Measuring and Understanding Variation in Benchmark Performance

Nicholas J. Wright, Shava Smallen, Catherine Olschanowsky, Jim Hayes
and Allan Snavely
San Diego SuperComputer Center, La Jolla CA 92093
{nwright, ssmallen,cmills,jhayes,allans}@sdsc.edu

Abstract

Runtime irreproducibility complicates application performance evaluation on today's high-performance computers. Performance can vary significantly between seemingly identical runs; this presents a challenge to benchmarking as well as a user who is trying to determine whether the change they made to their code is an actual improvement. In order to gain a better understanding of this phenomenon we measure the runtime variation of two applications, Paratec and WRF, on three different machines. Key associated metrics are also recorded. The data is then used to 1) quantify the magnitude and distribution of the variations and 2) gain an understanding as why the variations occur. Using our lightweight framework, Integrated Performance Monitoring (IPM), to understand the performance characteristics of individual runs, and the Inca framework to automate the procedure measurements were collected over a month's time. The results indicate that performance can vary up to 25% and is almost always due to contention for network resources. We also found that the variation differs between machines and is almost always greater on machines with lower performing networks.

1 Introduction

For a user trying to optimize the performance of their code it is essential that the runtime measurements that they make are accurate and consistent, otherwise they cannot determine if the change that they just made to their code was in fact an improvement or not. For an operator of a High Performance Computing (HPC) resource it is also important that runtime measurements are accurate, as this kind of benchmarking is the most often used method for determining if problems are present on the machine.

Unfortunately, on today's HPC platforms, there are many system components that are shared. This means that user's jobs can contend with each other for system resources, potentially causing slowdowns. Our observations, as well as anecdotal evidence, indicate that this occurs frequently. In this paper we examine exactly how often variations in runtime occur and look at the distributions of runtimes obtained. We also examine the timescale on which they occur to determine, for example, if all the jobs executed using a certain set of nodes are subject to the same performance impediment.

There have been several previous studies examining the effects of performance variation. Tabe *et al* examined performance degradation at scale on an IBM SP2 system and discovered it was caused by OS interference causing variation in the speed of various low level MPI routines [1]. Hensley *et al.* examined performance variation on an SGI Origin 3800, which was traced to contention for memory resources [2]. Perhaps the closest studies to this were Kramer and Ryan

[3] and Skinner and Kramer [4]. They examined the variation in performance for various benchmarks running on several late 1990's machines. In a sense this work is an update of these studies to today's architectures, although we also use profiling methods to give us more insight into the characteristics of the performance measurements made. One recent study related to performance variation is that of Edgar *et al* [5]. They have developed a method that compares the differences between MPI traces of applications in order to identify the causes of performance variation and apply it to the High Performance Linpack (HPL) benchmark. This study is primarily focused upon the effect of different values of the input parameters and different MPI libraries on the performance of an application and as such is quite different in aim from ours which is to quantify variability due to shared-resource contention.
Previous work from our group on this topic investigated similar issues involving Teragrid machines [6]; that work showed substantial variability of distributed Grid applications due to network weather and varying system load on time-shared servers—those results were perhaps not very surprising but perhaps the results of this present work showing similar variation on space-shared tightly-integrated supercomputers will seem more surprising to some.
This paper is organized as follows, Section 2 describes the experimental methods used, Section 3 presents our results and presents the statistical analysis used. Section 4 concludes and contains suggestions for further work.

2 Experimental Methods

Performance variation measurements are taken using a set of benchmarks that are representative of real HPC applications, profiling said benchmarks and using an automatic framework to run them repeatedly over a set time period.

2.1 Benchmarks

We used two application benchmarks PARATEC and WRF. Here we briefly describe each benchmark.

- PARATEC performs *ab initio* quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set [7]. The folded spectrum method described by Canning, *et al*. [5b] is used and entails conjugate gradient iteration to minimize a function whose evaluation requires 3D FFTs. PARATEC is written in Fortran 90 and uses several standard libraries. The benchmark problem considered here is the NERSC large problem which performs 10 conjugate gradient iterations for a system consisting of 686 silicon atoms in the diamond structure.
- WRF (Weather Research and Forecast) is a weather modeling system developed at NCAR [8]. WRF provides a limited-area model of the atmosphere for mesoscale research and operational numerical weather prediction. The WRF model represents the atmosphere as a number of variables of state discretized over regular Cartesian grids. The model solution is computed using an explicit high-order Runge-Kutta time-split integration scheme in the two horizontal dimensions with an implicit solver in the vertical. Since WRF domains are decomposed over processors in the two horizontal dimensions only, interprocessor communication is between-neighbor on most supercomputer topologies. In this work we use Version 3.0.1.1 of WRF and the 12 km

CONUS benchmark. This models the Continental U.S. with 12 km resolution and runs for 3 simulated hours.

We also used three low-level benchmarks, pingpong, naturally ordered ring, and random ring, which are part of the HPCC benchmark suite [9]. These all measure interconnect bandwidth, Pingpong is based on sending a message between two MPI tasks, natural ring sets up an ordered ring in MPI task space and each task sends a message to its left and right neighbors, random ring does the same thing but randomizes who each task sends to. These provide a hierarchy of contention effects, from low to high, and allow us to get a measure of the performance variation characteristics of low level probes. In this work each of the runs of the three low-level benchmarks mentioned above is run immediately following the run of the benchmark application mentioned above.

In this work all of the benchmarks were run with 256 MPI tasks, using all the available CPU cores within the node.

2.2 IPM – Integrated Performance Monitoring

In order to obtain more information than just the runtime about the performance characteristics of the benchmarks each run was instrumented with the Integrated Performance Monitoring (IPM) profiling infrastructure. IPM is a portable profiling infrastructure for parallel codes initially developed at Lawrence Berkeley National Laboratory [10]. It combines information from multiple sources -- hardware counters, the profiling interface to MPI (P-MPI) and the operating system -- into a single job level profile that spans all tasks to provide a performance summary of the computation and communication in a parallel program. IPM has extremely low overhead and is scalable, which means that it is ideal for this work, as it gives us a method of measuring performance in production and at scale without significantly perturbing the application's performance. IPM provides a breakdown of the run time in terms of time spent computing and time spent in MPI, with the latter in terms of the individual MPI calls made. In this work we primarily use IPM to provide us with this breakdown to give us further insights into *why* any performance variations are occurring.

2.3 High Performance Computing Systems Used

Three Teragrid High Performance Computing systems were used in this work.

	Location	Integrator	Peak TFlops	Processor Type	Interconnect	Node
Abe	NCSA	Dell	89.5	Intel Clovertown 2.33GHz	Infiniband	2 socket 8/16 GB
Kraken	NICS	Cray	600	AMD Barcelona 2.3 GHz	Cray Seastar2	2 socket 8 GB
Ranger	TACC	Sun	579	AMD Barcelona 2.3 GHz	Infiniband	4 socket 32 GB

2.4 Inca – Automation of the testing procedure

In order to automate the benchmark measurements of TeraGrid systems we leveraged a user-level monitoring tool called Inca, already in use by TeraGrid. Inca is a monitoring tool designed to detect infrastructure problems by executing automated, user-level testing of software and services [11]. Inca has been monitoring the TeraGrid since 2003 and is also used in other large-scale global grid projects including ARCS [12], DEISA [13], and NGS [14]. Inca is currently deployed to detect user-level failures and can assist system administrators and users to debug and resolve user account and environment issues.

To gather the instrumented benchmark measurements from the TeraGrid machines using Inca, a new deployment of Inca was installed on a server at SDSC called sapa.sdsc.edu, as shown in Figure1, and Inca clients were configured to run on NCSA's Abe machine, NICS' Kraken machine, and TACC's Ranger machine. A small Inca IPM "reporter" script was written to handle IPM environment setup, application invocation, clean up, and ftp of IPM XML data files to a disk at SDSC. This Inca IPM reporter was coupled with a special "batch reporter" that submitted the Inca IPM reporter as a batch job to a machine's particular batch queue system (e.g., SGE, PBS). To vary the machine measurements times, we configured Inca to execute each application twice each day; once during work hours at 6am and once during night hours at 6pm. To randomize measurement times the batch reporter waited a random period of time after being invoked (between 0-12 hours) to submit the Inca IPM reporter to the batch queue. The number of measurements collected during March 2009 for each application varied due to machine downtimes or system errors as shown in Table 1.

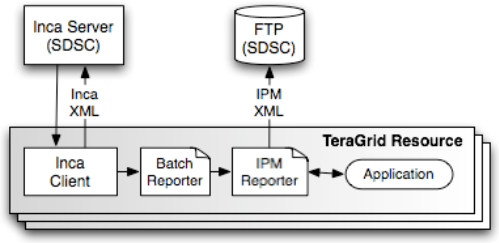

Figure 1. Schematic representation of the Inca deployment used to collect IPM data from TeraGrid resources.

	NCSA Abe	NICS Kraken	TACC Ranger
PARATEC	38	22	36
WRF	42	37	51

Table 1: Number of measurements collected for each application on each TeraGrid machine.

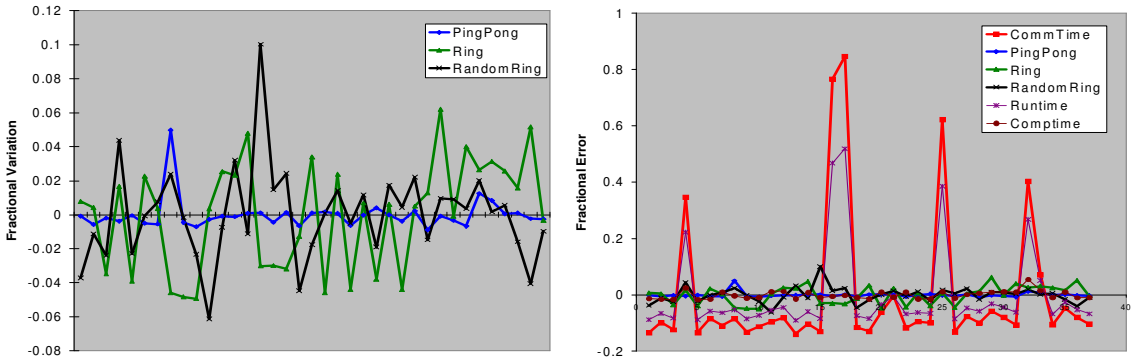

Figure 2. a) Fractional Variation of the measured PingPong, Ring and RandomRing Benchmarks measured on Kraken. b) Same as a) with the addition of the runtime, communication time and compute time as measured by IPM. Note that there is no correlation between the relative speed of one measurement and the speed of the measurement immediately following it.

3 Results

3.1 Granularity of Performance Variation

Figure 2 shows the variation of the performance measurements made for (2a) the low-level benchmarks and (2b) the application and the low-level benchmarks. Each point on the x-axis represents a series of measurements made during the same run, one after the other. Figure 2 shows that a vast majority of the time measurements that are made sequentially are not correlated; they are not all faster or slower than average by the same amount. In fact there are numerous cases where a 'fast' measurement is made in the same run as a 'slow' one. This implies that the granularity of performance variation on these systems is significantly smaller than the time it takes to run even a short the low-level benchmark, which is of the order of one minute. We note also that although Figure 2 only shows data for one machine the results for the other two machines are essentially identical, that is that the granularity of the variation is much smaller than the runtime of the low-level benchmarks for all of the machines. This result also indicates that the observed performance variations are *not* the result of bad hardware, because if they were all the measurements made during the same run would be detrimentally affected.

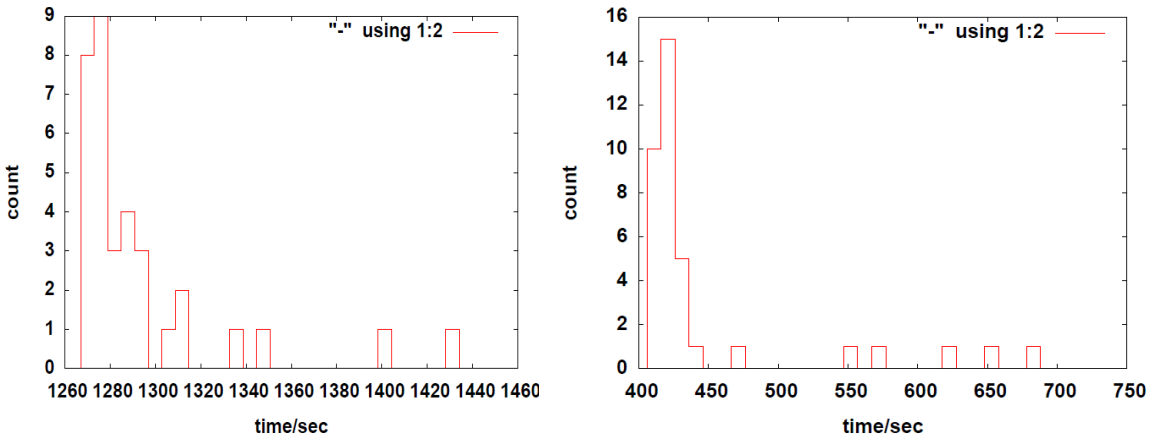

Figure 3. Histogram showing the distribution of runtimes observed a) Paratec and b) WRF on Kraken.

3.2 Performance Variability of Paratec and WRF

Figure 3 shows a histogram of the spread of runtimes observed for Paratec and WRF running on Kraken. (The histograms for Abe and Ranger are qualitatively similar.) The runtime distribution, upon visual inspection, appears to follow essentially a lognormal distribution, which is reasonable as the effects of contention upon the runtime will be cumulative. The range of runtimes observed for Paratec covers approximately a 17% range, whereas for WRF the slowest run was 66% longer than the shortest. This is because WRF has significantly larger I/O needs than Paratec and the few jobs that were delayed by a significant amount were caused by I/O issues.

The normalized cumulative distribution functions for Paratec and WRF are shown in Figures 4, 5 and 6. For kraken the .75 quartile corresponds to approximately 2.5% above the fastest runtime for Paratec, whereas for WRF the 0.75 quartile corresponds to 6%. For Abe the numbers are 2% and 5% and for Ranger they are 10% and 9%. Thus Ranger shows the largest variation in runtimes, and WRF shows more variability than Paratec, mostly because, as previously mentioned, the greater amount of I/O it performs.

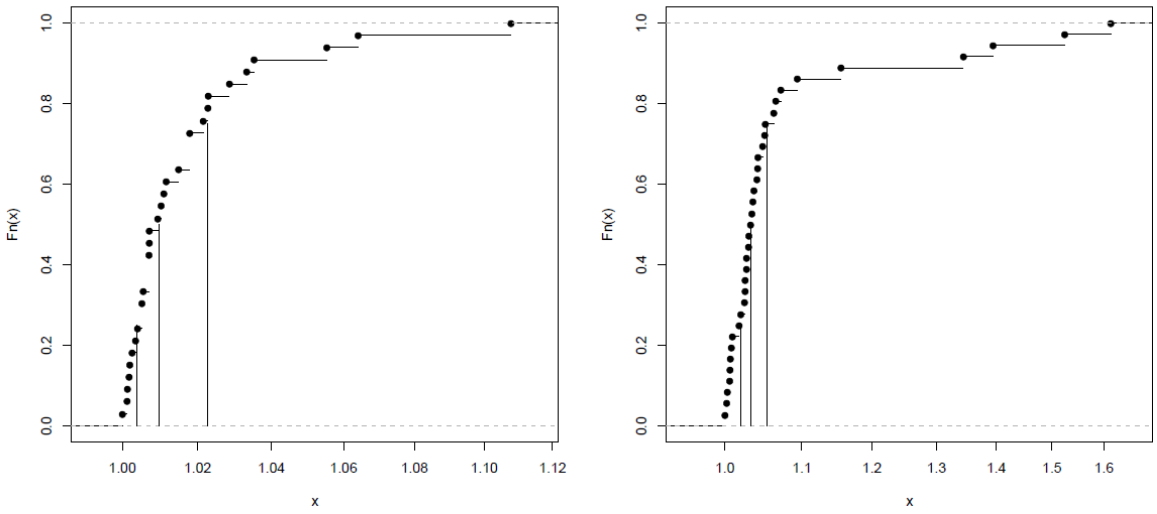

Figure 3. Cumulative frequency distribution of runtime observed a) Paratec and b) WRF on Kraken. Runtimes are normalized to 1270s for Paratec and 410 s for WRF. Vertical lines represent the 25, 50 and 75% of the cumulative probability.

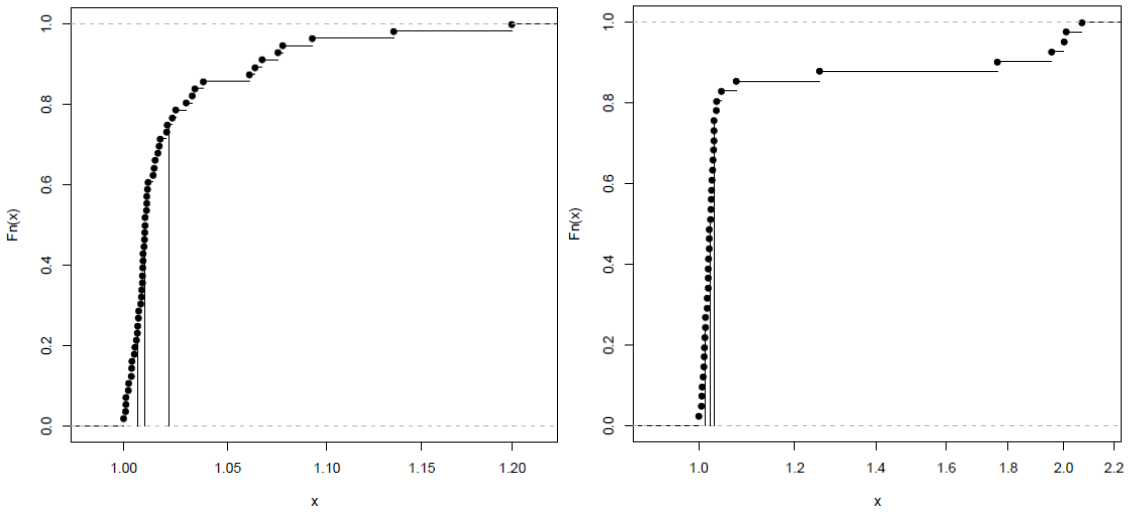

Figure 4. Cumulative frequency distribution of runtime observed a) Paratec and b) WRF on Abe. Runtimes are normalized to 1670s for Paratec and 586 s for WRF. Vertical lines represent the 25, 50 and 75% of the cumulative probability.

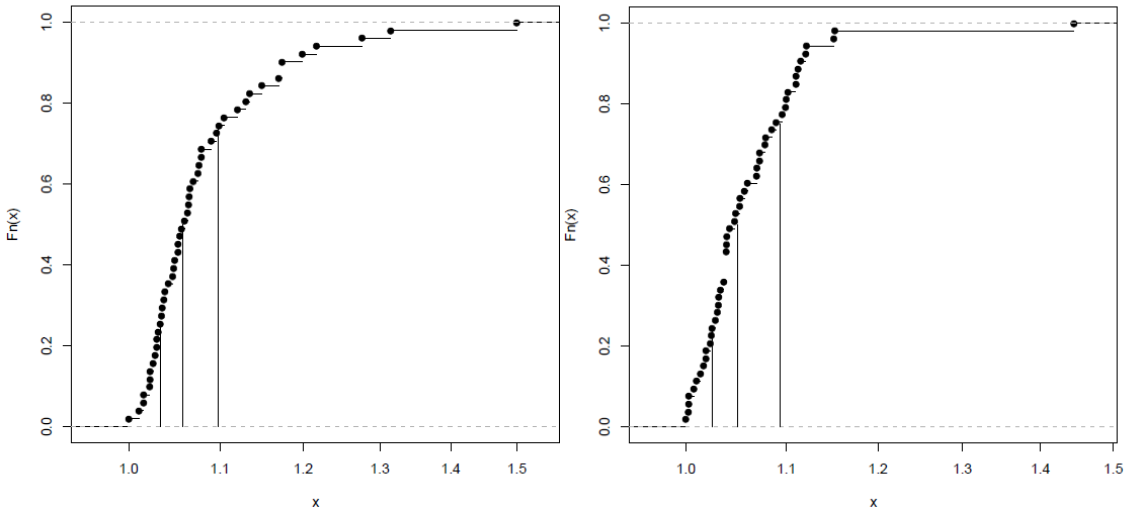

Figure 5. Cumulative frequency distribution of runtime observed a) Paratec and b) WRF on Ranger. Runtimes are normalized to 1286s for Paratec and 596 s for WRF. Vertical lines represent the 25, 50 and 75% of the cumulative probability.

Using our results from IPM we can further breakdown the runtime into time spent in computing and time spent in MPI. When we examined the histograms of computing times they showed no structure at all, indicating that all the performance variation is coming from the time the applications are spending in MPI.

In an attempt to understand the nature of what is causing the performance variations we analyzed the statistical distributions of the communication time measurements and the results are shown in Figures 6, 7 and 8. These are quantile-quantile plots and compare the observed distributions with those that would be expected if the statistics were log-normal. If the statistics are log-normal then the data will fall on a straightline on one of these quantile-quantile plots. For all of the machines the distribution does show approximate lognormal statistics for up to

roughly the .75 quartile. After that the behavior changes to that of a power law, and a very long tail is observed. This tail is longest on Kraken, which is the machine which shows the least performance variation, an observation for which we currently have no explanation.

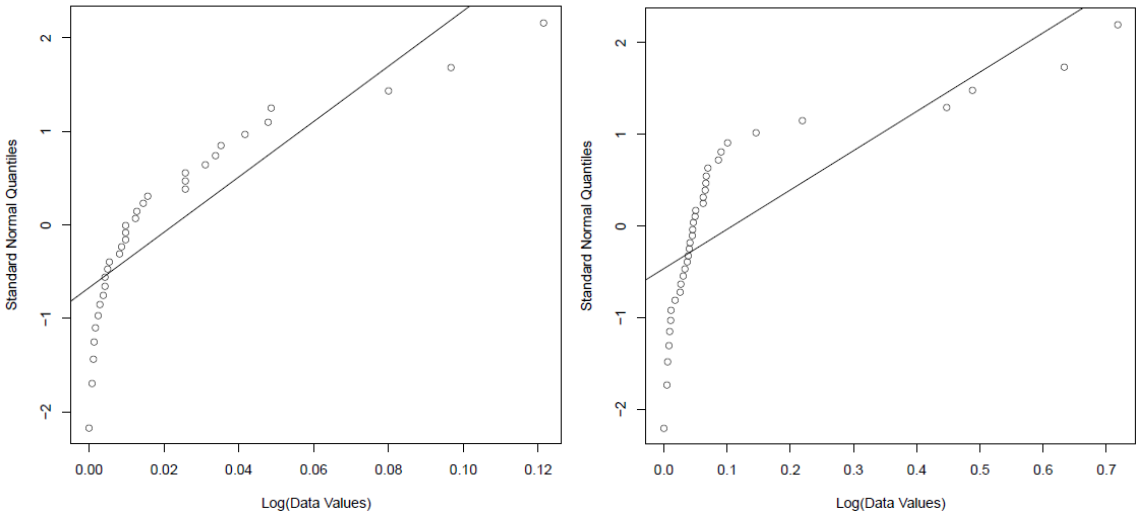

Figure 6. Quantile-Quantile plots of the communication time when compared to a log-normal distribution observed a) Paratec and b) WRF on Kraken.

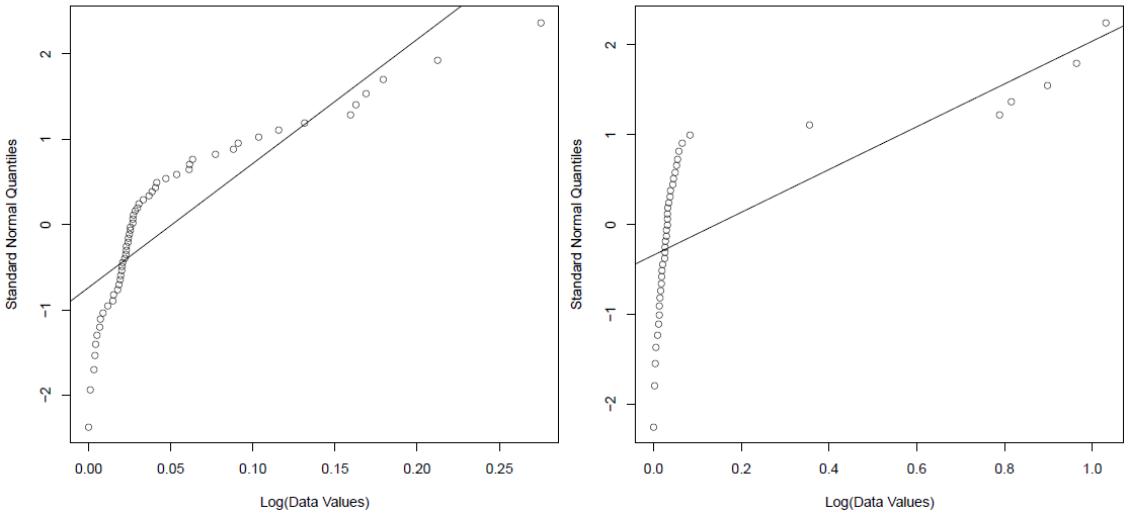

Figure 7. Quantile-Quantile plots of the communication time when compared to a log-normal distribution observed a) Paratec and b) WRF on Abe.

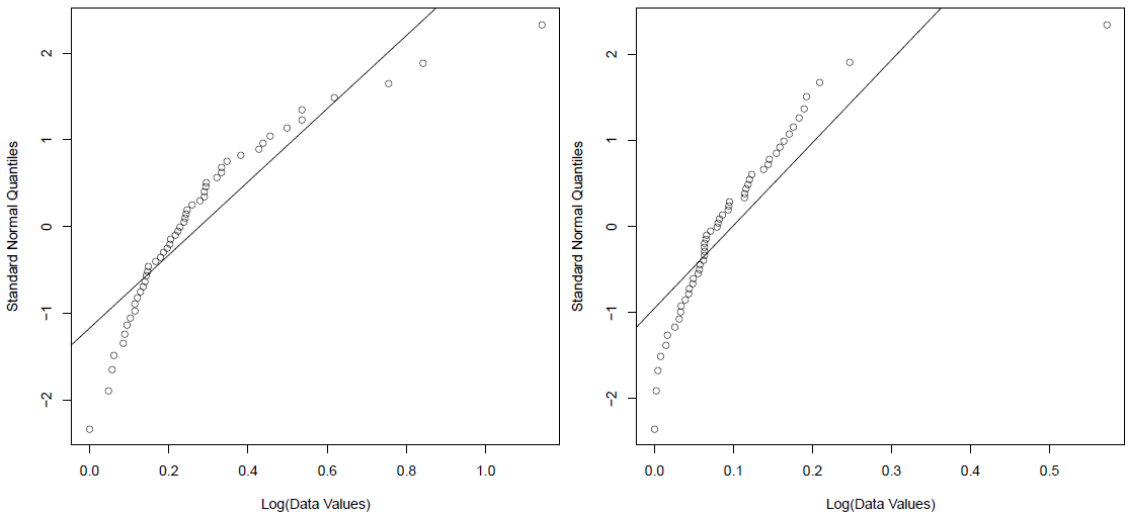

Figure 8. Quantile-Quantile plots of the communication time when compared to a log-normal distribution observed a) Paratec and b) WRF on Ranger.

4 Conclusions and Further work

Performance variability is a challenging issue for anyone trying to understand performance issues on High Performance Computing resources today. By automating the process of running applications and low level benchmarks with Inca, and instrumenting the runs with IPM, we were able to gain insight into the magnitude and the distributions of runtime variations on three Teragrid machines, Abe, Ranger and Kraken. Our results show that for these two benchmarks, Paratec and WRF, running on 256 MPI tasks, Ranger shows the most performance variation, with approximately 25% of the runs taking longer than 10% above the fastest measured result. By analyzing the breakdown of the runtimes into computation and communication components provided by IPM we are able to determine that the runtime variation is all due to variation in the MPI performance, presumably because of contention on the network. Our statistical analysis shows that the communication parts of the runtime have a lognormal distribution at low levels of contention with a power law taking over at higher levels, leading to some observations of performance degradation of up to 2x.

Based upon these results it seems that in order to obtain an accurate measurement of a benchmark the job should be run at least three times, with further runs if the results show significant spread.

In future work we intend to repeat our analysis at different CPU counts, to see how that affects the variations observed. We also want to try and understand the cause of the long power-law tail observed. This is responsible for roughly 20% of a users jobs being slowed down by more than 30%; which is unacceptable for a production computing environment in which one is trying to optimize ones performance. One culprit, in the case of WRF, is I/O contention, a well known issue on HPC machines today. This does not explain all of the WRF results though, or what is causing this for Paratec. One suspicion is that network topology effects are to blame, and it is suboptimal mapping of tasks to nodes that is causing these effects. Hopefully further investigation, probably in collaboration with the center operators, will allow us to understand this issue in more detail and develop potential resolutions.

Acknowledgements
This work was support by the National Science Foundation under grants OCI-0721397 and OCI-0721364.

References

[1] *Statistical Analysis of Communication Time on the IBM SP2.* T, B. Tabe, J. P. Hardwick, Q. F. Stout1. Computing Science and Statistics 27 (1995), pp. 347-351
[2] *Minimizing Runtime Performance Variation with Cpusets on the SGI Origin 3800.* J. H.ensley, R. Alter, D. Duffy, M. Fahey, L. Higbie, T. Oppe, W. Ward, M. Bullock and J. Becklehimer. ERDC MSRC PET Preprint No. 01-32 (2001).
[3] *Performance Variability of Highly Parallel Architectures.* W.C. Kramer and C. Ryan. Lecture Notes in Computer Science. v 2659 p698 (2003).
[4] *Understanding the Causes of Performance Variability in HPC Workloads*. D. Skinner and W. Kramer, , 2005 IEEE International Symposium on Workload Characterization (IISWC-2005), October 6-8, 2005, Austin, Texas.
[5] *A Framework for Comparative Performance Analysis of MPI Applications*. E. Gabriel, F. Sheng, R. Keller, M. M. Resch. International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2006), Las Vegas, June 26-29, 2006.
[6] *Measuring the Performance and Reliability of Production Computational Grids.* O. Khalili, J. He, C. Olschanowsky, A. Snavely, and H. Casanova. Proceedings of the 7th IEEE/ACM International Conference on Grid Computing 2006.
[7] a) PARAllel Total Energy Code (PARATEC), http://www.nersc.gov/projects/paratec. b) A. Canning, L.W. Wang, A. Williamson, and A. Zunger, "Parallel Empirical Pseudopotential Electronic Structure Calculations for Million Atom Systems," *J. Computational Physics,* v. 160, 24-41 (2000).
[8] WRF Model Users Site, http://www.mmm.ucar.edu/wrf/users
[9] HPC Challenge Benchmark, http://icl.cs.utk.edu/hpcc/index.html
[10] IPM: Integrated Performance Monitoring, http://ipm-hpc.sourceforge.net
[11] *User-level grid monitoring with Inca 2*. S. Smallen, K. Ericson, J. Hayes, and C. Olschanowsky. In Proceedings of the 2007 Workshop on Grid Monitoring. 2007. (Monterey, California, USA, June 25 - 25, 2007). GMW '07. ACM, New York, NY, 29-38.
[12] The Australian Research Collaboration Service Web Page, http://www.arcs.org.au/.
[13] The Distributed European Infrastructure for Supercomputing Applications Web Page, http://www.deisa.eu.
[14] The National Grid Service Web Page, http://www.grid-support.ac.uk/.