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

CPE 412/512
Fall Semester 2015

How to Compile OpenACC programs and use the GPU Batch Queuing System on the DMC System (dmc.asc.edu).

Step 1: Set up access to the Portland Group OpenACC compiler. To do this type from the command line

```
module load pgi
```

Step 2: Then change to your appropriate working directory

Step 3: Create the source file (for example *laplace2d_acc.cpp*) and compile for OpenACC in the usual manner (of course you can incorporate this in a makefile). For example type

```
pgc++ laplace2d_acc.cpp -o laplace2d_acc  -acc -ta=nvidia -Minfo=accel
```

Make sure to use the *-acc -ta=nvidia -Minfo=accel* compiler options in addition to your usual ones. Note that the -Minfo option produces a lot of information about the effectiveness of generating GPU kernels from your openACC code.

Step 4: Using your favorite text editor (vi, nano, emacs, pico, textcomandocallofduty, etc.) create a shell script file that contains the name of your executable along with any appropriate command line parameters. For example to create a simple bash shell to execute the test program above that has no command line parameters you would enter the following on two separate lines of the file.

```
#!/bin/bash
./laplace2d_acc
```

Give the file a meaningful name (such as “laplace.scr” in this example).

Step 5: Next give this script file execute privileges by typing

```
chmod 744 laplace.scr
```

Note: This file will be called by a queuing script specifically designed to select a processing core that is attached to a GPU node. **It is important to remember that an OpenACC program requires that such an accelerator be present for your program to execute. If you try to execute this file using the normal queues (such as run_script) it will fail!** The program can currently only be executed on the dmc system. The uv system currently has no GPU resources.

Step 6: Then type the command ***run_gpu laplace.scr***. This is an interactive script and the following dialog information will appear.

```
Choose a batch job queue:

Queue                  CPU    Mem # CPUs
-------------- ---------- ------ ------
small-serial     40:00:00    4gb      1
medium-serial    90:00:00   16gb      1
large-serial    240:00:00  120gb      1
class             2:00:00   64gb   1-64
daytime           4:00:00   16gb    1-4
express          01:00:00  500mb      1

Enter Queue Name (default <cr>: small-serial) class

This software can only use one processor core.


Enter Time Limit (default <cr>: 01:00:00 HH:MM:SS)

Enter memory limit (default <cr>: 1gb )
Enter GPU architecture [t10/fermi/kepler/any] (default <cr>: any) kepler

The job name will be laplacescrGPU (can be changed in .asc_queue file)

===========================================================
=====         Summary of your script     job          =====
===========================================================
  The script file is:  laplace.scr
  The time limit is  01:00:00  HH:MM:SS.
  The memory limit is:  1gb
  The job will start running after:  201311151924.40
  Look for:  laplacescrGPU  in queue:  class

Job number 1042701.mds1.asc.edu
```

enter <u>class</u> to select the "class" queue

by default it uses only a single CPU core and a single GPU Device

accept defaults press <enter> or change time

accept defaults press <enter>

enter <u>kepler</u> or --<u>fermi</u> here since they are the newer GPUs on the system

Enter **<u>class</u>** for *Queue Name* and **<u>kepler</u>** for the type of GPU it is to run on. Press the enter key to accept the defaults for the other prompts (or enter in values that are more appropriate for your application). All screen output should go to the file laplacescrGPU.o1042701 (or in general the <filename(no dots or underscores)>GPU.o<job number> file).

You can always view your jobs status using the ***qstat*** command. To do this type

qstat

with no arguments. If you are user *uahcls38* then output will be formatted in the manner shown below.

```
                                                                   Req'd  Req'd     Elap
Job ID                  Username    Queue    Jobname          SessID  NDS   TSK   Memory Time    S Time
----------------------- ----------- -------- ---------------- ------ ----- ------ ------ --------- - ---------
1042701.mds1.asc.edu    uahcls42    class    laplacescrGPU       --     1  1:gpu    1gb 01:00:00 Q       --
```

You can also remove a job from the queue before it completes execution by using the ***qdel*** command with your <u>job number</u> as the command line argument. For example if you realize you made a mistake in your code and do not want job number 1042701 to run after it has been submitted to the queue, simply type

qdel 1042701

to delete it.