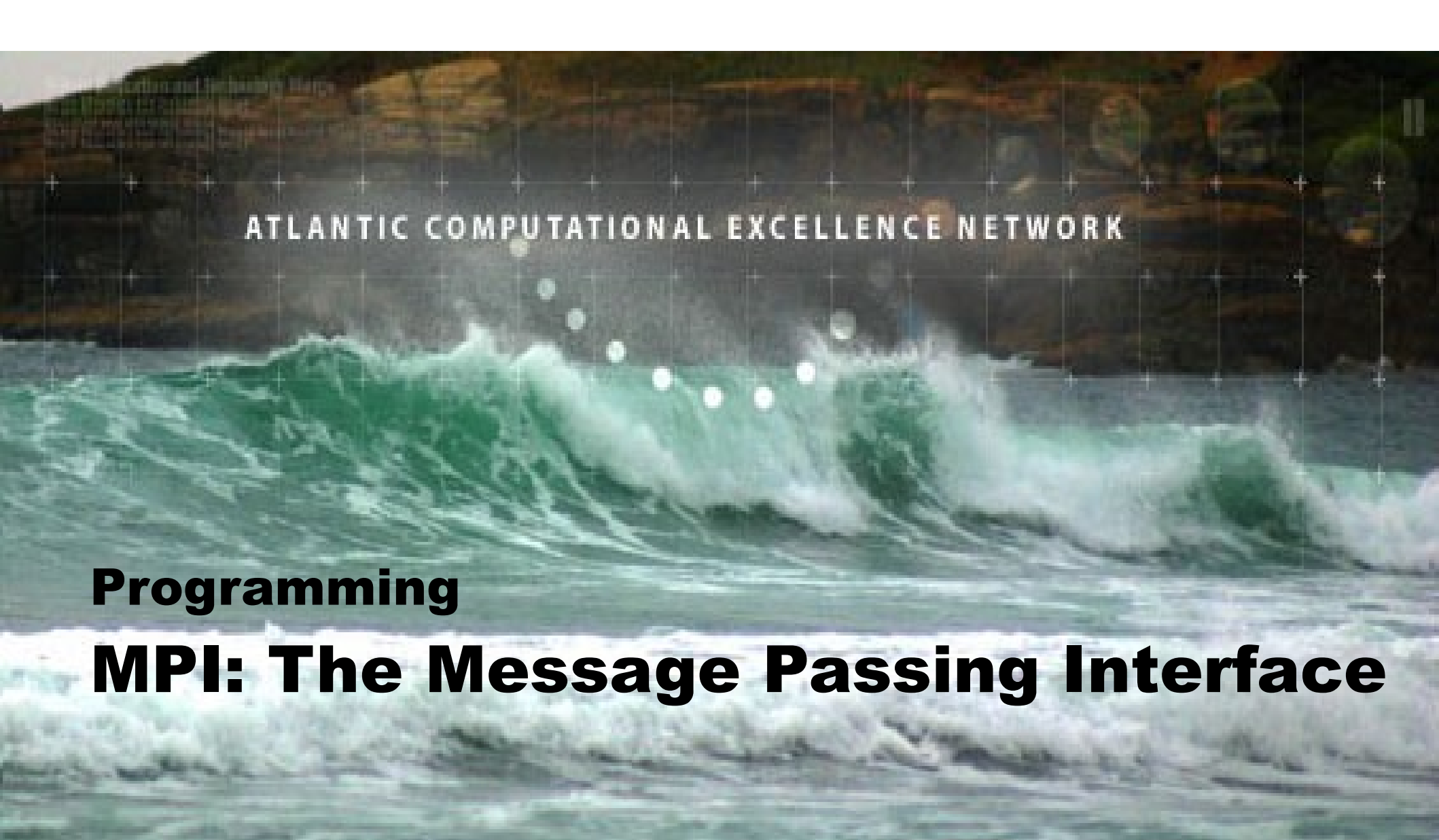




ATLANTIC COMPUTATIONAL EXCELLENCE NETWORK

Programming

MPI: The Message Passing Interface

Ross Dickson, Computational Research Consultant

What is MPI?

- Message Passing Interface
- *A standard*, not a product
 - First published 1994, MPI-2 published 1997
- *De facto* standard for distributed-memory parallel programming
- Many implementations:
 - Open MPI – MPICH – LAM-MPI –
– MVAPICH – others –
- Fortran, C and C++ bindings in standard
 - Python, Ocaml, .NET & others exist

Hello, Parallel World!

hello_mpi.c

```
#include "mpi.h"
#include <stdio.h>

int main( int argc, char *argv[] )
{
    MPI_Init( &argc, &argv );
    printf( "Hello, Parallel World!\n" );
    MPI_Finalize();
    return 0;
}
```

...but MPI does not *guarantee* that each process can write to stdout!

Building and Running (with Open MPI at ACEnet)

hello_mpi.c

```
$ which mpicc  
/usr/local/openmpi/bin/mpicc  
$ mpicc hello_mpi.c -o hello  
$ mpirun -np 2 hello  
Hello, Parallel World!  
Hello, Parallel World!  
$
```

Integration between Open MPI and Sun Grid Engine means
“-np 2” is unneeded inside a job script. Process count is set by
parallel environment “qsub -pe omp 2 ...”

Concepts

- Single-program, multiple-data (SPMD)
 - MPMD also supported but rarely used
- Point-to-point communications
 - MPI_Send, MPI_Recv
- Collective communications
 - MPI_Reduce, MPI_Bcast, MPI_Scatter...
- “Communicators”
 - MPI_COMM_WORLD
- Parallel I/O

Process rank and count

rank.f

```
program myrank
implicit none
include 'mpif.h'
integer iError,myRank,nProcs

call MPI_Init(iError)
call MPI_Comm_Rank(MPI_COMM_WORLD,myRank,iError)
call MPI_Comm_Size(MPI_COMM_WORLD,nProcs,iError)
write(*,*) 'This is process ',myRank,' of ',nProcs
call MPI_Finalize(iError)

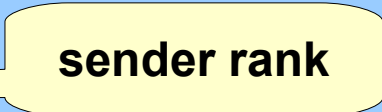
end
```

MPI_Comm_Rank returns different **rank** for each process
MPI_Comm_Size returns total number of processors

MPI_Send & MPI_Recv



rank.c

```
if (myRank != 0) {
    sprintf( msg, "Hello from process %d\n", myRank);
    MPI_Send( msg, NCHARS, MPI_CHAR,
              0, 
              TAG, MPI_COMM_WORLD );
}
else {
    for (source=1; source<nProcs; source++) {
        MPI_Recv( msg, NCHARS, MPI_CHAR,
                  source, 
                  TAG, MPI_COMM_WORLD, &status );
        printf( "%s", msg );
    }
    printf( "...and hello from rank %d.\n", myRank );
}
```

Point-to-point communication



- Every MPI_Send must match an MPI_Recv
- Message “envelope” consists of
 - Sender rank
 - Receiver rank
 - Tag (arbitrary integer)
 - Data

Who does what?



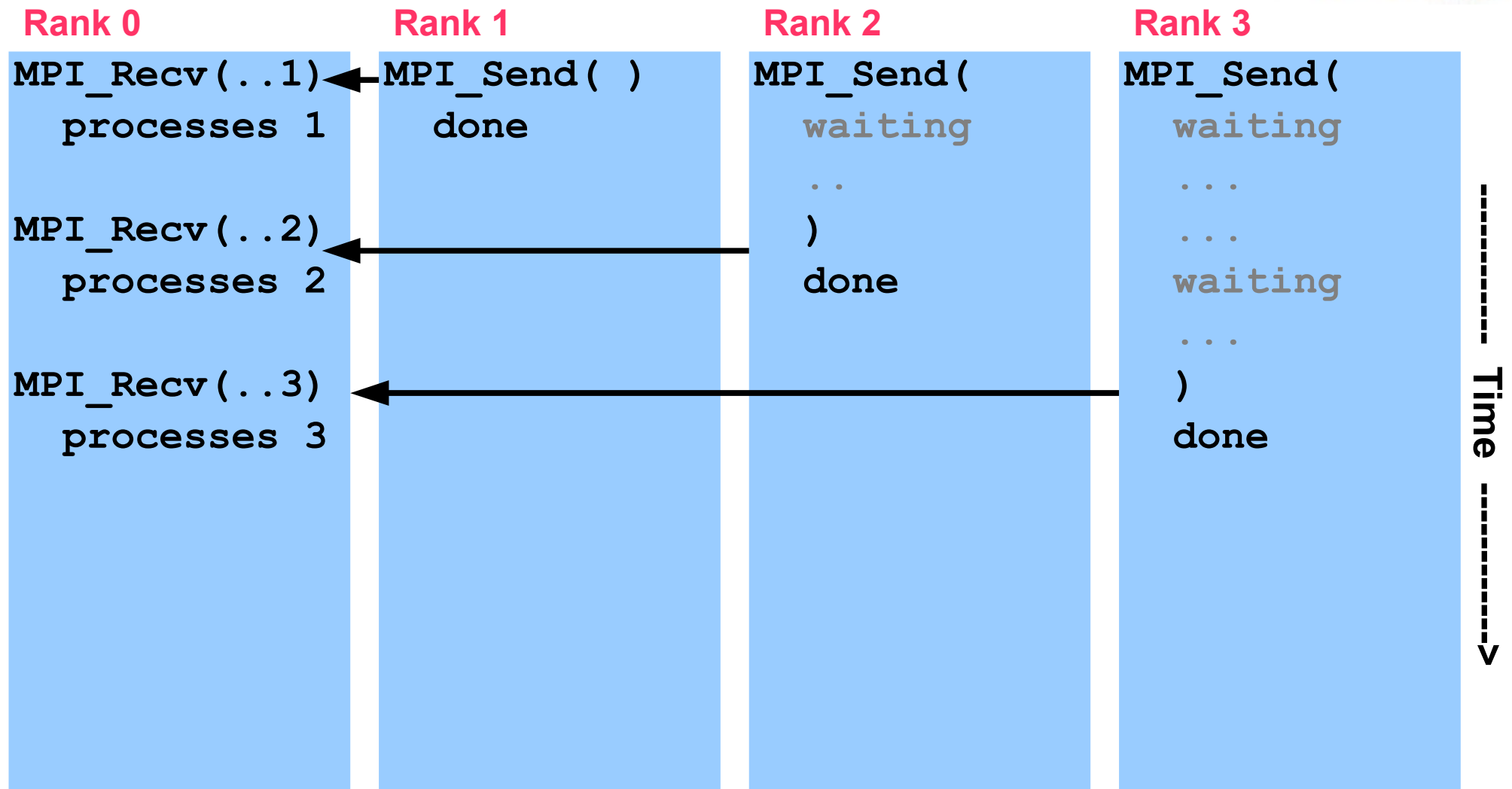
Rank 1, 2, 3, ...

```
if (myRank != 0) {    // WORKER
    sprintf(&msg, "%d\n", myRank);
    MPI_Send(msg, NCHARS,
             MPI_CHAR, 0, TAG,
             MPI_COMM_WORLD);
}
else { // myRank == 0
    for (src=1; src<n; src++) {
        MPI_Recv(msg, NCHARS,
                 MPI_CHAR, src, TAG,
                 MPI_COMM_WORLD,
                 &status);
        printf(" %s", msg );
    }
    printf("...and hello from
           rank 0.\n");
}
```

Rank 0

```
if (myRank != 0) {
    sprintf(&msg, "%d\n", myRank);
    MPI_Send(msg, NCHARS,
             MPI_CHAR, 0, TAG,
             MPI_COMM_WORLD);
}
else {                // MASTER
    for (src=1; src<n; src++) {
        MPI_Recv(msg, NCHARS,
                 MPI_CHAR, src, TAG,
                 MPI_COMM_WORLD,
                 &status);
        printf(" %s", msg );
    }
    printf("...and hello from
           rank 0.\n");
}
```

Slow motion replay

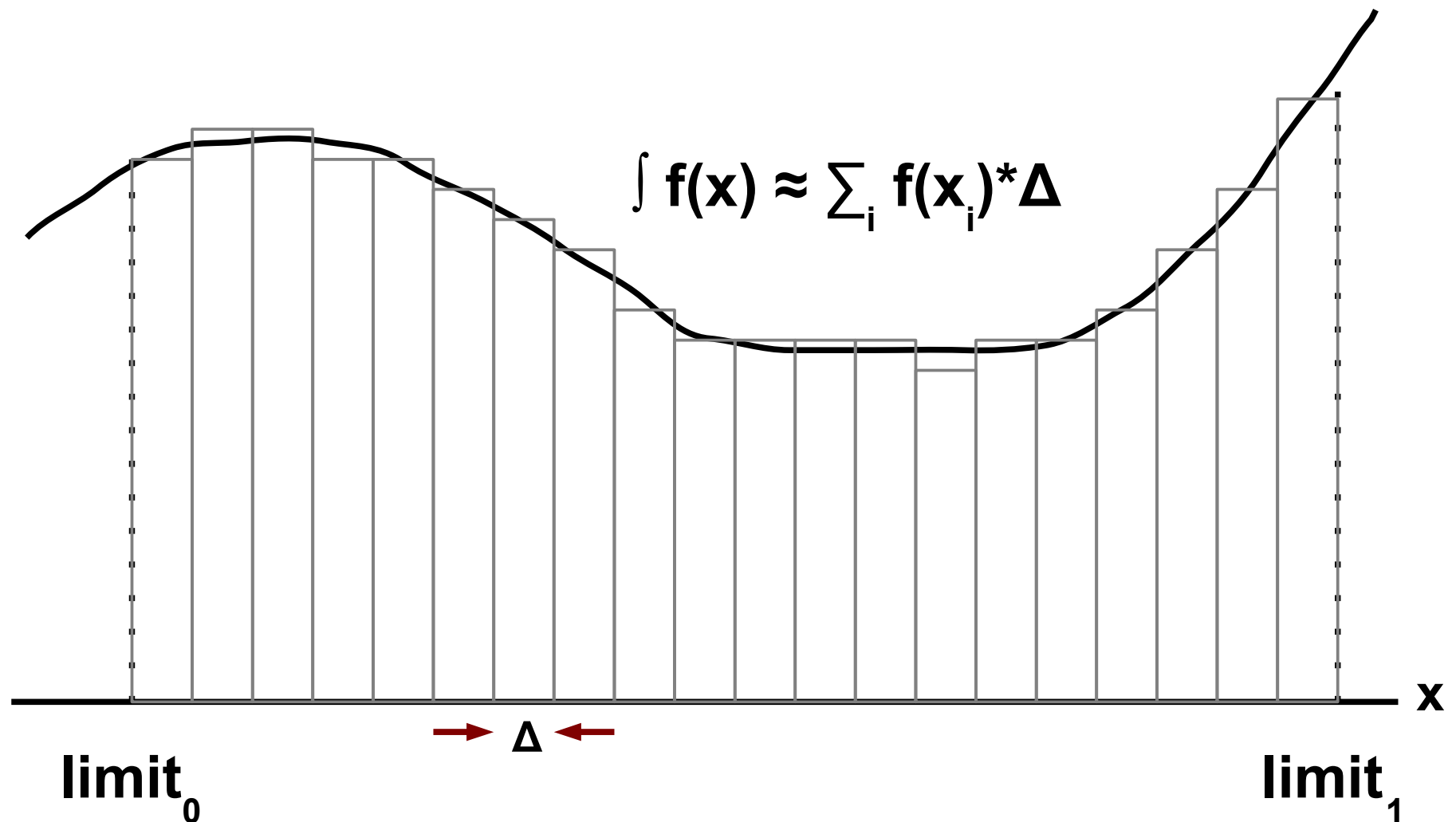


Look at all that time spent waiting!

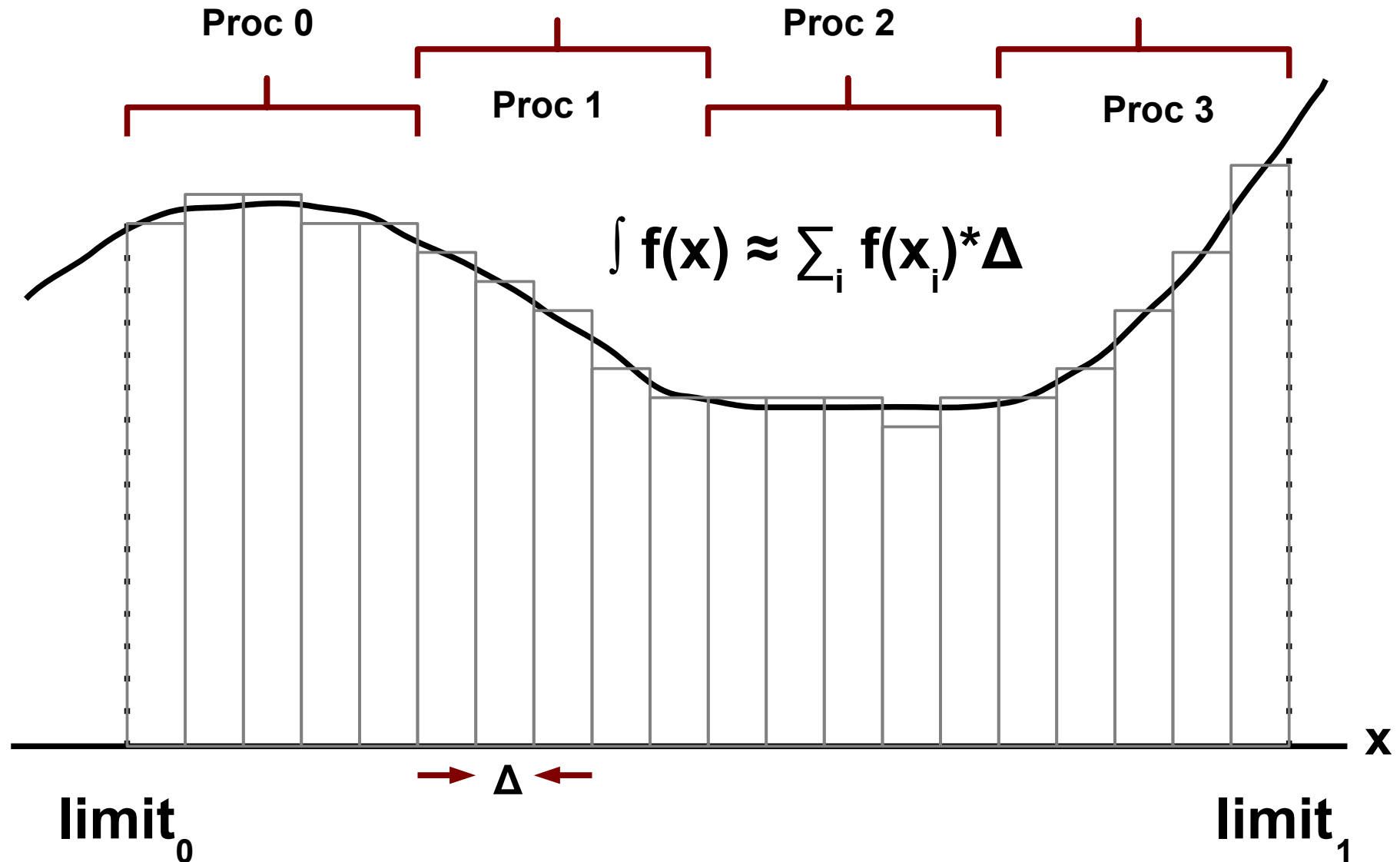
Collective communication

- Call same function from all processes
- Allows implementation to organize communications efficiently
 - ...and save the programmer some work
- Examples
 - MPI_Bcast (broadcast)
 - MPI_Reduce (global summation)
 - MPI_Scatter (distribute an array)
 - MPI_Gather (collect an array) ...

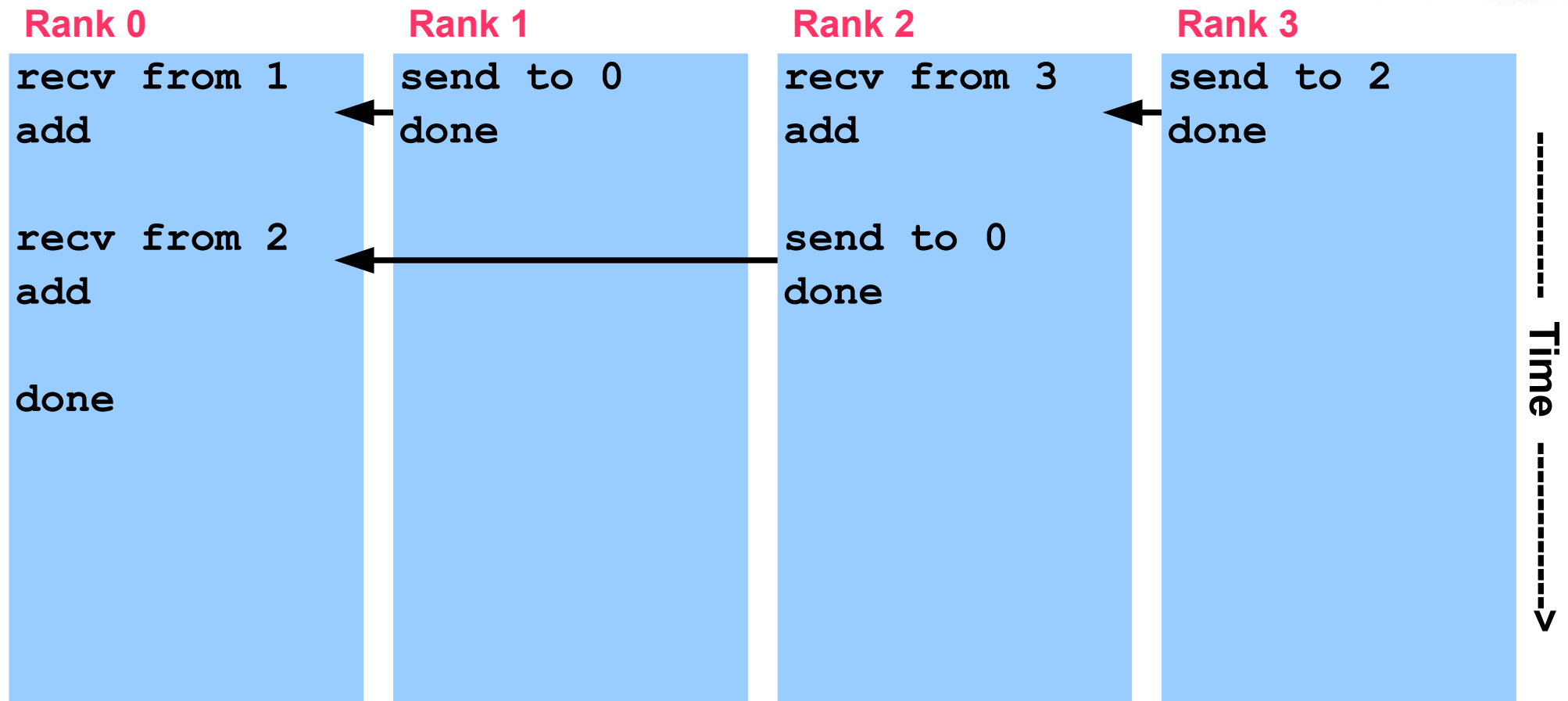
Numerical integration



Numerical integration, parallel



Overlapping communication



- Tree structure
- $O(\log n)$ time
- Let MPI do this for you!

Broadcast & Reduce

integrate.c

```
if (myRank == root) {
    ReadParams( limits ); }

MPI_Bcast( limits, 2, MPI_REAL, root,
           MPI_COMM_WORLD );

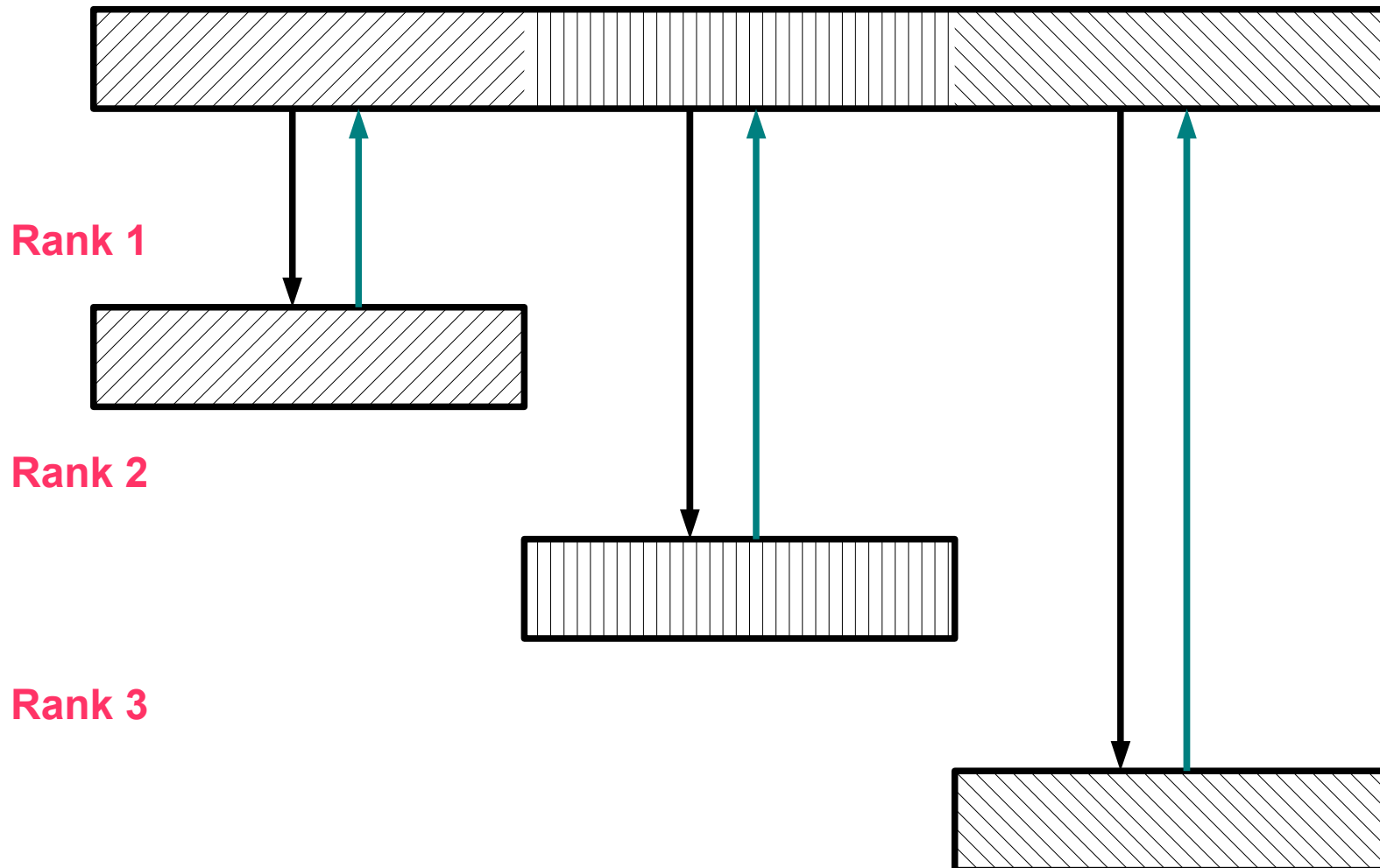
width = (limits[1]-limits[0])/nProcs;
myLimits[0] = limits[0] + width*myRank;
myLimits[1] = myLimits[0] + width;
mySum = Integrate( myLimits, nIntervals );

MPI_Reduce( &mySum, &globalSum, 1, MPI_REAL,
            MPI_SUM, root, MPI_COMM_WORLD );

if (myRank == root) {
    printf( "sum is: %f\n", globalSum ); }
```

Scatter & Gather

Rank 0



Communicators

- Only seen MPI_COMM_WORLD so far
- MPI_COMM_WORLD is the set of all processes in this MPI job
- Can define subsets called “communicators”
- Can do collective communications within subset

Parallel Input/Output

- Each process can open and close its own files in Open MPI
- This is “normal” input/output (I/O)
- Different MPI procs accessing the **same file** at the same time is “Parallel I/O”

Timing

```
double starttime, endtime;  
starttime = MPI_Wtime();  
// ... stuff to be timed ...  
endtime = MPI_Wtime();  
printf("That took %f seconds",  
      endtime-starttime);
```

- Standard time functions in C and Fortran 90 have shortcomings
- MPI_Wtime not guaranteed to be synchronized between processes

The real world (sort of)

- Numol, “Numerical Molecules”, quantum chem
 - A. D. Becke & R. M. Dickson, J. Chem. Phys. 92, 3610 (1990)
- ParNum, “Parallel Numol” (unpublished) uses eleven MPI functions:

```
MPI_Init, MPI_Finalize,  
MPI_Comm_Rank, MPI_Comm_Size,  
MPI_Send, MPI_Recv,  
MPI_Bcast, MPI_Reduce, MPI_Barrier,  
MPI_Wtime, MPI_Get_processor_name
```

- plus 7 constants and one communicator, MPI_COMM_WORLD

Tutorial Material on MPI - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.mcs.anl.gov/research/projects/mpi/tutorial/index.html

Most Visited W Wikipedia Bookmarks ACEnet Wiki

ACEnet Support Ticketing System :: Da... Tutorial Material on MPI

Tutorial material on MPI available on the Web

- [Advanced MPI: I/O and One-Sided Communication](#), presented at SC2005, by William Gropp, Rusty Lusk, Rob Ross, and Rajeev Thakur. A [shorter version](#) (presented at Euro PVMMPI'05) is also available. The [example programs](#) are available as a gziipped tar file.
- [Tutorial on MPI: The Message-Passing Interface](#) by William Gropp contains slides for a presentation and is also available as [Postscript for Tutorial on MPI: The Message-Passing Interface](#) and [Four-up Postscript for Tutorial on MPI: The Message-Passing Interface](#).

A [set of exercises](#) for this tutorial is available.

- [Tuning MPI Applications for Peak Performance](#)
- [A short introduction to MPI](#), an overview of MPI and MPI research in the MCS division at ANL.
- [A somewhat longer introduction to MPI](#), with some simple examples.
- [Laboratory for Scientific Computing's MPI Tutorials](#)
- [Introduction to MPI](#), from NAS at NASA Ames.
- [Norm Matloff's MPICH MPI Tutorial](#) and [LAM MPI Tutorial](#).
- [A draft of a Tutorial/User's Guide for MPI](#) by Peter Pacheco.
- [MPI](#), a May '97 talk by Marc Snir of IBM.
- [The LAM companion to "Using MPI..."](#) by Zdzislaw Meglicki (gustav@arp.anu.edu.au)
- [MPI: from Fundamentals to Applications](#) by David Walker

http://www-unix.mcs.anl.gov/mpi/tutorial/gropp/talk.html

http://www.cs.usfca.edu/~peter/ppmpi/

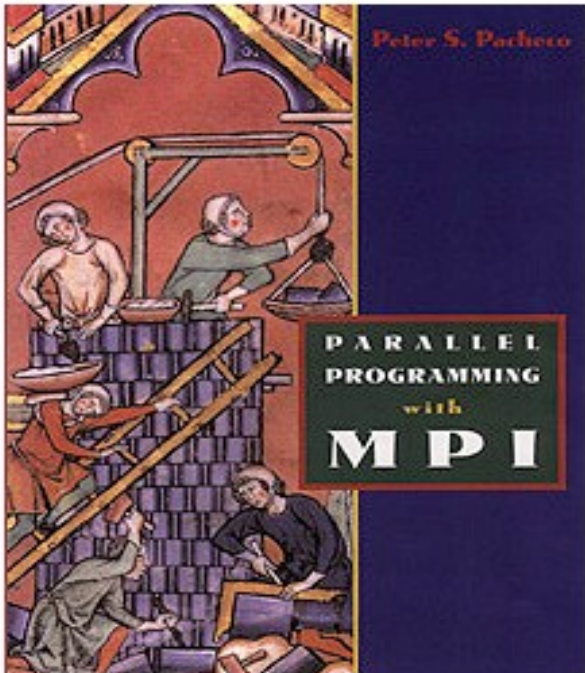
Parallel Programming with MPI - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.cs.usfca.edu/~peter/ppmpi/ MPI Pacheco

Most Visited W Wikipedia Bookmarks ACEnet Wiki

ACEnet Support Ticketing System :: Da... Parallel Programming with MPI



Parallel Programming with MPI

by
Peter Pacheco

Parallel Programming with MPI is an elementary introduction to programming parallel systems that use the MPI 1.1 library of extensions to C and Fortran. It is intended for use by students and professionals with some knowledge of programming conventional, single-processor systems, but who have little or no experience programming multiprocessor systems. It is an extensive revision and expansion of [A User's Guide to MPI](#).

Done

Open MPI: Open Source High Performance Computing - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.open-mpi.org/

Most Visited W Wikipedia Bookmarks ACEnet Wiki

Main Page - ACEnet Wiki Open MPI: Open Source High Perf...



Open MPI: Open Source High Performance Computing

Home | **Support** | FAQ | Search

A High Performance Message Passing Library

The Open MPI Project is an open source [MPI-2](#) implementation that is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to combine the expertise, technologies, and resources from all across the High Performance Computing community in order to build the best MPI library available. Open MPI offers advantages for system and software vendors, application developers and computer science researchers.

Features implemented or in short-term development for Open MPI include:

- Full MPI-2 standards conformance
- Thread safety and concurrency
- Dynamic process spawning
- Network and process fault tolerance
- Support network
- Many OS's supported (32 and 64 bit)
- Production quality software
- High performance on all platforms
- Portable and maintainable
- Tunable by installers and end-users
- Component-based design, documented APIs
- Active, responsive mailing list

hwloc v1.0.1 released
Minor bug-fix release
> [Read more](#)

Open MPI v1.4.2 released
Security / bug fix release
> [Read more](#)

Done

About
Publications
Open MPI Team
FAQ
Videos
Open MPI Software
Download
Documentation
Source Code Access
Bug Tracking
Regression Testing
Version Information
Sub-Projects
Hardware Locality
Resilient Cluster Mgr.
MPI Testing Tool
Portable Linux
Processor Affinity
Open MPI User Docs
Community

Example code

- On ACEnet clusters...
 - courtenay
 - fundy
 - glooscap
 - placentia
- ...in /home/rdickson/public/MPI_demo.tar
 - “tar xf /home/rdickson/public/MPI_demo.tar”
- <http://wiki.ace-net.ca> for local documentation