H Open Cluster

Jay Nietling, UNLV Physics & Astronomy

October 21, 2016

There is a new "old" compute cluster available for use by anyone with a Physics & Astronomy Unix login. The H cluster has 18 nodes with multi-core AMD Opteron processors. Computational jobs are run on the cluster with the SLURM resource manager. A distributed, replicated file store is provided by GlusterFS under the `/data` directory. In the near future a system with an Intel Phi co-processor will be available. Additional "old" servers will be added over time.

H Cluster Hardware

The H Cluster[1] is not state of the art. However, it is adequate for many tasks and can be used to experiment with various styles of multiprogramming. Each node has 2 dual core AMD Opteron 265 processors at 1.8Ghz and 16 gigabytes of RAM. In total the cluster has a bit less than a terabyte of shared disk space for running applications under the `/data` directory.

[1] Hostnames are h-oa.physics.unlv.edu, h-ob..., h-[1-16]... . Jobs can be started from any node.

SLURM (Simple Linux Utility for Resource Management)

Slurm is a light weight system for managing resources (cpu, memory, disk) among possibly many cooperating processes. If you want to use the cluster *you must use SLURM* to run your programs. There is a small learning curve to submitting simple jobs with SLURM. The overhead of submitting jobs with SLURM is not costly. After using a job queuing system like SLURM you'll find that it helps you get much more work done without checking long running processes or log files. It also allocates resources more efficiently than programmers of varied experience are able to.

Compute Job I/O Use: `/data`

GlusterFS is an open source, distributed file system capable of scaling to several petabytes (actually, 72 brontobytes!) and handling thousands of clients. GlusterFS clusters together storage building blocks over Infiniband RDMA or TCP/IP interconnect, aggregating disk and memory resources and managing data in a single global namespace.[2] The H cluster has slow interconnects and old drives so blazing performance cannot be expected. However, performance should be reasonably close to that of a NFS server. `/data` is a replicated file store that spans the cluster. Each file created is stored on two nodes to add resiliency. Whatever node a process runs on it can access all files in `/data` like a central shared NFS server.

[2] From the GlusterFS web site.

The convention for using `/data` is to create a directory for yourself named your username, e.g. `/data/jay`. For best performance, programs and data should then be copied into that directory (or sub-directory). Finally a job to run the program with associated data should be submitted through SLURM.

An example interaction with the Unix shell should get you started. Figure 1 is a short C program to printout the name of whatever host the program runs on. Figure 2 is a shell script with special comments that direct how a SLURM job is executed. Line 10 sets the working directory of the script before it runs. Lines 11 and 12 set the files for output and error messages. Line 16 sets which partition within SLURM to use. This is may be thought of as a job queue. We have *debug* and *open* partitions on the cluster. Lines 17-20 are resource allocation requests. We want to run the little `hello-hostname` program on all 18 nodes. It should surely finish in less than 5 minutes. Line 17 in the Unix session below shows the job being submitted and the output being checked on Line 31.

```
1  -bash-3.2$ ssh h-5
2  Last login: Wed Oct  8 11:39:19 2014 from snooks.physics.unlv.edu
3  -bash-3.2$ mkdir -p /data/jay/hello-world
4  -bash-3.2$ . /share/rc/slurm # setup slurm environment
5  -bash-3.2$ cp hh.sh hello-hostname.c /data/jay/hello-world/
6  -bash-3.2$ cd /data/jay/hello-world/
7  -bash-3.2$ chmod 775 hh.sh # make sure hh.sh is executable. man chmod if you don't understand
8  -bash-3.2$ gcc -o hello-hostname hello-hostname.c
9  -bash-3.2$ ./hello-hostname
10 h-5
11 -bash-3.2$ sinfo # get info on the SLURM job queues.  all quiet. good.
12 PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
13 debug*       up    1:00:00      2   idle h-0a,h-0b
14 open         up   infinite     18   idle h-[1-16],h-0a,h-0b
15 -bash-3.2$ ls
16 hello-hostname hello-hostname.c hh.sh
17 -bash-3.2$ sbatch hh.sh # submit the job to SLURM
18 Submitted batch job 73
19 -bash-3.2$ sinfo # some nodes still busy.
20 PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
21 debug*       up    1:00:00      2   idle h-0a,h-0b
22 open         up   infinite      7   comp h-[1,3-5,7-9]
23 open         up   infinite     11   idle h-[2,6,10-16],h-0a,h-0b
24 -bash-3.2$ sinfo # almost done.
25 PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
26 debug*       up    1:00:00      2   idle h-0a,h-0b
27 open         up   infinite      1   comp h-1
28 open         up   infinite     17   idle h-[2-16],h-0a,h-0b
29 -bash-3.2$ ls # two new files.  look at hh.sh for why.
30 hello-hostname        hello-hostname.c  hello.err  hello.out        hh.sh
31 -bash-3.2$ cat hello.out
32 h-1
33 h-5
34 h-11
35 h-12
36 .
37 .
38 .
39 -bash-3.2$ cat hello.out | wc -l
40 18
41 -bash-3.2$ cat hello.err
42 -bash-3.2$
```

```
1  -bash-3.2$ cat hello-hostname.c
2  /*
3   * print out my hostname
4   */
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  int main(void) {
9
10     char hostname[1024];
11     gethostname(hostname, 1024);
12
13     puts(hostname);
14
15     return EXIT_SUCCESS;
16 }
```

Figure 1: A C language program to print out my hostname.

```
1  -bash-3.2$ cat hh.sh
2  #!/bin/sh
3  #
4  # hh.sh:
5  #
6  # a little shell script to
7  # illustrate SLURM features
8  #
9  #SBATCH --job-name=hello-host
10 #SBATCH --workdir=/data/jay/hello-world
11 #SBATCH --output=hello.out
12 #SBATCH --error=hello.err
13 #
14 ## resource requirements:
15 #
16 #SBATCH --partition=open
17 #SBATCH --ntasks=18
18 #SBATCH --nodes=18
19 #SBATCH --time=00:05:00
20 #SBATCH --mem-per-cpu=200mb
21
22 srun hello-hostname
23 srun sleep 60
24
25 exit 0
```

Figure 2: A SLURM job submission shell script.