Early User Experience on and Lessons Learned from the NERSC Cori GPU Cluster

Kelly L. Rowland*, Brian Friesen*, Jack Deslippe*, Brandon Cook*, Ershaad Basheer*, and Max Katz†
* *National Energy Research Scientific Computing Center (NERSC)*
Lawrence Berkeley National Laboratory (LBNL)
Berkeley, CA, USA
Email: kellyrowland@lbl.gov, bfriesen@lbl.gov, bgcook@lbl.gov, jrdeslippe@lbl.gov, ebasheer@lbl.gov
† *NVIDIA Corporation*
Santa Clara, CA, USA
Email: mkatz@nvidia.com

***Abstract*—The next-generation NERSC supercomputer "Perlmutter" will feature a combination of nodes which are CPU-only and nodes which contain both CPUs and GPUs. To help users prepare for this system, NERSC has procured a Cray CS-Storm 500NX system of 18 CPU-GPU nodes. Despite having little in common architecturally to the current NERSC production system "Cori", this cluster has been fully integrated into Cori and is available to users by access request. These GPU-accelerated nodes are primarily for testing and development work, with priority access given to users participating in the NERSC Exascale Science Applications Program (NESAP). In this paper, we discuss management and deployment of the GPU-specific software provided by NERSC consultants for use on the nodes, job scheduling policies, and efforts in user communication.**

We also detail the challenges encountered in flexible and efficient software deployment, and the solutions we have found most effective. Specifically, we have explored environment modules, already used on the Cori XC system, as well as programming environments built into Shifter images, which address scalability and cross-compilation challenges. Additionally, we discuss our approach for working with early-access users to get their feedback and respond to it accordingly. We conclude our paper with a summary.

***Keywords*-GPU; user environment; scheduling**

I. Introduction

The National Energy Research Scientific Computing Center (NERSC) is the primary scientific computing facility for the Office of Science in the U.S. Department of Energy, located at Lawrence Berkeley National Laboratory (LBNL). NERSC is dedicated to providing computational resources and expertise for basic scientific research, and a significant part of this effort is ensuring that NERSC staff and users alike are prepared to successfully use the large-scale NERSC systems.

The incoming "Perlmutter" system will be an HPE Cray system and will be composed of GPU-accelerated nodes as well as CPU-only nodes. In preparation for the delivery of this system, NERSC has integrated an 18-node Cray CS-Storm cluster with 144 NVIDIA V100 GPUs into the current "Cori" Cray XC40 system. In this paper, we describe the experiences and lessons learned in supporting the users on these GPU nodes.

II. Motivation

To meet the mission of accelerating scientific discovery at the DOE Office of Science through high performance computing and data analysis, NERSC staff work with vendors and users in the leadup to a new system delivery to ensure application readiness and a successful deployment. This preparation work is particularly important for the incoming Perlmutter system since it will be the first full-scale NERSC system to feature GPU-accelerated nodes.

To this end, the NERSC Exascale Science Applications Program (NESAP) is a collaborative effort in which NERSC partners with code teams, vendors, and library and tool developers to prepare for advanced architectures and new systems. The primary purpose of the Cori GPU nodes includes facilitating R&D activities by NESAP teams and other strategic partners to help ensure the success of key workloads on the Perlmutter system. As is described in more detail in the following sections, NESAP users have been given priority access to the Cori GPU nodes to prepare application codes for the incoming architectures.

III. System Architecture

The hardware configuration of the Cori GPU nodes was guided by the design of the Perlmutter system, as it is intended to be a platform used to prepare for larger-scale use of Perlmutter. The system is a Cray CS-Storm 500NX - 4U cluster, with 18 nodes, each with 8 NVIDIA Tesla V100 ("Volta") GPUs and two sockets of 20-core, 2.4 GHz Intel Xeon Gold 6148 ("Skylake") processors. Each node has 384 GB DDR4 RAM, 930 GB on-node NVMe storage, and 4 dual-port NVIDIA Networking MT27800 (ConnectX-5) EDR InfiniBand network cards.

The node configuration is broadly similar to the NVIDIA DGX-1 appliance [1]. The 8 GPUs on each node are connected to each other in a 'hybrid cube-mesh' topology. In this arrangement, each GPU contains a single NVLink connection to each of two GPUs, and a doubly-bonded

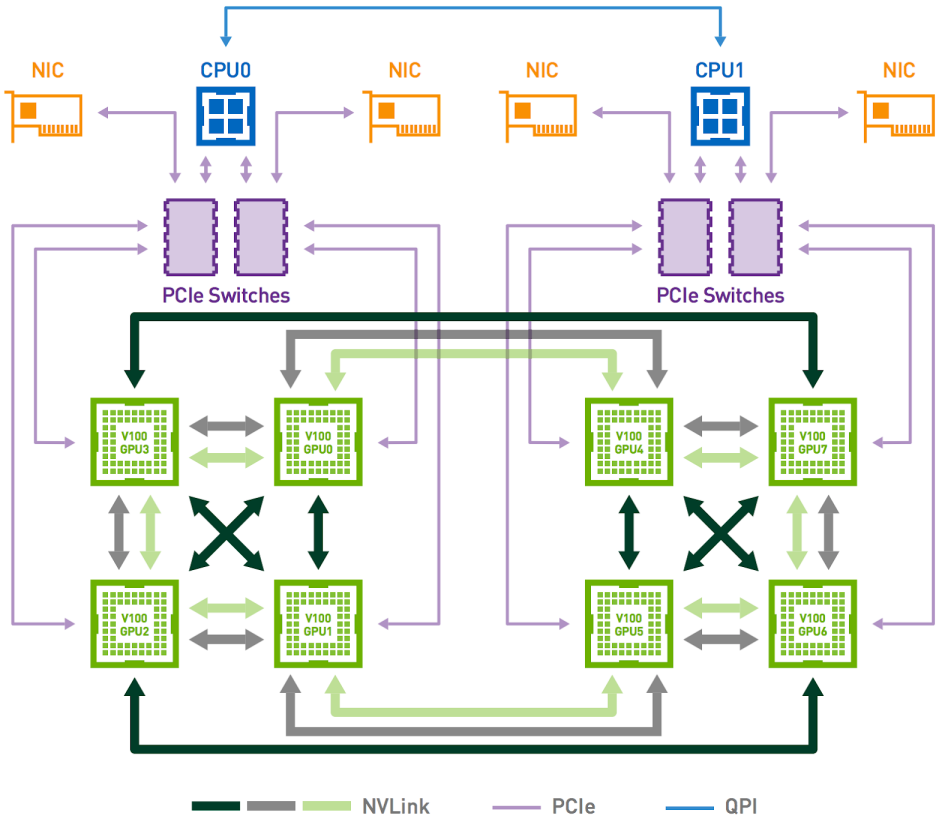

Figure 1. Node topology (adapted from NVIDIA [1]).

NVLink connection to two more GPUs, with twice the bandwidth of a single NVLink connection. So, each GPU is connected directly to 4 others. All GPUs are connected to the Skylake CPUs and the InfiniBand host channel adapters (HCAs) via PCIe 3.0; there are 4 PCI switches per node connecting the GPUs, NIC, and CPUs at a peak bandwidth of 16 GB/s in each direction. A diagram of this topology is provided in Figure 1.

The GPU nodes are accessed via the Cori login nodes but are on a separate Slurm controller and run OpenSUSE rather than a SLES-based Cray Linux Environment. The node images are constructed from a Cray image recipe maintained in Git using SMWFlow tools.

Two of the four HCAs on each node are connected to one of six leaf switches, two of which in turn connect to one of three root switches. A third HCA on each node is connected to a Lustre InfiniBand switch such that the GPU nodes can mount Cori's high-performance Lustre file system. The final HCA on each node is connected to the IBM Spectrum Scale routers, such that the `$HOME` and "community" file systems are also mounted on the GPU nodes. Making all of the NERSC file systems available to the GPU cluster has vastly increased productivity for NERSC users as they prepare their applications for Perlmutter.

IV. Software

Although the GPU cluster is tightly integrated into the Cori system, most Cray software on Cori targets the XC architecture, not the CS-Storm architecture, and thus does not work on the GPU nodes. Cross-compilation for Cori GPU nodes from Cori XC or eLogin nodes is difficult, due to differences in software available to each type of node. The chief examples are Mellanox OFED and CUDA libraries on Cori GPU, and Cray PE and MPI libraries on eLogin/XC nodes. Consequently, most software interactions for Cori GPU must take place directly on Cori GPU nodes. However, to present a consistent user-facing environment, we provide software to users for the Cori GPU nodes through environment modules as is done on the Cori XC system. Additionally, the use of containerized environments via NERSC's Shifter software package [2] has been a boon to staff and users alike.

A. Environment Modules

NERSC uses environment modules [3] to manage most of the software that is provided for users on the Cori XC system. Since users are accustomed to working with software through the module interface, it was a logical next

step for us to provide software for the Cori GPU nodes via environment modules as well.

Initially, the GPU-specific software modulefiles were located in the same system directory as the modulefiles for Cori XC software. However, this proved to be confusing for users, as it was not immediately clear which software modules were intended for use on which system. For example, version 4.0.2 of OpenMPI was intended for use on the Cori XC system, while version 4.0.3 of OpenMPI was intended for use on the Cori GPU nodes, but there was no clear indication to users that this was the case when the modulefiles were first put in place.

To mitigate this issue, the modulefiles specific to the software built for the Cori GPU nodes were moved to a separate directory that is not in a user's `$MODULEPATH` by default. Users are now directed to load a `cgpu` module; this module prepends the GPU-specific software modulefile path to the user's `$MODULEPATH` so that any Cori GPU software versions are found before Cori XC software. With this approach, users may also still load Cori XC software modules for use on the Cori GPU nodes in the event that they wish to do so, though this must be done with caution.

B. Shifter

Shifter is a software package that allows user-created images to run at NERSC [2]. Since it runs on both the Cori XC nodes as well as the GPU nodes, it is a natural solution to the aforementioned issues surrounding cross-compilation. Using Shifter provides a software development environment which enables compilation for Cori GPU nodes from any Cori node (another GPU node, eLogin nodes, XC nodes) and provides a natural transition of the software to the incoming Perlmutter system.

By default, when using Shifter on the Cori GPU nodes, CUDA drivers are copied into the Shifter image and the libraries are placed in the image environment's `LD_LIBRARY_PATH`. Additionally, a specific CUDA module is provided for use in Shifter containers for users who may wish to manually insert CUDA drivers into an image. Users are instructed to load the `cuda/shifter` environment module in their job script; this module defines the `SHIFTER_CUDA_ROOT` environment variable and points it to the version of the CUDA SDK installation which works in Shifter images.

To leverage existing vendor efforts, NERSC provides TensorFlow and PyTorch Shifter images based on NVidia's GPU Cloud Containers (NGC) [4] for users. These images are based on containers which are optimized for best performance on GPUs, allowing us to provide an enhanced user experience and ensure appropriate resource utilization without expending excess staff effort. As discussed in further detail in Section V, many users access the Cori GPU nodes through the NERSC Jupyter interface. To facilitate use of the optimized deep learning packages through this interface in addition to directly through scripts, we provide Jupyter kernels based on these images.

V. Scheduling and Resource Access

As mentioned, the primary purposes of the Cori GPU nodes include R&D activities by NESAP teams, NERSC staff training, and hackathons and workshops. This style of work necessitates a high-availability system ready for interactive use. Secondary goals for the Cori GPU nodes include providing opportunities for development and optimization work done by teams outside of NESAP, evaluation and testing of machine learning (ML) techniques in scientific workloads, and offering a platform for GPU-enabled software as system availability allows for. These secondary goals tend to require jobs with longer walltime and multiple nodes. So, we are faced with a unique challenge in providing user access to the Cori GPU nodes: how can we prioritize resource availability and additionally allow users to utilize the system's spare cycles?

To meet this challenge, we make use of various features of the Slurm workload manager. Our job queues on the Cori GPU nodes are structured such that:

1) Jobs running between 8:00 AM Pacific Time (3:00 PM UTC) and 8:00 PM Pacific Time (3:00 AM UTC) from Monday through Friday are limited to 4 hours of run time.
2) Jobs running before 8:00 AM Pacific Time (3:00 PM UTC) or after 8:00 PM Pacific Time (3:00 AM UTC), or on weekends, can run until 8:00 AM Pacific Time on the next weekday.
3) Users affiliated with NESAP projects may add an additional flag `-q special` to their batch and interactive jobs to be placed in a higher-priority queue.
4) Interactive jobs are limited to 2 GPUs. Jobs requiring more than 2 GPUs can be submitted via the `sbatch` command instead of `salloc`.

Points 1 and 2 in the above specifications are accomplished via a pair of floating reservations in Slurm, shown in Figure 2.

The `gpu_4hour_limit` reservation is set to be in effect 4 hours in the future from job submission between 8:00 AM Pacific Time (3:00 PM UTC) and 8:00 PM Pacific Time (3:00 AM UTC) from Monday through Friday and prevents jobs with requested walltime greater than 4 hours from starting during these times. The `gpu_8am_barrier` reservation is in place to ensure system availability at the start of each weekday (i.e., it is in place to prevent long-running jobs from running overnight and into the business day).

With these constraints in place on the system, the Cori GPU nodes are made readily available for interactive use during weekdays and then opened up for larger and longer jobs to run overnight (in continental US time zones) and on weekends. As noted in item 3 above, to ensure that users

```
ReservationName=gpu_4hour_limit
   StartTime=2020-08-20T20:36:46 EndTime=2020-08-20T21:36:46 Duration=01:00:00
   Nodes=cgpu[01-17] NodeCnt=17 CoreCnt=680 Features=(null) PartitionName=gpu
   Flags=OVERLAP,IGNORE_JOBS,SPEC_NODES,TIME_FLOAT
   TRES=cpu=1360
   Users=root Accounts=(null) Licenses=(null) State=INACTIVE BurstBuffer=(null)
   Watts=n/a MaxStartDelay=(null)

ReservationName=gpu_8am_barrier
   StartTime=2020-08-21T08:00:00 EndTime=2020-08-21T08:01:00 Duration=00:01:00
   Nodes=cgpu[01-17] NodeCnt=17 CoreCnt=680 Features=(null) PartitionName=gpu
   Flags=OVERLAP,IGNORE_JOBS,WEEKDAY,SPEC_NODES
   TRES=cpu=1360
   Users=root Accounts=(null) Licenses=(null) State=INACTIVE BurstBuffer=(null)
   Watts=n/a MaxStartDelay=(null)
```

Figure 2. Slurm reservations.

affiliated with the NESAP program are given priority access to the Cori GPU nodes, jobs submitted by these users can add the `-q special` flag to their Slurm job submissions. This option routes the job to a high-priority queue which supersedes other jobs submitted to the system. Lastly, as noted in item 4 above, interactive jobs are limited to 2 GPUs per job. So that the system queue is not blocked by any single user running a plurality of short jobs simultaneously, a given user is limited to having a maximum of 5 jobs running simultaneously.

The queue configuration for the Cori GPU nodes continues to be an iterative process, with feedback from NERSC staff and users informing the structure. For example, in a previous instance of the configuration, larger batch jobs were allowed to start during business hours but were routed to a queue where they could be preempted by other jobs. This turned out to be a source of frustration for users, as it was often the case that a larger job would start readily but get preempted before it could checkpoint or produce other useful output. Additionally, the system was opened to users with a default setting of exclusive access to nodes. This has since been changed to a default resource allocation of sharing nodes by default, with users encouraged to request only the minimum resources needed for a given job.

One method of job submission that is popular with users is GPU access through a Jupyter notebook. In this workflow, users log into the NERSC JupyterHub instance[1] with their NERSC credentials and then choose to "start" a job on a shared GPU node. This submits a batch job, requesting access to one GPU for four hours. Figure 3 shows a screenshot of the NERSC Jupyterhub interface, including the user option to start a Cori GPU job.

Figure 4 plots the cumulative number of jobs started on the Cori GPU nodes since the system came online. The jobs

[1]https://jupyter.nersc.gov/

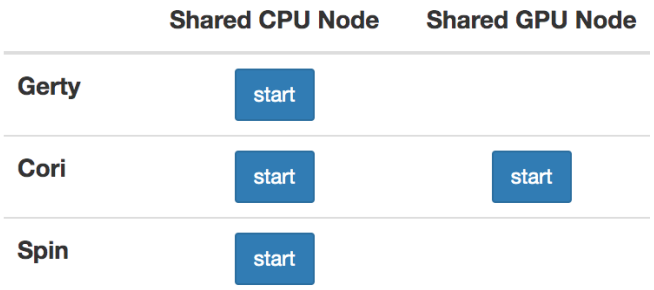

Figure 3. Jupyter access interface.

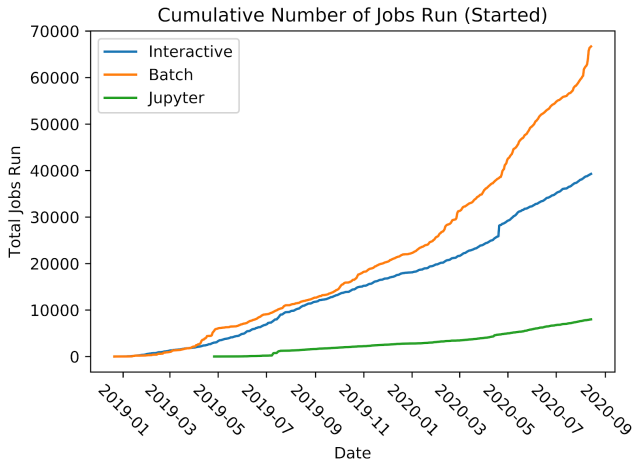

Figure 4. Number of jobs started on the Cori GPU nodes over time, broken down by job submission type.

are broken down by type of job submission: interactive jobs, batch jobs, and jobs submitted via the Jupyter interface.

When users were initially let onto the system, batch and interactive jobs were submitted at approximately the

same frequency. Over time, however, the cumulative number of batch jobs submitted continues to outpace the number of interactive jobs submitted, potentially indicating greater user readiness over time (e.g., a development pattern of interactive jobs followed by greater numbers of batch jobs once kernels have been made GPU-ready).

Even with the recent spike in batch job submissions, it is useful to note that approximately 40% of jobs which have started on the Cori GPU nodes were done so either interactively or via the Jupyter interface. Since this significant fraction of the workload on the Cori GPU nodes depends on immediate access to GPU resources, it behooves us to continue to ensure that nodes are readily available.

VI. User Communications

Access to the Cori GPU system is granted by request; not all NERSC users are enabled for use of the nodes. Any user affiliated with a NESAP project is given access to the system with no further vetting beyond the initial request; users affiliated with strategic partner facilities are also prioritized for access to the Cori GPU nodes. NERSC users whose work is not associated with any NESAP teams may still be granted access to the Cori GPU nodes if their use case is amenable to the constraints enumerated above.

All users enabled on the Cori GPU system are added to an email list for communications; NERSC staff use the list to announce major changes to system software, GPU-related training events, and upcoming calls with users. NERSC staff host video calls with users on a monthly basis to discuss planned changes to the system software and get user feedback. Users are encouraged to call in to ask questions or report issues; the call is left open for discussions following any news from NERSC staff.

Documentation for the Cori GPU system is provided online[2] and includes information about system hardware, job submission, and available software. These pages are hosted separately from the main NERSC user-facing documentation to clearly delineate that the guidance is specific to the Cori GPU nodes and not the Cori XC system. Examples of software usage and expected output are provided for users who are new to software such as the CUDA platform; the website material is updated by NERSC staff as more packages are made available on the system.

VII. Summary

Early experiences with the Cori GPU nodes have been a valuable learning opportunity for NERSC staff and users to prepare for the incoming Perlmutter system. Although there are minor architectural differences between the two systems, we anticipate facing challenges on Perlmutter which are similar to those that we have encountered with the Cori GPU nodes.

[2]https://docs-dev.nersc.gov/cgpu/

A primary challenge in managing the system has been maximizing node availability for interactive and Jupyter jobs while also achieving high cluster utilization. To achieve both of those goals, we continue to update the nodes' queue structure, incorporating job accounting data as well as feedback from NERSC staff and users. This challenge of managing and scheduling a relatively small number of high-density nodes will extend into our future production work; Perlmutter will have a much smaller number of nodes than Cori does, while the number of NERSC users has increased to over 7000 and continues to grow.

Finally, the challenges associated with cross-compilation as a first-class workflow have become readily apparent. This has been less of an issue on previous systems where the eLogin and compute nodes look fairly similar, but the Cori GPU nodes are distinct from the Cori eLogin nodes. The apparent distinction will also be the case on Perlmutter, and we are using the Cori GPU nodes in preparation for meeting this challenge at scale.

References

[1] NVIDIA Corporation, "NVIDIA DGX-1 With Tesla V100 System Architecture," https://images.nvidia.com/content/pdf/dgx1-v100-system-architecture-whitepaper.pdf.

[2] *Shifter: Containers for HPC*. Cray User Group, 2016. [Online]. Available: https://cug.org/proceedings/cug2016_proceedings/includes/files/pap103s2-file1.pdf

[3] X. Delaruelle, "Environment Modules," http://modules.sourceforge.net/.

[4] NVIDIA Corporation, "NVIDIA GPU Cloud Documentation," https://docs.nvidia.com/ngc/.