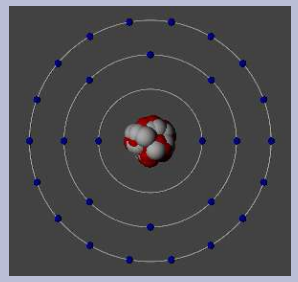
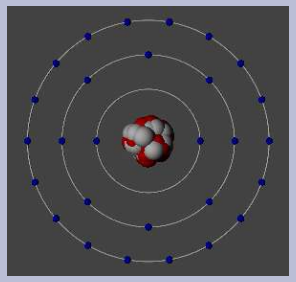


Паралелно програмиране с MPI

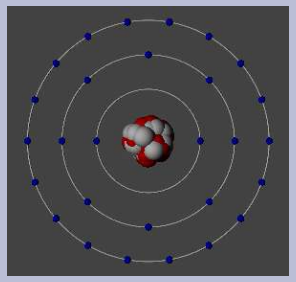
Неблокиращи операции. Колективни комуникации – синхронизация, разпределяне на работата, редукции.



Неблокиращи операции

- Базов метод за избягване на взаимно заключване и реализация на асинхронност в алгоритмите.
- Функциите връщат управление веднага след иницииране на операцията.

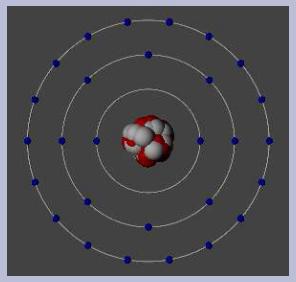
	Неблокираща форма
Стандартно изпращане	**MPI_Isend**
Синхронно изпращане	**MPI_Issend**
Буферирано изпращане	**MPI_Bsend**
Изпращане с готовност	**MPI_Rsend**
Приемане	**MPI_Irecv**



Неблокиращо изпращане

MPI_Issend (buf, count, datatype, src, tag, comm, request)

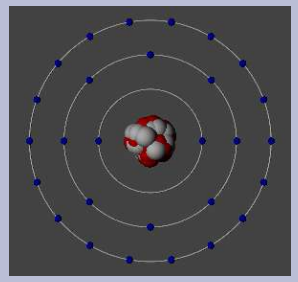
- Същите параметри, както и блокиращата версия с допълнителен изходен аргумент *request*.
- При връщане от функцията *request* съдържа комуникационнен манипулатор, който може да се използва във функциите за проверка на завършеност.
- buf не трябва да се променя преди завършване на операцията!!!



Неблокиращо приемане

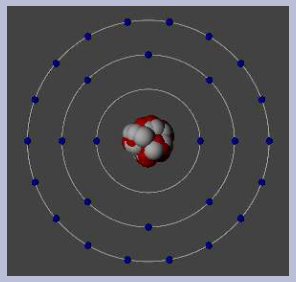
MPI_Irecv (buf, count, datatype, src, tag, comm, request)

- Изходният статусен аргумент е заменен с комуникационен манипулатор.
- Неблокиращата операция по приемане може да получава съобщения, изпратени както с неблокиращи, така и с блокиращи операции на изпращане.



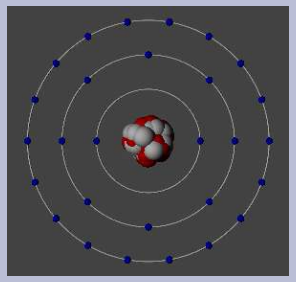
Тестване за завършеност

- Неблокиращите операции трябва да се тестват за завършеност, иначе не се гарантира физическото им приключване.
- Два класа тестващи функции:
  - изчакващ (WAIT) тип - блокират до приключване на операцията ***MPI_Wait (request, status)***
  - тестов (TEST) тип - връщат булев резултат, отговарящ на текущата завършеност на операцията ***MPI_Test (request, flag, status)***
- Версии ALL, ANY и SOME



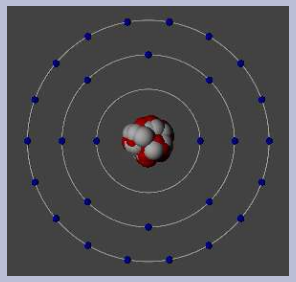
Синхронизация

- Синхронизацията в MPI може да бъде явна или неявна.
- Неявна синхронизация – чрез синхронизиращи примитиви като синхронното изпращане
- Явна синхронизация:
  - единствен явен синхронизиращ примитив ***MPI_Barrier (comm)***
  - бариера – блокира докато всички процеси в комуникатора *comm* не достигнат до точката на синхронизация



Глобална комуникация

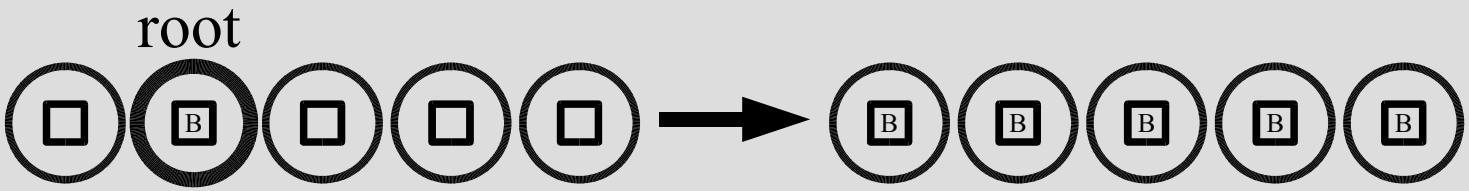
- В глобалната комуникация участват всички процеси в даден комуникатор.
- Три типа глобални комуникации:
  - глобално изпращане (broadcast)
  - разпределяне на работата (scatter/gather)
  - глобална редукция
- Всички глобални операции са синхронни и блокиращи.

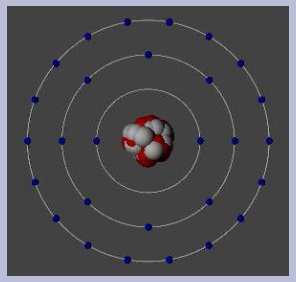


Глобално изпращане

MPI_Bcast (buf, count, datatype, root, comm)

- Инициатор на изпращането - *root*
- Процесът с ранк *root* изпраща данните, намиращи се в буфера buf
- Всички останали процеси получават данните в буфера buf

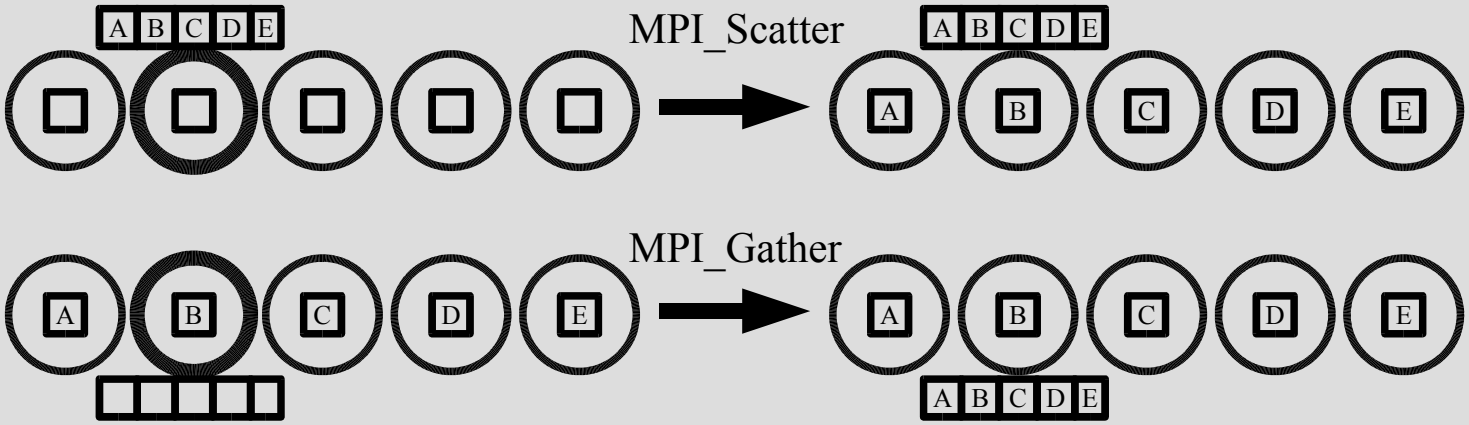


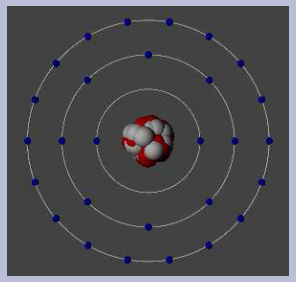


Разпределяне на работата

MPI_Scatter (sbuf, scount, sdtype, rbuf, rcount, rdtype, root, comm)

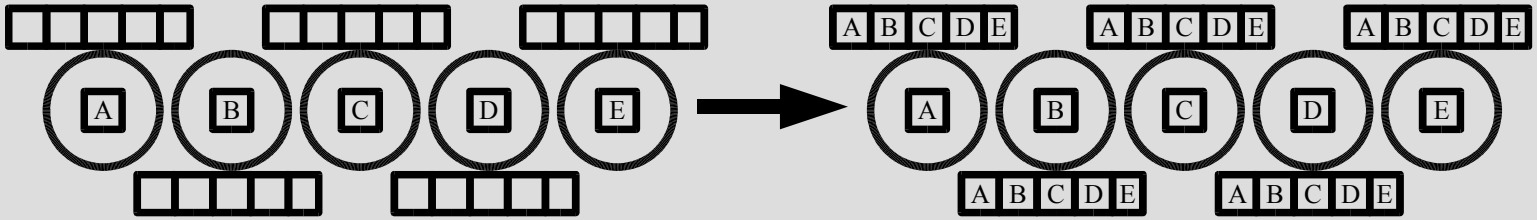
- Използва се за разбиване на буфера *sbuf* на *root* процеса в буферите *rbuf* на всички процеси в комуникатора.
- Аналогична функция ***MPI_Gather*** върши точно обраното.



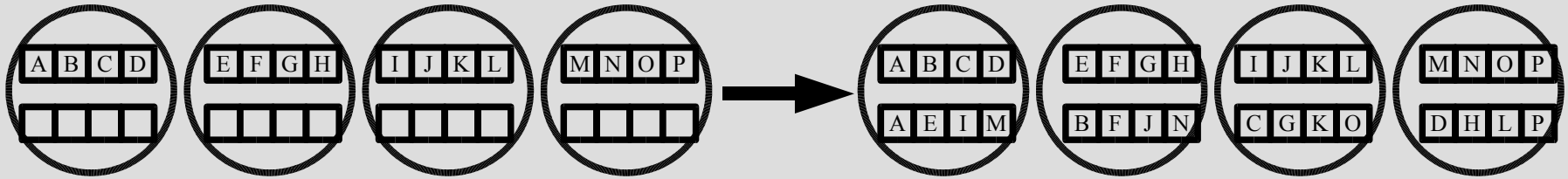


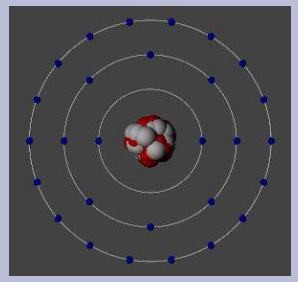
Други варианти

MPI_Allgather



MPI_Alltoall

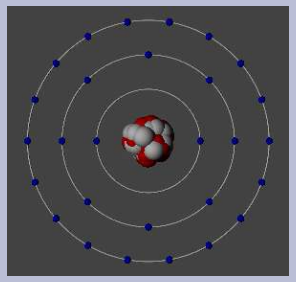




Глобална редукция

- Използва се за изчисляване на резултат, който зависи от данни, разпределени по всички процеси в даден комуникатор.
- Глобални операции:
  - аритметични (събиране, умножение)
  - логически и побитови (И, ИЛИ, Изкл. ИЛИ)
  - минимум/максимум и положение на елемента
  - потребителски
- Потребителските операции трябва да бъдат бинарни и асоциативни:

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

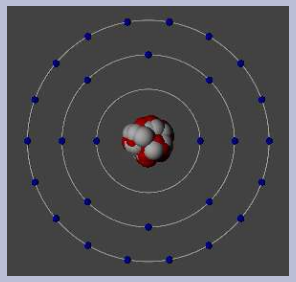


Глобална редукция (2)

MPI_Reduce (sbuf, rbuf, count, datatype, op, root, comm)

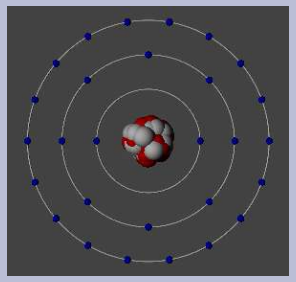
- Операторът *op* се прилага върху всички копия на *sbuf* и резултатът се получава в *rbuf* на процесът *root*.

- *op* е предефиниран MPI оператор, или манипулатор на потребителски такъв, регистриран с ***MPI_Op_create (function, commute, op)***



Предефинирани редукционни оператори в MPI

MPI оператор	Функция
MPI_MAX	Максимум
MPI_MIN	Минимум
MPI_SUM	Сума
MPI_PROD	Произведение
MPI_LAND	Логическо И
MPI_BAND	Побитово И
MPI_LOR	Логическо ИЛИ
MPI_BOR	Побитово ИЛИ
MPI_LXOR	Логическо Изк. ИЛИ
MPI_BXOR	Побитово Изк. ИЛИ
MPI_MAXLOC	Максимум и положение
MPI_MINLOC	Минимум и положение

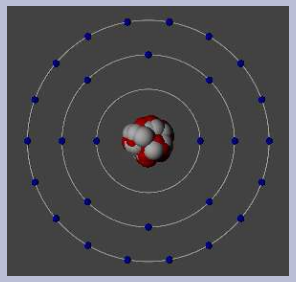


Потребителски редукционни оператори

- В програми на C:
  *typedef void MPI_User_function (void *invec, void *inoutvec, int *len, MPI_Datatype *datatype);*

- В програми на Fortran:
  SUBROUTINE USER_FUNCTION (INVEC(*), INOUTVEC(*), LEN, TYPE)
  <type> INVEC(LEN), INOUTVEC(LEN)
  INTEGER LEN, TYPE

- Потребителските оператори може да комутират, но може и да не комутират, като това се указва на ***MPI_Op_create***.



По-сложни редукции

- Глобалните редукции може да се комбинират с операции по разпределяне на работата.
- ***MPI_Allreduce*** – всички процеси получават пълния резултат
- ***MPI_Reduce_scatter*** – всеки процес получава част от пълния резулт
- ***MPI_Scan*** – всеки процес получава частична редукция, а само един – пълният резултат