A Micro-Benchmark Evaluation of Catamount and Cray Linux Environment (CLE) Performance

Jeff Larkin, *Cray Inc.*
Jeffery A. Kuehn, *Oak Ridge National Laboratory*

ABSTRACT: *Over the course of 2007 Cray has put significant effort into optimizing the Linux kernel for large-scale supercomputers. Many sites have already replaced Catamount on their XT3/XT4 systems and many more will likely make the transition in 2008. In this paper we will present results from several micro-benchmarks, including HPCC and IMB, to categorize the performance differences between Catamount and CLE. The purpose of this paper is to provide users and developers a better understanding of the effect migrating from Catamount to CLE will have on their applications.*

KEYWORDS: Catamount, Linux, CLE, CNL, HPCC, Benchmark, Cray XT

1. Introduction

Since the release of the Cray XT series [3,4,5,6,7,18,19] of MPP systems, Cray has touted the extreme scalability of the light-weight Catamount operating system from Sandia National Laboratory. To achieve its scalability, Catamount sacrificed some functionality generally found in more general purpose operating systems, including threading, sockets, and I/O buffering. While few applications require all of these features together, many application development teams have requested these features individually to assist with portability and performance of their application. For this reason, Cray invested significant resources to scale and optimize the Linux operating system kernel for large MPP systems, resulting in the Cray Linux Environment (CLE). Although Cray continues to support Catamount at this time, it is important to assess the performance differences that may exist between the two platforms, so that users and developers may make informed decisions regarding future operating system choices. Moreover, the availability of a two maturing operating systems, one designed as a lightweight kernel and one customized from a traditional UNIX system, provides a unique opportunity to compare the results of the two design philosophies on a single hardware platform. This paper takes the approach of using micro-benchmark performance to evaluate underlying communication characteristics most impacted by the differences between Catamount and CLE. We will briefly discuss each operating system and the benchmark methodology used. Next we will present the results of several benchmarks and highlight differences between the two operating systems. Finally we will conclude with an interpretation of how these results will affect application performance.

2. Operating Systems Tested

Catamount

The Catamount OS [14], also known as the *Quintessential Kernel* (Qk), was developed by Sandia National Laboratories for the Red Storm [1,2] supercomputer. As Cray built the Cray XT3 architecture, based on the Red Storm system, Catamount was adopted as the compute node operating system for the XT3 and future XT systems. By restricting the OS to a single threaded environment, reducing the number of available system calls and interrupts, and simplifying the memory model, Catamount was designed from the ground up to run applications at scale on large MPP systems. As dual-core microprocessors began entering the market, Catamount was modified to add *Virtual Node* (VN) mode, in which one processor acts as a master process and the second communicates to the rest of the computer through this process.

Cray Linux Environment (CLE)

Over the course of 2007 Cray worked to replace Catamount kernel with the Linux kernel on the compute nodes. This project was known as Compute Node Linux (CNL) [15], which is now a part of the Cray Linux Environment (CLE)[1]. Cray engineers invested significant effort into reducing application interruptions from the kernel (OS Jitter) and improving the scalability of Linux services on large systems. The Cray Linux Environment reached general availability status in the Fall of 2007 and has since been installed at numerous sites (at time of writing, CLE has been installed on more than half of all installed XT cabinets). Several of the features supported by CLE, but not Catamount, are threading, Unix Sockets, and I/O buffering.

3. Benchmarks and Methodology

HPCC

The HPCC [9,10,11,12] benchmark suite is a collection of benchmarks, developed as a part of the DARPA HPCS program, that aim to measure whole system performance, rather than stressing only certain areas of machine performance. It does this through a series of microbenchmarks over varying degrees of spatial and temporal locality, ranging from dense linear algebra (high locality) to random accesses through memory (low locality). Benchmarks are also performed on a single process (SP), every process (EP), and over all processes (Global) to measure the performance of the individual components and the system as a whole. Also included in the suite of benchmarks are measures of MPI latencies and bandwidths under different topological layouts. By measuring the machine through a range of benchmarks, HPCC can be used to understand the strengths and weaknesses of a machine and the classes of problems for which the machine is well suited. For the purpose of this paper, HPCC was run in a weak scaling manner, meaning that the problem size was adjusted at each process count so that each process has the same amount of work to be done. The benchmark was run at 64, 128, 256, 512, 1024, and 1280 processes and using both one and two processors per socket.

Intel MPI Benchmarks (IMB)

The majority of applications run on large MPP machines, such as Cray XT systems, communicate using MPI. For this reason it is valuable to measure the performance of the MPI library available on a given system. The Intel MPI Benchmarks (IMB) measure the performance of MPI method calls over varying process counts and message sizes. By having an understanding of how well a machine performs certain MPI operations, application developers can project how their application may perform on a given architecture or what changes they may need to make in order to take advantages of architectural strengths. This benchmark was run as process counts up to 1280 and message sizes up to 1024 bytes.

Test System

The above benchmarks were run on a machine known as *Shark*, a Cray XT4 with 2.6 GHz, dual-core processors and 2 GB of DDR2-667 RAM per core. Tests were run while the system was dedicated, so that the results could not be affected by other users. This system could be booted to use either Catamount or CLE on the compute nodes, a fairly unique opportunity. Catamount tests were run using UNICOS/lc 1.5.61, the most recent release as of April 2008. CLE tests were run on CLE 2.0.50 using both the default MPI library (MPT2), mpt/2.0.50, and the pre-release mpt/3.0.0.10 (MPT3), which was released in final form in late April 2008. The major difference between these two MPI libraries is the addition of a shared memory device for on node communication to MPT3, where on node messages in MPT2 were copied in memory after first being sent to the network interface. This new MPI library is only available for machines running CLE.

4. Benchmark Results

In this section we will present selected results from each of the benchmarks detailed above. Benchmarks that emphasized the communication performance differences between the two OSes were specifically chosen, as benchmarks that emphasize processor or memory performance showed little or no discernable differences. It is important to note that these benchmarks are only intended to be used in comparison of OS configurations previously described. No attempts were made to optimize the results, but rather a common set of MPI optimizations were chosen and a common set of input data was used. With some effort, any or all of these benchmark results could likely be improved, but this is outside of the scope of this paper. All tests were run with the following MPI environment variables set: `MPICH_COLL_OPT_ON=1`, `MPICH_RANK_REORDER_METHOD=1`, `MPICH_FAST_MEMCPY=1`.

HPCC

Parallel Transpose

As the name implies, the Parallel Transpose (PTRANS) benchmark measures the performance a matrix transpose operations for a large, distributed matrix. During such an operation, processes communicate in a

[1] For the purpose of this paper, the terms CLE and CNL will be used interchangeably, although CNL is actually a subset of the software provided in CLE.

pair-wise manner, performing a large point-to-point send/receive operation. This benchmark generally stresses global network bandwidth. Figure 1 illustrates HPCC PTRANS performance.

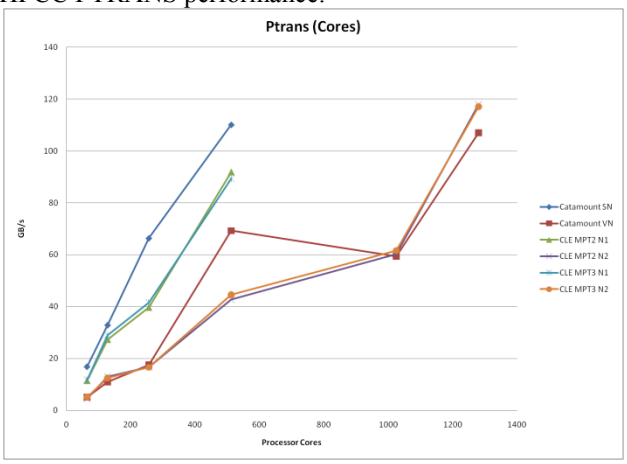

Figure 1: HPCC Ptrans performance, higher is better

The above graph shows two distinct groupings of results, corresponding to the single and dual core results. Due to contention for the shared NIC on dual core runs, one would expect lower performance from dual core, which we clearly see. At all processor counts, Catamount performed better than CLE when running on just one core. With the exception of one outlier, which is repeatable, dual-core performance is indistinguishable between Catamount and CLE. At the highest processor counts, CLE appears to be scaling better on dual core, but the data is insufficient to conclude that this trend will continue at higher scales.

MPI Random Access

As the name implies, the MPI Random Access (MPI-RA) benchmark measures the rate at which a machine can update random addresses in the global memory space of the system. Given a large enough problem space, this benchmark will have virtually no spatial or temporal locality and stresses the network latency of a given machine. Figure 2 shows MPI-RA performance.

The results of MPI-RA are not favourable for CLE. Due to contention for the single NIC per node, it is expected behavior for single core performance to be greater than dual core performance. What is surprising, however, is that Catamount outperforms CLE completely, even when comparing Catamount dual core performance to CLE single core performance. Because this benchmark does not benefit from on-node performance improvements, there is no discernable difference between MPT2 and MPT3 performance.

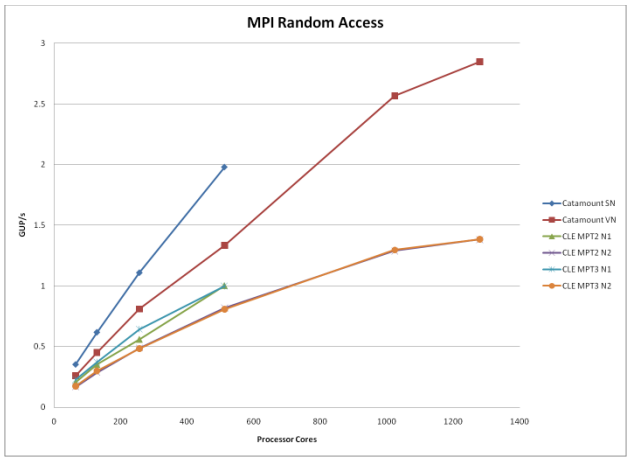

Figure 2: HPCC MPI Random Access performance, higher is better

The results of this benchmark were surprising, but the authors believe that the performance degradation is related to results presented in [17]. In this presentation the author noted that performance of the Parallel Ocean Problem (POP) had a significant degradation in performance when running on CLE unless MPI receives are pre-posted. By posting receives before the associated sends occur, the MPI library is able to more efficiently handle messages as they are delivered to the receiver. The authors of this paper believe that the performance lost in MPI-RA can be attributed to the fact that the benchmark does not pre-post receives.

MPI-FFT

The MPI-FFT benchmark performs a Fast Fourier Transform (FFT) over the global memory space. While this operation stresses the network in a very similar manner to PTRANS, it has a more significant on-node computation aspect. Figure 3 shows MPI-FFT performance.

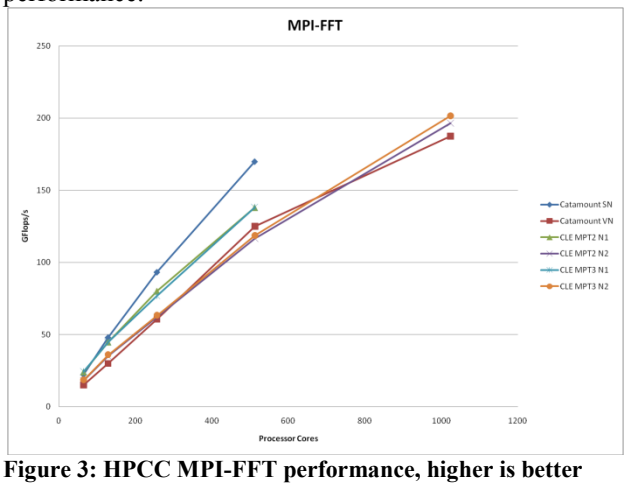

Figure 3: HPCC MPI-FFT performance, higher is better

As was the case with PTRANS, two clear clusters appear in Figure 3, one for single core and the other for dual core performance. Looking first at the single core performance, we see that Catamount is scaling to higher processor counts significantly better than CLE. Dual-core performance, however, shows no discernable difference in performance between Catamount and CLE. As was the case with PTRANS, larger scale runs would be needed to determine whether CLE will continue to perform better than Catamount at scales larger than 1024 processor cores.

Latency and Bandwidth

HPCC reports latencies and bandwidths with five different metrics, minimum, maximum, and average, which should be self explanatory, and also Natural Ring, where each processes communicates with the next MPI ranks, and Random Ring, where processors are ordered randomly. The graphs below will show Natural and Random Ring results, as these results are generally the most telling.

Figure 4 shows lower single core latency when running Catamount, while CLE appears slightly better when running on two cores. However Naturally Ordered Latency shows a more significant difference between the two OSes. In Figure 5 Catamount once again has lower latency than CLE when using just one core. When running on two cores, CLE with MPT2 has worse latency than Catamount, but CLE with MPT3 has significantly better latency, on par with single core performance. Although the latency is still several microseconds higher than single core Catamount results, it is notable that CLE with MPT3 has almost no latency increase when going from single core to dual core.

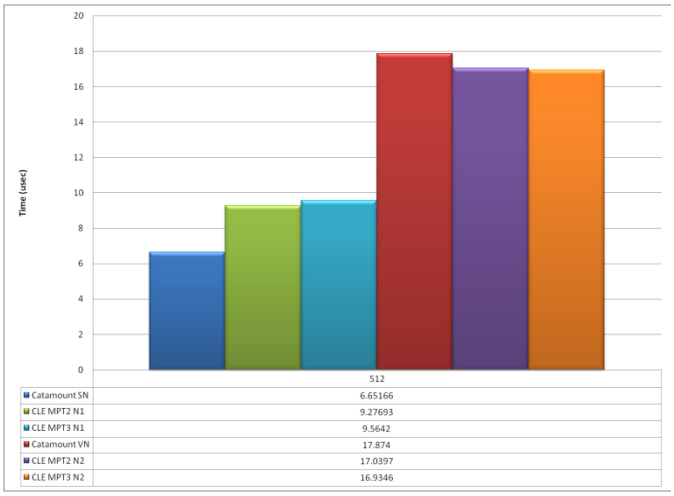

Figure 4: HPCC Random Ring Latency, lower is better

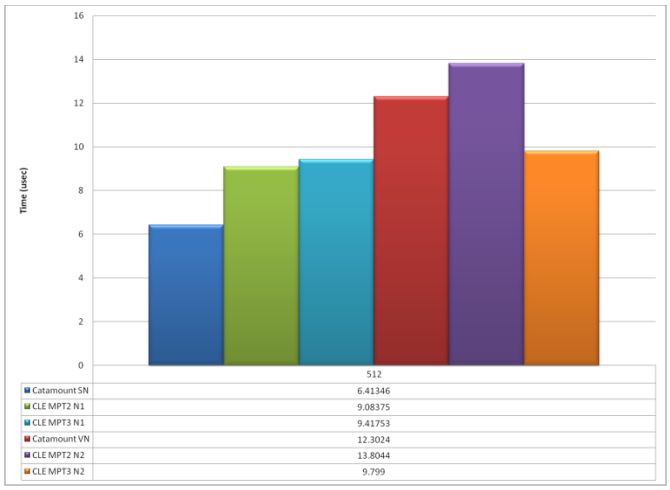

Figure 5: HPCC Naturally Ordered Latency, lower is better

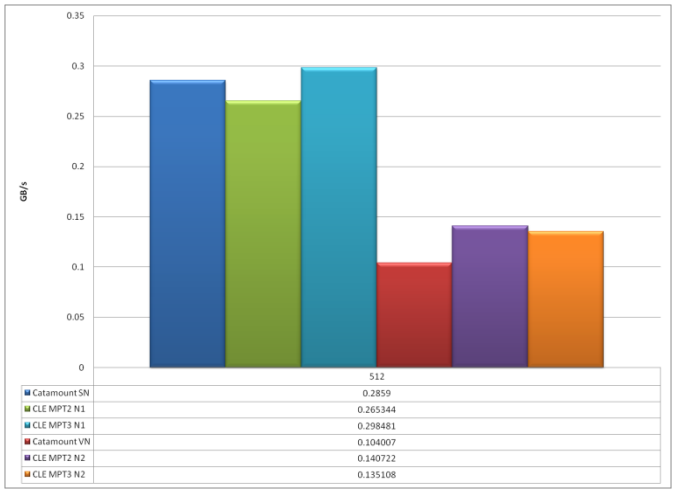

Figure 6: HPCC Random Ring Bandwidth, higher is better

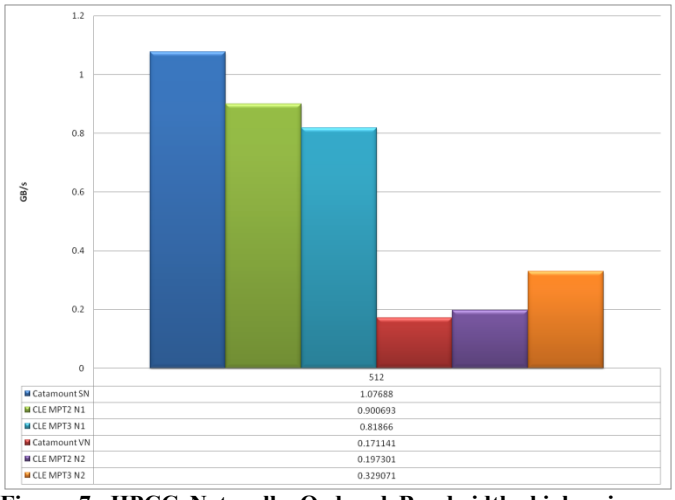

Figure 7: HPCC Naturally Ordered Bandwidth, higher is better

We see in Figure 6 the HPCC Random Ring Bandwidth. As before, Catamount performs better than CLE with MPT2 on a single core, but CLE with MPT3 actually outperforms both by a small margin. CLE again outperforms Catamount when run on two cores, this time MPT2 slightly outperforming MPT3.

IMB

IMB results are presented in two different forms below. For Ping Pong and Barrier benchmarks, the results will be presented as a line graph, where time is along the vertical axis and lower is better. For the remaining tests a 3D surface plot is used, showing the ratio of CLE performance over Catamount performance. For these surface plots, it can be assumed that results below 1.0 are in favor of Catamount and results above 1.0 are in favor of CLE. Both single and dual core performance will be presented for the Ping Pong operation, but only dual core results will be presented for all other operations.

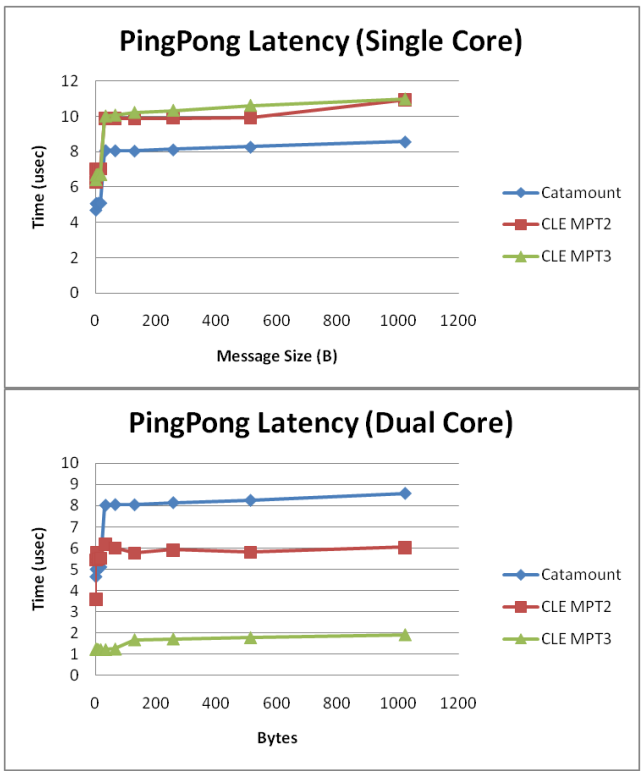

Figure 8: IMB Single and Dual Core Ping Pong Latency, lower is better

Ping Pong

The IMB Ping Pong test measures the time needed to pass a message back and forth, one round trip, between two MPI ranks. Two neighboring ranks are used for this test, so it is generally a best case scenario. The graphs in Figure 8 show the latency of a Ping Pong operation in single and dual core. Notice that Catamount has a lower latency when run on a single processor core and MPT version appears irrelevant in the CLE runs. However, the dual core results are significantly different. First notice that CLE with MPT2 drops significantly when compared to the single core runs outperforming Catamount. This latency decrease is due to the fact the CLE is able to recognize that the message is remaining on the same node and *short circuit* the message to use a memory copy. MPT3 introduces a shared memory device driver, which simplifies on-node message passing and further improves CLE latency. While this is a key practical difference, it should be noted that there is no theoretical barrier to providing similar functionality on Catamount.

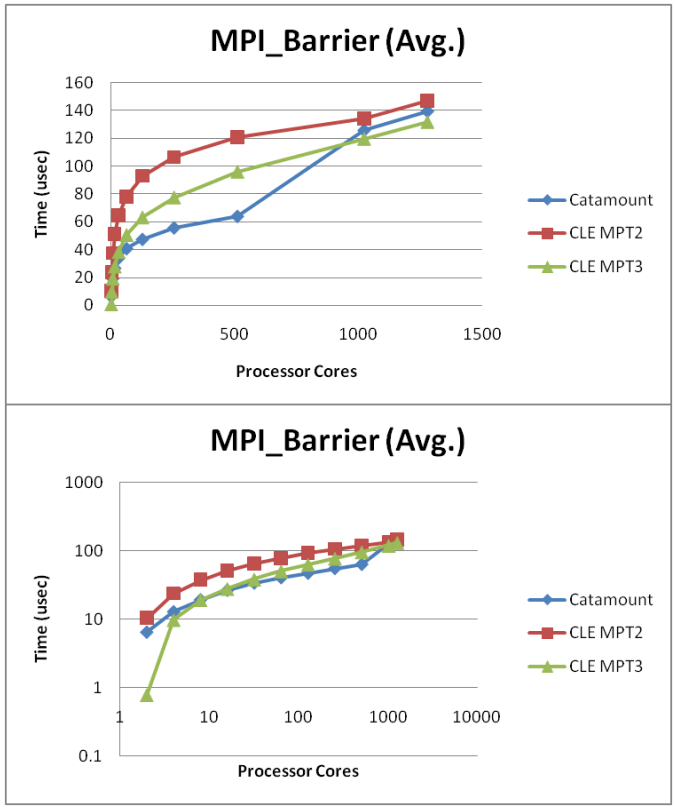

Figure 9: IMB Barrier, top: linear scale, bottom: log scale, lower is better

Barrier

The Barrier benchmark measures the time needed to synchronize all processes using the MPI_Barrier routine. This is a notoriously poor scaling routine, due to the fact that all processors must communicate their participation in the synchronization. Figure 9 show the performance of MPI_Barrier in both linear/linear (top) and log/log (bottom) scales. We clearly see that MPT3 outperforms MPT2, due to the improved on-node performance, and

Catamount outperforms CLE, until some point between 512 and 1024 processor cores, where it is likely that an algorithmic change occurs within the MPI implementation. By switching to a logarithmic scale we are able to see two things that were not obvious in the linear scale. First, notice that MPT3 actually performs better than Catamount up to roughly 8 processes, due to the aforementioned on-node performance improvements. Secondly, although CLE with MPT3 does have better absolute performance, the three lines do appear to converge such that the performance differences at larger scales are small.

SendRecv

The SendRecv benchmark measures the time needed to perform a point to point send and receive operation at varying message sizes and processor counts. In the top graph of Figure 10 we see that for small message sizes and for processor counts below 1024, Catamount performs better than CLE with MPT2. At some point between 512 and 1024, CLE catches up to and surpasses Catamount performance.

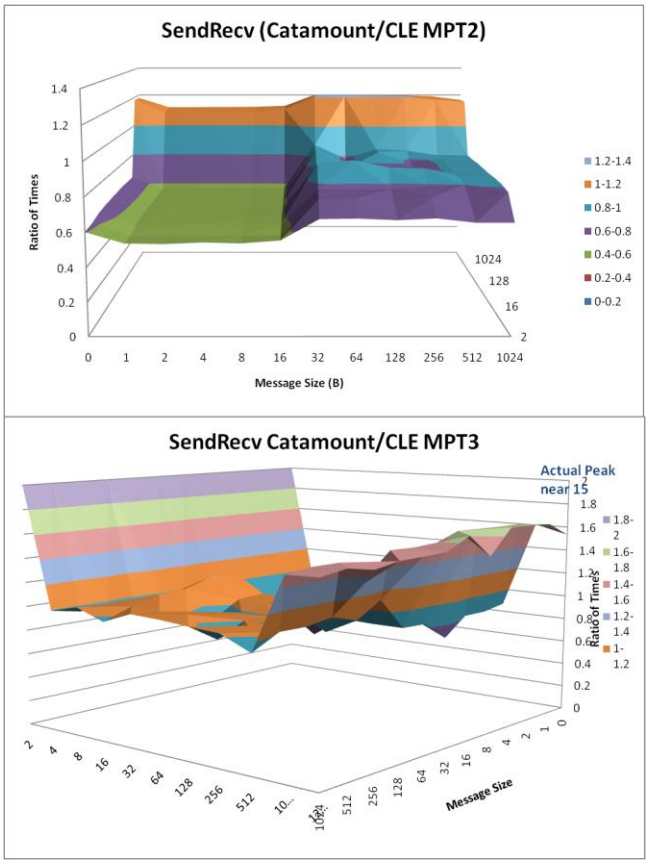

Figure 10: IMB SendRecv

When MPT3 is used with CLE instead, we see that the two OSes perform more similarly. At small processor counts, MPT3 outperforms Catamount by a factor of 15, due to the shared memory device. At moderate processor counts the two perform equally well, but as with MPT2 we see CLE outperforming Catamount at larger processor counts with the crossover again appearing between 512 and 1024 processors.

Broadcast

During an MPI Broadcast operation, one processor in the communicator sends some piece of data to all other processors in the communicator. This operation can be very expensive and details of how it is performed vary significantly between MPI implementations. We see in the both graphs of Figure 11 a distinctive *bath tub* shape, where CLE performs better at the extreme ends of processor counts, but Catamount performs better, by a large margin, in the middle. It does appear that CLE and Catamount performance are roughly equivalent by 1280 processor cores, but without further data we are unable to say whether this trend continues to larger processor counts.

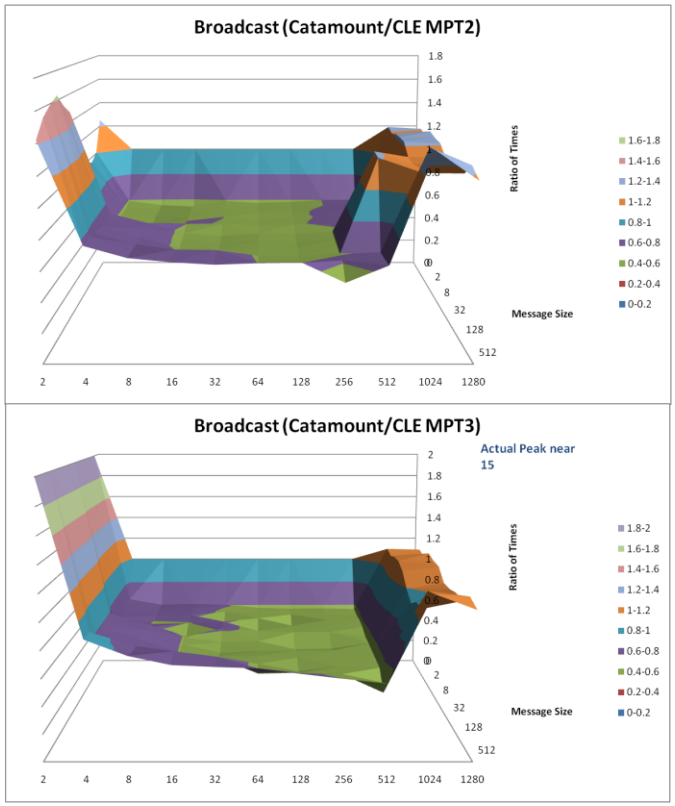

Figure 11: IMB Broadcast

Allreduce

The Allreduce benchmark measures the time needed to perform some operation over data from all processors in a communicator and report the result back to all processors. For example, an Allreduce may take a scalar value from each processor and report their sum back to all of the processors. In the top graph of figure Figure 12 we see that for nearly every processor count and message size, Catamount outperforms CLE with MPT2, although there are signs that CLE may begin to perform equally well at some point beyond 1280 processors. Due to the shared memory device in MPT3, the bottom graph of Figure 12 tells a significantly different story. The on-node performance benefits of MPT3 are obvious out to roughly 16 processors, at which point Catamount begins performing slightly better. Once again, a transition occurs between 512 and 1024 processors with the performance ratio favoring CLE at larger processor counts. Though further work is required to determine if this represents a consistent trend, we see that the underlying algorithm used in the MPT3 implementation of Allreduce is able to benefit significantly from on-node communication improvements.

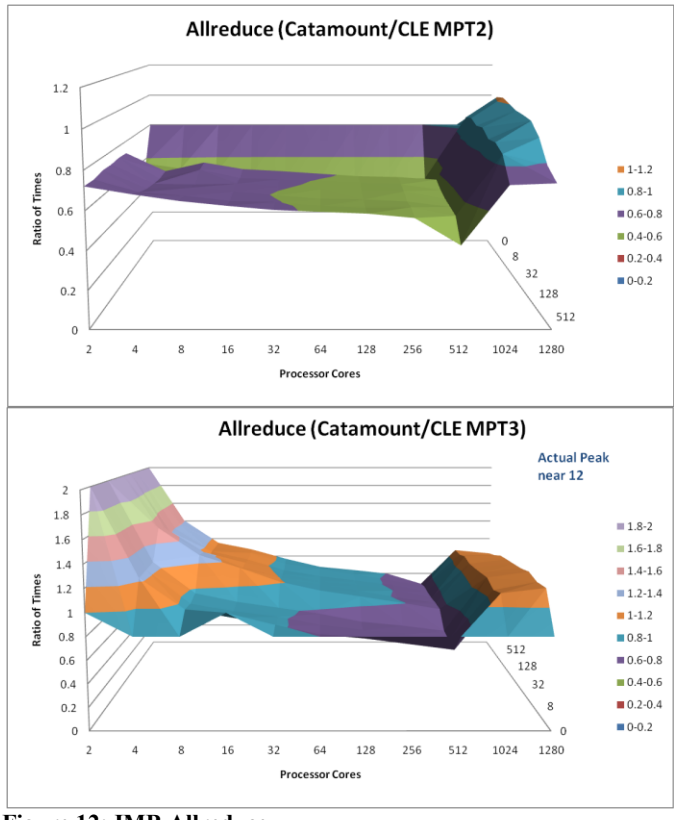

Figure 12: IMB Allreduce

AlltoAll

As the name implies, the MPI AlltoAll operation sends some message from each process in a communicator to each other process in that communicator. This operation has a significant memory requirements at large processor counts and message sizes. For this reason, AlltoAll was only measured up to 1024 processors. We see in Figure 13 that Catamount outperforms CLE with MPT2 across all message sizes and processor counts, although MPT2 appears to improve in comparison around 1024 processors. With the exception of extremely low processor counts, the same can be said for CLE with MPT3, although MPT3 does seem to outperform MPT2.

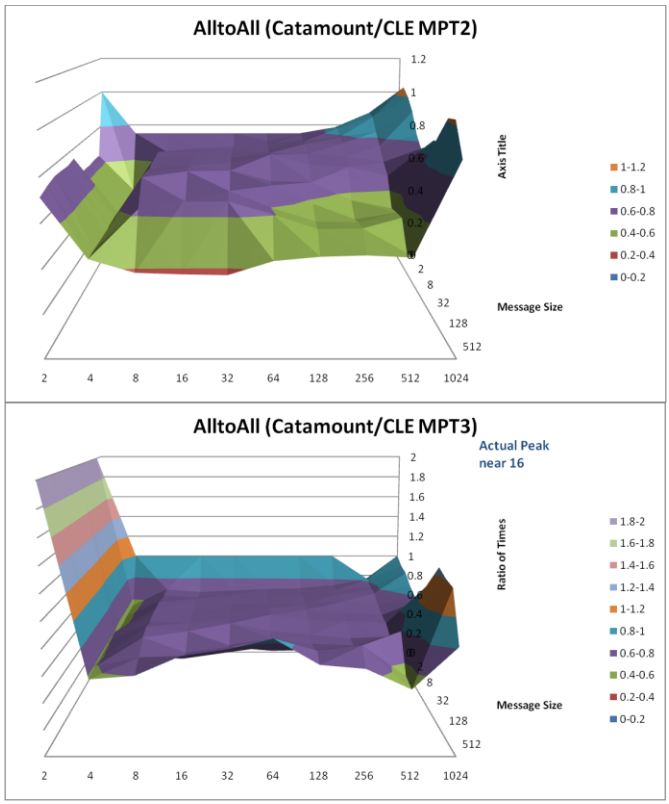

Figure 13: IMB AlltoAll

5. Conclusions and Further Work

While Catamount is a significantly more mature compute node operating system significant progress has been made in making Linux a lightweight and scalable compute node OS.

Given that Catamount is single-threaded and originally designed for single-process nodes, it is not surprising that Catamount has a performance advantage when running on just one core of a dual-core node. However, since Linux evolved in the workstation and server space, where multi-threading is a necessity, it is reasonable to expect it to demonstrate strong competition on multi-core nodes. Applications more sensitive to network latency and bandwidth rather floating point

potential may still benefit from running on Catamount using one processor core per node.

MPT3 seems to level the playing field between Catamount and CLE for almost every kernel we analyzed. On-node communication improvements were particularly beneficial to several benchmarks. Applications that perform primarily nearest-neighbor communications will likely see a significant performance improvement with CLE.

For reasons that we have not yet identified, numerous benchmarks showed a performance advantage for CLE at above 1024 processors. Based on the HPCC results, we speculate that this may be due to improved latency and bandwidth for CLE due to improved sharing of the NIC between the cores. We look forward to extending this study to larger processor counts.

When viewing this data as a whole, we can say that CLE, with MPT3, performs comparably to Catamount. And while individual application performance may vary, we see no reason, given the above data, for applications to perform significantly worse under CLE. In the future we would like to revisit these results in the context of application data.

Given the opportunity to perform further experiments in such a controlled environment, we'd like to also explore the effects of Linux buffering on I/O performance at scale. We are also seeking opportunities to repeat these experiments at larger scales.

6. References

1. W. J. Camp and J. L. Tomkins, "Thor's hammer: The first version of the Red Storm MPP architecture," Proceedings of the SC 2002 Conference on High Performance Networking and Computing, Baltimore, MD, November 2002.
2. Sandia Red Storm System.
3. S. R. Alam, R. F. Barrett, M. R. Fahey, J. A. Kuehn, O. E. B. Messer, R. T. Mills, P. C. Roth, J. S. Vetter, and P. H. Worley, "An Evaluation of the ORNL Cray XT3," International Journal of High Performance Computing Applications, 2006.
4. J. S. Vetter, S. R. Alam, et al., "Early Evaluation of the Cray XT3," Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2006.
5. Cray XT3 Data Sheet, http://cray.com/downloads/Cray_XT3_Datasheet.pdf
6. Cray XT4 Data Sheet, http://cray.com/downloads/Cray_XT4_Datasheet.pdf
7. J.A. Kuehn and N.L. Wichmann, "HPCC update and analysis," Proc. Cray Users Group 2006 Annual Meeting, 2006.
8. D. Weisser, N. Nystrom et al., "Performance of applications on the Cray XT3," Proc. Cray Users Group 2006 Annual Meeting, 2006.
9. P. Luszczek, J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. McCalpin, D. Bailey, D. Takahashi, "Introduction to the HPC Challenge Benchmark Suite," March, 2005.
10. J. Dongarra, P. Luszczek, "Introduction to the HPCChallenge Benchmark Suite," ICL Technical Report, ICL-UT-05-01, (Also appears as CS Dept. Tech Report UT-CS-05-544), 2005.
11. P. Luszczek, D. Koester, "HPC Challenge v1.x Benchmark Suite," SC|05 Tutorial-S13, Seattle, Washington, November 13, 2005.
12. High Performance Computing Challenge Benchmark Suite Website, http://icl.cs.utk.edu/hpcc/
13. Studham, R.S., Kuehn, J.A., White, J.B., Fahey, M.R., Carter, S., and Nichols, J.A., "Leadership Computing at Oak Ridge National Laboratory", Proc. Cray User Group Meeting
14. Kelly, Suzanne, Brightwell, Ron, "Software Architecture of the Lightweight Kernel, Catamount," Proc. Cray Users Group 2005 Annual Meeting, 2005.
15. Wallace, Dave, "Compute Node Linux: Overview, Roadmap & Progress to Date," Proc. Cray Users Group 2007 Annual Meeting, 2007.
16. Intel MPI Benchmarks: Users Guide and Methodology Description, Intel GmbH, Hermülheimer Str. 8a D-50321 Brühl, Germany, June 2006.
17. Worley, Patrick H., "More fun with the Parallel Ocean Problem", Second Annual North American Cray Technical Workshop, 2008.
18. Kuehn, Jeffery A., Larkin, Jeff, Wichmann, Nathan, "An Analysis of HPCC Results on the Cray XT4," Proc. Cray Users Group 2007 Annual Meeting, 2007.
19. Alam, S.R., Barrett, R.F., Fahey, M.R., Kuehn, J.A., Larkin, J.M., Sankaran, R., Worley, P.H., "Cray XT4: An Early Evaluation for Petascale Scientific Simulation," Proc. Of the SC07 International Conference on High Performance Computing, Networking, Storage, and Analysis, Reno, NV, November 2007.