Cluster and Slurm

Please read the following terms **carefully** before you proceed any further into this document:

1. Public server is for public service. Please cherish the public resources and do not use them for other purposes than running the required experiments for the course projects.
2. Upload your files in your personal directories (`/pvfsmnt/<student-id>` and `/home/<student-id>`). Any file outside the authorized paths will be cleaned up without notification and backup. Moreover, **subsequent plagiarism will be punished without any mercy or excuse** if it is caused by someone else accessing your code at unprotected places.
3. We **do not provide any warranty** to the safety and availability to any data stored on the server. Please backup your data constantly and do not upload sensitive personal information. We will not be responsible for any consequences caused by data loss and sensitive data leakage.
4. You are not allowed to share your account to others without the permissions of teaching staffs. Reselling is forbidden. Once confirmed, your account will be removed.
5. Teaching Staffs have the rights to kill your running processes and remove your account without any notification once your behaviors are identified to be disturbing or destructive, including (but not limited to) submitting waste tasks to queue, blocking the login node, creating junk files on server and cracking credentials.

Your login to the server will be regarded as your agreement to the terms above.

> **Notice**
>
> Please manage your time. In principle, under the circumstances that the server is functioning normally, we will not extend DDLs for your not being able to retrieve the results from the submission queue in time.

About the Server

1. There are 8 internal nodes each with one Nvidia 2080Ti GPU card. You will not be able to login to the internal node directly; instead, to run your program, you need to submit your job scripts to **Slurm**.
2. The CPU details are provided as the following:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    1
Core(s) per socket:    16
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
Stepping:              7
CPU MHz:               2100.128
CPU max MHz:           2100.0000
```

```
CPU min MHz:           800.0000
BogoMIPS:              4200.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              22528K
NUMA node0 CPU(s):     0-15
NUMA node1 CPU(s):     16-31
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall
nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl
xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 ds_cpl vmx
smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic
movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm
3dnowprefetch epb cat_l3 cdp_l3 invpcid_single intel_ppin intel_pt ssbd mba
ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid fsgsbase
tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f
avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt
xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm arat
pln pts pku ospke avx512_vnni md_clear spec_ctrl intel_stibp flush_l1d
arch_capabilities
```

3. Each node will have around 251 GB memory.
4. Nodes are interconnected with 16 Gbps ethernet (with infinite-band support).
5. At the master node, there are some ulimits set for you:

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) 262144
scheduling priority             (-e) 0
file size               (blocks, -f) 8388608
pending signals                 (-i) 254782
max locked memory       (kbytes, -l) 8388608
max memory size         (kbytes, -m) 67108864
open files                      (-n) 16384
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 65536
cpu time               (seconds, -t) unlimited
max user processes              (-u) 4096
virtual memory          (kbytes, -v) 536870912
file locks                      (-x) unlimited
```

(cpu time is not running time or wall time of your programs).

They will only affect your behavior on the master node; the resource limits are controlled by slurm for internal nodes.

PVFS

In order to run your MPI programs, you must put your file to a shared FS in order to make them visible to all nodes. We have mounted a PVFS at `/pvfsmnt`. The files appear under `/pvfsmnt` will also appear across all nodes at the same location. To run you program, you can first copy your executable files and related data files to your personal directory under

`/pvfsmnt`, and then submit your job via `slurm`.

Slurm

Resource Limit

You can use

```
sacctmgr show assoc where user=$(whoami)
format=cluster%20,account,partition,qos,maxsubmit,maxjobs,grptres%20,maxtres
%20,maxtresmins
```

to get you resource limit on the cluster.

For example, you may get:

Cluster	Account	Partition	QOS	MaxSubmit	MaxJobs	GrpTRES	MaxTRES	MaxTRESMins
hkjournalist	csc4005	debug	normal	32	8	cpu=128,mem=128G	cpu=64,mem=64G	cpu=640

This means you may at most run 8 jobs in the same time and have 32 jobs waiting in queue. Each job will at most take up 128 cores and 128 GiB. And the sum of all cpu usage can be at most 640 mins.

Interactive Approach

You can start an interactive session via `salloc`

To following command, for example, will prepare a session with 48 cores distributed on two nodes and the session can last for 10 mins:

```
salloc -N2 -n48 -t10
```

If you are acquiring resource more than allowed then your shell will hang there. Please also notice that if you do not set the time-limit to 10, it will use 1 on default; that is, your session will expire in 1 minutes and you will receive a warning about it.

After entering the salloc environment, you can use the following command to examine:

```
bash-4.2$ mpirun hostname
gpu01
gpu01
gpu01
gpu01
gpu01
gpu02
gpu01
gpu02
gpu01
gpu01
gpu01
gpu02
gpu01
...
```

Batch Approach

Another method to use `slurm` is `sbatch`. This is especially useful when you want to submit multiple jobs to queue.

We have prepared files for you to test.

First you can compile a demo mpi program with the following steps:

```
cd /pvfsmnt/$(whoami)      # enter your own dir
cp /pvfsmnt/test-mpi.cc .  # get the source
mpic++ test-mpi.cc -o test # compile
```

Then you get a template script:

```
cp /pvfsmnt/template.sh .
```

The script looks like:

```
#!/bin/bash
#SBATCH --account=csc4005
#SBATCH --partition=debug
#SBATCH --qos=normal
#SBATCH --ntasks=128
#SBATCH --nodes=4

echo "mainmode: " && /bin/hostname
mpirun /pvfsmnt/xxxxxxxxx/test
```

You can adjust the path, nodes, processes in it.

Then you can submit it with

```
sbatch template.sh
```

After the job is finished, you get a `slurm-XXX.out` file under the submitting directory.

For more options, see `sbatch --help` and for instance, you can add

```
#SBATCH --output=XXXX
```

to change your output.

Cancelling Jobs

To cancel a specific job, simply type

```
scancel <JOBID>
```

To cancel all jobs submitted by you, you can use:

```
scancel --user=$(whoami)
```

Statistics

To get the full work queue, you can use `squeue`:

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
  258     debug test.sba     root PD       0:00      2 (Resources)
  259     debug test.sba     root PD       0:00      2 (Priority)
  260     debug test.sba     root PD       0:00      2 (Priority)
  261     debug test.sba     root PD       0:00      2 (Priority)
  262     debug test.sba     root PD       0:00      2 (Priority)
  263     debug test.sba     root PD       0:00      2 (Priority)
  264     debug test.sba     root PD       0:00      2 (Priority)
  254     debug test.sba     root  R       0:06      2 gpu[01-02]
  255     debug test.sba     root  R       0:03      2 gpu[03-04]
  256     debug test.sba     root  R       0:03      2 gpu[05-06]
  257     debug test.sba     root  R       0:03      2 gpu[07-08]
```

You can see the current jobs running and waiting in the queue.

To see the status of all nodes, use `sinfo -a`:

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up      10:00      8  alloc gpu[01-08]
```

Normally, the `STATE` should be either `idle` or `alloc` indicating whether the node is used by others.

To see your submission history, type `sacct`,

```
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
208          interacti+      debug    csc4005         64     FAILED     63:0
208.0             orted               csc4005         64  COMPLETED      0:0
209          interacti+      debug    csc4005         64  COMPLETED      0:0
209.0             orted               csc4005         64  COMPLETED      0:0
209.1             orted               csc4005         64  COMPLETED      0:0
...
```