There Goes the Neighborhood: Performance Degradation due to Nearby Jobs

Abhinav Bhatele[†], Kathryn Mohror[†], Steven H. Langer[†], Katherine E. Isaacs[⋆]

[†]Lawrence Livermore National Laboratory, Livermore, California 94551 USA
[⋆]Department of Computer Science, University of California, Davis, California 95616 USA
[†]{bhatele, kathryn, langer1}@llnl.gov, [⋆]keisaacs@ucdavis.edu

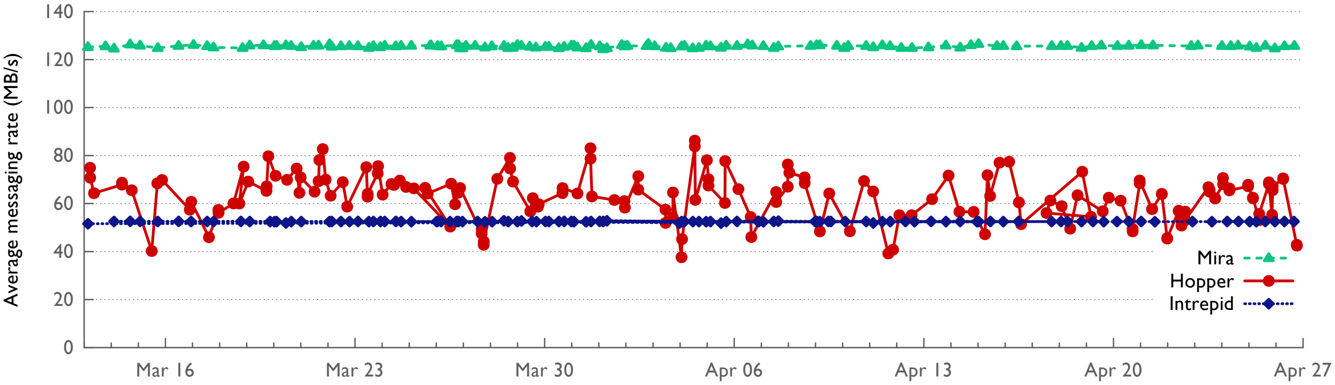

Figure 1: Average messaging rates for batch jobs running a laser-plasma interaction code on three architectures - IBM Blue Gene/Q (Mira), Cray XE6 (Hopper), and IBM Blue Gene/P (Intrepid). The slowest job on Hopper is nearly two times slower than the fastest.

ABSTRACT

Predictable performance is important for understanding and alleviating application performance issues; quantifying the effects of source code, compiler, or system software changes; estimating the time required for batch jobs; and determining the allocation requests for proposals. Our experiments show that on a Cray XE system, the execution time of a communication-heavy parallel application ranges from 28% faster to 41% slower than the average observed performance. Blue Gene systems, on the other hand, demonstrate no noticeable run-to-run variability. In this paper, we focus on Cray machines and investigate potential causes for performance variability such as OS jitter, shape of the allocated partition, and interference from other jobs sharing the same network links. Reducing such variability could improve overall throughput at a computer center and save energy costs.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—Network topology; C.4 [**Performance of Systems**]: Measurement techniques, Performance attributes

General Terms

Management, Performance

Keywords

resource management, torus networks, interference, system noise, communication performance

1. MOTIVATION

Application developers work tirelessly to tune their codes for optimum performance on supercomputing platforms. Even so, an increasing body of evidence shows that external factors can make the performance of an application variable within and across batch jobs [4, 5, 6, 7, 8, 10, 11, 13, 17]. These factors include noise from operating system (OS) daemons, communication variability arising from the shape of the allocated partition, and interference from other jobs sharing the same network links. This performance variability can result in individual jobs running slower, which in turn can lead to a longer wait for the science results, increase in the queue waiting time for other jobs and inefficient use of the time allocation. Resource contention can also lead to

SC '13, November 17-21, 2013, Denver, Colorado, USA
Copyright 2013 ACM 978-1-4503-2378-9/13/11 ...$15.00.
http://dx.doi.org/10.1145/2503210.2503247

	No. of nodes (cores/node)	CPU Type (Clock Speed, GHz)	Peak Tflop/s (Gflop/s/core)	Memory/ core (GB)	Network Topology	Link b/w (GB/s)	Injection b/w (GB/s)
Cielo (LANL, 2011)	8,944 (16)	Opteron 6136 (2.4)	1,374 (9.6)	2.0	3D torus	9.400	20.80
Dawn (LLNL, 2009)	38,864 (04)	PowerPC 450 (0.85)	500 (3.4)	1.0	3D torus	0.425	2.55
Hopper (LBNL, 2010)	6,384 (24)	Opteron 6172 (2.1)	1,280 (8.4)	1.33/2.66	3D torus	9.400	20.80
Intrepid (ANL, 2007)	40,960 (04)	PowerPC 450 (0.85)	557 (3.4)	0.5	3D torus	0.425	2.55
Mira (ANL, 2012)	49,152 (16)	PowerPC A2 (1.6)	10,066 (12.8)	1.0	5D torus	2.000	20.00

Table 1: Architectural details of the parallel machines used for experiments in this study. pF3D simulations discussed in this paper use single precision (floats). The peak single precision performance for Cielo and Hopper cores is twice the double precision number quoted above because the SIMD unit can operate on float operands. The SIMD units on Dawn, Intrepid and Mira have the same peak for float and double. The simulation runs on Mira used one hardware thread per core. The peak performance when operating in that mode is half the number given above.

lower overall throughput on a machine and higher energy costs [3].

Performance variability also impacts the development cycle for high-performance computing (HPC) applications. It can complicate tasks such as:

- debugging performance issues in an application code,
- quantifying the effects of code changes on performance,
- measuring the effects of compiler or system software changes,
- requesting time for a batch job, and
- writing proposals that request time on HPC systems.

HPC application developers spend a significant amount of time on getting their codes to run faster on HPC platforms. A typical development cycle involves measuring and visualizing application performance to identify issues, making code changes based on the performance data, and re-running the application to quantify the improvements. Programmers are often looking to glean performance gains as small as 5-10%. If the performance variability is greater than the expected gains, then statistical analysis of multiple application runs is required to determine the impact of changes made to the code. If the performance variability is significantly higher, a developer may be unable to quantify the improvements.

System software and compilers are frequently upgraded on supercomputers. Application developers typically quantify and document any differences in application performance due to such changes. Once again, run-to-run variability can force developers to resort to running the application several times and using a statistical approach to determine the performance impact of the changes.

Application users need to estimate how long their jobs will run when submitting to a batch system. If the application runs slower than expected, the final results will not be saved before the job runs out of time. If performance is unpredictable, the user must plan for the worst case which means scheduling less work during a batch time slot or overestimating the time required for the job to run. Unfortunately, while ensuring that final results will be saved before allocation termination, overestimating the time for the run can cause the job scheduler to be less efficient.

The problems described above encouraged us to investigate the issues that can lead to run-to-run performance variability on HPC systems. There can be several reasons for unpredictable performance. OS daemons may temporarily force an application off of a core, leading to load imbalance. This is often referred to as OS noise or jitter. On some systems, applications contend for access to the same interconnect links and introduce variability in message passing rates. Placement of processes by the resource manager on an Infiniband fat tree or a toroidal interconnect may vary from job to job. Increase in hop counts (distance traveled by messages on the network) for a "poor" placement can reduce message passing performance. Contention for access to the parallel file system can increase file I/O times. These effects and many others can introduce performance variability large enough to have an impact on application development and tuning tasks and execution.

In this paper, we use pF3D [12], a highly scalable, communication-heavy, parallel application to quantify performance reproducibility on three different parallel architectures: IBM Blue Gene/P, IBM Blue Gene/Q and Cray XE6. We conduct experiments on five machine installations at four U.S. Department of Energy (DOE) laboratories: Los Alamos, Lawrence Livermore, Lawrence Berkeley and Argonne. The experiments were conducted during normal batch processing periods so we could not control the activities of other applications on the system. We observe that the execution time on Cray XE systems can range from 28% faster to 41% slower than the average observed performance (also see Figure 1 showing the message passing rates for pF3D on different platforms over a period of 45 days). Blue Gene systems, on the other hand, have no noticeable run-to-run variability. These systems have negligible OS jitter and a private contiguous torus partition dedicated to each job. In this paper, we focus on Cray machines and perform experiments with the goal of attributing performance variability to OS jitter, irregular shapes of the partition allocated to each batch job, and inter-job interference from sharing the same network links. Reducing such variability could improve overall throughput at a computer center and save energy costs.

2. EXPERIMENTAL SETUP

In this section, we give a brief description of the machines used in this study, introduce the pF3D application, and provide details of our experimental design.

2.1 Machines used

We performed our experiments on an IBM Blue Gene/Q (BG/Q) and two installations each of IBM Blue Gene/P (BG/P) and Cray XE6 systems. Table 1 presents several key characteristics of these machines. There are significant differences in the processors of the IBM and Cray systems.

	Run	No. of	No. of	No. of	Period		Process	Domain	(x,y) FFT	Adv.
Machine	No.	nodes	cores	jobs	From	To	Topology	$n_x \times n_y \times n_z$	msg. (kB)	msg. (kB)
Cielo	CR1	2,048	32,768	46	Mar, 2011	Apr, 2011	$16 \times 16 \times 128$	$640 \times 192 \times 34$	0, 61	2949
	CR2	2,048	32,768	133	Nov, 2011	Apr, 2012	$16 \times 16 \times 128$	$640 \times 192 \times 56$	0, 61	2949
	CR3	4,096	65,536	29	Jun, 2012	Sep, 2012	$16 \times 32 \times 128$	$1280 \times 160 \times 34$	0, 51	4915
Dawn	DR1	24,576	98,304	10	Sep, 2012	Nov, 2012	$16 \times 32 \times 192$	$1280 \times 144 \times 22$	46, 46	4424
	DR2	24,576	98,304	14	Jul, 2012	Aug, 2012	$16 \times 32 \times 192$	$1280 \times 144 \times 22$	46, 46	4424
Hopper		512	8,192	153	Mar, 2013	Apr, 2013	$32 \times 16 \times 16$	$128 \times 128 \times 8$	4, 8	384
Intrepid		512	2,048	102	Mar, 2013	Apr, 2013	$32 \times 16 \times 4$	$128 \times 128 \times 8$	4, 8	384
Mira		512	8,192	116	Mar, 2013	Apr, 2013	$32 \times 16 \times 16$	$128 \times 128 \times 8$	4, 8	384

Table 2: Batch job configurations on different machines. The FFT message size is for messages sent off-node. If it is zero, all messages were passed via shared memory on node. Cielo and Dawn runs have a configuration name to simplify the discussion of the jobs.

The peak Gflop/s numbers in the table are for double precision performance, but pF3D is usually run using single precision (floats). The peak float performance per core is 19.2 Gflop/s for Cielo, 16.8 for Hopper, 6.4 Gflop/s for Mira (when run as in this paper), and 3.4 Gflop/s for Intrepid and Dawn.

The interconnect structure for the systems is similar in the overall topology but significantly different in the details. The BG/P and the XE6 both have a custom three-dimensional (3D) torus interconnect. The BG/P torus has a dedicated network router/switch for each compute node and two X, Y, and Z links per router. On the XE6, two compute nodes are connected to each Gemini router chip through HyperTransport links. There are four X and Z links and two Y links per Gemini router. BG/Q has a custom 5D torus with 10 communication links (A, B, C, D, E) per router/node. The Cray systems have a much higher link bandwidth than the Blue Gene systems. The injection bandwidth on an XE6 (20.8 GB/s) is lower than the total capacity of the 10 outgoing links (94.0 GB/s). This throttling of bandwidth at the source of injection reduces the impact of contention. The injection bandwidth is the metric to consider when assessing best case interconnect performance.

There are significant differences in the node allocation policies for the machines. The resource manager on Blue Gene systems assigns a private torus to each job that is larger than a midplane (512 nodes), while smaller jobs get a mesh in some dimensions. This implies that other jobs running on the system have no impact on the given job. In contrast, the resource manager on Cray XE6 systems assigns an arbitrary set of nodes to each job, spread across the global machine topology. Nodes in use by other jobs and service nodes of various types may cause gaps in the set of nodes allocated to a particular job (See Figure 7). If a job has communicating nodes in two disjoint sections of the torus, it may contend for links with jobs on the intervening nodes. Contention for links and shape variation may both cause variability in network performance.

The hardware on Cielo and Hopper is very similar, but they are configured differently to handle their different missions. Cielo's mission is to run large jobs (typically 64k processes). Cielo is usually rebooted once per week. Hopper's mission is to run a wide variety of jobs with small to medium sizes. Hopper is rarely rebooted. The practical consequence of these differences is that the nodes allocated to a job tend to be significantly more fragmented on Hopper than they are on Cielo. The more fragmented allocations increase the probability that messages sent by one job will pass through interconnect links that are used by another job. This suggests that there will be greater message rate variability and more interference between jobs on Hopper than on Cielo.

2.2 Application description

pF3D [1, 9, 12] is a multi-physics code that simulates the interaction between a laser beam and a plasma for experiments conducted at the National Ignition Facility (NIF). It solves wave equations for the laser light and two kinds of backscattered light. The light waves are coupled together through interactions with electron plasma waves and ion acoustic waves. Figure 2 shows the laser beam from a pF3D simulation.

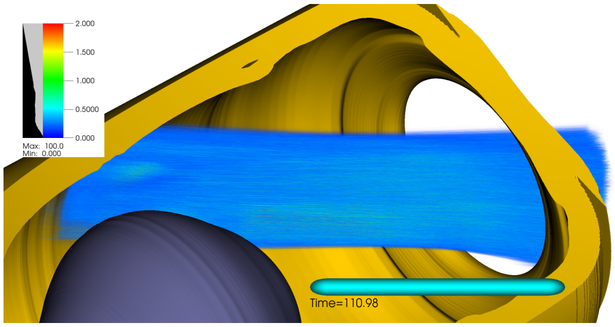

Figure 2: The figure shows a NIF laser beam (blue) propagating through a hohlraum. The hohlraum (gold) and capsule (dark blue) are shown to provide context, but are not part of the pF3D simulation. The capsule (lower left) is imploded by the x-rays generated at the hohlraum wall.

pF3D is a good choice for this sort of a study because it has excellent computational and communication load balance in an ideal scenario. Hence, any imbalance or performance variability is easy to measure and solely attributable to external factors. pF3D places heavy demands on the interconnect and the parallel file system, so inter-job interference in those areas has a measurable impact.

pF3D uses a regular 3D Cartesian grid for domain decomposition into blocks in all three directions. Blocks are assigned x, y, and z coordinates within the decomposition (process topology in Table 2). Each MPI process "owns" one block. Each block is further divided into zones in the x, y, and z directions (Domain in Table 2). The laser beam trav-

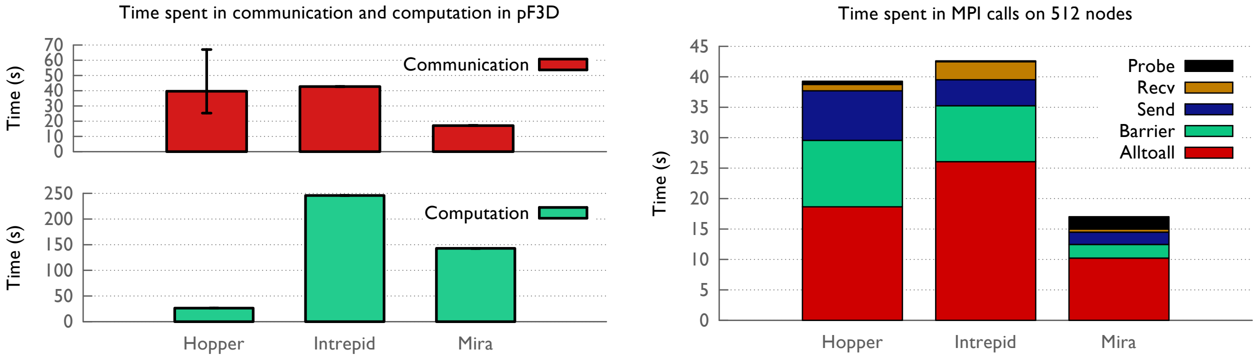

Figure 3: Average, minimum and maximum time spent in computation and communication during one time loop of pF3D on various platforms (left). The computation time is shortest on Hopper and the message passing time is shortest on Mira. Breakdown of communication time by MPI routines (right).

els in the z-direction. The wave equations are solved in the paraxial approximation, which assumes all light waves are traveling at small angles relative to the z-direction.

The wave propagation and coupling are solved using fast Fourier transforms (FFTs) in xy-planes. The 2D FFTs require message passing across the full extent of an xy-plane. These messages are sent using an `MPI_Alltoall`. MPI processes in an xy-slab form sub-communicators with other processes with the same x coordinate (x-communicators) and also with those with the same y coordinate (y-communicators). The all-to-all messages are passed on x-communicators when assembling rows and y-communicators when assembling columns.

The light propagation calculation requires passing planes of information in the z-direction using `MPI_Send` and `MPI_Recv`. pF3D uses a 6th order accurate advection scheme which requires passing three xy-planes between the MPI process domains that are adjacent in z. This communication traffic is referred to as "advection messages". Messages are also passed in several other phases, including hydrodynamic flow and I/O, but the total traffic is low enough that it is ignored in this study.

2.3 Experiment Configuration Details

We show the parameters for the pF3D simulations used in this study in Table 2. Simulations run on Dawn and Intrepid use 4 MPI processes per node. Simulations on Mira, Hopper, and Cielo use 16 MPI processes per node. This choice maximizes similarity in the message passing characteristics of these three systems, but means that 8 cores are left idle on each node of Hopper and only one hardware thread per core is active on Mira. In our tests on Mira, we found that the node throughput for pF3D nearly doubles when using two hardware threads per core but we do not use that setup in this paper. The runs on Dawn and Cielo were production simulations between March, 2011 and September, 2012 and have high node-counts. The data on Hopper, Intrepid and Mira was collected specifically for this study by doing short low node-count runs over a period of 45 days from March 13, 2013 to April 26, 2013.

We performed our detailed studies on Hopper, Mira and Intrepid. For the runs on these machines, we collected information for pF3D as well as for other jobs running on the system concurrently. For pF3D, we gathered the performance timing output from both the application itself and the mpiP performance tool [14]. The pF3D output contains summary information about the run, including the times in different phases of the computation, and the total time per time step. It also gives the average bandwidth over all message passing operations, or message rate, as well as the average message rate for FFT operations and advection messages. pF3D also records the execution times for a few representative all-to-alls for all sub-communicators. The light-weight mpiP tool collects summary information about MPI operations, e.g., total time spent in MPI by each rank and the aggregate time spent in the MPI operations that took the most time. For each job, before and after the process launch, we also recorded the output of the XE6 script `xtnodestat` to discover the node placement of pF3D as well as other jobs on the system. We used the script `xtdb2proc` to get a map of the system, including x, y, and z coordinates of each node, and the location of down and service nodes on the system. We also collected the queue status before and after pF3D ran with the `showq` command.

3. PF3D CHARACTERIZATION

The computational and communication behavior of pF3D on different platforms varies because of the differences in processor and network speeds. Figure 3 (left) shows the average computation and communication time for pF3D simulation runs on Hopper, Intrepid and Mira. Hopper has the fastest absolute computational rate of the three systems. The "efficiency" of a core can be defined as the ratio of the zones processed per second per core to the peak performance per core. If we use peak performance numbers adjusted as indicated in the caption for Table 1 and normalize so that the efficiency for Hopper cores is 1.0, the efficiency for Intrepid cores is roughly 0.48 and the efficiency for Mira cores is roughly 0.52.

Mira has the highest message passing rate (smallest communication time) per process even with links that have less than one-fourth the bandwidth of Cray links. Intrepid and Mira do not show any noticeable job-to-job variations in either computation or communication. However, on Hopper, the communication time time varies from 36% faster to 69% slower when compared to the average. The computation time is fairly constant. Changing the system configuration so that pF3D normally runs closer to the best performance would greatly reduce the time required to run a job. We focus on studying variations in message passing rates in this paper.

Figure 3 (right) shows a breakdown of the communication time by MPI routines. All-to-all messages sent as part of 2D FFTs consume more than half of the message passing time on all three systems. The other significant communication is the sends and receives for the advection messages. FFT message passing is performed for all planes in a domain while advection messages are only sent from the first and last xy-plane in the domain. The all-to-all (FFT) messages transfer 3.6× more bytes than the advection messages for the Hopper test run and 9.9× more bytes for job DR2 on Dawn. Advection messages have a lower messaging rate, but the amount of traffic is small enough that the FFT message time is greater than that for advection in the Hopper runs.

In the rest of the paper, we use the average messaging rate for one time loop of pF3D as the metric for evaluating performance. Figure 4 shows that a high messaging rate leads to low execution time and vice-versa. As seen in Figure 3 (right), most of the messaging time is spent in the FFTs. This is also evident from Figure 5 which also shows that the overall average messaging rate closely follows that for the FFT phase. As we try to make sense of some of the performance variability, we will pay closer attention to the mapping of the x and y FFT communicators on the torus and the corresponding messaging rates.

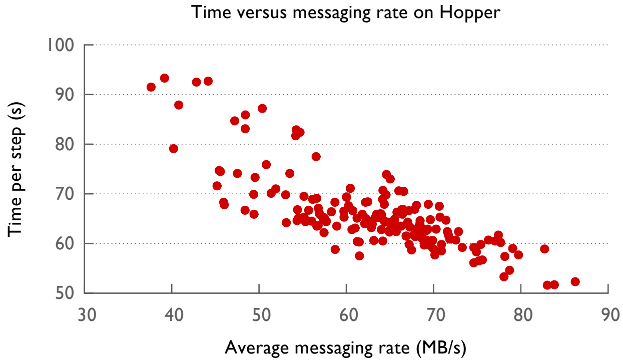

Figure 4: The execution time for one time loop of pF3D is inversely proportional to the average messaging rates.

The rest of the paper is organized as follows: We first discuss the variability of production runs on Cielo in Section 4. We then discuss detailed studies based on the experiments done on Hopper in Section 5. In Section 5.1, we discuss the impact of OS noise/jitter on performance. In Section 5.2, we discuss the effect of the shape of the allocated job partition on the performance variability and in Section 5.3, we analyze the impact of network interference from other jobs on performance.

4. STUDIES ON CIELO

Our first encounter with significant performance variability (up to 3× in message rates!) was during production pF3D runs that simulated specific NIF experiments. Figure 6 shows the average messaging rate for individual batch jobs in the course of production runs over a period of several months on Dawn and Cielo. There is hardly any variation in the rates on Dawn - less than 1% over all batch jobs in a simulation. These batch jobs used 96k processes on 24k nodes. The very low variability is due to all batch jobs running on a partition of exactly the same size and shape.

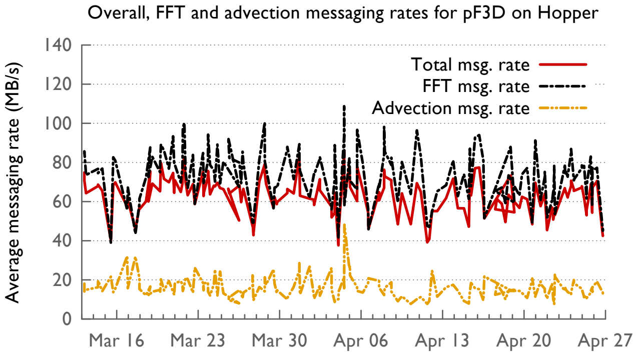

Figure 5: Average messaging rates for one time loop and two sub-phases within a time loop: FFT and light advection

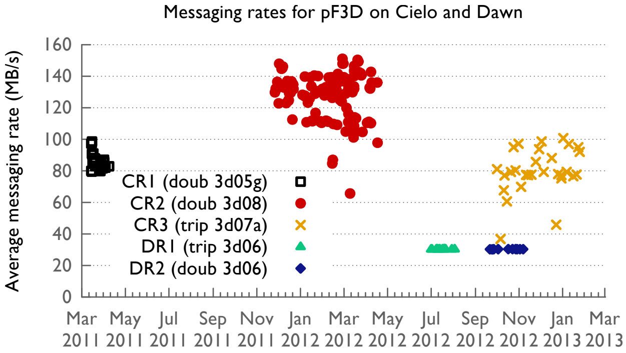

Figure 6: Average messaging rates for different pF3D jobs running over the course of several months on a Cray XE6 (Cielo) and an IBM Blue Gene/P (Dawn).

We noticed a much higher variability in the Cielo runs. Three different input configurations were simulated during the course of these runs. CR1 and CR2 used 32k processes, while CR3 used 64k (See Table 2). The x-communicators had 16 processes and were confined to a single node in these runs. The performance for FFTs in the x-direction was high because messages were passed through shared memory. All communication on the y-communicators (of size 16 or 32) went off-node.

The Cray resource manager allocates blocks of compute nodes along the z-direction of the torus. It allocates contiguous nodes as long as it can, then skips forward when it encounters a node that is in use. The allocation mode controls the extent of an allocation block in the x and y directions of the torus. Mode zero (used on Hopper) allocates 1 × 1 node blocks (single compute nodes). Mode one was used when Cielo first came online and allocates 1 × 2 node blocks. Mode two was adopted on Cielo in September 2011 and allocates 2 × 2 node blocks.

Job CR1 used mode one, so most y-communicators were 1×2×8 nodes. CR2 used mode two so most y-communicators were 2 × 2 × 4 nodes. CR3 also used mode two so most y-communicators were 2 × 2 × 8 nodes (32 processes required longer communicators). The messaging rate for CR2 was higher than for the other two runs, suggesting that a smaller extent, in this case by a factor of two, for the y-communicators in the z-direction of the torus has a strong impact on performance.

The variability in message rates between batch time slots was high for the Cielo simulations and was nearly a factor of three for CR3. pF3D is a bulk synchronous code, so the performance of the slowest communicator controls the performance of the job. Some of the communicators have a larger extent on the torus due to intervening nodes that were in use at job launch. Those "stretched" communicators are slower than typical communicators.

Figure 7 shows the location of I/O, login, and visualization nodes on Cielo. The visualization nodes form a compact brick along one edge of the torus and have little impact on the compactness of nodes allocated to pF3D. Service nodes are scattered throughout the torus and will be located between compute nodes allocated to pF3D in any large run. The "breaks" in pF3D's node allocation due to service nodes are much smaller than the breaks due to other jobs on Hopper (see Section 5).

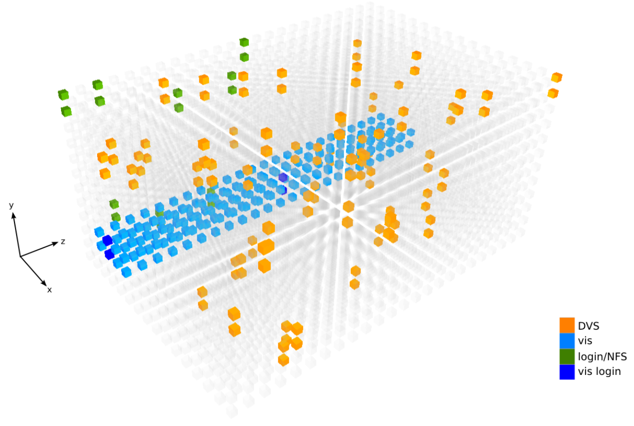

Figure 7: Location of service and visualization nodes on Cielo.

We do not have data on nodes that were in use by other jobs during our Cielo runs. The difference between the average message rates for the runs shown in Figure 6 are somewhat larger than the variability within a run. From this we infer that the shape of communicators has at least as big an impact on message rate variability as contention with other jobs on Cielo.

The rest of the paper describes more extensive tests on Hopper where we gathered performance data for each run to verify some of our hypotheses based on the Cielo runs.

5. STUDIES ON HOPPER

In this section, we investigate several potential sources of the performance variability we observed for pF3D on Hopper. We look at the impact of noise from OS daemons, coordinates of the allocated nodes on the torus, and contention from other jobs running concurrently with pF3D. These simulations use fewer nodes than the production simulations on Cielo, which might somewhat change the relative importance of different sources of variability.

5.1 Impact of OS jitter

OS jitter or noise [13, 11] has been the culprit for variability and the subject of numerous studies in the past. Thus, investigating the impact of jitter was a natural starting place for us to look for the cause of the performance variability within a pF3D job. We used the output from mpiP to calculate the average computation time over all 8,192 processes in each job on Hopper in Figure 8 (top). The error bar shows the minimum and maximum times. The average computation time is nearly constant at ˜26 seconds and the maximum and minimum times vary no more than 6% and 16% respectively. From this plot and also Figure 3 (left), we can conclude that there is little across job variability in computation time.

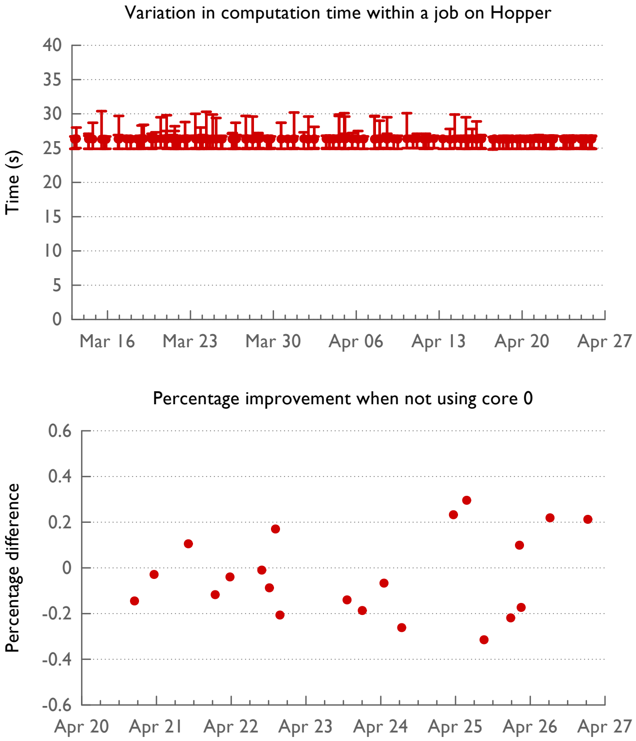

Figure 8: Average, minimum, and maximum computation times within each pF3D job shown for different runs (top). Performance differences when using different cores on the Opteron sockets of Hopper (bottom).

Because we are running on 16 of the 24 cores on each Hopper node, we have flexibility in deciding which physical cores to use. Since some kernel activities, e.g. interrupt handling, are typically handled by core 0, we wanted to make sure our process-to-core mapping didn't cause any unnecessary performance variability. In order to study this effect, we tried two configurations. In the first configuration, we used cores 0-3, 6-9, 12-15 and 18-21; in the second configuration, we used 1-4, 7-10, 13-16 and 19-22. We ran both configurations within the same job allocation and found that the performance difference between them was insignificant, varying at most by approximately 0.3%. (Figure 8 (bottom)).

Previous works have found that even small amounts of perturbation due to OS noise can be greatly amplified by collective communication operations [6, 10]. We don't believe this happens in our pF3D runs on Hopper. Amplification of noise should be small because we are communicating in chunks of 16 or 32 processes, not performing global collectives. We have evidence of this from our runs on Cielo, where interference from other jobs is much less likely than on Hopper. We observed very little variability in the message passing rate within a single Cielo batch job when using all 16 cores, meaning that there was very little OS jitter. Because Cielo and Hopper use the same OS, we have a strong case for concluding that the contribution to overall performance variability from OS jitter is small in our Hopper runs. The much larger overall differences in execution time we observed must be due to some other cause.

5.2 Impact of allocation shape

In this section, we investigate the impact of job shape, or the coordinates of the nodes assigned to a job on the interconnect, on the performance variability of pF3D. We begin by considering the degree of fragmentation of pF3D jobs when scheduled from the normal batch queue. Figure 9 shows the contiguity of node allocations of pF3D. We defined the contiguity metric as a measure of the job compactness. Contiguity is the number of nodes allocated for a job divided by the total number of nodes in the three-dimensional mesh formed by the farthest corners of the nodes in the job. If the contiguity is close to one, the allocation is very compact. If it is close to zero, the nodes are widely spread across the torus. The contiguity for most jobs is between 0.1 and 0.2, which is poor. There is little correlation between the average messaging rate and the contiguity.

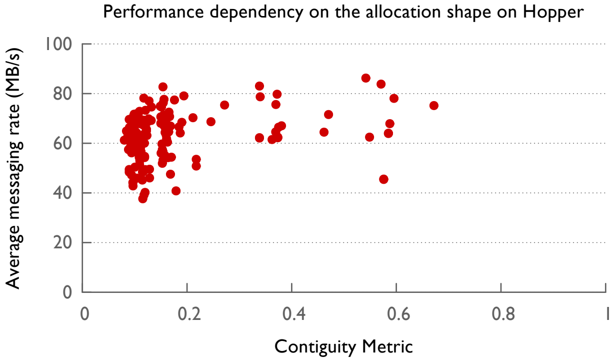

Figure 9: Degree of fragmentation of pF3D jobs when run in the normal batch queue. Contiguity is defined as the number of nodes allocated for a job divided by the total number of nodes in the three-dimensional mesh formed by the farthest corners of the nodes in the job. If the contiguity is close to one, the allocation is very compact.

As mentioned in Section 3, most message passing time in pF3D is spent in all-to-all messages that are part of 2D FFTs. The x-communicators (of size 32) are generally located on a single Gemini node and pass half of their data using shared memory. Messages on y-communicators hop across more links and have greater variability than x-communicators. Thus, we focus on y-communicators.

In the 8,192-process jobs run on Hopper, there are 512 sub-communicators of size 16 in the y-direction of the MPI process grid. The individual messages in an all-to-all are small for the jobs run on Hopper (see Table 2). Message passing rates should correlate with latency for these small messages in the absence of link contention. We calculate the average hops for each y-communicator by averaging the Manhattan distances between every pair of processes in the communicator.

Figure 10 shows the relationship between the FFT messaging rate and the average hops for all y-communicators. The correlation between average hop counts and message passing rates is weak, meaning hop count is not the major factor in the performance variability.

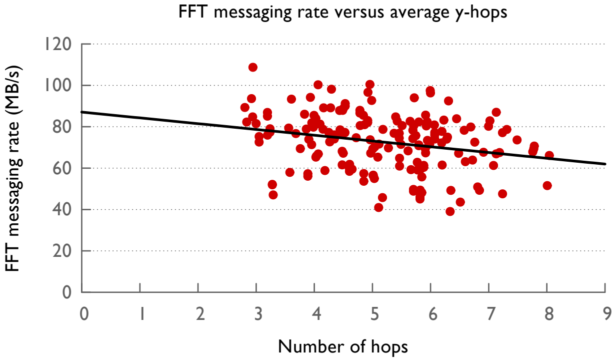

Figure 10: Variation of the FFT messaging rate with the average spread of y-communicators on Hopper.

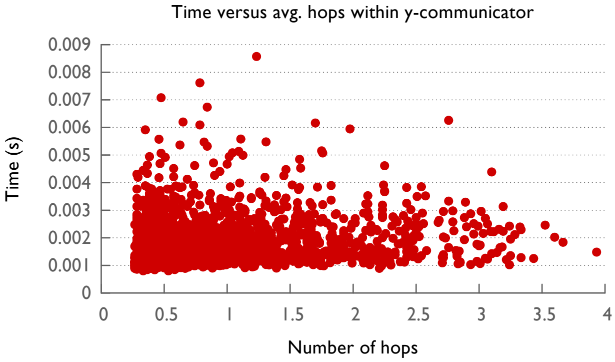

Figure 11: Time spent per y-communicator in ten all-to-alls plotted against the communicator's average hop count. There is a great deal of variability at each path length, indicating that any latency effects are dominated by some other factor.

pF3D records the timings for the first ten all-to-alls performed by the x and y-communicators. In Figure 11, we examine the correlation between the time for individual all-to-alls and the number of hops for their y-communicators.

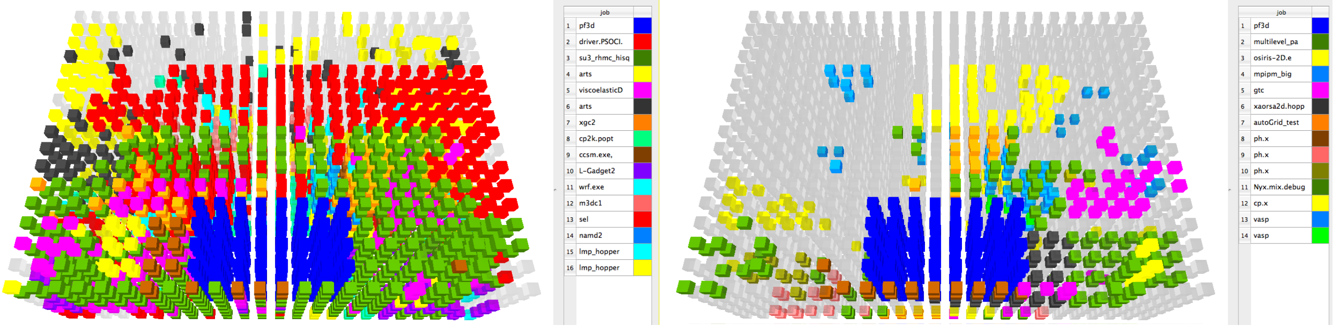

Figure 12: The placement of pF3D (blue) and conflicting jobs on Hopper for two separate short runs. The April 11 job (left) yielded a messaging rate nearly 25% below that of the April 16 job (right). The two jobs had the same node placement, but the slower April 11 job was surrounded by nodes of several other jobs, including a large communication-heavy job (green).

The results from several runs are shown in a single scatter plot. The correlation between all-to-all time and the number of hops is weak (Pearson, $r = 0.23$).

We also looked at the individual y-communicator shapes for a few jobs. We chose the jobs with the best and the worst FFT messaging rates and one that was run in a dedicated contiguous partition. Figure 13 shows the correlation of the hops for the y-communicators and their timings. There is a positive correlation between hop count and message passing time for the worst performing run. The spread in times is small for the contiguous run. Message passing times for the best performing run are all small, even for communicators with large hop counts. Thus, there doesn't seem to be a consistent correlation between hop counts and messaging performance.

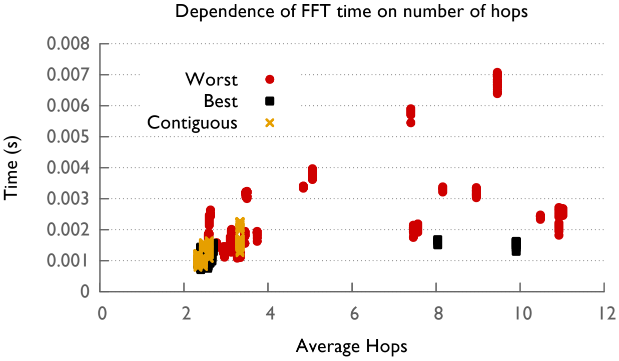

Figure 13: Time spent in each y-communicator plotted against the average hop count for that communicator. There is a positive correlation between hop count and message time for the worst performing run. The message passing time is only weakly correlated with hop count for the best performing run.

In summary, the correlation between hop count and message passing rates is weak in most cases. This strongly suggests that some other cause is responsible for most of the variability in message passing rates on Hopper.

5.3 Contention from other jobs

In the previous two sub-sections, we demonstrated that message passing variability on Hopper is, at best, weakly correlated with OS jitter and the shape of y-communicators. In this section we demonstrate that the character and location of other jobs running along side of pF3D has a strong influence on the messaging rates. We use the term "conflicting" to refer to the simultaneously running jobs which might utilize the same links as pF3D.

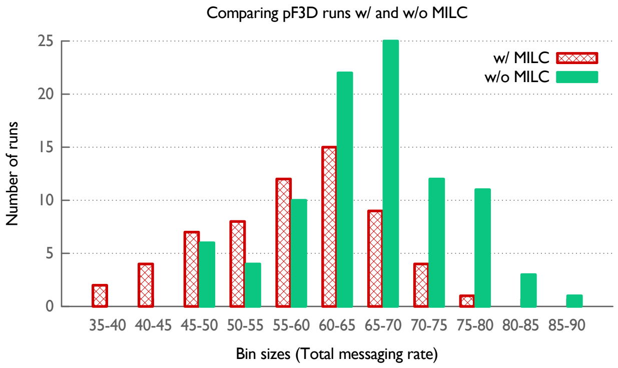

Figure 14: Distributions of messaging rates observed for runs coinciding with a MILC job (red, patterned) and runs with no coinciding MILC job (green, solid). The means of these groups are significantly different (ANOVA, $p = 1.76 \times 10^{-7}$).

MILC [2] is a communication-heavy application present during many of our pF3D runs which may have contended for links, thereby decreasing our observed messaging rates. We divided the pF3D messaging rates into two groups: the pF3D runs which had a conflicting MILC job (w/ MILC) and those that did not (w/o MILC). A histogram of the messaging rates of the two groups is shown in Figure 14. The w/ MILC group had a mean messaging rate of 58.0 MB/s with a standard deviation of 9.12 MB/s. The w/o MILC group had a mean messaging rate of 66.0 MB/s with a standard deviation of 8.69 MB/s. Our tests preserved our assumptions of normality (Shapiro-Wilk, $p_{w/MILC} = 0.145$, $p_{w/oMILC} = 0.536$) and equal variance (Bartlett, $p = 0.662$). We then showed the means of the groups were indeed different using a one-way ANOVA ($p = 1.76 \times 10^{-7}$).

Figures 12, 15, and 16 show the placement of pF3D (blue)

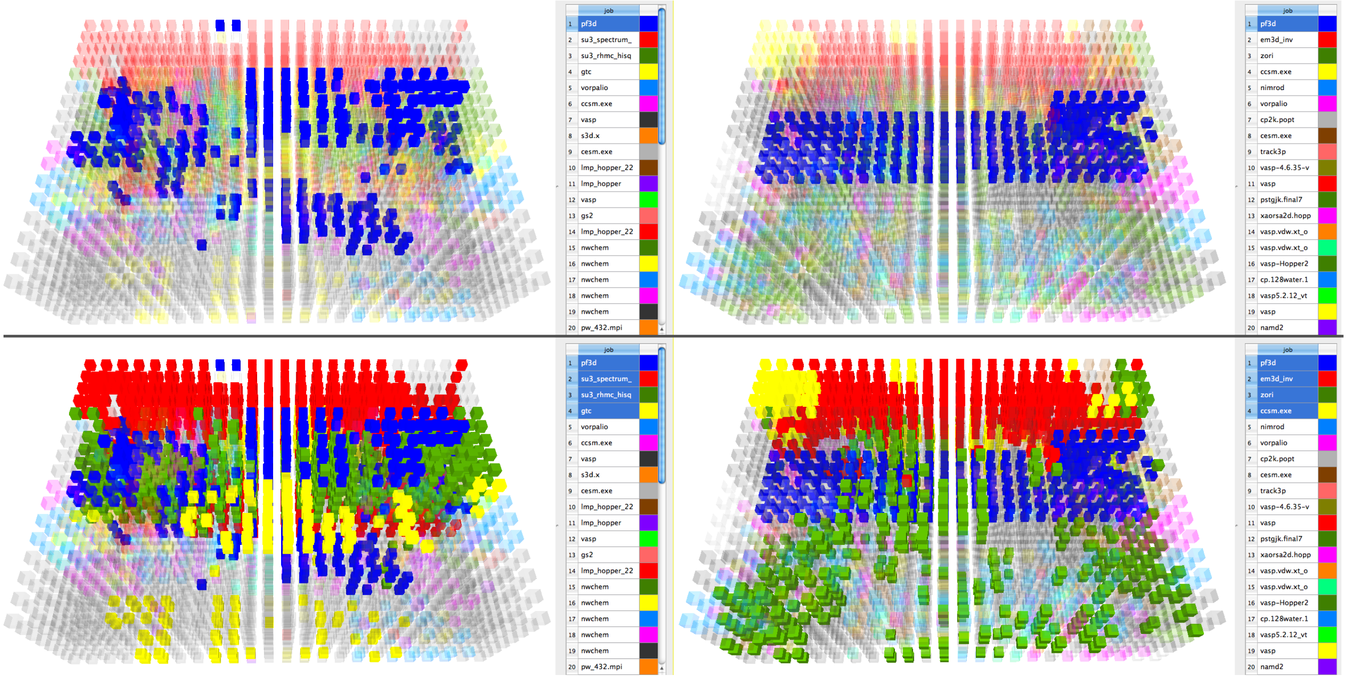

Figure 15: The placement of pF3D (blue) and conflicting jobs on Hopper for a low messaging rate run (left) and a high messaging rate run (right). The top image highlights pF3D and the bottom highlights both pF3D and the largest conflicting jobs. In the slower run, pF3D is interspersed with the green job. The red and yellow jobs have clusters of nodes that would need to utilize the same links as pF3D for communication. In the faster run, pF3D is largely confined to a contiguous slab. The large red and yellow jobs are mostly separated, requiring few links pF3D uses. The green job shows more interference but the majority of the paths among its nodes would not share links with pF3D. The minimal conflict from other jobs and the compactness of the layout led to a message rate 2.29 times that of the slower run.

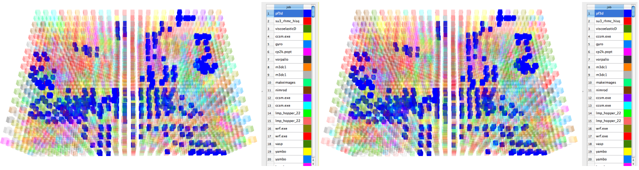

Figure 16: The placement of pF3D (blue) and conflicting jobs on Hopper for two separate short runs. Emphasis has been placed on pF3D. The nodes used in the two runs are 98.7% identical and the conflicting jobs are 92.8% identical. The left job has a messaging rate only 3% below the right job.

and all conflicting jobs. Each box represents a pair of nodes connected by a Gemini router. The light gray boxes are either unused or occupied by jobs which would not contend with pF3D. The largest conflicting jobs are listed in a legend alongside each depiction.

In Figure 12, pF3D occupies the same contiguous $4 \times 8 \times 8$ block in both runs. There were many more nodes vying for pF3D's links in the April 11 job than the April 16 job, including a communication heavy MILC job that "surrounded" the pF3D nodes and was present throughout the run. The April 11 job had a messaging rate almost 25% below that of the April 16 job. The only apparent cause is increased interference from other jobs. Figure 16 shows a case where the pF3D placement and conflicting jobs present are nearly the same and the two jobs reported similar messaging rates.

Figure 15 shows the job placements for the highest and lowest message rates we observed during short runs. The large non-pF3D jobs during the high rate run conflicted only minimally in contrast to the low rate run which suffered from three large jobs surrounding it, two of which were communication heavy MILC runs. This difference resulted in the high rate run reporting a messaging rate 2.29 times that of the low rate run.

Figure 17 shows the variation in messaging rate with the number of Gemini node pairs allocated amongst all the conflicting jobs. We refer to this as "conflicting routers" in the figure. The more routers that might contribute messages that contend with pF3D for the links, the more likely some will have a negative impact on the messaging rate. The observed impact depends on the character of the other jobs. For example, Figure 18 shows the April 11 job compared to another job (April 16b) of the same shape. Both pF3D allocations are surrounded by a large job. The large conflicting job on April 11 was communication heavy. The conflicting job for April 16b was LSMS [16], which spends most of its time in computation and performs I/O at larger intervals than the duration of these pF3D runs. These factors allowed the April 16b run to deliver a messaging rate 27.8% faster than the April 11 run.

Figure 19 shows the number of Gemini routers that were allocated to a pF3D job. This number is significantly higher

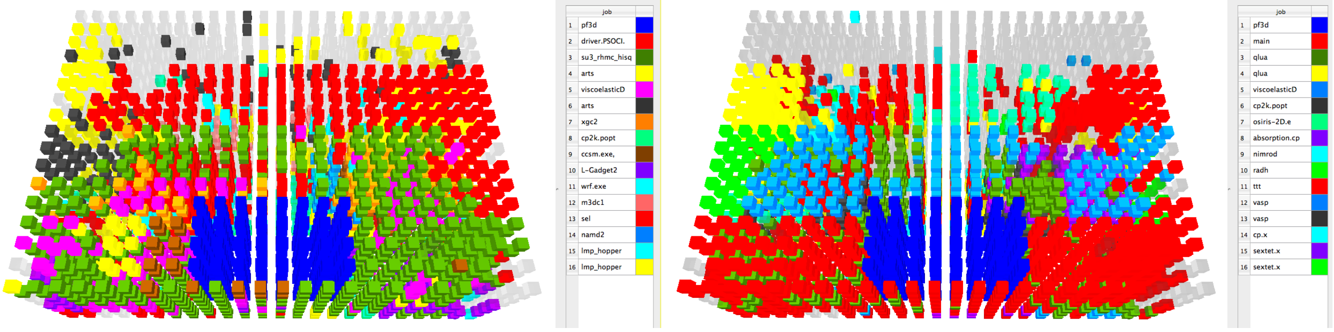

Figure 18: The placement of pF3D (blue) and conflicting jobs on Hopper for two separate short runs. Though both runs were surrounded by nodes from other applications, the right job had a messaging rate 27.8% above the job on the left. The large green job on the left is communication heavy while the large red job on the right is not.

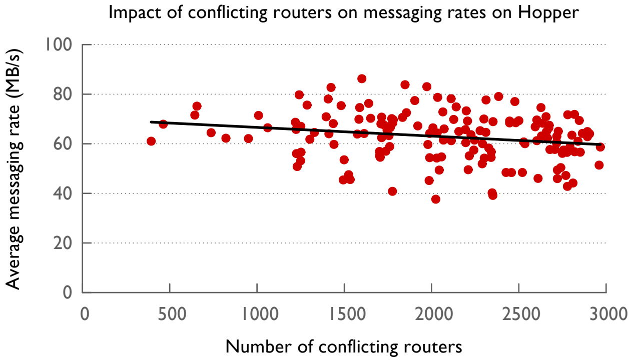

Figure 17: Performance dependence on number of Gemini routers allocated to jobs that might use the same links at pF3D. In general, the more nodes that could contend, the more likely pF3D would report slower messaging. The spread represents the dependence on the dominant communication character of the applications involved.

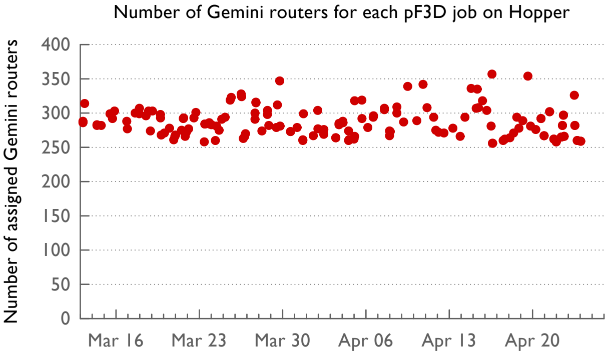

Figure 19: The number of Gemini routers on which one or both nodes are allocated to a pF3D job. For large jobs, the ideal situation would be to allocate both nodes on a Gemini router to the same job. However, if the number of one-node jobs running on the machine is high, then this might not be the best policy.

than 256 for most of the jobs which suggests that the two nodes on tens to hundreds of Gemini routers were shared by pF3D and another job. For large jobs, the ideal situation would be to allocate both nodes on a Gemini router to the same job. However, if the number of one-node jobs running on the machine is high, then this might not be the best policy.

Over longer running jobs, we observed variability in message rates. Figure 20 shows the message passing rate for four such pF3D jobs on Hopper. The message passing rate for the March 31 job increased almost 20% shortly before step 200. The most likely explanation is that another job that shared links with pF3D terminated at that time and the job which replaced it had a lower utilization of the interconnect.

The variability is more complex in the pF3D jobs run on April 10 and April 16. There are short periods during the April 10 run where pF3D achieves higher message rates between steps 100 and 250. This may be related to a large job switching between communication heavy and computation or I/O heavy phases. In contrast, the April 21 pF3D job experienced less variability. The jobs that started and ended during this run required fewer nodes than the April 10 and April 16 runs.

6. RELATED WORK

Several researchers have investigated the effects of OS noise on performance variability [7, 13, 11]. Hoefler et al. used simulation to study the effect of OS noise on communication operations and found that it most strongly affects collective operations [6]. Petrini et al. investigated poor application performance and found that OS noise was preventing the application from scaling as expected [10]. These results contrast with ours in that we found OS jitter to be less detrimental than network contention, given the characteristics of pF3D and the configuration we used for our runs.

Other works have investigated job placement on clusters and its effect on performance variability. Evans et al. looked at variability on Beowulf clusters and noted that tightly-placed jobs had better performance than widely-spaced ones [4]. Kramer et al. noted variability due to job placement and mitigated it by periodically migrating processes to create

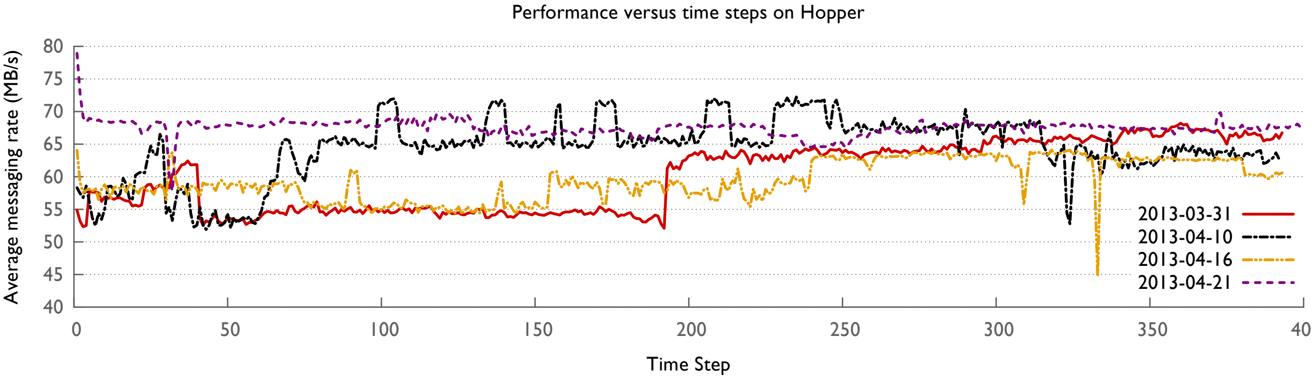

Figure 20: Performance variation within long running pF3D jobs on Hopper.

larger contiguous chunks [8]. Wang et al. looked at the variability of MILC on Hopper finding the correlation of job placement and run time to be weak [15], supporting our findings for pF3D.

Several researchers have focused on the effects of contention for shared resources due to other concurrently running jobs on performance. Hensley et al. noted high variability in performance due to contention for resources as processes accessed non-local memory regions on an SGI Origin 3800 [5]. Skinner et al. noted a 2-3× increase in `MPI_Allreduce` due to network contention from other jobs [11]. Wright et al. found up to 2× variability in performance, nearly always due to network contention [17]. Their results support the findings from our study and conclude that message passing or I/O bound applications suffer the largest interference from other concurrent jobs.

Our work differs from prior research in that we use information about the node placement and communication behavior of other jobs on the system to make conclusions about the relative performance of different runs of pF3D.

7. CONCLUSION

Our goal in this work was to investigate the sources of performance variability in parallel applications running on HPC platforms. We used pF3D, a highly scalable, communication-heavy, parallel application that is well-suited for such a study because of its inherent computational and communication load balance. We performed our experiments on three different parallel architectures: IBM Blue Gene/P (Dawn and Intrepid), IBM Blue Gene/Q (Mira), and Cray XE6 (Cielo and Hopper).

When comparing variability on the different architectures, we found that there is hardly any performance variability on the Blue Gene systems, and that there is a significant variability on the Cray systems. We discovered differences between the XE6 machines due to their node allocation policies and usage models. Since Hopper is designed to serve small to medium sized jobs, the nodes allocated to a job tend to be more fragmented than on Cielo, which mostly serves large jobs. This fragmentation on Hopper resulted in higher variability for pF3D, where the communication time varied from 36% faster to 69% slower when compared to the average. We focused our efforts in this paper on investigating the source of the variability on Hopper.

We investigated the impact of OS noise, shape of the allocated partition, and interference from other jobs on Hopper and concluded that the primary reason for higher variability is contention for shared network resources from other jobs. Our results showed that OS noise has negligible impact on pF3D due to its communication strategy and the configuration of our runs. We found that variations in the way a job is spread across compute nodes only has a weak correlation with performance, suggesting job interference as the dominant reason for the high variability. From queue logs collected during the pF3D runs, we plotted the position of the concurrent jobs relative to pF3D and examined the performance. We found multiple cases where there was strong evidence that the differences in performance are due to communication activities of competing jobs, with message passing rates of pF3D up to 27.8% slower when surrounded by a communication-heavy application.

We plan to extend our investigations on performance variation by developing a light-weight monitoring framework to collect information about job performance and archive it for data mining experiments. Our goal is to collect information from all jobs on a system so that we can reach more general conclusions about sources of variability on HPC systems.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This work was funded by the Laboratory Directed Research and Development Program at LLNL under project tracking codes 13-ERD-055 and 13-FS-002 (LLNL-CONF-635776).

This research used computer time on Livermore Computing's high performance computing resources at Lawrence Livermore National Laboratory. 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. This research also used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

The authors thank Todd Gamblin (LLNL) and Patrick Worley (ORNL) for providing scripts that were used for mining data in the paper. We also thank David Bailey, Leonid

Oliker, Woo-Sun Yang and Zhengji Zhao from LBNL for making the runs on Hopper possible through allocation time, prioritizing the jobs and special reservations.

In the process of writing this paper, we contacted several users of Hopper to obtain details on their jobs and we thank them for responding to our requests – Alexei Bazavov (BNL), Xingyuan Chen (PNNL), Frederico R. Fiuza (LLNL), Glenn Hammond (PNNL), Seung-Hoe Ku (PPPL), Michael Lang (LANL), Marcus Petschiles, Andrew Pochinsky, G. Malcolm Stocks (ORNL), Doug Toussaint (Arizona), Evan Um, Siddhartha Verma (Caltech), Yang Wang (PSC), Michael Wehner (LBNL), E. S. Yoon and Ming Zeng.

8. REFERENCES

[1] R. L. Berger, B. F. Lasinski, A. B. Langdon, T. B. Kaiser, B. B. Afeyan, B. I. Cohen, C. H. Still, and E. A. Williams. Influence of spatial and temporal laser beam smoothing on stimulated brillouin scattering in filamentary laser light. *Phys. Rev. Lett.*, 75(6):1078–1081, Aug 1995.

[2] C. Bernard, T. Burch, T. A. DeGrand, C. DeTar, S. Gottlieb, U. M. Heller, J. E. Hetrick, K. Orginos, B. Sugar, and D. Toussaint. Scaling tests of the improved Kogut-Susskind quark action. *Physical Review D*, (61), 2000.

[3] A. D. Breslow, L. Porter, A. Tiwari, M. Laurenzano, L. Carrington, D. M. Tullsen, and A. E. Snavely. The Case For Colocation of HPC Workloads. *Concurrency and Computation: Practice and Experience Preprint*, 2012.

[4] J. J. Evans, C. S. Hood, and W. D. Gropp. Exploring the Relationship Between Parallel Application Run-Time Variability and Network Performance in Clusters. In *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks*, LCN '03, 2003.

[5] J. Hensley, R. Alter, D. Duffy, M. Fahey, L. Higbie, T. Oppe, W. Ward, M. Bullock, and J. Becklehimer. Minimizing Runtime Performance Variation with Cpusets on the SGI Origin 3800. *ERDC MSRC PET Preprint.*

[6] T. Hoefler, T. Schneider, and A. Lumsdaine. Characterizing the Influence of System Noise on Large-Scale Applications by Simulation. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10)*, Nov. 2010.

[7] T. Jones, S. Dawson, R. Neely, W. Tuel, L. Brenner, J. Fier, R. Blackmore, P. Caffrey, B. Maskell, P. Tomlinson, and M. Roberts. Improving the Scalability of Parallel Jobs by Adding Parallel Awareness to the Operating System. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing (SC'03)*, 2003.

[8] W. T. C. Kramer and C. Ryan. Performance Variability of Highly Parallel Architectures. In *Proceedings of the 2003 international conference on Computational science: PartIII*, ICCS'03, 2003.

[9] S. Langer, B. Still, T. Bremer, D. Hinkel, B. Langdon, and E. A. Williams. Cielo full-system simulations of multi-beam laser-plasma interaction in nif experiments. *CUG 2011 proceedings*, 2011.

[10] F. Petrini, D. J. Kerbyson, and S. Pakin. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing (SC'03)*, 2003.

[11] D. Skinner and W. Kramer. Understanding the Causes of Performance Variability in HPC Workloads. In *Proceedings of the IEEE International Workload Characterization Symposium, 2005*, pages 137–149, 2005.

[12] C. H. Still, R. L. Berger, A. B. Langdon, D. E. Hinkel, L. J. Suter, and E. A. Williams. Filamentation and forward brillouin scatter of entire smoothed and aberrated laser beams. *Physics of Plasmas*, 7(5):2023, 2000.

[13] T. B. Tabe, J. Hardwick, and Q. F. Stout. Statistical Analysis of Communication Time on the IBM SP2. *Computing Science and Statistics*, 27:347–351, 1995.

[14] J. S. Vetter and M. O. McCracken. Statistical Scalability Analysis of Communication Operations in Distributed Applications. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 123–132, 2001.

[15] D. Wang and K. Antypas. "Application Performance Variability On Hopper", 2012. http://www.nersc.gov/users/computational-systems/hopper/performance-and-optimization/application-performance-variability-on-hopper/.

[16] Y. Wang, G. M. Stocks, W. A. Shelton, D. M. C. Nicholson, Z. Szotek, and W. M. Temmerman. Order-N Multiple Scattering Approach to Electronic Structure Calculations. *Physical Review Letters*, 75(15):2867 – 2870, Oct. 1995.

[17] N. Wright, S. Smallen, C. Olschanowsky, J. Hayes, and A. Snavely. Measuring and Understanding Variation in Benchmark Performance. In *DoD High Performance Computing Modernization Program Users Group Conference (HPCMP-UGC), 2009*, pages 438–443, 2009.