Application Sensitivity to Link and Injection Bandwidth on a Cray XT4 System

Kevin T. Pedretti, Courtenay Vaughan, K. Scott Hemmert, and Brian Barrett
Sandia National Laboratories [*]
P.O. Box 5800
Albuquerque, NM 87185
{ktpedre,ctvaugh,kshemme,bwbarre}@sandia.gov

Presented at the Cray User Group Conference
Helsinki, Finland
May 2008

Abstract

This paper describes our efforts to characterize application sensitivity to link and injection bandwidth on a Cray XT4 system. Link bandwidth is controlled by modifying the number of rails activated per network link. Injection bandwidth is controlled by modifying the speed of the HyperTransport connection between the Opteron and the SeaStar. A suite of micro-benchmarks and applications is evaluated at several different operating points. The experimental techniques developed by this work are expected to be useful for future architecture research.

1 Introduction

The ratio of a compute node's network I/O performance to its compute performance is an important metric used when designing and comparing supercomputers. Historically, the rule-of-thumb has been that a balanced system should provide 1 MB/s of network bandwidth for each MFLOPS of compute performance. This is expected to be unachievable on future exa-scale systems due to power, packaging, and economic constraints. Even with silicon photonics, initial estimates are that a balanced exa-scale system would require 100-500 mega-watts of power for the interconnect alone [4]. Given the unlikeliness of this, it is important to begin characterizing how reduced system balance will affect application performance. We are primarily concerned with Sandia production application workloads, which typically have non-trivial communication requirements, but the techniques we present could be used to study any application.

In this paper we describe how we are using the Cray XT4 platform to perform empirical experiments with different system balance ratios. So far we have focused on varying link bandwidth and injection bandwidth, but it would also be possible to tune other parameters such as network latency, message rate, and CPU and memory speed. The ability to tune these parameters on a real system allows much larger and longer experiments to be performed than is possible using hardware simulation techniques, which are typically limited to simulating a handful of nodes for a few seconds of simulated time. Empirical experiments also provide a means to validate simulation results and existing application models. The primary downside of our approach is that it is not possible to vary architectural characteristics such as network topology and compute node architecture. It is also not possible, obviously, to tune a parameter outside of the range supported by the hardware (e.g., there is no way to increase link bandwidth, only reduce it).

[*]Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

The remainder of this paper is organized as follows: Section 2 provides an overview of related work. Section 3 describes our approach for tuning link bandwidth and injection bandwidth. Application results from Red Storm running in degraded link bandwidth mode are presented in Section 4. Finally, Section 5 provides a summary and briefly discusses plans for future work.

2 Related Work

There is a large amount of work related to predicting application performance given a set of real or imagined system parameters. One approach is to analyze an application and generate a mathematical model describing it. This approach can often generate accurate performance predictions and has been used successfully to identify sub-optimal performance scaling on the ASCI Q supercomputer [7]. The PMAT (Performance Modeling and Analysis Team) group at Sandia is employing mathematical modeling to predict application performance on future platforms.[1] The chief difficulty of application modeling is the effort required to derive an accurate model for each application studied.

An alternative approach is to simulate the target hardware platform using software or hardware-based techniques (e.g., FPGAs) and then execute applications in the simulated environment. Generally, the performance of a simulator decreases with its fidelity. Cycle-accurate simulators are much slower than functional simulators, which are themselves often 1000x or more slower than real systems. The SST[2] (Structural Simulation Toolkit) is an example of a software simulator being used for computer architecture research at Sandia.

In contrast to these approaches, we detune a real capability supercomputer, Red Storm, and perform empirical experiments in real-time. To the best of our knowledge, this is a unique capability that has not been attempted by others at this scale. Commodity clusters and other proprietary systems, while technically configurable, are usually treated as closed systems from the end-user perspective. Sandia's close relationship to Cray in developing Red Storm provided us access to hardware specifications and system software source code, both of which were required to perform these experiments.

Our work is related to MPI-level detuning work performed on ASCI Red by Brightwell (see article 3 in [2]). The goal of that effort was to evaluate application performance at different system balances and provide input into the design of Red Storm. Similarly, our efforts are aimed at providing input into the design of future systems.

3 Approach

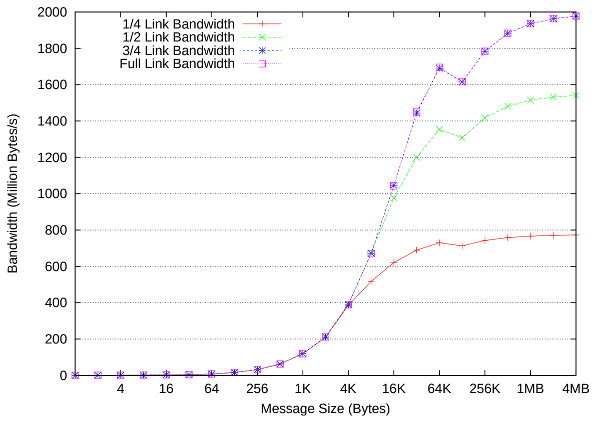
This section describes our approach for detuning the Cray XT4 mesh network links and each compute node's injection bandwidth into the mesh network.

3.1 Link Bandwidth Detuning

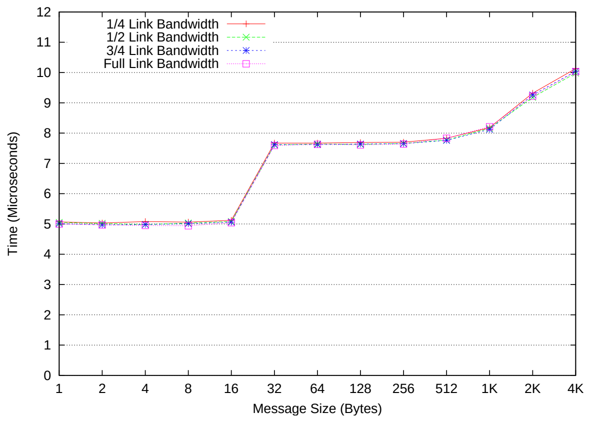
The Cray XT4 interconnection network is arranged in a 3-D mesh topology (some systems employ wrap-around links, producing a torus in some or all dimensions). Each node in the mesh is nominally connected to its six nearest neighbors by point-to-point communication links. Normally, these links are configured at boot time by the Cray route manager to operate at full speed. It is possible, however, to manually configure the links to operate in a degraded mode, resulting in reduced bandwidth.

We originally learned of the degraded link bandwidth mode from the XT4 product brochure [5] on the Cray web site. The document states that, "In the presence of a bad connection, a link can be configured to run in a degraded mode while still providing connectivity." Upon further investigation, we discovered that configuring network links for degraded mode was a simple matter of editing the router manager configuration file (/opt/cray/etc/rm.ini on the SMW) and rebooting. As far as we know, this file and the edits required

[1]http://www.sandia.gov/PMAT
[2]http://www.cs.sandia.gov/sst



(a) MPI Ping-Pong Bandwidth



(b) MPI Ping-Pong Latency

Figure 1: MPI ping-pong bandwidth and latency for the three supported link bandwidth degraded modes: 1/4 bandwidth, 1/2 bandwidth, and 3/4 bandwidth.

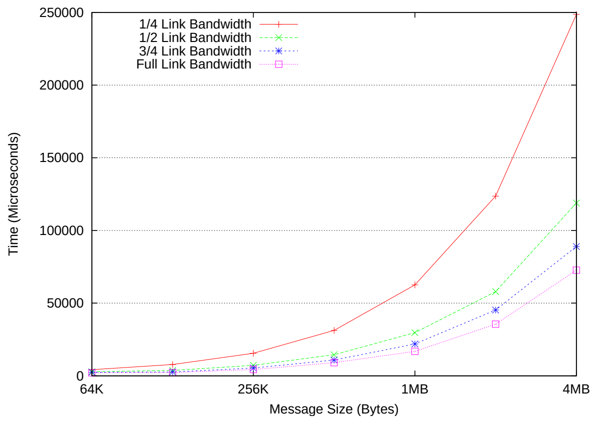
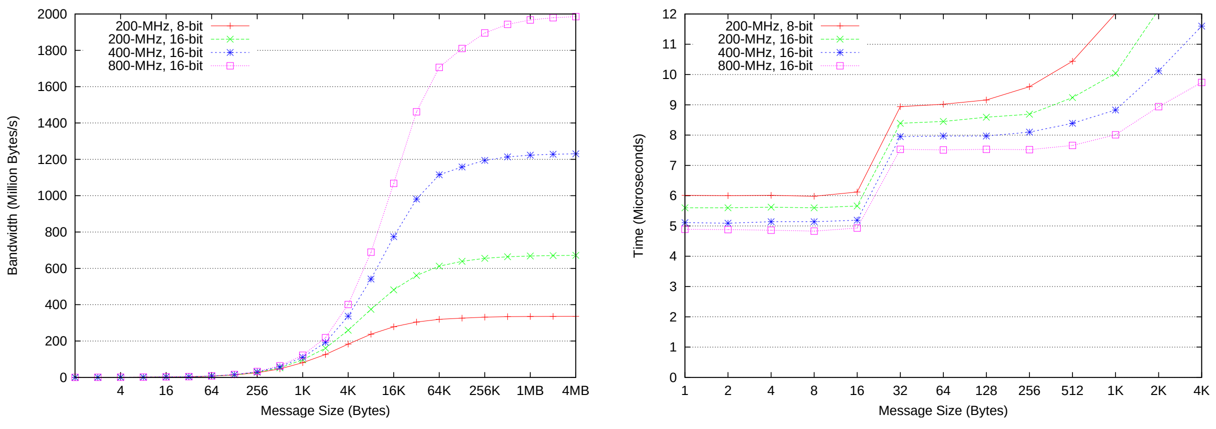


Figure 2: MPI_Alltoall() for 16 processes, one process per node.

to enter degraded mode are undocumented, but they can be derived by examining the SeaStar hardware specification and inspecting the route manager source code.

Three degraded modes are supported: 1/4 bandwidth, 1/2 bandwidth, and 3/4 bandwidth. Figure 1 shows the resulting MPI uni-directional bandwidth and latency for the degraded modes, as well as full bandwidth. The 3/4 and full link bandwidth results are limited by the compute node's injection bandwidth (i.e., the bandwidth of the HyperTransport link between the Opteron and the SeaStar), which is lower than the network link bandwidth. Based on the 1/4 bandwidth results, the effective link bandwidth is approximately 3092 MB/s in each direction. Small-message MPI latency is virtually unchanged for the degraded modes, indicating that link bandwidth can be controlled independently of latency.

Figure 2 shows the performance of a 16 process MPI_Alltoall() for the degraded link bandwidth configurations. MPI_Alltoall() puts a heavier load on the network than the MPI ping-pong test since more than two nodes are involved and every process must communicate with every other process. The increased link contention results in a noticeable difference between 3/4 and full link bandwidth for larger message sizes.



(a) MPI Ping-Pong Bandwidth (b) MPI Ping-Pong Latency

Figure 3: MPI ping-pong bandwidth and latency for the four tested HyperTransport link configurations: 200-MHz 8-bit (400 MB/s per direction), 200-MHz 16-bit (800 MB/s), 400-MHz 16-bit (1600 MB/s), and 800-MHz 16-bit (3200 MB/s).

3.2 Injection Bandwidth Detuning

The bandwidth of the point-to-point HyperTransport link connecting a compute node's Opteron to its SeaStar network interface gates a node's injection bandwidth. Normally, this link is setup at boot-time by Coldstart (Cray's BIOS replacement) to operate at the maximum speed supported by both the Opteron and the SeaStar. The speed negotiation protocol is currently hard-coded in Coldstart—there is no way to change it without editing the Coldstart source code, rebuilding, and rebooting. We used PCI configuration space accesses to determine that the SeaStar chips in our system supported 8-bit or 16-bit wide HyperTransport link configurations operating at 200 MHz, 400 MHz, or 800 MHz. This results in a maximum of 3.2 GB/s per direction (800 MHz * 2 bytes * 2 transfers/clock) and a minimum of 400 MB/s per direction.

Figure 3 shows the MPI ping-pong bandwidth and latency results for the HyperTransport link configurations tested. The maximum bandwidth achieved is approximately 2 GB/s, or 62.5 percent of the theoretical maximum. Unlink the link bandwidth detuning results, detuning the HyperTransport link results in increased small message latency. The difference between the maximum and minimum bandwidth configurations is approximately 1 microsecond.

We ran into a complication when running at the minimum bandwidth configuration where nodes would crash frequently due to HyperTransport watchdog timeout errors. We were able to work around this issue by significantly increasing the watchdog timer's initial count value. This was accomplished by using the xtmemio command to update the appropriate SeaStar configuration register.

4 Application Results

Two applications, CTH and Partisn, were tested on Red Storm in 1/4 degraded link bandwidth mode during a dedicated-time test window in late April. While we would have liked to perform more testing in different link and injection bandwidth configurations, there was only enough time to perform these experiments. The same dedicated-time test window was used to perform quad-core Catamount (N-way Catamount) and Compute Node Linux (CNL) testing on the same applications and problem sets, providing interesting data to compare against.

Only one trial was performed for each data-point presented in this section. Additionally, time prevented us from using deterministic node layouts (i.e., the mapping of MPI tasks to physical nodes). Instead we relied on the batch scheduler to allocate compute nodes. We also ran more than one job at a time, although the larger runs ran alone since they utilized virtually all available nodes. We hope to eliminate some of these caveats in future testing.

4.1 Partisn

Partisn [1] is a particle transport code developed by Los Alamos National Laboratory. The test problem used was setup to stress network latency and resulted in a 24x24x24 array of cells on each process (approximately 50 MB per process). Ideally, a test problem setup to stress network bandwidth would have been used, but we wanted to use the same problem that was used on earlier Catamount and CNL testing.

Figure 4 presents the Partisn results. At 8192 nodes, the 1/4 link bandwidth configuration is 10.3% worse than the full bandwidth configuration. The blue percent difference line in Figure 4(a) corresponds to the right y-axis and is quite jagged, indicating that it would be advisable to run more trials and calculate an average. It is puzzling that the 2, 4, and 8 node cases were actually faster with the degraded 1/4 bandwidth configuration.

Figure 4(b) compares Catamount to CNL, both configured to full link bandwidth. Up until 512 nodes both scale similarly. Catamount performs much better for larger node counts and the difference appears to be increasing with scale. At 8192 nodes CNL is 49% worse than Catamount. This is much more significant than the difference between Catamount configured for 1/4 link bandwidth and Catamount configured for full link bandwidth (10.3%). Figure 4(c) compares the performance of accelerated portals (a NIC offload implementation) [3] with generic portals (a host-based implementation, used for all other results). Accelerated portals has approximately 30% lower latency than generic portals, but produces little performance improvement, even though the test problem was chosen to stress network latency. This indicates that the performance difference between Catamount and CNL is due to something other than network latency and bandwidth.

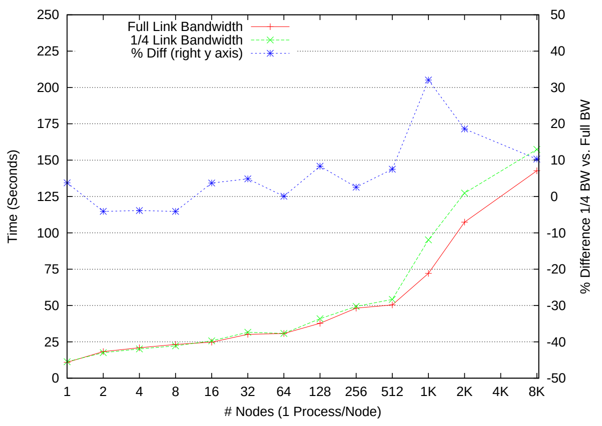
4.2 CTH

CTH [6] is a shock physics code developed by Sandia National Laboratories. The shaped charge test problem was used with either 1.75 million cells per process, 1.25 million cells per process, or 1.25 million cells per socket. Both SN mode ("single node mode", one process per node) and VN mode ("virtual node mode", two processes per node) were tested, although the 1.75M cells/process problem was too large for VN mode (each process is allocated half of the compute node's memory).

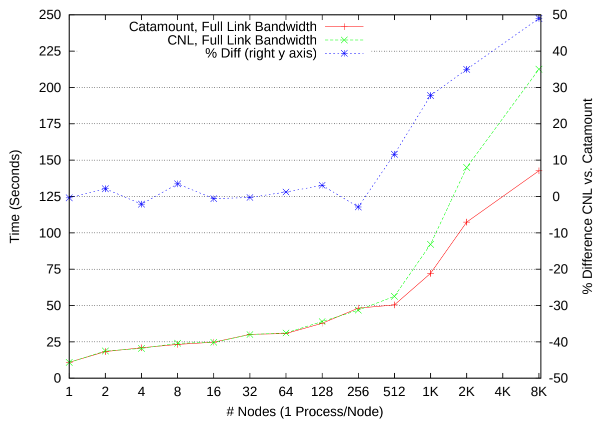
Figure 5 presents the CTH results. At 8192 nodes, the 1/4 bandwidth degraded mode configuration is 32.6% worse than the full bandwidth configuration. Similarly to the Partisn results, the percent difference between the two cases (blue line in Figure 5(a)) is quite variable.

Figure 5(b) shows the difference between Catamount and CNL, both configured to full link bandwidth. The difference between the two cases is much less than was the case for Partisn. One possible explanation for this is that CTH is less sensitive to OS noise (also known as OS jitter) than Partisn, and that CNL has higher OS noise than Catamount.

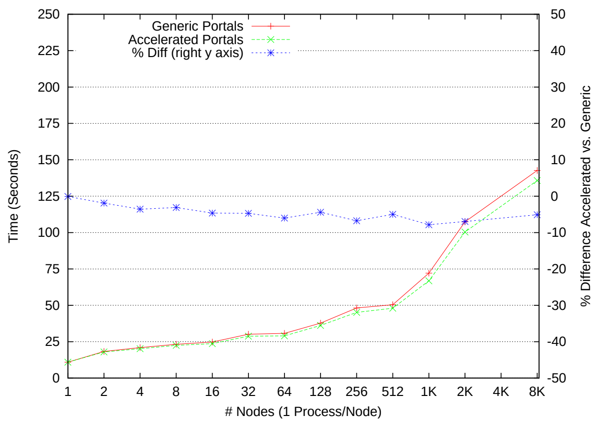
Figures 5(c) and 5(d) present VN mode results. Figure 5(c) compares SN to VN mode results. While the curves are somewhat jagged, the percent difference between 1/4 and full link bandwidth is greater for VN mode (21% at 8192 processes, 4096 nodes) compared to SN mode (13% at 8192 processes, 8192 nodes). This seems to make sense since using two processes per node should increase each node's injection bandwidth and put a heavier load on the network links, increasing the effect of reduced link bandwidth. Figure 5(d) compares VN mode results for two different problem sizes. It is puzzling that the results for 1, 2, and 4 processes are actually better in the 1/4 bandwidth configuration for both problem sizes. We plan to investigate this further on a small test system.



(a) Catamount, Full vs. 1/4 Link Bandwidth

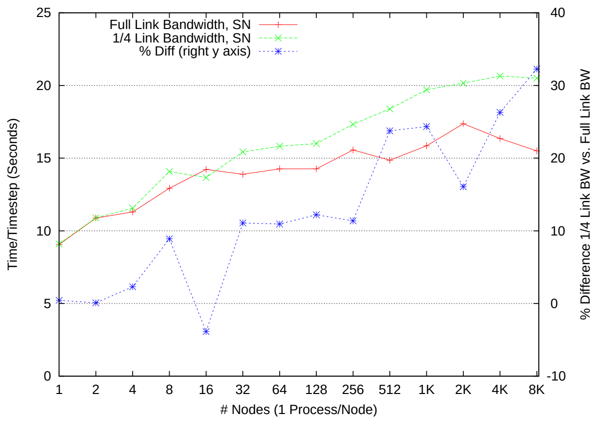


(b) CNL vs. Catamount

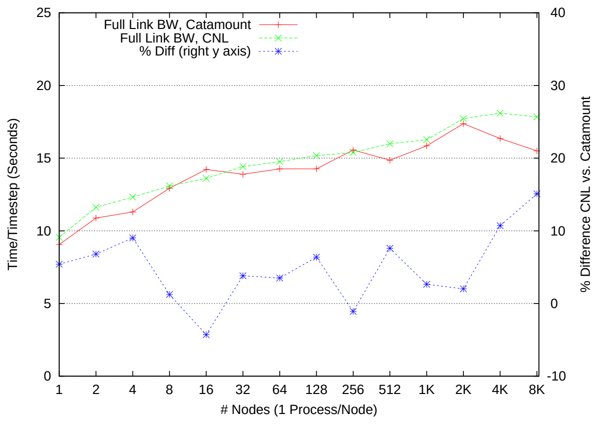


(c) Catamount, Accelerated vs. Generic Portals

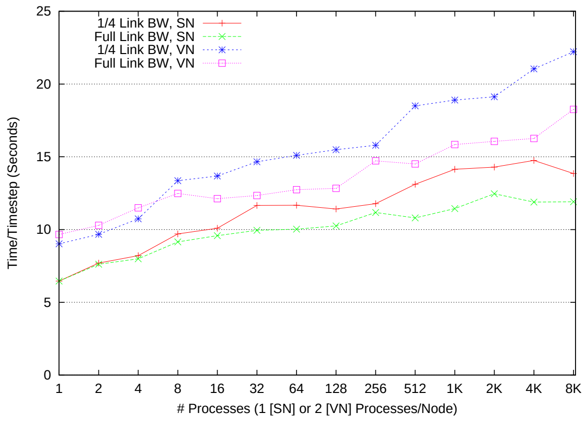
Figure 4: Partisn results.



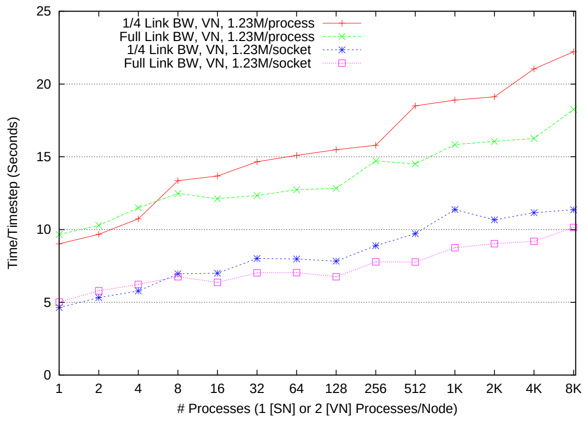
(a) Catamount, Full vs. 1/4 Link Bandwidth



(b) CNL vs. Catamount



(c) Catamount, SN vs. VN Mode



(d) Catamount, VN Mode, 2 Problem Sizes

Figure 5: CTH results.

5 Conclusion

This paper has described and demonstrated the ability to detune network link bandwidth and injection bandwidth on a Cray XT4 system. This capability enables empirical experiments to be performed on large scale systems with different network/compute balance ratios. This information is useful for validating application models, simulation results, and as input to future system design decisions.

In the future we would like to extend the test framework to allow independent control of message latency, message rate, CPU frequency, and memory configuration. We also plan to perform additional testing using checkpoints from production application runs and executing for only a few timesteps. This will allow faster turn-around times, speeding the rate at which the parameter space can be explored.

References

[1] R. E. Alcouffe, R. S. Baker, J. A. Dahl, S. A. Turner, and R. Wart. PARTISN: A Time-Dependent, Parallel Neutral Particle Transport Code System. Technical report, Los Alamos National Laboratory, 2005.

[2] J. A. Ang, D. Barnette, B. Benner, S. Goudy, B. Malins, M. Rajan, C. Vaughan, A. Black, D. Doerfler, S. Domino, B. Franke, A. Ganti, T. Laub, R. Leland, H. Meyer, R. Scott, J. Stevenson, J. Sturtevant, and M. Taylor. Supercomputer and Cluster Performance Modeling and Analysis Efforts: 2004-2006. Technical Report SAND2007-0601, Sandia National Laboratories, 2007.

[3] R. Brightwell, K. T. Pedretti, K. D. Underwood, and T. Hudson. Seastar interconnect: Balanced bandwidth for scalable performance. *IEEE Micro*, 26(3):41–57, 2006.

[4] B. Camp. The Path to Exa-scale: An Architectural Perspective. In *Presentation at the SOS12 Workshop*, 2008.

[5] Cray, Inc. *Cray XT4 Datasheet*.

[6] E. Hertel, J. Bell, M. Elrick, A. Farnsworth, G. Kerley, J. McGlaun, S. Petney, S. Silling, P. Taylor, and L. Yarrington. CTH: A Software Family for Multi-Dimensional Shock Physics Analysis. In *Proceedings of the 19th International Symposium on Shock Waves*, 1993.

[7] 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 *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, 2003.