Tools for Computational Physics
Week 4, Lecture 2: MPI
Hardware and Implementations

R. Rousseau
SISSA, 2-4 Via Beirut,
Trieste, Italy
Outline

• Introduction
• Basic Structure of MPI Library and implementations
• Bandwidth and Latency
• Hardware for Parallel Computing.
• How to set up MPI on a cluster.
• Examples of performances of various codes and networks.
Introduction

- The previous lecture set up the basics of the MPI protocol for parallel programming. The current lecture is designed to show you how to implement it on a Linux machine and what factors one needs to consider.

- Fundamental to this is an overview of the most commonly Used network technology for efficient MPI implementations.

- An overview of how the MPI commands interact with the network and an introduction of the considerations of Latency and Bandwidth will be presented and various networks characterized,

- A brief overview on how to implement MPI on a cluster will be presented and illustrative performance tests will be shown.
**MPI: from you code to the machine**

- **Your code**
  - User Application: calls MPI routines

- **MPI Library**
  - MPI Commands: MPI_INIT, MPI_SEND, MPI_RECV, MPI_BCAST etc etc
  - MPI Socket Interface: converts data to send to device driver

- **Operating System**
  - Socket Library: converts data to protocol used by device driver

- **Hardware**
  - Device driver
  - Network Hardware

---

**Source of Latency**
**MPI performance criteria**

**Bandwidth**: Speed at which data can be sent across a network

**Latency**: how much time is spent in processing the signal causing the CPU to wait

<table>
<thead>
<tr>
<th>Technology</th>
<th>Latency</th>
<th>Max. bandwidth</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI (Dolphin)</td>
<td>2.26 µs</td>
<td>2 016 Mbps</td>
<td><a href="http://www.dolphinics.com">www.dolphinics.com</a></td>
</tr>
<tr>
<td>Myrinet</td>
<td>12.00 µs</td>
<td>1 818 Mbps</td>
<td><a href="http://www.myrinet.com">www.myrinet.com</a></td>
</tr>
<tr>
<td>Gigabit Ethernet</td>
<td>23.00 µs</td>
<td>936 Mbps</td>
<td><a href="http://www.dolphinics.com">www.dolphinics.com</a></td>
</tr>
<tr>
<td>Infiniband</td>
<td>28.00 µs</td>
<td>3 768 Mbps</td>
<td>IEEE Symposium IPASS 2004</td>
</tr>
</tbody>
</table>

Source Dolphinics.com (note dolphin numbers depend on implementation of Their newest socket library). Latest Myrinet quotes latency ~2.5-4µs, bandwidth 220-330 Mbps.
Commonly Used Hardware: Gigabit Ethernet

- Gigabit Ethernet.
- Uses TCP/IP protocol.
- TCP/IP protocol adds extra overhead in passing messages latency.
- Large enough bandwidth for most applications.
- Many implementations available.
- Requires a switch.
- Cheap easy to configure-driver installed automatically during Linux set up.

Ideal for applications where only small amounts of data are to be transmitted or few machines will be used in parallel.
Myrinet

- Requires a card for each machine and a switch
- Has low latency and large bandwidth
- The Myrinet driver is the gm software which has its own protocol but can also emulate TCP/IP (for parallel file systems).
- Easy to configure.
- Both MPICH and LAM-MPI are available.

- CAN be expensive.

Network of this type are REQUIRED if you plan to run with lots of CPUs or pass large amounts of Data.
**Dolphin: SCI**

Only requires 1 card per machine (cheaper than myrinet).
No Switch—but cabling is time consuming
And tricky.
Low Latency big bandwidth.
Technology has not changes in ca 4 years!

Downside is the software,
ScaMPI and driver by Scali (proprietary).
No real TCP/IP support
For network file systems.
A free dolphin driver is available which can be “made” to work with Lam-MPI unfortunately lots of bugs.
Infiniband

- 1 Card per machine
- Requires a switch.
- Huge band width (good for file system)
- High latency (not so good for MPI)
- Installation is “easy”.
- Available with MPICH and Lam Implementations of MPI

This technology is still fairly new and there is not that many Implementations at present.
Air Conditioned Computer-Room

2 opteron 2GHz, 150 GB-IDE
4GB ECC-DDRAM,

Monitor/keyboard IBM

1.8GHz opteron,
2 190GB SATA disks

1000 MBps Ethernet (rsh commands)
1000 MBps Ethernet (NFS only)
100 MBps switch : management

Myrinet
Myrinet switch

Head Node

SISSA
## Common MPI Implementations

**MPICH**  Designed for high throughput, low latency and portability.  
Is available for most common hardware types  
More complicated to set up and configure.  

**LamMPI**  Designed for low latency on cheap cluster hardware  
Can work on almost all common hardware types.  
Easy to use and configure.

<table>
<thead>
<tr>
<th>MPI Library</th>
<th>MPI/Pro-1.5b7-2tv, MPI/Pro-1.6.1-1tv</th>
<th>MPICH-1.2.0 + MVICH-1.0a6.1</th>
<th>LAM-6.3.2-usysv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measurement</td>
<td>Latency</td>
<td>Bandwidth</td>
<td>Latency</td>
</tr>
<tr>
<td><strong>Units</strong></td>
<td>microseconds</td>
<td>Mbyte/s</td>
<td>microseconds</td>
</tr>
<tr>
<td>phase3-VIA</td>
<td>49 interrupt 17 polling</td>
<td>102.4 interrupt 102.8 polling</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>88 interrupt 25 polling</td>
<td>88.4 interrupt 86.4 polling</td>
<td>15</td>
</tr>
<tr>
<td>phase3-local</td>
<td>76 interrupt 17 polling</td>
<td>67.8 interrupt 81.0 polling</td>
<td>16</td>
</tr>
<tr>
<td>phase3-TCP</td>
<td>112 interrupt</td>
<td>11.1 interrupt</td>
<td>102</td>
</tr>
<tr>
<td>phase2-TCP</td>
<td>198 interrupt</td>
<td>11.1 interrupt</td>
<td>175</td>
</tr>
</tbody>
</table>
Installing and setting up MIP on a cluster

1. Install the SAME MPI implementation on all the machines you wish to run on.

2. Set up a consistent file system, and user id on all these machines.

3. Enable automatic login on all machines and make sure all MPI environment variables are set the same.

4. Compile your code and link with the installed MPI libraries.
Installing and setting up MPI on a cluster: Example LAM MPI

First decide how you wish to configure the MPI library and install it on all Machines. (note rpm install is usually only good for Ethernet. You have to compile by hand if you want something else).

```bash
node01:/usr/src/lam-7.0.6 # more give_me_lam.sh
#!/bin/sh
#
cd /usr/src/lam-7.0.6
CC=gcc
CXX=g++
FC=g77
FFLAGS=--fno-second-underscore
#CONFEXTRA="--without-threads --disable-tv"

export CC CXX FC FFLAGS

# Lam without gm
# ./configure --prefix=/opt/lam --with-rpi=usysv --without-threads --disable-tv-queue 
./configure --prefix=/opt/lam-gm --with-rpi=gm --without-threads --disable-rpi-gm-ptmalloc --disable-tv-queue --with-gm=/opt/gm
#
make
make install
exit
```
Installing and setting up MPI on a cluster: Example LAM MPI

Installation will place in a designated directory:
- The mpi boot, run halt etc commands.
- Compiler wrappers (that tell the compiler what libraries to get and where they are).
- And the libraries (static and dynamic)
Installing and setting up MPI on a cluster: Example LAM MPI

Other steps needed to make things easier for users
• Set up the /etc/profile.d/lam.sh
• MPI wrappers (for various compilers)
• .ssh authority keys for auto login
```
DEST = .
BIN = .

#QMMD_FLAGS = -D__QMECHCOUPL
#QMMD_LIBS = -L. -1mm
FFLAGS = -pc64 -tpp7 -02 -unroll -tune pn4 -arch pn4
LFLAGS = -L. -latlas_x86_64 -lacmlmv $(QMMD_LIBS) -Xlinker \ 
    -Bdynamic -lc -lm -Xlinker -Bstatic
#LFLAGS = -L. -lacml -lg2c -lacmlmv $(QMMD_LIBS) -Xlinker \ 
#    -Bdynamic -lc -lm -Xlinker -Bstatic
#FFLAGS = -pc64 -tpp7 -03 -unroll
#LFLAGS = -Bstatic -L. -latlas_x86_64 $(QMMD_LIBS)
CFLAGS = -02 -Wall -m64
CPP = /lib/cpp -P -C -traditional
CPPFLAGS = -D__Linux -D__PGI -DLAPACK -DFFT_DEFAULT -DPINTER8 \ 
    -D_GROMOS -DPARALLEL -DMP_LIBRARY=__MPI -DMYRINET
CPPFLAGS_GROMOS = -DEWALD -DEWATCUT -DHAIT_SHAPE -UNPACKED_GRID
NOOPT_FLAG =

#CC = env LAMHCC=gcc mpicc
#FC = env LAMHF77=ifort mpif77 -c
#LD = env LAMHF77=ifort mpif77 -static-libcxa
CC = mpicc -02 -m64
FC = mpi_ifort -c
```
Running an MPI code

- Initialize MPI environment on all machines
  Ex: >lamboot *hostfile*

- Run the MPI program on np CPUs:
  Ex: > mpirun -np 4 code.x *input.file > output.file*

- Terminate MPI environment on all machines
  Ex: >lamhalt *hostfile*

On many machines the three steps are all converted into a single Command called mpiexec. Normally executed in a batch queue
The mpiexec command will automatically boot the MPI environments on all nodes that are included in the nodefile produced by the queue.
Benchmarking MPI calculations

Typical comparison of performance of an MPI code on a given Platform is to plot time vs $N_{\text{CPU}}$.
A measure of the network performance for a job: scalability

A second useful quantity to measure network performance is to plot

\[ \text{Scalability} = \frac{t(N_{\text{CPU}})}{t(1_{\text{CPU}})} \text{ vs } N_{\text{CPU}} \]
**How a code scales depend on its implementation.**

PWSCF on the same myrinet network as CPMD in the previous slide-dying at ca 14 CPU
How a code scales depend on the latency and bandwidth of the network.
Conclusion

We have seen there is a lot of possible technologies, and implementations for using MPI in computational physics.

If you can keep the size of data you transmit small then a cheap network (gigabit) is the way to go. If you do not need a lot of cpu the same is true.

For jobs requiring a large number of CPUs Low latency is a must. Bandwidth is Nice but it is secondary Gigabit bandwidth is already pretty good even for plane wave codes.

Having MPI on a separate network as the File system is wise. File systems need a lot of bandwidth and are less forgiving with competing network traffic.

When in doubt grab the lion by the tail
And shake things up a bit!