A Quick Guide to Getting Started on ISI Vista Cluster

Mozhdeh Gheini (gheini@isi.edu)

What is Vista Cluster?

Vista Cluster is a grid computing system at USC ISI that provides high computational power to its users. It uses Portable Batch System (PBS) for job submission, job monitoring, and job management. We will get to it as we go on.

There are two types of machines on Vista:

- Submission host (qsubmit.isi.edu)
- Execution hosts

The submission host is where you can directly `ssh` into and use to submit jobs to the queue, and the execution hosts are used to execute them. One important thing to remember is that resources on qsubmit are limited and shared among all users. You should never run big (or even little) jobs on qsubmit; you have to use it to submit them to execution hosts.

Another thing to note is that your home directory storage is limited to 50GB and shouldn't be used as data file storage. In order to ask where you can store your files, please contact your supervisor.

Enough General Knowledge; Onto the More Technical Stuff

Before anything please add these three lines to your `~/.bashrc` file and `source` it: `$ source ~/.bashrc`.

```
source /nas/uge/sge-root/default/common/settings.sh
export DRMAA_LIBRARY_PATH=${SGE_ROOT}/lib/lx-amd64/libdrmaa.so.1.0
export PATH=${PATH}:/nfs/isicvlnas01/share/qstat-pretty/
```

Now you can use the commands referenced later in the tutorial. As mentioned previously, the cluster uses PBS as its job scheduler. After logging into qsubmit, you can start running your commands to submit jobs. Before requesting resources on any of the execution hosts, you have to have decided on three things:

1. The mode you want to work in
   You can either work in the interactive mode or in the batch mode. To work in the interactive mode, you can request a remote shell using the `qrsh` command. If you want to work in the batch mode, you can submit your job using the `qsub` command. We'll see examples of both a bit later.
2. The project you're part of
   The project you belong to determines the priority of your job. All users have access to `other` project which has the lowest priority, but you can always use it. To find out what projects you're part of, please refer to your supervisor. You can let the scheduler know the project by including the `-P` option in your command. For example, if you want to request an interactive remote shell (look above) and you belong to the project `pname`, you can type in:

```
$ qrsh -P pname
```

   Or if you want to submit your script `job.sh` as a part of that project, you can type in:

```
$ qsub -P pname job.sh
```

3. The resources that you'll need
   By default, your jobs are allocated 1 CPU core, 1GB of memory, 1 hour of runtime, and 0 GPUs. If you need more resources, you have to let the scheduler know by including the right options and values in your command.

   - More CPU cores: you can ask for more cores by using the `-pe mt` option. In case you're curious, pe stands for 'parallel environment'. If you need `C` CPU cores, you have to include `-pe mt C` in your command.

   Memory, time, and GPUs all can be managed using the `-l` option.

   - More memory: to ask for more memory, for example, 10GB, you want to include `-l h_vmem=10G` in your command. Note that this is actually memory-per-core; so if you've requested `C` cores, the memory that you request will be multiplied by `C`.
   - More time: you can ask for more by including `-l h_rt=hh:mm:ss` in your command where `hh` is the number of hours, `mm` is the number of minutes, and `ss` is the number of seconds. The maximum runtime allowed for all jobs is 24 hours. So to request the maximum, you can use `-l h_rt=24:00:00`. However, note that shorter runtime may help your job execute sooner than the jobs that have longer running times.
   - More GPUs: to ask for more GPUs, simply use `-l gpu=N` where `N` is the number of the GPUs that you need.

Other Options You Might Need

- `-cwd`: This option makes the job to be executed from the current working directory. So any `.` used in your script will point to the current directory.
- `-N`: When you submit a script, e.g. `job.sh`, your job automatically outputs STDOUT and STDERR to `job.sh.oXXXX` and `job.sh.e.XXXX` respectively where `XXXX` is your job ID. By using `-N` you can give a name to your job, e.g. `jname` and make the outputs to be written to `jname.oXXXX` and `jname.eXXXX`.
- `-l 'h=HOSTNAME'`: Sometimes you might need to work with or avoid a specific host. For example, let's say you want to work on `vista07`; you can simply include `-l 'h=vista07'` in your command. To avoid a host, use an `!` before the `HOSTNAME`.

To put all we saw together, here's an example with everything that we discussed:

```
$ qsub -N 'my_job' -P other -cwd -pe mt 2
↪   -l h_vmem=8G,gpu=1,h_rt=24:00:00 -l 'h=!evil_host'
↪   my_script.sh
```

Checking the Status of Your Job

To view the status of your running and queued jobs, simply run `qstat` or `pstat` (for a prettier output).

Things to Remember When Writing Your Scripts

Note that your job does not share your environment variables by default. So remember to always `source` your `~/.bashrc` at the beginning of your scripts.

FAQs

- Can I access the cluster from outside of ISI network?
  Yes, but first you have to establish a VPN connection. For more information, refer to ISI action's page on it.
- Where can I ask any other questions that I might have?
  Have someone invite you to the vista-cluster-info channel on Slack. As well as receiving general cluster-related announcements, you can ask your questions and get quick answers there.

Other Useful Tools You Might Want to Check Out

- Conda: Even if you just remotely think about doing anything in Python (or many other languages, Conda is language-agnostic) on a remote server, you need Conda: an open-source environment and package management system. Start from here.
- tmux: If you're familiar with GNU Screen, tmux is like that. It helps you manage your windows and sessions on a remote machine. To get started on tmux, I suggest the following links:
  - Basic tmux Tutorial, Part 1
  - Basic tmux Tutorial, Part 2
  - A tmux Crash Course