Workshop on
High Performance Computing (HPC08)
School of Physics, IPM
February 16-21, 2008

Tutorial on MPI: part I

Stefano Cozzini
CNR/INFM Democritos and
SISSA/eLab

Agenda   first part

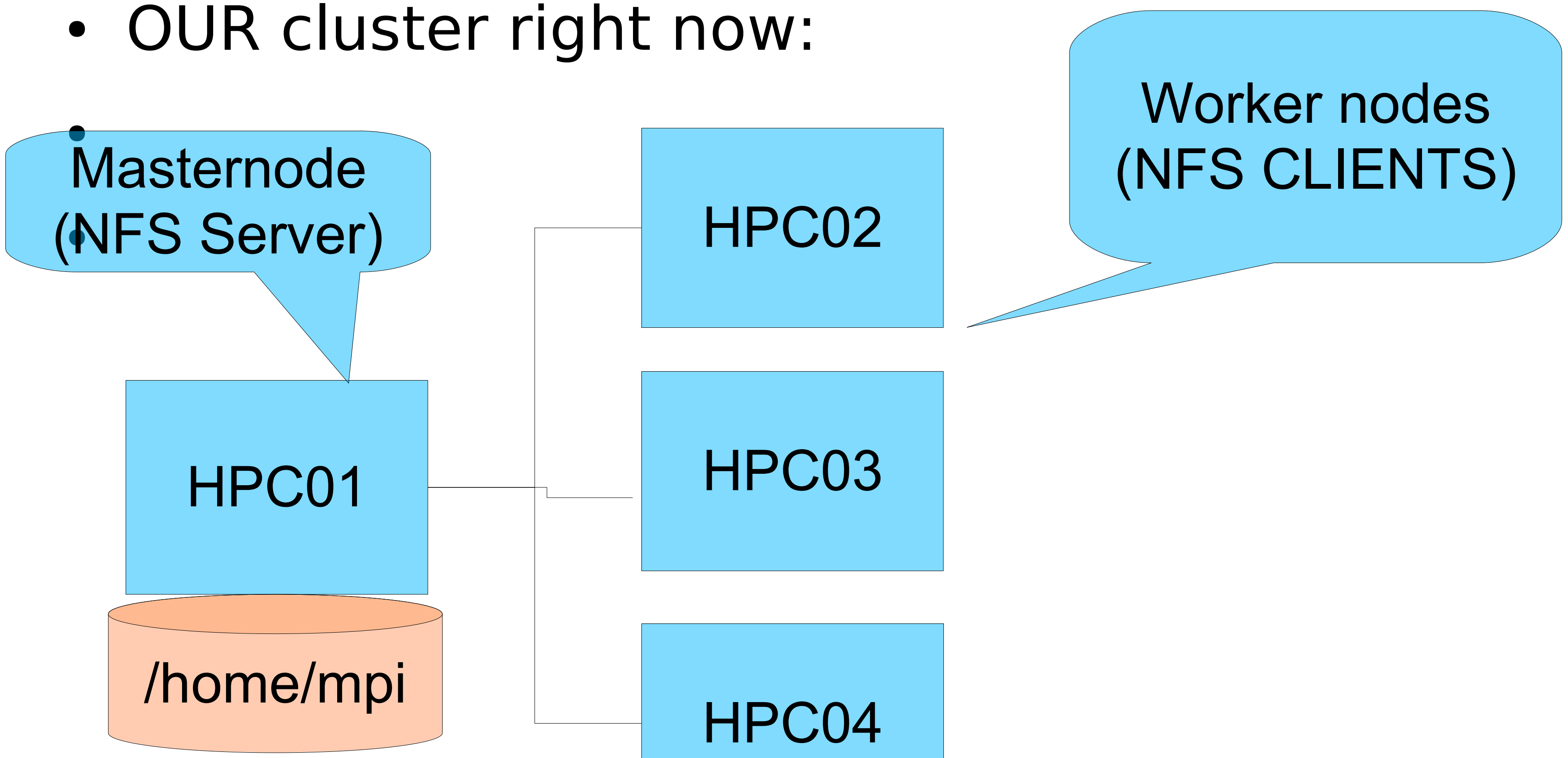
- WRAP UP of the yesterday's lab:
  - What we did
  - Remarks and problems..
  - Problems we run into..
- Introduction to models for parallel programming
- Message passing paradigm
- What is MPI ?
- Implementations of MPI
- Writing/Compiling/Running MPI programs..

WRAP-UP: what we did..

- Configured a basic Beowulf cluster:
  - Identify a masternode + some Worker Nodes
  - The Masternode is offering a central service NFS
  - We created on all the machine an account for MPI programming that is sharing the same home directory on all the node (thanks to NFS)
  - We setup a passwordless mechanism to login on client nodes from masternode and viceversa
  - We configure LAM-MPI to use all the CPUS of our cluster
  - We booted the MPI cluster..

Schematic view of our cluster

- OUR cluster right now:



remarks/problems on NFS procedures

- PROBLEM

  Firewall does not allow to mount the nfs filesytem on Worker node:

- Solution:
- Get rid of iptables on masternode !
  - Service iptables stop
  - Chkconfig –level 345 iptables off
- REMARK:
  - Consolidate the procedure done by hand !
    - Edit /etc/fstab on client
    - Make nfs service available on boot

remarks/problems on creating the MPI account

- Remarks:
  - Done by each member of the cluster
  - By default useradd create /home/mpi (fine for us..)
- Problem:
  - On some clusters mpi account had a different UID/GID
- Solution:
  - Fix the GID/UID (edit /etc/passwd and /etc/groups)

Passwordless mechanisms.

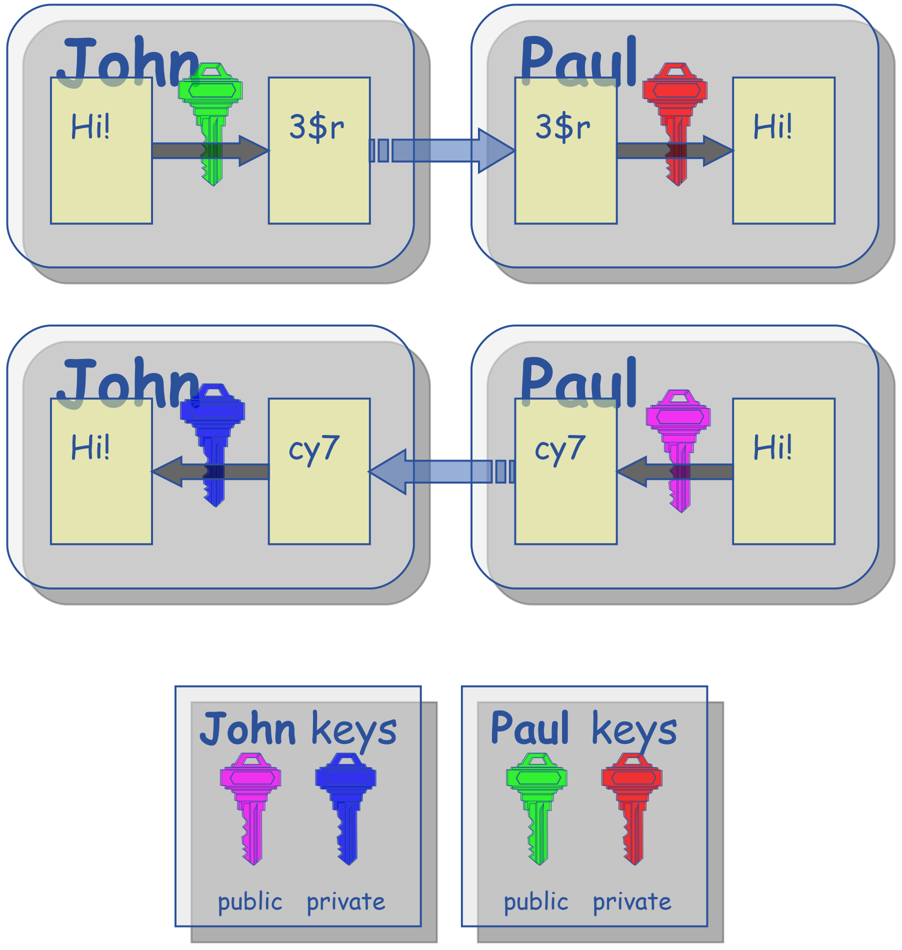
- In order to run jobs on the cluster, we need to set up passwordless login for internal cluster connections only.
- Security note:
  - DO NOT share the keys that you produce in this step with other hosts, and do not copy your keys from other hosts to this cluster.
- The two commands:
  - ssh-keygen -t rsa
  - cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys

This is Black magic isn't it ?

- NOTE: Passwordless keys require user only permissions on the .ssh directory. To ensure this is the case, use the following command
  - chmod -R 700 ~/.ssh

Public key mechanism: asymmetric encription

- Every user has two keys: one *private (secret)* and one *public*:
  - it is *impossible* to derive the private key from the public one;
  - a message encrypted by one key can be decrypted **only** by the other one.
- No exchange of private key is possible.
  - the sender cyphers using the *public* key of the receiver;
  - the receiver decrypts using his own *private* key;
  - the number of keys is O(n).
- Examples:
  - **RSA** (1978)



Ssh and private/public keys

- Ssh always try public key authentication:
  - The public key method allows the RSA or DSA algorithm to be used:
  - The client uses his private key, ~/.ssh/id_dsa or ~/.ssh/id_rsa, to sign the session identifier and sends the result to the server.
  - The server checks whether the matching public key is listed in ~/.ssh/authorized_keys and grants access if both the key is found and the signature is correct.

Ssh with/out shared home

- Without shared home directories:
  - Each member of the cluster have to generate its pair and copy the public keys on all the N-1 machines
  - 4 public keys to be copied in 4-1 node
  - N squared problem ! Does not scale at all: a nightmare every time a new node is inserted..
- With shared home directory:
  - 1 single step: generate the keys and copy the public one in the authorized file..
  - you will always use the same pair for all the N*N-1 different login you can do on the N node cluster !

LAM-MPI issues and configuration: later..

- Only one remark: how to identify CPUs on your machine:
  -
  - Cat /proc/cpuinfo | grep processor | wc -l

Modern Parallel Architectures

Two basic architectural scheme:

Distributed Memory

Shared Memory

Now most computers have a mixed architecture

Parallel Programming Paradigms

The two architectures determine two basic scheme for parallel programming

Data Parallel (shared memory)

Single memory view, all processes (usually threads) could **directly** access the whole memory

Message Passing (distributed memory)

all processes could **directly** access only their local memory

Parallel Programming Paradigms, cont.

Its easy to adopt a Message Passing scheme in a Shared Memory computers (*unix process have their private memory*).

Its less easy to follow a Data Parallel scheme in a Distributed Memory computer (*emulation of shared memory*)

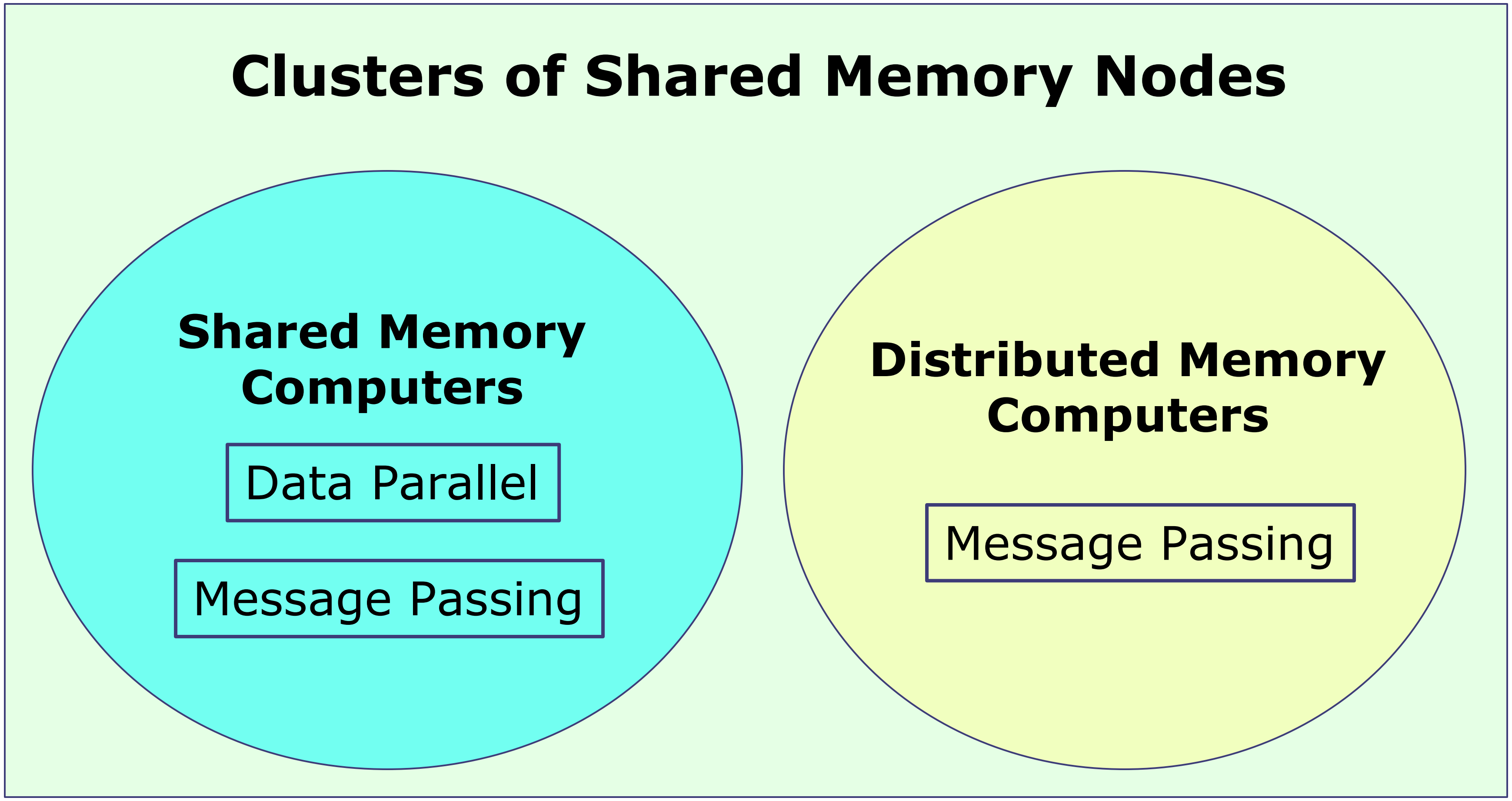
Its relatively easy to design a program using the message passing scheme and implementing the code in a Data Parallel programming environments (*using OpenMP or HPF*)

Its not easy to design a program using the Data Parallel scheme and implementing the code in a Message Passing environment.

Parallel Programming Paradigms, cont.

Programming Environments	
Message Passing	**Data Parallel**
Standard compilers	Ad hoc compilers
Communication Libraries	Source code Directive
Ad hoc commands to run the program	Standard Unix shell to run the program
Standards: **MPI**	Standards: **OpenMP**

Architectures vs. Paradigms



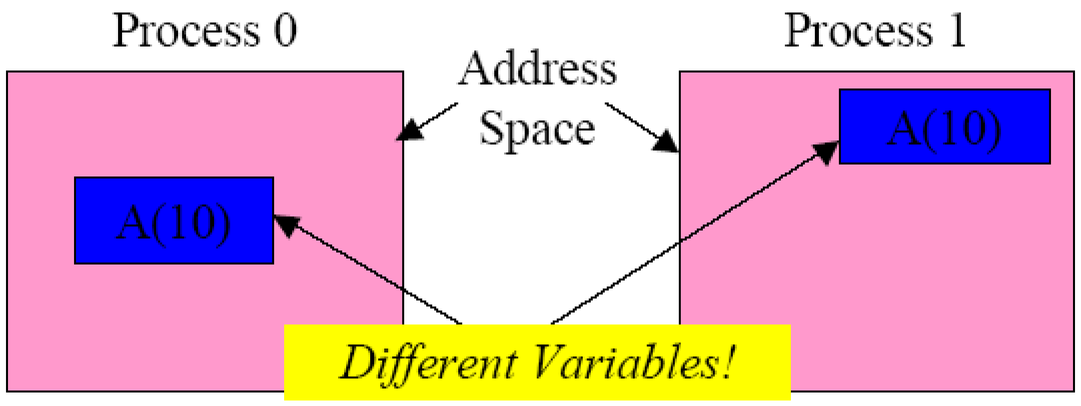
Parallel programming: a short summary..

Architectures	
Distributed Memory	Shared Memory
Programming Paradigms/Environment	
Message Passing	Data Parallel
Parallel Programming Models	
Domain Decomposition	Functional Decomposition

Message passing paradigm

- Parallel programs consist of separate processes, each with its own address space
  - Programmer manages memory by placing data in a particular process
- Data sent explicitly between processes
  - Programmer manages memory motion
- Collective operations
  - On arbitrary set of processes
- Data distribution
  - Also managed by programmer

Distributed memory (shared nothing approach)



What is MPI?

- A message-passing library specification
  - extended message-passing model
  - not a language or compiler specification
  - not a specific implementation or product
- For parallel computers, clusters, and heterogeneous networks
- Full-featured
- Designed to provide access to advanced parallel hardware for end users, library writers, and tool developers
- Currently MPI-1 (1.2) and MPI-2 (2.0)

What is MPI?

A STANDARD...

- The actual implementation of the standard is demanded to the software developers of the different systems
- In all systems MPI has been implemented as a library of subroutines over the network drivers and primitives
- many different implementations
  - LAM/MPI (today's TOY) www.lam-mpi.org
  - MPICH /MPICH2
  - OpenMPI (MPI-2)

Goals of the MPI standard

MPI’s prime goals are:

- To provide source-code portability
- To allow efficient implementations

MPI also offers:

- A great deal of functionality
- Support for heterogeneous parallel architectures

When do you need MPI ?

- You need a portable parallel program
- You are writing a parallel library
- You have irregular or dynamic data relationships that do not fit a data parallel
- model
- You care about performance

Where MPI is not needed

- You can parallel Fortran 90 or any other data parallelism mechanism
- You don’t need parallelism at all
- You can use libraries (which may be written in MPI)
- You need simple threading in a slightly concurrent environment

MPI references

- The Standard itself:
  - at http://www.mpi-forum.org
  - All MPI official releases, in both postscript and HTML
- Other information on Web:
  - at http://www.mcs.anl.gov/mpi
  - pointers to lots of stuff, including talks and tutorials, a FAQ, other MPI pages

How to program with MPI

- MPI is a library
  - All operations are performed with subroutine calls
  - Basic definitions are in
    - mpi.h for C/C++
    - mpif.h for Fortran 77 and 90
    - MPI module for Fortran 90 (optional)

Basic Features of MPI Programs

- Calls may be roughly divided into four classes:
  - Calls used to initialize, manage, and terminate communications
  - Calls used to communicate between pairs of processors. (Pair communication)
  - Calls used to communicate among groups of processors. (Collective communication)
  - Calls to create data types.

Is MPI Large or Small?

MPI is large. MPI-1 is 128 functions. MPI-2 is 152 functions.

  - MPI's extensive functionality requires many functions
  - Number of functions not necessarily a measure of complexity

- MPI is small (6 functions)
  - Many parallel programs can be written with just 6 basic functions.
- MPI is just right
  - One can access flexibility when it is required.
  - one need not master all parts of MPI to use it.

MPI basic functions (subroutines)

MPI_INIT: initialize MPI

MPI_COMM_SIZE: how many PE ?

MPI_COMM_RANK: identify the PE

MPI_SEND : send data

MPI_RECV: receive data

MPI_FINALIZE: close MPI

- All you need is to know this 6 calls

Compiling MPI Programs

- NO STANDARD: left to the implementations:
- Generally:
  - You should specify the appropriate include directory (i.e. -I/mpidir/include)
  - You should specify the mpi library (i.e. -L/mpidir/lib -lmpi)
- Usually MPI compiler wrappers do this job for you. (i.e. Mpicc)
    - Check on your machine...

Running MPI programs

- The MPI-1 Standard does not specify how to run an MPI program, just as the Fortran standard does not specify how to run a Fortran program.
- Many implementations provided mpirun –np 4 a.out   to run an MPI program
- In general, starting an MPI program is dependent on the implementation of MPI you are using, and might require various scripts, program arguments, and/or environment variables.
- **mpiexec <args>** is part of MPI-2, as a recommendation, but not requirement, for implementors.
- Many parallel systems use a *batch* environment to share resources among users
- The specific commands to run a program on a parallel system are defined by the environment installed on the parallel computer

A few notes about LAM-MPI

- L AM is a daemon based implementation of MPI.
- You boot LAM, it spawns daemons on your cluster machines.
  - lamboot -v hostfile
- You execute MPI programs while the LAM daemons exist in the background.
  - mpirun -np my_program
- Finally at the conclusion of the MPI session, you shutdown LAM.
  - lamhalt

Problems we run into..

- Firewall prevented to boot the LAM daemons on clients
  - Solution: turn off firewall on Worker node
- On two clusters lamboot complains about port 544 blocked...
  - Reason: Different LAM packages on different machines;
  - Solution: re-install LAM-MPI on the problematic nodes
  - Advanced solution (to try in the lab)
    - Use just one single installation of the LAM-MPI on all the cluster..

References

- www.democritos.it/hpc-wiki
- All the material will be made available for this workshop !