GPU Computing (CS 205)

- Odyssey and GPU Computing on Odyssey
- CUDA
  - Compiling and running CUDA code
- OpenACC
  - Using OpenACC
  - Links on OpenACC (with tutorials and sample OpenACC programs)
- PyCUDA
  - Using PyCUDA on Odyssey
  - PyCUDA links
- PyOpenCL
  - Using PyOpenCL on Odyssey
  - PyOpenCL links

Odyssey and GPU Computing on Odyssey

Before proceeding further, make yourself familiar with the basics of Odyssey and GPU computing on Odyssey:

https://rc.fas.harvard.edu/resources/odyssey-quickstart-guide/
https://rc.fas.harvard.edu/resources/documentation/gpgpu-computing-on-odyssey/

CUDA

Compiling and running CUDA code

1) Login to a node with a GPU, eg. holyseasgpu (for CS 205)
   srun --pty --x11=first -p holyseasgpu --mem 4000 -t 0-5:00 -n 1 -N 1 /bin/bash
2) In a command window, type:
   source new-modules.sh
   module load cuda/7.5-fasrc01
If your cuda program is 'test.cu', then you can compile with:
   nvcc test.cu -o test
You can also use makefiles. Here is an example cuda code with a makefile:
   main.cu
   makefile
   with these two files in the same directory, you can type, on the command line:
   make
   This will produce an executable 'square'
3) Run the cuda executable with the runscript:

   #!/bin/bash
   #SBATCH -p holyseasgpu #Partition to submit to
   #SBATCH -n 1 #Number of cores
   #SBATCH --gres=gpu
   #SBATCH -t 5 #Runtime in minutes
   #SBATCH --mem-per-cpu=100 #Memory per cpu in MB (see also --mem)
   #SBATCH --constraint=cuda-7.5
   source new-modules.sh
   module load cuda/7.5-fasrc01
time ./square > output.txt 2> errors.txt
-------------------------------------------------
OpenACC
On Odyssey the PGI OpenACC compiler suite is installed in /n/seasfs03/IACS/cs205/pgi. To make the compilers (pgcc, pgc++, pgf90, etc.) available in your path, add the following lines to your .bashrc file (assumes you are using bash, which is the default shell):

export PGI=/n/seasfs03/IACS/cs205/pgi
export PATH=$PGI/linux86-64/17.10/bin:$PATH
export MANPATH=$MANPATH:$PGI/linux86-64/17.10/man
export LM_LICENSE_FILE=$LM_LICENSE_FILE:$PGI/license.dat

Once you add these to your ~/.bashrc, to make these take effect, you can do, in a terminal:

source ~/.bashrc
or
./~/.bashrc
or you can log out and log back in.

Using OpenACC
You need to first compile your code (say code_acc.c or code_acc.f90) containing OpenACC (see below for example programs).

Note that pgcc and pgf90 should be available in your path for this to succeed (see above for instructions).

For c program:
pgcc -acc code_acc.c -Minfo=accel

For fortran program:
pgf90 -acc code_acc.f90 -Minfo=accel

will create an executable with name a.out. The option -Minfo=accel will display useful information on parallelization.

Slurm Script for running the job on odyssey:

--------------------------------
#!/bin/bash
#SBATCH -N 1  #Number of nodes
#SBATCH -p holyseasgpu  #Partition to submit to
#SBATCH --ntasks-per-node 2
#SBATCH --gres=gpu:1
#SBATCH -t 5  #Runtime in minutes
./a.out
---------------------------------

Links on OpenACC (with tutorials and sample OpenACC programs)

1) OpenACC example programs

On Odyssey, you can find the OpenACC example programs in:
$PGI/linux86-64/2016/examples/OpenACC

2) https://www.psc.edu/index.php/136-users/training/2430-xsed-hpc-workshop-december-6-2016-openacc
   Introduction to OpenACC (with sample code including Laplace) is in:
   https://www.psc.edu/images/xsedetraining/OpenACCDec2016/OpenACC_Introduction_To_OpenACC.PDF

3) https://devblogs.nvidia.com/parallelforall/openacc-example-part-1/

4) http://www.openacc.org/Sample_Codes

5) Introductory OpenACC tutorial (free, but requires an account):
   https://nvidia.qwiklab.com/quests/3?locale=en

PyCuda

Using PyCuda on Odyssey

The modules that need to be loaded for using pycuda on odyssey are as under (after sourcing new-modules.sh if you have not done that):

module load Anaconda/1.9.2-fasrc01
module load cuda/7.5-fasrc01
module load pycuda/2015.1.3-fasrc01

Pycuda examples come with the pycuda distribution. You can clone the pycuda repository as:

git clone --recursive http://git.tiker.net/trees/pycuda.git

The examples are in the directory:
pycuda/examples/

Using the example "demo_elementwise.py" (demo_elementwise.py), a sample slurm script would be (pycuda_submit_wiki.sh)

-------------------------------
#!/bin/bash
#SBATCH -p holyseasgpu
#SBATCH -n 4
#SBATCH --gres=gpu:1
#SBATCH --mem-per-cpu=4000
# Load required modules
source new-modules.sh
module load Anaconda/1.9.2-fasrc01
module load cuda/7.5-fasrc01
module load pycuda/2015.1.3-fasrc01
module list
# Run program
python demo_elementwise.py
-------------------------------

(The example "dump_properties.py" (in pycuda/examples) will list the properties of cuda node on which slurm ran)

PyCuda links
1) The following is one of the most important resources:

https://mathema.tician.de/software/pycuda/

2) Pycuda getting started tutorial:

https://documen.tician.de/pycuda/tutorial.html

3) More Pycuda examples (note many of these are for interactive work, i.e. not suitable for slurm batch mode and may require additional tools not available on Odyssey).

https://wiki.tiker.net/PyCuda/Examples

**PyOpenCL**

**Using PyOpenCL on Odyssey**

The module that needs to be loaded for using PyOpenCL on odyssey is "pyopencl". Thus, one needs to do:

```
source new-modules.sh
module load pyopencl
```

before PyOpenCL can be used.

PyOpenCL examples come with the PyOpenCL distribution. You can clone the PyOpenCL repository as:

```
git clone --recursive http://git.tiker.net/trees/pyopencl.git
```

The examples are in the directory:

```
pyopencl/examples/
```

Using the example "demo_elementwise.py", a sample slurm script would be:

```
#!/bin/bash
#SBATCH -J elementwise_pyopencl
#SBATCH -o elementwise.out
#SBATCH -e elementwise.err
#SBATCH -p holyseasgpu
#SBATCH -n 2
#SBATCH --gres=gpu:1
#SBATCH --mem-per-cpu=500
# Load required modules
export PYOPENCL_CTX=''
source new-modules.sh
module load pyopencl
#Run program
python demo_elementwise.py
```

**PyOpenCL links**

1) https://mathema.tician.de/software/pyopencl/

2) https://wiki.tiker.net/PyOpenCL
3) https://www.khronos.org/opencl/