

# Compiling and running on ThetaGPU

Kyle Gerard Felker, CPS Catalyst team

[felker@anl.gov](mailto:felker@anl.gov)

# Outline

- <https://www.alcf.anl.gov/user-guides>
- Compiling and Running on ThetaGPU (DGX A100)
  - System Overview
  - Queuing and running jobs with `qsub` & `mpirun`
  - Software & Lmod Environment Modules
  - Building your code
  - Key tips throughout
- **JaeHyuk Kwack**: Performance Profiling Tools
- **Corey Adams**: AI and Frameworks
- Q&A



See also: Chris Knight's "Getting Started on ThetaGPU" talk from ALCF Computational Performance Workshop 2021-05-04

<https://www.alcf.anl.gov/events/2021-alcf-computational-performance-workshop>

# Getting Started on ALCF Resources

- General documentation
  - ThetaKNL: <https://www.alcf.anl.gov/support-center/theta/theta-thetagpu-overview>
  - ThetaGPU: <https://www.alcf.anl.gov/support-center/theta-gpu-nodes>
  - Polaris: <https://www.alcf.anl.gov/polaris>
  - AI-Testbed: <https://ai.alcf.anl.gov/>
- Cobalt job scheduler documentation: <https://www.alcf.anl.gov/support-center/theta/running-jobs-and-submission-scripts>
- qsub is the primary Cobalt command for job submission
  - -q [job-queue] is either single-gpu or full-node
  - -A [project] is the allocation on ThetaGPU that the job is charged to
  - -t [wallclock time] sets the time limit for the job
  - -n [nodecount] specifies the number of nodes for the job

```
qsub -t 60 -n 1 -q single-gpu -A allocation -M user@anl.gov -I
```

- Home
- Technologies
- Sectors
- AI/ML/DL
- Exascale
- Specials
- Resource Library
- Podcast**
- Events
- Job Bank
- About
- Solution Channels



## Argonne Augments Theta Supercomputer with GPUs to Accelerate Coronavirus Research

December 3, 2020

Dec. 3, 2020 — Earlier this year, the U.S. Department of Energy's (DOE) Argonne National Laboratory unveiled an upgrade to its high-performance computing (HPC) resource Theta. The new hardware stands to substantially improve the performance and capability of the Theta supercomputer, an Intel-HPE/Cray XC40 system housed at the Argonne Leadership Computing Facility (ALCF). The ALCF is a U.S. DOE Office of Science User Facility.

Deployed rapidly in response to the global pandemic, the Theta upgrade and its supporting systems are currently being leveraged in the fight against the coronavirus and associated COVID-19 disease. Funding for the hardware was provided by the Coronavirus Aid, Relief and Economic Security (CARES) Act, signed into law in March.

With a core that combines both graphics processing unit (GPU) and central processing unit (CPU) capabilities, the augmented Theta architecture adds 24 NVIDIA DGX A100 nodes to the existing system. Each DGX A100 node comprises eight NVIDIA A100 Tensor Core GPUs and two AMD Rome CPUs that provide 320 gigabytes (7680 GB aggregately) of GPU memory for training artificial intelligence (AI) datasets, while also enabling GPU-specific and GPU-enhanced HPC applications for modeling and simulation.

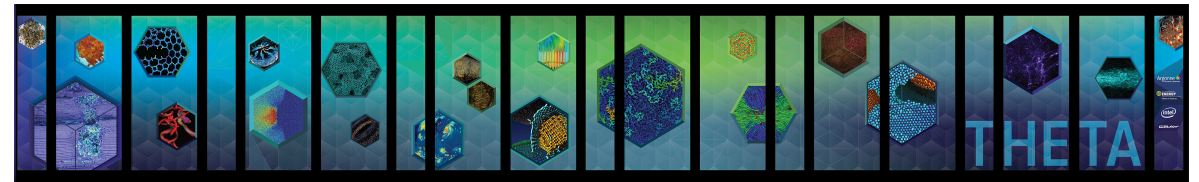
For system documentation, visit the [Theta User Guide](#)

To be considered for an allocation award, please submit your request using the ALCF's [Discretionary Allocation Request form](#).



ALCF staff members augment Theta with the installation of NVIDIA DGX A100 nodes. (Image: Argonne National Laboratory)

# ThetaGPU



- <https://www.alcf.anl.gov/support-center/theta/theta-thetagpu-overview>
- **ThetaGPU** = expansion of Theta originally to support coronavirus research (now open for general use)
- NVIDIA DGX A100 partition
  - 24 nodes each with
    - 8 NVIDIA A100 Tensor Core GPUs & 320 GB HBM memory
    - 2 AMD Rome 64-core CPUs & 1 TB DDR4
    - 15 TB SSD (4 x 3.84 TB), 25 Gb/s bandwidth
    - 8 HDR 200 NICs (compute network)
    - 2 HDR 200 NICs (storage network)
- Dedicated Compute Fabric
  - 20 Mellanox QM9700 HDR200 40-port switches in fat-tree topology
- Project filesystem is Theta's 10 PB Lustre with 240 GB/s throughput (/lus/theta-fs0/projects/)
  - Eagle (/lus/eagle/projects/): 100 PB Lustre with 650 GB/s throughput
  - Grand (/lus/grand/projects/): 100 PB Lustre with 650 GB/s throughput



# NVIDIA DGX versions (per node)

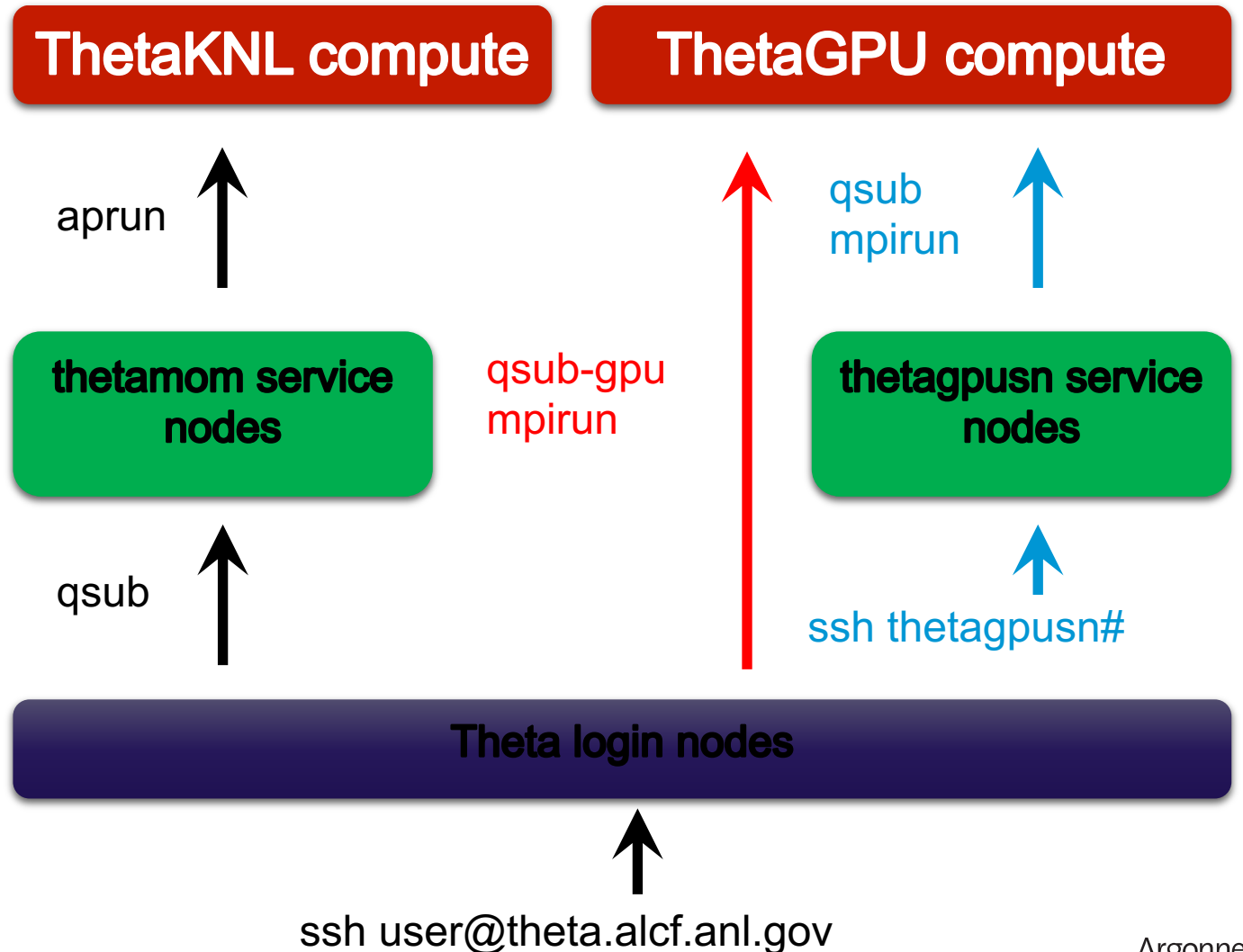
- DGX-1 (April 2016)
  - 8x P100 or V100 (first gen 16 GB) GPUs
  - 2x 20-core Intel Xeon E5-2698 v4 CPUs
- DGX-2 (May 2018)
  - 16x V100 (second gen 32 GB) GPUs
  - 2x 24-core Intel Xeon 8168 CPUs
- DGX A100 (May 2020)
  - 8x A100 (40 GB) GPUs
  - 2x 64-core AMD Rome 7742 CPUs



# ThetaGPU - Logging in and navigating

<https://www.alcf.anl.gov/support-center/theta/theta-thetagpu-overview>

- First, access ThetaKNL login nodes  
`$ ssh user@theta.alcf.anl.gov`
- **ThetaKNL**: launch job on service nodes, then run on compute nodes  
`$ qsub -l -n 4 -t 60 -q debug-cache-quad -A ...`  
`$ aprun -n 4096 -N 32 -d 4 -j 2 -cc depth ...`
- **ThetaGPU**: login to service nodes, then build and run on compute nodes  
`$ ssh thetagpusn1`  
`$ qsub -l -n 1 -t 60 -q full-node -A ...`  
`$ mpirun ...`



# Theta - Logging in and navigating

- Alternatively, you can stay on ThetaKNL login node and load/unload ThetaGPU scheduler as needed:

```
$ module load cobalt/cobalt-gpu
```



**hostname is your friend!**

# SSH onto the same Theta login node each time (e.g. for running a persistent `tmux` server)

On your laptop, `~/.ssh/config`:

```
Host theta
HostName thetaloginX.alcf.anl.gov
User username
ControlMaster auto
ControlPath ~/.ssh/cm_socket/%r@%h:%p
ControlPersist 10m
```

Where X is in [1, 6]. Then

```
ssh theta
```

# ThetaGPU - Submitting Script Jobs

- <https://www.alcf.anl.gov/support-center/theta-gpu-nodes/running-jobs-thetagpu>
- Executable is invoked within script (bash, csh, ...)

- mpirun is used to launch executables on compute nodes

- ```
> cat myscript.sh
```
- ```
#!/bin/sh
```
- ```
#COBALT -n 2 -t 15 -q full-node -A <project_name>
```
- ```
#COBALT --attrs pubnet
```
- ```
echo "Starting Cobalt job script"
```
- ```
mpirun -hostfile ${COBALT_NODEFILE} -n 16 -N 8 <app> <app_args>
```

Cobalt Options

MPI Ranks

Ranks per node

- ```
> qsub ./myscript.sh
```

```
123456
```

# ThetaGPU - mpirun Overview

- <https://www.alcf.anl.gov/support-center/theta-gpu-nodes/running-jobs-thetagpu>
- mpirun options
  - Total number of MPI ranks: `-n <total_number_ranks>`
  - Number of MPI ranks per node: `-N <number_ranks_per_node>`
  - Environment variables: `-x <VAR1=1> -x <VAR2=1>`
  - Display MPI process map: `-display-map`
  - Display detected resource allocation: `-display-allocation`
  - Process binding: `--bind-to <hwthread|core|socket|...>`
- Environment settings you may need
  - `-x OMP_NUM_THREADS=<num_threads>`
- See also `man mpirun`

# ThetaGPU - Queues

- <https://www.alcf.anl.gov/support-center/theta-gpu-nodes/queue-policy-thetagpu>
- Two queues currently available with simple First-In First-Out (FIFO) policy
  - full-node: request entire node
  - single-gpu: request single gpu; node resources shared by other users
    - analogous to debug queue to build applications and debug

| queue      | full-node | single-gpu |
|------------|-----------|------------|
| MinTime    | 5 minutes | 5 minutes  |
| MaxTime    | 12 hours  | 1 hour     |
| MaxQueued  | 20 jobs   | 1 job      |
| MaxRunning | 10 jobs   | 1 job      |

- [Check website for current policies](#)

# ThetaGPU - qsub attributes and networking quirks

- HTTP/S networking disabled by default on compute nodes; use a proxy

```
export http_proxy=theta-proxy.tmi.alcf.anl.gov:3128
export https_proxy=theta-proxy.tmi.alcf.anl.gov:3128
```
- SSH port 22 blocked on both service and compute nodes
  - cannot clone Git repos over SSH
- Enable Multi-Instance GPU (MIG) mode (**currently disabled and waiting for response from NVIDIA**)

```
--attrs mig-mode=True
```
- Enable high performance networking on compute nodes (typically over ESNet)

```
--attrs=pubnet
```

# Clone/push/pull via Git SSH URLs on `thetaloginX`

e.g. `tmux` running on `thetaloginX` with windows:

1. on `thetaloginX` for networking over SSH
2. on `thetagpusn1` for launching jobs, inspecting modules
3. on `thetagpuY` (compute node) for running job

# ThetaGPU – Lmod Modules



<https://lmod.readthedocs.io/en/latest/>

- A tool for managing a user's environment
  - Sets your PATH, LD\_LIBRARY\_PATH, shell functions, etc. to access desired front-end tools
  - Your TensorFlow, PyTorch, Horovod, MPI library, compiler version, etc. can be changed here
- Module commands
  - List available module commands: `module help`
  - List currently loaded modules: `module list`
  - List all available modules: `module avail`
  - Add module to environment: `module load <mod>`
  - Remove module from environment: `module unload <mod>`
  - Swap loaded module with new one: `module swap <mod_old> <mod_new>`
  - List information about module: `module show <mod>`
  - Include additional modules: `module use <path_to_extra_modules>`



```
→ ~ module avail
```

```
----- /usr/local/lmod/lmod/modulefiles -----  
Core/lmod      Core/settag  
----- /lus/theta-fs0/software/environment/thetagpu/lmod/modulefiles -----  
Core/StdEnv      (L,D)  hpctoolkit/2021.04.30-gpu      nvhpc-nompi/21.3  
aocl/blis-3.0    hypre/2.22.0                  nvhpc-nompi/21.7      (D)  
cmake/3.19.5     llvm/main-20210811           nvhpc/20.9  
conda/pytorch    nccl/nccl-v2.8.4-1_CUDA11     nvhpc/21.2  
conda/tensorflow nccl/nccl-v2.9.9-1_CUDA11.3   nvhpc/21.3  
conda/2021-06-26 (D)    nccl/nccl-v2.11.4-1_CUDA11.4 (D)  nvhpc/21.7      (D)  
conda/2021-06-28 netcdf/c-4.8.0-fortran-4.5.3   openmpi/openmpi-4.0.5_ucx-1.10.0_nvhpc-21.7  
conda/2021-09-22 nvhpc-byo-compiler/20.9       openmpi/openmpi-4.0.5      (L)  
conda/2021-11-30 nvhpc-byo-compiler/21.2       openmpi/openmpi-4.1.0_ucx-1.10.0  
darshan/3.3.0    nvhpc-byo-compiler/21.3       openmpi/openmpi-4.1.0_ucx-1.11.0_gcc-9.3.0  
hdf5/1.8.13      nvhpc-byo-compiler/21.7      (D)  openmpi/openmpi-4.1.0  
hdf5/1.8.22      nvhpc-mpi/21.7                openmpi/openmpi-4.1.1_ucx-1.10.1_gcc-9.3.0  
hdf5/1.12.0      (D)    nvhpc-nompi/20.9              openmpi/openmpi-4.1.1_ucx-1.11.2_gcc-9.3.0 (D)  
hdf5/1.12.1-nvhpc nvhpc-nompi/21.2  
  
----- /lus/theta-fs0/software/spack/share/spack/modules/linux-ubuntu20.04-x86_64 -----  
autoconf-2.69-gcc-9.3.0-qz4d4gi  libtool-2.4.6-gcc-9.3.0-uh3mpsu  
berkeley-db-18.1.40-gcc-9.3.0-pn2cxag  m4-1.4.18-gcc-9.3.0-eixehd4  
bison-3.7.6-gcc-9.3.0-b3ikzdr  m4-1.4.19-gcc-9.3.0-7fztfyz  
bison-3.7.6-gcc-9.3.0-d4ucav2  ncurses-6.2-gcc-9.3.0-n5vhymf  
clingo-bootstrap-spack-gcc-9.3.0-gbsyvaq  openssl-1.1.1k-gcc-9.3.0-rvjrbxt  
clingo-bootstrap-spack-gcc-9.3.0-uwngfyx  perl-5.32.1-gcc-9.3.0-ylpdp6  
cmake-3.20.2-gcc-9.3.0-r5aseli  perl-5.34.0-gcc-9.3.0-aa6w4wx  
cmake-3.20.3-gcc-9.3.0-57eqw4f  pkgconf-1.7.4-gcc-9.3.0-newgzwq  
diffutils-3.7-gcc-9.3.0-2eqr3yc  re2c-1.2.1-gcc-9.3.0-drh4oxu  
gdbm-1.19-gcc-9.3.0-g4xrv3g  readline-8.1-gcc-9.3.0-fvag4vk  
libiconv-1.16-gcc-9.3.0-w6zptbc  zlib-1.2.11-gcc-9.3.0-p7dmb5p  
libsigsegv-2.13-gcc-9.3.0-2fxv3ky  
  
Where:  
L: Module is loaded  
D: Default Module  
  
Use "module spider" to find all possible modules and extensions.  
Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".
```

# Current CUDA driver (2022-01-26)

- CUDA 11.4 is default toolkit/runtime and device driver version

```
Enter passphrase for /home/pecker/.ssh/ed_rsa:
→ ~ nvidia-smi
Wed Jan 26 00:17:51 2022
+-----+
| NVIDIA-SMI 470.82.01      Driver Version: 470.82.01      CUDA Version: 11.4      |
+-----+-----+-----+-----+-----+-----+
| GPU   Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|====+=====+====+=====+=====+=====+
0   NVIDIA A100-SXM...  On	00000000:07:00.0 Off	0
N/A   22C    P0      51W / 400W	0MiB / 40536MiB	0%      Default
		Disabled
+-----+-----+-----+-----+-----+-----+

```

- CUDA 11.3 and 11.0 libraries still available (<https://docs.nvidia.com/deploy/cuda-compatibility/>)

```
→ local hostname
thetagpu11
→ local pwd
/usr/local
→ local ll
total 56K
drwxr-xr-x  2 root root 4.0K Jan 18 16:32 bin/
lrwxrwxrwx  1 root root  22 Aug  9 09:58 cuda -> /etc/alternatives/cuda/
lrwxrwxrwx  1 root root  25 Jun 24  2021 cuda-11 -> /etc/alternatives/cuda-11/
drwxr-xr-x 15 root root 4.0K Jul 12  2021 cuda-11.0/
drwxr-xr-x 15 root root 4.0K Jun 24  2021 cuda-11.3/
drwxr-xr-x 16 root root 4.0K Nov  8 15:11 cuda-11.4/
drwxr-xr-x  4 root root 4.0K Nov  8 14:39 dcgm/

```

# Monitor your running job's GPU utilization with `nvidia-smi`

```
qstat -u username -f  
# find compute node name thetagpuX  
ssh thetagpuX  
watch -n 1 nvidia-smi
```

# ThetaGPU - GPU Assignment

- <https://www.alcf.anl.gov/support-center/theta-gpu-nodes/gpu-monitoring>
- Map processes to GPUs on each node
- Programming model and framework semantics (CUDA, Tensorflow, etc...)
  - `MPI_Comm_rank(MPI_COMM_WORLD, &me)`
  - `cudaGetDeviceCount(&num_devices);`
  - `cudaSetDevice(me % num_devices);`
- Environment variables (e.g. in helper scripts)
  - `export CUDA_VISIBLE_DEVICES=4`

# ThetaGPU - GNU Compilers

- <https://www.alcf.anl.gov/support-center/theta-gpu-nodes/compiling-and-linking-thetagpu>
- GCC w/ OpenMPI
  - Default environment
- GPU Programming Models: CUDA, OpenCL
- Use C/C++ wrappers: mpicxx, mpicc
- Fortran is not supported (gfortran not installed)

# ThetaGPU - NVIDIA Compilers

- <https://www.alcf.anl.gov/support-center/theta-gpu-nodes/compiling-and-linking-thetagpu>

- NVIDIA HPC SDK

- Load module

```
$ module use /lus/theta-fs0/software/environment/thetagpu/lmod/tmp
```

```
$ module swap openmpi openmpi-4.1.0_nvhpc-21.3
```

```
$ module list
```

Currently Loaded Modules:

```
1) Core/StdEnv 2) nvhpc-nompi/21.3 3) openmpi-4.1.0_nvhpc-21.3
```

- GPU Programming Models: CUDA, OpenCL, OpenACC, OpenMP

- Use pgc++, pgcc, pgf90, etc...

- Use mpicxx, mpicxx, mpif90, etc...

<https://developer.nvidia.com/hpc-sdk>

# ThetaGPU - NVIDIA Compilers

- <https://www.alcf.anl.gov/support-center/theta-gpu-nodes/compiling-and-linking-thetagpu>
- NVIDIA HPC SDK modules
  - Adds NVIDIA SDK compilers, libraries, and tools to paths

```
software/environment/thetagpu/lmod/modulefiles -----  
  llvm/release-12.0.0      (D)    nvhpc-nompi/21.3  
  nccl/nccl-v2.8.4-1_CUDA11  (D)    nvhpc/20.9          (D)  
  nvhpc-byo-compiler/20.9    (D)    nvhpc/21.2  
  nvhpc-byo-compiler/21.2    (D)    nvhpc/21.3  
  nvhpc-byo-compiler/21.3    (D)    openmpi/openmpi-4.0.5 (L)  
  nvhpc-nompi/20.9           (D)    openmpi/openmpi-4.1.0_ucx-1.10.0  
  nvhpc-nompi/21.2           (D)    openmpi/openmpi-4.1.0 (D)
```

- nvhpc: adds all NVIDIA SDK compilers, libraries, and tools to paths
- nvhpc-byo-compiler: identical to nvhpc, but doesn't set compiler environment variables
- nvhpc-nompi: excludes MPI libraries
  - Preferred module
  - Important to use ALCF provided OpenMPI modules for multi-node runs

<https://developer.nvidia.com/hpc-sdk>

# ThetaGPU - LLVM Compilers

- <https://www.alcf.anl.gov/support-center/theta-gpu-nodes/compiling-and-linking-thetagpu>

- LLVM w/ OpenMP offload

- Load module

```
$ module load llvm
```

```
$ module list
```

```
Currently Loaded Modules:
```

```
1) openmpi/openmpi-4.0.5 2) Core/StdEnv 3) llvm/release-12.0.0
```

- GPU Programming Models: CUDA, OpenCL, OpenMP

- Use clang, clang++

- Use mpicxx, mpicxx



**Build your software on a compute node  
(e.g. a single-gpu allocation), not on a service  
node!**

**Thank you!**

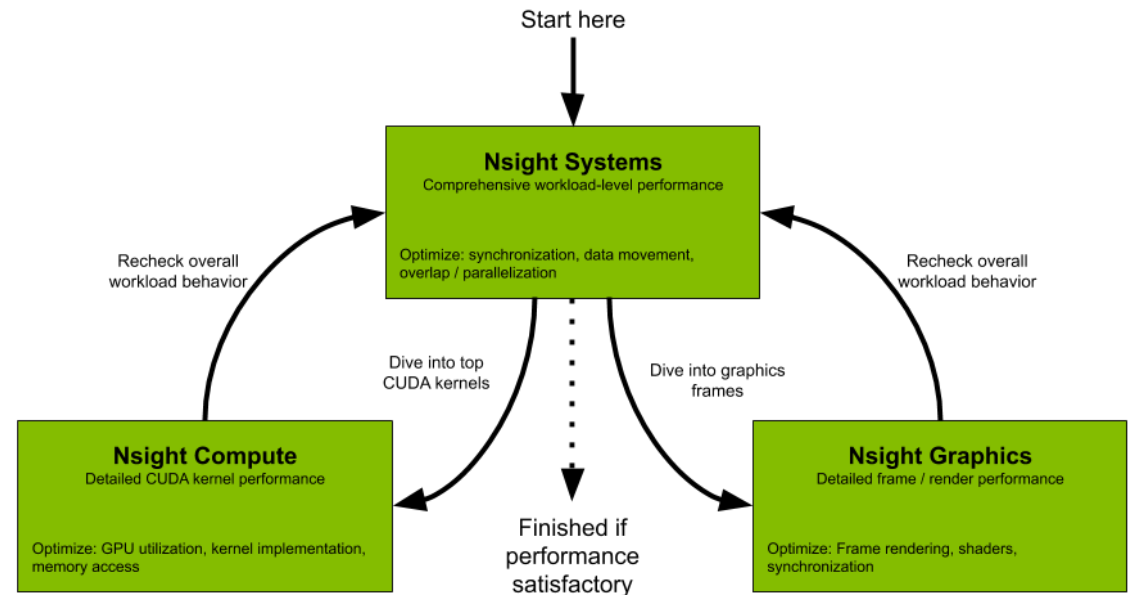
# Performance Profiling Tools

JaeHyuk Kwack, ALCF Perf. Engr. Group

[jkwack@anl.gov](mailto:jkwack@anl.gov)

# NVIDIA Tools overview

- Nsight Systems
  - A system-wide visualization of an application performance
  - To optimize bottlenecks to scale efficiently across CPUs and GPUs on ThetaGPU
- Nsight Compute
  - An interactive kernel profiler for applications
  - Providing detailed performance metrics and API debugging via a user interface and command line tool
  - Providing customizable and data-driven user interface and metric collection that can be extended with analysis scripts for post-processing results
- Nsight Graphics
  - A stand-alone tool for graphics applications



Credit: NVIDIA (<https://developer.nvidia.com/tools-overview>)

# Step-by-step guide on ThetaGPU (1/2)

- Common part on ThetaGPU

- Build your application for ThetaGPU

- Submit your job script to ThetaGPU or start an interactive job mode on ThetaGPU as follows:

- ```
$ module load cobalt/cobalt-gpu
```

- ```
$ qsub -I -n 1 -t 30 -q full-node -A {your_project}
```

- Nsight Systems

- Run your application with Nsight Systems as follows:

- ```
$ nsys profile -o {output_filename} --stats=true ./{your_application}
```

- Nsight Compute

- Run your application with Nsight Compute

- ```
$ ncu --set detailed -k {kernel_name} -o {output_filename} ./{your_application}
```

- Remark: W/o -o option, Nsight Compute provides performance data as a standard output

# Step-by-step guide on ThetaGPU (2/2)

- Post-processing the profiled data

- Post-processing via CLI

- `$ nsys stats {output_filename}.qdrep`

- `$ ncu -i {output_filename}.ncu-rep`

- Post-processing on your local system via GUI

- Install NVIDIA Nsight Systems and NVIDIA Nsight Compute after downloading both of them from the [NVIDIA Developer Zone](#).
    - Download nsys output files (i.e., ending with .qdrep and .sqlite) to your local system, and then open them with NVIDIA Nsight Systems on your local system.
    - Download ncu output files (i.e., ending with .ncu-rep) to your local system, and then open them with NVIDIA Nsight Compute on your local system.

- More options for performance analysis with Nsight Systems and Nsight Compute

- `$ nsys --help`

- `$ ncu --help`

# Nsight Systems Demo with a simple example

```
jkwack@thetagpu18:~/HPC_benchmarks/BabelStream/JK_thetaGPU$ nsys profile -o JKreport-nsys-BableStream --stats=true ./cuda-stream
```

```
...
```

```
Generating CUDA API Statistics...
```

```
CUDA API Statistics (nanoseconds)
```

| Time(%) | Total Time | Calls | Average    | Minimum | Maximum   | Name                  |
|---------|------------|-------|------------|---------|-----------|-----------------------|
| 44.8    | 280504347  | 4     | 70126086.8 | 1050249 | 276881346 | cudaMalloc            |
| 31.4    | 196878210  | 401   | 490968.1   | 381542  | 600948    | cudaDeviceSynchronize |
| 22.4    | 140280462  | 103   | 1361946.2  | 436597  | 32339232  | cudaMemcpy            |
| 1.0     | 6263864    | 4     | 1565966.0  | 1236542 | 1884610   | cudaFree              |
| 0.4     | 2729558    | 501   | 5448.2     | 4970    | 36269     | cudaLaunchKernel      |

```
Generating CUDA Kernel Statistics...
```

```
CUDA Kernel Statistics (nanoseconds)
```

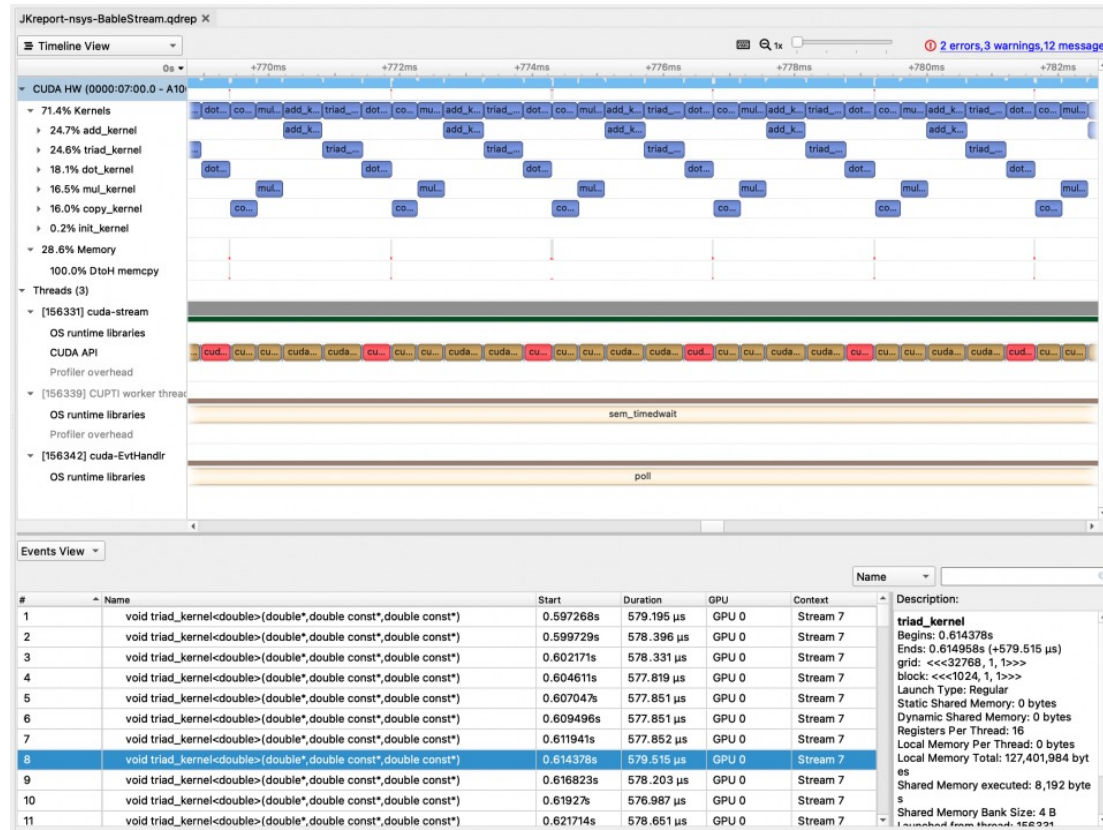
| Time(%) | Total Time | Instances | Average  | Minimum | Maximum | Name                                                                        |
|---------|------------|-----------|----------|---------|---------|-----------------------------------------------------------------------------|
| 24.7    | 58518170   | 100       | 585181.7 | 580347  | 594395  | void add_kernel<double>(double const*, double const*, double*)              |
| 24.6    | 58312184   | 100       | 583121.8 | 576987  | 595067  | void triad_kernel<double>(double*, double const*, double const*)            |
| 18.1    | 42942748   | 100       | 429427.5 | 419548  | 438333  | void dot_kernel<double>(double const*, double const*, double*, int)         |
| 16.5    | 39062588   | 100       | 390625.9 | 388733  | 392125  | void mul_kernel<double>(double*, double const*)                             |
| 16.0    | 37980930   | 100       | 379809.3 | 376541  | 392925  | void copy_kernel<double>(double const*, double*)                            |
| 0.2     | 521628     | 1         | 521628.0 | 521628  | 521628  | void init_kernel<double>(double*, double*, double*, double, double, double) |

```
...
```

```
Report file moved to "/gpfs/mira-home/jkwack/HPC_benchmarks/BabelStream/JK_thetaGPU/JKreport-nsys-BableStream.qdrep"
```

```
Report file moved to "/gpfs/mira-home/jkwack/HPC_benchmarks/BabelStream/JK_thetaGPU/JKreport-nsys-BableStream.sqlite"
```

# Nsight Systems Demo with a simple example





# Nsight Compute Demo with a simple example

```
jkwack@thetagpu18:~/HPC_benchmarks/BabelStream/JK_thetaGPU$ ncu --set detailed -k triad_kernel -o JKreport-ncu_detailed-triad_kernel-BableStream ./cuda-stream
BabelStream
Version: 3.4
Implementation: CUDA
Running kernels 100 times
Precision: double
Array size: 268.4 MB (=0.3 GB)
Total size: 805.3 MB (=0.8 GB)
==PROF== Connected to process 166971 (/gpfs/mira-home/jkwack/HPC_benchmarks/BabelStream/JK_thetaGPU/cuda-stream)
Using CUDA device A100-SXM4-40GB
Driver: 11000
==PROF== Profiling "triad_kernel": 0%...50%...100% - 19 passes
==PROF== Profiling "triad_kernel": 0%...50%...100% - 19 passes
==PROF== Profiling "triad_kernel": 0%...50%...100% - 19 passes
...
==PROF== Profiling "triad_kernel": 0%...50%...100% - 19 passes

Function    MBytes/sec  Min (sec)  Max      Average
Copy        1336793.345 0.00040    0.00042  0.00041
Mul         1307948.274 0.00041    0.00043  0.00042
Add         1335561.797 0.00060    0.00062  0.00062
Triad       976.089     0.82503    1.00961  0.87930
Dot         1081921.148 0.00050    0.00055  0.00053

==PROF== Disconnected from process 166971
==PROF== Report: /gpfs/mira-home/jkwack/HPC_benchmarks/BabelStream/JK_thetaGPU/JKreport-ncu_detailed-triad_kernel-BableStream.ncu-rep
```

# Nsight Compute Demo

Page: Details Launch: 0 - 867 - triad\_kernel Add Baseline Apply Rules Copy as Image

Current 867 ... Time: 572.19 usecond Cycles: 622,968 Regs: 16 GPU: A100-SXM4-40GB SM Frequency: 1.09 cycle/nsecond CC: 8.0 Process: [166971] cuda-stream

### GPU Speed Of Light

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

|                      |       |                                |           |
|----------------------|-------|--------------------------------|-----------|
| SOL SM [%]           | 6.23  | Duration [usecond]             | 572.19    |
| SOL Memory [%]       | 89.75 | Elapsed Cycles [cycle]         | 622968    |
| SOL L1/TEX Cache [%] | 15.70 | SM Active Cycles [cycle]       | 618515.41 |
| SOL L2 Cache [%]     | 69.01 | SM Frequency [cycle/nsecond]   | 1.09      |
| SOL DRAM [%]         | 89.75 | DRAM Frequency [cycle/nsecond] | 1.21      |

### Compute Workload Analysis

Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.

|                                   |      |                      |      |
|-----------------------------------|------|----------------------|------|
| Executed Tpc Elapsed [inst/cycle] | 0.22 | SM Busy [%]          | 5.50 |
| Executed Tpc Active [inst/cycle]  | 0.22 | Issue Slots Busy [%] | 5.50 |
| Issued Tpc Active [inst/cycle]    | 0.22 |                      |      |

### Memory Workload Analysis

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy).

|                                  |       |                      |       |
|----------------------------------|-------|----------------------|-------|
| Memory Throughput [Tbyte/second] | 1.39  | Mem Busy [%]         | 51.96 |
| L1/TEX Hit Rate [%]              | 0     | Max Bandwidth [%]    | 89.75 |
| L2 Hit Rate [%]                  | 50.06 | Mem Pipes Busy [%]   | 6.23  |
| L2 Compression Success Rate [%]  | 0     | L2 Compression Ratio | 0     |

### Scheduler Statistics

Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

|                                     |       |                                                 |       |
|-------------------------------------|-------|-------------------------------------------------|-------|
| Active Warps Per Scheduler [warp]   | 13.95 | Instructions Per Active Issue Slot [inst/cycle] | 1     |
| Eligible Warps Per Scheduler [warp] | 0.12  | No Eligible [%]                                 | 94.49 |
| Issued Warp Per Scheduler           | 0.06  | One or More Eligible [%]                        | 5.51  |

### Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.

|                                              |        |                                          |    |
|----------------------------------------------|--------|------------------------------------------|----|
| Warp Cycles Per Issued Instruction [cycle]   | 253.15 | Avg. Active Threads Per Warp             | 32 |
| Warp Cycles Per Issue Active [warp]          | 253.15 | Avg. Not Predicated Off Threads Per Warp | 32 |
| Warp Cycles Per Executed Instruction [cycle] | 253.61 |                                          |    |

### GPU Speed Of Light

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

|                      |       |                                |           |
|----------------------|-------|--------------------------------|-----------|
| SOL SM [%]           | 6.23  | Duration [usecond]             | 572.19    |
| SOL Memory [%]       | 89.75 | Elapsed Cycles [cycle]         | 622968    |
| SOL L1/TEX Cache [%] | 15.70 | SM Active Cycles [cycle]       | 618515.41 |
| SOL L2 Cache [%]     | 69.01 | SM Frequency [cycle/nsecond]   | 1.09      |
| SOL DRAM [%]         | 89.75 | DRAM Frequency [cycle/nsecond] | 1.21      |

### GPU Utilization

|            |       |
|------------|-------|
| SM [%]     | 6.23  |
| Memory [%] | 89.75 |

### SOL SM Breakdown

|                                                |      |
|------------------------------------------------|------|
| SOL SM: Inst Executed Pipe Lsu [%]             | 6.23 |
| SOL SM: Issue Active [%]                       | 5.47 |
| SOL SM: Inst Executed [%]                      | 5.48 |
| SOL SM: MioZ1 Writeback Active [%]             | 4.68 |
| SOL SM: Mlo Inst Issued [%]                    | 3.90 |
| SOL SM: Pipe Fma Cycles Active [%]             | 3.12 |
| SOL SM: Mlo Pq Write Cycles Active [%]         | 3.12 |
| SOL SM: Inst Executed Pipe Adu [%]             | 3.12 |
| SOL SM: Mlo Pq Read Cycles Active [%]          | 3.12 |
| SOL SM: Pipe FpB4 Cycles Active [%]            | 1.58 |
| SOL SM: Pipe Shared Cycles Active [%]          | 1.58 |
| SOL SM: Pipe Alu Cycles Active [%]             | 1.58 |
| SOL SM: Inst Executed Pipe Cbu Pred On Any [%] | 1.58 |
| SOL SM: Inst Executed Pipe Uniform [%]         | 0.78 |
| SOL IDC: Request Cycles Active [%]             | 0    |
| SOL SM: Inst Executed Pipe Xu [%]              | 0    |
| SOL SM: Inst Executed Pipe Tex [%]             | 0    |
| SOL SM: Inst Executed Pipe Isa [%]             | 0    |
| SOL SM: Inst Executed Pipe Fp16 [%]            | 0    |
| SOL SM: Pipe Tensor Cycles Active [%]          | 0    |

### SOL Memory Breakdown

|                                            |       |
|--------------------------------------------|-------|
| SOL GPU: Dram Throughput [%]               | 89.75 |
| SOL L2: T Sectors [%]                      | 51.96 |
| SOL L2: D Sectors [%]                      | 36.63 |
| SOL L2: D Sectors Fill Device [%]          | 34.85 |
| SOL L2: T Tag Requests [%]                 | 28.01 |
| SOL L2: XbarZits Cycles Active [%]         | 21.88 |
| SOL L2: Lts2xbar Cycles Active [%]         | 17.34 |
| SOL L1: M L1Tex2xbar Req Cycles Active [%] | 15.69 |
| SOL L1: M XbarZitHex Read Sectors [%]      | 12.47 |
| SOL L1: Data Pipe Lsu Wavefronts [%]       | 12.35 |
| SOL L1: Lsu Writeback Active [%]           | 7.79  |
| SOL L1: LsuIn Requests [%]                 | 6.23  |
| SOL L1: Data Bank Reads [%]                | 3.12  |
| SOL L1: Data Bank Writes [%]               | 3.12  |
| SOL L1: F Wavefronts [%]                   | 0.00  |
| SOL L1: Texin Sm2tex Req Cycles Active [%] | 0.00  |
| SOL L2: D Sectors Fill System [%]          | 0     |
| SOL L1: Data Pipe Tex Wavefronts [%]       | 0     |
| SOL L2: D Atomic Input Cycles Active [%]   | 0     |
| SOL L1: Tex Writeback Active [%]           | 0     |

### Floating Point Operations Roofline

Performance [FLOP/s] (1 = 1e12)

Arithmetic Intensity [FLOP/byte]

### Recommendations

- Bottleneck** The kernel is utilizing greater than 80.0% of the available compute or memory performance of the device. To further improve performance, work will likely need to be shifted from the most utilized to another unit. Start by analyzing workloads in the [Memory Workload Analysis](#) section.
- Roofline Analysis** The ratio of peak float (fp32) to double (fp64) performance on this device is 2:1. The kernel achieved 0% of this device's fp32 peak performance and 2% of its fp64 peak performance.

# Nsight Compute Demo

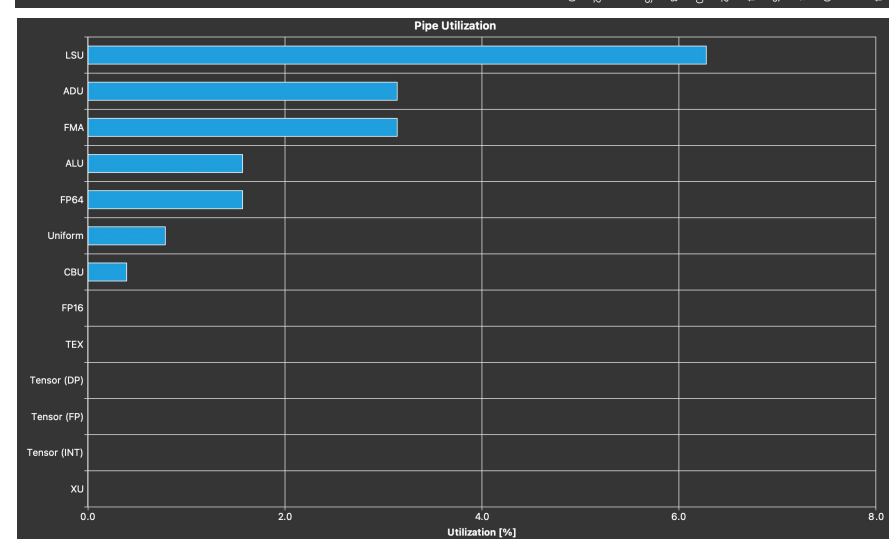
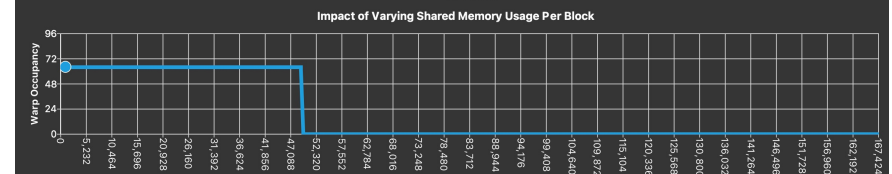
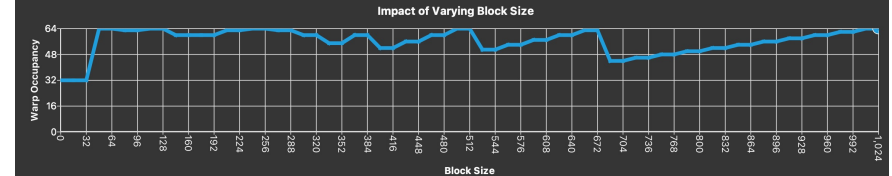
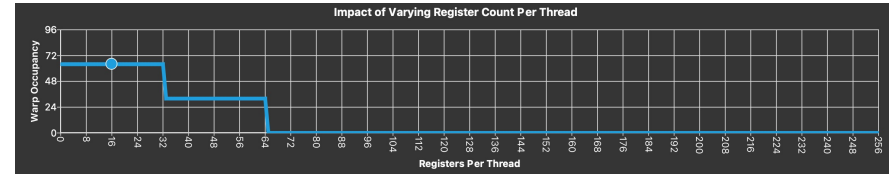
Page: Source Launch: 0 - 867 - triad\_kernel Add Baseline Apply Rules Copy as Image

Current 867 - triad\_ke... Time: 572.19 usecond Cycles: 622,968 Regs: 16 GPU: A100-SXM4-40GB SM Frequency: 1.09 cycle/nsecond CC: 8.0 Process: [166971] cuda-stream

View: SASS

Source: triad\_kernel Find... Navigation: Instructions Executed

| #  | Address          | Source                              | Live Registers | Sampling Data (All) | Sampling Data (Not Issued) | Instructions Executed | Flattened-Thread Executed | Memory Address Space | Memor C |
|----|------------------|-------------------------------------|----------------|---------------------|----------------------------|-----------------------|---------------------------|----------------------|---------|
| 1  | 00007f3b24f9c600 | MOV R1, c[0x0][0x28]                | 1              | 52                  | 37                         | 1,048,576             | 33,554,432                |                      |         |
| 2  | 00007f3b24f9c610 | SZR R8, SR_CTaid.X                  | 2              | 36                  | 10                         | 1,048,576             | 33,554,432                |                      |         |
| 3  | 00007f3b24f9c620 | MOV R9, 0x8                         | 3              | 5                   | 0                          | 1,048,576             | 33,554,432                |                      |         |
| 4  | 00007f3b24f9c630 | ULDC.64 UR4, c[0x0][0x118]          | 3              | 21                  | 8                          | 1,048,576             | 33,554,432                |                      |         |
| 5  | 00007f3b24f9c640 | SZR R3, SR_IID.X                    | 4              | 9                   | 0                          | 1,048,576             | 33,554,432                |                      |         |
| 6  | 00007f3b24f9c650 | IMAD R8, R8, c[0x0][0x0], R3        | 4              | 127                 | 33                         | 1,048,576             | 33,554,432                |                      |         |
| 7  | 00007f3b24f9c660 | IMAD.WIDE R2, R8, R9, c[0x0][0x168] | 5              | 53                  | 7                          | 1,048,576             | 33,554,432                |                      |         |
| 8  | 00007f3b24f9c670 | IMAD.WIDE R4, R8, R9, c[0x0][0x170] | 7              | 186                 | 41                         | 1,048,576             | 33,554,432                |                      |         |
| 9  | 00007f3b24f9c680 | LDG.E.64 R2, [R2.64]                | 7              | 66                  | 18                         | 1,048,576             | 33,554,432                | Global               |         |
| 10 | 00007f3b24f9c690 | LDG.E.64 R4, [R4.64]                | 7              | 69                  | 37                         | 1,048,576             | 33,554,432                | Global               |         |
| 11 | 00007f3b24f9c6a0 | IMAD.WIDE R8, R8, R9, c[0x0][0x160] | 7              | 6                   | 0                          | 1,048,576             | 33,554,432                |                      |         |
| 12 | 00007f3b24f9c6b0 | DFMA R6, R4, c[0x2][0x0], R2        | 9              | 15,400              | 14,674                     | 1,048,576             | 33,554,432                |                      |         |
| 13 | 00007f3b24f9c6c0 | STG.E.64 [R8.64], R6                | 5              | 35                  | 28                         | 1,048,576             | 33,554,432                | Global               |         |
| 14 | 00007f3b24f9c6d0 | EXIT                                | 1              | 6                   | 0                          | 1,048,576             | 33,554,432                |                      |         |
| 15 | 00007f3b24f9c6e0 | BRA 0x7f3b24f9c6e0                  | 0              | 248                 | 215                        |                       |                           |                      |         |
| 16 | 00007f3b24f9c6f0 | NOP                                 | 0              | 0                   | 0                          |                       |                           |                      |         |
| 17 | 00007f3b24f9c700 | NOP                                 | 0              | 0                   | 0                          |                       |                           |                      |         |
| 18 | 00007f3b24f9c710 | NOP                                 | 0              | 0                   | 0                          |                       |                           |                      |         |
| 19 | 00007f3b24f9c720 | NOP                                 | 0              | 0                   | 0                          |                       |                           |                      |         |
| 20 | 00007f3b24f9c730 | NOP                                 | 0              | 0                   | 0                          |                       |                           |                      |         |
| 21 | 00007f3b24f9c740 | NOP                                 | 0              | 0                   | 0                          |                       |                           |                      |         |
| 22 | 00007f3b24f9c750 | NOP                                 | 0              | 0                   | 0                          |                       |                           |                      |         |
| 23 | 00007f3b24f9c760 | NOP                                 | 0              | 0                   | 0                          |                       |                           |                      |         |
| 24 | 00007f3b24f9c770 | NOP                                 | 0              | 0                   | 0                          |                       |                           |                      |         |



**Thank you!**

# AI and Frameworks

Corey Adams, Datascience Team

[datascience@alcf.anl.gov](mailto:datascience@alcf.anl.gov)

# AI Frameworks

- ThetaGPU consists of high powered GPU accelerators that are prime for machine learning and deep learning applications.
- Here, I'll showcase the available tools and modules, as well as how to add your own tools, and links to further resources.
- We'll cover:
  - Modules;
  - Containers;
  - Scaling machine learning;
  - Investigating Machine learning performance.

# Machine Learning Modules

- Modules on ThetaGPU are available for the most popular machine learning frameworks:
  - Use ‘module avail’ to see all module options.
  - Machine Learning Frameworks are available with ‘**module load conda/\${DATE}**’, where date can be:
    - 2021-11-30
    - 2021-09-22
    - 2021-06-28
    - 2021-06-26
  - Once the module is loaded, you still need to do ‘**conda activate**’ to launch your conda environment.
- You can extend the conda environment with pip and virtualenv!
  - python -m venv --system-site-packages \${LOCATION}**
  - source \${LOCATION}/bin/activate**
  - pip install [desired packages]**
- Conda modules include pytorch, tensorflow, horovod, and other packages, if something is missing please open a support ticket or contact us.
- *More info on modules is here: [https://github.com/argonne-lcf/ai-science-training-series/blob/main/00\\_introToAlcf/02\\_howToSetupEnvironment.md](https://github.com/argonne-lcf/ai-science-training-series/blob/main/00_introToAlcf/02_howToSetupEnvironment.md)*

# Containers

- Containers are a great way to get the latest nvidia software in a portable way: you can use the same container on many systems, and run it in many locations.
- ALCF Systems support **singularity**, while many other locations support **docker**.
  - More info on singularity and docker are available here:
    - <https://sylabs.io/guides/3.5/user-guide/introduction.html>
    - <https://www.docker.com/>
- ALCF documentation pages have information about containers:
  - <https://www.alcf.anl.gov/support-center/theta-gpu-nodes/nvidia-containers>
- Significant tutorials and information about containers is available from the last computational performance workshop:
  - [https://github.com/argonne-lcf/CompPerfWorkshop-2021/tree/main/03\\_containers](https://github.com/argonne-lcf/CompPerfWorkshop-2021/tree/main/03_containers)
- Prebuilt containers for ThetaGPU are available here:
  - `/lus/theta-fs0/software/thetagpu/nvidia-containers/`**



# Scaling Deep Learning Software

- Scaling deep learning applications on ThetaGPU is easy with Horovod (TF and Pytorch) and DDP (Pytorch).
  - Documentation for the software is available online:
    - <https://horovod.ai/>
    - [https://www.tensorflow.org/guide/distributed\\_training](https://www.tensorflow.org/guide/distributed_training)
    - [https://pytorch.org/tutorials/intermediate/ddp\\_tutorial.html](https://pytorch.org/tutorials/intermediate/ddp_tutorial.html)
- The Datascience team regularly updates and presents tutorials (with examples) for scaling deep learning applications on ThetaGPU. For examples, please see:
  - [https://github.com/argonne-lcf/CompPerfWorkshop-2021/tree/main/05\\_scaling-DL](https://github.com/argonne-lcf/CompPerfWorkshop-2021/tree/main/05_scaling-DL)
  - Tomorrow's AI for Science Tutorial will cover distributed deep learning in great detail!

# Framework Performance

- In general, computations in python must be offloaded to the GPUs, with the deep learning frameworks or other tools, to achieve optimal performance.
- That said, it is easy to introduce bottlenecks into your python applications with one or two lines of poorly written code.
  - How do you find them? How do you improve it?
  - Significant information, including tutorials, examples, and guides for performance improvement, are available:
    - [https://github.com/argonne-lcf/CompPerfWorkshop-2021/tree/main/09\\_profiling\\_frameworks](https://github.com/argonne-lcf/CompPerfWorkshop-2021/tree/main/09_profiling_frameworks)
  - The deep learning frameworks also offer guides for performance measurements:
    - [https://pytorch.org/tutorials/recipes/recipes/profiler\\_recipe.html](https://pytorch.org/tutorials/recipes/recipes/profiler_recipe.html)
    - <https://www.tensorflow.org/guide/profiler>
- You should also pay attention to your data pipelines! Many deep learning applications stall out with data loading.
  - See this tutorial for tips and tricks: [https://github.com/argonne-lcf/ai-science-training-series/tree/main/03\\_dataPipelines](https://github.com/argonne-lcf/ai-science-training-series/tree/main/03_dataPipelines)

# Questions?

- Please ask here or send questions to:
  - [support@alcf.anl.gov](mailto:support@alcf.anl.gov) or
  - [datascience@alcf.anl.gov](mailto:datascience@alcf.anl.gov)

Thank you!!