Theta: Rapid Installation and Acceptance of an XC40 KNL System

Kevin Harms, Ti Leggett, Ben Allen, Susan Coghlan, Mark Fahey, Ed Holohan, Gordon McPheeters, Paul Rich
Leadership Computing Facility
Argonne National Laboratory
Argonne, IL
Email: kharms,tleggett,bsallen,smc,mfahey,eholohan,gmcpheeters,richp@anl.gov

***Abstract*—In order to provide a stepping stone from the Argonne Leadership Computing Facility's (ALCF) world class production 10 petaFLOP IBM BlueGene/Q system, Mira, to its next generation 200 petaFLOPS 3rd generation Intel Xeon Phi system, Aurora, ALCF worked with Intel and Cray to acquire an 8.6 petaFLOPS 2nd generation Intel Xeon Phi based system named Theta. Theta was delivered, installed, integrated and accepted on an aggressive schedule in just over 3 months. We will detail how we were able to successfully meet the aggressive deadline as well as lessons learned during the process.**

***Index Terms*—XC40; KNL; deployment; integration; acceptance;**

I. Introduction

In 2016 the Argonne Leadership Computing Facility [1] (ALCF) acquired a new Cray XC40, named Theta, as part of Argonne's CORAL (Collaboration of Oak Ridge, Argonne and Livermore) procurement for Aurora, ALCF's next leadership class supercomputer. Theta continues Argonne's focus on many-core processing and provides a new platform for computational scientists to prepare codes for the next generation of Xeon Phi platforms. Theta will become a resource under the Innovative and Novel Computational Impact on Theory and Experiment [2] (INCITE) 2018 program. Computational scientists from around the world will have the opportunity to compete for compute time on this platform. Theta will also be the spearhead of a new ALCF program, the ALCF Data Science Program (ADSP). This program targets non-traditional HPC applications that focus on the areas of data analysis, machine learning, deep learning and graph analytics among others. The ADSP serves to explore and develop the Theta platform for these new scientific uses.

Figure 1 shows stages of the Theta (ALCF-Lithium) project. Specifically, Theta was delivered to ALCF on June 27, 2016 and was accepted on September 30, 2016 - just over three months. However, site preparation started in March, 2016, completing with final rack connections shortly after the system arrived. Theta's acceptance period was split into 6 phases: Pre-Ship, Installation & Customization, Preparation, Early Science Testing, Functional & Performance, Stability. Each had specific entry/exit criteria whether that be milestones, target metrics, or duration.

This paper has been submitted as an article in a special issue of Concurrency and Computation Practice and Experience on the Cray User Group 2017.

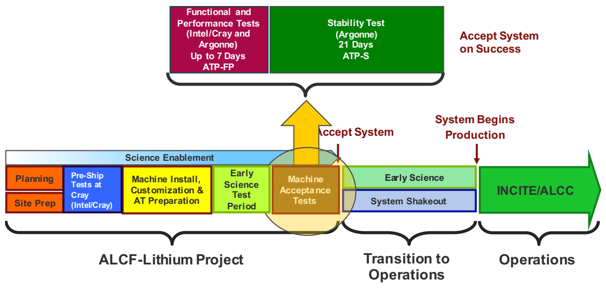

Functional and Performance Tests (Intel/Cray and Argonne) Up to 7 Days ATP-FP
Stability Test (Argonne) 21 Days ATP-S
Accept System on Success
Accept System
System Begins Production
Science Enablement
Planning
Site Prep
Pre-Ship Tests at Cray (Intel/Cray)
Machine Install, Customization & AT Preparation
Early Science Test Period
Machine Acceptance Tests
Early Science
System Shakeout
INCITE/ALCC
ALCF-Lithium Project
Transition to Operations
Operations

Fig. 1. Theta Installation and Acceptance timeline

We cover the key aspects of the physical installation process and integration into our facility in Section II. The hardware for the compute, storage, and support systems for Theta is described in Section III. In Section IV, the software management stack is examined, details of the porting and deployment of the Cobalt scheduler are shared, and various strategies to improve the systems are discussed. Finally, we look at acceptance test results and baseline performance results of a set of benchmarks and some applications in Section V, and highlight several issues encountered during acceptance in Section VI.

II. Installation Considerations

Almost every new supercomputer requires modifying or adding to the facility support services and Theta's installation was no different. Some of these were routine, such as installing new electrical wall panels, while others had unique challenges and required careful planning. Regardless of the scope, the data center facility enhancements are typically done very close to the delivery of the machine so that any last minute changes to the machine's facility requirements can be handled without modifying recently completed work. Mechanical/cooling, electrical, and flooring were facility areas modified for Theta. While some were more straight forward than others, each presented challenges and, with Theta's aggressive installation and acceptance schedule, required coordination and management. Also note that sufficient electrical and mechanical capacity for 24 cabinets was installed, although the initial delivery was 18 cabinets. Designing for extra capacity has already paid off greatly; in January of 2017, two more cabinets were added to

Theta and the installation was simple since no infrastructure needed to be installed.

Mechanical work started on March 10 and completed on June 20. The mechanical improvements included upgrading an existing water loop to provide the cooling for both Theta and our existing 10 PF IBM BlueGene/Q (BGQ) production supercomputer, Mira. Having Mira and Theta share a common water loop presented several challenges. First, the engineering team had to design for the modified loop to accomodate both the constant flow required by the BGQ direct water cooling and dynamically changing flow required by the XC40 hybrid cooling system. Because Mira's water cooling loop was designed with the necessary stubs and valves to accomodate the addition of a new water cooled system to the loop, the piping and valves for Theta's racks could be installed without any impact to the existing water loop until it was time to connect and power on the racks. It was necessary to upgrade the impellers on the pumps serving the water loop to a larger capacity to support the increased heat load. Since the pumps were originally installed in an N+1 configuration, the pumps could be upgraded one at a time without impacting normal day to day operations - though the upgrades were scheduled to coincide with regular maintenance windows to reduce risk. Another interesting challenge the mechanical designers had to overcome was the programming of the pump controls when the impellers on the two separate pumps and their corresponding capacities were different during this transition.

Electrical work started on March 24 and completed on June 20. We were able to make use of spare electrical capacity already present in the data center; however, that meant coordinating work around Mira's maintenance days and other data center tenants to reduce the impact of necessary outages. The electrical improvements were typical for such a supercomputer, including the installation of new wall panels and whips as well as two inline PDUs; however, the electrical substations were on the opposite side of the data center over 120 feet away. Therefore, a copper busway was installed overhead with 6 1200 Amp busducts, above the ceiling tiles and lights, and over existing production PDUs, networking racks, and disk racks. The actual datacenter ceiling is 20 ft high, so the workers had to use a specially constructed scaffolding that could easily be separated into two parts and moved easily among the production compute equipment while hanging the busducts from the high ceiling. Safety awareness was coordinated with all the data center tenants to ensure all were aware of the added hazards from this overhead work.

The raised floor was also upgraded to provide support for the XC40 cabinets. While that work only took several days in real time, the work was spread over several weeks to reduce the impact on other facility work in the data center. The existing floor tiles and pedestals would support a max weight individually of 925 lb while the XC40 cabinets, inline PDUs, and Sonexion disk cabinets would all surpass this weight limit. For example, the disk cabinets would weigh 4,000 lb and cover two tiles. Therefore the floor pedestals were upgraded to 2 mm thick steel and could support 9,000 lb, and the existing floor tiles were replaced with FS300 tiles with support for 5,400 lb. An identical floor upgrade was completed for the Mira installation just prior to its delivery as well.

After the new cabinets were installed, a hearing test was conducted by the Argonne Industrial Hygiene group since the noise level in the datacenter was noticeably higher. The report indicated for the area around Theta that the sound in that area measures from between 84 dBA to 93 dBA in front of Theta's blowers. ALCF has conducted tests that show baffles on the intake blower cabinets can abate some of the noise. ALCF is also considering acoustic barriers along exterior walls to abate noise leakage into the building office area.

III. Hardware Details

Theta is a Cray XC40 with self-hosted Intel Knights Landing (KNL) processors with the Cray Aries interconnect and a Cray Sonexion 3000 storage system. Figure 2 provides an overall block diagram of Theta including connections to our existing infrastructure. At the time of acceptance, Theta was composed of 18 compute cabinets, 11 cooling cabinets, 1 management cabinet, and 4 Sonexion 3000 storage cabinets with a total of 3240 compute nodes on 810 blades, 30 LNET nodes, 60 DVS nodes, 3 MOM, and 13 Tier2 nodes. In January 2017, we added two additional racks to Theta for a total of 3624 compute nodes. Table I shows the various component counts for Theta as of acceptance. We will examine the key components in the following subsections.

TABLE I
Theta node architecture

processor	Intel Xeon Phi 7230
cores/processor	64
core frequency	1.3 GHz[1]
sockets/node	1
DDR4/node	192 GiB
HBM/node	16 GiB
total node count	3240
total core count	207,360

A. Processor

All compute nodes on Theta feature the Intel Xeon Phi model 7230. This KNL [3] variant has 64 cores arranged into 32 "tiles" where each tile has 2 cores. Each core has an L1 data cache of 32KiB 8-way associative and the tile shares a single 1MiB L2 cache. Each core supports 4 SMT hardware threads and has 2 AVX512 vector units. The KNL features dynamic frequency scaling with a base frequency of the 1.3 GHz. A tile can turbo boost to 1.5 GHz while 3 or more cores can turbo boost up 1.4 GHz. The frequency can be scaled down to 1.0 GHz for power saving or thermal event responses. When the instruction stream reaches around 40% of AVX instructions the frequency drops to 1.1 GHz.

[1]See processor subsection for more detail.

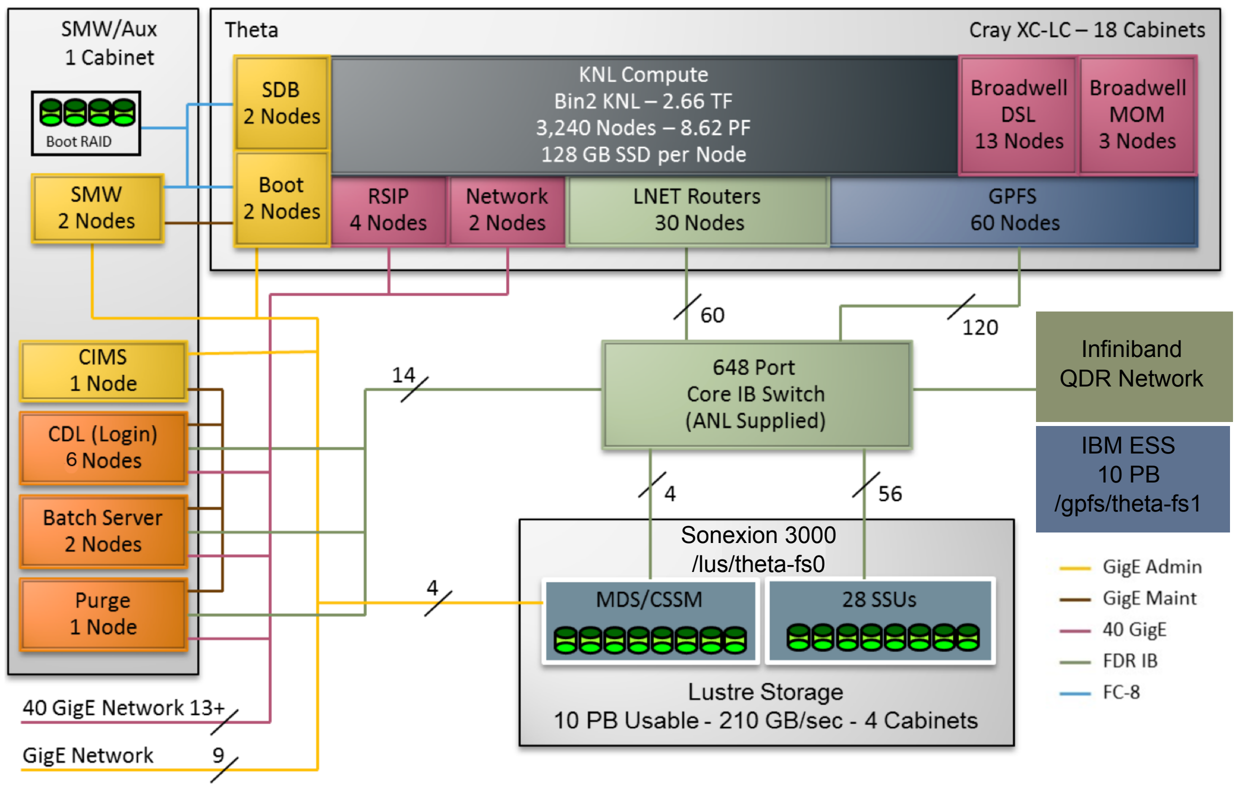


Fig. 2. Block diagram of Theta system - credit Cray, Inc.

B. *Memory*

The KNL memory subsystem features two types of memory which can be configured in several different modes. There is 16 GiB of high bandwidth Multi-channel DRAM (MCDRAM) which is located on the CPU package and has a peak bandwidth of around 450 GiB/s and a latency of 150 ns. The Theta compute nodes also have traditional DDR4 DRAM populating all 6 memory channels for around 90 GiB/s of bandwidth and 125 ns of latency. A unique feature of the KNL is a method to provision the MCDRAM in different configurations at boot time. The "memory mode" determines if the MCDRAM will be used like a last level cache for the DRAM, in which case the MCDRAM is not directly addressable. In this case, a Theta compute node has 128 GiB of main memory, as seen by Linux, and the hardware would transparently manage the MCDRAM as a direct-mapped cache layer. The memory mode can also be set to "flat mode" which provides direct access to the entire MCDRAM. In flat mode the MCDRAM appears as discreetly addressable memory to Linux, with the NUMA distance set father away then DDR4 so that the OS and runtimes do not allocate any of this memory by default, leaving it under the control of the application programmer. A combination of cache and flat modes is a hybrid mode that enables splitting the MCDRAM into portions of cache and flat, which are 25%/75%, 50%/50% and 75%/25%. In addition to setting the memory mode, the "cluster mode" can be configured. The cluster mode controls the NUMA affinity to memory controllers and the tag directory locations on the on-die mesh network. This mode can be set to all-to-all, quadrant and sub-NUMA{2,4}. During our installation, we configured the system for primarily two modes, 100% cache quadrant or 100% flat quadrant modes. Other modes were tested sparingly.

C. *Network*

1) Aries: The XC40 network is based on a high-bandwidth, low diameter topology called Dragonfly [4]. The Dragonfly network topology is constructed from a configurable mix of backplane (referred to as green links), copper (black links), and optical links (blue links). The XC40 implements the Dragonfly network with the Aries [5] interconnect ASIC. Each compute blade contains 4 Aries NICs and 1 Aries interconnect router chip. Each Aries NIC is mapped to an individual compute node and the 4 onboard Aries NICs are connected to the Aries router chip. Each Aries ASIC communicates via a standard PCI Express Gen3 16 host interface to each of the 4 compute nodes on that blade. Network requests are issued from the compute node to the NIC over the PCI bus. Network packets are typically routed adaptively, on a packet-by-packet basis, through the Aries network and therefore can leverage both minimal and non-minimal routing paths. The XC40 system has hardware and software support that allows the system to handle certain types of hardware failures without requiring a system reboot. In addition, the same technology allows for the removal and replacement of compute blades without a system reboot. However, blade warm swaps require a network quiesce that stops all high-sped network traffic temporarily, which will impact running applications. Because of this interruption we have chosen to bundle up all compute blade work for maintenance windows.

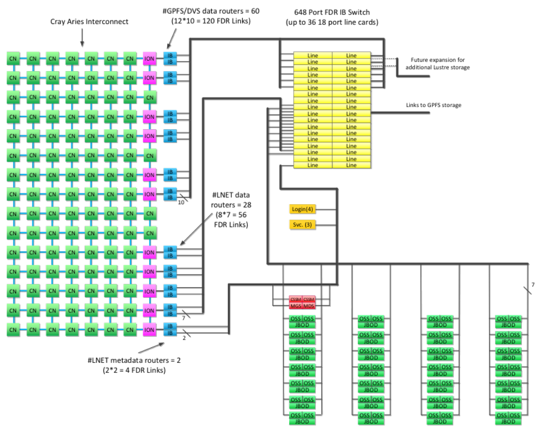
2) IB SAN: The installation of Theta had a key difference from prior ALCF system acquisitions. Unlike previous systems, where users were migrated en masse from one standalone compute and storage infrastructure to a new, replacement compute and storage infrastructure, it was necessary to provide continuous parallel computing over the same storage infrastructure used by both the Mira and Theta systems. To do this, the existing Mira storage area network, based on InfiniBand (IB), was upgraded to provide a number of features specific to the included Lustre storage system (see Section III-D), as well as to provide sufficient network capacity to fully utilize the capabilities of the existing GPFS filesystems that were to be shared between both systems. Along the way, new challenges were encountered when dealing with the limitations imposed by physically massive optical networks.

Figure 3 shows the logical configuration of the storage area network expansion supporting Theta. The arrangement of storage servers and Lustre LNET routers is based upon the Lustre Fine-Grained Routing implementation designed at the Oak Ridge Leadership Computing Facility [6]. The design was modified to allow for the use of a director-class InfiniBand switch, thus providing local switching for the Lustre filesystem as well as backbone switching for reaching the GPFS filesystems shared across multiple supercomputers.

Figure 4 shows connectivity between this new director and the existing fabric directors. By keeping Lustre traffic isolated through Fine-Grained Routing, contention is reduced between the experimental Lustre filesystem and the existing production GPFS filesystems. One important aspect of the installation, however, isn't easily illustrated in these block diagrams: the director feeding Theta is located 150 feet (46 meters) from the farthest InfiniBand-connected nodes in the Theta mainframe.

With such great distances between the director switch and connected nodes, it was necessary to utilize 50 meter long Fourteen Data Rate (FDR) InfiniBand cables. When IB cables reach this length, problems begin to arise due to the propagation delay of light in optical fiber. Similar to increasing the size of receive windows on high-latency WAN links, it is necessary to allocate larger buffers to have adequate space to receive frames delayed by the length of these cables. For InfiniBand, this is typically done by reducing the number of Virtual Lanes (VLs) available on a link. Reducing the available VLs allows each receive buffer to use a larger amount of the limited memory available on a switch or HCA.

Unfortunately, making fabric-wide changes was out of the question during the installation. With the aggressive schedule, most work had to be done live and with Mira fully operational. No disruptions could be tolerated that would impact filesystem access. Typically, reducing available Virtual Lanes is done by changing the setting on the subnet manager. However, the actual negotiated VLs are only updated when a link is first brought online. To accomplish this, individual switch port configurations were modified to reach full line-rate performance.


Cray Aries Interconnect
#GPFS/DVS data routers = 60
(12*10 = 120 FDR Links)
648 Port FDR IB Switch
(up to 36 18 port line cards)
Future expansion for additional Lustre storage
Links to GPFS storage
#LNET data routers = 28 (8*7 = 56 FDR Links)
Login(4)
Svc. (3)
#LNET metadata routers = 2
(2*2 = 4 FDR Links)
CN
ION
IB
Line
CSSM
MGS
MDS
OSS
JBOD
10
7
2
Fig. 3. Theta network diagram - credit Cray, Inc.

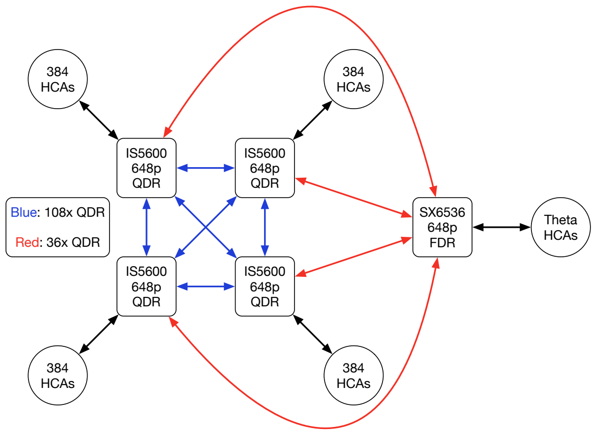


Fig. 4. ALCF Fabric Overview

The Mellanox SX6536 switch provides the ability to establish per-port Virtual Lane configurations. By reducing the number of available VLs on the ports with 50m cables, full performance was achieved without any modifications to the existing fabric. A reset of the port and benchmark testing quickly proved that this was a feasible solution for providing adequate performance to the Theta system while resulting in no impact to the ongoing work on the production Mira system.

D. Storage

The primary storage system for Theta is built using the Cray Sonexion 3000 Scale-Out Lustre Storage System summarized in Table II. Theta's Sonexion 3000 consists of 28 SSU (Scalable Storage Units), Lustre metadata servers, and management servers in four cabinets that are connected to the Mellanox SX6536 described above. All of the servers in the Sonexion are also connected to an internal ethernet management network.

There are two server types in the Sonexion 3000, servers used to serve the Lustre file system and servers used to manage the Sonexion itself. There are two Sonexion management servers configured with an Intel Xeon E5-2609 v3 CPU with six cores at 1.9 GHz and 64 GiB of RAM. There are 4 Lustre metadata/management servers and 56 Lustre data (OSS) servers. The Lustre metadata/management and data servers are based on Intel Xeon E5-2618L v3 each with 8 cores (HT enabled) at 2.30 GHz and 64 GiB of RAM.

The two Sonexion management servers, the Lustre Management Server (MGS), the first Lustre metadata server (MDS), and the two additional Lustre metadata servers (Distributed Name Space DNE/Auxiliary DNE Unit) are all installed in separate 2U form factors. The two Lustre Object Storage Servers (OSS) in a SSU are are SAS attached to a 84 drive 5U disk enclosure. There are a total of 82 SAS drives and 2 SSD drives in this enclosure. The SAS drives are 6 GB/7200 RPM drives and are configured using a parity declustered RAID [7] code called GridRAID [41 (8+2+2)]. The GridRAID configuration will present a pair of 41 disk RAID arrays (Lustre OST), one to each of the two OSS servers in the SSU. The two SSD drives are used by Lustre as journals for ldiskfs functions and are not used to directly store Lustre file system data. In the Theta file system there are 28 SSU's, for a total of 56 GridRAID OSTs, with a total of 2296 6TB SAS drives in the file system.

TABLE II
THETA PFS SUMMARY

PFS	Lustre	GPFS
Clients	3240 [2]	60
Data Servers	56	60
Metadata Servers	4	60 (shared with data server)
LNET/DVS nodes	30	60
Capacity(useable)	9 PB	8 PB
Peak B/W	240 GB/s	400 GB/s [3]
IOR	230-240 GB/s	150 GB/s [4]

Filesystems: During the initial acceptance period Lustre was also the `/home` file system but we have since transitioned to using an existing GPFS file system for `/home` while Lustre continues to provide the high performance PFS. The Theta compute nodes are configured with Lustre 2.7.1 and the OSS and MDS servers in the Sonexion are running Lustre 2.5.1. The Lustre file system has 56 OSSs setup as active-active failover pairs. Each of the 56 OSS servers is responsible for a single OST when not in a failover state. If one of the OSSs in a pair fails, the surviving OSS takes over the failed OSS's OST and serves both OSTs in the SSU. For the MDS servers, each MDS also has a single MDT. This provides for a total of 56 OSTs and 3 MDTs. Currently, only one MDS/MDT is being used as we aren't yet exploiting the Distributed Name Space feature.

During the acceptance period we also had a GPFS file system mounted on Theta which is hosted on an IBM ESS GL-4 system summarized in Table II. This system has 28 ESS-GL4 server pairs where each server pair has 4 JBOD enclosures for a total of 6496 2TB SAS disks. The GPFS file system is accessed on the XC40 nodes using Cray's Data Virtualization Service (DVS). Each compute node is a DVS client and the IO is forwarded to a 60 node GPFS cluster running on XC40 service nodes (Intel Xeon E5-2670 at 2.60GHz - 8 cores/HT enabled). The GPFS cluster in the XC40 has remotely mounted the file system from the ESS system. The GPFS file system is also remotely mounted on the eLogin nodes via a GPFS cluster that spans all eLogin nodes.

[2]Current 20 cabinet configuration has 3624 Lustre clients.

[3]Actual measured performance from Mira BG/Q.

[4]Performance limited by number of DVS nodes, number of FDR links, and current tuning.

E. External Services

In addition to the 106 service nodes in the XC cabinets, Theta also has 9 external serivce nodes: 6 external logins (eLogins/CDL), 2 batch servers, and 1 purge server. These can be seen on the left side of Figure 2 on page 3. The eLogins are the gateway login and development servers for Theta's users. The eLogins mounted all of the Lustre and GPFS filesystems described in Section III-D. During acceptance we dedicated one exclusively for external Cray personnel use, one exclusively for ALCF personnel use, and one exclusively for the test harness (see Section V-A, page 7). The remaining three were for general purpose login, compilation, and job submission activities. The batch and purge service nodes were not utilized during acceptance.

IV. System Software

A. Rhine/Redwood

To get a head start on understanding Cray's system software, ALCF staff met with both Los Alamos National Laboratory (LANL) and National Energy Research Scientific Computing Center (NERSC) staff. These meetings were to obtain an understanding of what they saw as their best practices and major issues. These meetings ranged from half-day meetings of ALCF operations staff with visitors from LANL and NERSC to multiple day visits of a single ALCF staff at LANL to witness first-hand an OS upgrade. These meetings were extremely valuable for ALCF providing ALCF an excellent starting point to begin their partnership with Intel/Cray.

Theta was delivered with UP01 of Cray's latest environment, Rhine/Redwood. Rhine/Redwood is the code name for Cray's system software stack, CLE 6.0/SMW 8.0, which was released in December 2015. Rhine/Redwood is a dramatic change in configuration management philosophy for managing Cray systems. The previous environment, Pearl/Pecos, used a shared filesystem for live configuration changes, whereas Rhine/Redwood is based primarily around the Ansible configuration management tool. At the time of Theta's install, Rhine/Redwood was extremely new with very few customers using it and many Cray engineers still being trained on its use. While the Cray install and integration team did a great job getting Theta usable in a compressed timeframe, there were some lessons learned described next.

Cray was responsible for configuring the machine for our environment after installation to be used for acceptance. While most basic configuration was straightforward, some more complex or unique site configuration, like LDAP or interface bonding, required more effort and coordination with ALCF staff. The newness of Rhine/Redwood and the accelerated acceptance schedule meant that there wasn't much time to develop best practices for configuration management in the new Rhine/Redwood environment. However, since there would be an unusually long period between acceptance and production, the emphasis was on getting a working configuration to meet our acceptance schedule. Even so, we did identify some practices that would create a more stable acceptance environment, aid in the post-acceptance knowledge transfer from Cray to ALCF, and minimize the effort required to get the machine into an operationally sustainable state.

Most importantly, it is critical to bring the various configuration management locations under revision control as early in the process as possible. Furthermore, it is important to establish a simple policy on how revision control should be used with different configuration management locations and share that with your Cray integration team ahead of time. Rhine/Redwood also includes a service called Simple Sync that allows arbitrary files to be pushed out outside of Ansible playbooks or worksheets. While this may be tempting to get a configuration file out, it should be used sparingly, if at all. It is hard, if not nearly impossible, to track who put the file in Simple Sync and why, and trivial to break configuration due to incorrect permissions or ownership. And while some configuration changes will most likely still get lost in the heat of the moment trying to get the system working for acceptance, these practices should help capture the majority of changes made to get a working system after acceptance.

B. Cobalt

The ALCF uses the Cobalt scheduler and resource manager [8], which was originally developed at Argonne and maintained by the ALCF. To maintain continuity for our users and operations staff, Cobalt was ported to the Cray Linux Environment. The port was done prior to the delivery of Theta utilizing a test system at Cray, with a goal of ensuring Cobalt was ready as soon as the Theta system was booted after install. Since Cobalt is the SRM on all ALCF production systems, which include IBM BlueGene/Q systems, general Linux clusters, and previously on the IBM BlueGene/P, and BlueGene/L platforms, it was imperative that the ability to support the current ALCF science production environment resources was maintained while extending the management capabilities of Cobalt.

Cobalt utilizes a component-based architecture to maximize reuse and reasonable consistency between different platforms. To enable this support, we extended Cobalt's component-based architecture to include a component specifically intended to communicate with ALPS as well as support job-to-resource-association mechanisms that would be specific to the Cray XC40 (Figure 5). The ALPS interface was chosen over Native mode due to the relatively short timeline leading up to acceptance, the requirement for high reliability, and the much cleaner abstraction layer. Additionally *aprun* itself would not

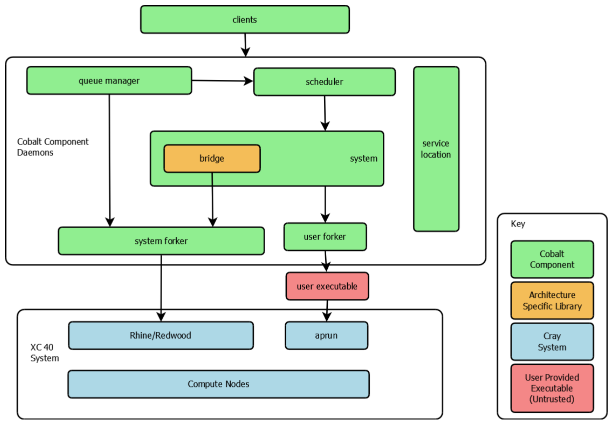


Fig. 5. Cobalt Component Structure on the Cray XC40

have to be reimplemented as it would under Native mode. Due to the way *aprun* authenticates a run to a SRM-allocated resource, additional infrastructure had to be added to the user script launcher to allow Cobalt to compensate for ALPS reservation timeouts, should job initialization take longer than expected. This also accommodates possible on-the-fly node reboots that are required to make use of different MCDRAM and NUMA modes on the KNL processors (Section III-B). These behaviors ensure that the user is able to properly authenticate to the nodes reserved for their job. Since Cobalt is replacing only the script_forker and system components, it is able to leverage all of its other functionality, allowing us to use its preexisting queue handling, advance reservation functionality, and auxiliary script plug-ins for rapid feature modification and access control.

Initial work on Cray system support began with an examination of available interfaces for the XC40 in May 2015 so that the work could be scoped and properly planned. Platform-neutral groundwork was started on the new system component in October 2015, as well as the finalization of the decision to use the ALPS interface as opposed to the Native interface. Direct tests with ALPS started in January 2016 with the arrival of a high-fidelity ALPS test simulator package from Cray. Cobalt was able to control a Cray-internal test system by the end of March 2016, and had the features known to be required for driving the Theta Acceptance Test suite by the end of April 2016. Initial integration of Cobalt with the Rhine/Redwood configuration management systems was complete by the end of April 2016 after a very successful week-long face-to-face session between ALCF and Cray developers. Cobalt was deployed to Theta, while it was in the factory, prior to delivery, by Cray in June 2016.

It should be noted that the build and stabilization of the new system component was greatly accelerated by a testing framework provided by Cray that emulates a running Cray system with different hardware configurations, as well as emulating all ALPS behaviors provided by BASIL [9]. This simulator allowed testing of communication to ALPS as well as checking behaviors as though Cobalt was communicating with a real system, and provides rudimentary scaling checks. Addtionally, access to the small testbed system with early hardware (called Kachina) combined with a week-long face-to-face meeting, gave a great opportunity to test both Cobalt's behavior against a genuine system. Moreover, it provided a way to test and integrate with the larger Rhine/Redwood management stack and its configuration management and provisioning system, and allowed us to leverage the experience that Cray had in integrating other SRMs into the Rhine/Redwood software stack. The factory installation of Cobalt helped establish greater confidence in Cobalt's functionality on the early system configuration at scale as well as giving Cray engineers some early experience running their tests through Cobalt. This early integration is considered essential to a rapid acceptance as well as rapid spin-up of early science development efforts as this not only provided a robust batch scheduler on the system from day one, but eliminated a major switch in running methods between acceptance testing and user-facing work. Because all tasks were being run via Cobalt or within a dedicated reservation within Cobalt, determining utilization and machine load was made much easier for acceptance testing. This also allowed us to leverage an Argonne developed automated testing framework that uses Cobalt for scheduling, managing, and reporting outcomes of tests (Section V-A).

There were a few differences between the Cray XC40 and the Rhine/Redwood software stack that required extra care to accommodate when compared to other systems that Cobalt supports. First is the mechanism that ALPS uses to communicate with SRMs in a language agnostic manner. BASIL is a process that is forked with custom XML-formatted requests sent to BASIL's stdin, with custom XML-responses returned via BASIL's stdout [10]. Most of Cobalt's processes are multi-threaded for responsiveness during IO-bound and remote-call work. Utilizing a POSIX fork once threads are running creates a near-certainty of deadlocks due to parent locks being set by the newly forked child process. While *atfork()* may be used to force the release of these locks, numerous libraries, including glibc, do not properly utilize this function. To work around this limitation, Cobalt has a set of single-threaded components known as "forkers" which are single-threaded shepherd daemons who actually handle forking and tracking all spawned processes, including prescripts, postscripts, and user jobs. BASIL is invoked via the *system_script_forker* for every ALPS instruction so that it may be run in a safe manner.

Additionally, we encountered an issue with "disabled" nodes and their return to service. When a Cray XC40 system is booted, nodes in distress may be put into a "disabled" state. This prevents the nodes from interfering with the boot with the side effect of the existence of these nodes being unreported by most of the Cray software stack. This causes the node to "disappear" from Cobalt on restart as the node no longer appears in the machine's inventory. When the node is later repaired and swapped in, a fairly heavyweight request for system inventory occurs to get full data on the node. Initially, this caused a hang in Cobalt as the BASIL call processing would ultimately time out, in addition to the node having to be added on-the-fly back into Cobalt's view of the machine. This hang was corrected prior to acceptance, though the node disappearance has other problems when constructing resource reservations for the full system for maintenance as the node either disappears across the restart, or the node cannot be reserved due to its invisibility to Cobalt.

Cobalt's job prescript and postscript was used to provide a mechanism for on-the fly memory mode control via CAPMC. One script checks and determines if a memory mode switch is needed and a second postscript reverts the nodes back to a default memory configuration. These were tested on Kachina. Due to the long reboot times as well as early reboot reliability concerns, this functionality was not used during the acceptance test period. A set of queues were used for each MCDRAM and NUMA mode configuration under test. Users could queue up their jobs for their requested memory modes at any time, and the the jobs would run when there were sufficient nodes operating in the desired memory mode attached to that queue. This allowed for tests of multiple configurations with a moderate amount of administrative overhead. Work is ongoing as to how to best automate these switches, particularly with respect to workloads running at 20% of the machine or larger in production, and minimizing the switch cost.

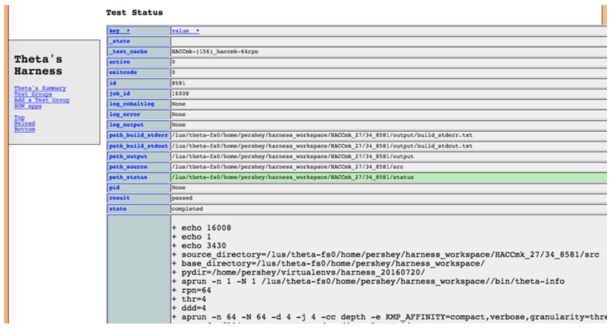
Management of user interactive shell jobs proved challenging in the eLogin environment. Due to the use of wrappers that ssh and execute commands on a net node in the eLogin environment, there were problems in *aprun* authentication to a batch-mode ALPS reservation from the eLogins. This was corrected by causing Cobalt's *qsub* command to first log the user into a MoM internal service node, invoking a sub-instance of *qsub* from there, and then allowing the interactive mode run to proceed. This interactive mode provides the ability for users to run *aprun* from their shell, but it does not place the user on the compute nodes, unlike cluster compatibility mode. This is very similar to both the restrictions and is consistent with the method of running that exists in the BlueGene environment.

Acceptance was run with an early version of Cobalt and was run using the ALPS first-fit allocator. Draining of resources for larger jobs was done either opportunistically by increasing the score of jobs or utilizing holds to ensure that large jobs ran right after machine maintenance periods, or by using Cobalt's advance-reservation mechanism to provide a similar function. Immediately after acceptance a version was deployed that provided resource draining with backfill behavior to avoid starving large nodecount jobs. This has provided good throughput as well as good utilization of Theta for early science purposes.

V. Early Results

A. Acceptance

Theta was installed in early July 2016 with an accelerated timeline. The system completed testing and was accepted on September 30, 2016. The testing stage consisted of 4 weeks of acceptance testing and a two week period for running Early Science [11] projects. We had approximately six weeks to prepare and validate our test setup prior to beginning the



Test Status

key •	value •
_state	
_test_cache	HACCmk-(156)_haccmk-64rpn
active	0
exitcode	0
id	8581
job_id	16008
log_cobaltlog	None
log_error	None
log_output	None
path_build_stderr	/lus/theta-fs0/home/pershey/harness_workspace/HACCmk_27/34_8581/output/build_stderr.txt
path_build_stdout	/lus/theta-fs0/home/pershey/harness_workspace/HACCmk_27/34_8581/output/build_stdout.txt
path_output	/lus/theta-fs0/home/pershey/harness_workspace/HACCmk_27/34_8581/output
path_source	/lus/theta-fs0/home/pershey/harness_workspace/HACCmk_27/34_8581/src
path_status	/lus/theta-fs0/home/pershey/harness_workspace/HACCmk_27/34_8581/status
pid	None
result	passed
state	completed

```
+ echo 16008
+ echo 1
+ echo 3430
+ source_directory=/lus/theta-fs0/home/pershey/harness_workspace/HACCmk_27/34_8581/src
+ base_directory=/lus/theta-fs0/home/pershey/harness_workspace/
+ pydir=/home/pershey/virtualenvs/harness_20160720/
+ aprun -n 1 -N 1 /lus/theta-fs0/home/pershey/harness_workspace//bin/theta-info
+ rpn=64
+ thr=4
+ ddd=4
+ aprun -n 64 -N 64 -d 4 -j 4 -cc depth -e KMP_AFFINITY=compact,verbose,granularity=thre
```

Fig. 6. Screen capture of Test Harness test instance

testing process. For the acceptance process, we leveraged the ALCF Test Harness [12], which helped automate the building, running, and validation of the test results, as well as provided an archive of each job's code, binaries, and results. The test harness is a combination of a web backend where a user can enter the test setup and an application backend that executes on a node within the target system. Figure 6 is sample screen shot of an individual instance of a test.

In our case, specifically, we ran the client program on one of the Theta login nodes and the backend ran on an internal ALCF virtual machine. The third part of the testing system is a Git repository that holds all of the test sources, the build scripts, and run scripts. The test harness executes a given test by checking out the source, building the executable, and then submitting the test through the Cobalt scheduler. When the test executes, the script runs the executable and upon completion validates the test's performance and/or correctness. If the test completed successfully with expected results, a zero exit code is returned by the script. The script would return an exit code 1 if the application failed with a signal, such as a crash or hardware error occurred. An exit code 2 or 3 would be returned if the correctness or performance metric failed to match what was expected. This system allowed us not only to rapidly isolate particular failures, but also identify similar problems across multiple applications or nodes.

In order to accelerate our schedule, we had to prepare these applications prior to the delivery of Theta. The application configurations and inputs were prepared on our existing machine Mira. This allowed us to establish our baseline values that could be used for comparison of the results on Theta. The build environment, however, is very different and as such we needed some other way to prepare the build portion. For this, we used NERSC's Edison system [13], which had CLE version 6 available. Edison provided us a platform to setup our build scripts and tweak the application build setups. Once Theta arrived we were quickly up to speed building and testing our applications. The final piece of combining the Cobalt scripts with the ALPS *aprun* command had to wait until Theta was available as this was the first system with both of these elements.

Once the acceptance testing processes began, it was essential to be able to isolate issues and understand failures that occurred. In order to handle this, we implemented the job failure analysis (JFA) processes that we use for our production resources. This process is followed to analyze every job failure. For any job failure, the associated standard output and error logs for the job are examined along with the key system logs. Tools have been developed that interact with the scheduler to extract all the jobs that had a failure during a given time period and then pull out events that occurred during the jobs run time. These procedures account for every job failure that occurred during acceptance. For any related failure a root cause is identified, if possible. This job failure process not only identifies all failures, but also any extreme variability between job runs of the same application. The JFA process is critical to establishing confidence in the system's behavior by providing a definitive understanding to any failure and the frequency at which failures are occurring.

The evaluation of the system was done using a combination of functional tests, benchmarks and science applications. A total of 170 major test cases were defined, the bulk of those being functional tests which covered various aspects of the machine and software stack. Here we will cover the benchmarks and the associated performance and then look at the science application scaling results.

B. Benchmarks

The performance evaluation of Theta was done using the Sustained System Performance [14] (SSP) method that was developed by NERSC for the NERSC8 acquisition. The SSP defines a performance for the overall system measured in TFLOPs. The FLOPS for each application are set using a baseline run on NERSC's Hopper system using a single node. For each application, the solver time is measured for a given number of nodes and this time is used to compute the FLOP rate. The FLOP rate is then scaled up to the full number of nodes of the system. The final system SSP is then the geometric mean of the individual benchmarks. For Theta, six applications were considered for the SSP score: AMG2013, GTC, MILC, MiniDFT, and MiniFE. Table III provides the listing of the SSP results.

TABLE III
THETA SSP RESULTS

Benchmark	Nodes	Pi
AMG2013	768	0.012061731
GTC	1200	0.049908531
MILC	384	0.056735604
MiniDFT	47	0.392259086
MiniFE	2662	0.013050658
MiniGhost	768	0.124849456
Geomean	-	0.052865462
SSP		171.2840965

The final SSP value was **171.28** TFLOP/s. For our evaluation, all of these benchmarks were run under the KNL cache quadrant configuration. For the memory bandwidth dependent applications, we would have seen an improvement by running the test case under the flat quadrant mode; however, we elected to run all of our applications under a consistent mode.

In addition to the SSP benchmarks, we ran the Sandia MPI Benchmark (SMB) message rate, STREAM Triad, and I/O benchmarks. The SMB message rate benchmark measures message rates for a given message size between pairs of end points using 8 nodes. Table IV shows the results for 8-byte and 1024-byte messages using the pair-wise method.

TABLE IV
THETA SMB RESULTS

Message Size	Message Rate
8 byte	16.1 mmps
1024 byte	7.3 mmps

The STREAM [15] benchmark measures memory bandwidth for different workloads although we just examined the *Triad* result. Given the different memory and NUMA mode configurations the KNL offers, we ran this benchmark under the different configurations although the majority of all other testing was done using cache mode. Each test was done using a single node with 64 MPI processes per node each with a single thread. Table V show the results for the different memory modes with the cluster mode being set to quadrant for all cases. The size reported in Table V is the total size reported by the STREAM benchmark for all of the arrays used.

TABLE V
THETA STREAM RESULTS

Mode	Size (GiB)	BW (GiB/s)
Flat	7.8	447.9
Cache	7.8	308.2
Hybrid (50/50)	7.8	442.5
Hybrid (50/50)	97.6	57.9

The I/O performance was evaluated using two I/O benchmarks, IOR and mdtest. The IOR [16] benchmark was used to evaluate the I/O bandwidth of the Lustre file system using an optimal workload to get the maximum performance.

TABLE VI
THETA IOR RESULTS

Command	Nodes	Write BW	Read BW
`-a POSIX -e -g -[w|r] -t 64M -b 12g -F -k -E -o <file>`	1792	247 GB/s	235 GB/s

The mdtest [17] application evaluated metadata rates in terms of operations per second for different types of operations. The mdtest application run across the entire machine testing the Lustre volume. The Lustre volume was configured with no DNE support, so this test is measuring the rate of a single MDS/MDT. Mdtest is run with the following arguments: *-i 1 -d /lus/theta-fs0/... -n 10*.

TABLE VII
THETA MDTEST RESULTS

Operation	Rate (ops/s)
Directory creation	24129.947
Directory stat	133717.879
Directory removal	11199.838
File creation	43459.599
File stat	130676.940
File read	93707.416
File removal	29923.935
Tree creation	194.406
Tree removal	42.541

C. Applications

Along with various benchmarks, we tested application codes. These applications were primiarily used for testing and validation of the compilers, libraries and hardware rather than performance. Table VIII lists the applications tested. We decided to test all applications in cache mode as we felt this would be the primary usage model with the system initially. However, although we used cache mode, we sized the input sets to fit within 16 GiB of MCDRAM. This was done to prevent odd artifacts in the strong scaling test cases where a dataset could from residing in DDR to fitting completely within MCDRAM. This resulted in a very small amount of work per core at the large scale process counts.

TABLE VIII
THETA SCIENCE APPLICATIONS

Application	Description	Small (nodes)	Large (nodes)	Scaling factor
AMG2013	Parallel algebraic multi-grid solver	216	3072	14.2x
CAM-SE	High-Order Methods Modeling Environment for atmospheric climate modeling	256	1024	4x
HACC	N-body methods for formation of structure in collisionless fluids under the influence of gravity in an expanding universe	96	3240	33.75x
HSCD	Turbulent combustion	4	128	32x
LAMMPS	Molecular Dynamics	128	2048	16x
NEKbone	Computational Fluid Dynamics with spectral element method	216	3240	15x
MILC	Quantum Chromodynamics (su3_rhmd_hisq)	128	3072	24x
QBOX	First-principles Molecular Dynamics	216	3240	15x
QMCPACK	Continuum Quantum Monte Carlo	216	3240	15x

Figure 7 shows the relative scaling efficiency for each application with the “small” size being the baseline. This demonstrates the “out-of-the-box” performance with no developer time spent tuning these applications on the system.

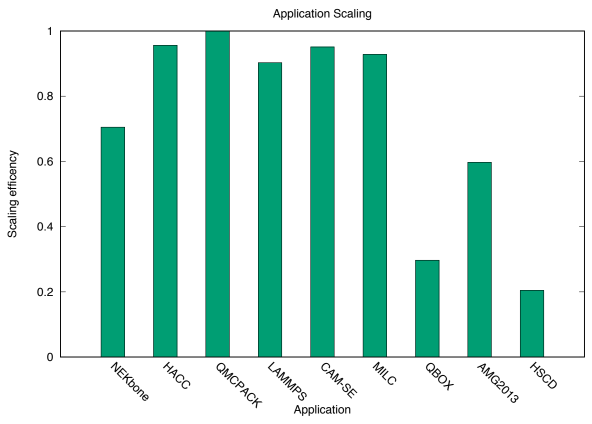


Fig. 7. Application scaling

VI. Challenges

As with any new machine, especially one of the first of its kind with hardware and software, problems are discovered after installation. Theta installation and acceptance was no different. One of the interesting things about our system is that it was purchased from Intel as the prime, with Cray as the integrator. This was a new situation for all parties involved, and a significant concern for ALCF was how we would work together during the deployment and acceptance phases when the usual stress from tight deadlines and early system unknowns would begin to arise. It took a few weeks, but by the judicious use of targeted calls, explicit building of trust between the three partners, and ensuring that we had a strong ALCF/Cray/Intel on-site team working together in a dedicated war room next to the data center stocked with snacks and sodas, and fancy coffee and tea makers, we were able to perform some of the fastest debug and fixes of very complicated problems that we've experienced in the ALCF deployments to-date. Here we highlight a few of the challenges we encountered and how they were resolved during the accelerated schedule.

A. MILC Failures

MILC [18] is a QCD application that is heavily used on our machines. It is also useful in stressing machines at scale in ways that typical diagnostic programs don't. Because of this MILC is a staple in our acceptance test suite. During the final preparations for acceptance, we were seeing approximately 50% failure rates of 3072 node MILC runs due to a kernel panic. Smaller sized jobs did not exhibit this and the time to failure was not consistent. The code and inputs were provided to Cray and Intel. Finding the root cause proved difficult for 3 reasons. Job size - at the time, Cray only had one internal KNL machine large enough to run the application. The time to failure - the average time to fail was almost 2 hours. Most problematic was that adding any instrumentation to discover where in the kernel the failure was occurring caused the failure to completely go away. A great deal of credit needs to be given to Intel and Cray who worked tirelessly together over a 2 week period, eventually discovering a micro-code issue and developing a fixed BIOS.

B. Science Tests Readiness

A great deal of work was required to prepare the science and benchmark codes for the acceptance tests. While early preparations on other systems went well, the delivery and acceptance test preparation of Theta overlapped with other high-priority ALCF activities. We had failed to gain a commitment from the production side of enough people familiar with the science codes and the test harness to adequately prepare. As a result, the final development of the input decks and test harness configurations were left to a very small set of people, most of whom were not familiar with the science codes. This led to some very late nights and weekends of scrambling to get the codes tested and updated to run properly at scale, risking the burnout of several deployment team members before the acceptance test had even begun. For the next deployment, we will ensure there is larger set of people, including a lead for each of the science codes, who are dedicated to the preparation work regardless of what else is happening at the ALCF.

C. Early Science Test Period

One of our important activities prior to starting acceptance is the Early Science two week test period. During this period the Early Science projects are allowed onto the system and tasked with running their codes at scale. We started this with the Mira deployment, where we utilized it to track down and debug some significant problems on the machine that would otherwise not have been found until after acceptance. Because of that work, we were able to begin running science at scale on the day Mira's acceptance was completed and to move to production ahead of schedule. For time reasons, this period was scheduled for Theta immediately prior to the start of the acceptance test. Unfortunately, that was exactly the time we needed to do a number of quick fixes requiring reboots and major patch updates, which was frustrating to the users, Cray, and the deployment team. In the future, we need to ensure a minimum of a week between the early science test period and the start of acceptance.

D. PDU Commissioning

Two power distribution units for the Sonexion and XC-40 management racks were commissioned prior to the delivery of Theta. ALCF staff reviewed the commissioning report and found numerous errors and inconsistencies. Accurate commissioning reports are vital to ensure the safety of our staff and proper operating environments for our hardware. The errors discovered included numerous invalid voltage readings which, if accurate, might have damaged equipment. While these appeared to be data entry errors, they called into question the methodology and diligence of the contractor involved. The ALCF immediately rejected the findings and requested that the contractor send a new technician to perform a new commissioning report. After some lengthy deliberation, the contractor remedied the situation, performanced a new commissioning, and produced a valid report. One lesson-learned from this experience is that such work by contractors ought to be observed by ALCF subject-matter experts. This would perhaps help to reduce the schedule impact of any discovered issues.

E. Flat-Quad Performance

Early in the preparation phase we compiled and ran all of the SSP tests to ensure that they completed and met their anticipated performance targets. The preliminary results were inline with results achieved at the factory. However, once the functional and performance phase (ATP-FP) had begun, the tests meant to be run in the flat-quad memory mode were performing about twice as slow as they should. Working with Cray and Intel we were able to determine that a specific patch set that had been applied during the integration phase to fix an unrelated issue was the cause. Normally the fix would be simple - back out the problematic patch set and rerun. Unfortunately there were multiple challenges. First, because ATP-FP had officially started, no changes to software or configuration were allowed. If any needed to be made, ATP-FP would have to be restarted with all already completed jobs having to be rerun. These flat-quad jobs were run at the end of ATP-FP so this would have added at least 3 more days to the schedule to rerun everything again. Second, the problematic patch set was necessary to fix a blocking issue so a new patch needed to be developed to fix the performance issue. Cray quickly developed a patch and tunings for these benchmarks to run fast enough under the cache-quad memory mode to achieve our SSP targets. Once ATP-FP was completed, we briefly booted with the patch to verify that the problem was solved and then returned the machine to the official acceptance configuration. These performance tests have been added to the ALCF regression test suite used to exercise the system after each major installation.

VII. Conclusion

ALCF received, installed, and accepted a Cray XC40 named Theta in just over three months from late June 2016 to September 2016. ALCF did a significant amount of prepratory work from facility enhancments (busduct, increased weight capacity, water loop, and even cable trays) to designing and installing the IB SAN for the Sonexion 3000, and a GPFS filesystem to porting the Cobalt scheduler that was ready for Theta day one. This prepratory work made it possible to begin testing benchmarks and applications soon after power on. There were several issues that were identified, some that required a lot of expert work by ALCF, Intel, and Cray (during and after acceptance.) Regardless, acceptance was started just 2 months after delivery and after a 4 week acceptance period, Theta was accepted.

Acknowledgment

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

We would like to recognize all the ALCF staff who contributed on the rapid integration efforts and acceptance testing.

We would like to thank the following Cray personnel for their extraordinary efforts: Joe Glenski, Richard Walsh, Clark Snyder, Jon Bouvet, Mike Solari.

We would like to thank Jacob Wood and the Intel system debug team for their extraordinary efforts.

References

[1] [Online]. Available: http://www.alcf.anl.gov
[2] [Online]. Available: http://www.doeleadershipcomputing.org
[3] A. Sodani, "Knights Landing: 2nd generation Intel Xeon Phi processor." Hot Chips '15, 2015.
[4] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ser. ISCA '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 77–88. [Online]. Available: http://dx.doi.org/10.1109/ISCA.2008.19
[5] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, "Cray XC series network." [Online]. Available: http://www.cray.com/sites/default/files/resources/CrayXCNetwork.pdf
[6] D. A. Dillow, G. M. Shipman, S. Oral, and Z. Zhang, "I/O congestion avoidance via routing and object placement," in *Proceedings of Cray User Group Conference (CUG 2011)*, 2011.
[7] M. Holland and G. A. Gibson, *Parity declustering for continuous operation in redundant disk arrays*. ACM, 1992, vol. 27, no. 9.
[8] COBALT: Component-based lightweight toolkit. [Online]. Available: http://trac.mcs.anl.gov/projects/cobalt
[9] *basil(7) Miscellaneous Information Manual*, April 2016.
[10] *apbasil(1) General Commands Manual*, June 2012.
[11] "ALCF Theta Early Science Program." [Online]. Available: https://www.alcf.anl.gov/early-science-theta-cfp
[12] E. Pershey, "Automated testing of supercomputers." [Online]. Available: https://www.alcf.anl.gov/files/AutomatedTestingofSupercomputers_0.pdf
[13] "Nersc edison system." [Online]. Available: http://www.nersc.gov/users/computational-systems/edison/
[14] "Sustained system performance." [Online]. Available: http://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/ssp/
[15] J. D. McCalpin, "Memory bandwidth and machine balance in current high performance computers," *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pp. 19–25, Dec. 1995.
[16] "Parallel filesystem I/O benchmark." [Online]. Available: https://github.com/LLNL/ior
[17] "mdtest." [Online]. Available: https://github.com/MDTEST-LANL/mdtest
[18] "MIMD Lattice Computation (MILC) Collaboration." [Online]. Available: http://physics.indiana.edu/~sg/milc.html