Using ISMLL Cluster

Tutorial

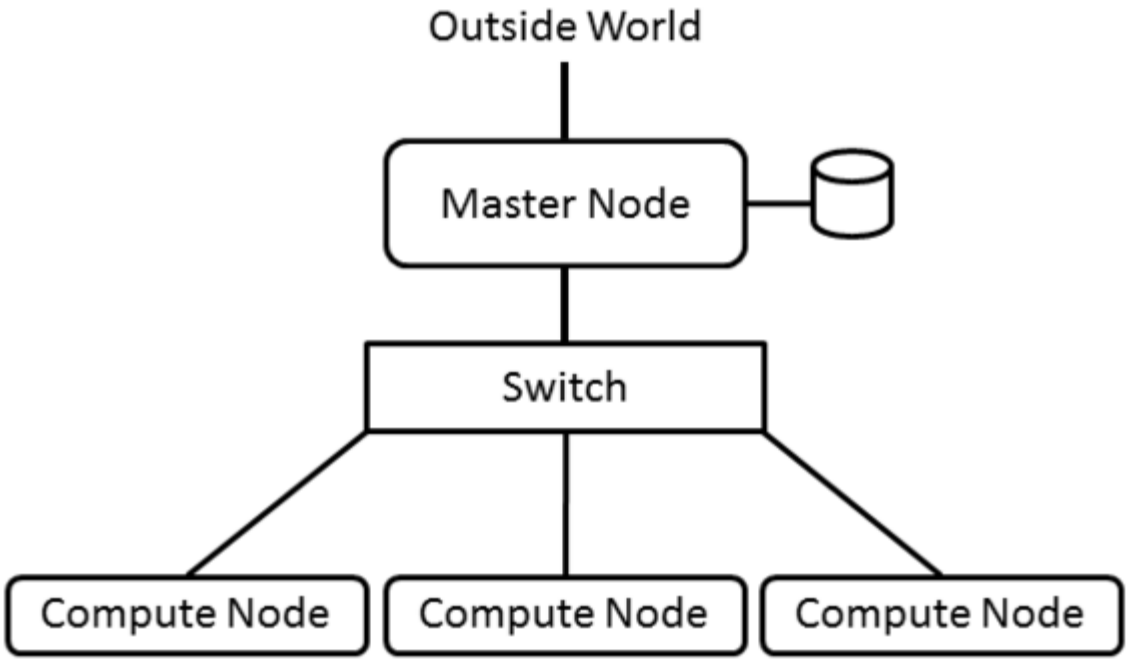
Lec 5

Mohsan Jameel, Information Systems and Machine Learning Lab, University of Hildesheim

Agenda

- Hardware
- Useful command
- Submitting job

Computing Cluster



http://www.admin-magazine.com/HPC/Articles/Building-an-HPC-Cluster

- Any problem or query regarding cluster access can be directed to me (mohsan.jameel@ismll.de) using email subject

  subject: bd-16 ismll cluster issue

Login to headnode

- To log into cluster do ssh to headnode
  - ssh <user_name>@cluster.imsll.de
- To transfer file from your machine
  - Linux: scp file <user_name>@cluster.imsll.de:~/<directory>
  - Windows: Openssh client, filezilla etc
- Other useful command
  - cd - change directory
  - pwd - show current directory
  - ls - list files and directory
  - touch – create new file
  - vim - command line editor
  - rm - remove file or directory
  - mkdir - create directory
- To view available compute node
  - rocks list host
- To view number of jobs running
  - jc a.k.a jobcount

Managing job

- qsub - submit a job to the batch scheduler
- qstat - examine the job queue
- qdel - delete a job from the queue
- qhost - show execution host status from command line

Submitting job

- Directly using qsub
  - qsub -b y -j n -cwd -p -100 -l mem=2G -o logs/output.$JOB_NAME.$TASK_ID.txt -e logs/output.$JOB_NAME.$TASK_ID.err -N HelloJob java -Xmx1G -XX:ParallelGCThreads=1 -jar HelloWorld.jar
- Using job script
  - chmod 755 seq_java_qsub_script.sh
  - qsub seq_java_qsub_script.sh

qsub script (1/3)

```
#!/bin/bash -e
#seq_java_qsub_script.sh

# Number of slots/cores that need to be utilized are given in the
#following line.
#$ -pe serial 1

#  Setting parameter for qsub command
#You can target different queues by using the -q <name-of-queue>
#$ -q all.q

#set job name (which is shown by qstat and the web frontend)
#$ -N <JOB_NAME>

#you must use the parameter -l mem=xx (xx is the amount in m, M, g or G of memory your job needs). if you don't use the memory request, the job will be declined by the job verifier.
#$ -l mem=2G
```

qsub script (2/3)

#Execute the job from the current working directory; important if you use relative file paths.
#$ -cwd

#Send email on job completion to EMAIL@ismll.de. 'a' abort and 'e' end
#$ -m ae
#$ -M EMAIL@ismll.de

#Makes bash the shell for the job (important if it is a shell script)
#$ -S /bin/bash

for 'n' Do not merge STDIN and STDERR and 'y' merge STDIN and STDERR
#$ -j n

qsub script (3/3)

```
# note that logs directory should exist for both options below
#Write output to FILE
#$ -o logs/output.$JOB_NAME.$TASK_ID.txt

#Write STDERR to FILE
#$ -e logs/output.$JOB_NAME.$TASK_ID.err

#Set priority to -1; priorities > 0 require super user rights
#$ -p -100

# done with parameter setting for qsub

# Now will write Program to execute and parameter related to it
java -Xmx1G -XX:ParallelGCThreads=1 -jar HelloWorld.jar

#end of script
```

Good Practice

- Your home folders are mounted through shared file system. When every your job read or write to your home folder it generates network traffic which passes through head node. If your job has frequent read/write (I/O) it will slow down whole cluster.
- So Avoid puny I/O:
  - If your program has many iteration, avoid to writing output in each iteration. You can buffer them in memory and after certain iteration you can write them in bulk.
  - Avoid to frequent read in your program from file (HDD). You can read large chunk of file once into the memory and then as need can again read more later. Or you can read complete file at once into the memory and perform in-memory operation.