PS3 Programming Cluster

Outline

- Cluster setup
- Message Passing Interface (MPI)
- Case study
- Homework/project

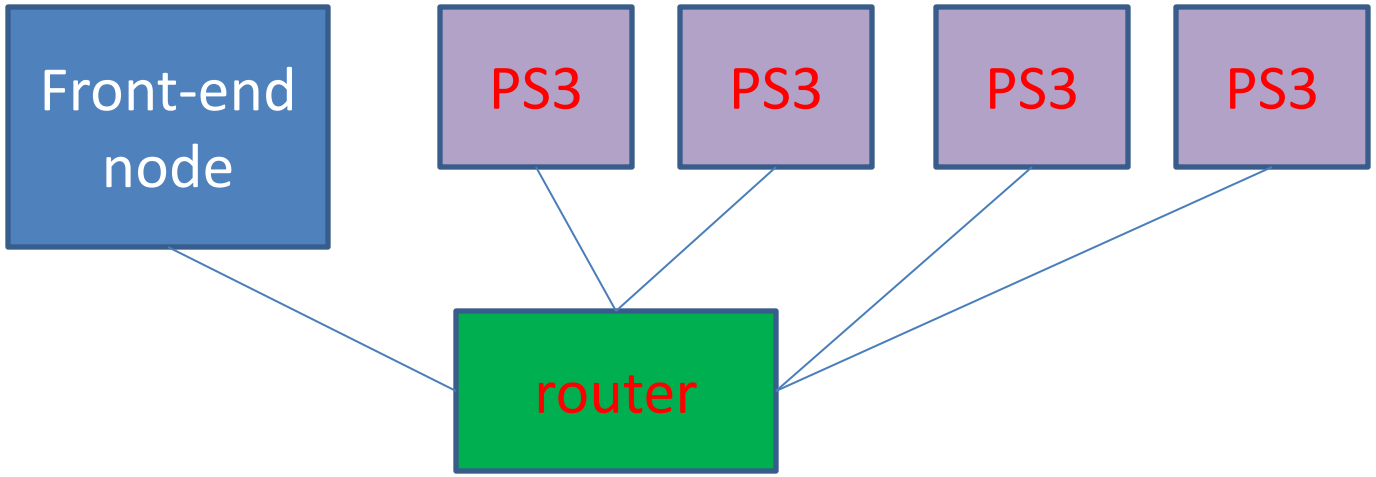
CLUSTER SETUP

Check list

- Physical nodes setup
- Network set up
- User/group accounts
- Network file system
- Job scheduler
- MPI

Network Configuration

- Network setup
  - Better security and scalability



Network system configuration

- Static IP address `/etc/sysconfig/network-scripts/ifcfg-eth0`
  - DNS is set by editing `/etc/resolv.conf`
  - A list of nodes is set by editing `/etc/hosts`
  - Host name is set by `$hostname node01`
- Dynamic IP: DHCP

```
DEVICE=eth0
BOOTPROTO=static
HWADDR=xx:xx:xx:xx:xx:xx
IPADDR=192.168.1.10
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
ONBOOT=yes
NAME=eth0
```

```
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=xx:xx:xx:xx:xx:xx
ONBOOT=yes
NAME=eth0
```

User/user group

- All the users and user groups appear on every node with the same name and ID
  - UID for users and GID for groups
  - Create a new user by adduser
  - You can edit them in the files in /etc/, such as passwd, shadow, group and gshadow
- We will not do that manually. Use the package NIS (next slide).

Network Information Service

- NIS server (ypserv) is installed on the front-end node; NIS clients (ypbind) are installed on each PS3
- Once the NIS is set up, new users need only be added to the front-end node

- More details in http://tldp.org/HOWTO/NIS-HOWTO/index.html

Network file system.

- Provides shared disk volumes to cluster users
- NFS server is on the front-end node.
  - The physical disks are in the front node.

- More details: http://tldp.org/HOWTO/NFS-HOWTO/index.html

Job scheduler

- used to manage the usage of a cluster by different users through a number of queues where jobs can be submitted.
- OpenPBS scheduler that can be obtained from http://www.openpbs.org/

MPI installation

- Message Passing Interface (MPI) is one of the programming tools in cluster environment
  - Including compilers and APIs
- Choices: MPICH1, MPICH2, Open MPI
  - http://www-unix.mcs.anl.gov/mpi/mpich1/
  - http://www-unix.mcs.anl.gov/mpi/mpich2/
  - http://www.open-mpi.org/

MESSAGE PASSING INTERFACE

Parallel computation model

- On clusters (distributed memory systems), the parallel computation is usually carried out by message passing method.
- Two language independent protocols
  - MPI (Message Passing Interface): for homogeneous systems
  - PVM (Parallel Virtual Machine): for heterogeneous systems

MPI components

- Virtual topology
- Synchronization
- Communicator
- Communication functions (over 100)
  - Point-to-point basics
  - Collective basics
- Derived data type

Six basic functions

- MPI_Init (&argc, &argv); /* starts MPI */
- MPI_Comm_rank (MPI_COMM_WORLD, &rank); /* get current process id */
- MPI_Comm_size (MPI_COMM_WORLD, &size); /* get number of processes */
- MPI_Send()
- MPI_Receive()
- MPI_Finalize();

MPI_Send and MPI_Recv

- **int MPI_Send(void** **buf*, **int** *count*, **MPI_Datatype** *datatype*, **int** *dest*, **int** *tag*, **MPI_Comm** *comm* **);**

- **int MPI_Recv(void** **buf*, **int** *count*, **MPI_Datatype** *datatype*, **int** *source*, **int** *tag*, **MPI_Comm** *comm*, **MPI_Status** **status* **);**

Compilation and execution

- After installing MPI, the system will create compilers for MPI
  - mpicc, mpif77, mpif90
- Use mpixxx compiler to compile the program
- Use mpirun to execute the compiled program
  - Specify the number of processors, etc

CASE STUDY

The SUMMA Algorithm

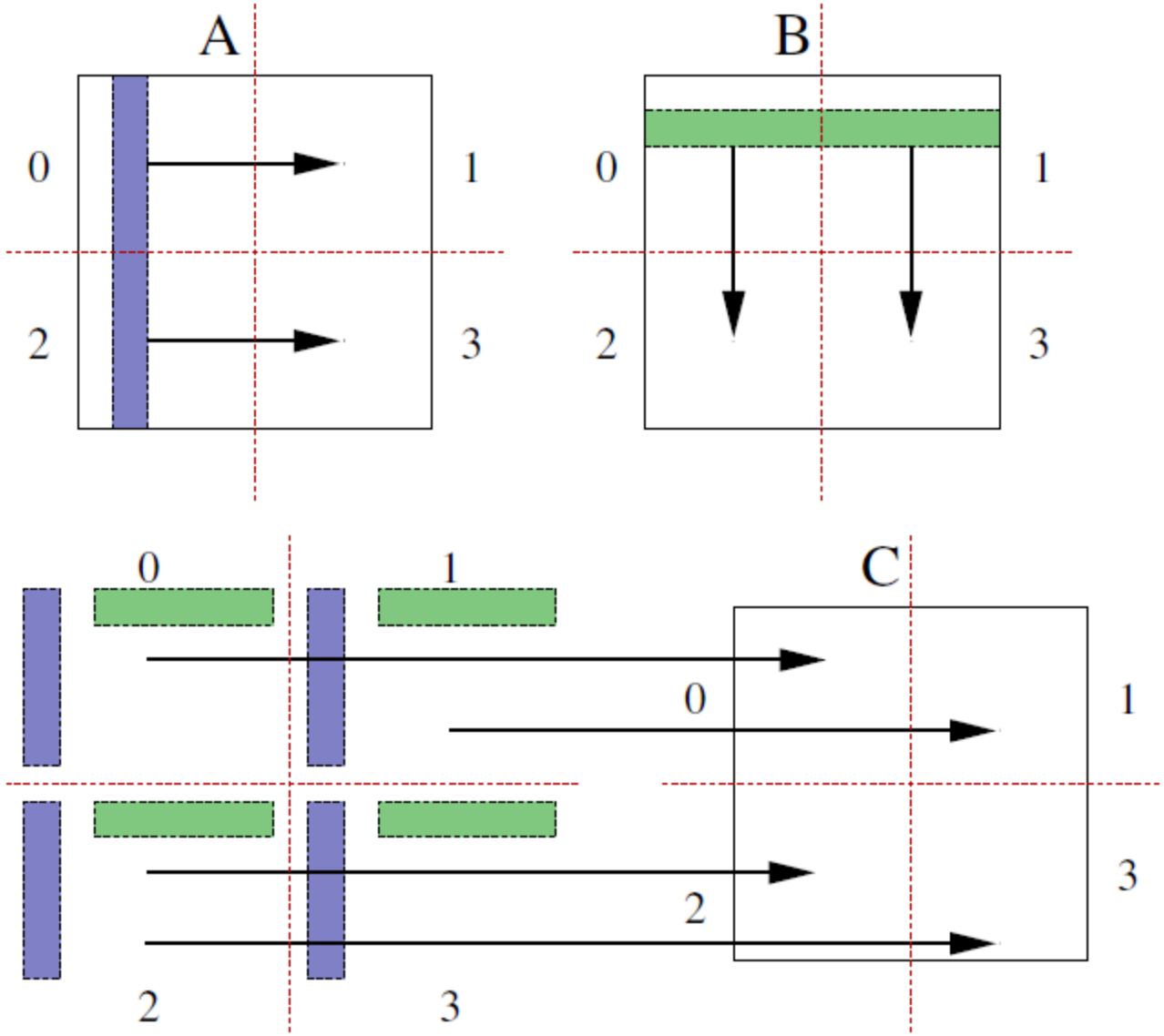
- SUMMA is an algorithm for parallel matrix multiplication
- A. Buttari, J. Kurzak, and J. J. Dongarra. Lapack working note 185: Limitations of the PlayStation 3 for high performance cluster computing.
  - http://www.netlib.org/lapack/lawnspdf/lawn185.pdf

Algorithm

Algorithm 1 SUMMA

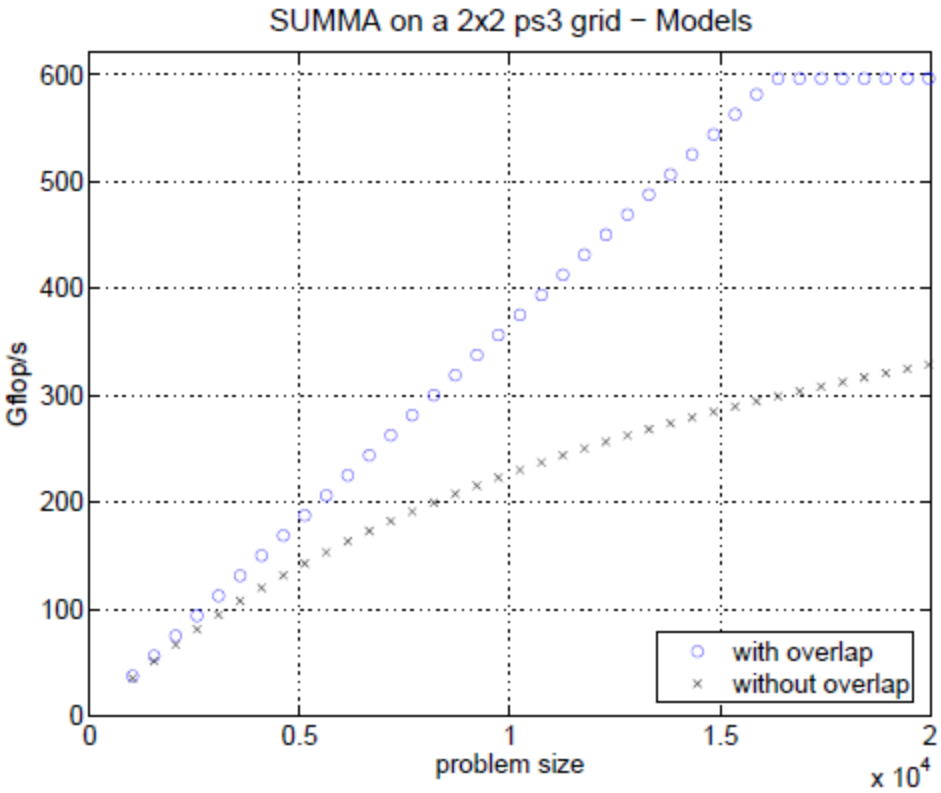
1: **for** $i = 1$ to n/nb **do**
2: **if** I own A_{*i} **then**
3: Copy A_{*i} in *buf1*
4: Bcast *buf1* to my proc row
5: **end if**
6: **if** I own B_{i*} **then**
7: Copy b_{i*} in *buf2*
8: Bcast *buf2* to my proc column
9: **end if**
10: $C = C + buf1 * buf2$
11: **end for**

2x2 Example



Overlapping comm/computation

- PPU does the communication; SPUs do the computation.
- In PPU, using double buffering
  - In step k, broadcast data in step k+1



HOMEWORK/PROJECT

Project

- Buildup the PS3 cluster
- Implement SUMMA algorithm on it
  - Try the overlap and non-overlap version