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 previous 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
submit-c 142% module load slurm
```

ssh over to a DGX submission machine – submit-a and submit-b will also work

Type this right away to set your path correctly.
How to Check on the DGX Systems

Check on the queues

PARTITION| NAME| USER| ST| TIME| NODES| NODELIST
---| ---| ---| ---| ---| ---| ---
mime4| c_only| jayasurw| R| 1-10:32:19| 1| compute-e-1
share| CH3COOH| chukwuk| R| 1-23:36:45| 1| compute-2-6
nerhp| tcsh| dionnec| R| 8:59:45| 1| compute-h-8
dgx2| bash| heli| R| 1-22:50:44| 1| compute-dgx2-1
dgx2| bash| chenju3| R| 13:47:36| 1| compute-dgx2-4
dgx2| bash| mish rash| R| 5-16:48:09| 1| compute-dgx2-5

dgx2| bash| azieren| R| 5-16:48:09| 1| compute-dgx2-3

dgx2| bash| azieren| R| 6-17:34:00| 1| compute-dgx2-3

dgx2| bash| azieren| R| 6-17:24:44| 1| compute-dgx2-3

Submit a CUDA job to the DGX Systems using Slurm

Create a shell file

submit.bash:

```bash
#!/bin/bash
#SBATCH  -J  MatrixMult
#SBATCH  -A  cs475-575
#SBATCH  -p  class
#SBATCH  --gres=gpu:1
#SBATCH  -o  matrixmul.out
#SBATCH  -e  matrixmul.err
#SBATCH  --mail-type=BEGIN,END,FAIL
#SBATCH  --mail-user=joeparallel@oregonstate.edu
/usr/local/apps/cuda/cuda-11.1/bin/nvcc -o matrixMul matrixMul.cu
./matrixMul
```

Submit the job described in your shell file

Submit batch job 474

Check the output
(I like sending my output to standard error, not standard output)

Auto-Notifications via Email

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

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

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

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

Slurm workload manager Slurm_job_id=3980 Name=MatrixMul Ended, Run time 00:00:12, COMPLETED, Exit Code:0
Slurm workload manager Slurm_job_id=3980 Name=MatrixMul Began, Queued time 00:00:01
submitloop.bash:

```bash
#!/bin/bash
#SBATCH -J MatrixMul
#SBATCH -A cs475-575
#SBATCH -p class
#SBATCH --gres=gpu:1
#SBATCH -o matrixmul.out
#SBATCH -e matrixmul.err
#SBATCH --mail-type=BEGIN,END,FAIL
#SBATCH --mail-user=joeparallel@oregonstate.edu
for t in 1 2 4 8 16 32
do
  /usr/local/apps/cuda/cuda-11.1/bin/nvcc -DNUMT=$t -o matrixMul matrixMul.cu
  ./matrixMul
done
```

submit-c 153% sbatch submitloop.bash
Submitted batch job 475

submit-c 154% tail -f matrixmul.err
Displays the latest output added to matrixmul.err.
Keeps doing it forever.
Control-c to get out of it.

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

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

submit-c 164% scancel 475

submit.bash:

```bash
#!/bin/bash
#SBATCH -J MatrixMul
#SBATCH -A cs475-575
#SBATCH -p class
#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-11.1/lib64/libOpenCL.so.1.1 -lm -fopenmp
```

Results for Multiplying two 1024x1024 Matrices

GigaFlops during Matrix Multiplication

NUMT

(A CUDA block was actually NUMT x NUMT threads)

Submitting an OpenCL job to the DGX Systems using Slurm

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

<table>
<thead>
<tr>
<th>Number of Platforms = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Platform #0:</strong></td>
</tr>
<tr>
<td>Name = 'NVIDIA CUDA'</td>
</tr>
<tr>
<td>Vendor = 'NVIDIA Corporation'</td>
</tr>
<tr>
<td>Version = OpenCL 1.2 CUDA 11.2.153</td>
</tr>
<tr>
<td>Profile = 'FULL_PROFILE'</td>
</tr>
<tr>
<td>Number of Devices = 1</td>
</tr>
<tr>
<td><strong>Device #0:</strong></td>
</tr>
<tr>
<td>Type = 0x0004 = CL DEVICE_TYPE_GPU</td>
</tr>
<tr>
<td>Device Vendor ID = 0x10de (NVIDIA)</td>
</tr>
<tr>
<td>Device Maximum Compute Units = 80</td>
</tr>
<tr>
<td>Device Maximum Work Item Dimensions = 3</td>
</tr>
<tr>
<td>Device Maximum Work Item Sizes = 1024 x 1024 x 64</td>
</tr>
<tr>
<td>Device Maximum Work Group Size = 1024</td>
</tr>
<tr>
<td>Device Maximum Clock Frequency = 1530 MHz</td>
</tr>
<tr>
<td><strong>Device Extensions:</strong></td>
</tr>
<tr>
<td>cl_khr_global_int32_base_atomics</td>
</tr>
<tr>
<td>cl_khr_global_int32_extended_atomics</td>
</tr>
<tr>
<td>cl_khr_local_int32_base_atomics</td>
</tr>
<tr>
<td>cl_khr_local_int32_extended_atomics</td>
</tr>
<tr>
<td>cl_khr_fp64</td>
</tr>
<tr>
<td>cl_khr_byte_addressable_store</td>
</tr>
<tr>
<td>cl_khr_icd</td>
</tr>
<tr>
<td>cl_khr_gl_sharing</td>
</tr>
<tr>
<td>cl_nv_compiler_options</td>
</tr>
<tr>
<td>cl_nv_device_attribute_query</td>
</tr>
<tr>
<td>cl_nv_pragma_unroll</td>
</tr>
<tr>
<td>cl_nv_copy_profiler</td>
</tr>
<tr>
<td>cl_nv_create_buffer</td>
</tr>
</tbody>
</table>

For reference, rabbit’s graphics card has 15 Compute Units