вместо **sbuf** можно использовать **MPI_IN_PLACE**, в этом случае содержимое **sbuf** должно находиться в **rbuf**.

Создание пользовательской глобальной операции MPI

- **int MPI_Op_create(MPI_User_function* user_fn, int commute, MPI_Op* op)**

  user_fn - пользовательская функция операции редукции

  commute - флаг коммутативности операции

  op - идентификатор глобальной операции
- Создание пользовательской глобальной операции. Если **commute= true**, то операция должна быть коммутативной и ассоциативной. Иначе порядок фиксируется по увеличению номеров процессов. Порядок выполнения может быть изменён, используя свойство ассоциативности операции, если **commute= false**.
- **typedef void MPI_User_function(void* invec, void* inoutvec, int *len, MPI_Datatype *datatype);**
- **int MPI_Op_free(MPI_Op *op)**

  Удаление пользовательской глобальной операции

Задача

- Найти норму вектора с помощью коллективных операций
  (создать глобальную операцию)