The OSU College of Engineering DGX System
for Advanced GPU Computing

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

How to SSH to the DGX 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 Check on the DGX Systems

Check on the queues

<table>
<thead>
<tr>
<th>JOBS</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER ST TIME</th>
<th>NODES NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mime4</td>
<td>_only</td>
<td>jayasurw K 1-16:31:03</td>
<td>1 compute-e-2</td>
</tr>
<tr>
<td></td>
<td>mime4</td>
<td>_only</td>
<td>jayasurw K 1-16:31:03</td>
<td>1 compute-e-2</td>
</tr>
<tr>
<td></td>
<td>share</td>
<td>_only</td>
<td>chukwuk K 1-23:34:45</td>
<td>1 compute-2-6</td>
</tr>
<tr>
<td></td>
<td>share</td>
<td>_only</td>
<td>chukwuk K 1-23:34:45</td>
<td>1 compute-2-6</td>
</tr>
<tr>
<td></td>
<td>dgp2</td>
<td>_only</td>
<td>bahu K 1-22:50:44</td>
<td>1 compute-dgp2-3</td>
</tr>
<tr>
<td></td>
<td>dgp2</td>
<td>_only</td>
<td>bahu K 1-22:50:44</td>
<td>1 compute-dgp2-3</td>
</tr>
<tr>
<td></td>
<td>dge2</td>
<td>_only</td>
<td>mishra K 5-16:49:09</td>
<td>1 compute-dge2-5</td>
</tr>
<tr>
<td></td>
<td>dge2</td>
<td>_only</td>
<td>mishra K 5-16:49:09</td>
<td>1 compute-dge2-5</td>
</tr>
<tr>
<td></td>
<td>dge2</td>
<td>_only</td>
<td>aimesen K 6-18:24:44</td>
<td>1 compute-dge2-3</td>
</tr>
<tr>
<td></td>
<td>dge2</td>
<td>_only</td>
<td>aimesen K 6-18:24:44</td>
<td>1 compute-dge2-3</td>
</tr>
</tbody>
</table>

submit-c 167% srun

PARTITION AVAIL TIMELIMIT NODES STATE NODELIST

share* up 7-00:00:00 2 drain compute-4-[3-4]
share* up 7-00:00:00 1 mix compute-2-6
sharegpu up 7-00:00:00 3 idle compute-dgxs-[2-3],compute-gpu
dgp2 up 7-00:00:00 1 drain compute-dgp2-2
gp2 up 7-00:00:00 5 mix compute-dgp2-[1-6]
gpu up 7-00:00:00 2 mix compute-gpu[3-4]
gp2 up 7-00:00:00 1 idle compute-gp2
gp2 up 7-00:00:00 1 down compute-gp2
dx up 7-00:00:00 3 mix compute-dgx-[4-6]
dgx up 7-00:00:00 1 mix compute-dgx-2
up1 up 7-00:00:00 2 idle compute-dgx-[2-3]
class up 1:00:00 2 idle compute-dgx-[2-3]
class up 7-00:00:00 1 mix compute-2-6

Create a bash shell file that looks like this

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

submit-c 166% srun

```
#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-c 143% sbatch

```
submit.bash:
```

check the output

| 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

Submitting a Batch Final-CUDA job to the DGX System

submit.c 143% sbatch

Submitted batch job 474

submit.c 144% cat montecarlo.err

What is the Difference Between the Partitions
classgpufinal and classgpufinal?

classgpufinal 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

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 real tired of fielding the bounced emails when people spell their own email address wrong.

---

Submitting a Loop

submitloop.bash:

```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
```

These 8 lines are actual bash code

submit-c 153% submitloop.bash

Submitted batch job 475

submit-c 154% tail -f montecarlo.err

Displays the latest output added to montecarlo.err Keeps doing it forever. Control-c to get out of it.

---

What Showed up in my Email (which I spelled correctly)

From: Slurm workload manager
To: Slurm Job_id=3980 Name=MATRIXMul Ended. Run time 00:00:12. COMPLETED, ExitCode 0

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

submit-c 163% sbatch submitloop.bash
Submitted batch job 476

submit-c 164% scancel 476
Submitting an OpenCL job to the DGX System

submit.bash:

```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
```

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_int
      cl_khr_pthreads
      cl_khr_sharing
      cl_khr_compiler_options
      cl_khr_diagnostics
      cl_khr_external_semaphore
      cl_khr_external_mem
      cl_khr_get_mem_info
      cl_khr_itel
      cl_khr_map旳
      cl_khr_pragma_unroll
      cl_khr_pragma_unroll
      cl_khr_pragma_vectorize
      cl_khr_profiler
      cl_khr_vulkan
      cl_khr_copy_host
      cl_khr_create_buffer

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