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

CPE 412/512

Fall Semester 2017

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

Step 1: First 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 within a makefile). For example to compile the laplace2d_acc.cpp source file 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.sh" in this example).

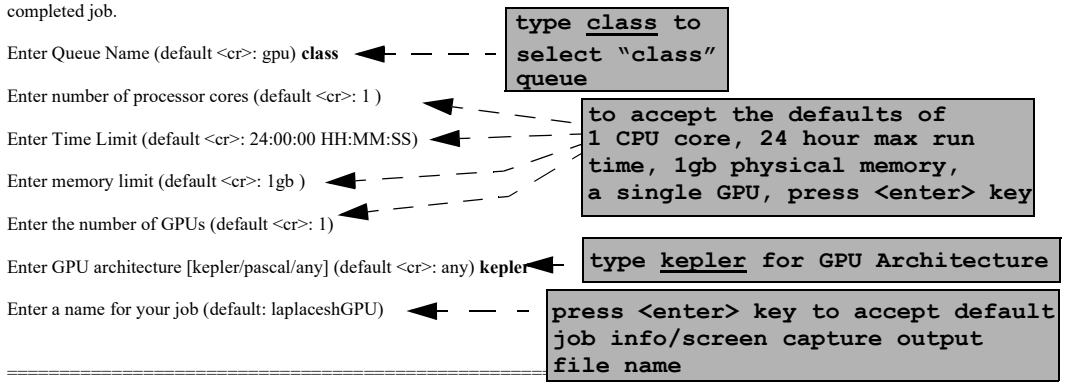
Step 5: Next give this script file execute privileges by typing

chmod 744 laplace.sh

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 start the run_gpu script by typing on the command line ***run_gpu*** followed by the name of the script file you have created (for example type ***run_gpu laplace.sh***). This is an interactive script and the following dialog information will appear.

```
This runs a script in the current directory via the queue system
Report problems and post questions to the HPC staff (hpc@asc.edu)


Choose a batch job queue:

gpu                  360:00:00    30gb     1-10
class                 24:00:00    64gb     1-60
Your job will have a shorter wait time if your memory request is
reasonable (about 20% more than needed), and your time request is
reasonable (about 50% more than needed).
Find this out by running 'jobinfo -j JOB_NUMBER' for a correctly
completed job.

Enter Queue Name (default <cr>: gpu) class
Enter number of processor cores (default <cr>: 1 )
Enter Time Limit (default <cr>: 24:00:00 HH:MM:SS)
Enter memory limit (default <cr>: 1gb )
Enter the number of GPUs (default <cr>: 1)
Enter GPU architecture [kepler/pascal/any] (default <cr>: any) kepler
Enter a name for your job (default: laplaceshGPU)

=========================================================
=====       Summary of your script     job         =====
=========================================================
 The script file is: laplace.sh
 The time limit is 24:00:00 HH:MM:SS.
 The memory limit is: 1gb
 The job will start running after: 2017-11-09T11:08:32
 Job Name: laplaceshGPU
 Virtual queue: class
 QOS: --qos=class
 Constraints:
 Queue submit command:
sbatch --qos=class  -J laplaceshGPU --begin=2017-11-09T11:08:32 --requeue --mail-user=wellsbe@uah.edu -o laplaceshGPU.o%A -t
24:00:00 --gres=gpu:1 -N 1-1 -n 1 --mem-per-cpu=1000mb

Submitted batch job 95553
```



Enter **class** for *Queue Name* at the first prompt and press the <enter> key. The next prompts asks you to enter the number of CPU type processing cores, the maximum amount of time you are requesting, the maximum amount of memory you are requesting and the number of GPU's you are requesting. To accept the defaults simply press the <enter> key without entering any text. For OpenACC you should select the kepler GPU architecture. The last prompt will ask you for your job name. If you accept the default by pressing the <enter> key without entering a different name the common screen output will be sent to a file that is given a default name of the general form

<*filename*>GPU.o<job number>,

where *filename* is the name of your script file where certain characters such as '.' and '_' are suppressed. For example, in the vector addition case that was generated by the *laplace.sh* script file this default output file was named laplaceshGPU.o95553, where 95553 was the queue job number assigned by the queuing system.

You can always view your jobs status using the ***squeue4*** command. To do this type

squeue4

with no arguments. If you are user *uahcls01* then the output will be formatted in the manner shown below.

```
                                                                                                  Req'd    Req'd       Elap
    JobIDRaw      User        QOS    JobName  ReqCPUS     ReqMem  Timelimit      State    Elapsed  Partition
------------ --------- ---------- ---------- -------- ---------- ---------- ---------- ---------- ---------
-
95553         uahcls01      class laplacesh+        1     1000Mc 1-00:00:00    RUNNING   00:00:10 dmc-haswe+
95553.0                            hostname        1     1000Mc               RUNNING   00:00:00
```

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 95553 to run (or its child GPU process 95553.0) after it has been submitted to the queue, simply type

scancel 95553

to delete it.