Characterizing Parallel Scientific Applications on Commodity Clusters: An Empirical Study of a Tapered Fat-Tree

Edgar A. León, Ian Karlin, Abhinav Bhatele, Steven H. Langer,
Chris Chambreau, Louis H. Howell, Trent D'Hooge, and Matthew L. Leininger

Lawrence Livermore National Laboratory
Livermore, California 94550 USA
Email: {leon, karlin1, bhatele, langer1, chambreau1, howell4, tdhooge, leininger4}@llnl.gov

Abstract—**Understanding the characteristics and requirements of applications that run on commodity clusters is key to properly configuring current machines and, more importantly, procuring future systems effectively. There are only a few studies, however, that are current and characterize realistic workloads. For HPC practitioners and researchers, this limits our ability to design solutions that will have an impact on real systems.**

We present a systematic study that characterizes applications with an emphasis on communication requirements. It includes cluster utilization data, identifying a representative set of applications from a U.S. Department of Energy laboratory, and characterizing their communication requirements. The driver for this work is understanding application sensitivity to a tapered fat-tree network. These results provided key insights into the procurement of our next generation commodity systems. We believe this investigation can provide valuable input to the HPC community in terms of workload characterization and requirements from a large supercomputing center.

Index Terms—**High performance computing, high-speed networks, network topology, computer performance, scientific computing.**

I. Introduction

Understanding the characteristics and requirements of applications that run on high performance computing (HPC) systems is an important component in the design, configuration, and optimization of current and future machines. Hardware architects and system administrators, for example, can use this information to tune machine parameters and software libraries to meet a specific power budget, improve application performance, or maximize throughput. At the same time, those responsible for future hardware procurements can make informed decisions regarding a range of available options with respect to the performance and reliability of processor, memory, and network components. A question one may ask is whether doubling the network bandwidth is worth sacrificing upgrades in processor speed, given a fixed capital budget. The right balance depends on the suite of applications that will execute on such a system. The more we understand the characteristics and requirements of applications, the better decisions we can make to provide the best performance within a given capital or power budget.

The interconnection network is a key component that distinguishes HPC systems from more loosely coupled clusters. Many HPC application workloads require both local and global communication, and are therefore, sensitive to network bandwidth, message latency, message injection rate, and implementations of collective operations. While much emphasis is placed on HPC networks, it is just as crucial not to over-provision a network. For example, some applications may be dominated by local communication or may involve a large number of smaller ensemble runs. Systems dominated by such workloads may or may not require a powerful global network. Therefore, studies that provide insights into the network requirements of application workloads can be extremely useful in designing and procuring future systems. This is particularly important since HPC networks are typically 15-25% of the overall machine cost.

In this work, we provide a detailed characterization study of several applications that run on commodity clusters at Lawrence Livermore National Laboratory (LLNL), a U.S. Department of Energy (DOE) laboratory. We focus on two aspects: identifying a set of workloads, input problems, and problem sizes that are representative of day-to-day simulation needs of DOE's Advanced Simulation and Computing (ASC) program; and characterizing these workloads from an HPC network perspective. The goal of this study is to provide a meaningful and detailed characterization that researchers, system developers, and domain scientists can use to understand *real* workloads in high-end scientific applications with an emphasis on DOE needs. Although benchmarks and micro-benchmarks are helpful in a number of ways, having an informed understanding of the characteristics of real workloads can greatly improve the effectiveness and accuracy of proposed approaches to improve performance, scalability, and throughput.

This paper focuses on three areas. First, we present detailed information about jobs that are executed on LLNL's commodity systems with a particular emphasis on job size and completion time (Section II). Second, we identify a representative set of applications, and for each application, describe the physics

SC16; Salt Lake City, Utah, USA; November 2016
978-1-4673-8815-3/16/$31.00 ©2016 IEEE

involved, input problems, and problem sizes (Section III). And, finally, we describe in detail the communication characteristics of these applications and present a case study on how this characterization work has been used to evaluate performance sensitivity to a tapered fat-tree network (Sections IV and V).

We undertook this case study to inform the procurement of new commodity systems at LLNL. From a procurement perspective, this investigation enables a better evaluation of the tradeoffs presented by various machine features, their cost, and their impact on application performance. For example, is it worth purchasing additional network bandwidth for a given capital cost? Or, can these funds be used more effectively from an application throughput perspective, by purchasing additional compute nodes? We summarize the contributions of this study as follows:

- A workload characterization of scientific applications on commodity clusters that is current and representative of a large supercomputing center at a DOE laboratory. This characterization includes the physics involved, job sizes, network bandwidth utilization, and communication characteristics of applications.
- An experimental study on a large cluster demonstrating the impact of a tapered fat-tree on application performance. This investigation is guiding the procurement of a next-generation commodity cluster to increase application throughput and productivity.

We believe the methodology used in this work can be used by other supercomputing centers to understand the characteristics of their own applications. Further, our representative workload characterization can be used as a reference by HPC practitioners and researchers to understand the impact of their techniques on real applications.

II. Cluster Utilization in Production

In this section, we describe the production usage of two of LLNL's commodity systems focusing on two aspects: the types of jobs and their sizes, and the network utilization.

A. Execution Environment

We used two Linux clusters at LLNL: *Cab* and *Zin*. The Cab machine is comprised of 1,296 compute nodes, while Zin is larger and has 2,916 nodes. Both systems have the same architecture and run the same software stack as described here. Each node has two Intel Sandy Bridge (Xeon E5-2670) processors and 32 GB of memory. Each processor has eight cores with two hardware threads per core (Hyper-Threading). The nodes are connected via an InfiniBand QDR (QLogic) single-rail network and the routing algorithm employed is FTREE, a fat-tree optimized routing with no credit loops.

The machines run the Tri-lab Operating System Software (TOSS). At the time of the experiments, Cab and Zin were running TOSS version 2.3, which is based on Red Hat Enterprise Linux Server release 6.6. Each processor has a theoretical peak memory bandwidth of 51.2 GB/s using 1.6 GHz DDR3. With the exception of Section II-C, where we use both Cab and Zin, all other experiments were executed on Cab.

B. Network Utilization

In order to understand how the network is being exercised in production, we recorded the utilization of every inter-switch link on Cab for a period of two weeks. Measurements were taken every three minutes. We captured a week of data for the full fat-tree in August 2015. In September 2015, we reconfigured the network to a tapered fat-tree (as explained in Section IV) and captured similar data for a week.

Figure 1 shows the maximum and average link utilization over all links in the fat-tree. The full fat-tree graph shows that most of the links utilized less than 50% of their maximum capacity based on the average utilization during three-minute recording intervals. Furthermore, the average link utilization is significantly low–at most 2.4%. This large difference between the average and maximum link utilization indicates that network traffic is not distributed evenly over all links. We also verified this with a heat map of the utilization of all the links.

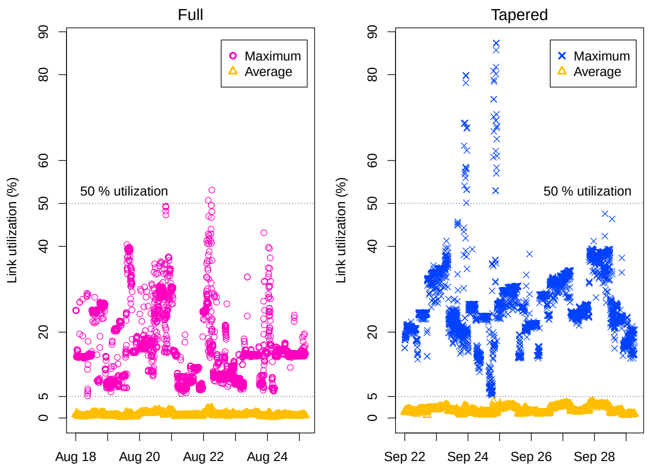

Fig. 1. Network link utilization over one week during production runs. We recorded the average utilization of each inter-switch link over three-minute intervals. The Y-axis shows the average and maximum link utilization across all links in the fat-tree for each three-minute interval.

We observe a similar behavior for the tapered fat-tree, except for several data points where a few links were utilized at 50 to 90% of their maximum capacity. After further analysis, we attributed these data points to two jobs from a single user. Each of these jobs used 48 nodes (less than 4% of the cluster size) for four hours. Although a few links were exercised significantly by a single user, most of them were not, the average link utilization reached no more than 3.9%. The tapered fat-tree does show an increase in network link utilization compared to the full fat-tree but still we did not observe a noticeable increase in maximum link utilization over time other than a few outliers.

It is important to note that, on an average, 95.8% of the nodes in the cluster were allocated to jobs during the week of interest in August and 96.0% during the week of interest in September 2015. This indicates that even though the machine was mostly full in both fat-tree setups, the running jobs did not fully exercise the network. This is an important observation to consider as we procure future systems, but we need to verify that the performance of individual applications is not degraded when switching to the tapered fat-tree. We therefore analyze the types of jobs, their sizes, their communication characteristics, and their required network bandwidth for a representative suite of applications. This will help us understand why the network is not being fully utilized and identify opportunities to optimize investments without affecting job throughput.

C. Workload Characterization

In this section, we describe the characteristics of the jobs running on Cab and Zin at LLNL. We focus on two aspects: the number of jobs of a given size and the time consumed by these jobs. Our commodity systems record information about every submitted job such as time to completion, number of nodes allocated, and number of processes per node. We collected this information on Cab and Zin for four months to understand how users utilize these systems.

We start with the Cab system. First, we show the overall machine time consumed by jobs of a given size ranging from 1-node jobs to 1,050-node jobs. The time consumed by a given job j is weighed according to the number of nodes it employs:

$$WeightedTime(j) = AbsoluteTime(j) \times NumNodes(j)$$

$$\%TimeUsed(j) = \frac{WeightedTime(j)}{\sum_i WeightedTime(i)} \times 100$$

The left graph of Figure 2 shows the cumulative sum of the machine time consumed by job size. We observe that a large percentage of the time is consumed by small jobs. Jobs using 64 or fewer nodes consume 75% of the time. Jobs using 128 or fewer nodes use almost 90% of the total time. Jobs using 256 or more nodes use only 3.4% of the time.

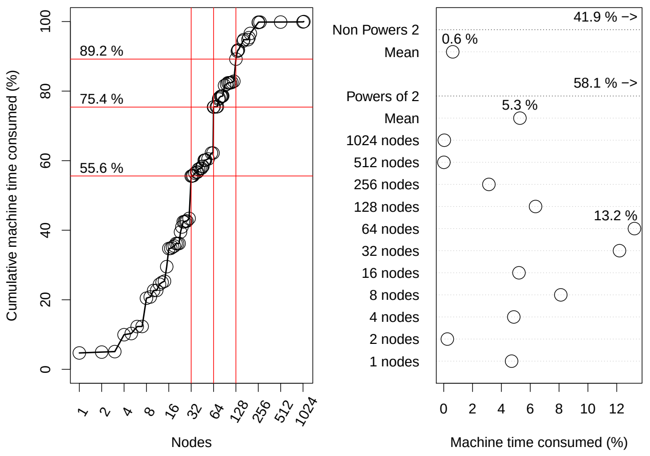

Fig. 2. Machine time consumed by jobs of a given size.

The right graph of Figure 2 shows the time consumed by each job grouped by power-of-two and non-power-of-two nodes. For powers of two, the graph shows all of the jobs in this category. We note that power-of-two node jobs occupy a disproportionately large amount of time: 58.1% of the time is spent on 11 job sizes, while 41.9% is consumed by the remaining 67 job sizes. From all of the sizes, jobs of size 32 and 64 nodes are the ones that accumulate the most time.

Second, we do a similar analysis for the number of jobs of a specified size relative to the total number of jobs during the time period mentioned above. The left graph of Figure 3 shows most of the executed jobs are very small: 80.8% of the jobs are of size 4 nodes or smaller, while 95% of the jobs are of size 16 or smaller. This is because 76% of the jobs are 1-node jobs, followed by 16-node jobs representing another 4% of all jobs, as shown in the graph on the right.

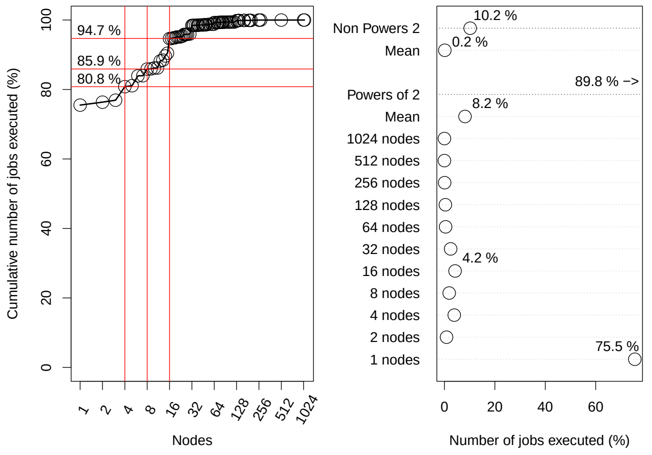

Fig. 3. Number of executed jobs of a given size.

TABLE I
JOB CHARACTERISTICS OF ZIN AND CAB. JOB SIZE REFERS TO THE NUMBER OF NODES USED BY THAT JOB.

	Cab	*Zin*
Machine comprises Y nodes	1,296	2,916
Jobs of size ≤ 64 use $Y\%$ of the overall machine time	75	69
X power-of-two job sizes occupy $Y\%$ of the machine time	11, 59	11, 82
X non-power-of-two job sizes occupy $Y\%$ of the machine time	67, 41	46, 18
Jobs of size ≤ 16 comprise $Y\%$ of all jobs	95	91
Jobs of size X comprise $Y\%$ of all jobs	1, 76	8, 19
$X\%$ of jobs are of power-of-two size	90	76

We performed a similar analysis for the Zin system and created Table I to compare the two machines. It is important to note that Zin, a classified machine, is 2.25 times larger than Cab, while Cab, an unclassified machine, has a more diverse mix of jobs and more users. Even with these differences, Zin shows similar patterns: a large percentage of time (69%) is consumed by 64-node and smaller jobs; power-of-two job sizes are even more salient on Zin occupying 82% of the overall time (compared to 59% on Cab) and comprising 76% of all the jobs. Because of the larger mix of jobs on Cab, Zin presents a smaller number of job sizes: 11 power-of-two sizes and 46 otherwise. A significant difference between the two system is that the most frequent job size on Zin is 8 nodes (19% of all jobs) while for Cab it is 1 node (76% of all jobs).

From these experiments, it is evident that power-of-two node jobs of small size are extremely important to understand, characterize, and optimize the utilization of a cluster not only in terms of performance but also for power and energy concerns as we move to future systems.

III. Representative Applications

We provide an empirical study of the sensitivity of applications to a tapered fat-tree by employing a representative suite of HPC parallel codes. These codes include five MPI production applications that are often run at LLNL and two MPI+OpenMP proxy applications from the CORAL benchmarks[1]. The CORAL suite, used to procure three 100+ Petaflop/s computers, represents U.S. Department of Energy workloads and technical requirements. The chosen codes represent diverse physics and application areas including hydrodynamics simulations, Monte Carlo particle transport, nuclear reactor criticality, and laser-plasma interactions.

The rest of this section provides a brief description of each application and the problems used in this study. Each input problem and size was carefully selected with the guidance of the application developers that regularly run and maintain these codes to ensure the problems tested were representative of production-class configurations. We discuss how our codes are changing to adapt to new architectures in Section VI.

ALE3D is an unstructured mesh Arbitrary Lagrange Eulerian (ALE) hydrodynamics code [1]. It runs a diverse set of application use cases, including, but not limited to fracture and fragmentation, magneto-hydrodynamics and fluid structure interaction. ALE3D is used by various U.S. government agencies including LLNL, the DOD and NASA. For ALE3D, we tested a penetrator problem, which is representative of problems that have a mixing of materials. In this problem, a moving penetrator impacts a stationary target material. The problem size used is typical of 3D runs in ALE3D that use 10,000 to 15,000 zones per processor.

Mercury is a production Monte Carlo particle transport code [2]. MPI parallelism in Mercury is over spatial domains. Since some domains will have more particles than others, the code is typically run with more MPI tasks than domains so that the code can replicate expensive domains onto multiple tasks handling different sets of particles to better balance the load. We ran a 3D neutron transport test problem called Godiva-in-water. A sphere of uranium is immersed in a water moderator which in turn is surrounded by air; we ran an alpha eigenvalue calculation on this system which required 40 cycles (pseudo-timesteps) of the Monte Carlo algorithm. Though the geometry is relatively simple, this problem uses real materials and relies on domain replication for load balancing since the particle densities vary dramatically in different spatial regions. These properties and the 15,000 particles used per MPI task make the problem representative of typical Mercury production runs. All tests used a $48 \times 48 \times 48$ spatial mesh divided into 64 domains (independent of the number of MPI tasks).

Ardra is a discrete ordinates (S_n) neutron transport code used for various engineering problems [3]. Ardra solves the Boltzmann transport equation on a structured grid. In this paper, we focus on the 2D time dependent problem. We focus on this problem because Ardra usually runs 2D problems on commodity clusters and the time dependent problem highlights the sweep communication patterns. The other communication pattern in Ardra is when it calls AMG for eigenvalue computations used in reactor criticality problems, but AMG is tested on its own. Our test problem has 200 zones per task, with 192 angles and 48 groups per zone and is representative of typical production sizes.

Miranda is a radiation hydrodynamics code designed for direct numerical solution (DNS) or large-eddy simulation (LES) of multicomponent flows with turbulent mixing [4]. It is often used for simulating Rayleigh-Taylor [4] and Richtmyer-Meshkov [5] instability growth. For experiments in this paper, we used a 2D Noh problem that is decomposed into sub-domains of size 48×48 grid points per MPI task. MPI tasks are arranged in a logical 2D grid and each process is assigned a sub-domain.

AMG2013 is an algebraic multigrid benchmark derived from the BoomerAMG solver in Hypre [6]. The Hypre linear solvers library is used in a variety of multi-physics applications at LLNL including KULL [7], BLAST [8], and ALE3D [1]. We use AMG to test two common problems. The first is a standard 7-point stencil, Laplace problem in 3D with grid and anisotropy (1.0 in each direction). The second is a Laplacian in 3D with a 27-point stencil. The solver we employ uses algebraic multigrid as a preconditioner for GMRES (Generalized Minimum Residual) with 10 Krylov vectors. Unlike CG (Conjugate Gradient), GMRES works for more general non-symmetric matrices. We ran both problems and obtained similar results. Thus, we only focus on the Laplacian with 7-point stencil for the reminder of the paper.

pF3D simulates laser-plasma interactions (LPI) in National Ignition Facility (NIF) [9] experiments. It solves coupled LPI equations for the propagation and interaction of the laser light, the scattered light, and the plasma. Some terms in the LPI equations are treated using 2D FFTs in planes transverse to the laser direction and others are treated using finite differences. pF3D also simulates the hydrodynamic response of the plasma to heating by the laser. pF3D uses a regular 3D Cartesian grid with a 3D spatial decomposition into equal-sized MPI domains. The laser beam propagates in the z-direction. A set of MPI domains spanning the full grid in the x and y directions and one domain thick in z is referred to as an xy-slab. We ran a test problem, typical of runs on x86_64 clusters, comprised of 384k zones per domain and 4×16 domains per xy-slab. We

[1]https://asc.llnl.gov/CORAL-benchmarks

ran a weak scaling study using this test problem. Weak scaling is achieved by keeping the height, width, and thickness of an xy-slab constant and adding more slabs in the z-direction.

UMT2013 is a deterministic transport (S_n) proxy application [10]. It solves 3D radiation transport equations on an unstructured grid. UMT uses both threads and MPI to increase available parallelism and scalability. A typical run has around 1,000 zones per MPI task to allow arrays to fit in memory. Problems with 3,000 zones per task can be run when more memory per MPI process is available. UMT is typically run with 15 groups, although some runs use up to 200 groups. For S_n order, S_6 is standard and S_{16} is extreme. For problems with high angular and energy resolution, the number of zones can be reduced down to 100 per task. For this work, we used the Su-Olson test with the following parameters: $12 \times 12 \times 12$ zones per MPI task, 15 groups, order 6, 8 polar angles, and 4 azimuthal angles.

A. *Summary of Application Areas and Inputs*

The applications we selected represent a diverse set of physics, solver methods, and grid types. While it is not practical to highlight all the areas within scientific computing these applications belong to, we summarize the key high-level characteristics described in this section in Table II. We also summarize the input problems and their sizes in Table III.

TABLE II
PHYSICS AND MATHEMATICS DOMAINS OF APPLICATIONS AT A GLANCE.

	Transport	Hydrodynamics	Monte Carlo	Sparse Linear Algebra	Plasma Interactions	Unstructured Grid	Structured Grid
ALE3D		🟩				🟩	
Mercury	🟩		🟩				
Ardra	🟩			🟩			🟩
Miranda		🟩					🟩
AMG2013				🟩			🟩
pF3D		🟩			🟩		🟩
UMT2013	🟩					🟩	

IV. A Tapered Fat-tree

The *Cab* system at LLNL uses a fully provisioned fat-tree network with two logical tiers of switches. As Figure 4 shows there are two *core* switches at the top, each with 648 InfiniBand 4X QDR (4x10 Gb/s) ports. Each core switch is constructed using 54 36-port switches arranged in two levels. Each core switch is connected down to eight *scalable units* (SU), each comprising nine second-level switches and 162 compute nodes. Each second-level switch or *edge* switch has 36 ports: 18 of which connect down to compute nodes (one port per node), nine up to core switch 1, and nine up to core switch 2. The total number of ports at the top level are calculated as follows: 8 SUs $\times$ 9 edge switches $\times$ 9 $\times$ 2 ports up to core switches = 648 $\times$ 2 top-level ports.

TABLE III
APPLICATION INPUT PROBLEMS AND SIZES. ALL CODES WERE EXECUTED USING 16 MPI TASKS PER NODE, WHICH RESULTS IN 1 TASK PER CORE.

	Problem	Size
ALE3D	Penetrator	10 K – 15 K zones per task
Mercury	Godiva-in-water eigenvalue	15 K particles per task
Ardra	Time dependent	200 zones per task, 192 angles, 48 groups
	Eigenvalue	
Miranda	2D Noh	48×48 grid points per task
AMG2013	Laplace, 27-point stencil	$40 \times 40 \times 40$ grid points per task
	Laplace, 7-point stencil with grid and anisotropy	
pF3D	Laser-plasma interaction	384 K zones per domain, 4×16 domains per XY-plane
UMT2013	Su-Olson test	$12 \times 12 \times 12$ zones per task, order 6, 15 groups

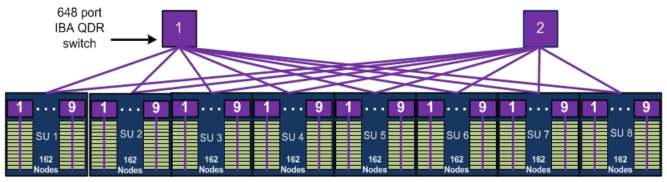

Fig. 4. A fully provisioned fat-tree.

We created a 2:1 over-subscribed fat-tree by cutting network bandwidth at the top level of the fat-tree in half. We accomplish this by turning off one core switch as shown in Figure 5. This results in a tapered fat-tree with half as many links at the top level: 8 SUs $\times$ 9 edge switches $\times$ 9 ports up to core switches = 648 top-level ports.

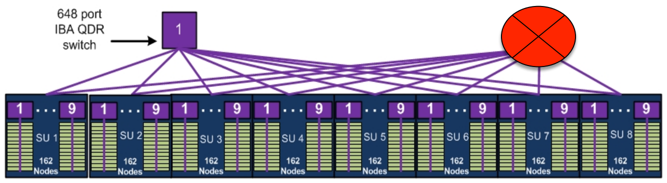

Fig. 5. A 2:1 tapered fat-tree.

A. *Evaluation using Micro-benchmarks*

We measured network performance differences between the full and tapered fat-tree setups to empirically validate the tapered fat-tree configuration on Cab. We used the OSU micro-benchmarks[2] to measure network latency and message injection rate (in million messages per second or MM/s) over the entire system. Table IV shows the results. The latency is the same for both fat-tree configurations. This is also the case for the message injection rate of small messages. However, injection rate of messages greater than 4 KB starts to differ.

[2]http://mvapich.cse.ohio-state.edu/benchmarks

TABLE IV
BENCHMARK RESULTS CONFIRM A REDUCTION IN AGGREGATE BISECTION BANDWIDTH.

	Full	Tapered
Latency (μs)	2.31	2.31
Message injection rate (MM/s for 1 byte)	6509	6657
Aggregate near-neighbor bandwidth (GB/s)	3577	3586
Aggregate bisection bandwidth (GB/s)	**3088**	**1866**

Using the Phloem MPI benchmarks[3], we measured two types of aggregate bandwidth: near-neighbor traffic between nodes on the same edge switch and bisection bandwidth across the top-level switches. As expected, the aggregate bandwidth for nodes on the same edge switch was the same while the bisection bandwidth was reduced by roughly half. As Figure 6 shows, we also observed that the full fat-tree did not provide a significant bandwidth increase for small messages. The bandwidth increase is significantly larger for messages greater than 32 KB.

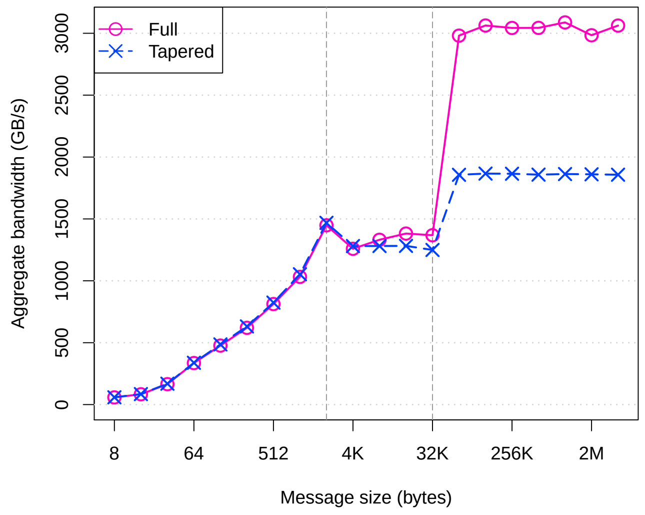

Fig. 6. Aggregate bandwidth for the full and tapered fat-trees. Based on the bandwidth changes, we consider 2 KB and 32 KB the thresholds for classifying messages into small, medium, and large.

The over-subscribed bisection results are not precisely half of the full fat-tree bisection results due to differences in node allocation by the resource manager and total node count between runs: 1,164 for full and 1,180 for tapered.

To summarize, the results of the micro-benchmarks demonstrated the expected reduction in bisection bandwidth and showed no significant impact on latency, injection rate of small messages, and near-neighbor bandwidth.

V. CHARACTERIZING APPLICATIONS

In this section, we describe the communication characteristics of the applications we tested. This characterization includes an analysis of their sensitivity to a tapered fat-tree as described in Section IV. The results presented in this section were collected in a dedicated system reservation on Cab over a period of two days. The machine was run in the tapered fat-tree setup on the first day and the full fat-tree setup on the second day. Each application was run in both setups. For each application, problem size, and node count, we executed at least five runs. The number of nodes used ranges between 8 and 1024 depending on the application. All experiments were run using 16 MPI tasks per node. In addition, we used the mpiP profiling library[4] to break down the MPI time into individual communication operations and their message sizes.

Based on the full and tapered fat-tree results, we have grouped the applications depending on their sensitivity to network bandwidth and their runtime variability. ALE3D (Group I) shows no sensitivity to network bandwidth or runtime variability. Group II, comprising Mercury, Ardra, Miranda, and AMG, demonstrates no sensitivity to network bandwidth but has significant run-to-run variability due to system noise. pF3D (Group III) is similar to Group II, except the runtime variability comes from a different unidentified source. UMT (Group IV) shows sensitivity to network bandwidth but no variability.

We also present a high-level overview of the size and type of messaging found in our applications in Table V. We classified messages into small, medium, and large with thresholds of 2 KB and 32 KB. These thresholds were chosen based on the network bandwidth changes observed when measuring the aggregate bandwidth for the full and tapered fat-trees as shown in Figure 6. Note that pF3D does not have significant global communication. It has significant collective communication because all-to-all operations on sub-communicators account for the majority of message passing time and bytes transferred.

TABLE V
MESSAGE SIZES AND TYPE OF COMMUNICATION OPERATIONS AT A GLANCE. THE THRESHOLDS FOR SMALL, MEDIUM, AND LARGE POINT-TO-POINT MESSAGES ARE 2 KB AND 32 KB.

	Small	Medium	Large	Collectives
ALE3D		🟩		🟩
Mercury		🟩	🟩	🟩
Ardra	🟩			🟩
Miranda	🟩			🟩
AMG2013	🟩			🟩
pF3D			🟩	🟩
UMT2013			🟩	🟩

For the rest of this section, we use three graphs to summarize our results across all applications. Figure 7 shows the breakdown of execution time into computation and individual communication operations. Figure 8 shows the average message sizes weighted over the different call-sites of each communication operation. Finally, Figure 9 shows the execution time of the applications under the full and tapered fat-tree.

[3]https://asc.llnl.gov/sequoia/benchmarks

[4]http://mpip.sourceforge.net

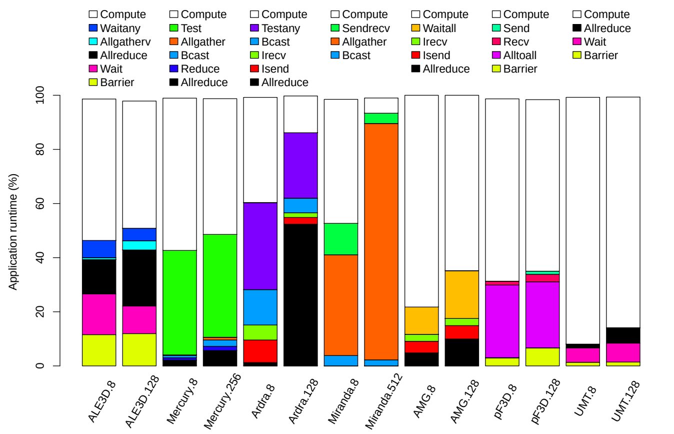

Fig. 7. Percentage of application runtime spent in computation and communication for small and large node counts. Communication calls with runtime less than 1% of the overall time are not shown, thus, not all bars reach 100%. The X-axis labels consist of the application name followed by the job size in number of nodes. Each application has its own legend.

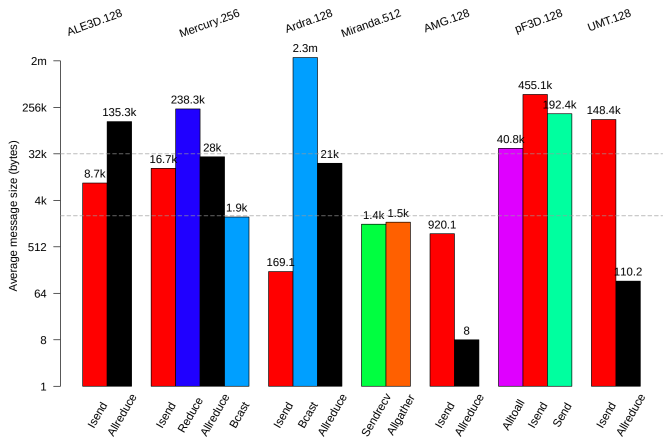

Fig. 8. Average message sizes for a large node count execution. These sizes are the weighted averages over the different call-sites of a given communication call. The calls shown in this graph are those whose call-sites appear in the top 20 sizes in descending order, whose aggregated sent percentage is greater than 0.009, and those that are represented in Figure 7. Within an application, communication calls are shown in descending order of percentage of data sent. Dotted horizontal lines mark the thresholds between small, medium, and large message sizes.

A. *Group I: ALE3D*

Applications in Group I had little or no run-to-run performance variability and were not impacted by tapered bandwidth reductions. The only application we ran that fit into this category was ALE3D. Figure 7 shows most of the messaging time in ALE3D is in global communication (Barrier, Allreduce) or wait functions. Time spent in global communication is driven by load imbalance. Physics timers in the code show the longest domain takes 1.5-2$\times$ longer than the shortest and this grows to 2-2.5$\times$ for domains with slide surfaces. This imbalance leads to significant waiting at global synchronization points and for asynchronous messages leading to most of the MPI time.

Figure 8 shows that messages in ALE3D are mostly of medium size. The average Allreduce is large but this is affected by initialization, and over 99% of the bytes in ALE3D are from Isends in halo exchanges or other point-to-point messages. In addition, most of the Allreduce time is for 8 byte messages. What is not shown on the graph is that the average size for the point-to-point messages decreases with scale from about 13 KB at 8 nodes to 8.7 KB shown at 128 nodes.

Figure 9 shows that ALE3D's runtime is identical for both network configurations. In addition, ALE3D's runtime was repeatable from run to run. Tapering did not impact ALE3D because it sends mostly medium size messages and does not communicate much data. If all of its communication was off node at maximum line rate it would take just 5% of the total execution time. In practice, a significant number of messages stay within a node and ALE3D overlaps significant amounts of its communication with computation. Therefore, ALE3D is not sensitive to network bandwidth on our test cluster. In addition, system noise from collectives likely does not have an effect due to load imbalance. Since most processors are waiting at a collective, only a few stragglers need to arrive before it can be completed and jitter impacts are minimized. Overall, the data shows that ALE3D is not very sensitive to the network, but would benefit from better load balance.

B. *Group II: Mercury, Ardra, Miranda, and AMG2013*

Group II consists of four applications that were sensitive to system noise, but not affected by bandwidth tapering. In this section we first present their messaging characteristics individually followed by a combined description of their sensitivity to tapering.

Mercury uses Isends to communicate particles that cross into neighboring spatial domains, and collective operations such as AllReduce and Reduce. Two point-to-point message call sites account for over 90% of all messages and 80% of all bytes sent. The larger one with message sizes averaging 16 KB is about 80% of the messages and 78% of the bytes regardless of scale (see Figures 7 and 8). Most of the remaining data is transferred in Reduce and Allreduce operations. The MPI runtime is mostly spent in Test and collective operations. Test takes 75% of the MPI time at 8 nodes and drops to 50% at 256 nodes, with collective operations taking the remaining MPI runtime.

Communication load imbalance is prominent in Mercury with the minimum and maximum MPI times differing by a factor of 3 to 5. The load imbalance is due to the fact that particle generation and interaction differ greatly in different materials. In addition, Mercury needs frequent testing to determine whether all particle transport for a phase of the computation is complete [2]. Although this check is done with Iallreduce, the synchronization point leads to load imbalance.

Ardra has two main communication patterns. One pattern is small-message wavefront sweeps that occur concurrently from all corners (four in 2D and eight in 3D) of the mesh. These sweeps are typically pipelined with one angle at a time

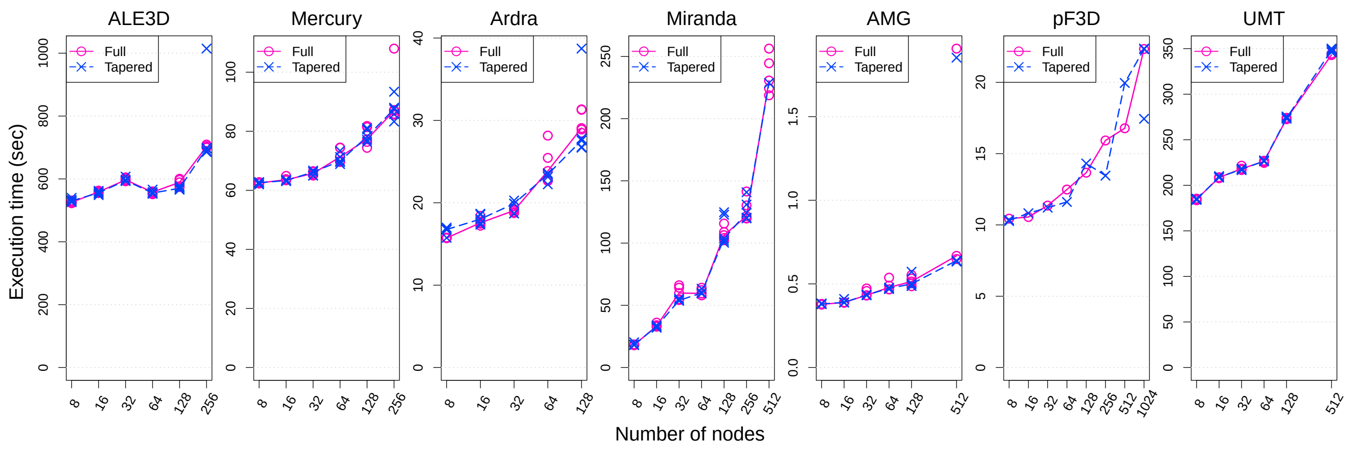

Fig. 9. Application execution time for the full and tapered fat-tree configurations. The Y-axis maximum value is different for each application since they have significantly different runtimes. All applications were run in a weak scaling mode.

traversing each zone. For some problems, Hypre is used to speed up the convergence of the solve. When Hypre is used, messaging occurs in a multigrid solver with similar properties to AMG [3], but it passes fewer bytes than the sweeps.

Figure 7 shows that most of the runtime is spent in Testany at small scale and Allreduce at large scale. This is due to the serial dependence of the sweep causing processors to wait on data to propagate through the mesh to begin their work. Figure 8 shows the message sizes for Ardra. Although the synchronous operations on average are large, they account for less than 3% of the total bytes sent over the network. Isend operations used in the sweep account for 97% or more of the bytes communicated and are always 168 bytes.

Miranda transposes data to perform sparse linear (penta-diagonal) solves and FFTs, which require a lot of MPI communication. Weak scaling of the 2D Noh problem puts a small amount of computation on each processor. Even on eight nodes, the code spends nearly 54% of its time in communication (Figure 7). At this scale, a significant amount of the MPI time is spent in Allgather and Sendrecv. As we scale up, the time spent in MPI increases rapidly to 84% on 64 nodes and 94% on 512 nodes, respectively. All messages in Miranda are smaller than 1.5 KB.

The MPI time increase is attributed, primarily, to the Allgather operation. This could be due to a number of reasons including an inefficient implementation of this operation in the MPI library, load imbalance, which manifests as time spent in Allgather, and OS noise. Since all messages are around 1.5 KB in size, it is unlikely that network congestion is a factor.

AMG2013 has two primary message patterns, halo exchange of local data for sparse matrix-vector multiplication and Allreduce operations to compute inner products. The relative costs of these operations are shown in Figure 7. The Allreduces are always 8 bytes in size and account for just 0.01% of all bytes sent, but can consume over one-fourth of the MPI time. The halo exchanges communicate the rest of the data and message sizes vary with the communication pattern and the level of the multigrid hierarchy. Message sizes shown in Figure 8 are small, but vary throughout the run. The initial input problem level has the largest messages and messages become smaller as AMG moves to coarser levels.

As we weak scale, more multigrid levels are added that require smaller messages on average. This results in the average point-to-point message size shrinking from 1,090 to 884 bytes as we scale from 8 to 1,024 nodes. In addition, more Allreduce operations are needed with point-to-point and collective messages increasing by $O(\log(N))$ where N is the number of processors.

AMG also exhibits load imbalance at its coarsest grid levels because there are not enough degrees of freedom for all processors to participate. This can result in a load imbalance of $2\times$ in the number of messages sent per processor. Despite these imbalances, the amount of time spent in the MPI library only differs by 25%. The smaller differences are due to the coarsest levels being faster than the finest.

Impact of tapered fat-tree: None of the applications in Group II saw a noticeable performance impact from the tapered setup as shown in Figure 9. Their run-to-run variability shows that they are more adversely affected by system noise or jitter at scale. A previous study documented this for Mercury, Ardra, and AMG [11] and Miranda shares many similar characteristics with these applications. It is possible that tapering adversely impacted run time but with jitter causing significant run-to-run variations, we were unable to run enough tests to statistically rule out a small negative impact from tapering.

Ardra, AMG, and Miranda all send mostly small messages and spend a significant amount of time in collectives. These messages are small enough that they are message injection limited and unable to drive the full bandwidth of either network configuration as shown in Section IV. Therefore, we do not expect their performance to be affected significantly from the reduced bandwidth.

Mercury is different from the other three applications with medium sized messages on average. However, this hides significant details. A domain in Mercury batches up particles before sending them to a neighboring domain to increase the average message size. Communication of these large messages is overlapped with computation with the receiving domain only looking at received particles when it has completed work on its current batch. Overall, these messages use about 10% of the available bandwidth assuming the worst case of all off-node traffic. Only at the end of a phase when there are few active particles is it likely that messages will not be overlapped. At this point particles are sent in smaller batches, or even one at a time. Therefore, most of the bandwidth-heavy communication is overlapped and the latency-heavy communication is on the critical path and subject to system noise. The overlap of the big messages and small messages (Isend and Iallreduce) at the end of a phase explain why Mercury is not sensitive to tapering of bandwidth, but is sensitive to system noise.

C. Group III: pF3D

pF3D is the only application in group III, which is characterized by applications that have runtime variability not caused by system noise but are insensitive to bandwidth tapering. As shown in Figure 7, most of the communication time in pF3D is spent on Alltoall operations, which are used to perform 2D FFTs on xy-slabs of the problem. These FFTs occur simultaneously and account for the bulk of the communicated bytes (see Figure 8). The simultaneous FFTs and the large message sizes, of about 40 KB, can cause network congestion and runtime variability. Furthermore, other work has shown that network contention with other applications or pF3D itself is an important source of variability on systems with toroidal interconnects [12], [13]. We suspect that contention for shared links is also important on InfiniBand machines such as Cab. In addition, we do not attribute runtime variability to system noise on account of related work [11].

The halo exchange during light advection sends the second highest number of bytes, but uses significantly less time than the FFT message exchange. Send, Recv, and Barrier operations are used in this phase. pF3D predominantly sends large messages and spends a significant fraction of its runtime in communication, so we would expect some sensitivity to network bandwidth. Surprisingly, however, Figure 9 shows no significant differences between the full and tapered fat-tree setups. The performance differences (if any) are smaller than the run-to-run variability and more tests, including network-aware mapping of tasks, are needed to determine the cause of performance degradation at scale and its network bandwidth sensitivity. Finally, we note that production runs of pF3D scale better because they have more zones per domain in the z-direction, but otherwise these results are representative of production runs.

D. Group IV: UMT2013

Group IV includes only UMT and shows no significant variation in execution time between runs but demonstrates sensitivity to the tapered network. In UMT each task solves a full radiative transfer problem on its portion of the mesh. Boundary exchanges, using non-blocking point-to-point operations, are performed to exchange fluxes across neighboring faces. Synchronization points, which are implemented with Allreduce and Barrier operations, are used to monitor for convergence and calculate a new time step.

The average point-to-point message in UMT is large, greater than 148 KB, and its most frequent Allreduce operations are 64, 240, and 8 bytes in size (110.2 bytes weighted average), as shown in Figure 8. Although this figure only shows data for the large problem size, its message sizes are representative of other configurations since they remain fairly constant across different node counts. Figure 7 shows the amount of time UMT spends in MPI and that it increases from 9% at 8 nodes to 15% at 128 nodes. This increase can be attributed to Allreduce operations, whose percentage of runtime increases from 1.4% to 5.69% and, to a lesser extent, the amount of time waiting (Wait) for point-to-point messages.

UMT, like pF3D, has the potential to use a significant amount of network bandwidth with its large point-to-point messages. When analyzing the impact of the tapered fat-tree on the performance of UMT (see Figure 9), we observed a performance slowdown of 1-2% at larger node counts (≥ 512), but no impact at smaller configurations. Since the slowdown is small, it is hardly visible in the graph. UMT spends 85-90% of its time computing and only a small fraction of its time communicating. Examining the communication time exclusively revealed a 6-7.5% slowdown with the tapered fat-tree. However, since UMT spends only a small fraction of its time sending large messages, the tapered fat-tree has a small effect on its overall execution time but does have a significant impact on communication.

VI. Impact and Lessons Learned

The results of this study allow us to quantitatively document the impact of tapering bandwidth on a set of applications representative of day-to-day workloads at LLNL. These findings have had a significant impact on system procurement and configuration at LLNL. The National Nuclear Security Administration's (NNSA) Commodity Technology Systems (CTS-1) procurement will provide production cycles to Lawrence Livermore, Los Alamos, and Sandia National Laboratories. The presented study provides evidence that representative workloads would not be impacted severely by reducing the network bandwidth from a full fat-tree to a 2:1 tapered fat-tree. Within a fixed procurement budget, reducing the network investment allows increasing the number of nodes, memory, or storage with the additional funds.

The savings from our tapered network configuration are worth 6-7% of the overall machine cost. For a typical 2,000-3,000 node procurement, we can buy approximately 200 additional nodes. At our laboratory these findings influenced two procured systems in this size range.

While this is a point-in-time study on a moderate number of applications, we believe the techniques we used are generally applicable to other centers in determining their networking needs. In particular, by combining measurements of current usage, experiments on key applications, and discussions with users of the applications, we now understand when and how the networks on LLNL clusters are exercised. We can now estimate the impact of the network on application performance without running all possible applications.

One important lesson is that applications sending only small or medium-sized messages are not affected by tapering. These results are consistent with our benchmark data that shows tapering only changes the delivered bandwidth for large messages (Figure 6). UMT, which sends large point-to-point messages, was slightly impacted by the reduced bandwidth. While the amount of time UMT spends in communication increased by 6-7.5%, the overall impact on runtime was only 1-2% because messaging time is small (9-15%). This data suggests that message sizes coupled with benchmarking data are a good proxy for application sensitivity to decreased bandwidth. Only if the application is likely to be affected are runs necessary to determine the overall runtime impact.

A. *Considerations for Future Systems*

As mentioned earlier, the data presented here is a point-in-time snapshot for current workloads on existing machines. Future hardware and application changes will impact our bandwidth needs in various ways. In this section, we briefly mention implications for Ardra and pF3D, and continue with hardware trends for next generation systems.

The Ardra team is developing a GPU version of their code targeted at a CORAL system comprised of IBM Power processors coupled with NVIDIA GPUs. The GPUs are motivating a different design that will increase the message sizes by 10-1000×, depending on choices made for performance tuning. At the lower end, messages would still be very small and the findings in this paper would continue to apply. However, at the higher end, message sizes would be similar to UMT and more benchmarking would be needed.

The pF3D test problem we ran on Cab used 4×16 domains per xy-slab, so a slab was split amongst 4 nodes. The Trinity system at Los Alamos National Laboratory[5] has two 16-core Intel Haswell processors per node. pF3D could be run using 4×8 domains per xy-slab on Trinity so that FFT messages would remain within a node and be passed via fast shared memory. The larger node thus reduces the network bandwidth required for good pF3D performance.

In next generation systems, the total compute performance in flop/s is growing faster than network bandwidth. For the procurement discussed in this study, we were considering a transition from Intel Sandy Bridge processors with QDR at 40 Gb/s to Intel Broadwell processors with a 100 Gb/s network. The total compute performance per node increases by about 4×, total network bandwidth by 2.5×, and memory bandwidth by about 1.5×. Most of the applications we analyzed, which are representative of real workloads, are either memory bandwidth bound or memory latency bound. For these applications, on-node performance is expected to increase slower than off-node bandwidth. Benchmarking experiments showed we should expect about a 2× throughput gain per node as our workloads are not completely memory bandwidth bound. In addition, larger nodes may present opportunities to take advantage of surface to volume gains (more processes and memory per node) to reduce the amount of communication needed per unit of computation.

Processors such as Intel Knights Landing (KNL) and NVIDIA GPUs provide a large memory bandwidth and flop/s increase relative to a Sandy Bridge processor. Fast stacked or in-package memory, when used effectively, will significantly increase the ratio of memory to network performance. KNL with its advertised 400+ GB/s peak bandwidth [14] could provide 5-6× the memory bandwidth and about 10× the compute performance. With fast in-package memory, codes may strong scale to reduce time to solution, resulting in a larger number of smaller messages. Alternatively, they may use fewer MPI ranks and add threading. This will lead to larger messages (as in the case of Ardra) for the same per-node memory footprint. Since network bandwidth is increasing faster than latency is decreasing, the size cutoff for latency dominated messages will increase and may reduce the sensitivity of applications to network bandwidth. If real problems cannot be strong scaled, they will spend significant amounts of time staging between DDR and fast memory (or directly accessing DDR) and our conclusions should hold for our applications.

Whatever the future holds in terms of application and hardware changes, we believe the techniques we have used and our quantitative approach will allow us to optimize the use of our future procurement dollars for application productivity.

VII. Related Literature

Micro-benchmarks, proxy applications, and full applications are often used to compare and contrast the behavior and performance of different supercomputing platforms [15]–[18]. In particular, researchers from Sandia and Los Alamos National Laboratories have studied the performance of DOE applications on commodity machines [19]–[21] to compare the performance of different system generations. We build on these studies by combining machine monitoring with application experiments to quantify cost-benefit tradeoffs for current and future machines.

Vetter and Mueller [22] and Raponi et al. [23] quantify the communication characteristics of a set of scientific applications highlighting common patterns and operations between them. Alam et al. [24] study the balance between CPU, memory, network and I/O requirements for different applications. Pedretti et al. [25] study the sensitivity of application performance to injection and link bandwidth on a Cray XT4 system, while Rosenthal and León [26] focus on the impact of dual-rail networking. Kamil et al. [27] use several scientific applications to understand their communication requirements and present designs for a new network.

Our findings extend this previous work by contributing new observations to the communication requirements of real,

[5]http://www.lanl.gov/projects/trinity

contemporary scientific applications including a surprising tolerance to a tapered fat-tree, which is consistent across different types of applications even those using large messages to communicate. We also corroborate previous results on the importance of collective operations with small payloads among several applications but not all of our codes. In addition, we relate the characteristics of applications to other factors such as run-to-run variability. Finally, we highlight the importance of this study on the procurement of future systems, which had key implications on the configuration and capital cost of LLNL's upcoming commodity clusters to be delivered in 2016.

VIII. Conclusion and Future Work

We present an empirical study characterizing a representative set of applications on a commodity cluster. We focus on the communication aspect of applications to inform the procurement of NNSA's new commodity technology systems. The results of a tapered fat-tree investigation reveal that a tapered network would have minimal to no performance impact on the applications of interest.

This work includes a detailed analysis of production jobs and cluster network monitoring data. This data shows how machine cycles are distributed among different job sizes and that the network is typically under-utilized even when we reduce its bisection bandwidth by turning off one "core" network switch. For a more detailed study, we selected five production applications and two proxy applications that are representative of day-to-day workloads at LLNL. When run at typical job sizes and larger, only one application (UMT) showed any runtime sensitivity to a tapered fat-tree network and that sensitivity was small. Informed by the results of this study we procured a system with a tapered network for next-generation commodity technology systems that will be installed in 2016. In addition, we have shared these results with laboratory partners who are performing their own studies to determine appropriate configurations for their own needs and applications.

While the results presented here have had a significant impact on current procurements, we need to continue monitoring the evolution of systems and applications to provide the most computing capability per dollar to the user community. Future work includes determining how our results map to a 2017 CORAL system comprised of IBM Power processors coupled with NVIDIA GPUs. Significant differences from existing systems include multiple levels of memory, the presence of accelerators, and significantly more memory and compute capability per node. These changes will affect how applications are developed and run. For example, Ardra developers expect message sizes to increase by 10-1,000$\times$ depending on how tuning parameters best map to GPUs. Also, while more memory per node will result in a better surface to volume ratio, more compute relative to network bandwidth can increase the rate of messages leaving a node. Starting with the data analyzed in this paper and working closely with vendor partners, we are assessing the implications of this work and identifying additional experiments needed to prepare for this new system.

Acknowledgment

We would like to thank Andy Anderson, Teresa Bailey, Patrick Brantley, Tom Brunner, Andy Cook, and Matt O'Brien for their guidance in setting up realistic input problems and sizes for the applications studied in this paper. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-CONF-681011.

References

[1] W. Futral, E. Dube, J. Neely, and T. Pierce, "Performance of ALE3D on the ASCI machines," in *Nuclear Explosives Code Development Conference*, ser. NECDEC '98, Oct. 1998, uCRL-JC-132166.

[2] P. S. Brantley, S. A. Dawson, M. S. McKinley, M. J. O'Brien, D. E. Stevens, B. R. Beck, E. D. Jurgenson, C. A. Ebbers, and J. M. Hall, "Recent advances in the Mercury Monte Carlo particle transport code," in *International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering*, ser. M&C'13, Sun Valley, ID, May 2013.

[3] U. Hannebutte and P. Brown, "Ardra: Scalable parallel code system to perform neutron–and radiation–transport calculations," Lawrence Livermore National Laboratory, Tech. Rep. UCRL-TB-132078, 1999.

[4] W. H. Cabot, A. W. Cook, P. L. Miller, D. E. Laney, M. C. Miller, and H. R. Childs, "Large-eddy simulation of Rayleigh-Taylor instability," *Physics of Fluids*, vol. 17, no. 9, 2005.

[5] D. L. Cotrell and A. W. Cook, "Scaling the incompressible Richtmyer-Meshkov instability," *Physics of Fluids*, vol. 19, no. 7, 2007.

[6] V. E. Henson and U. M. Yang, "BoomerAMG: A parallel algebraic multigrid solver and preconditioner," *Applied Numerical Mathematics*, vol. 41, no. 1, Apr. 2002.

[7] J. A. Rathkopf, D. S. Miller, J. M. Owen, L. M. Stuart, M. R. Zika, P. G. Eltgroth, N. K. Madsen, K. McCandless, P. F. Nowak, M. K. Nemanic, N. A. Gentile, N. D. Keen, and T. S. Palmer, "KULL: LLNL's ASCI inertial confinement fusion simulation code," Lawerence Livermore National Laboratory, Tech. Rep. UCRL-JC-137053, Jan. 2000.

[8] V. A. Dobrev, T. V. Kolev, and R. N. Rieben, "High order curvilinear finite elements for elastic-plastic Lagrangian dynamics," *Journal of Computational Physics*, vol. 257, pp. 1062–1080, Jan. 2014.

[9] E. I. Moses, R. N. Boyd, B. A. Remington, C. J. Keane, and R. Al-Ayat, "The National Ignition Facility: Ushering in a new age for high energy density science," *Phys. Plasmas*, vol. 16, no. 041006, pp. 1–13, April 2009.

[10] P. F. Nowak and M. K. Nemanic, "Radiation transport calculations on unstructured grids using a spatially decomposed and threaded algorithm," in *International Conference on Mathematics and Computation, Reactor Physics and Environmental Analysis in Nuclear Applications*, Madrid, Spain, Sep. 1999.

[11] E. A. León, I. Karlin, and A. T. Moody, "System noise revisited: Enabling application scalability and reproducibility with SMT," in *International Parallel and Distributed Processing Symposium*, ser. IPDPS'16. Chicago, IL: IEEE, May 2016.

[12] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs, "There goes the neighborhood: performance degradation due to nearby jobs," in *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC'13. IEEE Computer Society, Nov. 2013.

[13] A. Bhatele, N. Jain, K. E. Isaacs, R. Buch, T. Gamblin, S. H. Langer, and L. V. Kale, "Optimizing the performance of parallel applications on a 5D torus via task mapping," in *International Conference on High Performance Computing*. IEEE, Dec. 2014.

[14] A. Sodani, "Knights Landing (KNL): 2nd generation Intel Xeon Phi processor," in *Hot Chips: A Symposium on High Performance Chips*. Cupertino, CA: IEEE, Aug. 2015.

[15] A. Hoisie, G. Johnson, D. Kerbyson, M. Lang, and S. Pakin, "A performance comparison through benchmarking and modeling of three leading supercomputers: Blue Gene/L, Red Storm, and Purple," in *SC 2006 Conference, Proceedings of the ACM/IEEE*, Nov 2006, pp. 3–3.

[16] L. Oliker, A. Canning, J. Carter, C. Iancu, M. Lijewski, S. Kamil, J. Shalf, H. Shan, E. Strohmaier, S. Ethier, and T. Goodale, "Scientific application performance on candidate petascale platforms," in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, March 2007, pp. 1–12.

[17] A. Bhatele, L. Wesolowski, E. Bohm, E. Solomonik, and L. V. Kalé, "Understanding application performance via micro-benchmarks on three large supercomputers: Intrepid, Ranger and Jaguar," *Int. J. High Perform. Comput. Appl.*, vol. 24, no. 4, pp. 411–427, Nov. 2010. [Online]. Available: http://hpc.sagepub.com/content/24/4/411

[18] M. Cordery, B. Austin, H. Wassermann, C. Daley, N. Wright, S. Hammond, and D. Doerfler, "Analysis of Cray XC30 performance using Trinity-NERSC-8 benchmarks and comparison with Cray XE6 and IBM BG/Q," in *High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation*, ser. Lecture Notes in Computer Science. Springer, 2014, vol. 8551.

[19] M. Rajan, D. Doerfler, C. T. Vaughan, M. Epperson, and J. Ogden, "Application performance on the tri-lab Linux capacity cluster–TLCC," *International Journal of Distributed Systems and Technologies*, vol. 1, no. 2, Apr. 2010.

[20] S. Pakin and M. Lang, "Performance comparison of Luna and Typhoon," Los Alamos National Laboratory, Tech. Rep. LA-UR-12-26426, Nov. 2012.

[21] M. Rajan, D. Doerfler, P. Lin, S. Hammond, R. Barrett, and C. Vaughan, "Unprecedented scalability and performance of the new NNSA tri-lab Linux capacity cluster 2," in *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, ser. PMBS'12, Salt Lake City, UT, Nov. 2012.

[22] J. S. Vetter and F. Mueller, "Communication characteristics of large-scale scientific applications for contemporary cluster architectures," in *International Parallel and Distributed Processing Symposium*, ser. IPDPS'02. Fort Lauderdale, FL: IEEE, Apr. 2002.

[23] P. G. Raponi, F. Petrini, R. Walkup, and F. Checconi, "Characterization of the communication patterns of scientific applications on Blue Gene/P," in *International Workshop on System Management Techniques, Processes, and Services*, ser. SMTPS'11, Anchorage, AK, May 2011.

[24] S. R. Alam and J. S. Vetter, "An analysis of system balance requirements for scientific applications," in *International Conference on Parallel Processing*, ser. ICPP'06. Columbus, OH: IEEE, Aug. 2006.

[25] K. T. Pedretti, C. Vaughan, K. S. Hemmert, and B. Barrett, "Application sensitivity to link and injection bandwidth on a Cray XT4 system," in *Cray User Group Annual Technical Conference*, Helsinki, Finland, May 2008.

[26] E. Rosenthal and E. A. León, "Characterizing application sensitivity to network performance," in *International Conference for High Performance Computing, Networking, Storage and Analysis; Research Poster*, ser. SC'14. New Orleans, LA: IEEE/ACM, Nov. 2014.

[27] S. Kamil, L. Oliker, A. Pinar, and J. Shalf, "Communication requirements and interconnect optimization for high-end scientific applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 2, Feb. 2010.