Scheduler Optimization for Current Generation Cray Systems

Morris Jette
SchedMD, jette@schedmd.com

Douglas M. Jacobsen, David Paul
NERSC, dmjacobsen@lbl.gov, dpaul@lbl.gov

Abstract **- The current generation of Cray systems introduces two major features impacting workload management: DataWarp burst buffers and Intel Knights Landing (KNL) processors with their variety of NUMA and MCDRAM modes. DataWarp provides a high-bandwidth, cluster-wide file storage system for applications on Cray systems. In a typical use case, DataWarp resources are allocated to a job and data is staged-in before compute resources are allocated to that job. Similarly, DataWarp resources are retained after computation is complete for staging-out of data. Knights Landing are the latest generation of Intel Xeon Phi processors, supporting five different NUMA modes and three MCDRAM modes. Applications may require a specific KNL configuration to execute or its performance may vary considerably depending upon processor configuration used. If KNL resources with the desired configuration are not available for pending work, the overhead of rebooting compute nodes must be weighed against running the application in a less than ideal configuration or waiting for processors already in the desired configuration to become available. These are some of the issues faced on the Cori system at National Energy Research Scientific Computing Center (NERSC), with 2,004 Xeon Haswell processor nodes, 9,688 KNL nodes and 1.5 PetaBytes of DataWarp storage. This paper will present the algorithms used by the Slurm workload manager with respect to DataWarp and KNL scheduling, a statistical analysis of NERSC's workload, and experiences with Slurm's management of DataWarp and KNL. Topics covered will include core specialization, task affinity, zonesort, failure management and scalability. The paper will conclude with identification of areas where Slurm and Cray infrastructure might be improved to increase system fault tolerance, scalability, and utilization.**

Overview of Intel Knights Landing

Knights Landing (KNL) is the name of Intel's second generation Many Integrated Core (MIC) architecture. It contains up to 72 Airmont (Atom) cores with a two dimensional mesh interconnect [1]. Each core contains four threads. KNL contains up to 16 GB of MCDRAM, a version of high bandwidth memory (or on-package memory), which can be configured as "cache" - allowing transparent migration of pages between main memory (DRAM) and the MCDRAM, or "flat", allowing the application to directly address the MCDRAM with no transparent caching, or a combination of the these two extremes. The NUMA modes adjust the relationship between the many cores to optimize internal communication depending on an application's requirements. An important feature of KNL is the ability to modify it's NUMA and/or MCDRAM configuration at boot time as shown in Table 1. This mode change is accomplished by dynamically reconfiguring and rebooting nodes during live operation of the system and is fully under the control of the user within some limits granted by the workload manager.

Type	Name	Description
NUMA	a2a	All to all
NUMA	hemi	Hemisphere
NUMA	snc2	Sub-NUMA cluster 2
NUMA	snc4	Sub-NUMA cluster 4
NUMA	quad	Quadrant
MCDRAM	cache	All of MCDRAM used as cache
MCDRAM	equal	MCDRAM used partly as cache and partly combined with primary memory
MCDRAM	flat	MCDRAM combined with primary memory into "flat" space

Table 1: NUMA and MCDRAM modes available on KNL.

Overview of DataWarp

DataWarp is Cray's implementation of burst buffers, direct-attached solid-state disk (SSD) storage providing higher bandwidth than an external parallel file system [2]. Like an external file system, DataWarp storage is made available as a global file system rather than storage bound to individual compute nodes. Two modes of use are supported: job-specific and persistent. As the name suggests, job-specific allocations are bound to a specific job allocation. The lifetime of a job-specific allocation typically extends both before and after the job's allocation of compute resources (i.e. CPUs and memory) in order to stage-in and stage-out files as needed. Persistent allocations are not tied to any specific job, but can be used by multiple jobs. Access to a

persistent allocation can be requested by any job and controlled using POSIX file permissions on its individual files and directories. Persistent allocations must be explicitly created and destroyed.

The DataWarp implementation on NERSC's Cori system contains 288 DataWarp servers containing 576 Intel SSDs, with a total of 1.5 PBs of storage. The DataWarp software (DWS) combined with Slurm, provides users with a wide variety of choices to configure and access this high performance I/O subsystem. Cori's DataWarp performance has been measured at ~1.6 TB/sec. and up to ~12.5 million IOPS.

NERSC's current layout takes advantage of Slurm's support for multiple alternate DataWarp pools to offer the choice of three separate pools. The default wlm_pool is comprised of 200 DW-servers, with an allocation size of 85 GB (granularity). The sm_pool is comprised of 80 servers with a granularity of 20 GB, allowing for wider striping at smaller space allocations. The dev_pool is comprised of the remaining 8 DW-servers and is reserved for testing different configurations and DWS features (ex. transparent cache).

Presently DW-servers provide filesystem I/O services only, with no concurrent compute functionality. This capability is currently under development by both Cray and SchedMD. This functionality will be useful to a number of user workflows to offload additional overhead of I/O from the compute nodes in a job.

OVERVIEW OF SLURM

Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system [3] in use on roughly half of the Top-500 systems [4] including the NERSC Cori system [5]. Some of the capabilities provided by Slurm include accounting, advanced reservations, backfill scheduling, fair-share scheduling, a multitude of resource limits, and sophisticated job prioritization algorithms. Approximately 60% of Slurm's code is in a common kernel with the remainder in various plugins. Slurm includes approximately 130 different plugins designed to support a wide variety of architectures and configurations. The plugins are either identified in a Slurm configuration file or command line option and loaded when the daemon or command is started. For example Slurm network topology plugins available include "tree", "hypercube" and "3d_torus". Other plugin selections are user controlled, such as the MPI implementation used for a particular job.

While Slurm can operate as a scheduler on top of Cray's ALPS resource manager [6]. On most systems, including Cori, Slurm uses Cray APIs to interface directly with the underlying Cray infrastructure in order to eliminate the need for ALPS. Such a configuration makes use of several Slurm plugins designed specifically for Cray systems and a Slurm daemon running on each compute node, responsible for launching the user application and binding it to the appropriate resources. Of particular note, this configuration provides the ability to run multiple applications belonging to multiple users on each compute node.

Cray provides several commands used by Slurm to manage KNL and DataWarp. The *cnselect* command reports currently active MCDRAM and NUMA modes on the KNL nodes. The *capmc* command identifies KNL nodes, node states, available MCDRAM and NUMA modes on each node plus the MCDRAM and NUMA mode to be used on next node reboot (not necessarily the modes currently in effect). *Capmc* provides a mechanism to change a node's MCDRAM and NUMA mode and reboot the node. Also used by Slurm, but outside of this paper's scope, *capmc* provides options to monitor and manage each node's power consumption. The *dw_wlm_cli* command provides DataWarp status information, the ability to create burst buffers, stage-in and stage-out files, and later delete burst buffers.

Slurm provides support for KNL including the ability for user's to modify the KNL MCDRAM and NUMA modes and reboot the nodes [7]. At Slurm startup, it uses Cray's *capmc* and *cnselect* commands to determine which nodes include a KNL processor, their current and available MCDRAM and NUMA modes, and total high bandwidth memory. This information is recorded in Slurm's node state information, making use of "Active Features" and "Available Features" fields to record the current MCDRAM and NUMA modes plus the modes which can be made available through node reboot. The MCDRAM mode is recorded and managed as a Slurm Generic Resource [8].

Slurm configuration parameters specific to KNL include:

- MCDRAM and NUMA modes which can be made available (possibly a submit of the MCDRAM and NUMA modes which are possible on the KNL nodes)
- Identification of users allowed to modify MCDRAM or NUMA modes (all users by default)
- Expected node boot time
- Path to Cray's *capmc* and *cnselect* programs
- Various timeouts and failure recovery retry intervals

When a user submits a job to Slurm, he can identify the required MCDRAM and/or NUMA modes as job constraints. If specified, the job will not be initiated until those modes are active on the nodes to be allocated. If necessary, and the user has the appropriate authentication, nodes will be booted into the appropriate MCDRAM and NUMA modes required by the job. Since the time required to boot nodes can be significant, Slurm's configuration includes an expected boot time for optimizing scheduling decisions. If resources are expected to be available for allocation to a pending job in a more timely fashion by waiting for resources to become available as running jobs

terminate rather than rebooting currently available nodes, the job will remain pending. The process of rebooting a node can not start until after all jobs currently allocated on that node have terminated. If accounting is configured, the job will be charged for not only its execute time, but also the time consumed for rebooting nodes. However, the boot time will not count against the job's time limit.

When a node reboot completes and the Slurm daemon starts on the node, it's current MCDRAM, NUMA and high bandwidth memory configuration are validated and the job initiated. If the reboot of any allocated nodes fails, the affected node will be drained and the job requeued. The requeued job will be allocated a new set of resources, possibly including previously allocated resources, and begin execution when possible.

Slurm provides several mechanisms for managing task layout on the allocated nodes. Slurm provides options to specify the number of tasks per node, per NUMA or socket, per core, and per thread (--ntasks-per-node, --ntasks-per-socket, --ntasks-per-core and --ntasks-per-thread respectively). Slurm also provides options to manage how task ranks are distributed at each level of resource hierarchy, either using explicit information from the user or in an automated fashion using simple directives. For example the option "--distribution=block:cyclic:cyclic" indicates that ranks should be consecutive on each node (from the first "block"), then sequentially across the NUMA of each node (the first "cyclic"), then sequentially across each core of the NUMA (the second "cyclic"). A Slurm option is also available to perform the specified distribution on groups of consecutive ranks rather than individual ranks (i.e. allocate 2 tasks on each core before cycling to the next core) [9].

Once a task is allocated to some resource, it can be implicitly bound to that resource using cpusets or Linux cgroups. By default tasks are bound to the appropriate entity based up the ratio of tasks to NUMA, cores or threads. For example if a job has one task per core, each task will be bound to a core. User options are available to bind tasks at a different level. For example, one can easily bound each task to an individual thread even if there is one task per core.

Given the number of threads on a KNL, the compute resources consumed by the operating system and daemons (including Slurm) may adversely impact user applications. This may be addressed by reserving some count of threads or cores on each node for system use, which is referred to in Slurm as thread or core specialization [10]. Memory can also be reserved for system use, which is referred to as memory specialization. These resources are removed from availability to user applications, and in the case of specialized threads, system processes are explicitly bound to those resources.

Memory resources are managed in a similar fashion to compute resources. Users can specify the desired memory on a per-node or per-CPU basis. The application is bound to the allocated memory using Linux cgroups. User options are available to bind each task to the memory in its local NUMA or to prefer its local NUMA. Application performance can be adversely impacted on KNL if the free pages are not regularly sorted [11]. Slurm has an option which will automatically perform a sort on allocated NUMA at application start ("--mem_bind=sort").

SCHEDULING A HETEROGENEOUS SYSTEM

The NERSC Cori system operates a very demanding and heterogeneous workload including data-intensive jobs on the 2004 Haswell nodes as well as a more traditional HPC workload on the primary capacity of ~9600 KNL nodes. The Haswell nodes are collectively considered the "data partition" by NERSC and have been configured to provide High Throughput Computing-friendly queues, including both node-exclusive and single-core scheduling as well as advanced capabilities, such as Shifter for virtualized environments, to enable many non-traditional HPC workflows. The KNL nodes, on the other hand, are more focused on the HPC workload. That said, both Haswell and KNL are available and accessible for scientists to perform any relevant calculation within their DOE-assigned allocation.

Slurm is critical infrastructure for supporting this hybrid workload, both as a scheduler as well as a resource manager. As a resource manager, Slurm grants NERSC the ability to run as many jobs as there are cores on the nodes, which is used provide support large serial job arrays on the Haswell nodes. In addition, the plugin architecture enables NERSC to operate a number of custom job-control plugins to customize the node environment to match user-specifications. As a scheduler, Slurm provides flexible backfill scheduling enabling many different fine-grain policies to be adopted while still effectively performing future planning for hundreds of jobs (resource reservations), and allowing many thousands of jobs access to backfill capacity.

NERSC has a number of early science participants on the KNL partitions as part of its NERSC Exascale Science Applications Program (NESAP). The NESAP codes have been updated collaboratively between the code development team, NERSC, Cray, and Intel to prepare for the Many Core architecture of KNL. These users were granted early access to KNL, enabling both higher priority, larger scale and longer walltimes for those users, while general general users only have access to relatively smaller allocations. This is achieved by setting different resource limits in different Slurm Qualities of Service (QoS), and then mapping jobs and users to specific QoS based on both the user status as well as the resource requests. Slurm is capable to specifically allow certain users access to some QoS, but not

others. The ability to gate access to KNL will be maintained even into production to ensure that large jobs that individually consume vast system resources are proven to work well on the Cori system.

The NERSC Application Performance and Advanced Technology Groups have identified that the NUMA "quad", and MCDRAM "cache" mode are likely to be the best fit for the NERSC workload, at least for now. However, there is much interest from a subset of our users, as well as some applications that have been specifically tuned for directly managing the MCDRAM, making the other possibly KNL modes (specifically quad/flat) attractive for some users. To this end, NERSC does allow users to directly request any KNL mode, and does permit some level of node reboots. Because of the delay generated by a node reboot (ranging from 25 to 50 minutes on the Cori system, depending on quantity being booted and other system conditions), as well as the non-zero chance of node failure during the boot process, we do prefer to minimize node boots, and especially "mode thrashing" – repeatedly booting back and forth between two popular modes. While we are still iterating on the specific parameters of the configuration, the general structure is that the KNL nodes are split over three overlapping partitions:

- "knl" – all quad/cache jobs are routed here
- "knl_reboot" – all other modes are routed here
- "knl_regularx" – all large jobs exceeding 85% of the system are routed here

The "knl" and "knl_regularx" are fully overlapping at this time, though in the future as more refined debug and interactive capabilities are added, the "knl" partition will be made smaller. The "knl_reboot" fully overlaps the "knl" partition, representing the final 3,400 nodes of it. This limits the maximum scale of non-quad/cache jobs, ensuring that some mode-changes large enough to potentially disrupt the system do not occur automatically. By routing our "default" mode of quad/cache to KNL, and then granting it most of the resources, we ensure that most jobs are able to proceed nominally without the disruption of a mode change.

Staring with Slurm version 17.02, we are able to define a "delay_boot" parameter which allows a job with sufficiently high priority to be delayed in order to attempt to make use of nodes already booted to the desired mode. We are still refining and tuning our configuration with respect to this, but one can think of the "delay_boot" parameter as providing a movable barrier between flexible sub-queues of the KNL partitions each representing different modes. It is best not to think of the "delay_boot" as a delay that a job would necessarily have to endure, but rather an maximum wait time before reconfiguring the system. This should allow Cori to adopt meta-stable states based on the demand in the queues without manual intervention by the systems administrators. The delay_boot parameter can be over-ridden by users who are in a hurry and don't mind devoting their allocation to rebooting nodes.

At present, Haswell and KNL jobs are largely separate – meaning users are required to select either Haswell or KNL, and no user-accessible partitions allow a job to span the two. Moving into the future, we do see a growing use-case for fully heterogeneous jobs, making use of both Haswell and KNL nodes. To ensure we have this capability in the future, Cori is already co-scheduling all Haswell and KNL jobs – the system is "fully integrated", meaning that both segments of the system share the same network and share the same scheduler instance. Both sets of jobs use a common priority scheme and are scheduled using most of the same policies. Tuning backfill scheduling to handle the volume of requests has been an ongoing process, but does make use of both new scheduling algorithms created based on NERSC's requirements by SchedMD (bf_min_prio_reserve), as well as NERSC customizations to the backfill scheduling algorithm (dynamic priority management). With these advanced algorithms for backfill scheduling, NERSC can successfully backfill many thousands of jobs, guaranteeing every possible backfill opportunity is found and used.

DataWarp Scheduling

Aside from the I/O performance provided by DataWarp, the largest benefit for NERSC users is the stage-in/stage-out capabilities offered via a single command line option (or multiples if desired) that bypass the idling of precious compute cycles waiting for data to be copied in to or out of DataWarp. The compute resources remain available for other jobs while data is being staged in. Likewise, the compute resources become available immediately upon job completion, even while data is staged out in the background. Slurm accepts DataWarp directives in several forms:

1. The directives can be included within a batch script using a prefix of "#DW" on each line.
2. The directives can be specified in a stand-alone file using the same "#DW" prefix. The stand-alone file is typically used for interactive jobs, which otherwise lack a file.
3. A subset of DataWarp directives can be specified on the execute line of the Slurm command used to create the job allocation. This mechanism is typically used for interactive jobs with only simple directives. Slurm uses these command line options to construct a file of DataWarp directives as in case #2 above.
4. In addition to the standard DataWarp directives with the "#DW" prefix in a script, Slurm uses a "#BB" prefix for extended capabilities. This provides a mechanism to create and destroy persistent allocations.

In all of these instances, Slurm makes use of DataWarp tools to perform resource allocation, stage-in, stage-out and deallocation operations. Slurm's role is largely in the

scheduling of these operations to optimize the overall system responsiveness and utilization.

DataWarp resources can be consumed by either job-specific, persistent burst buffers, swap and transparent cache from each of the available DataWarp pools. The current default user quota is 52 TB and supports quota increases on a per user basis. Job-specific buffers must be allocated prior to job execution in order to potentially stage-in files as needed. They typically must also persist after the job execution until file stage-out is complete. Persistent burst buffers can be created by a user job, but persist until explicitly destroyed, potentially by another user job. In both cases, Slurm's scheduling of DataWarp resources is coordinated using these factors:

1. Slurm's resource limit, which can be configured on a per-job, per-user, or per-account
2. Slurm advanced reservations, which can reserve some quantity of DataWarp resources at a specific time for specific users or accounts
3. The expected initiation time of pending jobs (i.e. when compute resource, licenses, etc. are expected to be available)

NERSC users have taken advantage of Persistent Reservations (PR) where the data produced or staged-in is available to multiple jobs of the same user or to other collaborating user jobs via POSIX file permission controls.

Slurm will allocate DataWarp resources to the jobs expected to be initiated soonest. Once the DataWarp resource allocation completes, file stage-in begins. The allocation of compute resources to a pending job will not happen until it's DataWarp resources have been allocated and file stage-in completes. Since the expected initiation time of pending jobs can change through time, job-specific DataWarp resources allocated to pending jobs can be revoked at any time and re-allocated to other pending jobs which are expected to start sooner than the job originally allocated the resources. Persistent burst buffers are never revoked after allocation, even if the job creating the buffer is deleted. The most common failure with respect to DataWarp scheduling is a failure in the stage-in or stage-out operation. In the case of a file stage-in failure, the burst buffer allocation is revoked and the job held for review by the user or a system administrator. In the case of a file stage-out failure, the burst buffer is retained and the job is kept in a completing state for investigation by the user and/or system administrator.

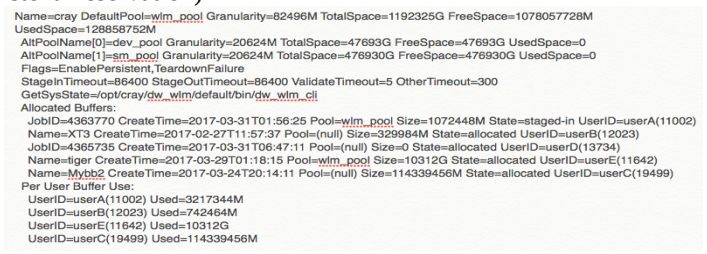
Support for KNL mode change (reboot) is fully integrated with Slurm/DWS. A job requiring a different mode will first allocate DataWarp space and stage-in data (if required), once compute nodes are assigned and the reboot completed (spanning multiple minutes) the DataWarp allocation is then mounted on the compute nodes at job launch.

A job which has not started execution will release DataWarp resources without staging-out any files. Once a job using DataWarp has started execution, files will be staged-out at termination no matter what the reason for termination. The only exception to this is if the job is explicitly cancelled by the owner or a system administrator using Slurm's *scancel* command with the "hurry" option. In that case, the DataWarp resources will be released for use by other jobs without going through the stage-out process.

Slurm provides a number of configuration control parameters. Two parameters used by NERSC are; EnablePersistent and TeardownFailure. EnablePersistent controls whether users are allowed to create their own Persistent Reservation as opposed to restricting permission to system administrators. TeardownFailure releases DataWarp allocations in the event of a failure (job or staging). This avoids wasting DataWarp resources while awaiting failure notification and analysis.

The Slurm "scontrol show burst" command displays a summary of the DataWarp configuration, availability (total/free), usage (user, size, jobID, creation time, etc.). A sample of the scontrol output is shown in Diagram 1.

(Example diagram: *Named* allocations indicate a Persistent Reservation)



(Example diagram: *Named* allocations indicate a Persistent Reservation) scontrol-show-burst

STATISTICAL ANALYSIS OF WORKLOAD AT NERSC

The general NERSC workload has been previously characterized in terms of the distribution of applications [12]. Given that the focus of this paper is on the new technologies accessible in Cray XC40 systems, we will exclusively focus on how NERSC users are making use of KNL and DataWarp technologies here.

Cori Haswell and KNL nodes were fully integrated in September, 2016, and made available to users in mid-October 2017. The usage data shown here is *all* early access data in which the system was being used for a number of different acceptance, benchmarking, and user activities. The usage patterns present are expected to change in the future. NERSC has recommended the quad/cache mode to most users as a very usable and performant default mode. Based

on that recommendation, and scheduling policies which enforce it, users have used quad/cache 91% of the time since Cori was integrated. Quad/flat mode is the second most popular with roughly 7.5% of the usage, and the balance being used at extremely small percentages, mostly focused on performing specific benchmarking tasks or for rare user requests.

In the time since Cori was integrated, NERSC has progressively allowed more and more users to make use of KNL, both in a limited fashion (all users can access KNL as of March, 2017 for short, small jobs), while also admitting more and more users to "full scale" access permissions. Since NERSC has been continuously been adding users and adjusting policies it has been difficult to reliably track the present overhead of scheduling with respect to node reboots. At present, users are using the KNL nodes >97% of the possible time, with >98% of that being accessible for user codes (not rebooting). Slurm has initiated node reboots over 12,300 times on Cori since KNL was added to the system, rebooting over 160,000 nodes. The bulk of these changes were moving between quad/cache and quad/flat modes.

General access to DataWarp for all NERSC users was enabled in December 2016. For the first quarter of 2017 users averaged 500 jobs per month requesting DataWarp storage. There were 190,351 nodes with over 5PBs of storage allocation that were requested and scheduled.

Areas for improvement

Ideally there would be options to manage task layout across five levels in a KNL: socket (if multiple KNL per node), NUMA, tile, core, and thread. Options would be available to control the number of tasks at each level in the five resources tiers as well as options to manage how the task ranks are distributed across each (i.e. cyclic or block at each tier). Slurm has command options and data structures to manage only three tiers of compute resources on a node: socket, core and thread. In the case of KNL, each NUMA is mapped to a socket, which precludes full control over task layout on a compute node containing multiple KNL processors. There is also no mechanism for managing resources at a tile level. There are no current plans to address these shortcomings.

Slurm's current design assumes the number of cores per socket (or NUMA) is identical for all sockets on a node, which is not the case for all KNL processors in all NUMA modes. For example 68-core KNL processor configured in SNC4 NUMA mode will have two NUMA with 16 cores each, plus two NUMA with 18 cores each. Slurm computes a value of 17 cores for each of the 4 NUMA. Addressing this problem would require major restructuring of Slurm logic and is not currently planned. The best option today would be to disable snc4 NUMA mode on processors which can not be configured with 4 identical NUMA.

Slurm can make use of Intel's zonesort kernel module to sort pages at application start, but does not currently have a mechanism to repeat the process on a regular interval. In any case, that would in most cases best be managed by the user rather than Slurm in order to coordinate the timing of such events so as to minimize its impact upon the application. NERSC maintains a capability to do this using a custom SPANK plugin, allowing user management of this aspect of MCDRAM cache-mode maintenance.

Failure notifications from Cray's *capmc* command are detailed for some options, identifying specific nodes for which an operation failed with an explanation. The response for node reboot failures does not identify the nodes affected, which complicates failure management. Initially Slurm repeated the operation using subsets of the original list of nodes in an effort to identify the specific nodes which could not be rebooted, but that introduced scalability problems as the reboot operations would be performed sequentially on each subset of nodes. Presently Slurm requeues the job and gathers node status information in an attempt to identify and drain the faulty node and execute the job on good resources.

Presently Slurm has no time estimates for DataWarp file stage-in or stage-out operations. If that information could be made available then Slurm could potentially coordinate scheduling to achieve higher resource utilization.

Perhaps Slurm's greatest shortcoming is limitations in support of heterogeneous resource allocations. For example, one job might require 16 KNL nodes in SNC4/flat mode plus 32 Haswell nodes plus two DataWarp nodes with their CPUs and burst buffer space. This might further be complicated by different memory requirements, task layouts, etc. on each node type. The best mechanism available in Slurm today to achieve such a resource allocation is to create three separate job allocations and then merge those allocations into a single Slurm job [13]. Work has been in progress for some time to support fully heterogeneous Slurm jobs, but the time frame for deployment of this capability is not currently known [14].

Conclusion

The advanced capabilities of recent Cray systems are revolutionizing HPC systems, and the NERSC Cori system integrating all these features along with the Slurm scheduler to present a highly usable and capable combined HPC and data-intensive computing system. Intel KNL is providing unprecedented power efficiencies while still maintaining familiar programming and computing interfaces. The open source nature of Slurm is highly valuable to the HPC provider both as a vehicle for customizing the user experience as well as increasing the efficiency of communicating with Cray and SchedMD. With Slurm version 17.02, SchedMD has delivered a scheduling and resource management technology specifically tuned for

making all these technologies accessible to users in a performant, configurable, and flexible that enabling high utilization and innovation.

References

[1] https://en.wikipedia.org/wiki/Xeon_Phi

[2] D. Hensler, et. al. “Architecture and Design of Cray DataWarp” in *Proc. Cray Users’ Group Technical Conference (CUG)*, 2016.

[3] https://slurm.schedmd.com

[4] https://www.top500.org

[5] http://www.nersc.gov/users/computational-systems/cori

[6] https://cug.org/5-publications/proceedings_attendee_lists/2006CD/S06_Proceedings/pages/Authors/Karo-4C/Karo_alps_paper.pdf

[7] https://slurm.schedmd.com/intel_knl.html

[8] https://slurm.schedmd.com/gres.html

[9] https://slurm.schedmd.com/mc_support.html

[10] https://slurm.schedmd.com/core_spec.html

[11] http://registrationcenter-download.intel.com/akdlm/irc_nas/11177/xppsl_user_guide.pdf

[12] http://portal.nersc.gov/project/mpccc/baustin/NERSC_2014_Workload_Analysis_v1.1.pdf

[13] https://slurm.schedmd.com/faq.html#job_size

[14] https://slurm.schedmd.com/SLUG15/Heterogeneous_Resources_and_MPMD.pdf