Master in High Performance Computing

PARALLEL PROGRAMMING

Scuola Internazionale Superiore di Studi Avanzati

The Abdus Salam International Centre for Theoretical Physics

The MPI_BARRIER

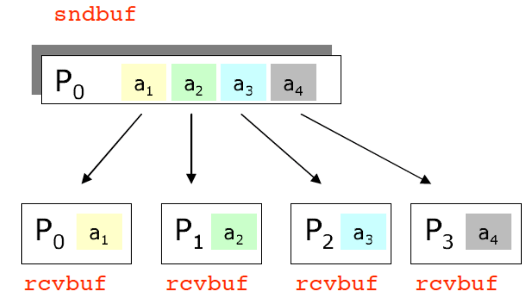
Blocks until all processes have reached this routine

```
INCLUDE ’mpif.h’

MPI_BARRIER(COMM, IERROR)

 INTEGER    COMM, IERROR
```

Master in High Performance Computing

Collective Communications

- At the bottom line are based on point 2 point (you could build your own)
- The MPI include a series of subroutines to handle different patterns of communications: 1 to N, N to 1 and N to N
- **Collective Communications imply a *synchronization point* among processes**

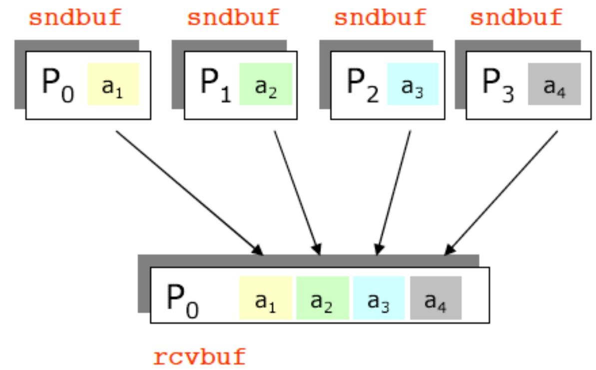


MPI_Scatter

One-to-all communication: different data sent from root process to all others in the communicator.

MPI_SCATTER(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, root, comm)
[IN sendbuf] starting address of send buffer (choice)
[IN sendcount] number of elements sent to each process (integer, significant only at root)
[IN sendtype] data type of send buffer elements (significant only at root) (handle)
[OUT recvbuf] address of receive buffer (choice)
[IN recvcount] number of elements in receive buffer (integer)
[IN recvtype] data type of recv buffer elements (handle)
[IN root] rank of receiving process (integer)
[IN comm] communicator (handle)

MPI_Gather



One-to-all communication: different data collected by the root process, from all others processes in the communicator.

It is the opposite of Scatter

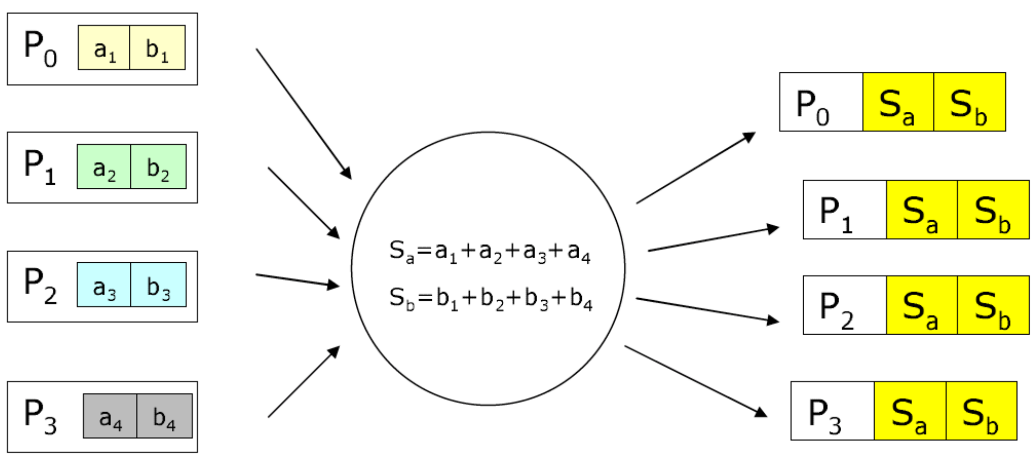
MPI_GATHER(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, root, comm)
[IN sendbuf] starting address of send buffer (choice)
[IN sendcount] number of elements in send buffer (integer)
[IN sendtype] data type of send buffer elements (handle)
[OUT recvbuf] address of receive buffer (choice, significant only at root)
[IN recvcount] number of elements for any single receive (integer, significant only at root)
[IN recvtype] data type of recv buffer elements (significant only at root) (handle)
[IN root] rank of receiving process (integer)
[IN comm] communicator (handle)

Reduction

The reduction operation allow to:

- Collect data from each process
- Reduce the data to a single value
- Store the result on the root processes
- Store the result on all processes
- **<u>Overlap of communication and computing</u>**

MPI op	Function
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Sum
MPI_PROD	Product
MPI_LAND	Logical AND
MPI_BAND	Bitwise AND
MPI_LOR	Logical OR
MPI_BOR	Bitwise OR
MPI_LXOR	Logical exclusive OR
MPI_BXOR	Bitwise exclusive OR
MPI_MAXLOC	Maximum and location
MPI_MINLOC	Minimum and location



MHPC
Master in High Performance Computing

Calling MPI_REDUCE

MPI_REDUCE(in, out, count, type, op, receiver, comm, err)

in:	data to be sent (from all)
out:	storage for reduced data (on receiver)
count:	number of data items to be reduced
type:	type (=size) of data items
op:	reduction operation, e.g. **MPI_SUM**
receiver:	rank of sending processor of data
communicator:	group identifier, **MPI_COMM_WORLD**
err:	error status or **MPI_SUCCESS**

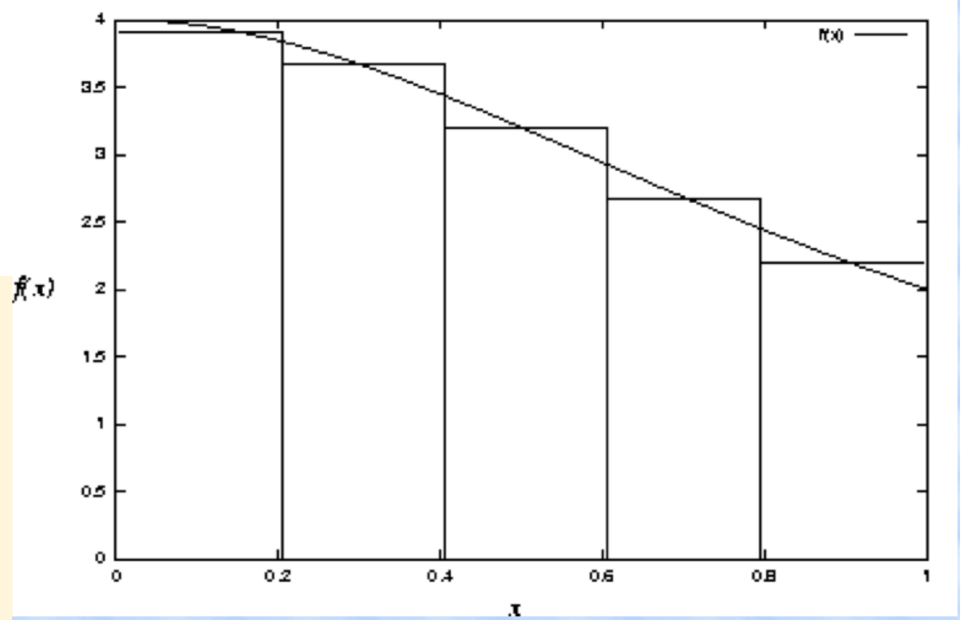
ma per seguir virtute e canoscenza
SISSA
Scuola Internazionale Superiore di Studi Avanzati

ICTP The Abdus Salam International Centre for Theoretical Physics

Approximate PI Using MPI collectives

$$\int_0^1 \frac{1}{1+x^2}dx = arctan(x)\Big|_0^1 = arctan(1) - arctan(0) = arctan(1) = \frac{\pi}{4}$$

$$\pi = 4\int_0^1 \frac{1}{1+x^2}dx$$



Integrate, i.e determine area under function numerically using slices of h * f(x) at midpoints

Ivan Girotto
igirotto@ictp.it

```
#include <stdio.h>

int main(){
 long n , i ;
 double  w,x,sum,pi,f,a;

 n = 100000000;
 w = 1.0/n;
 sum = 0.0;

 for ( i = 1 ; i <= n ; i++ ) {
     x = w * (i - 0.5);
     sum = sum + (4.0 / (1.0 + x * x ) );
  }

 pi = w * sum ;
 printf("Value of pi: %.16g\n", pi);

 return 0;
}
```

Master in High Performance Computing

Assignment

1) Implement the PI approximation in parallel using the Message Passing paradigm

SISSA
Scuola Internazionale Superiore di Studi Avanzati

The Abdus Salam International Centre for Theoretical Physics

STANDARD BLOCKING SEND - RECV

MPI_SEND(buf, count, type, dest, tag, comm, ierr)

MPI_RECV(buf, count, type, dest, tag, comm, status, ierr)

Buf array of MPI type **type**.

Count (INTEGER) number of element of **buf** to be sent/recv

Type (INTEGER) MPI type of **buf**

Dest (INTEGER) rank of the destination process

Tag (INTEGER) number identifying the message

Comm (INTEGER) communicator of the sender and receiver

*** Status** (INTEGER) array of size **MPI_STATUS_SIZE** containing communication status information

Ierr (INTEGER) error code

**** used only for receive operations***

NON-BLOCKING SEND - RECV

MPI_ISEND(buf, count, type, dest, tag, comm, request, ierr)

MPI_IRECV(buf, count, type, dest, tag, comm, request, ierr)

Buf array of MPI type **type**.

Count (INTEGER) number of element of **buf** to be sent/recv

Type (INTEGER) MPI type of **buf**

Dest (INTEGER) rank of the destination process

Tag (INTEGER) number identifying the message

Comm (INTEGER) communicator of the sender and receiver

Request (INTEGER) request handler, used for checking the communication status

Ierr (INTEGER) error code

No-Blocking Checkpoint

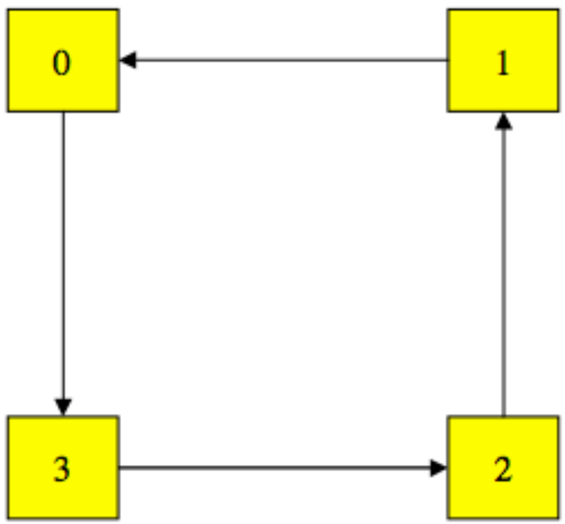
MPI_WAIT(request, status, ierr)

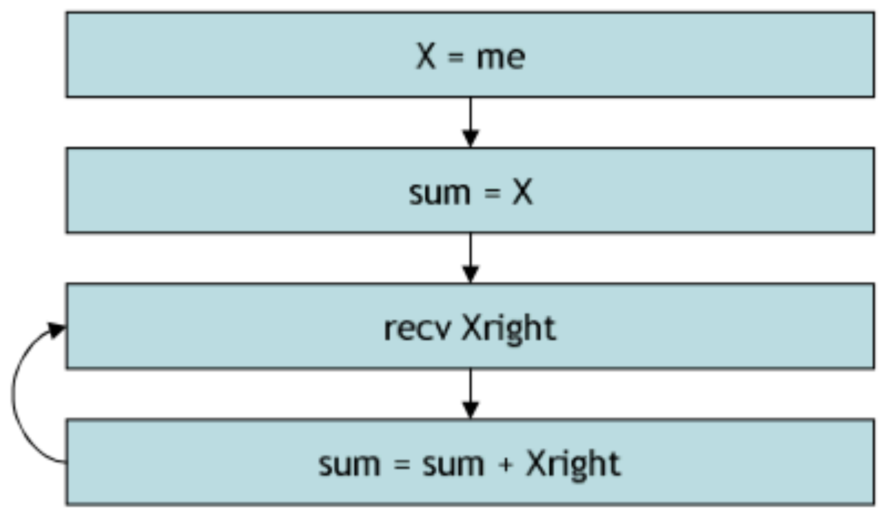
Request (INTEGER) request handler, used for checking the communication status

Status (INTEGER) array of size **MPI_STATUS_SIZE** containing communication status information

Ierr (INTEGER) error code

Wait until the communication handled by the object request is terminated. For test only use MPI_TEST, for checkpoint of many communication use MPI_WAITALL





sum = 6 (on 4 processors)

Who/when does the SEND?

What does it send?

How many times?

Master in High Performance Computing

Exercise

Implement the proposed exercise, first exchanging one single element (mype) among processes as illustrated in class as well as on the previous slide. Try to optimize the code for sending in the ring a large set of data and overlapping the computation (Σ) and the communication (send-recv). In case of a dataset larger than one element the local sum is considered a vector sum (element by element).

ma per seguir virtute e canoscenza
SISSA
Scuola Internazionale Superiore di Studi Avanzati

ICTP The Abdus Salam International Centre for Theoretical Physics

External MPI Resources

Here are some links to tutorials and literature:

- CI-Tutor at NCSA: http://www.citutor.org/
- MPI reference and mini tutorial at LLNL: http://computing.llnl.gov/tutorials/mpi/
- Designing and Building // Programs, by Ian Foster: http://www.mcs.anl.gov/~itf/dbpp/
- MPI standards: http://www.mpi-forum.org/
- OpenMPI: http://www.open-mpi.org
- MPICH: http://www.mcs.anl.gov/research/projects/mpich2