The OSU College of Engineering DGX System for Advanced GPU Computing

Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License
dgx_system.pptx

OSU’s College of Engineering has six Nvidia DGX-2 systems

Each DGX server:
- Has 16 NVidia Tesla V100 GPUs
- Has 28TB of disk, all SSD
- Has two 24-core Intel Xeon 8168 Platinum 2.7GHz CPUs
- Has 1.5TB of DDR4-2666 System Memory
- Runs the CentOS 7 Linux operating system

Overall compute power:
- Each V100 NVidia Tesla card has 5,120 CUDA Cores and 640 Tensor Cores
- This gives each 16-V100 DGX server a total of 81,920 CUDA cores and 10,240 Tensor cores
- This gives the entire 6-DGX package a total of 491,520 CUDA Cores and 61,440 Tensor Cores

Performance Comparison with one of our other Systems

BTW, you can also use the rabbit machine:
ssh rabbit.engr.oregonstate.edu
It is a good place to write your code and get it to compile.
It is not a good place to do the final run of your code.

How to SSH to the DGX Systems

flip3 151%
ssh submit-c.hpc.engr.oregonstate.edu
It is a good place to write your code and get it to compile.
It is not a good place to do the final run of your code.

Submitting a Non-batch Test-CUDA job to the DGX System

run.bash:

```
#!/bin/bash
/usr/local/apps/cuda/cuda-10.1/bin/nvcc -o montecarlo montecarlo.cu
./montecarlo
```

submit-c 142%
srun -A cs475-575 -p classgputest --pty bash run.bash
Number of Trials =       2048, Blocksize =        8, MegaTrials/Second =    58.8235, Probability =  26.92%

How to Check on the DGX Systems

submit-c 144%
sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST

submit-c 145%
squeue
JOBID PARTITION     NAME    USER ST    TIME      NODES NODELIST(REASON)

Submit a Non-batch Test-CUDA job to the DGX System

Create a bash shell file

```
#!/bin/bash
```

Note: A single dash (-) is used for a single character flag
A double dash (--) is used for a word (more than a single character) flag

```
run.bash:
```

```
Note: A single dash (-) is used for a single character flag
A double dash (--) is used for a word (more than a single character) flag
```

These 2 lines are actual bash code

```
submit-c 166% srun -A cs475-575 -p classgputest --pty bash run.bash
```

```
submit-c 167%
```

Submit a Non-batch Test-CUDA job to the DGX System

Class partitions

The is the partition name that we use for our class when running Matlab

```
Note: A single dash (-) is used for a single character flag
A double dash (--) is used for a word (more than a single character) flag
```

The bash script

```
run.bash:
```

```
Note: A single dash (-) is used for a single character flag
A double dash (--) is used for a word (more than a single character) flag
```

These 2 lines are actual bash code

```
submit-c 167%
```

Submit a Non-batch Test-CUDA job to the DGX System

Class partitions

The is the partition name that we use for our class when running Matlab

```
Note: A single dash (-) is used for a single character flag
A double dash (--) is used for a word (more than a single character) flag
```

These 2 lines are actual bash code

```
submit-c 167%
```

Submit a Non-batch Test-CUDA job to the DGX System

Class partitions

The is the partition name that we use for our class when running Matlab

```
Note: A single dash (-) is used for a single character flag
A double dash (--) is used for a word (more than a single character) flag
```

These 2 lines are actual bash code

```
submit-c 167%
```
Submitting a Batch Final-CUDA job to the DGX System

Create a bash shell file that looks like this:

```bash
#!/bin/bash
#SBATCH -J MonteCarlo
#SBATCH -A cs475-575
#SBATCH -p classgpufinal
#SBATCH --constraint=v100
#SBATCH --gres=gpu:1
#SBATCH -o montecarlo.out
#SBATCH -e montecarlo.err
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --mail-user=joeparallel@oregonstate.edu
/usr/local/apps/cuda/cuda-10.1/bin/nvcc -o montecarlo montecarlo.cu
./montecarlo
```

Submit the job described in your shell file.

What is the Difference Between the Partitions `classgputest` and `classgpufinal`?

`classgputest` lets your program get into the system sooner, but it might be running alongside other jobs, so its performance might suffer. But, you don't care because you are just compiling and debugging, not taking performance numbers for your report.

`classgpufinal` makes your program wait in line until it can get dedicated resources so that you get performance results that are much more representative of what the machine can do, and thus are worthy to be listed in your report.

Auto-Notifications via Email

```bash
#SBATCH --mail-user=joeparallel@oregonstate.edu
```

You don't have to do this, but if you do, please be sure you get your own email address right!

Our IT people are getting really tired of finding the bounced emails when people spell their own email address wrong.

Submitting a Loop

```bash
#!/bin/bash
#SBATCH -J MonteCarlo
#SBATCH -A cs475-575
#SBATCH -p classgpufinal
#SBATCH --constraint=v100
#SBATCH --gres=gpu:1
#SBATCH -o montecarlo.out
#SBATCH -e montecarlo.err
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --mail-user=joeparallel@oregonstate.edu

for t in 2048 8192 131072 2097152
   do
      for b in 8 16 32 64 128 256
         do
            /usr/local/apps/cuda/cuda-10.1/bin/nvcc -DNUMTRIALS=$t -DBLOCKSIZE=$b -o montecarlo montecarlo.cu
            ./montecarlo
         done
   done
```

Use slurm's `scancel` if your Job Needs to Be Killed

```bash
submit-c 163% sbatch submitloop.bash
Submitted batch job 475
```

```bash
submit-c 164% scancel 475
```
Submitting an OpenCL job to the DGX System

submit.bash:

```
#!/bin/bash
#SBATCH  -J  PrintInfo
#SBATCH  -A  cs475-575
#SBATCH  -p  classgpufinal
#SBATCH  --constraint=v100
#SBATCH  --gres=gpu:1
#SBATCH  -o  printinfo.out
#SBATCH  -e  printinfo.err
#SBATCH  --mail-type=BEGIN,END,FAIL
#SBATCH  --mail-user=joeparallel@oregonstate.edu

g++ -o printinfo printinfo.cpp /usr/local/apps/cuda/cuda-10.1/lib64/libOpenCL.so.1.1 -lm -fopenmp
./printinfo
```

Here's what `printinfo` got on one graphics card on the DGX System

Number of Platforms = 1

Platform #0:
- Name = 'NVIDIA CUDA'
- Vendor = 'NVIDIA Corporation'
- Version = OpenCL 1.2 CUDA 11.2.153
- Profile = 'FULL_PROFILE'

Number of Devices = 1

Device #0:
- Type = 0x0004 = CL_DEVICE_TYPE_GPU
- Device Vendor ID = 0x10de (NVIDIA)
- Device Maximum Compute Units = 80
- Device Maximum Work Item Dimensions = 3
- Device Maximum Work Item Sizes = 1024 x 1024 x 64
- Device Maximum Work Group Size = 1024
- Device Maximum Clock Frequency = 1530 MHz

Device Extensions:
- cl_khr_global_int32_base_atomics
- cl_khr_global_int32_extended_atomics
- cl_khr_local_int32_base_atomics
- cl_khr_local_int32_extended_atomics
- cl_khr_fp64
- cl_khr_byte_addressable_store
- cl_khr_icd
- cl_khr_gl_sharing
- cl_nv_compiler_options
- cl_nv_device_attribute_query
- cl_nv pragma unroll
- cl_nv_copy_opts
- cl_nv_create_buffer

For comparison, rabbit's graphics card has 15 Compute Units.