Using the computational resources at the GACRC

An introduction to zcluster

Shan-Ho Tsai

Georgia Advanced Computing Resource Center (GACRC)
University of Georgia

Talk given at the CSP lunch seminar - 09/02/2014
What is GACRC?

- Georgia Advanced Computing Resource Center (GACRC)
- Formerly called Research Computing Center (RCC)
- Provides computing hardware and network infrastructure in support of high-performance computing at UGA.
- Collaboration between the Office of the Vice President for Research (OVPR) and the Office of the Vice President for Information Technology (OVPIT).
- Guided by a faculty advisory committee (GACRC-AC).

GACRC’s Website: http://gacrc.uga.edu

GACRC’s Linux cluster: zcluster
Connecting and transferring files to zcluster

- Use SSH to login to zcluster.rcc.uga.edu (login node).
  - Example: ssh -X username@zcluster.rcc.uga.edu
- Use SCP to transfer files from/to your local machines to copy.rcc.uga.edu (copy nodes).
- For faster file transfer, use FTPS with SSL explicit encryption (using e.g. FileZilla)

For more information on how to transfer files to/from zcluster and between filesystems on zcluster, please see https://wiki.gacrc.uga.edu/wiki/Transferring_Files
## Hardware resources

### CPU resources on zcluster

<table>
<thead>
<tr>
<th>Nodes</th>
<th>cores/node</th>
<th>CPU type</th>
<th>RAM/node</th>
<th>Total cores</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>12</td>
<td>Intel Xeon</td>
<td>48GB</td>
<td>540</td>
<td>Regular</td>
</tr>
<tr>
<td>150</td>
<td>8</td>
<td>Intel Xeon</td>
<td>16GB</td>
<td>1200</td>
<td>Regular</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>Intel Xeon</td>
<td>16GB</td>
<td>256</td>
<td>Infiniband</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>Intel Xeon</td>
<td>192GB</td>
<td>32</td>
<td>Medium memory</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>Intel Xeon</td>
<td>256GB</td>
<td>120</td>
<td>Medium memory</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>Intel Xeon</td>
<td>512GB</td>
<td>64</td>
<td>Big memory</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>AMD Opteron</td>
<td>64GB</td>
<td>192</td>
<td>Many cores</td>
</tr>
<tr>
<td>2</td>
<td>48</td>
<td>AMD Opteron</td>
<td>132GB</td>
<td>96</td>
<td>Interactive</td>
</tr>
</tbody>
</table>

Total peak performance: 23 Tflops
GPU resources on zcluster

The zcluster has the following NVIDIA GPU resources:

- **Tesla S1070**, 4 GPU cards, each with 30 multiprocessors (MP) and 8 cores/MP = 960 cores
- **Tesla M2070 (Fermi)**, 12 GPU cards, each with 14 multiprocessors and 32 cores/MP = 5376 cores
- **Tesla C2075 (Fermi)**, 1 GPU card = 448 cores
- **Tesla K20Xm (Kepler)**, 32 GPU cards, each with 14 multiprocessors and 192 cores/MP = 86,016 cores
Data Storage at GACRC

Panasas primary storage: Total usable space: 150 TB

- **Home directory**: `/home/groupname/username`
  Example: `/home/simulphy/jsmith`
  - One volume per group, use `quota_rep` to see group usage.
  - Visible to all nodes.
  - Has snapshots in each directory, e.g. `/home/simulphy/jsmith/test/.snapshot`

```
zcluster> cd test
zcluster> cd .snapshot
zcluster> ls
2014.08.31.00.00.01.weekly/ 2014.09.01.01.00.01.daily/
2014.08.30.01.00.01.daily/ 2014.09.02.01.00.01.daily/
2014.08.31.01.00.01.daily/
zcluster> cd 2014.09.02.01.00.01.daily
zcluster> ls
```
There are 2 types of scratch space - Not backed up!

- **Local scratch area**: `/lscratch/username`. On local disk of each node (not visible to other nodes).

- **Ephemeral scratch**:
  - Create a directory with `make_escratch` on zcluster.
  - Visible to all nodes in the systems.
  - A user can create one escratch dir per day.
  - 4TB user quota for all his/her escratch dirs combined.
  - Files deleted after 37 days.
Data Storage at GACRC

Archival storage (slow):

- Mounted as `/oflow/groupname` on login nodes
- Not mounted on compute nodes: cannot run jobs from it
- Can use copy nodes and `copyq` to transfer files.

```
zcluster> ssh copy
copy1> mv test/file /oflow/groupname/username
```

Or submit a script to the queue:

**Sample script: subcopy.sh**

```
#!/bin/bash
cd ${HOME}/test
mv file /oflow/groupname/username
```

```
zcluster> qsub -q copyq subcopy.sh
```
Software installed on zcluster

- Compilers: Intel, Portland Group and GNU C/C++ and Fortran; LLVM-Clang, Java
  https://wiki.gacrc.uga.edu/wiki/CodeCompilation_on_zcluster
- Debuggers: GNU gdb, PGI pgdbg (with MPI support, GUI)
- Profilers: GNU gprof, PGI pgprof
- Math libraries: BLAS, LAPACK, FFTW, GSL, etc
- MPI libraries: MPICH, MPICH2, OpenMPI
  https://wiki.gacrc.uga.edu/wiki/MPI
- perl, python, awk, sed
- matlab, maple, mathematica, R
- Queueing system: Univa Grid Engine

For a list of applications installed see:
https://wiki.gacrc.uga.edu/wiki/Software
# Compilers on zcluster

<table>
<thead>
<tr>
<th></th>
<th>PGI</th>
<th>PGI/MPICH2</th>
<th>Intel</th>
<th>GNU 4.1.2</th>
<th>GNU 4.4.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortran77</td>
<td>pgf77</td>
<td>mpi77</td>
<td>ifort</td>
<td>gfortran</td>
<td>gfortran44</td>
</tr>
<tr>
<td>Fortran90</td>
<td>pgf90</td>
<td>mpi90</td>
<td>ifort</td>
<td>gcc</td>
<td>gcc44</td>
</tr>
<tr>
<td>C</td>
<td>pgcc</td>
<td>mpicc</td>
<td>icc</td>
<td>g++</td>
<td>g++44</td>
</tr>
<tr>
<td>C++</td>
<td>pgCC</td>
<td>mpiCC</td>
<td>icc</td>
<td>g++</td>
<td>g++44</td>
</tr>
</tbody>
</table>

### Notes about PGI compilers:
- Support CUDA, including CUDA/Fortran
- Support GPU accelerator using OpenACC directives
- To check default version: `pgcc -V`

### Also available in non-default locations:
- GNU 4.7.1: `/usr/local/gcc/4.7.1/bin`
- NVIDIA CUDA: `/usr/local/cuda`
Software installed

MPICH libraries on zcluster

- PGI/MPICH2 1.4.1p1: default for mpicc, mpif90, mpirun, etc
- MPICH2: in /usr/local/mpich2/3.0.4. E.g.
  - GNU 4.1.2: /usr/local/mpich2/3.0.4/gcc412/bin
  - GNU 4.4.7: /usr/local/mpich2/3.0.4/gcc447/bin
  - GNU 4.5.3: /usr/local/mpich2/3.0.4/gcc453/bin
  - PGI 12.10: /usr/local/mpich2/3.0.4/pgi1210/bin
  - Intel 14.0: /usr/local/mpich2/3.0.4/intel140/bin
- OpenMPI: several compilations in /usr/local/openmpi
- PGI/MVAPICH2: /usr/local/mvapich2/1.8/pgi124/bin
- GNU/MVAPICH2: /usr/local/mvapich2/1.8/gcc444/bin
Running interactive jobs on zcluster

- No jobs should be run on the zcluster login node!
- To run an interactive job: use the qlogin command:

```bash
zcluster> qlogin
Your job 1391816 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 1391816 has been successfully scheduled.
...
compute-14-7.local>
```

- This will open a session on an interactive node.
- Current maximum runtime is 12 hours.
- Please remember to exit the session when you are done.
- Detailed example of how to run an interactive parallel job is at
  https://wiki.gacrc.uga.edu/wiki/Running_Jobs_on_zcluster
# Main batch queues on zcluster

<table>
<thead>
<tr>
<th>Nodes</th>
<th>cores/node</th>
<th>CPU type</th>
<th>RAM/node</th>
<th>Total cores</th>
<th>Queue name</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>12</td>
<td>Intel Xeon</td>
<td>48GB</td>
<td>540</td>
<td>rcc-30d</td>
</tr>
<tr>
<td>150</td>
<td>8</td>
<td>Intel Xeon</td>
<td>16GB</td>
<td>1200</td>
<td>rcc-30d</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>Intel Xeon</td>
<td>16GB</td>
<td>256</td>
<td>rcc-ib-30d</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>Intel Xeon</td>
<td>192GB</td>
<td>32</td>
<td>rcc-m128-30d</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>Intel Xeon</td>
<td>256GB</td>
<td>120</td>
<td>rcc-m128-30d</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>Intel Xeon</td>
<td>512GB</td>
<td>64</td>
<td>rcc-m512-30d</td>
</tr>
<tr>
<td>2</td>
<td>48</td>
<td>AMD Opteron</td>
<td>132GB</td>
<td>96</td>
<td>interq</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>AMD Opteron</td>
<td>64GB</td>
<td>192</td>
<td>rcc-mc-30d</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>4 GPU S1070 cards</td>
<td>48GB</td>
<td></td>
<td>rcc-sgpu-30d</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>10 GPU M2070 cards</td>
<td>48GB</td>
<td></td>
<td>rcc-mgpu-30d</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>32 GPU K20X cards</td>
<td>96GB</td>
<td></td>
<td>rcc-kgpu-30d</td>
</tr>
</tbody>
</table>
Running serial batch jobs on zcluster

**Step 1:** Create a job submission script file. Example:

```bash
#!/bin/bash
cd ${HOME}/testdir
time ./prog.x
```

Note: in the last line of the script, the `time` command was added to record the amount of time used by the application (prog.x).
Running jobs

**Step 2: Submit it to the queue**

```
zcluster> qsub -q rcc-30d sub.sh
Your job 132 ("sub.sh") has been submitted
```

Options for `qsub`:

- `-q` : specify a queuename, e.g. `rcc-30d`.
- `-m e` : receive email when job ends
- `-l mem_total=20g` : to ensure that job goes to a node with 48GB RAM (for `rcc-30d`)
**Step 3:** To check the status of your jobs

```
zcluster> qstat
```

<table>
<thead>
<tr>
<th>job-ID</th>
<th>name</th>
<th>user</th>
<th>state</th>
<th>submit/start at</th>
<th>queue</th>
<th>slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
<td>sub.sh</td>
<td>amy</td>
<td>r</td>
<td>08/20/2012 1:32:12</td>
<td><a href="mailto:rcc-30d@compute-10-5.local">rcc-30d@compute-10-5.local</a></td>
<td>32</td>
</tr>
<tr>
<td>132</td>
<td>sub.sh</td>
<td>amy</td>
<td>qw</td>
<td>08/22/2012 1:34:23</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>130</td>
<td>sub.sh</td>
<td>amy</td>
<td>Eqw</td>
<td>08/20/2012 1:10:23</td>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>

Common states for jobs:

- **r**: job is running, job start time shown
- **qw**: job is pending, job submission time is shown
- **Eqw**: job is in error state, needs attention

To view more details of a job

```
zcluster> qsj jobid
zcluster> qstat -j jobid
```
Common UGE commands:

- **qsub**: Submit a job to the queue
- **qdel**: Cancel a queued or running job
  - `qdel -u username`: deletes all your jobs
  - `qdel 10407`: deletes the job with JOBID 10407
- **qstat**: Check the status of all queued and running jobs
  - `qstat`: shows your jobs
  - `qstat -u "*"`: shows all the jobs in the pool
  - `qsj 10407`: gives detailed information about the job with JOBID 10407
  - `qstat -g t`: lists all the nodes used by your jobs
- **qacct**: To view information on job that ended
  - `qacct -j 10407`: gives info about job 10407 (cputime, memory used, etc)
Running a shared memory application

Example: Run prog.x with 4 threads
OpenMP Code: `pgcc -mp hello_threads.c -o prog.x`

Example of shell script: subthreads.sh

```bash
#!/bin/bash
cd ${HOME}/testopenmp
export OMP_NUM_THREADS=4
./prog.x
```

`zcluster> qsub -q rcc-30d -pe thread 4 subthreads.sh`
Compiling and Running MPI C program

Basic C code compilation command:
- Serial Code: `pgcc prog.c -o prog.x`
- MPI Code: `mpicc mpiprog.c -o mpi.x`

Basic command to run the executable:
- Serial Code: `./prog.x`
- MPI Code: `mpirun -np N -f hostlist mpi.x`
  where $N =$ number of processes, and hostlist is a text file with the name of the machines to be used (need $N$ entries).

Sample hostlist file

<table>
<thead>
<tr>
<th>Sample hostlist file</th>
</tr>
</thead>
<tbody>
<tr>
<td>c14-7</td>
</tr>
<tr>
<td>c14-7</td>
</tr>
<tr>
<td>c14-7</td>
</tr>
</tbody>
</table>
Running a batch MPI job on zcluster

Basic C code compilation command:

- **Serial Code:** `pgcc prog.c -o prog.x`
- **MPI Code:** `mpicc mpiprog.c -o mpi.x`

**Sample script submpi.sh**

```bash
#!/bin/bash
cd ${HOME}/testmpi
mpirun -np $NSLOTS ./mpi.x
```

**Note:** When running batch jobs using MPICH2 or OpenMPI, there is no need to specify a list of hosts.

Submit it to the batch queue with (e.g using 4 CPUs):

```
zcluser> qsub -q rcc-30d -pe mpi 4 submpi.sh
```
To compile a CUDA/C code on zcluster

For bash:
In your .bash_profile (or in your .profile) file add the line

```
. $HOME/.bashrc
```

Create a file called .bashrc in your home dir. In it add:

```
export PATH=/usr/local/cuda/5.0.35/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/5.0.35/lib64:$LD_LIBRARY_PATH
```

To compile a CUDA/C code called prog.cu

```
zcluster> nvcc -arch sm_13 prog.cu -o prog.x
```
To run a GPU job on zcluster

Sample script file: subgpu.sh

```bash
#!/bin/bash
cd $HOME/subdir
./prog.x
```

Submit it to the queue with:

```bash
zcluster> qsub -q rcc-kgpu-30d -l cuda=1 subgpu.sh
```
How do I monitor memory usage of running jobs?

Use the command

```
zcluster$ qsj jobid
```

The maximum virtual memory used so far by the job is given by the parameter `maxvmem`. Example:

```
zcluster$ qsj 12345
===========================================
submission_time: Fri Feb 28 17:24:25 2014
owner: jsmith
cwd: /lustre1/escratch1/jsmith_Jun_23
hard_queue_list: rcc-30d
script_file: sub.sh
usage 1: cpu=66:16:05:15, mem=50089783.45171 GBs, io=0.0 1083, vmem=11.648G, maxvmem=11.648G
```
How do I view memory used by job that ended?

Use the command

```
zcluster$ qacct -j jobid
```

The maximum virtual memory used by the job is given by the parameter `maxvmem`. Example:

```
zcluster$ qacct -j 12345
hostname compute-10-5.local
owner jsmith
jobname job1.sh
start_time Fri Feb 28 17:17:12 2014
end_time Fri Feb 28 21:05:40 2014
ru_wallclock 13708
cpu 27976.713
maxvmem 1.966G
```
How to get support?

The preferred method to ask questions and report problems is to use the online form available at:

http://help.gacrc.uga.edu

GACRC communication resources:

- **Web Site** - general overview: http://gacrc.uga.edu/
- **Wiki** - software docs, how-to’s: https://wiki.gacrc.uga.edu/
- **Blog** - announcements: https://blog.gacrc.uga.edu/
- **Forums** - user discussion area: https://forums.gacrc.uga.edu/
- **Email** - notifications sent to all users.