Parallel Computing with Examples (MPI)

Shelley Knuth, Research Computing, University of Colorado-Boulder

shelley.knuth@colorado.edu

Questions? #RC_Meetups

Link to survey on this topic: http://goo.gl/forms/8VidcwOhRT

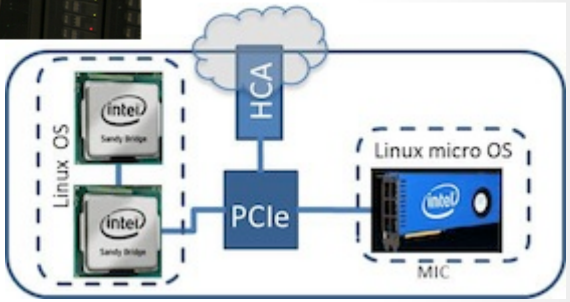
Slides: https://github.com/ResearchComputing/Final_Tutorials

Outline

- Distributed memory
- What is MPI?
- How is MPI used?
- Communicating
- Examples

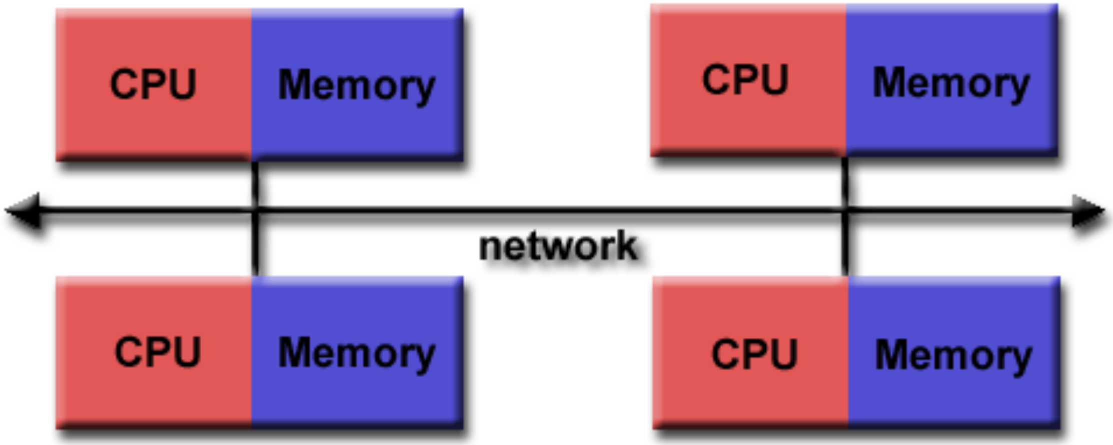
Programming to Use Parallelism

- Parallelism across processors/threads - OpenMP
- Parallelism across multiple nodes - MPI



www.scan.co.uk

Distributed-memory Model



Infiniband

Distributed memory requires a communication network to connect memory

Programmers explicitly define how processors access other processor's memory

Source: https://computing.llnl.gov/tutorials/parallel_comp/#ModelsShared

MPI

- MPI is a library specification for message passing
- Widely used standard
- Can run on shared, distributed, or hybrid memory models
- Exchange data between processes through communication between tasks – send and receive data
- MPI can get complicated
- Programmers must explicitly implement parallelism using MPI constructs
- Portable

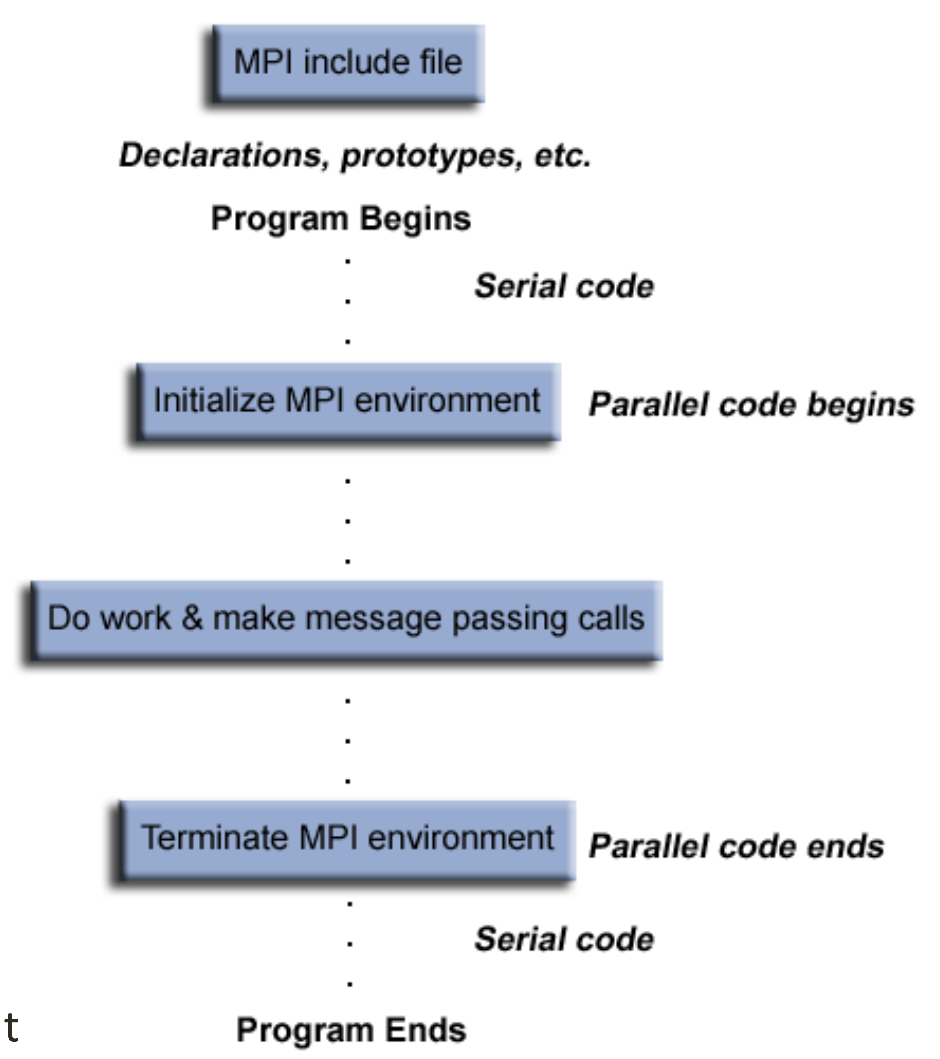
General MPI Code Structure

- You must have your header file at the top of any script you develop that uses MPI
- For C:

```
#include mpi.h
```

- For Fortran:

```
  include mpif.h
```

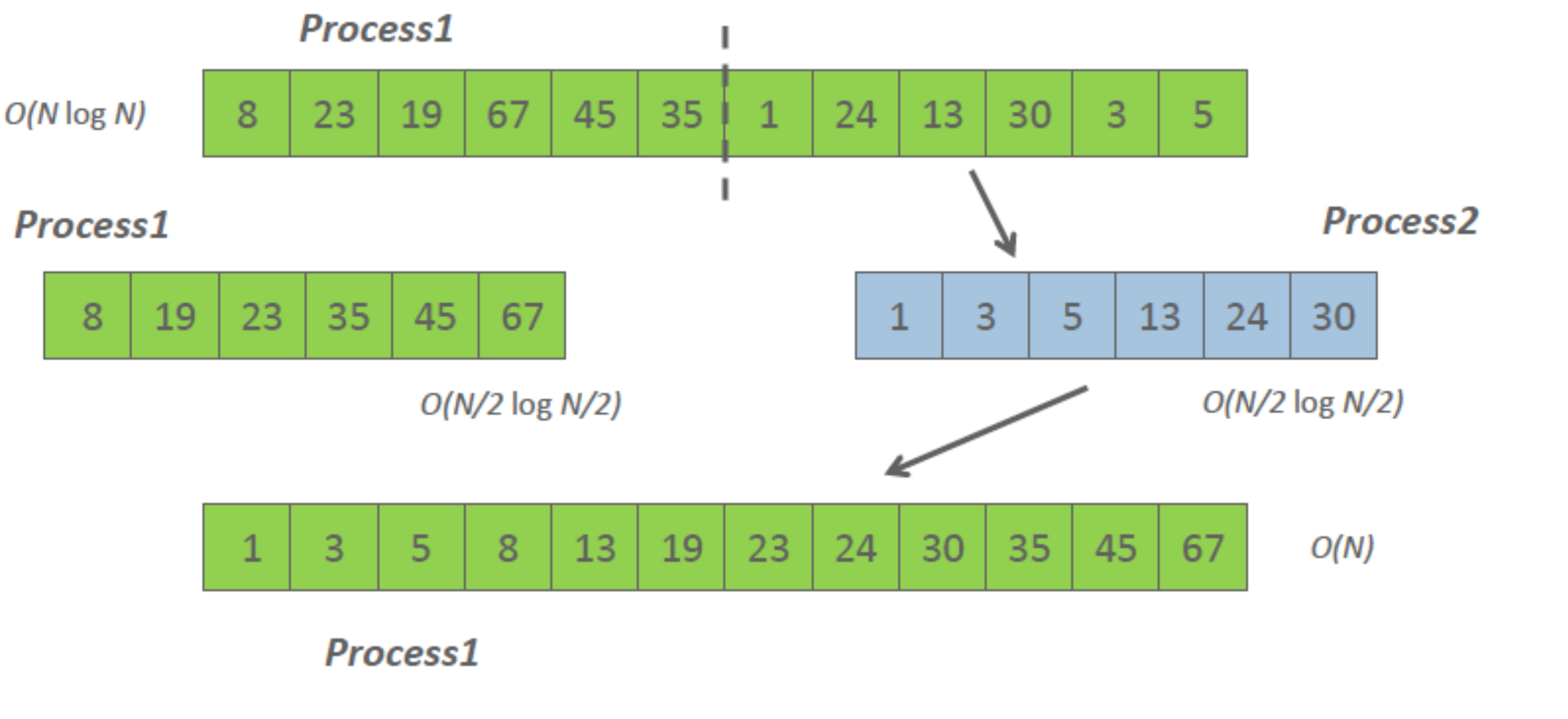

MPI include file
Declarations, prototypes, etc.
Program Begins
Serial code
Initialize MPI environment
Parallel code begins
Do work & make message passing calls
Terminate MPI environment
Parallel code ends
Serial code
Program Ends

https://computing.llnl.gov/tutorials/mpi/#What

Message Passing

- A program that runs on a node is called a **process**
- When a program is run a process is run on each processor in the cluster
- These processes communicate with each other using message passing
- Message passing allows us to copy data from the memory of one process into another
- Message passing systems must at a minimum support system calls for sending and receiving messages

Example – Sorting Integers



Process1

O(N log N)

8	23	19	67	45	35	1	24	13	30	3	5

Process1

8	19	23	35	45	67

O(N/2 log N/2)

Process2

1	3	5	13	24	30

O(N/2 log N/2)

1	3	5	8	13	19	23	24	30	35	45	67

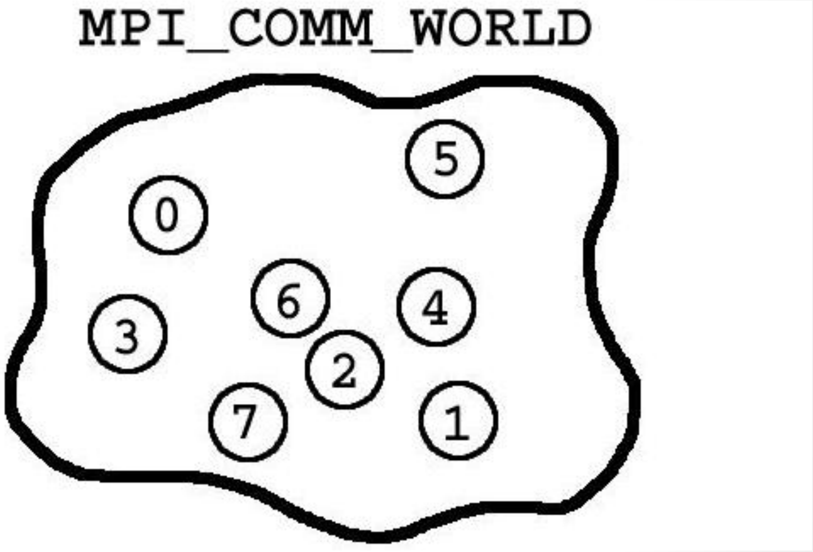
O(N)

Process1

http://htor.inf.ethz.ch/teaching/mpi_tutorials/ppopp13/2013-02-24-ppopp-mpi-basic.pdf

MPI Communicators

- Communicators used to group collections of processes allowed to communicate with each other
- Assigns integers to each process at initialization
  - Called “rank”
- Programmer uses rank to specify destination or source for sending/receiving
- Initially all processes grouped into MPI_COMM_WORLD



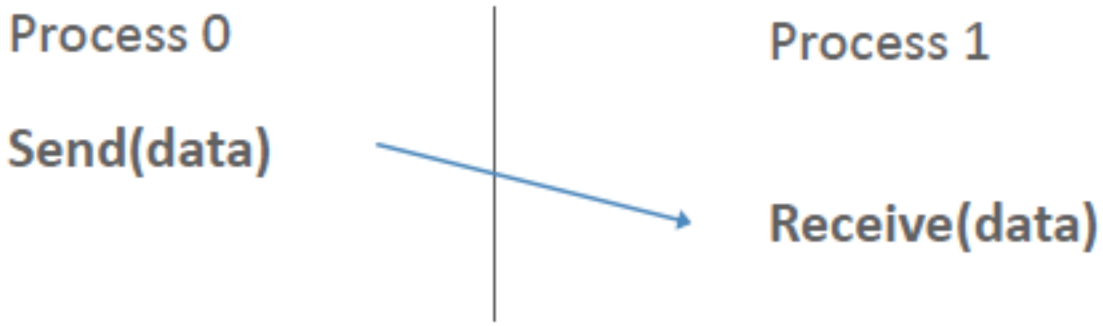
https://www.rc.usf.edu/tutorials/classes/tutorial/mpi/chapter2.html

Environment Management Routines

- These routines set the MPI execution environment, and cover many purposes
- Some common routines:
  - MPI_INIT
  - MPI_COMM_SIZE
  - MPI_COMM_RANK
  - MPI_FINALIZE

How Do I Write A Program in MPI?

- Application needs to specify:
  - How do you compile and run the MPI application?
  - How will the processes be identified?
  - How will the data be described?



http://htor.inf.ethz.ch/teaching/mpi_tutorials/ppopp13/2013-02-24-ppopp-mpi-basic.pdf

Compiling and Running an MPI Application

- MPI applications can be written in C, C++, or Fortran and appropriate calls to MPI can be added where required

- Compiling code:
  - Regular code:

```
gcc test.c –o test        ifort test.f -o test
```

  - MPI applications:

```
mpicc test.c –o test      mpifort test.f –o test
```

- Running code:
  - Regular code:

```
./test
```

  - MPI applications (running with 16 processes):

```
mpiexec –np 16 ./test
```

MPI Library on Janus

- Unlike OpenMP, with MPI you need to have the appropriate library loaded in your environment
- Research Computing recommends impi
- To load these, just type:

```
    ml gcc
```

then

```
    ml impi
```

At the command line

Compiling An Application

- Before compiling an application, you MUST:

- Include the MPI header file
  - Needed to use all the MPI Library calls
- Initialize the MPI environment
  - MPI_INIT()
- Specify an end to the MPI environment at end of program
  - MPI_Finalize()

Example Fortran Code

Fortran code: simple.f90

To run:

```
ml  slurm
ml  gcc
ml  impi
sinteractive  --reservation=meetup
mpif90  simple.f90  -o simple
mpiexec  -np  8  ./simple
```

OpenMP vs. MPI

Fortran code: hello.f90

The same code we ran as OpenMP modified for MPI

To run:

```
mpif90 hello.f90 -o hello
mpiexec -np 8 ./hello
```

Communication

- One process sends a copy of data to another process and that process receives it
- Requires the following information
  - Sender needs to know
    - Who to send the data to
    - What kind of data to send
    - A tag (like an email subject) so the receiver understands what's being sent
  - Receiver maybe needs to know
    - Who is sending the data
    - What kind of data is sending
    - The tag

MPI_SEND (Fortran)

- MPI_SEND(buf, count, datatype, dest, tag, comm, ierr)
- Basic sending operation
- Routine returns only after the application buffer in the sending task is free for reuse
  - In some sense, a send cannot complete without acknowledgment from the receiving process
  - Can be changed
  - Out of scope here

What does this mean?

- **Buffer**: Usually variable name that is to be sent/received
- **Count**: number of data elements of a particular type to be sent
- **Datatype:** pre-defined data type of data (MPI_CHARACTER, MPI_INTEGER, etc)
- **Dest:** destination – indicates the process where the message should be delivered. Sent as the rank of the receiving process

- MPI_SEND(buf, count, datatype, dest, tag, comm, ierr)

What does this mean?

- **Tag**: Arbitrary number assigned by the programmer to identify a message.
- **Comm:** communicator. Usually MPI_COMM_WORLD
- **Ierr:** error message

- MPI_SEND(buf, count, datatype, dest, tag, comm, ierr)

MPI_RECV (Fortran)

- MPI_REV(buf, count, datatype, source, tag, comm, status, ierr)

- **Status:** implies the source of the message
  - Integer array the size of MPI_STATUS_SIZE
- **Tag:** Can use MPI_ANY_TAG to receive any message regardless of tag

MPI Communication

Fortran code: ping.f90

To run:

```
mpif90 ping.f90 –o ping
mpiexec -np 8 ./ping
```

References

Material for this talk is used from

- https://computing.llnl.gov/tutorials/mpi/
- http://htor.inf.ethz.ch/teaching/mpi_tutorials/ppopp13/2013-02-24-ppopp-mpi-basic.pdf
- https://www.rc.usf.edu/tutorials/classes/tutorial/mpi/
- These are great tutorials – we encourage you to go there for more information!

Questions?

- Email rc-help@colorado.edu
- Twitter: @CUBoulderRC

- Link to survey on this topic: http://goo.gl/forms/8VidcwOhRT

- Slides: https://github.com/ResearchComputing/Final_Tutorials

- Questions? #RC_Meetup