Job Submission on the Nanolab Cluster

Derek Stewart
stewart@cnf.cornell.edu

The job submission system, slurm, has now been set up on the CNF cluster. This queuing system should provide an easy and transparent way to streamline calculations on the cluster and make cluster-use more efficient. After examining several job submission systems, I decided to go with slurm because it offers several user friendly features:

(1) It has the ability to run parallel calculations using LAM-MPI by allocating a set of nodes specifically for that task.
(2) Users running serial jobs with minimal input/output can start them directly from the command line without resorting to batch files.
(3) Slurm has the ability to handle batch jobs as well, allowing users to still take advantage of the scratch space available on the Nanolab nodes.

For complete information on SLURM, I encourage you to check out the program's website:

http://www.llnl.gov/linux/slurm

Below I have provided a brief introduction to some of the commonly used slurm commands.

Common Commands to use with Slurm

Cluster activity (squeue)

When you get ready to run a calculation, you can check the activity on the cluster by running **squeue**. This command will provide a list of the currently running jobs, how long they have been running, and which nodes they are running on. An example is shown below:

```
[der12@nanolab der12]$ squeue
 JOBID PARTITION     NAME     USER  ST    TIME  NODES NODELIST(REASON)
    70      nlab nt_light    der12   R    3:22:11      1 b29
    71      nlab nt_light    der12   R    3:21:19      1 b30
    72      nlab nt_light    der12   R    3:20:48      1 b31
    74      nlab nt_light    der12   R    2:45:51      1 b33
    76      nlab nt_light    der12   R    2:44:54      1 b34
    77      nlab nt_light    der12   R    2:34:47      1 b32
    82      nlab node_run    der12   R    1:41:24      1 b35
    83      nlab node_run    der12   R    1:36:31      1 b36
    84      nlab node_run    der12   R    1:31:48      1 b37
    85      nlab nt_light    der12   R    1:02:32      1 c12
    86      nlab nt_light    der12   R    1:01:43      1 c13
    87      nlab nt_light    der12   R    1:00:50      1 c14
    89      nlab nt_light    der12   R      40:52      1 c15
    90      nlab nt_light    der12   R      39:39      1 c16
    91      nlab nt_light    der12   R      38:40      1 b6
```

Each calculation is given a JOBID. This can be used to cancel the job as well if necessary. The PARTITION field describes the available nodes. In this case, we have all nodes set up in the partition nlab. The field NAME gives the name of the program being used for the calculation. The NODES field shows which node each calculation is running on.

How many nodes are free? (sinfo)

You can also get an idea of how many nodes are free to run calculations by typing **sinfo**. This command shows which nodes are down, allocated to current jobs, or idle.

```
[der12@nanolab der12]$ sinfo
PARTITION AVAIL  TIMELIMIT NODES  STATE NODELIST
nlab*        up   infinite     1  down* nanolab
nlab*        up   infinite    15  alloc b[6,29-37],c[12-16]
nlab*        up   infinite    27   idle b[3-5,7-16,18-20,38-42],c[10-
11,17-20]
```

Starting a calculation (srun)

In this case, we see that 27 nodes are ready to run calculations. You can start a calculation directly from the prompt by using **srun**. This command submits jobs to the job submission system and it can also be used to start the same command on multiple nodes. We will start with a simple example.

```
[der12@nanolab der12]$ srun -N4 /bin/hostname &
b41
b38
b39
b40
[der12@nanolab der12]$
```

In the example above, we use srun to start the command *hostname* on 4 nodes in the cluster. The option –N4 tells slurm to run the job on four nodes of its choice. The command hostname prints out the hostname of each node that was used.

With many calculations it is important to pipe data in (<) from an input file and pipe date out (>) to an output file. The program may also have command line options as well. *$program [options] < input.dat > output.dat*.

One of the nice features of srun is that it preserves this ability to redirect input and output. Just remember that any options directly after srun such as –N will be used by srun. However, any options or piping commands after your program name will be used by the program only.

Dealing with batch files (-b option for srun)

In many cases, you will want to run your calculation on the scratch space for a particular node. This will prevent your calculation from writing to the NSF mounted /home directory and insure that you are not wasting time due to unnecessary data transfer between nodes. However, the srun command doesn't know which node you want to run on or in which directory your calculation will need to run. In these cases, it is essential to write a batch file that will guide your calculation along. The essential steps to include in your batch file (may24run.bat) are:

(1) Create an unique directory in your scratch space for this calculation (substitute your user name is the commands below):
   a. `mkdir /scratch/user_name/may24_run2/`
(2) Report back to a file in your home directory as to which node the calculation is running on and the time it started. This will help you track down the results when it is finished.
   a. `/bin/hostname > /home/user_name/may24_run2.log`
   b. `/bin/date >> /home/user_name/may24_run2.log`
(3) Now that we have created the directory, we need to move all the necessary input files over from the home directory.
   a. `cp /home/user_name/input_store/input.dat /scratch/user_name/may24_run2/`
(4) Hop into the directory we have created
   a. `cd /scratch/user_name/may24_run2/`
(5) Start the calculation
   a. `/home/user_name/bin/cool_program.x < input.dat > may24.out.run`
(6) Report back when it is finished. Leave some info in our home directory log file.
   a. `echo "Job Done" >> /home/user_name/may24_run2.log`
   b. `/bin/date >> /home/user_name/may24_run2.log`
(7) *(Optional)* At this point, the results can be accessed either by logging into the node where the calculation ran or by copying the results to a directory in your home space.
   a. `cp /scratch/user_name/may24_run2/may24.out.run /home/user_name/output_store/`

I will provide a sample batch file that you can adapt on the cluster soon. It provides the steps outlined above. Please adjust it to match your username and the program you will be using. When you finish setting it up, we will need to insure that it is executable.

```
$chmod +x batch_file.run
```

Also, when you start the batch file with srun, you need to make sure you use the –b option so srun knows that the file contains a series of commands

```
$srun –b batch_file.run
```

If everything goes well, you should see a new job listed when you type squeue.

Running parallel calculations:

Often we need to run parallel calculations that take advantage of several of the nodes on the cluster. This can be done using the slurm job submission with a few small modifications. Instead of running the calculation directly with srun, we are only going to use srun to reserve the nodes we need for the calculation. Let's say we want to run a parallel calculation on 4 nodes. First we allocate them for the calculation:

```
$srun –n 4 –A
```

The (-n) option tells slurm that we need 4 nodes for the calculation. The (-A) option reserves them for a parallel calculation.

After you type this, you will still see the same prompt, but srun has really opened a shell for all the following commands.

So we can now boot up LAM-MPI on these allocated nodes.

```
$lamboot
$mpirun –C parallel_program   (Run the program on the nodes allocated)
$lamclean                     (Clean up the lam universe after the run)
$lamhalt                      (Halt the lam universe)
$exit                         (Exit out of the shell the srun created)
```

Stopping a calculation (scancel)

There will be times when it will be necessary to stop a calculation. To do this we need the JOB ID for the calculation and the command scancel. The JOB ID can be determined using the squeue command described above. To cancel the job with ID = 84, just type:

```
$scancel 84
$
```

When you type squeue again that job should be gone.