SAND2008-2898C

Application Performance under Different XT Operating Systems

Courtenay T. Vaughan, John P. Van Dyke, and **Suzanne M. Kelly,**, *Sandia National Laboratories*[1]

ABSTRACT: *Under the sponsorship of DOE's Office of Science, Sandia has extended Catamount (XT3/Red Storm's Light Weight Kernel) to support multiple CPUs per node on XT systems while Cray has developed Compute Node Linux (CNL) which also supports multiple CPUs per node. This paper presents results from several applications run under both operating systems including preliminary results with quad-core processors.*

KEYWORDS: Red Storm, XT3, XT4, catamount, CNL, CNW

1. Background

Since the early 1990's Sandia National Laboratories and commercial partners have collaborated to deploy massively parallel processor (MPP) supercomputers based on a hardware and software model of node specialization. These MPP systems have successfully run capability-class problems, where the entire machine can efficiently run a single application on all nodes and achieve a high degree of parallelism.

The most recent collaboration was with Cray, Inc. and Sandia's Red Storm system became the basis for Cray's XT3, XT4, and XT5 products. This product line implements a two-partition hardware and software architecture. Nodes in the service partition have hardware support for PCI-based devices and run a full distribution of the SUSE Linux operating system. On XT3 systems, the nodes in the compute partition run the Catamount Light Weight Kernel (LWK) Operating System (OS) [1]. Starting with the XT4, rather than using the Catamount LWK, the yod job launcher, and the compute processor allocator, Cray is providing the ALPS runtime software. The ALPS software is all custom, newly developed software, with the exception of the compute node operating system. Cray is using a Linux software base that has been tuned to minimize jitter and remove/disable unnecessary services. This version of Linux is called Compute Node Linux (CNL).

CNL and ALPS, like any new software development effort, are subject to the usual risks associated with schedule, stability, and performance. The DOE Office of Science initiated a risk mitigation project that funded Sandia to develop a new version of Catamount. The project was in support of Oak Ridge National Laboratory's (ORNL) XT4 system called Jaguar which is being upgraded to quad-core processors. The immediate goal was to create an enhanced Catamount to support 4 processors per node, suitable to run on a Cray XT4 computer populated with quad-core AMD Budapest Opteron processors.

2. Catamount N-Way (CNW)

The UNICOS 1.4 and 1.5 releases provided a version of Catamount that supported single or dual core AMD Opteron Processors. This version is called Catamount Virtual Node (CVN) since each core operates as a virtual node, supporting a unique MPI rank within a parallel job. The implementation delivered for the risk mitigation project was to be N-way (not just 4-way) and be able to run on single or dual core processors without recompilation. Although untestable, this OS is believed to support 8-core Opterons, should they become available. For this reason, we refer to the latest version as Catamount N-Way, or CNW. The requirements and design for CNW are described in [2]. Briefly, the design is to extend the virtual node concept to every core on a node.

Besides support for four cores, there were two additional functional requirements imposed on this version of Catamount over its predecessor. A second implementation of the Portals networking software is provided. The original version performs protocol processing on the host CPU while the additional one uses

[1] This research was sponsored by Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94550. 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-AC04-94-AL85000.

the processor on the SeaStar network interface chip for protocol processing. The second new functional requirement is support for dual-core and quad-core Opterons in one job. Previous versions of Catamount and the current version of CNL require that the same number of processes run on each node in the job.

Although the predecessor to Catamount, called Cougar, provided an OpenMP implementation, the feature remains unavailable in all versions of Catamount. As core counts continue to grow, Catamount could re-introduce the feature if an all MPI-solution for parallelism becomes unfeasible. The feature was lightly used in Cougar since application developers found it difficult to successfully manage both node-level and thread-level parallelism.

3. Comparison of CNL and CNW

This section provides a brief overview of the two operating systems that can run on an XT computer. Since the purpose of this paper is to present results when the same applications were run under both OSes, an understanding of the architecture differences might illuminate performance variations.

Compute Node Linux and Catamount N-Way have very different heritages and architectural foundations. CNL is based on the Linux kernel, which serves primarily the desktop and server markets. It supports multiple, concurrent users and multiple, independent processes and services. It runs on a wide range of processors and supports a wide range of attached devices. It has large and dynamic [3] code base. Since it is ubiquitous, problems are identified and resolved quite quickly. New software features are added at an astonishing rate.

In contrast, CNW is a limited functionality kernel intended to run one process (per core) for one user application/job. Its only device drivers are for console output and to communicate over the SeaStar Network Interface Chip using the Portals protocol. It has no support for virtual memory and memory addressing is physically contiguous. It supports both 4 KB and 2 MB pages for user applications. The CNW operating system contains approximately 20,000 lines of code, primarily written in the C language. Its goal is to provide the necessary services for an application to run across every node in the system. Further, it does not provide services/features that are known to not scale to the full size of the machine, such as dynamic process creation, dynamic libraries, and virtual memory.

4. Results

In this paper, we have collected results from several machines, including large scale results with dual-core processors on ORNL's Jaguar system and Sandia's Red Storm system, and various small test systems with four quad-core processors.

A. Results from Jaguar

We ran several applications of interest to ORNL last summer on Jaguar, which was then configured as a mix of dual-core XT3 and XT4 compute nodes. These applications include the Gyrokinetic Toroidal Code (GTC) – a 3-d PIC code for magnetic confinement fusion, the Parallel Ocean Program (POP) – an ocean modeling code, and VH1 – a multidimensional ideal compressible hydrodynamics code. The results are shown in Table 1, with the CNL results coming from ORNL.

	CNL 2.0.03+	***CNW 2.0.05+***
	PGI 6.1.6	**PGI 6.1.3**
GTC		
1024 cores XT3 only	595.6 secs	584.0 secs
20000 cores XT3/XT4	786.5 secs	778.9 secs
4096 cores XT3 only	614.6 secs	593.8 secs
POP		
4800 cores XT3 only	90.6 secs	77.6 secs
20000 cores XT3/XT4	98.8 secs	75.2 secs
VH1		
1024 cores XT3 only	22.7 secs	20.9 secs
20000 cores XT3/XT4	1186.0 secs	981.7 secs
4096 cores XT3 only	137.1 secs	117.4 secs

Table 1. Early Jaguar results

The times in the table are run times, so lower numbers represent better performance. These results are somewhat dated since there have been improvements to both CNL and CNW since these were run. These results show an improvement from 1% to 31% for CNW over CNL.

B. Recent Large Results from Red Storm

This summer Sandia is upgrading part of Red Storm to quad-core processors. As part of testing CNW for use after the upgrade, we ran a full machine test to identify any problems with CNW and to compare CNW to CNL. Both systems were based on UNICOS 2.0.44 and the tests were compiled with PGI 6.2.5. We ran a scaling study for two codes and the results presented here are using only one core per processor (the current nodes are dual-core). We ran CTH which is a shock hydrodynamics code with a shaped charge problem and PARTISN which is a time-dependent, parallel neutral particle transport code. Both codes were run in a weak scaling mode with a constant amount of work per processor. The results are shown in Figures 1 and 2.

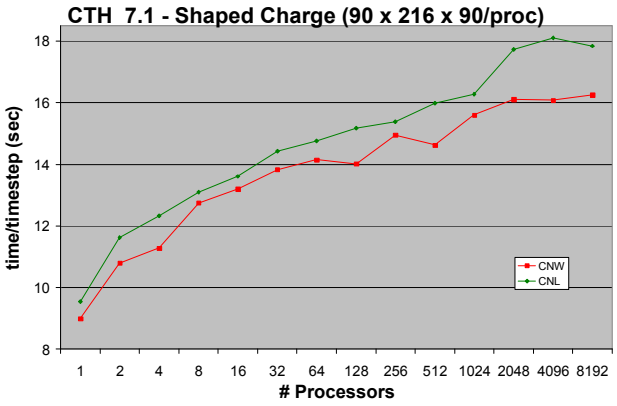

Figure 1. CTH, CNW better at scale

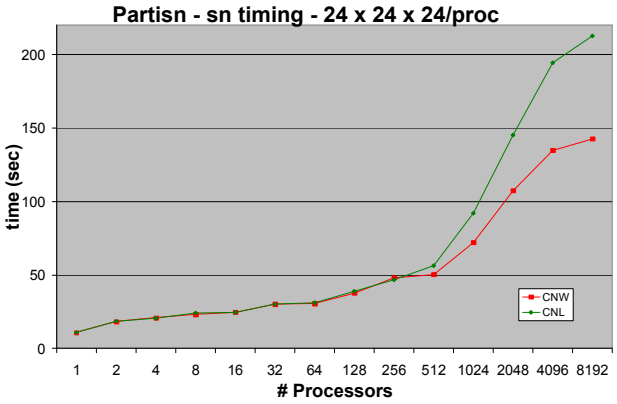

Figure 2. PARTISN, CNW shows better scalability

At 8192 processors, CTH is 9.8% faster with CNW than CNL and PARTISN is 49% faster. The bumps in the CNW CTH runs are from using the Moab queuing system. Red Storm has a mix of nodes with 2GB, 3GB, and 4GB of memory. Moab preferentially uses the 2GB nodes which are located on one end of the machine and all along the fifth row, so the jobs can be laid out in a non-compact form on the mesh. On the other hand, we do not have a queuing system for CNL and the jobs got laid out in a compact form. We are not sure why CTH is showing differences on 1 processor, but the performance differences between CNL and CNW seem to get larger as the number of processors increase. This is shown even more clearly with PARTISN in that the two curves overlay each other up to 256 processors and then diverge. CTH tends to send large messages and is more affected by bandwidth while PARTISN sends more small messages and is affected by message latency.

We also ran the HPC Challenge (HPCC) benchmark suite [4] which provides a variety of benchmarks that span the space of processor and network performance for parallel computers. These benchmarks include HPL (factor a large dense matrix) which emphasizes processor performance, PTRANS (matrix transposition) which tests network bisection bandwidth, STREAMS (vector operations) which tests memory performance, RandomAccess (modify random memory locations across the entire machine) which stresses small message network performance, and FFT (a large 1-D Fast Fourier Transform) which is a coupled processor and network test. For this test, we did not run HPL and ran optimized versions of RandomAccess and FFT. We ran version 1.2 of HPCC on 16384 cores (8192 nodes) and the results are shown in table 2.

Benchmark	units	CNL	CNW	CNW/CNL
PTRANS	GB/s	598.7	894.1	1.49
STREAMS	GB/s	24721	36499	1.48
Random	GUP/s	12.7	23.4	1.85
FFT	GFLOPS	1963.8	2272.2	1.16

Table 2. HPCC on 16384 cores

The numbers in the table are performance measurements and larger numbers indicate better performance. Part of the difference between CNL and CNW for the HPCC tests is due to CNL using small pages while CNW is using large pages. Most of these tests run somewhat better with large pages [5], but that does not explain the whole difference. Benchmarks can tend to be harder on a system than most application codes, but the PTRANS and STREAMS benchmarks have similar performance to PARTISN.

C. Results from Budapest Quad-Core processors

Sandia has a test machine with four quad-core Budapest nodes, each having 8 GB of memory. The base operating system for this machine is UNICOS 2.0.44 and the PGI 6.2.5 compiler was used for all of the tests. We ran two types of tests on these processors. We ran on 16 cores using all four cores on each node, and we also ran a series of tests using four cores in different configurations to explore the utilization of the additional cores on the processors. By running four cores using four nodes with one core per node, two nodes with two cores per node, and all four cores on a node, we are able to see the effect of the contention between the cores for the memory and access to the NIC since the amount of communication and computation is the same for all three cases.

We start by presenting results from running version 1.0 of HPCC in both of these modes. All of the tests are run from the normal configuration of the benchmark suite with no optimized tests. The results are shown in Table 3.

Benchmark	Num MPI Ranks	Cores Per Node	CNL	CNW	CNW/ CNL
PTRANS GB/s	16	4	1.612	2.792	1.73
HPL GFLOPS	16	4	66.55	68.02	1.02
STREAMS GB/s	16	4	31.98	35.13	1.10

Random GUP/s	16	4	0.017	0.035	2.04
FFT GFLOPS	16	4	3.331	3.518	1.06
PTRANS GB/s	4	1	0.576	1.606	2.83
HPL GFLOPS	4	1	17.88	17.90	1.00
STREAMS GB/s	4	1	25.21	25.84	1.02
Random GUP/s	4	1	0.006	0.012	1.83
FFT GFLOPS	4	1	1.609	1.646	1.02
PTRANS GB/s	4	2	0.488	1.551	3.18
HPL GFLOPS	4	2	17.78	18.03	1.01
STREAMS GB/s	4	2	16.45	18.11	1.10
Random GUP/s	4	2	0.006	0.012	1.88
FFT GFLOPS	4	2	1.337	1.360	1.02
PTRANS GB/s	4	4	0.287	1.244	4.33
HPL GFLOPS	4	4	17.59	17.72	1.01
STREAMS GB/s	4	4	7.85	9.95	1.27
Random GUP/s	4	4	0.006	0.011	1.92
FFT GFLOPS	4	4	0.902	0.959	1.06

Table 3. HPCC on Quad-Core Processors

The results here are similar to those obtained for a larger number of processors on Red Storm. HPL, which was not run before, shows little difference between CNL and CNW. Most of the tests show similar differences between CNL and CNW except for PTRANS which shows more difference when all four cores on a node are being used. Again, the CNL tests were run using small pages while the CNW tests were run with large pages. However, on the Budapest nodes, the number of TLB entries for large pages is 128 which has been raised from 8 on the older dual-core Opteron processors. In other tests that we have conducted with these new processors, large pages is almost always an advantage, which is generally from about 1% to 3%, where with the old processors, small pages could be an up to 50% advantage.

We also ran similar tests with ten applications. In addition to the applications that we have already mentioned, we have also run LSMS – an electron structure code, S3D – a combustion modeling code, PRONTO3D – a structured analysis code, SAGE – a hydrodynamics code, SPPM – a benchmark code for 3-D gas dynamics, and UMT2K – an unstructured mesh radiation transport code. We were unable to run VH1 for this test. Table 4 shows the results for running on 16 cores (4 nodes using 4 cores per node) and the numbers are times in seconds.

Application	CNL (sec)	CNW (sec)	CNW/CNL improvement
CTH	1513.1	1298.2	16.6%
GTC	664.9	670.6	-0.85%
LSMS	290.1	276.7	4.84%
PARTISN	499.3	491.3	1.62%
POP	153.8	151.9	1.22%
PRONTO	241.5	222.0	8.78%
S3D	1949.1	1948.9	0.01%
SAGE	267.8	234.9	14.0%
SPPM	847.8	845.0	0.33%
UMT	502.7	472.3	6.44%

Table 4. Results on 16 Budapest cores

The average improvement in CNW performance is about 5% for these applications, which is less than the improvement for the HPCC tests on 16 cores.

Application	Cores Per node	CNL (sec)	CNW (sec)	CNW/CNL Improvement
CTH	1	861.4	816.7	5.47%
GTC	1	583.1	577.7	0.93%
LSMS	1	1160.6	1105.6	4.97%
PARTISN	1	175.1	165.5	5.75%
POP	1	428.0	425.5	0.61%
PRONTO	1	175.8	164.2	7.06%
S3D	1	1327.8	1282.5	3.53%
SAGE	1	170.0	158.9	6.94%
SPPM	1	294.6	293.1	0.51%
UMT	1	1768.8	1701.0	3.99%
CTH	2	949.7	877.8	8.19%
GTC	2	592.9	589.5	0.58%
LSMS	2	1177.3	1118.6	5.25%
PARTISN	2	245.5	234.4	4.77%
POP	2	440.1	435.7	1.01%
PRONTO	2	186.8	175.0	6.74%
S3D	2	1482.2	1439.7	2.95%
SAGE	2	179.9	165.3	8.85%
SPPM	2	297.3	295.2	0.71%
UMT	2	1816.2	1760.4	3.17%
CTH	4	1219.5	1037.8	17.51%
GTC	4	622.8	622.4	0.06%
LSMS	4	1208.1	1144.6	5.55%
PARTISN	4	447.1	441.9	1.16%
POP	4	467.3	464.3	0.66%
PRONTO	4	209.1	195.1	7.18%
S3D	4	1937.3	1940.4	-0.16%
SAGE	4	233.4	190.2	17.47%
SPPM	4	301.1	297.8	1.11%
UMT	4	1944.6	1827.6	6.40%

Table 5. Results on 4 Budapest cores

Table 5 shows the same applications running on four cores in the same three modes that we ran HPCC. As with the 16 core case, times are in seconds for the run of the code. A couple of the codes such as GTC and S3D have large I/O operations in the test problem that was run which is timed as part of the run. Other tests that we have run show that CNL is generally faster with I/O than CNW and it shows in these results. These results also show that the average advantage of CNW over CNL goes up with the use of more cores per node. As with HPCC, part of the explanation of the difference may be that CNL uses

small pages while CNW uses large pages. There are also differences in intra-node message passing such as differences in locking algorithms.

5. Conclusions and Future Work

Sandia has developed and tested a version of the Catamount operating system called CNW (Sandia'a Catamount N-Way) that runs with quad-core processors. In testing that we have done comparing CNW to Catamount, we have found no regressions including regressions in application performance. We have run and compared several applications under CNL (Compute Node Linux) and CNW on several machines with different AMD Opteron processors. In most cases, applications run somewhat faster running with CNW. On large numbers of dual-core processors, CNW shows progressively better performance. On four quad-core processors, the difference between CNL and CNW varies with what code is being run. Some of the differences with the quad-core results can be attributed to the page size that each operating system uses. File I/O performance may be another factor. CNL can make use of on-node buffering whereas I/O is entirely synchronous on CNW. Our testing showed that CNW's iobuf library can alleviate some of the disparity, but still cannot achieve the same I/O performance as CNL.

In the future, we will be testing machines with large numbers of quad-core processors to see if the trends that we have seen with a large number of processors continue with quad-core processors and to see if the trends we saw with four quad-core processors continue on more nodes and how the two effects combine.

References

1. Suzanne M. Kelly and Ronald B. Brightwell, "Software Architecture of the Light Weight Kernel, Catamount," Cray User Group, May 2005.

2. John P. Van Dyke, Courtenay T. Vaughan and Suzanne M. Kelly, "Extending Catamount for Multi-Core Processors," Cray User Group, May 2007.

3. Obed Koren, "A Study of the Linux Kernel Evolution:, SIGOPS Operating Systems Review, 40(2):110-112, 2006.

4. P. Luszczek, J. Dongarra, D. Koester, R. Rabensiefner, R. Lucas, J. Kepner, J. McCalpin, D. Baily, and D. Takahasi, "Introduction to the HPC challenge benchmark suite," March 2005, http://icl.cs.utk.edu/hpcc/pubs/index.html.

5. Courtenay T. Vaughan, "The Effects of System Options on Code Performance," Cray User Group, May 2007.