

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](https://creativecommons.org/licenses/by-nc-nd/4.0/)



Oregon State
University
Computer Graphics

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

2

Each DGX server:

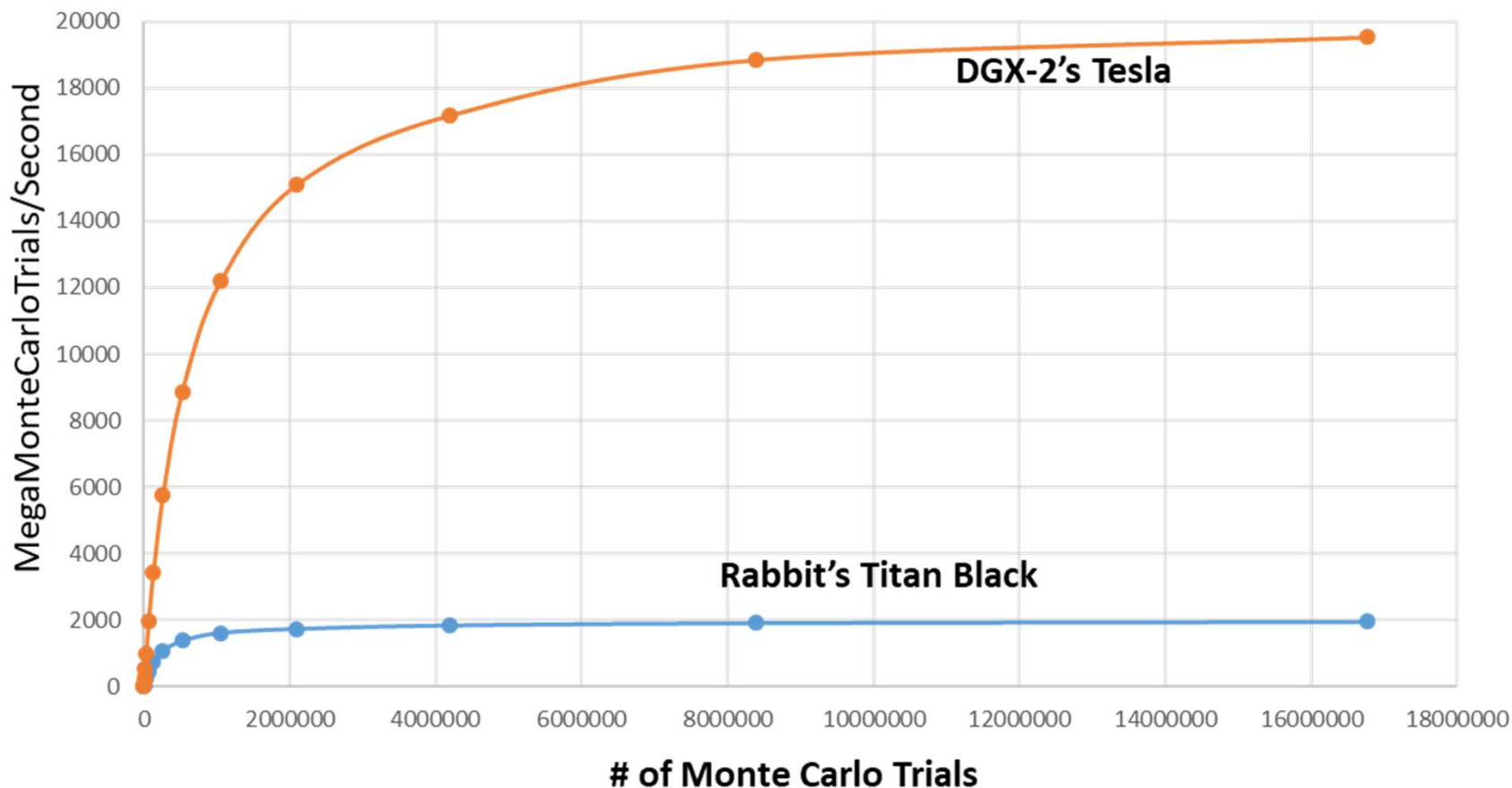
- Has 16 Nvidia Tesla V100 GPUs
- Has 28TB of disk, all SSD
- Has two 24-core Intel Xeon 8168 Platinum 2.7GHz CPUs
- Has 1.5TB of DDR4-2666 System Memory
- Runs the CentOS 7 Linux operating system

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



DGX2 vs. Rabbit for Monte Carlo Calculations



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.



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

flip3 151% ssh submit-c.hpc.engr.oregonstate.edu

submit-c 142% module load slurm ← Type this right away to set your path correctly



Check on the queues

submit-c 143% squeue

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
3923	mime4	c_only	jayasurw	R	1-10:32:19	1	compute-e-1
3963	mime4	2Dex	jayasurw	R	16:21:03	1	compute-e-2
3876	share	CH3COOH_	chukwuk	R	1-23:36:45	1	compute-2-6
3971	nerhp	tcsh	dionnec	R	8:59:45	1	compute-h-8
3881	dgx2	bash	heli	R	1-22:50:44	1	compute-dgx2-1
3965	dgx2	bash	chenju3	R	13:47:36	1	compute-dgx2-4
3645	dgx2	bash	mishrash	R	5-16:48:09	1	compute-dgx2-5
3585	dgx2	bash	azieren	R	6-17:34:00	1	compute-dgx2-3
3583	dgx2	bash	azieren	R	6-18:26:44	1	compute-dgx2-3

System Information

submit-c 144% sinfo

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	1	mix	compute-dgx2-1
sharegpu	up	7-00:00:00	3	idle	compute-dgx2-[2-3], compute-gpu
dgx2	up	7-00:00:00	1	drain	compute-dgx2-2
dgx2	up	7-00:00:00	5	mix	compute-dgx2-[1,3-6]
gpu	up	7-00:00:00	2	mix	compute-gpu[3-4]
gpu	up	7-00:00:00	1	idle	compute-gpu2
gpu	up	7-00:00:00	1	down	compute-gpu1
dgx	up	7-00:00:00	3	mix	compute-dgx2-[4-6]
dgxs	up	7-00:00:00	1	mix	compute-dgx2-1
dgxs	up	7-00:00:00	2	idle	compute-dgx2-[2-3]
class	up	1:00:00	1	mix	compute-dgx2-1
class	up	1:00:00	2	idle	compute-dgx2-[2-3]
eecs	up	7-00:00:00	1	mix	compute-2-6

Submitting a Batch CUDA job to the DGX System

Create a bash shell file that looks like this

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

submit.bash:

```
#!/bin/bash
#SBATCH -J MonteCarlo
#SBATCH -A cs475-575
#SBATCH -p classgputest
#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
```

Your Job Name (makes it easier to find in the queue)

Our class account

This is the partition name that we use for our class when **debugging and testing**.

Double dash

These 2 lines are bash code

```
{ /usr/local/apps/cuda/11.7/bin/nvcc -o montecarlo montecarlo.cu
./montecarlo }
```

submit-c 143% sbatch submit.bash
Submitted batch job 474

Submit the job described in your bash file

submit-c 144% cat montecarlo.err

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



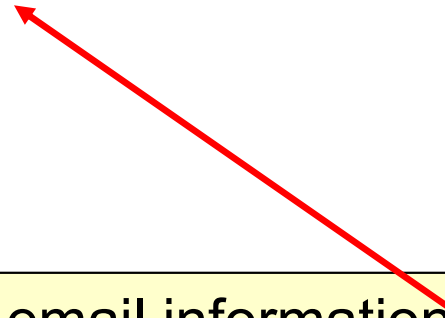
What is the Difference Between the Partitions *classgptest* and *classgpufinal*?

classgptest 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 debugging and testing, 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.



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



You don't have to ask the system to email information to you, but if you do, *please be sure you get your own email address right!*

Our IT people are getting *really* tired of fielding the bounced emails when people misspell their own email address.



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

From	Subject ▼
Slurm workload manager	Slurm Job_id=3980 Name=MatrixMul Ended, Run time 00:00:12, COMPLETED, ExitCode 0 2
Slurm workload manager	Slurm Job_id=3980 Name=MatrixMul Began, Queued time 00:00:01 1



submitloop.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
```

These lines are bash code

```
for t in 2048 8192 131072 2097152
do
  for b in 8 16 32 64 128 256
  do
    /usr/local/apps/cuda/11.7/bin/nvcc -DNUMTRIALS=$t -DBLOCKSIZE=$b -o montecarlo montecarlo.cu
    ./montecarlo
  done
done
```

submit-c 153% sbatch 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.



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**



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

{
module load cuda/10.1
g++ -o printinfo printinfo.cpp /usr/local/apps/cuda/11.7/lib64/libOpenCL.so.1.1 -lm -fopenmp
./printinfo
}
```



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**

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

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