Intro to MPI

Adam Jundt
User Services Consultant
HPC @ LSU, LONI
February 23, 2010
Outline

• About MPI
• Basic MPI Functions, Required System Calls, & Exercises
• Point to Point Communication & Exercises
• Collective Communication & Exercises
Motivation

Why use parallel computing?
- Faster Computation
- Solve larger problems
- Increase size of memory
- Shift to multi-core architectures
Machine Architecture

- How does a processor access the memory of another node?
### Available Systems

<table>
<thead>
<tr>
<th>System Name</th>
<th>System Type</th>
<th>Nodes</th>
<th>SMP Size</th>
<th>Total CPUs</th>
<th>Memory/Node</th>
<th>TFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queen Bee</td>
<td>Dell</td>
<td>680</td>
<td>8</td>
<td>5440</td>
<td>8 GB</td>
<td>50.7</td>
</tr>
<tr>
<td>Eric</td>
<td>Dell</td>
<td>128</td>
<td>4</td>
<td>512</td>
<td>4 GB</td>
<td>4.772</td>
</tr>
<tr>
<td>Oliver</td>
<td>Dell</td>
<td>128</td>
<td>4</td>
<td>512</td>
<td>4 GB</td>
<td>4.772</td>
</tr>
<tr>
<td>Louie</td>
<td>Dell</td>
<td>128</td>
<td>4</td>
<td>512</td>
<td>4 GB</td>
<td>4.772</td>
</tr>
<tr>
<td>Poseidon</td>
<td>Dell</td>
<td>128</td>
<td>4</td>
<td>512</td>
<td>4 GB</td>
<td>4.772</td>
</tr>
<tr>
<td>Bluedawg</td>
<td>IBM P575</td>
<td>13</td>
<td>8</td>
<td>104</td>
<td>16 GB</td>
<td>0.851</td>
</tr>
<tr>
<td>Ducky</td>
<td>IBM P575</td>
<td>13</td>
<td>8</td>
<td>104</td>
<td>16 GB</td>
<td>0.851</td>
</tr>
<tr>
<td>Zeke</td>
<td>IBM P575</td>
<td>13</td>
<td>8</td>
<td>104</td>
<td>16 GB</td>
<td>0.851</td>
</tr>
<tr>
<td>Neptune</td>
<td>IBM P575</td>
<td>13</td>
<td>8</td>
<td>104</td>
<td>16 GB</td>
<td>0.851</td>
</tr>
<tr>
<td>LaCumba</td>
<td>IBM P575</td>
<td>13</td>
<td>8</td>
<td>104</td>
<td>16 GB</td>
<td>0.851</td>
</tr>
</tbody>
</table>
Message Passing Interface

Advantages

• Scales to over 100k processors
• Distributed
• Not a new language
  – Library of functions
• Portable

Disadvantages

• Explicit parallel programming
Basic Functions

• Remember to

\[
\text{#include "mpi.h" for C & C++}
\]
\[
\text{include 'mpif.h' for Fortran}
\]

• MPI_Init(int *argc, char ***argv)
  – Initializes the execution environment, and passes in command line options
  – C Example: MPI_Init(&argc, &argv);
  – Fortran: MPI_Init(int ierror)

• MPI_Finalize()
  – Ends execution environment
  – Fortran: MPI_Finalize(int ierror)
Basic Functions Cont’d

• **MPI_Comm_rank(MPI_Comm comm, int *id)**
  - Gives the rank/id of the processor
  - Note on Communicators:
    ▪ Object used by MPI to determine which processors are included in a communication
    ▪ We’ll focus on MPI_COMM_WORLD (every processor)

C Example: `MPI_Comm_rank(MPI_COMM_WORLD, &myid)`
Fortran: `MPI_Comm_rank(MPI_COMM_WORLD, myid, ierror)`
Basic Functions Cont’d

- **MPI_Comm_size(MPI_Comm comm, int *size)**
  - Gives the number of processors in communicator
  - C Example: `MPI_Comm_size(MPI_COMM_WORLD, &nprocs)`
  - Fortran: `MPI_Comm_size(MPI_COMM_WORLD, nprocs, ierror)`
Structure of Fortran MPI Program

program hello
  include 'mpif.h'
  variables...
  MPI_Init(...)
  Program as normal...
  MPI_Finalize(...)
end program hello
Structure of C MPI Program

```c
#include ...

int main(...)
{
    variables...
    MPI_Init(...)
    Program as normal
    MPI_Finalize()
}
```
Execution

- **Only one executable**

  - P0
    ```python
    if(id == 0):
        print('hi')
    ```
  - P1
    ```python
    if(id != 0):
        print('bye')
    ```
  - P2
    ```python
    if(id != 0):
        print('bye')
    ```

Same executable, rely on ID’s to do different tasks
Compile and Run

• **Compile**
  - `mpicc program.c` (C programs)
  - `mpiCC program.C` (C++ programs)
  - `mpif90 program.f90` (Fortran codes)

• **Execute**
  - `qsub -I -l nodes=1:ppn=4 -l walltime=00:01:00`
  - `mpirun -np 4 ./a.out`
Run Cont’d

• PBS Script

```
#!/bin/bash
#PBS -A allocation
#PBS -q checkpt
#PBS -l nodes=1:ppn=4
#PBS -l walltime=00:01:00

cd $PBS_WORKDIR
mpirun -np 4 ./a.out
```
Run Cont’d

To find your allocation name, execute:
[user@machine]$: balance

To submit the PBS Script, execute:
[user@machine]$: qsub scriptname

To check status of job, execute:
[user@machine]$: qstat –u username

Adam Jundt, User-Services Consultant, HPC @ LSU
Exercises

• Write a program that prints out "Hello world from proc X out of Y"

• Write a program that has all even procs print "Even X" and odd procs print "Odd Y"

• Example solutions online at:
  - http://docs.loni.org/wiki/c_mpi_examples
  - http://docs.loni.org/wiki/fortran_mpi_examples
Point to Point Communication

- **MPI_Send**(void *buf,
  int count,
  MPI_Datatype datatype,
  int dest,
  int tag,
  MPI_Comm comm,
  int ierror) ← fortran only

Example use:

```
int a[3] = {1, 2, 3}
MPI_Send(a, 3, MPI_INT, 0, 0, MPI_COMM_WORLD)
```
MPI_Datatype

• Predefined type of data, choices (make changes for Fortran and C table):

<table>
<thead>
<tr>
<th>Fortran</th>
<th>C/C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI_INTEGER</td>
<td>MPI_INT</td>
</tr>
<tr>
<td>MPI_DOUBLE_PRECISION</td>
<td>MPI_DOUBLE</td>
</tr>
<tr>
<td>MPI_REAL</td>
<td>MPI_FLOAT</td>
</tr>
<tr>
<td>MPI_CHARACTER</td>
<td>MPI_CHAR</td>
</tr>
</tbody>
</table>
Point to Point Cont’d

- MPI_Recv(void *buf,
  int count,
  MPI_Datatype datatype,
  int source,
  int tag,
  MPI_Comm comm,
  MPI_Status *status,  (Use MPI_STATUS_IGNORE)
  int ierror) ← fortran only

Example:

```c
int *recv = (int *) malloc(3*sizeof(int))
MPI_Recv(recv, 3, MPI_INT, 0, 0, MPI_COMM_WORLD,
  MPI_STATUS_IGNORE)
```
Exercises

• Implement the previous examples and execute on two processors
Collective Communication

MPI_Bcast(void *buf, int count, MPI_Datatype datatype, int source, MPI_Comm comm, [ierror])

P0 A
P1
P2
P3

Broadcast

P0 A
P1 A
P2 A
P3 A

Adam Jundt, User-Services Consultant, HPC @ LSU
MPI_Allreduce

- Combines values from all processes and distribute the result back to all processes
- `int MPI_Allreduce ( void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm )`
  - `MPI_MAX` → Maximum
  - `MPI_MIN` → Minimum
  - `MPI_PROD` → Product
  - `MPI_SUM` → Sum

Adam Jundt, User-Services Consultant, HPC @ LSU
MPI_Scatter

- Sends data from one task to all other tasks

MPI_Scatter( void *sendbuf
    int sendcnt,
    MPI_Datatype sendtype,
    void *recvbuf,
    int recvcnt,
    MPI_Datatype recvtype,
    int root,
    MPI_Comm comm)

[http://www.mpi-forum.org/docs/mpi21-report/node96.htm]
MPI_Alltoall

- MPI_Alltoall( void *sendbuf,
  int sendcount,
  MPI_Datatype sendtype,
  void *recvbuf, int recvcnt,
  MPI_Datatype recvtype,
  MPI_Comm comm )

[http://www.ncsa.illinois.edu/UserInfo/Resources/Hardware/CommonDoc/MessPass/]
MPI_Alltoallv

- int MPI_Alltoallv ( void *sendbuf,
  int *sendcnts,
  int *send_displs,
  MPI_Datatype sendtype,
  void *recvbuf,
  int *recvcnts,
  int *recv_displs,
  MPI_Datatype recvtype,
  MPI_Comm comm )
Additional Functions

• Many other functions, such as:
  – Gather
  – Isend
  – Wait
  – …

• List of all functions and documentation:
Exercises

• Have proc 0 define array 'a' with the values [0, 1, 2], and broadcast to all other processors.

• Have each processor define variable 'a' to a random number, perform a MPI_Allreduce with operation MPI_SUM, and MPI_MAX

• Have proc 0 define array 'a' with the values [0, 1, 2, 3], and use MPI_Scatter to split the array.
Additional Examples

The following exercises are take home programs. Algorithms, templates, and solutions can be found online @

https://docs.loni.org/wiki/advanced_c_mpi_examples
https://docs.loni.org/wiki/advanced_fortran_mpi_examples
Summation Example

• Task: add the numbers from 1 to N together.
  • for(i=1;i<=N;i++) total += i;

• How to do this in parallel?

• How to do this with MPI?
Exercises

• Write a MPI program that sums the numbers from 1 to \( N \) (where \( N \) is provided as a program argument), and has processor 0 print the result.

• Assume \( N \) can be evenly divided by the number of processors.

• Example template and solution online at:
  • https://docs.loni.org/wiki/advanced_c_mpi_examples
  • https://docs.loni.org/wiki/advanced_fortran_mpi_examples
Pi Example

• How can we approximate pi with a monte carlo simulation?

• Area = \pi r^2
Pi Example Cont’d

• How to do this serially?

• Use a unit circle (r = 1).
• Generate random X and Y values [between 0 and 1]
• If ((x*x + y*y) <=1 ) pt is within the sphere
• Area = 4*count_in / number_tests = pi

Adam Jundt, User-Services Consultant, HPC @ LSU
Exercises

• Write a MPI program approximates pi and has processor 0 print this approximation.

• Example template and solution online at:
  • https://docs.loni.org/wiki/advanced_c_mpi_examples
Matvec Example

Matrix times a vector

\[
\begin{align*}
A & \quad \begin{pmatrix} x \end{pmatrix} \\
\begin{pmatrix} x \\ \vdots \\ x \end{pmatrix} & = \\
\begin{pmatrix} y \\ \vdots \\ y \end{pmatrix} \\
\end{align*}
\]

Serial Code:
for \( i = 1:n \)
for \( j = 1:n \)
\[ y[i] += A[i][j]*x[j] \]
Matvec Example

• How can we parallelize this example?

• How can we do this with MPI?
Matvec Example Cont’d

Row-wise 1D partitioning

What is the maximum number of processors we could use in this situation?

Adam Jundt, User-Services Consultant, HPC @ LSU
Exercises

• Write a MPI program solves the problem:

\[
\begin{bmatrix}
1 & 2 & 3 & 4 \\
2 & 2 & 3 & 4 \\
3 & 2 & 3 & 4 \\
4 & 2 & 3 & 4
\end{bmatrix}
\begin{bmatrix}
1 & 2 & 3 & 4
\end{bmatrix}
\cdot
\begin{bmatrix}
2 & 2 & 3 & 4
\end{bmatrix}
= ?
\]

• Example template and solution online at:
  * https://docs.loni.org/wiki/advanced_c_mpi_examples
Another Solution

(a) Initial data distribution and communication steps to align the vector along the diagonal

<table>
<thead>
<tr>
<th>Matrix $A$</th>
<th>Vector $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>$P_1$</td>
</tr>
<tr>
<td>$P_{\sqrt{p}}$</td>
<td>$\leq$</td>
</tr>
<tr>
<td>$P_{2\sqrt{p}}$</td>
<td>$\wedge$</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\wedge$</td>
</tr>
<tr>
<td>$P_{p-1}$</td>
<td>$\leq$</td>
</tr>
</tbody>
</table>

(d) Final distribution of the result vector

<table>
<thead>
<tr>
<th>Matrix $A$</th>
<th>Vector $y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td>$P_1$</td>
</tr>
<tr>
<td>$P_{\sqrt{p}}$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$P_{2\sqrt{p}}$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$P_{p-1}$</td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>

Adam Jundt, User-Services Consultant, HPC @ LSU
Questions