Title
The NERSC Cori HPC System

Authors
Austin, Katie Antypas Brian
Bard, Deborah
Bhimji, Wahid
et al.

Publication Date
2019-05-08

DOI
10.1201/9781351036863-11

Peer reviewed

Chapter 11

The NERSC Cori HPC System

Katie Antypas Brian Austin, Deborah Bard, Wahid Bhimji, Brandon Cook, Tina Declerck, Jack Deslippe, Richard Gerber, Rebecca Hartman–Baker, Yun (Helen) He, Douglas Jacobsen, Thorsten Kurth, Jay Srinivasan, and Nicholas J. Wright

NERSC, Lawrence Berkeley National Laboratory

11.1 Overview

11.1.1 Sponsor and Program Background

The National Energy Research Scientific Computing Center (NERSC) is the mission high performance computing facility for the Department of Energy's Office of Science (DOE SC). NERSCs primary goal is to accelerate scientific discovery for the DOE SC workload through high performance computing, related technology development and data management and analysis. Toward this end, NERSC provides large-scale, state-of-the-art computing, storage and networking for the DOE SCs unclassified research programs.

With thousands of users from universities, national laboratories and industry, NERSC supports a large and diverse research community. The NERSC workload represents the wide variety of research performed by its users, including simulations that run at the largest scales. In addition, the analysis of massive datasets is becoming increasingly important at NERSC in support of DOE SC experimental facilities. Over the years, NERSC's resources have contributed to ground breaking science. Five of NERSC's users have won the Nobel Prize and each year over 2000 publications are attributed to using NERSC's systems.

The NERSC Center was founded in 1974 at Lawrence Livermore National Laboratory and became the first unclassified computing center. At the time, the Center was focused on fusion energy research. In 1983 the Center's focus expanded to support high performance computing to all the programs in what is now the Department of Energy Office of Science. In 1996 NERSC moved to Berkeley Lab and became part of the Computing Sciences program. In 2000, NERSC outgrew its location on the main Berkeley Lab campus and temporarily moved to the Oakland Scientific Facility while a new facility was built to house the supercomputers and staff. More than a decade later, in 2016, NERSC moved back to the main Berkeley Lab campus into a state-of-the-art energy efficient building.

The Cori system, a 30PF Cray XC system, described in detail in this chapter, was designed to support data intensive workloads and be a platform that would begin to transition the user community to more energy efficient architectures. Preparing for and deploying the Cori system [7] in a brand new facility was a top priority for NERSC in 2016.

11.1.2 Timeline

The Cori system was procured as part of the Alliance for Application Performance at Extreme Scales (APEX), a partnership between Berkeley Lab, Los Alamos National Lab and Sandia Lab. The intention of the collaboration was to procure two systems in the 2016 time frame that would begin to serve to transition the user community to more advanced architectures on the path to exascale. One system would be installed at LANL, in partnership with Sandia and the other would be installed into the new Wang Hall facility at Berkeley Lab for NERSC. The systems were procured through a competitive process and Cray was selected as the vendor for the Cori system. The Cori system was delivered in two phases. The first phase, known as the Cori Data Partition, or Cori Haswell, was delivered in the fall of 2015 and was the first system installed into the new Wang Hall facility. The second phase of Cori, the KNL partition, was delivered in the summer of 2016 and installed and integrated into NERSC later that year. The system was accepted in 2016. Early users from the NERSC Exascale Science Application Program (NESAP) were the first enabled on the system and given priority access to the system until it transitioned into production on July 1st, 2017.

11.2 Applications and Workloads

As the mission high performance scientific computing facility for the Office of Science in the United States Department of Energy, NERSC supports research in a wide variety of disciplines, including climate modeling, research into new materials, simulations of the early universe, analysis of data from high-energy physics experiments, investigations of protein structure, and many others.

NERSC has a diverse user base of over 6,000 users in more than 700 projects running more than 600 different codes. However, the top ten codes make up 50% of the workload, and the top 25 codes make up more than 2/3 of the workload. But even these 25 codes are quite diverse in both application area as well as execution characterization. Codes run at NERSC utilize all seven original motifs (Dense Linear Algebra, Sparse Linear Algebra, Computations on Structured Grids, Computations on Unstructured Grids, Spectral Methods, Particle Methods and Monte Carlo) and more.

11.2.1 Benchmarks

Benchmarking is essential for ensuring that HPC systems provide correct results and satisfactory performance, both upon installation and throughout the lifetime of the machine. [29] Benchmarks may be designed to test specific aspects of system performance (i.e., the STREAM micro-benchmark measures memory bandwidth) or the performance of a complete scientific application. While NERSC routinely performs and monitors the performance of several micro-benchmarks, we emphasize the importance of application benchmarks (over micro-benchmarks and kernels) because application performance is a direct indication of the experiences of our users and requires balanced integration of the entire stack of system hardware and software stack. By focusing on applications rather than micro-benchmarks, the risk of tuning the system for niche uses that do not benefit real simulation codes is avoided.

The APEX benchmark suite [6] used to evaluate Cori consists of eight applications:

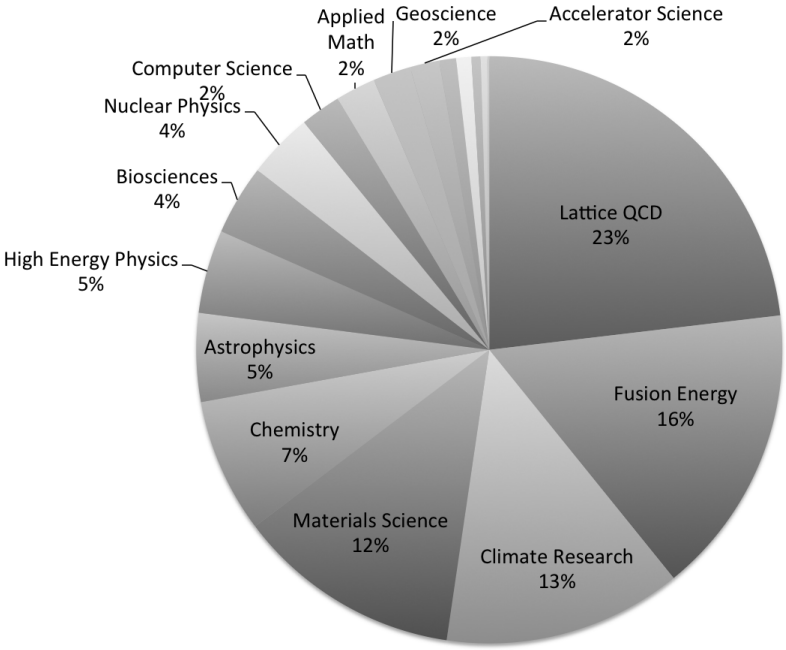

FIGURE 11.1: NERSC 2016 usage by science category.

- *GTC:* 3D gyrokinetic particle-in-cell simulation of Tokomak fusion devices
- *MILC:* Lattice Quantum Chromodynamics
- *MiniDFT:* Plane-wave density functional theory
- *MiniFE:* Finite element generation, assembly and solution of an unstructured grid problems
- *MiniGhost:* 3D finite difference stencil
- *AMG:* Algebraic multigrid solver for linear systems
- *UMT:* 3D, deterministic multigroup photon transport simulation on unstructured meshes
- *SNAP:* discrete ordinates neutral particle transport

The first three of these codes (GTC, MILC and MiniDFT) were carefully selected to represent a cross-section of the NERSC workload in terms of node-hours used, algorithmic diversity and coverage of scientific domains. The remaining applications were selected by other members of the APEX partnership and are similarly important to their workloads, but also relevant to NERSC. (For example, the stencil algorithms and communication patterns exercised by MiniGhost appear in many NERSC codes as well.)

The Sustained System Performance (SSP) is the primary metric used to evaluate Cori's performance. SSP is simply the average (geometric mean) of the FLOP rate achieved by each application in the benchmark suite, scaled to the full size of the size of the system.[29] Cori's SSP, using the benchmarks, is 562 GF/s, a 10.8× improvement over Hopper, the Cray XE6 system that preceded it.

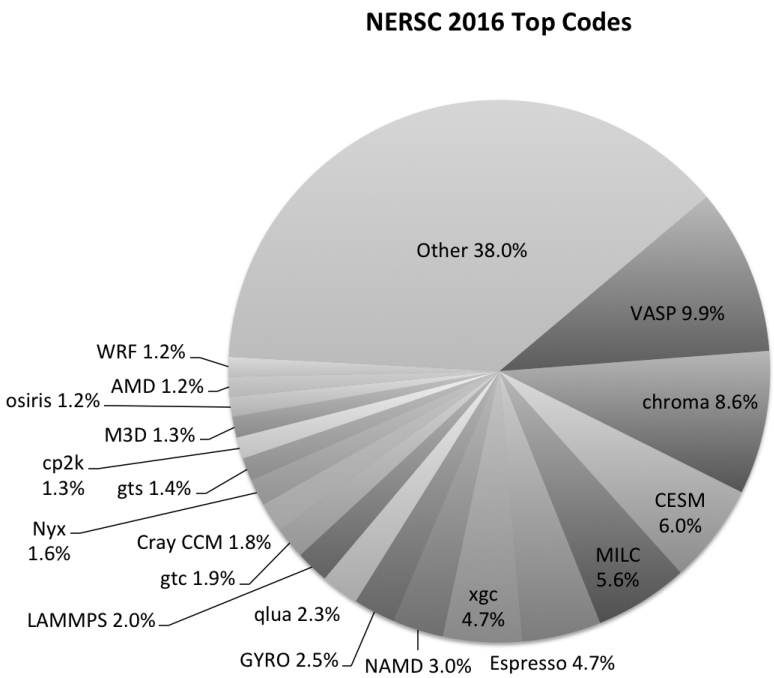

FIGURE 11.2: Top 20 codes used at NERSC in 2016.

11.3 System Overview

NERSCs newest supercomputer, named Cori after Nobel prize winning biologist Gerty Cori, introduces the NERSC user community to the advanced energy efficient architectures necessary to reach exascale levels of performance. Cori is a Cray XC40 system made up of three primary components. The main system, a Cray XC40, is the compute pool, the external login nodes allows users to access the system, and the scratch storage, a Cray Sonexion 2000.

The XC40 has two compute pools consisting of 9,688 single socket nodes with Intel Xeon Phi Knights Landing (KNL) manycore processors. Cori also has 2,388 dual-socket Intel Xeon Haswell nodes on the same Cray Aries high-speed interconnect network. This mix of node types makes Cori a novel system at this scale in HPC. In addition, the system has a Burst Buffer, a schedulable, fast, flash-based, intermediary storage pool between the compute pool and the Lustre file system, The burst buffer has a capacity of 1.6 PB and a bandwidth of over 1.5 TB/sec. The storage system is a 28 PB Lustre file system contained in 18 cabinets of Cray Sonexion 2000 racks providing over 700 GB/second of IO performance.

11.4 Hardware Architecture

The system design of the XC40 is similar to previous Cray architectures. Compute nodes are placed 4 to a compute blade, 16 blades make up a chassis and there are three chassis per cabinet, leading to a maximum of 192 nodes per cabinet. The cabinets are cooled transversely with air that passes through a water-cooled radiator. Cori can use inlet water up to 75 degrees.

11.4.1 Node Types and Design

There are two basic blade types, compute and service. Each compute blade has 2 processor daughter cards each of which contains 2 nodes. The compute nodes contain either an Intel Haswell or Intel Knights Landing processor.

11.4.1.1 Xeon Phi "Knights Landing" Compute Nodes

These single socket nodes contain an Intel Xeon Phi "Knights Landing" (KNL) Many Integrated Cores (MIC) processor. This features 68 cores per node with 4 hardware threads each, 512 bit vector units, 16 GB on-package MCDRAM (Multi-Channel DRAM) high bandwidth memory and 96 GB of DDR4 memory (6 channels).

The peak performance of each node is approximately 3 Tflops, and measured DGEMM performance is 2.1 TF/s. The MCDRAM memory can sustain 415 GB/s on STREAM triad and the DDR memory can sustain 90 GB/s. The two different memory types on the KNL processor allow different configurations of the memory (which can be changed with a reboot). The node can either be configured with the MCDRAM as a cache for the DDR4 memory, or with a flat memory space covering both memory types. If the MCDRAM is configured as a cache, a performance penalty is incurred and the STREAM triad bandwidth drops to 315 GB/s.

In order to obtain good performance from the KNL nodes, it is essential to use many of the 68 cores to make full use of the vector units and to use the MCDRAM as much as possible.

11.4.1.2 Xeon "Haswell" Compute Nodes

These dual socket nodes contain two 16 core Intel Xeon E5-2698 v3 "Haswell" processors running at 2.3 GHz and 128 GB of DDR4 memory running at 2133 MHz. Each core supports two hyperthreads and has two 256-bit wide vector units. Each core has a peak performance of 36.8 Gflops, making the peak performance of the node 1.2 TFlops. Each socket supports 4 memory channels giving a peak memory bandwidth of 137 GB/s and a STREAM triad bandwidth of 115 GB/s.

11.4.1.3 Service Nodes

The service blades have 2 nodes per blade. Each contains an Intel Xeon "Sandy Bridge" processor and has 2 PCIe gen3 slots available. Cori has several different types of service nodes: network, bridge (used for the Software Defined Networking (SDN) connections), Data Virtualization Services (DVS), DataWarp (burst buffer), Lustre Router (LNET), and Slurm controller nodes. What the node will be used for determines what type of HBAs are installed. The network, Slurm controller nodes and bridge nodes have 2 dual Mellanox (QDR-IB) cards installed that are connected using bonded vlans to the NERSC local internal and external

networks. The internal network is used primarily for access to storage and external is used for LDAP, and general connections from outside the NERSC environment. The DVS nodes have one IB and one Ethernet card installed. The InfiniBand provides fast access to the NERSC global file systems and the Ethernet is used for the management connections to the NERSC global file systems. The LNET nodes have two Mellanox InfiniBand HCAs for access to Coris scratch file system.

11.4.2 Interconnect

The System uses a Cray proprietary network interconnect, referred to as Aries, in a Dragonfly topology. The Aries network interconnect is comprised of Aries chips and cabling. The Aries chip is comprised of four Network Interface Controllers (NICs) and a 48-port tiled high radix router. The NICs and router are all on a single die. The entire Cray XC interconnect consists of a direct network of Aries chips using copper and optical cabling. The Aries network interconnect connects all nodes (Compute, Service, and the Cray Burst Buffer if the option is exercised) in the Cray XC system. Each compute node connects to the Aries network via PCI express, at a peak bandwidth of 16 GB/s.

The dragonfly network has 3 levels. The lowest level connects each node in a chassis to each other, the second level connects each chassis in a pair of cabinets together, finally each cabinet pair is connected with optical cables to form the third level of the dragonfly. This leads to a maximum of 5 hops between any pair of nodes in the system. The peak bisection bandwidth of the machine is 45 TB/s. The Aries interconnect has numerous adaptive routing features designed to avoid congestion and hot spots.

11.4.3 Storage - Burst Buffer and Lustre Filesystem

The Burst Buffer (Cray DataWarp) is made up of service nodes that are peers on the Aries network with the compute nodes. Each contains 2 Intel P3608 SSD cards that provide 3.2 TB of available storage per card. The Burst Buffer is configured to use the Cray Data Virtualization Service (DVS) software to handle I/O transactions. In addition, numerous features to manage allocations of space on a per-job or persistent basis are provided by the Cray burst buffer service, which are accessible via the job scheduler.

The system also has a 30 PB Lustre file system based on the Cray Sonexion 2000 with 124 scalable storage units (SSUs) and 2 additional Distributed Namespace (DNE) units to provide both fast data paths and large storage capacity. Four additional metadata servers to help spread the metadata load increasing performance.

11.5 System Software

11.5.1 System Software Overview

The system software stack on Cori consists of several closely inter-related pieces: an independent operating system for each of the nodes in the system, management software for the collection of nodes that make up the system, and software for operating and managing system components, including the storage and compute resources as well as other services (login nodes external to the Cray Aries network, network gateway nodes, workload dispatch nodes, etc.). This section presents an overview of all this software as well as details on

specific functionality and customizations that NERSC has enabled in order to support the workload that runs on Cori.

The Operating System on Cori is a variant of Linux. The compute nodes of the system run a modified version of the Linux kernel called Compute Node Linux. The operating system itself is based off the SuSE distribution and is, on external (user facing) and service nodes, the SLES software stack. Since the compute nodes on Cori are heterogenous, with both Knights Landing (KNL) nodes as well as Haswell (HSW) nodes, each of these variants of the compute nodes have different node images loaded on them. The service nodes, which include the (external) login nodes, compute gateway nodes, storage service nodes (including LNET nodes to talk to the Lustre filesystem servers), DVS nodes (for projection of the GPFS filesystems on to the compute nodes), and other management nodes for the system have a different full-featured SLES-based image loaded on them.

11.5.2 System Management Stack

The Cray system management software consists of two parts, the System Management Workstation (SMW) software and the Cray Linux Environment (CLE) software. The system software stack is pictured in Figure 11.3 below. The function of the system management

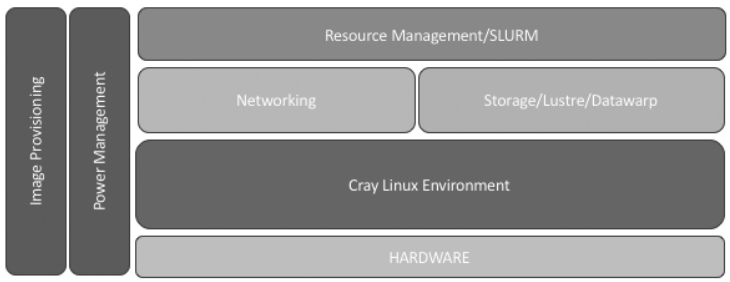

FIGURE 11.3: The system management software stack on Cori.

software on a Cray system is to generate system images for each of the different types of nodes on the system, deploy (boot) the nodes, configure them appropriately for the services they will provide and then, collect overall information about the system, including health information, performance information, etc. In addition, the system management software allows for managing nodes on an ongoing basis, including setting of various power states and rebooting of nodes.

NERSC's vision of the system management stack hews closely to this description, but expands on it by applying it in a uniform fashion to multiple systems. NERSC operates four very similar Cray systems, two large scale systems (Cori and a previous generation Cray system Edison) and two testbeds, one for each large scale system. Though there are some differences in hardware and scale, the intention is to manage them in as similar a way as possible. The system management methodology followed [27] allows NERSC to utilize the same system configuration method (Ansible) to operate all the Cray systems, with the software determining the appropriate configuration to apply to each system.

11.5.3 Resource Management

Within Cori, Intel's Knights Landing processors with High-Bandwidth memory (MCDRAM) allows each node to be configured in twenty different configurations or "modes" [26].

While the vast majority of our workload can be run using one or two of these configurations, being able to provide users with the ability to select configurations and to quickly provide access to the appropriately configured resource is critical to the efficient operation of the system. NERSC utilizes the SLURM workload manager to provide users access to compute resources, and, in combination with Cray's DataWarp software, to associated Burst Buffer resources. We have worked extensively with SchedMD [28] to design and develop SLURM functionality that allows us to:

- Efficiently schedule and run a highly diverse workload on Cori, with job sizes spanning a single core to the entire 10,000+ nodes on the system
- Run jobs on an existing configuration of KNL nodes of varying modes and HSW nodes.
- Run single jobs across KNL and HSW nodes
- Quickly allocate and configure KNL nodes into a new mode (which requires a node reboot)
- Provide access to all the different storage resources (described below) for user jobs, if necessary, on demand

11.5.4 Storage Resources and Software

The Cori system supports both traditional HPC simulations as well as data analysis workloads. This spectrum of user needs requires an equally broad range of storage resources. Being able to provide the right kind of storage for jobs is a necessity if the system is to be utilized effectively. The Cori system provides users four different kinds of storage resources:

- The system-local, high-performance Lustre "scratch" filesystem.
- Center-wide, GPFS filesystems, providing home directories shared amongst all our systems, "Project"-specific storage space as well as "common" space for frequently and widely used libraries and programs.
- The I/O acceleration storage layer or "Burst Buffer" provided by Cray as the DataWarp system backed by the Lustre scratch filesystem.
- "Node-local" storage, created on demand on the scratch filesystem for jobs using Shifter.

All of these different storage resources can be seamlessly accessed by all jobs on the system. In addition, the Lustre filesystem is "external" and can be available to users even if the compute nodes are not. This is accomplished by use of Lustres LNET protocol and the use of dedicated LNET servers, allowing compartmentalized access to the storage system from the compute nodes of Cori, the external login nodes of Cori, the Cray XC-30 "Edison" system as well as dedicated Data Transfer Nodes (DTNs) optimized and used for wide-area network traffic and grid-based data transfer.

NERSC staff have also worked closely with Cray and SchedMD to develop the DataWarp software to support the needs of our Data-Intensive users. In addition to being able to request DataWarp storage on-demand per job, users are able to pre-stage data and store data on the resource both for the job lifetime alone as well as across multiple jobs. Future functionality being developed includes the ability to use the DataWarp layer as a transparent cache to the Lustre backing store.

11.5.5 Networking Resources and Software

Cori provides a high-performance network (Aries) between the compute nodes and between the compute and service nodes, resulting in a well-integrated system that allows for efficient communication within processes and for I/O traffic. For effective support of data-intensive workloads, however, external network connectivity at reasonably high-bandwidth and low latency is required. Crays standard solution for traffic out of the Aries network has been to use a combination of Realm-Specific Internet Protocol (RSIP) servers and traditional network gateways. For performance and reliability reasons this solution is not optimal for the NERSC workload. To allow for greater flexibility in shaping external traffic both in to and from the compute nodes of the system, we have replaced the RSIP nodes with bridge nodes that send and receive the traffic to and from software-based routers (Figure 11.4). The bridge nodes (each configured to handle a subset of the compute nodes) allow for communication between the Aries network and external Ethernet networks. In addition, to improve resilience, the bridge nodes can easily be configured to serve a different set of compute nodes in case of failure.

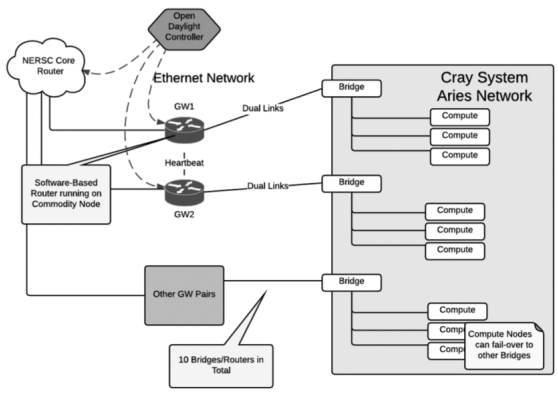

FIGURE 11.4: RSIP replacement configuration. (From [22].)

NERSCs vision of networking to Cori (and to Cray systems in general) is to provide our users with a fully software-defined network that can handle the needs of our most demanding data ingest requirements (such as the Light Sources) to configure and shape the network seamlessly between resources external and internal to the system.

11.5.6 Containers and User-Defined Images

The use of container technology has rapidly developed over the last few years to enable greater control of the system environment by users. Users, especially those who develop or use specialized software for simulation or analysis that is closely tied to a specific operating system environment, have found this to be an indispensable way to migrate their workload to different systems without explicitly porting the software over. To meet the needs of these users, we have investigated the use of Docker-like containers on Cori. Containers have, however, not been widely used in HPC environments for various reasons including security concerns with allowing user-defined environments in tightly integrated systems and a lack of easy to use tools to deploy them in such environments.

NERSC, in collaboration with Cray, developed a tool called Shifter [27], that allows the creation and distribution of Docker-like images on to the Cray platforms, including changes

to support the use of the global filesystem, addressing security concerns of deployment of the user-defined images in a production environment, and most importantly, integration with the workload manager. These features allow users to generate images and simply submit a job that requests the image, which allows them to quickly use existing codebases to run jobs at very large scale on Cori. Users from light sources such as LCLS at SLAC and from high energy physics facilities such as the LHC have successfully utilized Cori this way [27].

In summary, a well-integrated, easy to deploy, manage and use system software stack is critical to the efficient operation and use of a large complex system such as Cori. NERSC's philosophy of not just taking what is provided with the system, but carefully picking the most useful parts of the stack, modifying it if needed, and adding functionality of interest to staff and users allows us to deploy a stable state-of-the-art system, both useful to users and one that allows for continuous improvement over its lifetime.

11.6 Programming Environment

11.6.1 Programming Models

The majority of existing HPC applications deployed on NERSC resources use MPI as their primary means of expressing parallelism after the transition from vector to distributed memory architectures. With 68-cores per Cori KNL node with four hardware threads each, pure MPI will still work for most applications at some level, though it will unlikely achieve optimal performance. The 512-bit wide vector processing units offer additional parallelism. Application developers will need to explore more on-node parallelism through threading and vectorization. Hybrid MPI/OpenMP is the most common programming model on Cori as it allows for existing MPI applications to be more gradually modified by adding OpenMP for thread scaling.

The key reason why a hybrid MPI/OpenMP programming model is needed on KNL is that a pure MPI approach may no longer fit in memory since each MPI task runs a copy of the program. The number of MPI tasks vs OpenMP threads will depend on the application and the specific MPI and OpenMP implementation for a given code. On Cori, applications often use a range of between 4 and 16 MPI tasks per node and 2 to 8 OpenMP threads per task. OpenMP introduces a compiler-directive agnostic way of expressing SIMD parallelism at the loop or function level; and it also allows a degree of application portability to a variety of supercomputing architectures.

While MPI+OpenMP is the most common programming mode, the Cori system supports other programming models and languages such as UPC, UPC++, PGAS, Pthreads, C++11, TBB, Kokkos and others. Applications can also use non-MPI solutions such as thread-aware task-based runtime system and abstractions (e.g., CHARM++, Legion and HPX).

11.6.2 Languages and Compilers

On the Cori system, three compilers are supported: Intel, Cray and GNU. Cray's PrgEnv modules initialize the programming environment for a specific compiler and simplify users experiences of building applications. The provided Cray compiler wrappers (ftn for Fortran, cc for C and CC for C++ codes) automatically link in necessary MPI and other Cray libraries that are loaded.

Each of these compilers support Fortran and C/C++11 language standards, and support OpenMP, SHMEM, Global Arrays, Pthreads. The Cray and Intel compilers support Fortran

CoArrays, UPC and the KNL MCDRAM memory placement extensions. Only the Cray compiler supports C/C++ CoArrays and OpenACC.

Java and Python libraries are available under the GNU programming environment. Python modules are available for scalable service via Data Virtualization Service (DVS) on the compute nodes.

11.6.3 Libraries and Tools

The default MPI library on Cori is Cray's MPICH, derived from Argonne National Laboratory MPICH [5]and implements the MPI-3.1 standard. Cray's MPICH has custom optimizations for the Aries interconnect and Intel Xeon Phi KNL architectures, which can be utilized via runtime tuning options for memory allocation in DDR or MCDRAM, IO hints, hugepages, adaptive routings, default message communication protocols, etc. User-level Generic Network Interface (uGNI) is available to directly program the Aries network. The Distributed Memory Applications (DMAPP) interface can be targeted at one-sided languages such as PGAS and SHMEM. Intel MPI and OpenMPI based on OpenFabrics [4] are also available.

Cray's Scientific and Math Libraries (CSML, more familiar to users as LibSci) include BLAS, CBLAS, LAPACK, ScaLAPACK, BLACS, PBLACS, Iterative Refinement Toolkit, and CrayBLAS. The Cray MPICH libraries and Cray's scientific libraries are loaded by default on Cori.

Other available common libraries are FFTW, PETSc, Trilinos, and Cray's Third Party Scientific Libraries and Solvers collections, which include MUMPS, ParMetis, SuperLU, SuperLU_DIST, Hypre, Scotch and Sundials. The available IO libraries are: NetCDF, Parallel NetCDF, HDF5, and Parallel HDF5, ADIOS.

Performance monitoring and profiling tools available on Cori include Cray's Performance Monitoring and Analysis Toolkit (CPMAT, more familiar to users as Perftools); Intels VTune and Advisor tools; Allineas MAP tool; Scalasca; TAU; and Barcelona Supercomputing Centers Paraver and Extrae. Debugging tools include Cray's CCDB, LGDB, Intels Inspector tool, Allinea DDT, and Totalview.

11.6.4 Building Software for a Heterogeneous System

The heterogeneous Haswell/KNL system is considerably more complex than a homogeneous cluster. Cori has both Haswell and KNL compute nodes, and all login nodes are Haswell. Compute nodes on Cori KNL are configured without the full build environment, thus applications targeting either Haswell or KNL must be built on the Haswell login nodes.

Executables that run on KNL need to be cross-compiled on Haswell to target KNL. The Cray provided target architecture modules craype-haswell and craype-mic-knl can easily be switched back and forth for building applications to run on the desired target compute nodes. Although a Haswell binary will run on KNL (but not vice-versa), that binary and the libraries it calls will use compiler optimizations targeting Haswell rather than KNL, so NERSC recommends that the appropriate target module be loaded when building applications. The Intel compiler has an optional flag of "-axMIC-AVX512,CORE-AVX2 to build a merged binary that contains instructions appropriate to both Haswell and KNL. This flag is handy when building applications to target both, especially those that will benefit from key numerical libraries that can take advantage of the KNL architecture.

A frequently-encountered difficulty when building software targeting the KNL nodes is that in some build systems (such as autoconf or cmake), a small test program needs to run in order to generate a Makefile. This will not work when building on Haswell for KNL since the binary targeted for KNL cannot run on Haswell. A workaround for this autoconf

issue is to perform the configure step in the Haswell environment, then switch to the KNL environment before compile. The suite of CMake tools were created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects Cray has worked with Kitware to increase the compatibility of CMake with the programming environment on the Cray systems to enable cross-compiling [16].

NERSC is actively working on streamlining the software build process by using the Spack software package manager for building supported packages on all the Cray systems at NERSC, including Edison Ivy Bridge, Cori Haswell, and Cori KNL[31].

11.6.5 Default Mode Selection Considerations

KNL nodes have the option of being configured at boot-time in a variety of sub-NUMA clustering (SNC) modes and memory modes. Depending on the mode, a single KNL node can appear to the OS as having 1 (quad mode), 2 (snc2 mode) or 4 (snc4 mode) NUMA domains. It is also possible to configure the MCDRAM as a direct-map cache (cache mode) or to expose it as a directly accessible memory domain separate from the DDR (flat mode).

NERSC tested these different configurations with a variety of benchmark applications [6, 1] including MILC, GTC-P, AMG, MiniDFT, SNAP and PENNANT and some user applications. In our testing it was found that the SNC modes introduce significant complexity into user job scripts, especially the snc4 mode due to additional NUMA issues caused by the uneven number of cores in each quadrant. We saw the differences from quad+cache from other modes are at most 5-10% differences.

The MCDRAM configured in flat mode outperforms cache mode by a few percent in memory bandwidth bound benchmarks such as STREAM when the working set size is less than 16 GB. In real applications, the performance difference is often minimal, with cache mode sometimes outperforming flat mode. Effectively utilizing a node configured in flat mode with data sizes larger than 16 GB requires memory to be directly allocated into the MCDRAM and requires additional programming effort. Utilizing the flat mode also introduces more settings into the standard batch script.

For these reasons we chose a default of quad,cache mode for our KNL nodes. However, for those users who are able to take advantage of other modes, we allow a fraction of the Cori KNL nodes to be rebooted by the job scheduler. We introduced this limit to avoid overhead related to rebooting nodes, which could last 30 min or longer for each reboot.

11.6.6 Running Jobs

On Cori, the scheduling software SLURM is used in 'native' mode, meaning SLURM directly manages the computing resources without going through another layer of the Cray Application Level Placement Scheduler (ALPS). Having 68 cores per KNL compute node, with 34 dual-core tiles and different cluster and memory modes, complicates the task placement for hybrid MPI/OpenMP applications. The general recommendations are: Use 64 cores per node in most cases and use 1 or 2 cores per node for core specialization, which is a feature designed to isolate system overhead (system interrupts, etc.) to designated cores on a compute node. The srun flags -c and –cpu_bind=cores are critical for getting the optimal process affinity. Without these, multiple MPI ranks will run on the same physical cores. The portable OpenMP process binding options (OMP_PROC_BIND and OMP_PLACES) are then used to fine tune thread affinity. Either numactl or the srun –mem_bind option can be used to specify how MCDRAM memory is used.

For running large jobs, we suggest users to sbcast (the SLURM command to broadcast) the executable to each compute node first before running the job to minimize the job

startup delay, which can be up to several minutes sometimes. Using hugepages is generally recommended.

We provide an online Job Script Generator tool so users can input some simple run parameters and get a ready-to-use batch script. Pre-built binaries to report process and thread affinity information compiled with different compilers are prepared for users to check their desired run time settings for their applications.

11.7 NESAP

11.7.1 Introduction

The NERSC Exascale Science Applications Program (NESAP [11]) was established concurrently with the procurement of Cori in order to help users transition to advanced architectures like Intel's Xeon Phi, Knights Landing processor. The NESAP program provides support for preparing science applications for Cori not only by providing staff and postdoc resources, but also by connecting science code teams with vendor experts at Intel and Cray.

Since NERSC has more than 6,000 users running 700 different applications, it is not feasible to devote staff to each application running at the center. Instead, twenty application teams [9] were selected to participate in the NESAP program, from which an optimization strategy and set of best practices to share with the greater community would be created. The 20 selected applications cover all six programs within the Department of Energy Office of Science and represent about 60% of the overall NERSC CPU hours, either directly or as proxy codes. The selected codes are displayed in Table 11.1.

The domain scientists and application developers received intensive advanced training in the form of on-site visits from Intel and Cray staff, hackathons at Intel campuses as well as on-going collaborations with NERSC and Cray staff as part of a Center of Excellence. Furthermore, these teams had access to pre-production Xeon Phi hardware prior to the delivery of the Cori system, which significantly increased optimization productivity. Eight application teams were additionally paired up with postdoctoral researchers at NERSC who devoted a large fraction of their time to the profiling and optimization efforts.

The lessons learned from these efforts are extensively documented in general Xeon Phi documentation guidelines [12] as well as specialized case studies [10] on the NERSC website. This facilitates the dissemination of the knowledge gained to the rest of the NERSC user base.

11.7.2 Optimization Strategy and Tools

The Cori system uses the same Aries interconnect and dragonfly topology as its predecessor Edison [8]. The most disruptive changes in the system come from the novel architecture of the Intel Xeon Phi processor. Therefore, NESAP code optimization effort is mainly focused on node-level performance.

Compared to e.g. Xeon multi-core processors, the Intel Xeon Phi many-core chip features more (slower) cores, more hyper-threads, wider vector units supporting more complex instructions as well as high-bandwidth on-package MCDRAM. In order to obtain good code performance on Xeon Phi, it is mandatory to utilize at least some of these features. Navigating this complex optimization space is difficult and therefore NERSC developed an optimization strategy based on the roofline performance model [25, 38, 37, 14]. The approach includes the identification of hot kernels, i.e., code regions that consume a large

TABLE 11.1: Overview of NESAP applications.

Name	Scientific Field
ACME	Climate Modeling
BerkeleyGW	Materials Science
BoxLib	Multiple
CESM	Climate Modeling
Chombo-Crunch	Multiple
Chroma	Nuclear Physics
DWF	High Energy Physics
EMGeo	Geophysics
GROMACS	Materials Science
HACC	high Energy Physics
HISQ	High Energy Physics
HMMER	Bioinformatics
Meraculous	Bioinformatics
M3D	Fusion Research
MFDN	Nuclear Physics
MILC	High Energy Physics
MPAS-O	Climate Modeling
NWchem	Basic Energy Sciences
Parsec	Materials Science
Qbox	Materials Science
Quantum ESPRESSO	Materials Science
VASP	Materials Science
WARP	Accelerator Physics
XGC1	Fusion Research

fraction of the wall-time, and the determination of their arithmetic intensity and performance in terms of FLOPS/s. The data points are plotted on a 2D roofline plane and may be visually compared against the Xeon Phi roofline ceilings (determined by limits on memory bandwidth and cpu speed). The position of each of these kernels on the roofline relative to the ceilings indicate the most promising optimization targets. For example, it informs the user if a kernel is memory bound, compute bound or potentially latency bound. The model additionally allows a developer to visually track the effect of optimizations as they are implemented in an application.

In order to obtain the relevant performance metrics, NERSC developed a methodology in cooperation with Cray and Intel staff. This approach employs Intels VTune [2], SDE [15] and Vector Advisor [3] products. The latter recently incorporated a variant of the roofline mode, the so-called cache-aware roofline, into it's basic functionality.

Figure 11.5 depicts the performance of a specific kernel in the BerkeleyGW [24] application after a series of optimization steps (2 - addition of OpenMP, 3 - loop reordering for vector code generation, 4 - cache blocking, 5,6 - hyperthreading and refined vectorization).

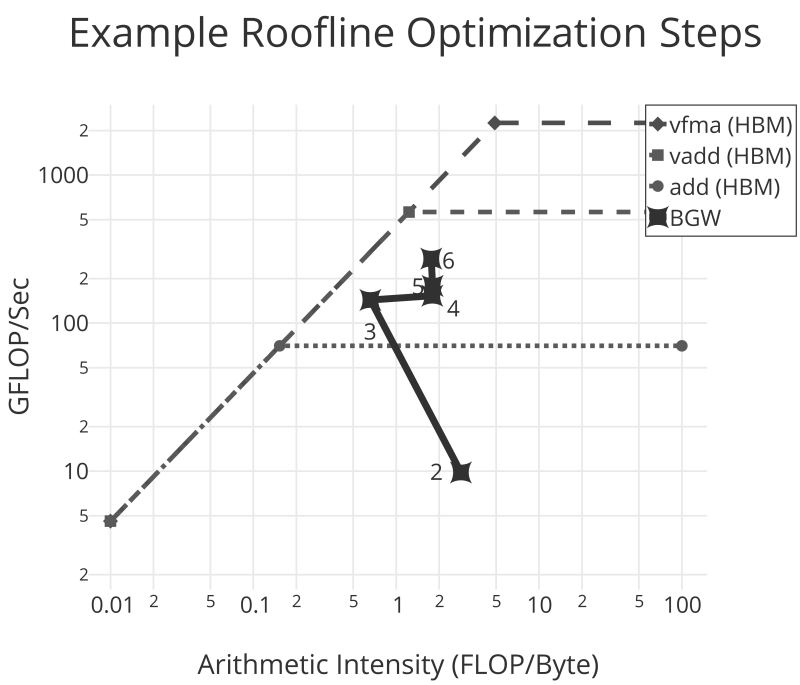

FIGURE 11.5: BerkeleyGW kernel trajectory after a series of optimization steps.

11.7.3 Most Effective Optimizations

As discussed before, the most promising optimizations for a given kernel depend on the position on the roofline plot. For memory-bound kernels, cache blocking and utilizing MCDRAM have proven to be most efficient. The former is especially important as Xeon Phi is not equipped with a comparably big L3 cache that can efficiently mitigate L2 misses. For compute-bound kernels on the other hand, efficient vectorization and instruction-level parallelism such as utilization of FMA instructions are the most promising optimization targets. However, this simple picture becomes complicated when considering more subtle effects such as hardware prefetching: if code is not sufficiently vectorized, single-word loads and stores can overwhelm the prefetcher while vectorized loads and stores reduce the number of in-flight memory operations by a factor of up to 32.

The following optimizations were identified as the most effective across a series of applications:

- *identifying and exploiting parallelism / creating more work for individual threads*: This may be the most important thing to consider when switching from multi-core to many-core architectures. Small OpenMP sections that do not contain enough work for multiple threads will hurt performance significantly due to implicit barriers at the end of these sessions. Where possible, loop nests should be collapsed to maximize parallelism.

- *loop tiling*: Cache blocking to achieve cache locality of heavily used arrays can be realized by reordering and tiling inner loops. This is especially important on Xeon Phi as there is no L3 cache to mitigate the impact of L2 misses on application performance. Unfortunately, as this is a manual code transformation rather than a directive, code

can become less readable and more brittle. Nevertheless, it was found that blocking to shared L2, i.e., 512 KiB per core, performs best for most applications.

- *short loop unrolling*: short loops should be unrolled as they do not provide enough work either for threading or for vectorization.
- *ensuring efficient vectorization*: This can be a major challenge as it may entail loop restructuring as well as data layout transformations. Where the compiler fails to detect vectorization opportunities automatically, loops can be annotated with the OpenMP 4 `simd` pragmas to enforce the compiler to vectorize these loops.
- *using optimized mathematical functions*: AVX512 supports intrinsics for reduced precision divisions and certain expensive mathematical functions such as transcendentals. Those can be enabled by instructing the compiler to use a relaxed floating-point model.

11.7.4 NESAP Result Overview

Figure 11.6 shows the summary of the node-to-node code performance on Edison and Cori before and after applying optimizations. The numbers are collected from multi-node benchmarks representing larger scale science runs being performed by the teams.

The speedups are normalized with respect to the performance of the original (baseline) codes on Edison. The number of nodes for each application is displayed in parentheses.

One observes an approximate 3x speedup on average on Xeon Phi between original and optimized codes. Furthermore, one finds an approximate 1.5x average architecture advantage in favor of Cori (Xeon Phi) vs. Edison.

NERSC further studied impacts of various Xeon Phi-specific hardware features on application performance. Figure 11.7 shows the relative speedups when utilizing MCDRAM (in either cache or flat mode depending on the application preference) vs. ignoring the MCDRAM (running in flat mode utilizing only the traditional DRAM). In addition, one can compare performance when compiling with full-optimization (`-xMIC-AVX512`) vs. disabling vectorization (`-no-vec -no-simd`). It should be noted that the latter test does not include libraries or AVX intrinsic codes where such compiler flags are ignored. The NESAP optimization efforts overview and results are documented in more detail in [17, 30].

11.7.5 Application Highlights

This section presents some selected applications and the major optimizations used to improve their performance on Xeon Phi.

11.7.5.1 Quantum ESPRESSO

Quantum ESPRESSO is an open source density functional theory (DFT) code and widely used in Materials Science and Quantum Chemistry to compute properties of material systems, such as atomic-structures, total-energies, vibrational properties etc. Accurate calculations of important electronic properties, like band-gaps and excitations energies, are achievable for many systems through so called Hybrid Functional calculations employing a certain contribution for the exact exchange (exx) potential - which represents the contribution of the Pauli-Exclusion principle.

The most expensive part of an exact exchange (exx) calculation is the calculation of the exact exchange functional. This function represents essentially a sequence of element-wise products of large complex arrays interleaved with Fast Fourier Transforms (FFT) of

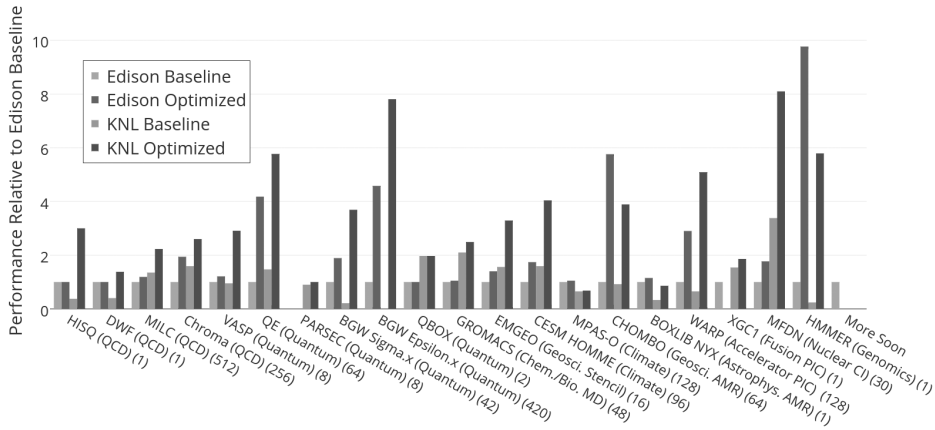

FIGURE 11.6: Applications performance relative to Edison baseline on multiple nodes. The number in parentheses represents the number of nodes used to measure performance.

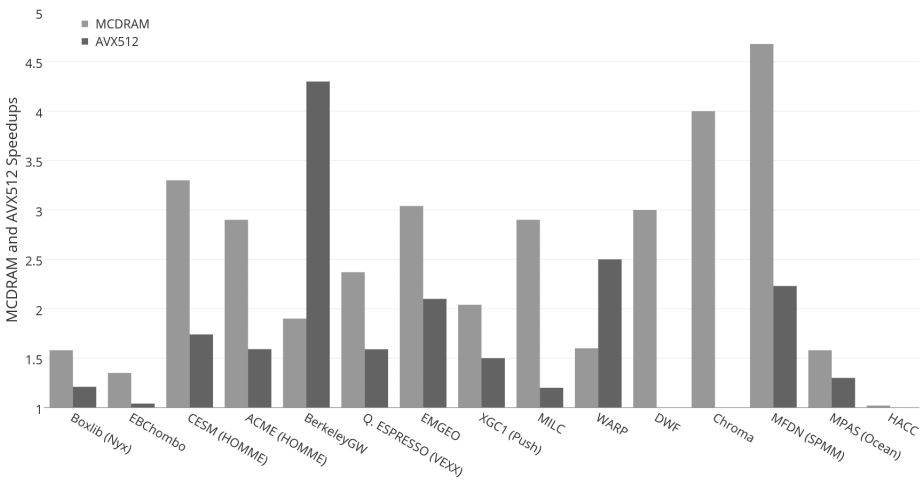

FIGURE 11.7: Applications performance speedup using MCDRAM vs. using DRAM (light bars) as well as using AVX512 vs. using AVX2 (dark bars).

these arrays. In the original code, the OpenMP parallelization was applied to the innermost loop, but the code performance did not scale beyond ∼16 threads (cf. Figure 11.8). VTune profiling revealed that the run time of these sections was extremely short and thus overhead from forking and merging sections dominated. This problem was fixed by moving to a coarse grained parallelization, i.e., moving the OpenMP pragmas on the element-wise products from the inner to the outer loops. Since these sections are interleaved with FFTs, those loops had to be chunked into blocks and the sizes of the involved arrays had to be increased so that they can store results for a full chunk. This further enabled pooling of multiple FFTs within a chunk, which resulted in a ∼20% performance improvement. On the element-wise products, the now doubly nested loops were additionally cache blocked into chunks of 2048 double-complex vectors. The two innermost loops were further decorated with `omp parallel for simd collapse(2)` directives in order to exploit full parallelization. In addition to this, MPI scaling was improved by re-arranging the data structure layout so that more parallelism relevant to this part of the calculation could be exploited. This change also enabled independent, node-local, FFTs, which significantly improved performance over the original distributed ones. In order to ensure compatibility between the exact exchange and the other parts of the calculation, a data structure transformation was performed. The overhead of this procedure is mitigated by the huge performance improvement gained in the expensive exact exchange calculation. Figure 11.9 displays the strong scaling for the full calculation (left panel) and the exact exchange part (right panel) for the original code (annotated QE 5.2.0) and our improved version.

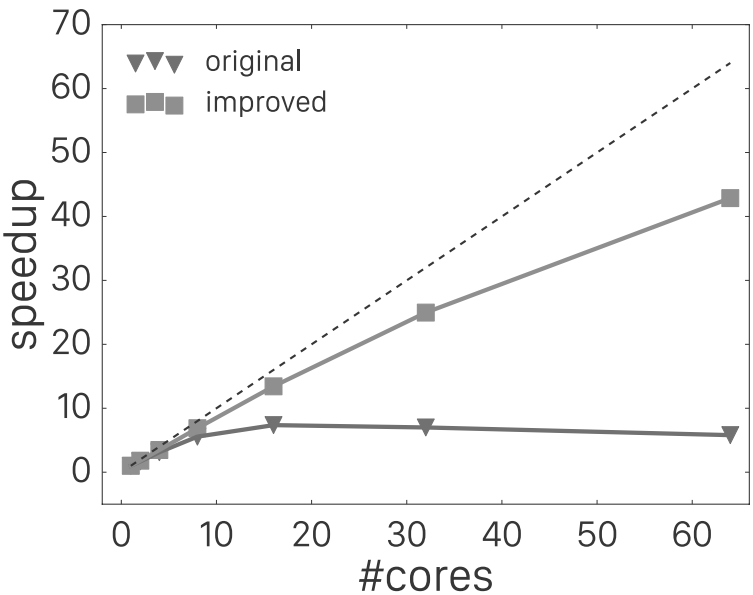

FIGURE 11.8: Strong thread scaling for Quantum ESPRESSO exact exchange calculations improved vs. original code for a system comprised of 64 water molecules.

More detailed discussions are available at [18, 13].

11.7.5.2 MFDn

Many-Fermion Dynamics—nuclear, or MFDn, is a configuration interaction (CI) code for nuclear structure calculations currently in use at multiple machines at NERSC, ALCF and OLCF for *ab initio* calculations of atomic nucei [21, 32, 33, 34]. To provide high accuracy

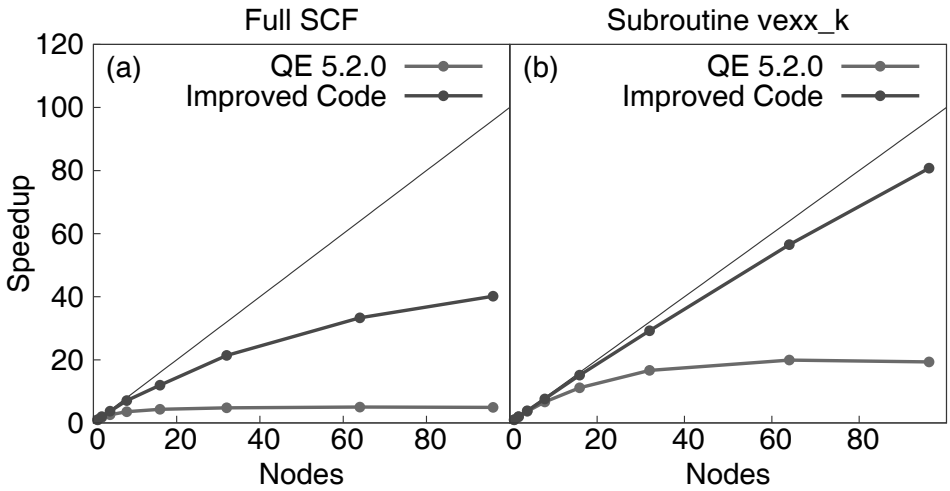

FIGURE 11.9: Multi-node scaling of full hybrid-DFT calculation (left) and of the exx-subroutine (right).

with realistic two and three-body forces with a CI code, a very large and sparse Hamiltonian is required, and therefore a highly scalable code is needed to effectively utilize the aggregate memory of a cluster. For problems of physical interest the matrix dimensions can exceed $10^{10} \times 10^{10}$ with over 10^{13} non-zero elements. The code is written in portable Fortran 90 with a hybrid MPI/OpenMP programming model.

A typical run of MFDn involves three phases:

- Matrix construction
- Obtain lowest eigenvalue/eigenvector pairs
- Compute observable properties

The matrix construction and computation of observables are compute intensive, but contain very few flops. The dominant operations are integer comparisons, bit operations and random access to lookup tables. In order to obtain good performance in these phases on Xeon Phi, improvements to data locality and efficient use of the vector units were essential. Promotion of occupied state bit masks from 32-bit to 64-bit integers, manual loop tiling and OpenMP 4.0 simd pragmas were used to improve vector and cache efficiency. The tile size is now a compile time option and a low multiple of the vector width is chosen in practice. Manual tiling and use of OpenMP pragmas instead of compiler options preserves the portability of the code.

The sparse matrix-matrix multiple kernel (SPMM) is the most expensive operation in the iterative eigensolver. Changing from the Lanczos to LOBPCG eigensolver enabled the use of SPMM, which has a higher arithmetic intensity than SPMV operations. Choosing the best number of MPI ranks per node and OpenMP threads per rank is also important on Xeon Phi.

The high OpenMP scalability of MFDn allows it to run with as few as 1 MPI rank per node (See Figure 11.10), which would be the preferred mode for best memory utilization. However, the best performance on Cori is obtained with multiple ranks as a single Xeon Phi core is not able to fully utilize the high speed network. Optimizations for MFDn resulted in better performance on all platforms and Xeon Phi saw more improvement than Xeon, highlighting that the Xeon Phi platform is more sensitive to code issues than Xeon.

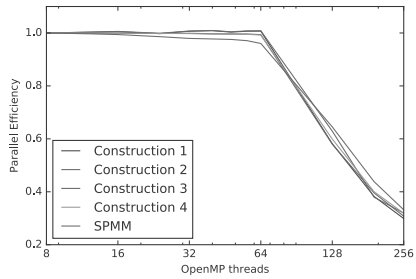

FIGURE 11.10: Strong thread scaling for MFDn for matrix construction and SPMM kernels.

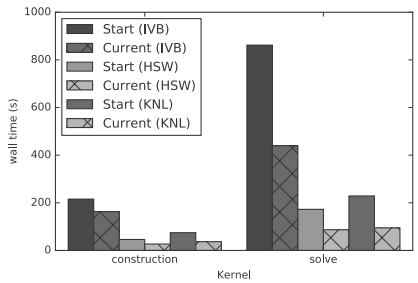

FIGURE 11.11: Comparison of effect of optimizations on 32 nodes of Edison, Cori-Haswell and Cori-KNL.

More detailed discussions are available at [23, 36].

11.8 Data Science

The Cori Haswell partition has been configured to specifically support large-scale data analysis. With increasing dataset sizes coming from experimental and observational facilities, including telescopes, sensors, detectors, microscopes, sequencers and, supercomputers, scientific users from the Department of Energy, Office of Science are increasingly relying on NERSC for extreme scale data analytics. To support these requirements, the Cori system includes hardware, software and policy changes to support this new and growing workload.

11.8.1 IO Improvement: Burst Buffer

One of the top improvements NERSC users consistently request in requirement reviews and feedback is better IO performance. To address this, Cori contains a Burst Buffer, based on the Cray DataWarp technology. This is an intermediate layer of non-volatile storage that sits between the fast on-node DRAM and the slower (but higher capacity) parallel file system (PFS). This Burst Buffer provides users with a configurable layer of fast IO that can improve application IO in several ways:

TABLE 11.2: Burst Buffer IOR performance, using 11120 compute nodes (HSW+KNL), 4 ranks per node.

	Posix FPP (GB/s)	MPIIO shared file (GB/s)	IOPs
Best measured read	1745	1320	28.2M
Best measured write	1566	1364	13.1M

- Improved IO bandwidth for reads/writes, for example, for checkpoint-restart applications.
- Improved performance for complex IO patterns, for example, high IOPS (IO operations per second).
- Improved capability for complex workflows, for example, combining simulation, analysis and visualization codes.

The DataWarp SSDs sit on specialized nodes that bridge the internal Aries interconnect of the compute system and the SAN fabric of the PFS, through the IO nodes. The flash memory is attached to Burst Buffer nodes that are packaged two nodes to a blade. Each Burst Buffer node contains an Intel Xeon processor with 64 GB of DDR3 memory and two 3.2 TB NAND flash SSD modules attached over two PCIe gen3 x8 interfaces. The Burst Buffer nodes are attached to the Cori Cray Aries network interconnect over a PCIe gen3 x16 interface. Each Burst Buffer node provides approximately 6.4 TB of usable capacity and a peak of approximately 5.7 GB/sec of sequential read and write bandwidth, with an aggregate bandwidth for the full Burst Buffer reaching over 1.7TB/sec and 28M IOPs (see Table 11.2 for details).

Access to the Burst Buffer resource is integrated with the SLURM scheduler. When a user submits a job requesting a Burst Buffer allocation, an XFS filesystem is mounted for that allocation so that the user sees a single namespace, even though data might be striped over several DataWarp nodes.

After considerable effort from NERSC staff and Cray engineers to refine the DataWarp performance, users generally see excellent performance from the Burst Buffer [20]. For example, the ATLAS collaboration has used the Burst Buffer to analyze data from the Large Hadron Collider [19], which typically involves several stages of filtering data to identify useful events for further analysis. This "derivation" process involves large I/O reads and is up to 7 times faster using the Burst Buffer compared to Cori Scratch (see Figure 11.12). Subsequent analysis stages of the filtered data exhibit a very different I/O pattern, requiring large amounts of small random reads and writes from the filtered data files. Obtaining an optimal performance in the analysis stage required tweaking the application caching from 2MB to to 100MB, which allowed the application to take better advantage of the available bandwidth to the Burst Buffer. This improved the application performance by a factor of 17, with the Burst Buffer out-performing Cori Scratch by a factor of 5 consistently at all job scales, as shown in Figure 11.13.

Another example application is the coupling of ChomboCrunch and Visit that demonstrates both high bandwidth and a complex workflow using the Burst Buffer [35]. Figure 11.14 illustrates that the simulation, visualization and analysis can be run simultaneously using the Burst Buffer, enabling higher spatial and temporal resolution, and Figure 11.15 shows that the bandwidth out-performs Lustre at all scales.

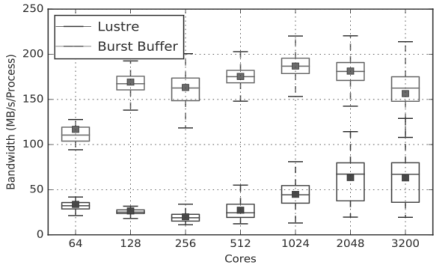

FIGURE 11.12: Comparison of application bandwidth to Cori Scratch file system (Lustre) and the Burst Buffer, for ATLAS data filtering.

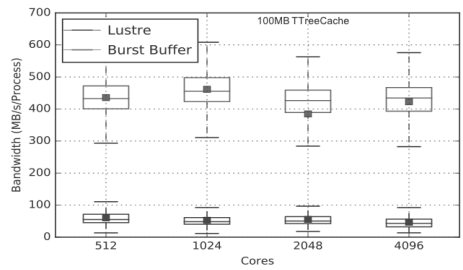

FIGURE 11.13: Comparison of application bandwidth to Cori Scratch file system (Lustre) and the Burst Buffer, for ATLAS data analysis.

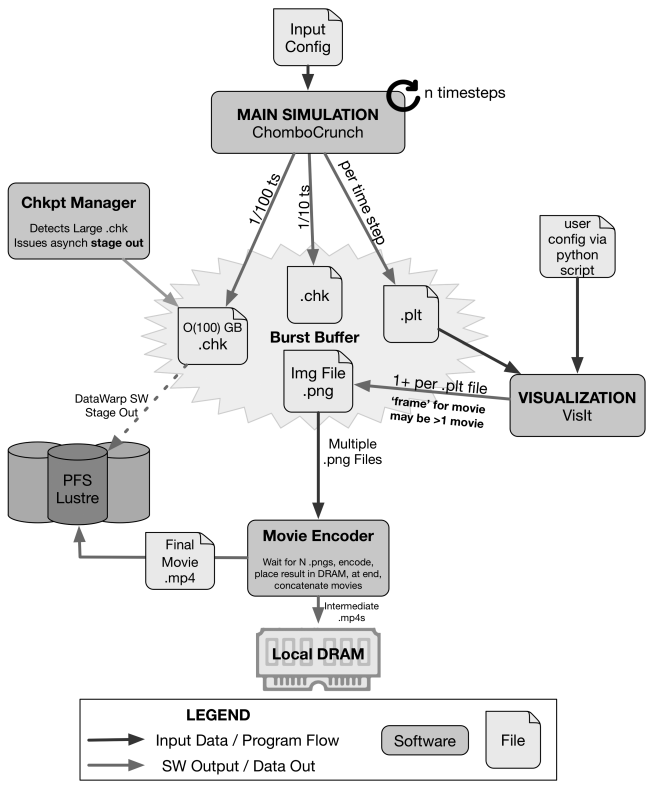

FIGURE 11.14: Chombo-Crunch + VisIt workflow.

11.8.2 Workflows

NERSCs mission has been expanding into closer interactions with experimental and observational facilities, whose users often have different requirements than traditional HPC modeling and simulation users. Users analyzing large data sets from an experimental facility have more complex workflows including filtering data, moving data and running multiple pipelined analysis codes on the data. The Cori system has been designed with data-intensive workflows in mind.

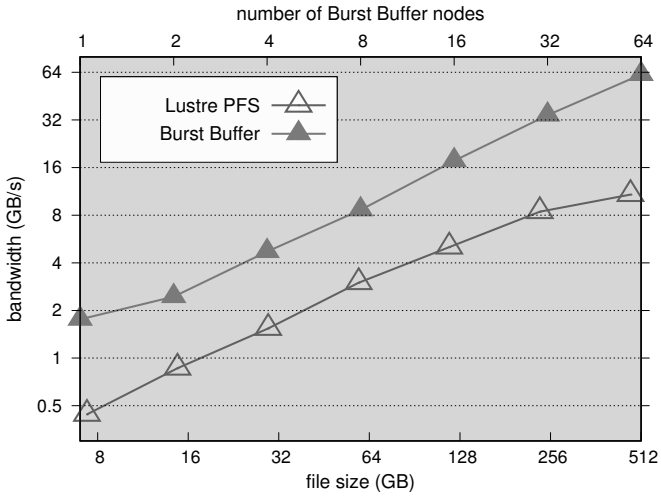

FIGURE 11.15: Chombo-Crunch I/O bandwidth scaling.The compute node to Burst Buffer node ratio is fixed at 16:1.

11.8.2.1 Network Connectivity to External Nodes

Many workflow systems are managed by a control node or database that manages the different tasks and stages in a workflow - this may be a persistent service that lives outside the Cori network. Individual workflow tasks may also need to pull down units of work or data, or publish results to/from a data service. Cori compute nodes therefore are able to talk to external services directly.

11.8.2.2 Burst Buffer Filesystem for In-situ Workflows

Multi-Stage Workflows will generate data between each step in the workflow - for example, a simulation generates data files, which need to be analyzed by an independent application, which then needs to be visualized by a third application (such as the Chombo-Crunch and Visit workflow mentioned in section 11.8.1 and [35]). As the size of the data grows, it becomes increasingly impractical to move the data in and out of the file system. The burst buffer provides a very convenient intermediate staging area for this data.

11.8.2.3 Real-time and Interactive Queues for Time Sensitive Analyses

Cori supports a real-time queue for time sensitive analyses. Users can request a small number of on-demand nodes if their jobs have special needs that cannot be accommodated through the regular batch system. The real-time queue enables immediate access to a set of nodes, for jobs that are under the real-time wallclock limit. Typically this is used for real time processing linked to an experiment (e.g., LCLS) or event (supernova). In addition to this, in mid-2017 NERSC began providing 192 nodes (Haswell and KNL) for high-turnaround 'interactive' usage. This allows all users to have instant turnaround for single analytics jobs (that can use multiple nodes for several hours).

11.8.2.4 Scheduler and Queue Improvements to Support Data-intensive Computing

In addition, the scheduler and scheduling policies we had traditionally used to support our modeling and simulation workloads proved inflexible for dealing with the more dynamic needs of data users. One of the key changes made to support this new workload was a change to Native SLURM. The schedulers closer ties to the compute nodes provide easier diagnostics, cleaner access to the data and faster startup. Users from experimental facilities also often come with expectations of a specific operating system or require a complex set of installed libraries. For these users, NERSC is allowing users to bring their own images to Cori, by way of a new capability called Shifter [8], which enables users to import and use their Docker containers.

11.9 System Statistics

As mentioned in Section 11.7.1, NERSC has over 700 projects and more than 6,000 users. To facilitate moving codes to the Intel Knight's Landing processors, the NESAP codes chosen covered a large spectrum of science fields within the DOE Office of Science mission. Since these teams had been working on their codes to prepare for the system, they were given exclusive and priority access to KNL nodes starting from November 2016. In January 2017, access to a subset of the KNL nodes was granted to all users for code development, debugging and optimization. Non-NESAP users started to gain full access to the KNL nodes once they had demonstrated some degree of application-readiness. The system entered full production mode in July 2017.

11.9.1 System Utilizations

Figure 11.16 shows the 30-day rolling medium utilization in node days on Cori KNL. NESAP users ramped up the system usage very quickly from the end of November 2016 to early January 2017. As shown in Figure 11.18, the system utilization has been mostly in the 90% range once users were provided access.

Figure 11.17 shows the breakdown of hours used on Cori KNL nodes from December 22, 2016 after Cori was accepted, through June 15, 2017. Lattice QCD codes were ready to use KNL before many others. VASP and Quantum Espresso are material science applications that NERSC installs on systems and users execute. The VASP and Quantum Espresso users were enabled mid-January, explaining the large usage in materials science.

The workload is highly parallel as indicated in the Job Size breakdowns in Figure 11.18. About half of the hours from January to June 2017 were used by jobs using more than 512 nodes and more than 25% of the total hours were used by jobs running on more than 1,024 nodes. The system utilization chart also indicated a couple of Cori system maintenance periods: February 28 to March 3, and April 18-21 for new cabinets integration, and March 22-24, 2017 for an OS upgrade.

11.9.2 Job Completion Statistics

Job completion statistics help to indicate how successfully user applications are running on the system and to discover potential system problems. On Cori, the SLURM accounting database is used to analyze success rate, failure categories and causes.

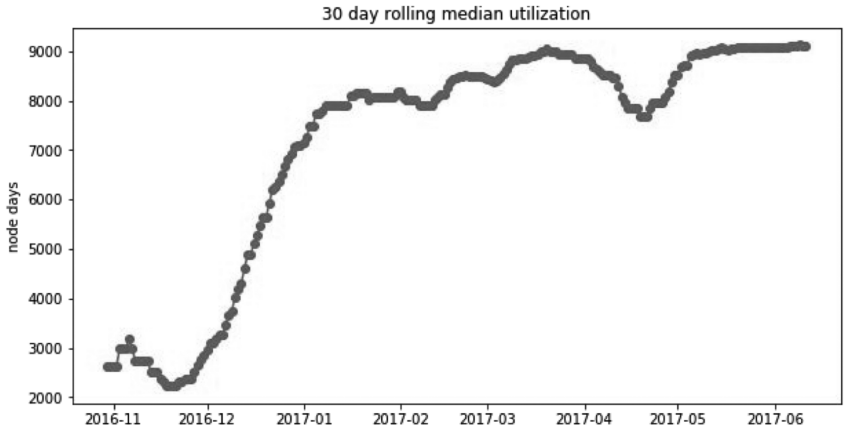

FIGURE 11.16: KNL system utilization from November 2016 to June 2017.

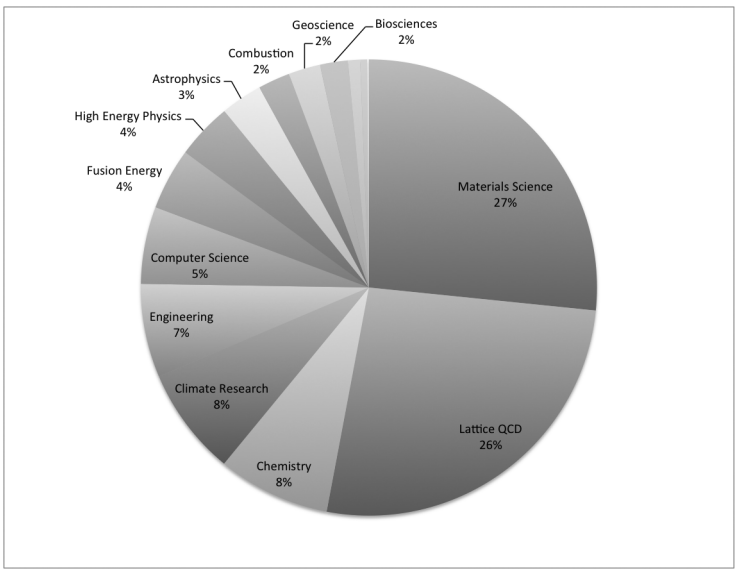

FIGURE 11.17: KNL node hours used by various science category from January to June 2017.

Figure 11.19 shows the percentage of KNL node-hours spent by jobs completed with each category. A small number of jobs in the categories of Node Fail, Boot Fail and Cancelled System are considered as job failures related to system issues. Jobs in the "Completed" category are completed successfully. Jobs in the Timeout categories are unknown of whether from system or user issues. Many NERSC workflows choose to intentionally timeout and restart from regular checkpointing data. Cancelled or Failed jobs can be from either user or system cause, which mostly represent user activities debugging their workloads.

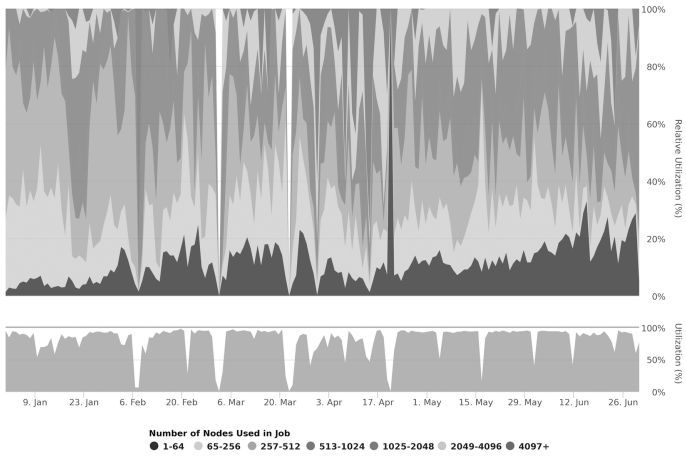

FIGURE 11.18: KNL Job size and system utilization from January to June 2017.

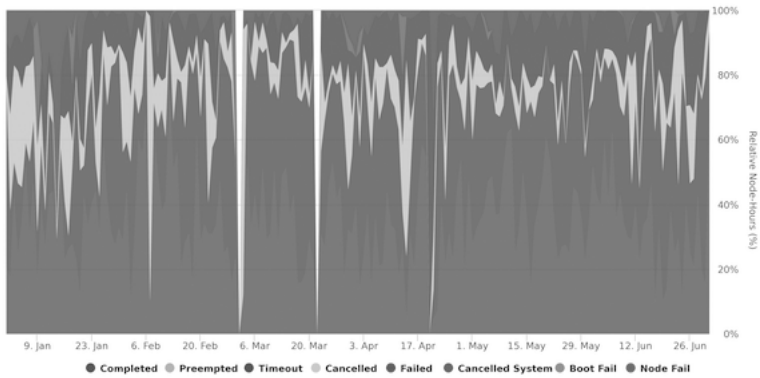

FIGURE 11.19: Job Completion Breakdown from January to June 2017.

11.10 Summary

In summary, the Cori system has been configured to support both large-scale simulation as well as extreme data analysis for the broad Department of Energy, Office of Science workload. With a number of new architectural features such as the Burst Buffer and the Knights Landing processor with many light weight cores and high bandwidth memory, users are getting exposed to new system features that are expected in next generation systems on the path to exascale. Through the NESAP program, users have optimized applications through improved parallelism and vectorization and the use of on-package memory. On average, NESAP applications have seen a 3x speedup on the Cori Knights Landing compute nodes.

Innovations such as Shifter and SDN as well as new queue policies to support data workloads have enabled the Cori system to support users from analyzing data from experimental facilities.

Since the system was installed, users have rapidly ported codes and system utilization has been high. NERSC is continuting to add new capabilities to the system, which will be in service for approximately 5 more years.

11.11 Acknowledgments

The authors would like to thank Woo-Sun Yang and Austin Leung for helping in collecting some system statistics. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Bibliography

[1] APEX Benchmarks. https://www.nersc.gov/research-and-development/apex/apex-benchmarks/.

[2] Intel VTune Amplifier. https://software.intel.com/en-us/intel-vtune-amplifier-xe.

[3] Intel®Advisor. https://software.intel.com/en-us/intel-advisor-xe.

[4] Libfabric OpenFabrics. https://ofiwg.github.io/libfabric.

[5] MPICH. http://www.mpich.org.

[6] NERSC-8 Benchmarks. https://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/.

[7] NERSC Cori System. https://www.nersc.gov/users/computational-systems/cori.

[8] NERSC Edison System. https://www.nersc.gov/users/computational-systems/edison.

[9] NESAP. http://www.nersc.gov/users/computational-systems/cori/nesap/nesap-projects.

[10] NESAP Application Case Studies. http://www.nersc.gov/users/computational-systems/cori/application-porting-and-performance/application-case-studies/.

[11] NESAP Projects. http://www.nersc.gov/users/computational-systems/cori/nesap.

[12] NESAP Xeon Phi Application Performance. http://www.nersc.gov/users/application-performance/preparing-for-cori/.

[13] Quantum ESPRESSO Case Study. http://www.nersc.gov/users/computational-systems/cori/application-porting-and-performance/application-case-studies/quantum-espresso-exact-exchange-case-study/.

[14] Roofline Performance Model. http://crd.lbl.gov/departments/computerscience/PAR/research/roofline.

[15] SDE: Intel Software Development Emulator. https://software.intel.com/en-us/articles/intel-software-development-emulator.

[16] Tips for Using CMake and GNU Autotools on Cray Heterogeneous Systems. http://docs.cray.com/books/S-2801-1608//S-2801-1608.pdf.

[17] Taylor Barnes, Brandon Cook, Douglas Doerfler, Brian Friesen, Yun He, Thorsten Kurth, Tuomas Koskela, Mathieu Lobet, Tareq Malas, Leonid Oliker, and et al. *Evaluating and Optimizing the NERSC Workload on Knights Landing*. Jan 2016.

[18] Taylor A. Barnes, Thorsten Kurth, Pierre Carrier, Nathan Wichmann, David Prendergast, Paul R.C. Kent, and Jack Deslippe. Improved treatment of exact exchange in quantum {ESPRESSO}. *Computer Physics Communications*, 214:52 – 58, 2017.

[19] W. Bhimji, D. Bard, K. Burleigh, C. Daley, S. Farrell, M. Fasel, B. Friesen, L. Gerhardt, J. Liu, P. Nugent, D. Paul, J. Porter, and V. Tsulaia. Extreme i/o on hpc for hep using the burst buffer at nersc. Computing in High-Energy Physics, 2016.

[20] W. Bhimji, D. Bard, M. Romanus, D. Paul, A. Ovsyannikov, B. Friesen, M. Bryson, J. Correa, G.K. Lockwood, V. Tsulaia, Byna S., S Farrell, D. Gursoy, C. Daley, V Beckner, B. Van Straalen, D. Trebotich, Tull C., G.H. Weber, N.J. Wright, K. Antypas, and Prabhat. Accelerating science with the nersc burst buffer. Cray User Group, 2016.

[21] S. Binder, A. Calci, E. Epelbaum, R. J. Furnstahl, J. Golak, K. Hebeler, H. Kamada, H. Krebs, J. Langhammer, S. Liebig, P. Maris, U.-G. Meißner, D. Minossi, A. Nogga, H. Potter, R. Roth, R. Skiniński, K. Topolnicki, J. P. Vary, and H. Witała. Few-nucleon systems with state-of-the-art chiral nucleon-nucleon forces. *Phys. Rev. C*, 93(4):044002, 2016.

[22] R.S. Canon, T. Declerck, B. Draney, J. Lee, D. Paul, and D. Skinner. Enabling a superfacility with software defined networking. Cray User Group, 2017.

[23] Brandon Cook, Pieter Maris, Meiyue Shao, Nathan Wichmann, Marcus Wagner, John ONeill, Thanh Phung, and Gaurav Bansal. High performance optimizations for nuclear physics code mfdn on knl. In *International Conference on High Performance Computing*, pages 366–377. Springer, 2016.

[24] Jack Deslippe, Georgy Samsonidze, David A. Strubbe, Manish Jain, Marvin L. Cohen, and Steven G. Louie. Berkeleygw: A massively parallel computer package for the calculation of the quasiparticle and optical properties of materials and nanostructures. *Computer Physics Communications*, 183(6):1269 – 1289, 2012.

[25] Douglas Doerfler, Jack Deslippe, Samuel Williams, Leonid Oliker, Brandon Cook, Thorsten Kurth, Mathieu Lobet, Tareq Malas, Jean-Luc Vay, and Henri Vincenti. *Applying the Roofline Performance Model to the Intel Xeon Phi Knights Landing Processor*, pages 339–353. Springer International Publishing, Cham, 2016.

[26] P. Hill, C. Synder, and J. Sygulla. Knl system software. Cray User Group, 2017.

[27] D.M. Jacobsen. Extending cle6 to a multicomputer os. Cray User Group, 2017.

[28] M. Jette, D.M. Jacobsen, and D. Paul. Scheduler optimization for current generation cray systems. Cray User Group, 2017.

[29] William TC Kramer, John M Shalf, and Erich Strohmaier. The sustained system performance (ssp) benchmark.

[30] Thorsten Kurth, William Arndt, Taylor Barnes, Brandon Cook, Jack Deslippe, Doug Doerfler, Brian Friesen, Yu He, Tuomas Koskela, Mathieu Lobet, Tareq Malas, Leonid Oliker, Andrey Ovsyannikov, Samuel Williams, Woo-Sun Yang, and Zhengji Zhao. *Analyzing Performance of Selected Applications on the Cori HPC System*. Jun 2017. Accepted for IXPUG Workshop Experiences on Intel Knights Landing at the One Year Mark, ISC 2017, Frankfurt, Germany.

[31] Melara M, Gamblin T, Becker G, French R, Belhorn M, Thompson K, Scheibel P, and HartmanBaker R. Using spack to manage software on cray supercomputers. In *Proceedings of Cray User Group*, 2017.

[32] P. Maris, M. A. Caprio, and J. P. Vary. Emergence of rotational bands in ab initio no-core configuration interaction calculations of the Be isotopes. *Phys. Rev. C*, 91(1):014310, 2015.

[33] P. Maris, J. P. Vary, P. Navratil, W. E. Ormand, H. Nam, and D. J. Dean. Origin of the anomalous long lifetime of ^{14}C. *Phys. Rev. Lett.*, 106(20):202502, 2011.

[34] Pieter Maris, James P. Vary, S. Gandolfi, J. Carlson, and Steven C. Pieper. Properties of trapped neutrons interacting with realistic nuclear Hamiltonians. *Phys. Rev. C*, 87(5):054318, 2013.

[35] A. Ovsyannikov, M. Romanus, B. Van Straalwn, G. Weber, and D. Trebotich. Scientific workflows at datawarp-speed: Accelerated data-intensive science using nersc's burst buffer. IEEE, 2016.

[36] Meiyue Shao, Hasan Metin Aktulga, Chao Yang, Esmond G Ng, Pieter Maris, and James P Vary. Accelerating nuclear configuration interaction calculations through a preconditioned block iterative eigensolver. *arXiv preprint arXiv:1609.01689*, 2016.

[37] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, April 2009.

[38] Samuel Webb Williams. *Auto-tuning Performance on Multicore Computers*. PhD thesis, Berkeley, CA, USA, 2008. AAI3353349.