THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE
Department of Electrical and Computer Engineering

CPE 412/512
Fall Semester 2017

Using the Batch Queuing System on the dmc.asc.edu system using *run_script* utility for MPI programs

The Batch Queuing system is the method by which students should execute programs on the Alabama Supercomputer DMC system. It allows your program to have exclusive access to the specified portion of the system.

To invoke the queuing system for MPI programs follow the following steps.

Step 1: Change to your appropriate working subdirectory under your home directory

Step 2: Create the source file and compile for mpi in the usual manner.

ex. **mpic++ hello_world_MPI.cpp -o hello_world_MPI**

This may require that you load the mpi environment using the

module load openmpi

command before executing the **mpic++** command.

Step 3: Then using your favorite text editor (vi, nano, emacs, pico, etc.) create a bash script file that contains the **module load** command for openmpi and at least one **srun** command to start the MPI system. The **srun** command functions in a similar manner to the **mpirun** and **mpiexec** commands but does not require a separate parameter to be entered as part of the command to specify the number of MPI processes that will be generated. With the **srun** command the number of MPI processes generated is automatically set to match the number of processing cores that was specified by the user when the job was submitted to the queuing system. For example if you want to have a two process MPI run for a file with an executable name *"hello_world_MPI"* you may create the file *"hello_world_MPI.sh"* which contains the following lines of text.

```
#!/bin/bash
module load openmpi
srun  hello_world_MPI
```

In this example, the number of MPI processes can then be set to 2 by requesting that two processing cores be used when this MPI program is executed by the batch queuing system. This is done in Step 5.

Step 4: Make sure that your execution script file has execute privileges

chmod 744 hello_world_MPI.sh

Step 5: To invoke the batch queuing system use the ***run_script*** command as the command line argument the name of your file. In this case you would type

run_script hello_world_MPI.sh.

The ***run_script*** command is interactive requiring you to respond to a number of questions as shown below:.

```
Queue                      Wall Time      Mem  # Cores
------------------------ ---------- -------- --------
express                      4:00:00     16gb      1-4
small                       60:00:00      4gb      1-8
medium                     150:00:00     16gb     1-16
large                      360:00:00    120gb     1-64
huge                       360:00:00    256gb   64-128
class                       16:00:00     64gb     1-64   enter class to
                                                        select "class"
Enter Queue Name (default <cr>: small) class            <- queue

Enter number of processor cores (default <cr>: 1 ) 2    <- enter 2 to reserve
                                                           two processing cores
Enter Time Limit (default <cr>: 16:00:00 HH:MM:SS)      <- accept defaults
                                                           press <enter>
Enter memory limit (default <cr>: 2gb )                 <- accept defaults
                                                           press <enter>
Enter a name for your job (default: helloworldMPIshSCR)

Run on; uv, dmc, nehalem, ivy-bridge, etc (default: any)
dmc <------------------------------------------------------ enter dmc

============================================================
=====         Summary of your script     job           =====
============================================================
  The script file is: hello_world_MPI.sh
  The time limit is 16:00:00 HH:MM:SS.
  The memory limit is: 2gb
  The job will start running after: 2016-09-14T20:35:53
  Job Name: helloworldMPIshSCR
  Virtual queue: class
  QOS: -p dmc,uv,knl --qos=class
  Constraints: --constraint=dmc
  Queue submit command:
sbatch -p dmc,uv,knl --qos=class -J helloworldMPIshSCR --begin=2016-09-14T20:35:53 --
requeue --mail-user=wellsbe@uah.edu -o helloworldMPIshSCR.o%A -t 16:00:00 -N 1-1 -n 2
--mem-per-cpu=1000mb --constraint=dmc                   <- Batch Queue
                                                           Job ID
Submitted batch job 49509
```

For this example you would answer the questions in the following manner. Enter **class** for *Queue Name*, **2** (in this case) for number of *processor cores* (will be used by the srun command(s) in the script file), and **dmc** for the machine it is to run on. Press the <Enter> key to accept all other defaults for the other prompts. All screen output should go to the file helloworldMPIshSCR.o49509 (i.e. the <filename>SCR.o<job number> file in general).

,

You can view your status using the ***squeue, squeue1, squeue2, squeue3,*** or ***squeue4*** commands which vary from one another in terms of the information format and job/queue attributes that are shown. For example, you can type

squeue4

to view the status of your job(s). If you are user *uahcls01* then the following output could be generated.

```
       JobID      User        QOS    JobName  ReqCPUS     ReqMem  Timelimit      State    Elapsed  Partition
------------ --------- ---------- ---------- -------- ---------- ---------- ---------- ---------- ----------
49509         uahcls01      class helloworl+       2     1000Mc   16:00:00    RUNNING   00:00:13 dmc-ivy-b+
```

Unlike program jobs run without this queue management in place, jobs run using the batch queue will be the only ones executed on the portion of the system you have reserved while your program is running. If your job run time exceeds the *Timelimit* then your job will be terminated and dequeued from the batch system. The *Timelimit* (hours:minutes:seconds) represents the maximum time product of parallel execution time multiplied by the number of processing cores that you specified when the job entered the queuing system. In this example the default value was taken when the queue was invoked.

You can also remove a job from the queue before it completes execution by using the ***scancel*** command with your job number as the command line argument. For example if you realize you made a mistake in your code and do not want job number 49509 run after it has been submitted to the queue, simply type

scancel 49509

to delete it.