Performance Measurements of the NERSC Cray Cascade System

Brian Austin, Matthew J. Cordery, Harvey J. Wasserman and Nicholas J. Wright
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
Berkeley, CA 94720

Abstract— **Cray began delivery of their next generation XC30 supercomputer systems in late 2012. One of the first systems, "Edison," was delivered to NERSC and in this paper we present preliminary performance results obtained on this machine. The primary new feature of the XC30 architecture is the Cray "Aries" interconnect that includes a 48-port high radix router with a dragonfly topology. To demonstrate the Aries' substantial improvements in bandwidth, latency, message rate, and scalability, we present measurements of the basic performance characteristics of the system and examine the scalability of several network-centric "microbenchmarks." Although some low-level microbenchmark results for Aries have been published previously (using prototype hardware), the unique contribution of this work consists of performance results for the NERSC Sustained System Performance (SSP) application benchmarks. The SSP benchmarks span a wide range of science domains, algorithms and implementation choices, and provide a more holistic performance metric. We examine the performance and scalability of these benchmarks on the XC30 and compare performance with other state-of-the-art HPC platforms. Edison nodes are composed of two eight-core Intel "Sandy Bridge" processors, which provide single-node performance to complement the networking improvements afforded by the Aries interconnect. Counting two hyperthreads per core, Edison has 32 hardware threads per node; thus, multi-threading is essential for obtaining optimal performance. We report the OpenMP, core-specialization and hyperthreading settings that maximize SSP on the XC30.**

Keywords—XC30; Aries; hyperthreading; core specialization; sustained system performance

I. INTRODUCTION

Twenty years ago, Cray launched its first massively parallel supercomputer architecture, the T3D [1]. Both that system and its immediate follow-on, the T3E [2], used what became a common industry approach to building scalable multiprocessor platforms, using commodity microprocessor nodes surrounded by custom integrated interprocessor network technology.

Recently Cray has unveiled its latest distributed memory architecture, a system developed as part of Cray's DARPA High Productivity Computing System program. This architecture, known internally as the Cray Cascade system and by the product name Cray XC30, includes Cray's newest interconnect, Aries, which includes new technologies such as a dragonfly topology and adaptive routing.

The National Energy Research Scientific Computing (NERSC) Center has recently installed the first phase of an XC30 system which will achieve about 2-PF theoretical peak performance after its second phase upgrade is installed in late 2013. The first phase system is the subject of this report. The system has been bestowed with the name Edison, in honor of the American inventor Thomas Alva Edison.

In addition to characterizing Edison's interconnect performance, we are particularly interested in two system features: hyperthreading (HT) and core specialization (CS). HT is a feature of the Intel Xeon processor family that allows multiple threads to be executed on a single hardware core. Additional threads share the architectural state of the master thread but not its instruction stream. One of the primary benefits of HT is to keep the execution unit pipeline filled in case the currently executing thread stalls. These stalls may occur as a result of a cache miss or a branch misprediction, for example. In the case of highly optimized code that avoids these effects, the advantages of HT may be minimal. HT may also permit the simultaneous use of the different execution ports for floating-point and integer calculations [9].

Core Specialization [3] is a feature of the Cray operating system that allows the user to reserve one (or more) cores per node for handling system services and thus reduce the effects of OS jitter, at the expense of possibly requiring more nodes to run an application for a given number of compute tasks. These cores may also be used in conjunction with Cray's MPI Asynchronous Progress Engine [3] to improve the overlap of communication and computation if non-blocking communications are used by an application. In the absence of CS, the compute cores themselves must service their own non-blocking MPI requests.

Hyper-threading complicates questions about the most effective use of processor resources. HT doubles the number of streams that can schedule computation, but does not increase other resources (e.g., floating-point units or cache). A key question for users is whether these 'virtual cores' are best used by allocating additional MPI processes to the nodes, by using multiple threads per MPI task, or by devoting some to the

All authors were supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

operating system and others to the MPI progress engine via CS. We address this question by comparing the performance of several scientific applications analyzing the effects of HT and CS features on their performance. We provide an estimate of aggregate application performance for the three scenarios and also compare with other HPC platforms.

II. Computational Systems

A. Edison

This paper focuses on the performance characteristics of the Edison Cray XC30 system at NERSC. The initial phase of the system was delivered in Q4 2012 with full configuration and user availability expected in Q2 2013. In its current configuration, Edison has 664 compute nodes. The current phase-1 system is comprised of four cabinets, each containing 42 blades. Each blade is comprised of four dual-socket nodes, each containing two 8-core 2.6-GHz Intel Sandy Bridge processors and 64 GB of 1,866-MHz DDR3 memory (nominally 4 GB/core). Each of the two dies on a node is connected to each other and to their own memory via Intel's QPI interface, resulting in potential NUMA penalties for crossing the NUMA domains. Using the new AVX SIMD hardware, each compute core is capable of eight double-precision floating-point operations per cycle, yielding a theoretical performance of 332.8 Gflops/node.

The Cray XC30 system also incorporates the new Aries interconnect, which, in addition to its its raw bandwidth performance, is distinguished by a novel flattened dragonfly topology and its adaptive routing capabilities. The four nodes on an XC30 blade share a 48-port Aries router chip, which controls traffic between the rank-1, rank-2 and rank-3 networks. The rank-1 network handles traffic within a 16-blade chassis (up to 64 nodes); the rank-2 network handles traffic between all chassis in a two-cabinet 'group' (up to 384 nodes); and the rank-3 network handles all traffic between all groups. There are only two groups in the Edison phase-1 system. The rank-1 and rank-2 networks are connected through the backplane (rank-1) or via inexpensive copper wires (rank-2) and are capable of 14 Gbps transfer rates. The rank-3 network is connected via more expensive optical cables and is capable of transfer rates of up to 12.5 Gbps. The adaptive routing capabilities of the interconnect become significant only when messages traverse the rank-3 network. If the adaptive network determines that its initial routing between groups is congested, it will select a different less congested route, thus making use of the full capabilities of the system and reducing network contention. For more details see [5].

B. Hopper

For comparison purposes, some tests are also performed on Hopper, which is a Cray XE6 system and has been described previously [4]. Hopper has 6,384 compute nodes connected by a Cray Gemini interconnect. Within a Hopper node, there are two 2.1-GHz, 12-core AMD Magny-Cours processors, and 32 GB of 1,330-MHz RAM. The Magny-Cours processor is itself made up of two multi-chip modules; thus, there are four 8-GB NUMA domains connected by HyperTransport. Pairs of nodes are connected by HyperTransport to a Gemini network chip, that represents one point on a 3D torus network capable of transferring 3–6 GB/s per direction; see [13] for details.

III. Network Performance

The distinctive feature of the Cray Cascade architecture is the Aries network ASIC and the interconnection network based upon it. Aries has been described in great detail previously [5]. In this section, we use several network microbenchmarks to assess Aries' performance, reproducing measurements from the prototype system in [5] and confirm results from earlier simulations. We use the OSU MPI benchmark suite [7]. For point-to-point measurements, we ensure that sending and receiving pairs are attached to different Aries ASICs.

Fig. 1 shows the measured Edison inter-processor MPI “ping-pong” message latency as a function of message size, measured by the OSU MPI multi-pair latency benchmark [7]. Although not shown, the values do not appear to vary appreciably as a function of the location of the pair of communicating nodes within the Edison interconnect topology; the 8-byte latency remains at a value of about 1.5 μs for both the nearest and furthest nodes in the network. This value may be compared with approximately 1.8 μs for the (busy) XE6 Hopper system. Note that when a single pair of cores (on two nodes) are communicating, the Edison MPI latency is only slightly larger than that reported for the prototype system in [5]. Fig. 1 also shows that the latency increases when more pairs of cores are communicating; the 8-byte value is about 2.4 μs when all cores in both nodes are active. This value is considerably smaller than that observed when all (24) cores are communicating on the Hopper system (4.2 μs).

Fig. 2 shows the MPI “ping-pong” message bandwidth as a function of message size. The asymptotic values observed on Edison, i.e., 10 GB/s for unidirectional and about 15 GB/s for bidirectional, agree with the point-to-point “put” bandwidths reported in [5].

The Aries interconnect provides substantially more global bandwidth than Gemini, thus providing more scalable performance for nonlocal communication patterns. We measure runtimes for MPI_Alltoall calls using the OSU benchmark and one MPI process per node. Fig. 3 shows the all-to-all injection bandwidth per node (*InjectionBandwitdhPerNode = MessageSize x (Nodes-1) / AlltoallTime*) for several message sizes and concurrencies. Unlike the all-to-all data in [5], these data are measured (not simulated) on a busy systems. Edison's all-to-all performance is remarkably better than Hopper's, achieving up to three times more injection bandwidth per node for 1-MB messages. Furthermore, that this rate is nearly constant up to 256 nodes indicates that Aries' high global bandwidth within a group provides excellent all-to-all scalability. Above 384 nodes, the benchmark must span multiple two-cabinet groups and is forced to use the slower rank-3 network link, which explains

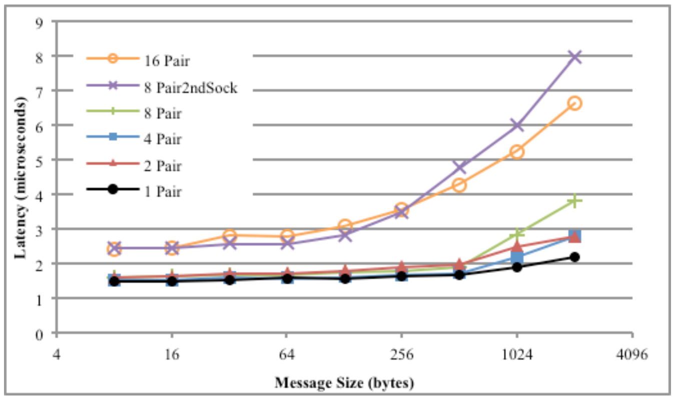

Fig. 1. MPI multi-pair latency on Edison.

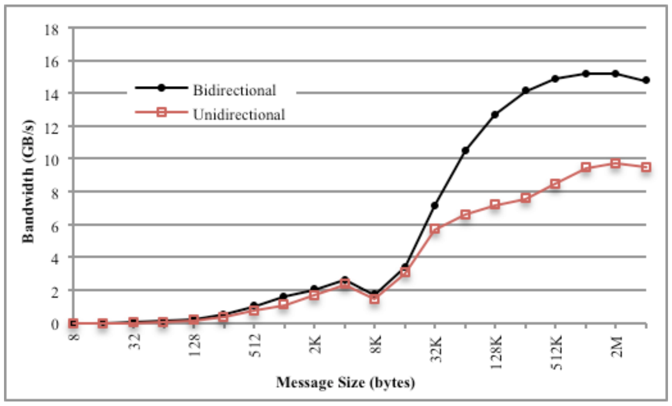

Fig. 2. MPI ping-pong bandwidth

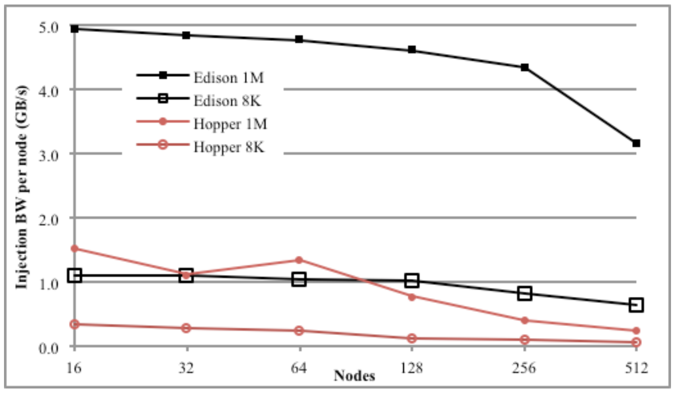

Fig. 3. Alltoall injection bandwidth per node.

the drop in injection bandwidth observed for 512 nodes. The dragonfly's rank-3 network has an all-to-all topology, so the 3 GB/s injection bandwidth per node should be maintained as the system expands to more groups.

Core Specialization (CS) reserves one (or more) cores per node for the operating system. When the MPI asynchronous progress feature is enabled, nonblocking MPI functions can be offloaded to the specialized core. As noted in [10], the Gemini interconnect has no special hardware to facilitate MPI handshake protocols and many CPU cycles can be consumed processing MPI requests. CS has the potential to process these requests independent of the user's process and enable better overlap between communication and computation.

To assess the CS and asynchronous progress features, we measure the fraction of computational work that can be overlapped with communication, as described in [10]. Fig. 4 shows the overlap-fraction with and without CS. For messages smaller than 8 KB, overlap-fractions of 50% are achieved with and without CS. When only one pair communicates, CS dramatically decreases overlap at the 8-KB threshold, where Aries switches between the Fast Memory Access (FMA) and Block Transfer Engine (BTE) mechanisms, and begins to use the progress engine. CS gradually becomes more useful as the message size increases, and is essential for obtaining overlap for 2-MB messages. However, with 16 pairs, 85% overlap can be observed for large messages without CS, and up to 98% overlap is possible with CS.

The overlap fraction is a measure of a relative speedup between blocking and non-blocking versions of a communication-computation loop. It is also informative to compare the change in runtime for the non-blocking loop with and without CS, as shown in Fig. 5. For one or 16 pairs, CS improves runtimes only for messages larger than about 1MB. For 512 pairs, CS has a significant negative impact.

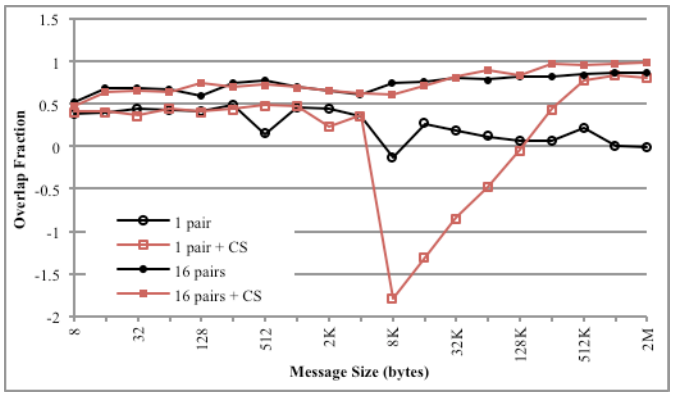

Fig. 4. Overlap fraction measured on Edison. The computation to communication ratio is 1.0 for all message sizes.

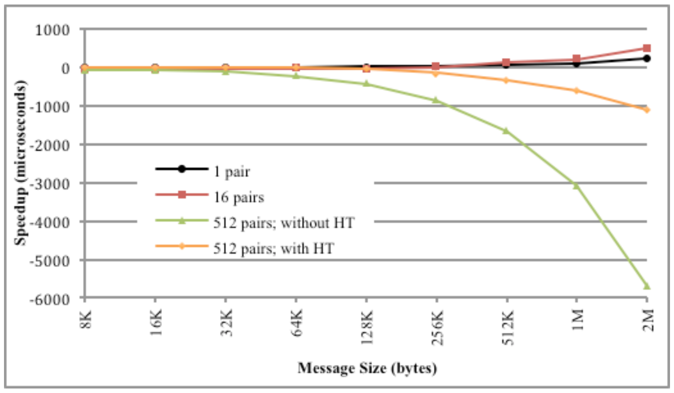

Fig. 5. Speedup due to CS.

IV. Application Benchmarks

As the primary computing center for the U.S. DOE Office of Science, the NERSC Center serves approximately 5,000 users working on some 700 projects that involve nearly 700 codes for a wide variety of scientific disciplines. The application benchmarks used here have been selected from the NERSC workload to survey a range of algorithms and domains [8]. This section describes each of the benchmark codes and covers the effects of HT and CS on their performance.

A. Code Descriptions

1) CAM

Climate simulations comprise a significant part of the NERSC workload. Much of this work is performed using the Community Earth System Model (CESM) developed by the National Center for Atmospheric Research (NCAR). Within CESM, the atmosphere model, CAM, represents a dominant fraction of the total model computational burden. Because of this, CAM has been an important component of the NERSC benchmark suite for many years. Within CAM, the main computational burdens are the dynamical core (which solves the equations of fluid motion in the atmosphere) and the physical core (which implements a number of sub-scale grid processes that affect the dynamics, such as long- and short-wave radiative transfer, precipitation, turbulent mixing, etc.). The three-dimensional computational grid of CAM represents variables described by latitude, longitude, and vertical height. There are several dynamical cores available for CAM, each using different methods to solve the same basic set of equations. In this study, we use the finite volume (FV) dynamical core. The FV core decomposes the grid in two different ways: by latitude-longitude coordinates and by latitude-vertical coordinates. In the physical core, which is common to all dynamical cores, the MPI decomposition follows the latitude-longitude decomposition of the dynamical core. In general, interprocessor communications are dominated by non-blocking sends and blocking receives, though the code may be configured at run time to use different protocols. The OpenMP implementation in the FV core is, in general, over latitude or levels. The OpenMP implementation in the physical core is different. All of the vertical cells at a particular latitude-longitude point are referred to as a column. A 'chunk' is a collection of columns and the physical core uses this chunking strategy as a means of implementing a second level of parallelism via OpenMP.

The benchmark used in this paper is the D resolution (nominally 0.5 degrees) which translates to a 576 x 361 x 28 grid. The benchmark uses 240 MPI tasks with a 60x4 MPI decomposition and either one or two OpenMP threads and runs for five simulated days.

2) GAMESS

The General Atomic and Molecular Electronic Structure System (GAMESS) is capable of a wide variety *ab-initio*, density-functional and semiempirical quantum chemistry calculations. Within GAMESS, molecular orbitals are represented as linear combinations of Gaussian basis functions. A large fraction of its calculations consist of linear transforms between the Gaussian and molecular orbital basis sets, which are characterized by stride-1 memory access.

GAMESS provides its own communication library, the Distributed Data Interface (DDI), which allows each process to access a global pool of memory. The DDI version used on Edison is implemented with MPI. Within a NUMA-node, DDI uses SHMEM, and one MPI task manages off-node communication for the remaining compute tasks. GAMESS communication pattern emphasizes collective functions.

The GAMESS benchmark performs a Hartree-Fock + MP2 energy and gradient calculation for a 43-atom molecule with 306 electrons and 1025 basis functions.

3) GTC

The Gyrokinetic Toroidal Code (GTC) simulates turbulent transport in magnetically confined plasmas by solving the gyro-averaged 3-D Vlassov equations describing the motion of particles in a self-consistent electromagnetic field. The electromagnetic field is determined by the Particle-In-Cell (PIC) method using a grid that that follows the curved field lines of the confining toroidal potential. A non-spectal solver is used to evaluate Poisson's equation on the grid. The parallel decomposition divides the spatial domain into 1-D segments along the toroidal dimension, and additional parallelism is achieved using a particle decomposition within the toroidal domains.

The GTC benchmark simulates 6.6×10^9 particles in 2.1×10^6 cells for 248 timesteps. The 2048 MPI processes partition 64 toroidal domains and with a 32-way particle decomposition. At this concurrency, three phases of the calculation dominate the GTC walltime. Deposition of the particles' charge onto the grid is both computationally demanding and, due to it's use of indirect addressing, a challenge for random access memory latency. The 'push' phase updates the position and momentum of each particle (based on the grid's field data). The 'shift' phase arises when particles move between toroidal domains and is characterized by bandwidth-limited nearest-neighbor communication.

4) IMPACT-T

IMPACT-T represents NERSC's accelerator physics workload. It simulates the relativistic motion of a beam of charged particles as they travel through the electromagnetic field generated by the accelerator structure. Coulomb forces among the particles are evaluated using a Particle-In-Cell algorithm. While there are fundamental similarities between the PIC algorithms used in GTC and IMPACT-T, there are two main reasons why these codes exercise the system differently. The external fields in the accelerator intentionally manipulate the particles into a nonuniform phase-space distribution, which manifests in simulations as a nontrivial load balance problem that is not present in GTC. Second, IMPACT-T uses an FFT-based algorithm to evaluate the particles' electric field, making it sensitive to MPI collective performance. The IMPACT-T benchmark uses 1024 MPI processes to simulate 400 million particles on a 128 x 256 x 256 grid for 200 timesteps.

5) MILC

The MILC code is used to study quantum chromodynamics, which is the theory of strong interactions between quarks and gluons which comprise hadrons (e.g., protons, neutrons, etc.). The code uses lattice gauge methods to implement the SU(3) Yang-Mills theory on a four-dimensional grid (three spatial coordinates and a single time coordinate). The variables stored on the sites and links are updated using the result of a large, sparse, near-singular system of linear equations. A Conjugate Gradient (CG) algorithm is used for the linear solve, and as a result of this iterative scheme, MILC performs many complex-valued matrix-vector operations and is sensitive to memory bandwidth [11]. MILC is parallelized using a 4-D domain decomposition designed to minimize the surface-to-volume ratio of the subdomains. The MPI communication pattern is a 4-D halo exchange implemented with nonblocking sends and receives, and results in messages between widely separated nodes on network topologies that do not map well to the 4-D decomposition. Additional MPI traffic is involved due to the all-reduce collectives required by the CG solver. Additional OpenMP parallelism was implemented by the authors following previous guidance by Gottlieb and Tamhankar (2001). In general, the OpenMP directives are aimed at exploiting parallelism inherent in loops over the number of 'sites' in the local lattice of an MPI task (a site is a data structure containing all variables at a point in the lattice).

The benchmark used in this paper is the 'extra large' MILC problem from previous NERSC benchmark suites, with a total lattice size of 64^3x144 (x,y,z,time), four trajectories, 15 steps per trajectory, and a timestep of 0.02. The benchmark uses 8,192 MPI tasks yielding an 8^3x9 local lattice and either one or two OpenMP threads.

6) MAESTRO

MAESTRO simulates low Mach number astrophysical flows such as, in this benchmark, convection within a white dwarf as it evolves toward a Type 1a supernova explosion. MAESTRO uses the BoxLib adaptive mesh refinement library [12] to integrate the relevant PDEs on a hierarchical patchwork of non-overlapping grids of different sizes and resolutions. A coarse-grained 3-D domain decomposition balances both the computation and communication loads among processors. MAESTRO's communication pattern is irregular, with a broad range of message sizes. MAESTRO has low computational intensity, making it sensitive to memory performance, especially latency. The MAESTRO benchmark problem propagates a fixed-size, block-structured 1024^3 grid (bypassing MAESTROs AMR capability) for ten timesteps, using 2048 MPI tasks.

7) PARATEC and MiniDFT

PARATEC and MiniDFT are plane-wave density functional theory codes for modeling materials. Given a set of atomic coordinates and pseudopotentials, they compute self-consistent solutions of the Kohn-Sham equations. For each iteration of the self-consistent field cycle, the Fock matrix is constructed and then diagonalized. To build the Fock matrix, fast Fourier transforms are used to transform orbitals from the plane wave basis (where the kinetic energy is most readily evaluated) to real space (where the potential is evaluated) and back. A CG (PARATEC) or Davidson diagonalization (MiniDFT) algorithm is used to compute the orbital energies and update the orbital coefficients.

PARATEC has been a component of past NERSC benchmark suites, which permits comparison between current and historical computational systems. More recently, the Quantum ESPRESSO (QE) package has surpassed PARATEC in popularity at NERSC. QE also implements 'task-group' parallelism for computing FFTs of multiple bands simultaneously, extending its potential to use higher levels of concurrency anticipated in future computational systems. MiniDFT is a minimal DFT code or "mini-app", developed by extracting essential routines from the full-featured QE code. Hybrid parallelism is implemented in MiniDFT using OpenMP and threaded BLAS and FFT libraries.

The PARATEC benchmark calculation requires 1,024 MPI processes and performs a single point SCF and force calculation for a 7 x 7 x 7 supercell of silicon with a plane-wave cutoff energy of 25 Ry. The MiniDFT benchmark requires 10,000 MPI processes and performs one iteration of the SCF cycle for a 10 x 10 x 10 supercell of magnesium oxide with a plane wave cutoff energy of 130 Ry.

B. *Application Performance Experiments*

We consider five possible use cases for HT and compare their utility for the application benchmarks described above. In the baseline scenario, 16 MPI tasks are assigned to each node, one for each physical core. The second case investigates the simplest approach to using HT - 32 MPI tasks are assigned to each node, one for each virtual core. (Two per physical core.) For a fixed total number of MPI tasks, this use of HT uses half as many nodes, thus decreasing the charge to users, since NERSC allocations are charged on a per-node basis. The runtime will typically increase in this case, but if it increases by less than a factor of two, then this scenario will be a net win. This perfomance increase can usually be attributed to some combination of HT's latency-hiding effects or reducing the amount of off-node communication. The third use case combines HT and CS; 31 MPI tasks are assigned to each node, and the remaining virtual core is reserved for OS functions. To accommodate the MPI tasks that were displaced by CS, slightly more nodes are required than the simple MPI+HT case.

For the four benchmark applications that include OpenMP directives (CAM, GTC, MILC and MiniDFT), we also perform hybrid MPI+OpenMP calculations with 16 tasks per node and use HT by assigning two OpenMP threads per MPI task. Contrasting these results to those of the MPI+HT test may clarify whether performance enhancements associated with HT (if any) are due to on-node latency-hiding or changes to the communication topology. The fifth use case combines the hybrid applications with HT and CS. Each node hosts 15 MPI tasks with two threads apiece, and one core is set aside for CS. TABLE I. lists the aprun options used to launch the executable in each experiment. For experiments using CS, the following environment variables were used to enable the MPI Progress Engine: MPICH_NEMESIS_ASYNC_PROGRESS=1, MPICH_MAX_THREAD_SAFTETY=multiple.

TABLE I. APPLICATION BENCHMARKING EXPERIMENTS.

Experiment	aprun options
MPI-Only	-N16
MPI+HT	-N32 -j2
MPI+HT+CS	-N31 -r1 -j2
Hybrid+HT	-N16 -d2 -j2 -cc numa_node
Hybrid+HT+CS	-N15 -r1 -d2 -j2 -cc numa_node

The Intel compiler and MKL libraries are the default programming environment and were used to compile all applications except PARATEC, for which the Cray compiler and Libsci libraries were used.

C. *Application performance results*

Results of the application benchmark experiments are listed in TABLE II. HT enables runs with twice as many processes per node, but the added competition for CPU resources roughly doubles their runtimes. If HT offers performance advantages, then the runtime will increase by less than a factor of two. Conversely, CS may decrease runtimes, but at the cost of slight increases in the number of nodes required. To account for a large range of absolute performance and resource allocations, we use performance per node (i.e., *FlopCount / Runtime / NodeCount*) as the basis for comparing multi-core use modes.

Fig. 6 shows the performance per node for each application code, relative to the MPI-Only performance. The MPI+HT columns (red) are typically greater than one indicating that HT improves performance per node for all applications except MAESTRO and MILC, and minimally for GAMESS. The Hybrid+HT mode outperforms the MPI+HT mode for MILC, but is worse for all other codes. CS (green and blue) impedes performance of all codes. The remainder of this section analyzes the performance of individual codes in more detail.

1) CAM

The CAM benchmark reports the time spent in the 'stepon' region, which is CAM's main time-stepping section over the dynamical and physical cores. Communication in this benchmark is dominated by vector gather/scatters in both the physics and dynamics and waits associated with the point-to-point communications in the dynamics. Besides the waits, non-blocking communications contribute very little to the overall run time. HT significantly increases the runtime, but uses half as many nodes, resulting in a 20% net increase in performance per node. This improvement is likely due to HT taking advantage of instruction stream stalls. The total amount of time spent in MPI communications also very nearly doubles indicating there is little overlap in communication and computation that the second thread can exploit. Indeed, the increase in communication is largely dominated by the waitalls associated with point-to-point communications. When CS is added, the overall stepon time increases again, largely due to increases in MPI communications time. In this case, even non-blocking communication time increased so the overall increase in communication time may result from more communications having to go off-node (to processes that were displaced by CS) even though there is less contention for the network on node.

TABLE II. BENCHMARK RESULTS

Code	Runtime (s)				
	MPI	*MPI+HT*	*MPI+HT+CS*	*Hybrid+HT*	*Hybrid+HT+CS*
CAM	156	259	270	147	168
GAMESS	487	963	>1800	N/A	N/A
GTC	515	833	829	665	687
IMPACT-T	280	483	491	N/A	N/A
MAESTRO	885	1540[a]	1700[a]	N/A	N/A
MILC	544	1168	1129	538	>7200
PARATEC	201	356	353	N/A	N/A
Mini-DFT	325	575	594	348	N/A

[a.] MAESTRO results with HT use 24 processes. See text.

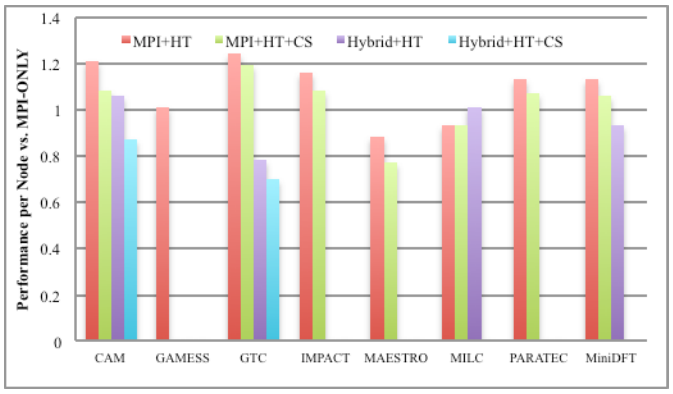

Fig. 6. Application performance per node, relative to MPI-Only case. Values greater than 1.0 indicate that HT increases in application throughput.

The Hybrid+HT case is interesting in that runs in nearly the same time (slightly less) as the MPI only case, indicating that, while each thread has one-half the work of an MPI-only task, each thread is sharing the total number of cycles approximately equally. These two effects balance each other so that the run time is very close to that of the MPI-only case. Profiling with IPM reveals that, while the run time decreased slightly relative to the MPI only case, the total amount of MPI time hardly changed, in agreement with the fact that no MPI calls occur within OpenMP regions, so threading has no impact on the number of MPI calls or the volume of data transferred by MPI. When we add CS (Hybrid+HT+CS), we note that the run time increases, as did adding CS to the MPI+HT case, even though the overall amount of time spent in MPI routines increases, possibly for the same reasons.

2) GAMESS

HT has no significant impact on GAMESS performance. The majority of the GAMESS benchmark time is spent transforming and contracting 4-index arrays- operations that can be implemented with BLAS routines that do not benefit from HT. The computational intensity of the transformations is high, and the DDI data-servers ensure good overlap between communication and computation, so doubling the communication volume per node via HT does not influence GAMESS performance. The GAMESS calculations using

DDI+HT+CS took more than twice as long as DDI+HT and did not complete within the allocated runtime. This is difficult to explain; it is possible that this DDI data-server implementation is not compatible with the MPI asynchronous progress feature.

3) GTC

GTC exhibits a 23% increase in performance per node when switching from MPI-only to MPI+HT, with 13% coming from HT's latency hiding effects (during the charge deposition and interpolation phases of the PIC algorithm), and 10% from the increased locality of the MPI_Allreduce functions. (With HT, all 32 particle domains within a toroidal domain are on the same node.) The Hybrid+HT run is 15% slower than the MPI-only run, suggesting that HT does not compensate for OpenMP thread synchronization. This underscores the observation that a large fraction of the MPI+HT performance improvement is due to reduced MPI communication volume and improved topology.

4) IMPACT-T

IMPACT-T's performance per node increases by 15% when HT is enabled. Roughly half of this improvement is due directly to recouping processor resources that are underutilized when running without HT. Interpolation of the electric field from the grid to the particles requires many random memory accesses, and HT can take advantage of these interruptions to the instruction stream. The other half of IMPACT-T's performance improvement is due to MPI_Allreduce and MPI_Barrier functions, which (for IMPACT-T) do not increase when HT is used. IMPACT-T spends very little time in non-blocking MPI functions, so CS has little potential to improve its absolute performance and decreases its performance per node.

5) MAESTRO

MAESTRO jobs that used more than 24 MPI processes per node consistently fail. (We are continuing to investigate and address these errors.) In TABLE II. and Fig. 6, the MPI+HT and MPI+HT+CS results are for 24 processes per node. This limitation makes MAESTRO's performance per node node anomalously low because one quarter of the node's CPU resources are not used.

To better understand MAESTRO's performance with respect to HT, we ran with 16 processes per node and pinned all processes to cores on the first socket. The benchmark time for the single-socket HT run was 1482.1 s, which corresponds to a 19% increase in performance per socket used (but leaves half of the allocated sockets unused). The network bandwidth, data volume and topology are the same as the MPI-only case, so this improvement is due entirely to HT's latency hiding effects, which is not surprising in light of MAESTRO's irregular memory access pattern.

6) MILC

The reported benchmark times for MILC are for the main computational section and exclude any initialization or I/O overhead. Communication in MILC is dominated by the MPI_Allreduce required by its conjugate gradient solver and waits associated with non-blocking communications. The actual non-blocking calls (Isend/Irecv) take little time in comparison. Furthermore, all of the MPI calls are serialized on the master thread with no MPI communications occurring in OpenMP regions. Adding HT to the MPI-only case more than doubles the run time. Examining the 'compute time' (run time – MPI time), we note that the compute time of the MPI+HT case more than doubles, strongly implying that there is contention for memory resources between the two tasks. Adding CS slightly decreases the run time; however, while the MPI time relative to the MPI+HT runs goes up, the 'compute time' goes down relative to the the MPI+HT case. This result may be a combination of MPI wait and collective times increasing due to the greater off node communication and compute times decreasing because CS is handling more system services (e.g. I/O) on node.

As with CAM, when running with two OpenMP threads per MPI task (where each thread has half the work of an MPI-only task) the run time is nearly the same as the MPI only run as each thread's instruction stream is nearly perfectly interleaved. The amount of time spent in MPI calls is also nearly the same since only one OpenMP thread is handling MPI traffic. Adding CS presents an interesting problem as the run did not finish after two hours, which was more than sufficient time to complete based on previous runs and experience. Indeed, similar runs (384 and 1536 MPI tasks) on smaller models completed in the expected time.

7) PARATEC

HT improves PARATEC's performance per node by 13%. Closer examination of the profiling data collected with IPM shows that when the MPI time is excluded, HT decreases performance per node by 3%. This is expected because PARATEC's computation phases are dominated by extremely efficient BLAS and FFT kernels. The MPI_Allreduce function accounts for the largest fraction of PARATEC's MPI time, and increases by only 12% when HT is enabled. This is consistent with a prefix-sum reduction that sends the same volume of data off-node (or off-blade) regardless of the number of ranks per node.

CS provides very little improvement for PARATEC's absolute performance and decreases its performance per node. It is not surprising that PARATEC does not benefit from CS; PARATEC uses non-blocking MPI calls only for the parallel transpose phase of the 3D-FFTs, and very little computation is available to overlap with communication.

8) MiniDFT

For MiniDFT, the MPI+HT runs have 13% higher performance per node than MPI-only. This improvement is due to a decrease in the fraction of time spent in MPI_Barrier calls. MiniDFT uses only half of the available cores during the diagonalization phase of the SCF cycle- the others wait in an MPI_Barrier, creating a significant load imbalance. HT provides a performance benefit during this because the diagonalization processes can monopolize more processor resources on cores that are shared with waiting processes. Comparison of the MPI and Hybrid+HT times also indicates that MPI behavior is responsible for the improved performance of the MPI+HT run. MiniDFT does not use non-blocking communication functions and does not benefit from CS. The

Hybrid+HT+CS run was not attempted because it requires more nodes than are available on the current Edison system.

V. Cross-platform performance comparison

The “NERSC-6” SSP suite consists of all the codes in the preceding sections excluding MiniDFT. To compare the XC30 and XE6 architectures we show, in Fig. 7, the performance per core for the SSP benchmarks using the MPI-only settings from TABLE I. Edison improves upon Hopper’s performance per core by 1.7 – 3.0x. MILC’s improvement is somewhat less dramatic than other codes, which is due in part to its high concurrency, which forces use of the rank-3 network that has significantly lower bandwidth than the intra-group networks. MILC may also have lower performance gains because Edison has fewer cores per node than Hopper, so a larger fraction of the MPI calls used for its 4-D halo exchange require off-node communication. Fig. 7 also shows the relative change in performance per node for the MPI-Only case, which may be a fairer basis for comparing the systems because electrical power per node is the same on both systems. For the “optimized” performance per node, we select the best performing HT and CS use mode for each code. Core for core, Edison is 2.2x faster than Hopper (based on geometric mean on SSP benchmarks). On a node-for-node basis, Edison is 1.4x faster, and offers 1.6x greater throughput if HT is used.

VI. Conclusion

We have evaluated the performance of Edison, the first phase of the Cray XC30 system being installed at NERSC. Our measurements of the point-to-point latency (1.5μs) and bi-directional bandwidth (15GB/s) for the Aries interconnect match earlier reports from a prototype system, and constitute significant improvements over the Cray XE6. Dramatic increases in global bandwidth are reflected in the performance and scalability of all-to-all communication benchmarks. We have measured the effects of HT and CS on the performance of a suite of benchmark applications. HT increases performance for six of the eight codes studied, and CS hurts performance for all but one code. Based on the NERSC-6 SSP benchmarks, Edison 2.2x faster per core than Hopper, and 1.4x faster per node. Selective use of HT increases Edison’s performance per node to 1.6x that of Hopper.

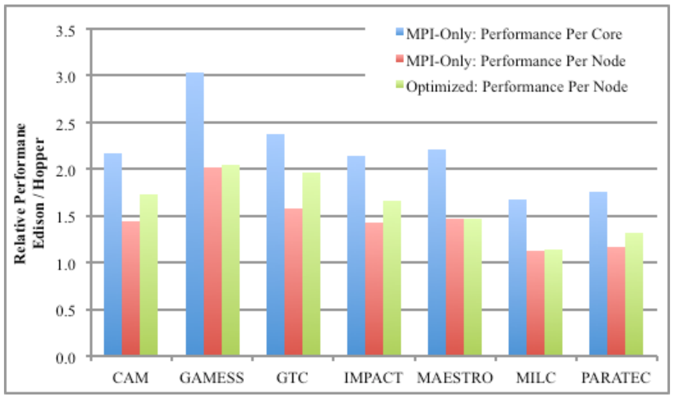

Fig. 7. Relative performance for SSP benchmarks on Edison and Hopper.

References

[1] Kessler, R. E. and Schwarzmeier, J. L., “Cray T3D: A New Dimension for Cray Research,” Proc. Papers, COMPCON Spring ‘93, San Francisco (February): 176-182, 1993.

[2] Scott, S. L. and Thorson, G. M., “The T3E Network: Adaptive Routing in a High Performance 3D Torus,” Proc. HOT Interconnects IV, Stanford University, August 15-16, 1996.

[3] H. Pritchard, D. Roweth, D. Henseler, and P. Cassella, “Leveraging the Cray Linux Environment Core Specialization Feature to Realize MPI Asynchronous Progress on Cray XE Systems,” Proc. Cray User Group, 2012

[4] K. Antypas, T. Butler, J. Carter, “The Hopper System: How the Largest XE6 in the World Went From Requirements to Reality”, Proceedings of Cray User Group, 2011

[5] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins and J. Reinhar, “Cray Cascade: a Scalable HPC System based on a Dragonfly Network,” SC12, November 10-16, 2012, Salt Lake City, Utah, USA

[6] Gottlieb S, and S. Tamhankar, Benchmarking MILC code with OpenMP and MPI. Nucl.Phys.Proc.Suppl. 94 (2001) 841-845

[7] D. Bureddy, H. Wang, A. Venkatesh, S. Potluri, and D.K. Panda, “OMB-GPU: A Micro-Benchmark Suite for Evaluating MPI Libraries on GPU Clusters, Recent Advances in the Message Passing Interface, Lecture Notes in Computer Science, Treff, Jesper Larsson, Benkner, Siegfried Dongarra, Jack J., Eds., Springer Berlin Heidelberg, 2012. Available from http://mvapich.cse.ohio-state.edu/benchmarks/

[8] Antypas, K., Shalf, J., Wasserman, H., “NERSC-6 Workload Analysis and Benchmark Selection Process”, LBNL Technical Report, August 13, 2008, LBNL 1014E

[9] Intel Corp., “Intel 64 and IA-32 Architetures Optimization Reference Manual, Intel Order Number 248966-025”, June 2011

[10] H. Shan, N. J. Wright, J. Shalf, K. A. Yelick, M. Wagner, and N. Wichmann, “A preliminary evaluation of the hardware acceleration of Cray Gemini interconnect for PGAS languages and comparison with MPI”, SIGMETRICS Performance Evaluation Review 40 (2012) 92-98

[11] J. Carter, Y. He, J. Shalf, H. Shan, E. Strohmaier, H. Wasserman, “The Performance Effect of Multi-Core on Scientific Applications”, Proc. Cray User Group, 2007, May 2007, LBNL 62662

[12] BoxLib http://ccse.lbl.gov/BoxLib/index.html

[13] Cray Inc. “The Gemini Network”, Rev 1.1, August 8, 2010