Speed: The ENCS Cluster

What It Comprises

- Sixteen, 32-core nodes, each with 512 GB of memory and approximately 1 TB of volatile-scratch disk space.
- Five AMD FirePro S7150 GPUs, with 8 GB of memory (compatible with the Direct X, OpenGL, OpenCL, and Vulkan APIs).

What It Is Ideal For

- Jobs that are too demanding of a desktop, but that are not worth the hassles associated with the provincial and national clusters.
- Single-core batch jobs; multithreaded jobs up to 32 cores (i.e., a single machine).
- Anything that can fit into a 500-GB memory space, and a scratch space of approximately 1 TB.
- CPU-based jobs.
- Non-CUDA GPU jobs.

Getting Started

To use the cluster you will need to be added to the LDAP group that governs access to speed-submit.encs.concordia.ca. Please submit your request to, rt-ex-hpc@encs.concordia.ca, detailing who you are, your username (e.g., what you would use to access login.encs.concordia.ca, for example), and the lab that you are associated with. That same username will then be given a scheduler account. Once that you can SSH to speed (an alias for speed-submit.encs.concordia.ca), you will need to source the scheduler file:

source /local/pkg/uge-8.6.3/root/default/common/settings.csh

You may consider adding the source request to your shell-startup environment (i.e., to your .tcshrc file). If sourcing has been successful, you have access to the scheduler commands. For example, if, 'qstat -f -u "*"', returns something non-error related, you are in business.

Job Submission Basics

Let's look at a basic job script, tcsh.sh (you can copy it from /home/n/nul-uge):

```
-----
#!/encs/bin/tcsh

#$ -N qsub-test
#$ -cwd
#$ -l h_vmem=1G

sleep 30
module load gurobi/8.1.0
module list
-----
```

This script sleeps on a node for 30 seconds, uses the module command to load the gurobi/8.1.0 environment, and then prints the list of loaded modules into a file. Concentrating on the first four lines, the first line is the shell declaration; the next three lines are submission options passed to the scheduler. The first, '-N', attaches a name to the job (otherwise it is called what the job script is called), the second, '-cwd', tells the scheduler to execute the job from the current working directory, and not to use the default of your home directory (potentially important for output-file placement), and the third, '-l h_vmem', requests and assigns a memory space to the job (this is an upper bound, and jobs that attempt to use more will be terminated). Note that this third option

is *not* optional (if you do not specify a memory space, submission of the job will fail). Also notice the syntax that denotes a scheduler option, the, '#$'.

The scheduler command, 'qsub', is used to submit (non-interactive) jobs. To submit this job: 'qsub ./tcsh.sh'. You will see, "Your job X ("qsub-test") has been submitted". The command, 'qstat', can be used to look at the status of the cluster: 'qstat -f -u "*"'. You will see something like this:

```
queuename                      qtype resv/used/tot. np_load  arch          states
---------------------------------------------------------------------------------
l.q@speed-05.encs.concordia.ca BIP   0/0/32         0.00     lx-amd64
      144 100.00000 qsub-test  nul-uge      r     12/03/2018 16:39:30         1
---------------------------------------------------------------------------------
l.q@speed-17.encs.concordia.ca BIP   0/0/32         0.00     lx-amd64
---------------------------------------------------------------------------------
l.q@speed-19.encs.concordia.ca BIP   0/0/32         0.00     lx-amd64
---------------------------------------------------------------------------------
l.q@speed-20.encs.concordia.ca BIP   0/0/32         0.00     lx-amd64
---------------------------------------------------------------------------------
l.q@speed-21.encs.concordia.ca BIP   0/0/32         0.00     lx-amd64
---------------------------------------------------------------------------------
l.q@speed-22.encs.concordia.ca BIP   0/0/32         0.00     lx-amd64
---------------------------------------------------------------------------------
l.q@speed-31.encs.concordia.ca BIP   0/0/32         0.00     lx-amd64
---------------------------------------------------------------------------------
l.q@speed-32.encs.concordia.ca BIP   0/0/32         0.00     lx-amd64
etc.
```

Remember that you only have 30 seconds before the job is essentially over, so if you do not see a similar output, either adjust the sleep time in the script, or execute the qstat statement more quickly. The qstat ouput listed above shows you that your job is

running on node speed-05, that it has a job number of 144, that it was started at 16:39:30 on 12/03/2018, and that it is a single-core job (the default).

Once the job finishes, there will be a new file in the directory that the job was started from, with the syntax of, "job name".o"job number", so in this example the file is, qsub-test.o144. This file represents the standard output (and error, if there is any) of the job in question. If you look at the contents of your newly created file, you will see that it contains the output of the, 'module list', command.

Congratulations on your first job!

Job Management

Here are useful job-management commands:
qstat -f -u "*": display cluster status for all users.
qstat -j [job-ID]: display job information for [job-ID] (said job may be actually running, or waiting in the queue).
qdel [job-ID]: delete job [job-ID].
qhold [job-ID]: hold queued job, [job-ID], from running.
qrls [job-ID]: release held job [job-ID].
qacct -j [job-ID]: get job stats. for completed job [job-ID]. maxvmem is one of the more useful stats.

Advanced qsub Options

In addition to the basic qsub options presented earlier, there are a few additional options that are generally useful:
-m bea: requests that the scheduler e-mail you when a job (b)egins; (e)nds; (a)borts. Mail is sent to the default address of, "username@encs.concordia.ca", unless a different address is supplied (see, '-M'). The report sent when a job ends includes job

runtime, as well as the maximum memory value hit (maxvmem).
-M email@domain.com: requests that the scheduler use this e-mail notification address, rather than the default (see, '-m').
-v variable[=value]: exports an environment variable that can be used by the script.
-l h_rt=[hour]:[min]:[sec]: sets a job runtime of HH:MM:SS. Note that if you give a single number, that represents *seconds*, not hours.
-hold_jid [job-ID]: run this job only when job [job-ID] finishes. Held jobs appear in the queue.
The many qsub options available are read with, 'man qsub'. Also note that qsub options can be specified during the job-submission command, and these *override* existing script options (if present). The syntax is, 'qsub [options] /PATHTOSCRIPT', but unlike in the script, the options are specified without the leading "#$" (e.g., qsub -N qsub-test -cwd -l h_vmem=1G ./tcsh.sh).

Requesting Multiple Cores (i.e., Multithreading Jobs)

For jobs that can take advantage of multiple machine cores, up to 32 cores (per job) can be requested in your script with:
#$ -pe smp [#cores]
Do not request more cores than you think will be useful, as larger-core jobs are more difficult to schedule. On the flip side, though, if you are going to be running a program that scales out to the maximum single-machine core count available, please (please) request 32 cores, to avoid node oversubscription (i.e., to avoid overloading the CPUs). Core count associated with a job appears under, "states", in the, 'qstat -f -u "*"', output.

Interactive Jobs

Job sessions can be interactive, instead of batch (script) based. Such sessions can be useful for testing and optimising code and resource requirements prior to batch submission. To request an interactive job session, use, 'qlogin [options]', similarly to a

qsub command-line job (e.g., qlogin -N qlogin-test -l h_vmem=1G). Note that the options that are available for 'qsub' are not necessarily available for 'qlogin', notably, '-cwd', and, '-v'.

Scheduler Environment Variables

The scheduler presents a number of environment variables that can be used in your jobs. Three of the more useful are TMPDIR, SGE_O_WORKDIR, and NSLOTS:
$TMPDIR=the path to the job's temporary space on the node. It *only* exists for the duration of the job, so if data in the temporary space are important, they absolutely need to be accessed before the job terminates.
$SGE_O_WORKDIR=the path to the job's working directory (likely a NFS-mounted path). If, '-cwd', was stipulated, that path is taken; if not, the path defaults to your home directory.
$NSLOTS=the number of cores requested for the job. This variable can be used in place of hardcoded thread-request declarations.
Here is a sample script, using all three:

```
 #!/encs/bin/tcsh

#$ -N envs
#$ -cwd
#$ -pe smp 8
#$ -l h_vmem=32G

cd $TMPDIR
mkdir input
rsync -av $SGE_O_WORKDIR/references/ input/
mkdir results
```

STAR --inFiles $TMPDIR/input --parallel $NSLOTS --outFiles $TMPDIR/results

rsync -av $TMPDIR/results/ $SGE_O_WORKDIR/processed/

SSH Keys For MPI

Some programs effect their parallel processing via MPI (which is a communication protocol). An example of such software is Fluent. MPI needs to have “passwordless login” set up, which means SSH keys. In your NFS-mounted home directory:

- 'cd .ssh'
- 'ssh-keygen -t rsa' (default location; blank passphrase)
- 'cat id_rsa.pub >> authorized_keys' (if the authorized_keys file already exists) *OR* 'cat id_rsa.pub > authorized_keys' (if does not)
- Set file permissions of “authorized_keys” to 600; of your NFS-mounted home to 700 (note that you likely will not have to do anything here, as most people will have those permissions by default).

Example Job Script: Fluent

#!/encs/bin/tcsh

#$ -N flu10000
#$ -cwd
#$ -m bea
#$ -pe smp 32
#$ -l h_vmem=160G

module load ansys/19.0/default

cd $TMPDIR

fluent 3ddp -g -i $SGE_O_WORKDIR/fluentdata/info.jou -sgepe smp > call.txt

rsync -av $TMPDIR/ $SGE_O_WORKDIR/fluentparallel/

This job script runs Fluent in parallel over 32 cores. Of note, I have requested e-mail notifications (‘-m’), am defining the parallel environment for, ‘fluent’, with, ‘-sgepe smp’ (very important), and am setting $TMPDIR as the in-job location for the “moment-rfile.out” file (in-job, because the last line of the script copies everything from $TMPDIR to a directory in my NFS-mounted home). Job progress can be monitored by examining the standard-out file (e.g., flu10000.o249), and/or by examining the “moment” file in /disk/nobackup/<yourjob> (hint: it starts with your job-ID) on the node running the job. Caveat: take care with journal-file file paths.

Java Jobs

Jobs that call java have a memory overhead, which needs to be taken into account when assigning a value to h_vmem. Even the most basic java call, 'java -Xmx1G -version', will need to have, '-l h_vmem=5G', with the 4-GB difference representing the memory overhead. Note that this memory overhead grows proportionally with the value of -Xmx. To give you an idea, when -Xmx has a value of 100G, h_vmem has to be at least 106G; for 200G, at least 211G; for 300G, at least 314G.

Scheduling On The GPU Nodes

There are two GPUs in both speed-05 and speed-17, and one in speed-19. Their availability is seen with, 'qstat -F g' (note the capital):

queuename qtype resv/used/tot. load_avg arch states

```
l.q@speed-05.encs.concordia.ca BIP   0/0/32         0.01     lx-amd64
        hc:gpu=2
---------------------------------------------------------------------------
l.q@speed-17.encs.concordia.ca BIP   0/0/32         0.03     lx-amd64
        hc:gpu=2
---------------------------------------------------------------------------
l.q@speed-19.encs.concordia.ca BIP   0/0/32         0.01     lx-amd64
        hc:gpu=1
---------------------------------------------------------------------------
l.q@speed-20.encs.concordia.ca BIP   0/0/32         0.01     lx-amd64
---------------------------------------------------------------------------
etc.
```

This status demonstrates that all five are available (i.e., have not been requested as resources). To specifically request a GPU node, add, '-l g=[#GPUs]', to your qsub (statement/script) or qlogin (statement) request. For example, 'qsub -l h_vmem=1G -l g=1 ./count.sh'. You will see that this job has been assigned to one of the GPU nodes:

```
queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------
l.q@speed-05.encs.concordia.ca BIP   0/0/32         0.01     lx-amd64
        hc:gpu=2
---------------------------------------------------------------------------
l.q@speed-17.encs.concordia.ca BIP   0/0/32         0.01     lx-amd64
        hc:gpu=2
---------------------------------------------------------------------------
l.q@speed-19.encs.concordia.ca BIP   0/1/32         0.04     lx-amd64
        hc:gpu=0 (haff=1.000000)
      538 100.00000 count.sh   sbunnell     r     03/07/2019 02:39:39     1
```

And that there are no more GPUs available on that node (hc:gpu=0). Note that no

more than two GPUs can be requested for any one job.

Important Limitations

As a new user, you are limited to 32 cores (a restriction that is removed for responsible users).

Jobs are assigned a maximum runtime of one week, unless otherwise specified.

Home directories (data directories) are served over NFS. NFS is great for acute activity, but crappy for chronic activity. Any data that a job will read more than once should be copied at the start to the scratch disk of a compute node using $TMPDIR (and, perhaps, $SGE_O_WORKDIR), any intermediary job data should be produced in $TMPDIR, and once a job is near to finishing, those data should be copied to your NFS-mounted home (or other NFS-mounted space) from $TMPDIR (to, perhaps, $SGE_O_WORKDIR). In other words, IO-intensive operations should be effected locally whenever possible, saving network activity for the start and end of jobs.

Your current resource allocation is based upon past usage, which is an amalgamation of approximately one week's worth of past wallclock (i.e., time spent on the node(s)) and cpu activity (on the node(s)).

Jobs should NEVER be run outside of the province of the scheduler. Repeat offenders risk loss of cluster access.

Tips/Tricks

- files/scripts must have Linux line breaks in them (not Windows ones).
- use rsync, not scp, when moving data around.
- if you are going to move many many files between NFS-mounted storage and the

cluster, tar everything up first.
- If you intend to use a different shell (e.g., Bash), you will need to source a different scheduler file, and will need to change the shell declaration in your script(s).
- The load displayed in qstat by default is np_load, which is load/#cores. That means that a load of, "1", which represents a fully active core, is displayed as 0.03 on the node in question, as there are 32 cores on a node. To display load "as is" (such that a node with a fully active core displays a load of approximately 1.00), add the following to your .tcshrc file: setenv SGE_LOAD_AVG load_avg
- Try to request resources that closely match what your job will use: requesting many more cores or much more memory than will be needed makes a job more difficult to schedule when resources are scarce.
- e-mail, sbunnell@encs.concordia.ca, with any concerns/questions.