CS542200
Parallel Programming

BO-YU, KUO

2016/10/03

Outline

- Introduction to the platform
- Compile and execute program on the platform
- Benchmark

Outline

- Introduction to the platform
- Compile and execute program on the platform
- Benchmark

About the platform-Hardware

DEBUG CLUSTER

1 + 8 nodes

each node has:

- 24GB memory
- 2TB HDD Storage
- 2 x 6-core
  Intel(R) Xeon(R) CPU
  L5640 @ 2.27GHz

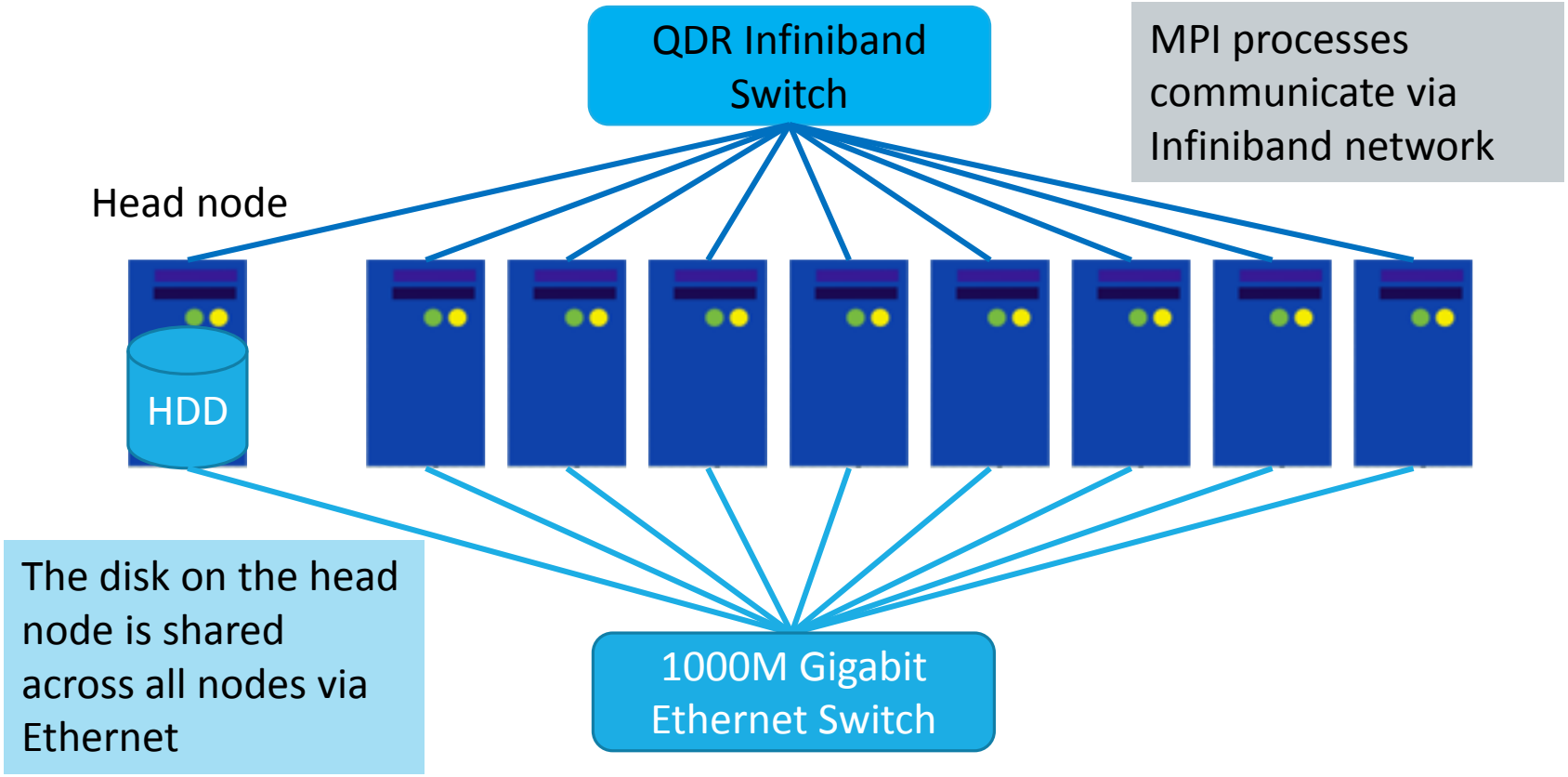
BATCH CLUSTER

1 + 10 nodes

each node has:

- 96GB memory
- 2TB HDD Storage
- 2 x 6-core
  Intel(R) Xeon(R) CPU
  X5670 @ 2.93GHz

The network configuration



Network File System (NFS)

NFS is a **distributed file system protocol.**

It defines a standard network file access protocol interface, but it is **not a real file system.**

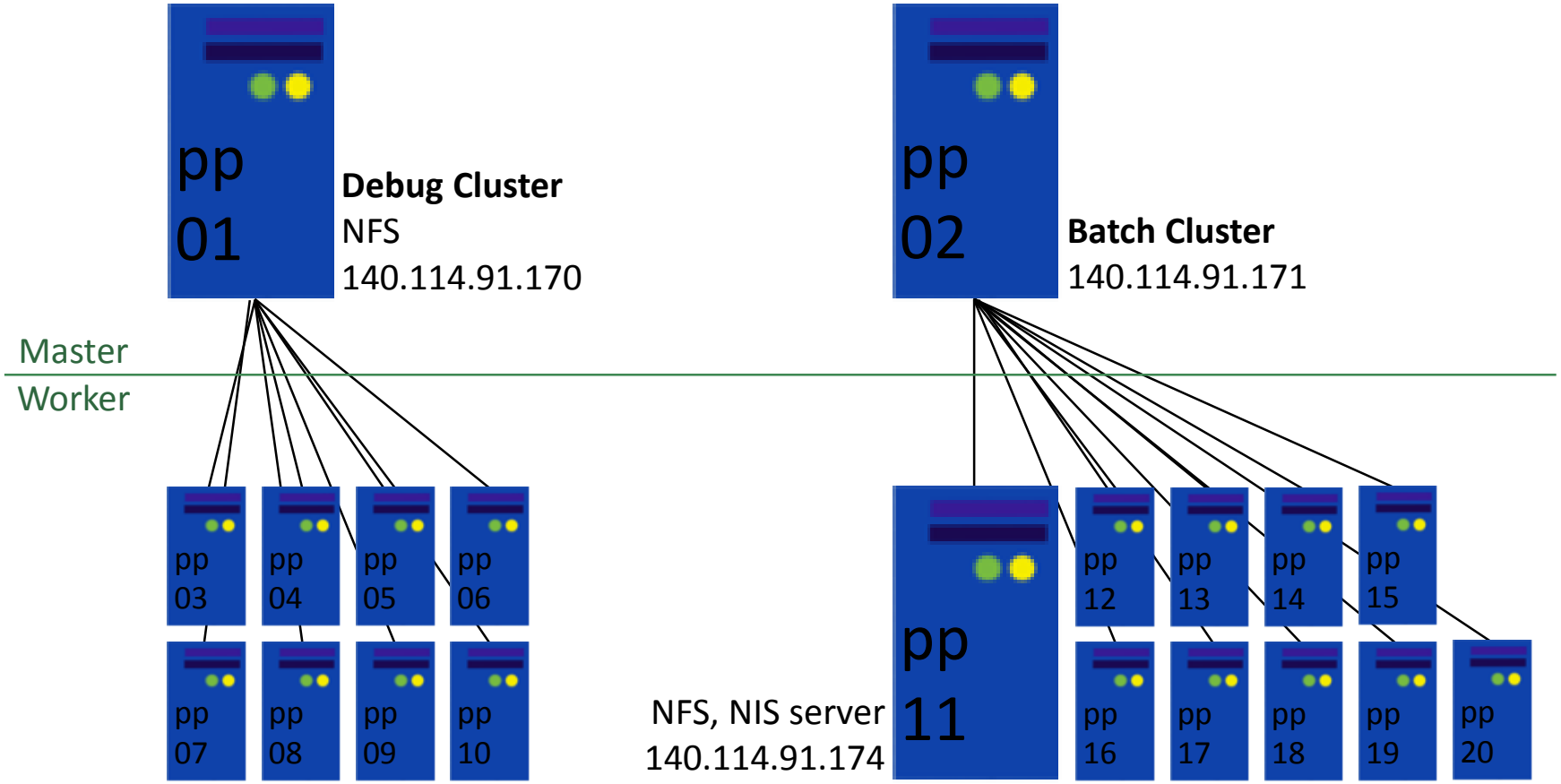
It allows us to **mount a remote file system** as a local file system for **data sharing.**

Network Information Service(NIS)

All machines use **the same authentication data** to do authentication

Only need to create user and password on the administration server

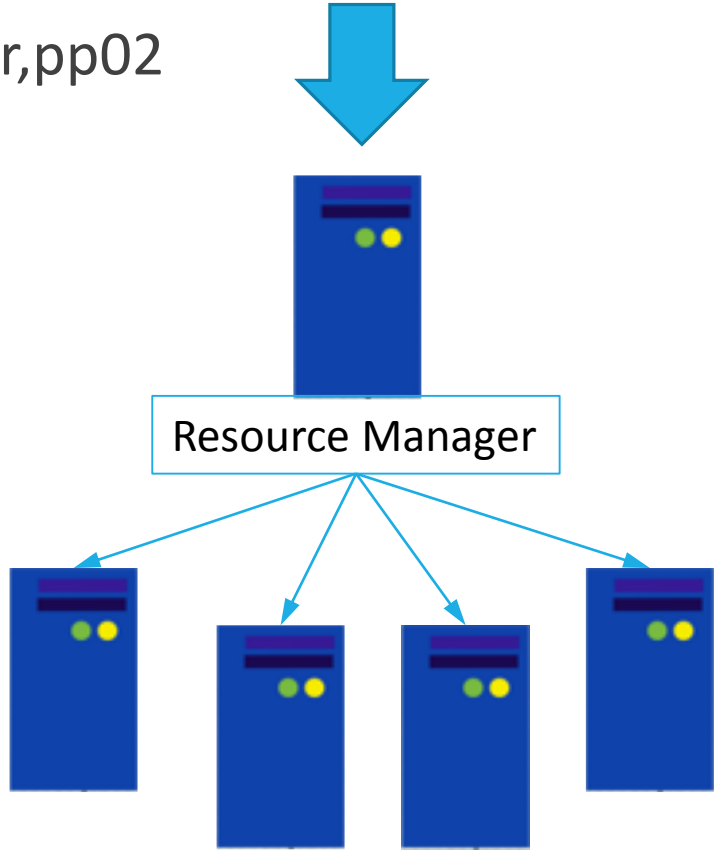
Parallel Programming platform



User view on the server

On each cluster, there are:

- 1 log-in node (pp01 for debug cluster,pp02 for batch cluster)
  - login
  - write/compile codes
  - submit jobs
- several computing nodes
  - run jobs



DO NOT ssh TO COMPUTING NODES DIRECTLY!
This will affect other legal users!
If we ever find you logged in to computing node illegally, you will get 0 point for the HW!
You can only log in to pp01 and pp02.

Login to server: from Linux

Open terminal first

SSH login

- ssh USER@HOST [-X] [-C]
- -X: enable X11 window forwarding
- -Y: enable trusted X11 window forwarding
- -C: enable compression (can speedup connection when using X window)

SFTP file transfer

- scp [-r] [-C] [[USER@]HOST1:]PATH1 [[USER@]HOST2:]PATH2
- -r: recursive (for directory)
- -C: enable compression
- e.g.

```
scp my_file s104567890@140.114.91.170:hw/
```

Login to server: from Windows

SSH login: **Putty or Pietty**

- Putty: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
- Pietty: http://ntu.csie.org/~piaip/pietty/download

SFTP file transfer: **FileZilla**

- FileZilla: https://filezilla-project.org/

Alternatively, you can use **MobaXterm** which integrates all features above! (recommended)

- MobaXterm: http://mobaxterm.mobatek.net/download-home-edition.html

Parallel Programming Platform

IP address

- If you want to submit job for quick debugging purpose: **140.114.91.170**
- If you want to submit job for benchmarking purpose : **140.114.91.171**

account

- ID: **studentID**
  - E.g. 105012345
- If your student id starts with ‘t’ ,’e’=> ID**: studentID (without t and e)**
  - E.g. 105012345

password

- (will announce on Lab)

Change your password

You are required to change your password at the 1st login.

You can also change you password in the future by typing

- `passwd`

Home directory

After login, you will see batch directory like below:

```
batch -> /pp11-home/pp2016-batch/101062229
```

will be your account

Because we have 2 NFS server:

- If you want to submit to debug server

-> use the original directory

- If you want to submit to batch server

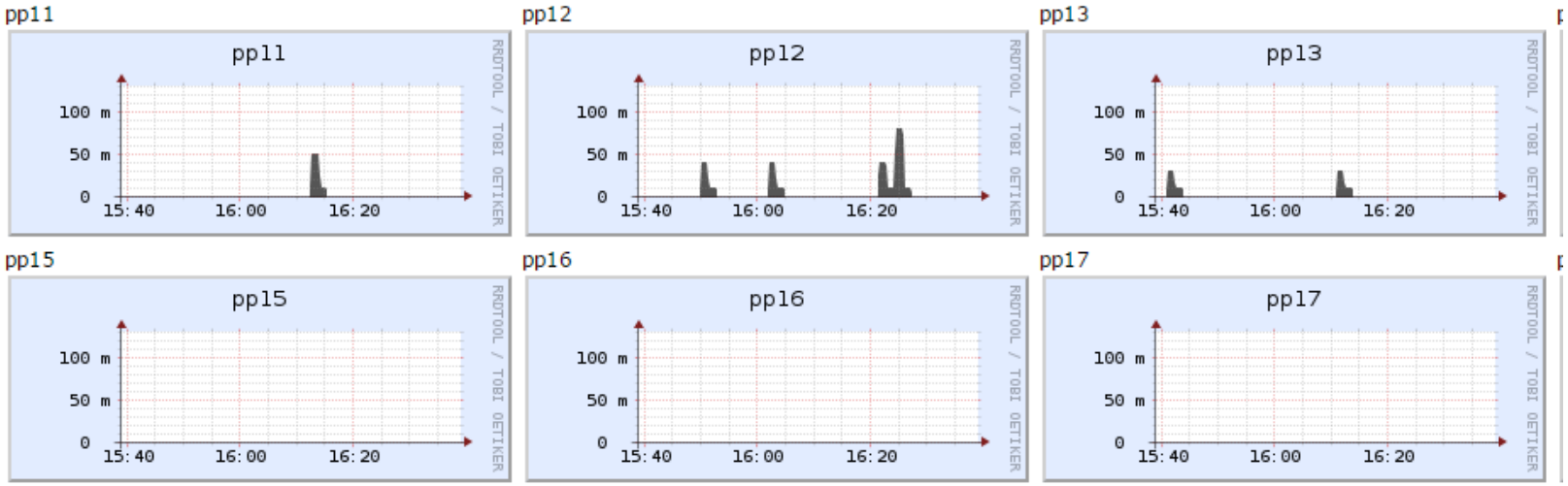
-> you can also use the original directory, but copy all the file you want to run to batch directory may run faster

Ganglia

Ganglia is a scalable distributed monitoring system for high-performance computing systems.

You can use websites below to see the status of both debug and batch cluster.

- http://140.114.91.170/ganglia/ status of debug cluster
- http://140.114.91.171/ganglia/ status of batch cluster



Outline

- Introduction to the platform
- Compile and execute program on the platform
- Benchmark

Compile and Execute Parallel Program

MPI

- Compile: mpicc MPI_CODE.c [-o MPI_EXE]
- mpic++ MPI_CODE.cpp [-o MPI_EXE]
- Execute (job queue): mpiexec ./MPI_EXE
- Execute (directly): mpirun [-n N_PROCS] [-hostfile HOST_FILE] ./MPI_EXE
- **NOTE: DO NOT execute MPI on headnode directly!**
- **=> Submit your job through resource manager.**

Pthread

- Compile: gcc CODE.c [-o EXE] -pthread
- g++ CODE.cpp [-o EXE] -pthread

OpenMP

- Compile: gcc CODE.c [-o EXE] -fopenmp
- g++ CODE.cpp [-o EXE] -fopenmp

Job Queues

Resource Manager: TORQUE-5.1.1.2

Scheduler: Maui-3.3.1

There are 2 queues in the system:

- **debug** for quick debugging purpose
- **batch** for benchmarking purpose

NOTE: Job submitted from pp01 will run on debug node!
Job submitted from pp02 will run on batch node!

Job Queues: constraints

debug --- for quick debugging purpose

- Max nodes = 2
- Max total processes = 24
- Max walltime = 5 minute
- Max jobs queuable at any time = 2
- Max jobs runnable at any time = 1

batch --- for benchmarking purpose

- Max nodes = 4
- Max total processes = 48
- Max walltime = 30 minutes
- Max jobs queuable at any time = 8
- Max jobs runnable at any time = 2

Job Queues: priority

The scheduler will

- favor **short running** jobs (based on walltime)
- favor **less resource demanding** jobs (based on nodes, ppn)
- favor jobs which are **queued for a long time**

If you submit job to debug server, you will run with others' program. But if you submit job to batch server, you will not run with others' program.

Be sure to request *reasonable* amount of resources according to your own requirements.

Before Submit a Job: password-less ssh

You just need to do this once! (Type the below command after $)

```
$ ssh-keygen -t rsa
```

- Press **enter** through all questions

```
$ cd ~/.ssh

$ cat id_rsa.pub >> authorized_keys

$ chmod 400 authorized_keys

$ /home/pp2016/shared/copy_ids.sh

$ cd
```

ssh to pp02 to see if you don’t need to type password now !

Before Submit a Job: Job script

Login to server and copy lab1 directory to your home directory

- cp -r /home/pp2016/shared/lab1 . && cd lab1
- vim job.sh

JOB_NAME

nodes: How many nodes
ppn: process per node
Total processes = nodes x ppn

Max time to run

It will go to the directory where you submit your job

```
#PBS -N MY_JOB
#PBS -r n
#PBS -l nodes=2:ppn=2
#PBS -l walltime=00:01:00
#PBS -e /mypath/error.txt
#PBS -o /mypath/output.txt

cd $PBS_O_WORKDIR
mpiexec ./executable args
```

Specify name and path of error and output file

more flags: http://www.democritos.it/activities/IT-MC/documentation/newinterface/pages/runningcodes.html

Processes Layout (nodes & ppn)

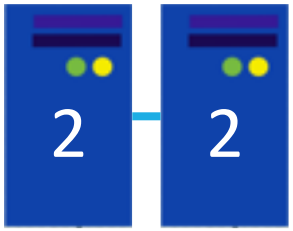
For example, how to request **4** processes?
There are 3 possible ways:

nodes=4:ppn=1

In this case, the performance my suffer.

Use this if you want to observe network overhead.

nodes=2:ppn=2



Hybrid parallelism

Use this when you have MPI + OpenMP

nodes=1:ppn=4

For Pthread & OpenMP only this configuration works.

NOTE: ppn must <= 12, because we only have 12 cores per node

Submit a Job: Job control

Submit:

- qsub JOB_SCRIPT.sh

Kill jobs:

- qdel JOB_ID [JOB_ID2 [JOB_ID3…]]
- qdel all

Monitor:

- qstat -a

Again, **DO NOT TRY TO ssh DIRECTLY TO COMPUTING NODES!**

Job state (qstat -a)

```
                                                Req'd    Req'd        Elap
eue    Jobname         SessID  NDS   TSK    Memory   Time    S   Time
------ --------------- ------ ----- ------ ------ --------- - ---------
tch    TEST3            22284     4     96     2gb  00:05:00 C       --
bug    TEST             22464     2      4     2gb  00:01:00 C       --
tch    TEST3              --      4     96     2gb  00:05:00 Q       --
tch    TEST4               0      4     96     2gb  00:01:00 R  00:00:21
tch    TEST4              --      4     96     2gb  00:01:00 Q       --
```

C: Completed

R: Running

Q: Queuing

Practice time

You should be able to see these files in your lab1 directory:

- HelloWorld.c
- job.sh
- README.md

The program HelloWorld.c will print " Hello world", the hostname of the running node, and rank number of your processes.

[Compile] `mpicc HelloWorld.c -o HelloWorld`

[Edit job script] vim job.sh:

- `mpiexec ./HelloWorld`

[Run] `qsub job.sh` (You can send your job from pp01 or pp02.)

Parallel version of Hello world

1. **Use 1 node and 4 ppn, submit the job to debug server, you should see the result like below. Because of the scheduler, you may run on other nodes(pp03~pp10)**

hostname of the node this program is running on

rank number of the process

```
Hello world from node pp10, rank 1 out of 4 processes
Hello world from node pp10, rank 0 out of 4 processes
Hello world from node pp10, rank 3 out of 4 processes
Hello world from node pp10, rank 2 out of 4 processes
```

Because you are running a parallel program, the execution order of the processes will not be the same !!!

Parallel version of Hello world

2. Use 2 node and 4 ppn, submit the job to batch server, you should see the result like below. Because of the scheduler, you may run on other nodes.(pp11~pp20)

hostname of the node this program is running on

rank number of the process

```
Hello world from node pp20, rank 1 out of 8 processes
Hello world from node pp19, rank 6 out of 8 processes
Hello world from node pp19, rank 5 out of 8 processes
Hello world from node pp20, rank 0 out of 8 processes
Hello world from node pp20, rank 2 out of 8 processes
Hello world from node pp19, rank 7 out of 8 processes
Hello world from node pp20, rank 3 out of 8 processes
Hello world from node pp19, rank 4 out of 8 processes
```

Because you are running a parallel program, the execution order of the processes will not be the same !!!

Outline

- Introduction to the platform
- Compile and execute program on the platform
- Benchmark

Benchmark

What is a benchmark?

- Benchmark is a specially designed program to measure the performance of the computer.
- The design goal of a benchmark is to meet and show requirements of real applications.

Why do we need benchmark?

- To have a publicly accepted method to measure the performance of different computers.
- To find out the bottleneck of the computer.

Some famous HPC benchmark:

- HPCC
- HPL/Linpack
- HPCG

HPL

High Performance Linpack

- To solve linear matrix equation.

$$Ax = b; \quad A \in \mathbf{R}^{n \times n}; \; x, b \in \mathbf{R}^n$$

- Measure floating point operation per second (FLOPS).
- Used in Top 500
- Main algorithm:
  - Divide a matrix into many pieces
  - All parameter must be determined by user.

HPL tuning – library

MPI

- Mvapich2
- Openmpi
- Intel MPI
- ……

Blas(Basic Linear Algebra Subprograms)

- OpenBLAS
- GotoBlas2
- ATLAS
- ……

Cluster Performance & Efficiency

R-Peak

Definition: CPU Clock Speed * flops/cycle * CPUs * Cores/CPU

Take our batch server for example: 2.93 GHz * 4 * 2 * 6 = 140.64 Gflops

Notice: If the cpu support AVX, then flops/cycle should be 8

If the cpu support AVX2, then flops/cycle should be 16

R-Max

Definition: measurements from the benchmark program

Efficiency

Definition: R-max / R-peak

ex: If R-max = 1.229e+02 Gflops

➔ Efficiency = 122.9/ 140.64 ~= 87.4%

Practice time

Login to server pp02 and copy hpl-2.1 directory to **your batch directory** under your home directory

- cp -r /home/pp2016/shared/hpl-2.1 . && cd hpl-2.1/bin/Nehalem

You should be able to see these files in your hpl-2.1/bin/Nehalem directory:

- HPL.dat (All tuning parameters are in this file.)
- xhpl

HPL.dat

Input Parameters

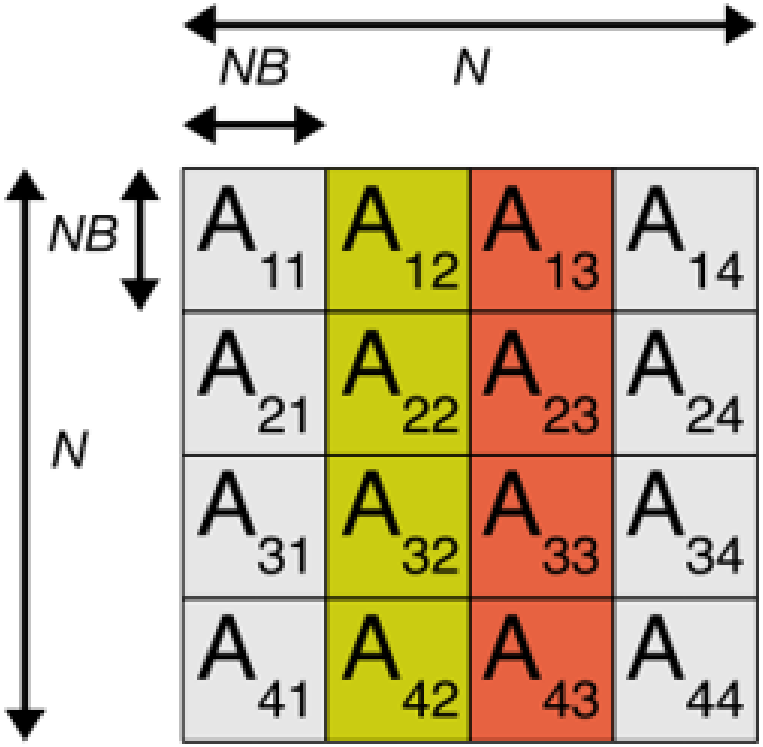
- N : matrix size
- NB : Block size for calculation
- P * Q : process grid ratio

NOTE: P * Q should equal to your total running processes!

Ex: If you use nodes=2:ppn=4 , your total running processes are 2 * 4 = 8, so P * Q should equal to 8

```
1            # of problems sizes (N)
8000         Ns
1            # of NBs
20           NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
1            Ps
1            Qs
```

HPL Input parameter



Running HPL

You can copy job.sh script to your hpl-2.1/bin/Nehalem directory

[Edit HPL.dat] vim HPL.dat:

- Modify N,NB,P,Q for tuning

[Edit job script] vim job.sh:

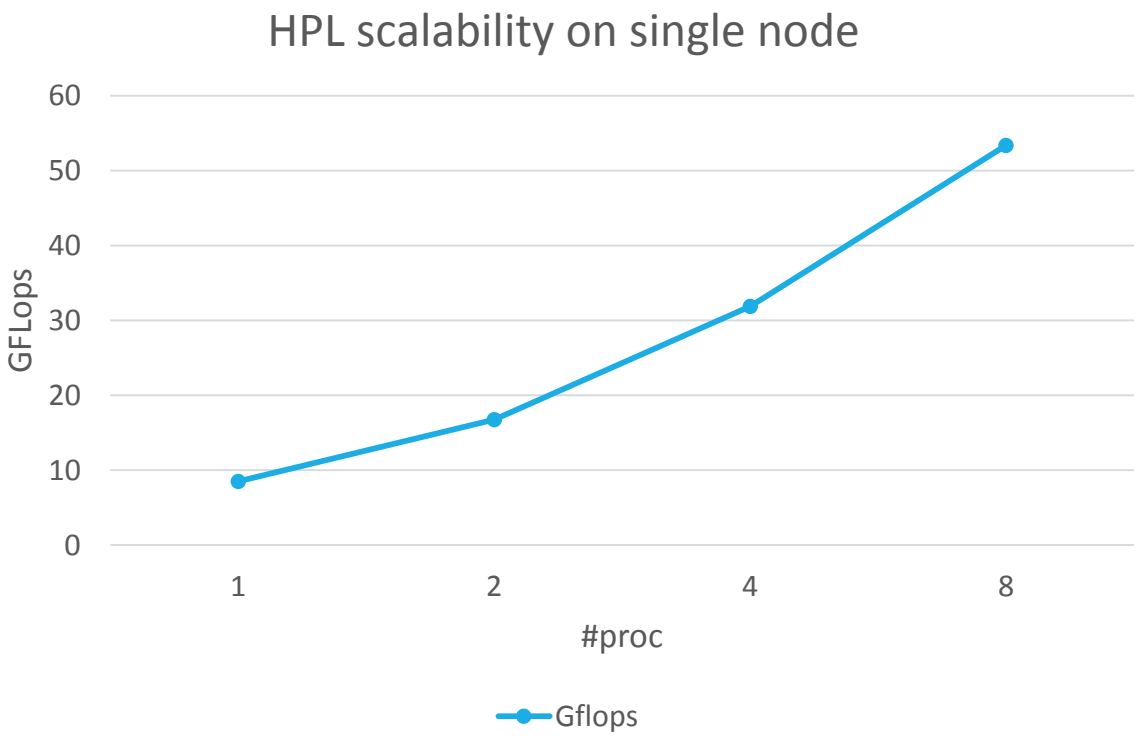
- `mpiexec ./xhpl`
- `(Change number of nodes and ppn to fit your need.)`

[Run] `qsub job.sh` (You can send your job from pp01 or pp02.)

NOTE: If you change the number of nodes or the number of processes, you have to modify both P,Q in HPL.dat and node number and ppn in job.sh

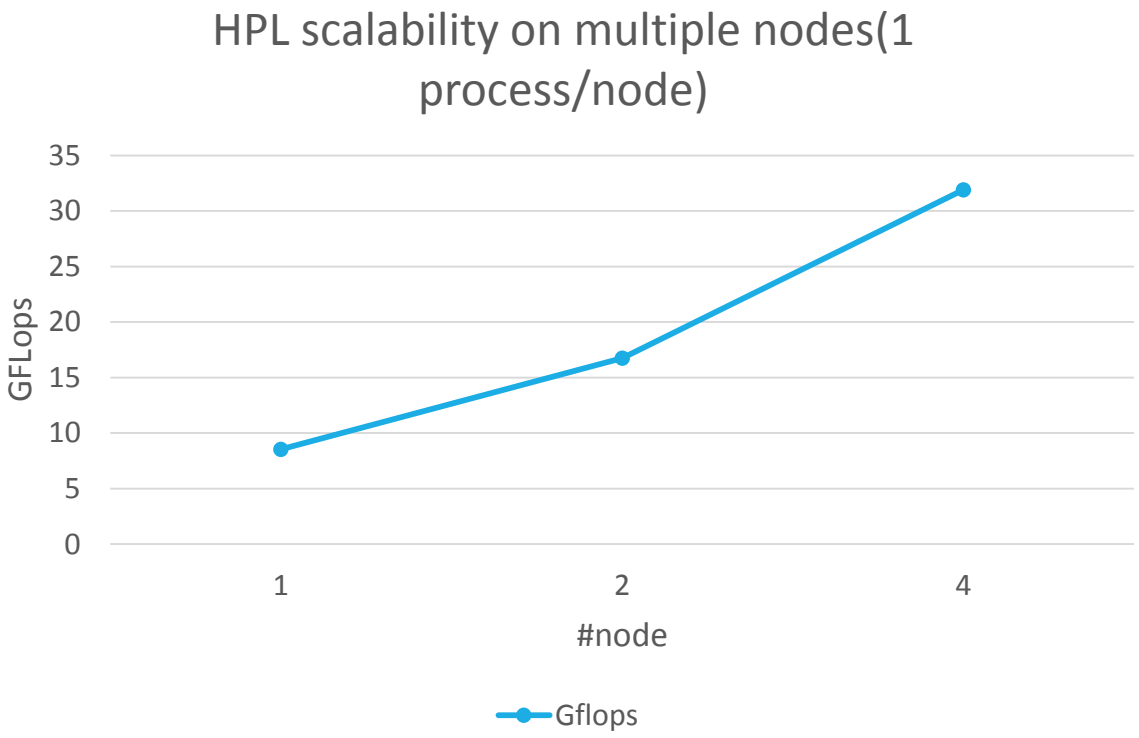
Assignment1-scalability

a. Scaling on a single node by using #process: 1, 2, 4, 8 and plot it like below: (use the default N (8000) and NB(20) in HPL.dat !!)



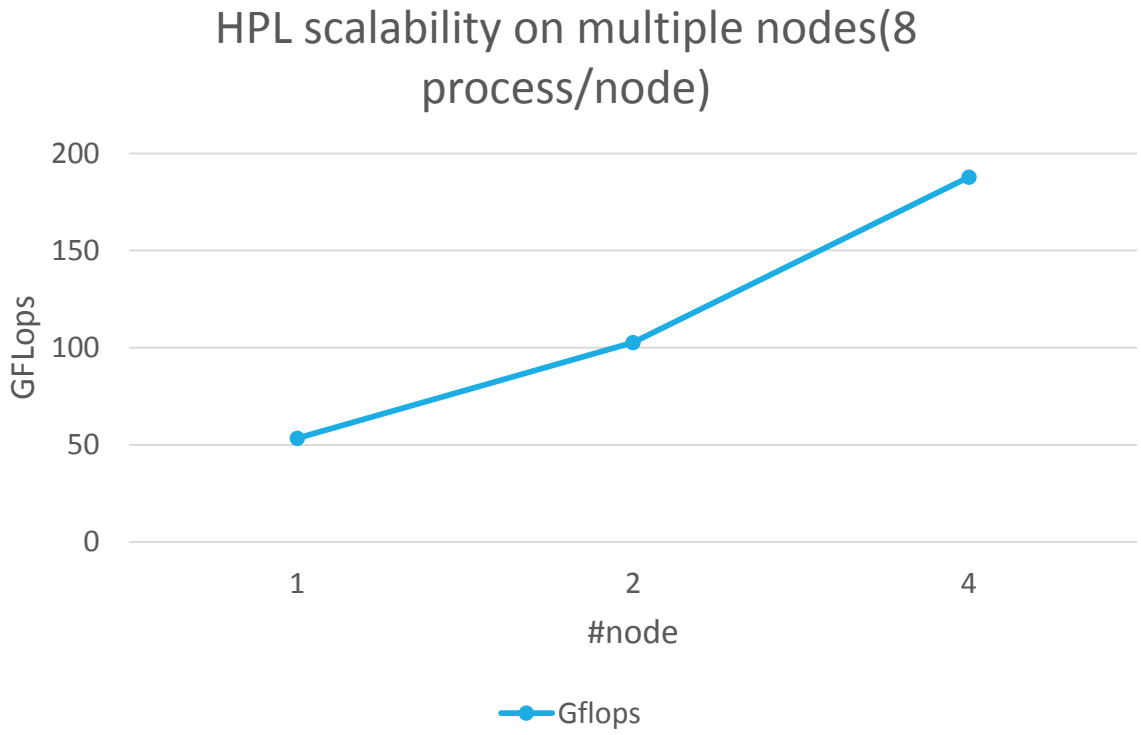
Assignment1-scalability

b. Scaling across server by using #node: 1,2,4 (1 process/node) and plot it like below: (use the default N (8000) and NB(20) in HPL.dat !!)



Assignment1-scalability

c. Scaling across both process & node by using #node: 1, 2, 4 (12 process/node) and plot it like below: (use the default N (8000) and NB(20) in HPL.dat !!)



Assignment 1

Please finish lab1 assignment1 before leaving the Lab.

If you cannot finish it in lab, please send an email of the 3 scalability plots to tiffanykuo@lsalab.cs.nthu.edu.tw

before 10/13 23:59

Assignment2-tuning

Adjust N,NB,P,Q in HPL.dat or build your own HPL with different library to get your best performance result !

- You can not use your own server/computer to run.
- Upload :
  - STUDENTID_HPL.out (ex: 105062553_HPL.out): the output file of HPL
  - STUDENTID_library.txt (ex: 105062553_library.txt): If you build your own HPL with different library, write a simple txt file about what library you use. (If you use the HPL we built, you don't need to upload this file.)

  to ILMS before 10/13 23:59
- Bonus points will be given for top performance.