Lecture 7

- Demonstration of the resources available on Prince HPC cluster
- Fourier methods for financial models and applications

7.1 Accessing Prince HPC Cluster and Using CUDA C on Linux Operating System

- Go through steps for setting up and logging into HPC cluster

  https://wikis.nyu.edu/display/NYUHPC/High+Performance+Computing+at+NYU

  https://wikis.nyu.edu/display/NYUHPC/Clusters+-+Prince

- Do Nsight Eclipse IDE, and show code examples

  - multi-thread in C++ on Linux is a little different: use pthread

- Run examples on Prince HPC: multi-threading example that uses multiple GPU’s

- Executing batch jobs on Prince

7.1 Prince Cluster and CUDA C on Linux Operating System (cont.)

- Logging into Prince cluster is a 2 step process that requires your NYU netid. There is information on the NYU network - see web pages

- First, you need to download Putty, Xming, and WinSCP, if you plan to log in to Prince from a Windows machine

- Open Xming by clicking on icon. It runs in the background

  First, you need to open a tunnel using Putty. Instructions on how to set this up are on web site. Leave this screen in the background. Then you start a 2nd Putty. Again, the web site has instruction for setting this up. This is a somewhat complex 2 step process, but it works. This places in a Linux terminal session on the HPC cluster.

- → Examples

7.1 **Prince Cluster and CUDA C** (cont.)

- Use WinSCP to transfer files to your space on the Prince. On left panel, click on “Transferring data to/from Prince cluster

  got to https://wikis.nyu.edu/pages/viewpage.action?pageId=84612833

- You will need to use Linux commands on the cluster

ls	list directories and files in current directory
ls –la	list directories and files, with file sizes
cd name	change to directory, may need full path, cd ~/MyTestFiles/SourceCode
cd ..	Go back to the next highest directory
mkdir name	make a new directory, in current location
rm filename	removes, or deletes filename (be careful)
rmdir name	removes directory name
cat filename	prints filename to the screen
less filename	prints filename to the screen, in pages (space bar to continue, q to quit)
cp	copy files, cp filename1 ~/Directory/filename2

You can use man command to retrieve information on Linux commands

Highlight text and copy with right-click if you are on Windows

7.1 Prince Cluster and CUDA C (cont.)

- To build or make executable files to run C++ or Cuda C programs, use Nsight Eclipse. You can copy C++ and Cuda C programs to Prince cluster, but you must do some editing and rebuild on Prince (Windows executables do not run on Linux systems)

- Copies your source code and parameter text files to Prince using WinSCP. I recommend editing text files with gedit. Use Nsight to edit and rebuild C++ and Cuda C programs. Linux systems also have Emacs

- Use the following commands to set up cuda on Prince.

```
$ module avail        →  to see modules available
$ module load cuda/10.0.130
```

- Version 8 and 9 are also available. Use $ module spider cuda to get m ore information.

- Start the Nsight Eclipse IDE by executing the following command

```
$ nsight
```

- This GUI can be slow to start up.

7.1 Prince Cluster and CUDA C (cont.)

- Open Nsight

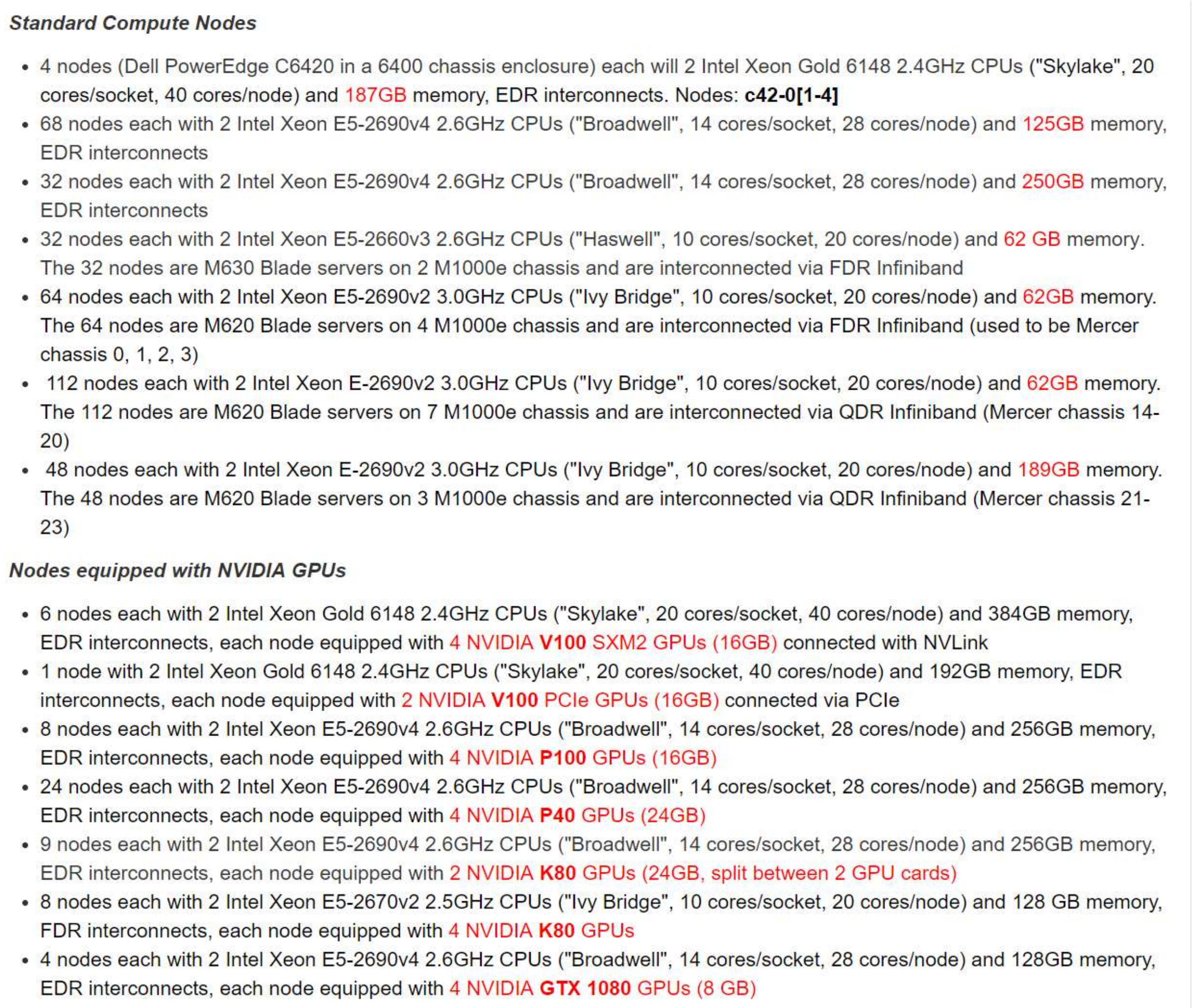
- To start a new C++ project, right click the tab at the top left of Nsight and choose New → C++ project

  1) on the next screen, choose Empty Project and Linux GCC,, and type in a project name
  2) in the directory created for this project, create an src directory (mkdir src)
  3) copy your source code and header files into the src directory
  4) open the project
  5) recommend: right-click on the project and click on "Close unrelated projects"
  6) right click on project and go to Properties at the bottom of menu. From the next screen, choose Build → Settings. For multi-threading, you need to add –pthread to G++ and G++ Linker.
  7) After completing edits, use Save All. Do a Build using Debug and do a second Build using Release. You will need to add –pthread to the Build Settings again (both G++ and G++ Linker) for release

- To start a new Cuda C project, choose New → Cuda C/C++ project

  1) on the next screen, choose Empty Project and Cuda C, and type in a project name
  2) on the next screen, check 3.7 for the K80 and 7.0 for the V100 in both rows
  3) in the directory created for this project, create an src directory (mkdir src)
  4) copy your source code and header files into the src directory
  5) open the project
  6) recommend: right-click on the project and click on "Close unrelated projects"
  7) After completing edits, use Save All. Do a Build using Debug and do a second Build using Release.

7.1 Prince Cluster and CUDA C (cont.)



Standard Compute Nodes

- 4 nodes (Dell PowerEdge C6420 in a 6400 chassis enclosure) each will 2 Intel Xeon Gold 6148 2.4GHz CPUs ("Skylake", 20 cores/socket, 40 cores/node) and 187GB memory, EDR interconnects. Nodes: **c42-0[1-4]**
- 68 nodes each with 2 Intel Xeon E5-2690v4 2.6GHz CPUs ("Broadwell", 14 cores/socket, 28 cores/node) and 125GB memory, EDR interconnects
- 32 nodes each with 2 Intel Xeon E5-2690v4 2.6GHz CPUs ("Broadwell", 14 cores/socket, 28 cores/node) and 250GB memory, EDR interconnects
- 32 nodes each with 2 Intel Xeon E5-2660v3 2.6GHz CPUs ("Haswell", 10 cores/socket, 20 cores/node) and 62 GB memory. The 32 nodes are M630 Blade servers on 2 M1000e chassis and are interconnected via FDR Infiniband
- 64 nodes each with 2 Intel Xeon E5-2690v2 3.0GHz CPUs ("Ivy Bridge", 10 cores/socket, 20 cores/node) and 62GB memory. The 64 nodes are M620 Blade servers on 4 M1000e chassis and are interconnected via FDR Infiniband (used to be Mercer chassis 0, 1, 2, 3)
- 112 nodes each with 2 Intel Xeon E-2690v2 3.0GHz CPUs ("Ivy Bridge", 10 cores/socket, 20 cores/node) and 62GB memory. The 112 nodes are M620 Blade servers on 7 M1000e chassis and are interconnected via QDR Infiniband (Mercer chassis 14-20)
- 48 nodes each with 2 Intel Xeon E-2690v2 3.0GHz CPUs ("Ivy Bridge", 10 cores/socket, 20 cores/node) and 189GB memory. The 48 nodes are M620 Blade servers on 3 M1000e chassis and are interconnected via QDR Infiniband (Mercer chassis 21-23)

Nodes equipped with NVIDIA GPUs

- 6 nodes each with 2 Intel Xeon Gold 6148 2.4GHz CPUs ("Skylake", 20 cores/socket, 40 cores/node) and 384GB memory, EDR interconnects, each node equipped with 4 NVIDIA **V100** SXM2 GPUs (16GB) connected with NVLink
- 1 node with 2 Intel Xeon Gold 6148 2.4GHz CPUs ("Skylake", 20 cores/socket, 40 cores/node) and 192GB memory, EDR interconnects, each node equipped with 2 NVIDIA **V100** PCIe GPUs (16GB) connected via PCIe
- 8 nodes each with 2 Intel Xeon E5-2690v4 2.6GHz CPUs ("Broadwell", 14 cores/socket, 28 cores/node) and 256GB memory, EDR interconnects, each node equipped with 4 NVIDIA **P100** GPUs (16GB)
- 24 nodes each with 2 Intel Xeon E5-2690v4 2.6GHz CPUs ("Broadwell", 14 cores/socket, 28 cores/node) and 256GB memory, EDR interconnects, each node equipped with 4 NVIDIA **P40** GPUs (24GB)
- 9 nodes each with 2 Intel Xeon E5-2690v4 2.6GHz CPUs ("Broadwell", 14 cores/socket, 28 cores/node) and 256GB memory, EDR interconnects, each node equipped with 2 NVIDIA **K80** GPUs (24GB, split between 2 GPU cards)
- 8 nodes each with 2 Intel Xeon E5-2670v2 2.5GHz CPUs ("Ivy Bridge", 10 cores/socket, 20 cores/node) and 128 GB memory, FDR interconnects, each node equipped with 4 NVIDIA **K80** GPUs
- 4 nodes each with 2 Intel Xeon E5-2690v4 2.6GHz CPUs ("Broadwell", 14 cores/socket, 28 cores/node) and 128GB memory, EDR interconnects, each node equipped with 4 NVIDIA **GTX 1080** GPUs (8 GB)

7.1 **Prince Cluster and CUDA C** (cont.)

- Go to class examples on Prince cluster
- Recommended optimizations for g++ or gcc, for code on CPU only (on Linux machines)
  -O3 –ffast-math –march=native –funroll-loops
- Recommended optimizations for Cuda C, for code on GPU

  use -O3 for CPU or host code
  choose –O3 and fast math option
- Optimizations in Windows Visual Studio
  - For C++ CPU only code, go to Properties or Property Pages, and choose C/C++ → Optimization → Maximize speed (//O2)
  - For programs running on GPU, under Cuda C, choose Yes for Use Fast Math and Maximize Speed (//O2) for Optimization

7.1 **Prince Cluster and CUDA C** (cont.)

- One CANNOT run GPU executables on Prince. The Prince cluster is set up for batch jobs only.

- See the web site for instructions on how to set up batch jobs using *.sh files

  https://wikis.nyu.edu/display/NYUHPC/Submitting+jobs+with+sbatch

- From the command line you type $ sbatch filename.sh

- For information on Prince GPU's go to

  https://wikis.nyu.edu/display/NYUHPC/Clusters+-+Prince

- To access GPU's , add the following lines to batch file

  ```
  #SBATCH --gres=gpu:1    or    #SBATCH --gres=gpu:v100:1  to use a V100
  #SBATCH --gres=gpu:k80:2  will set up batch job to use 2 GPU's
  ```

- Do examples of batch files, and run one on Prince

7.1 Prince Cluster and CUDA C (cont.)

- LOSBatch_GPU.sh -- batch file to run a multi-threaded program on 2 GPU’s

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=3
#SBATCH --time=5:00:00
#SBATCH --mem=4GB
#SBATCH --job-name=LOS_Test_HW_Model_GPU
#SBATCH --mail-type=END
#SBATCH --mail-user=ls127@nyu.edu
#SBATCH --output=slurm_%j.out

module purge
cd ~/MytestRuns
Release/Test_FD_LIBOROption_GPU

RUNDIR=$SCRATCH/my_project/run-${SLURM_JOB_ID/.*}
mkdir -p $RUNDIR
```