Latinamerican School for Computational Materials Science.

Introduction to MPI

Stefano Cozzini

CNR-INFM DEMOCRITOS, Trieste

Santiago, Chile - Enero, 2009

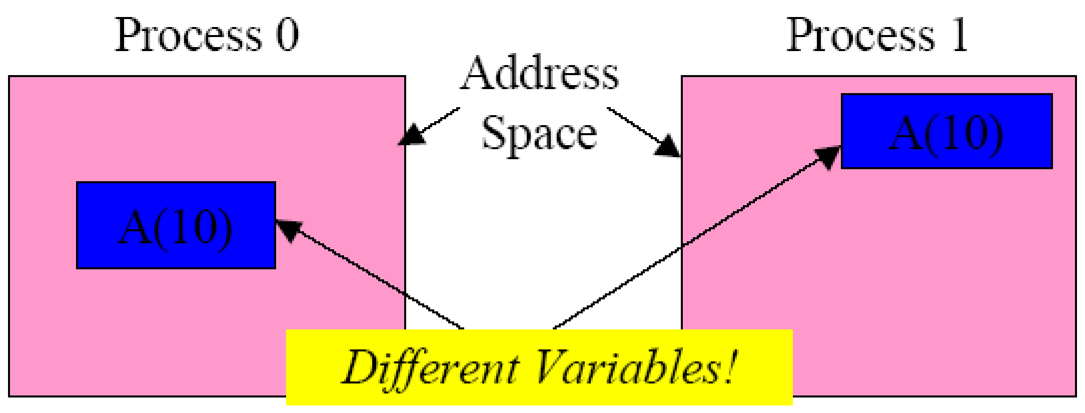
Agenda

- What is MPI ?
- Implementations of MPI
- Compiling/Running MPI programs..
- Setting up an MPI cluster

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   www.lam-mpi.org
  - MPICH /MPICH2
  - OpenMPI (MPI-2) (today's TOY)

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

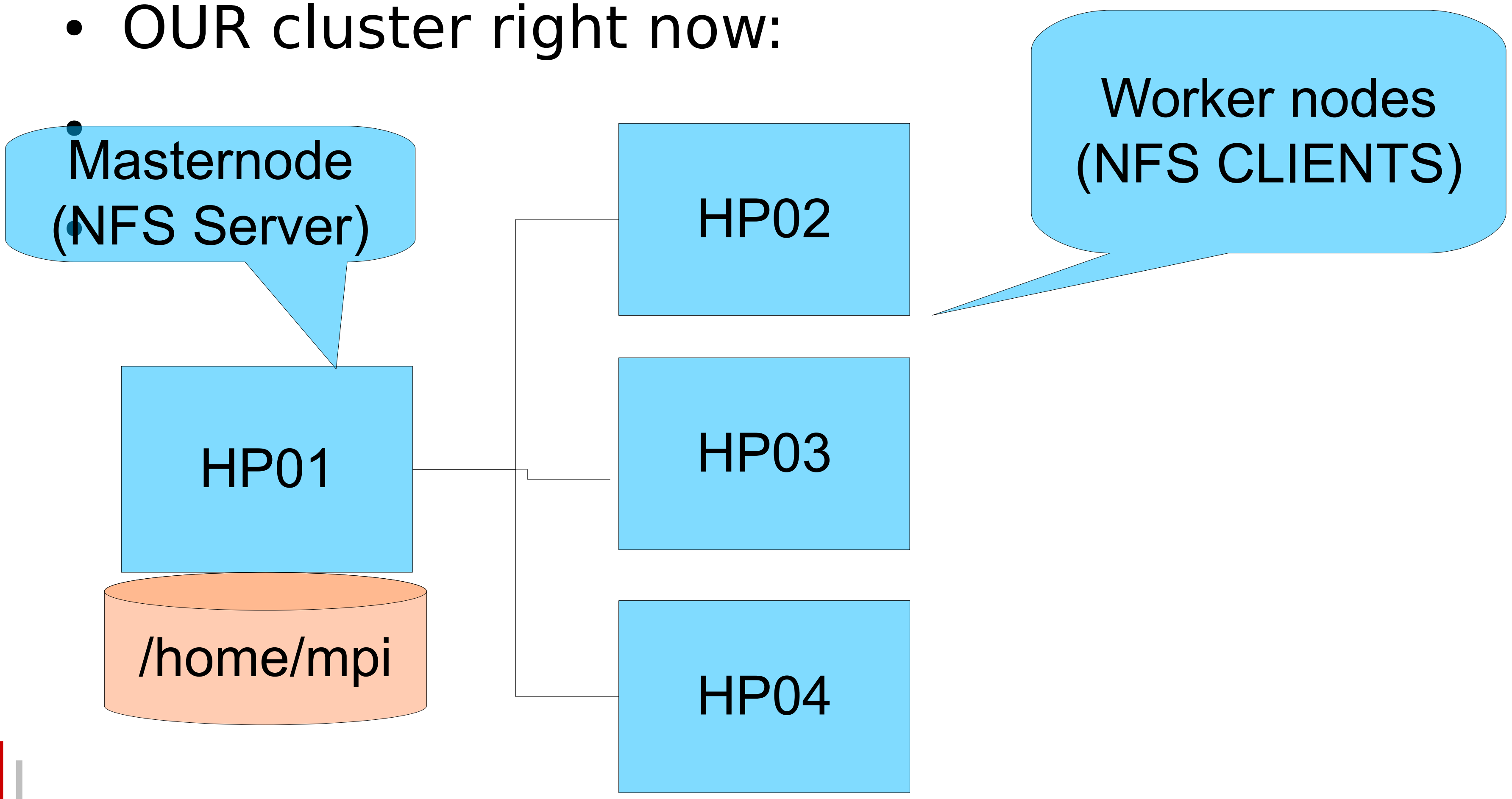
- The MPI-1 Standard does not specify how to run an MPI program, just as the Fortran standard does not specify how to run a Fortran program.
- Many implementations provided mpirun –np 4 a.out   to run an MPI program
- In general, starting an MPI program is dependent on the implementation of MPI you are 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

Setting up a MPI cluster...

- Configure a basic Beowulf cluster:
  - Identify a masternode + some Worker Nodes
  - The Masternode is offering a central service NFS
  - We create 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 openMPI to use all the CPUS of our cluster
  - We will run some benchmarks

Schematic view of our cluster

- OUR cluster right now:



remarks/problems on creating the MPI account

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

Passwordless mechanisms.

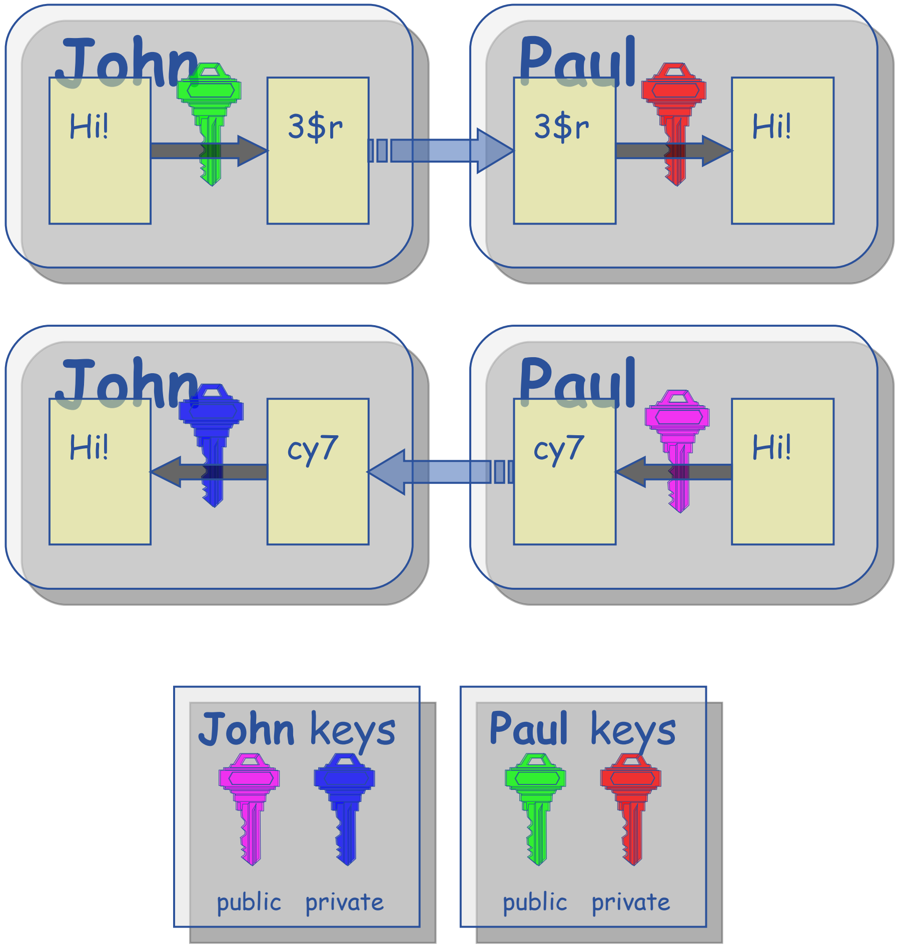
- In order to run jobs on the cluster, we need to set up passwordless login for internal cluster connections only.
- Security note:
  - DO NOT share the keys that you produce in this step with other hosts, and do not copy your keys from other hosts to this cluster.
- The two commands:
  - ssh-keygen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

openMPI installation

- Debian does it for us..
- Be sure that everything is ok..