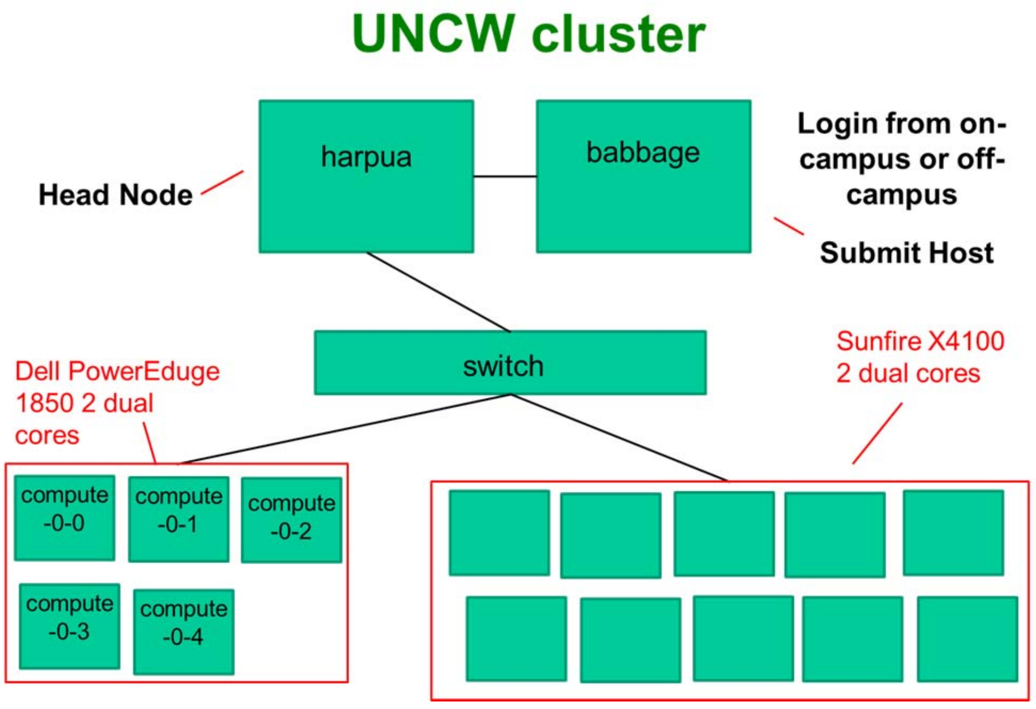
Using the UNC-W Cluster

C. Ferner and B. Wilkinson, August 17, 2014

Overview

The UNC-Wilmington cluster is shown below:



Note this cluster USES a job scheduler (the SGE job scheduler).

Login is through the submit host: **babbage.cis.uncw.edu**.

Editing and Compiling

Editing and compiling programs is done using normal resident Linux editors and compilers.

Running

Programs are submitted to the SGE scheduler with **qsub** command and job description file is necessary. For example, to submit a MPI executable called **hello**, you will need to create a file call **hello.sge** with the contents shown in Figure 1.

```
#!/bin/sh
#
# Usage: qsub hello.sge

#$ -S /bin/sh

#$ -pe orte 4          # Specify how many processors we want

# -- our name ---
#$ -N Hello            # Name for the job
#$ -l h_rt=00:01:00    # Request 1 minute to execute
#$ -cwd         # Make sure .e and .o files arrive in working directory
#$ -j y         # Merge standard out and standard error to one file

mpirun -np $NSLOTS ./hello
```

Figure 1: Job Submission File for Hello World

The first line of this script indicates that this is a shell script. The line `#$ -pe orte 4` indicates to SGE that you want 4 processing elements (processors). The line `#$ -N Hello` indicates that the output files should be named “HelloXXX”, where XXX contains other characters to indicate what it is and the job number. The line `#$ -cwd` tells SGE to use the current directory and the working directory (so that you output files will go in the current directory). The line `#$ -j y` will merge the stdout and stderr files into one file called “Hello.oXXX”, where XXX is the job number. Normally, SGE would create separate output files for the stdout and stderr, but this isn’t necessary.

The line `#$ -l h_rt=00:01:00` tells SGE to kill the job after a minute. This helps to keep the system clean of old jobs. If you find that a job is still in the queue after it has finished, you can delete it using `qdel #`, where # is the job number. If you expect your job to take longer than a minute to run, you will need to increase this time.

The last line of the submission file tells SGE to run the hello program using MPI run and the same number of processors as indicated in the `#$ -pe orte 4` line above.

To submit your job to the SGE, you enter the command:

```
qsub hello.sge
```

You job is given a job number, which you will use for other SGE command. The output of your program will be send to a filed called Hello.oXX, where XX is the job number. There will also be a file called Hello.poXX, which you don’t need. ***You will want to delete the output files you don’t need. Otherwise, your directory will fill up.***

To see the list of jobs that have been submitted and have not yet been terminated, use the command **`qstat`**. If you find that a job is still in the queue after it has finished, you can delete it using **`qdel #`**, where # is the job number.

After your job has completed, you should get output in the file Hello.oXX, with contents that should look something like that of Figure 2.

```
Master process 0 starting.
Hello world from process 0 on machine compute-0-0.local.
Hello world from process 1 on machine compute-0-0.local.
Hello world from process 2 on machine compute-0-0.local.
Hello world from process 3 on machine compute-0-0.local.
Goodbye world from process 0.
```

Figure 2: Hello world output

OpenMP

Compile an OpenMP program **`omp_hello.c`** with the regular gcc compiler using the command:

```
gcc -fopenmp omp_hello.c -o omp_hello
```

To execute the program, create a job submission file such as the one in Figure 3. Notice that the "**`#$ -pe orte 4`**" line is missing as well as the **`mpirun`** command. This is because the program does not use MPI.

Submit this job to the SGE scheduler using **`qsub`**.

```
#!/bin/sh
#
# Usage: qsub omp_hello.sge

#$ -S /bin/sh

# -- our name ---
#$ -N OMP_Hello           # Name for the job
#$ -l h_rt=00:01:00   # Request 1 minute to execute
#$ -cwd       # Make sure .e and .o files arrive in working directory
#$ -j y       # Merge standard out and standard error to one file

./omp_hello
```

Figure 3: Job Description File for an OpenMP program

Combined (hybrid) OpenMP/MPI programs

Compile hybrid program `omp-mpi_hello.c` with:

```
mpicc -fopenmp omp-mpi_hello.c -o omp-mpi_hello
```

Note this command invokes the regular gcc compiler.