

SAND2008-4039P
Unlimited Release
Printed June 2008

Catamount N-Way (CNW): An Implementation of the Catamount Light Weight Kernel Supporting N-cores Version 2.0*

**Suzanne M. Kelly, John P. Van Dyke, and Courtenay T. Vaughan
(smkelly, jpvandy, ctvaughn)@sandia.gov
Scalable System Software Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1319**

ABSTRACT

This document contains the project artifacts from the XT4 Catamount Risk Mitigation Project funded by DOE's Office of Science. The deliverable is Catamount N-Way (CNW), a version of the Catamount Light Weight Kernel operating system that can run on Cray XT4 supercomputer systems with quad-core Opteron processors. The requirements and design of the revisions to the Catamount Virtual Node operating system are contained herein. The preliminary results of running applications on CNW at small scale are provided and compare 1) CNW and Compute Node Linux application results and 2) Generic Portals versus Accelerated Portals protocol results. Lastly, in this second version of the document, the results of running applications on up to 7832 quad-core nodes on the Jaguar XT4 system are included.

* 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.

Acknowledgement

The authors thank Fred Johnson of DOE's Office of Science and Al Geist of Oak Ridge National Laboratory for their encouragement and support of this effort. We also thank Don Maxwell of Oak Ridge National Laboratory for his help during our Jaguar test runs.

Table of Contents

EXECUTIVE SUMMARY	4
1. INTRODUCTION AND PROJECT OVERVIEW	5
2. REQUIREMENTS.....	6
2.1. REGRESSION-LESS FUNCTIONALITY AND PERFORMANCE.....	6
2.2. NETWORKING.....	6
2.3. PROCESSES PER NODE.....	7
2.4. SCALABILITY.....	7
2.5. OTHER REQUIREMENTS.....	7
3. DESIGN	8
3.1. LIMIT MEMORY REQUIREMENTS.....	8
3.2. CHANGE “2” TO “N”.....	8
3.3. PCT – QK INTERFACE.....	8
3.4. PROCESS MIGRATION.....	8
3.5. QK MULTI-CPU CODE.....	9
3.6. PORTAL PROCESS IDENTIFIER.....	9
3.7. PORTALS NETWORKING SOFTWARE.....	9
4. RESULTS AS OF MARCH 30, 2008	10
4.1. APPLICATION TESTING.....	10
4.1.1. <i>Large Scale Dual-Core Testing on Jaguar.....</i>	<i>10</i>
4.1.2. <i>Testing on four Quad-Core Budapest Nodes – Small Pages.....</i>	<i>11</i>
4.1.3. <i>Testing on four Quad-Core Budapest Nodes SN vs VN4.....</i>	<i>12</i>
4.1.4. <i>Testing on four Quad-Core Budapest Nodes CNW vs CNL.....</i>	<i>15</i>
4.2. MICRO BENCHMARKS.....	20
4.2.1. <i>OS Noise – Selfish.....</i>	<i>20</i>
4.2.2. <i>Memory – Streams.....</i>	<i>20</i>
4.2.3. <i>Network – Pallas.....</i>	<i>20</i>
4.2.4. <i>File I/O - IOR.....</i>	<i>27</i>
4.3. UNIT TESTING.....	27
5. ACCELERATED PORTALS AS OF MARCH 30, 2008	30
5.1. APPLICATION TESTING.....	30
5.1.1. <i>Testing on four Quad-Core Budapest Nodes.....</i>	<i>30</i>
5.2. MICRO BENCHMARK.....	30
5.2.1. <i>Networking - Pallas.....</i>	<i>30</i>
5.3. UNIT TESTING.....	42
6. RESULTS OF RUNNING ON JAGUAR JUNE 9-10, 2008.....	43
6.1. APPLICATION TESTING – CNL AND CNW RESULTS.....	43
6.2. CTH SCALING STUDY WITH CNW.....	48
6.3. HPL AND HPCC RESULTS.....	49
7. FUTURE PLANS.....	50

Executive Summary

The XT4 Catamount Light Weight Kernel Risk Mitigation Project began in January 2007 and completed in June 2008. The project's charter was to provide a quad-core version of the Catamount compute node operating system for ASCR's flagship Cray XT4 system, called Jaguar, located at Oak Ridge National Laboratory (ORNL). DOE and ORNL are now running the Cray-provided Compute Node Linux (CNL) operating system on Jaguar. However, CNL was a new software development effort and therefore subject to the usual risks associated with schedule, stability, and performance. The Catamount project was the backup software development effort and like CNL, it too met with success. Either Catamount or CNL could support Jaguar in production mode.

This document provides an integrated set of project artifacts. The requirements and design specifications are provided. 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 the processor on the SeaStar network interface chip for protocol processing. This second Portals implementation is complete. 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. This feature is complete. During the design process, we changed the Catamount internal logic from a master processor with at most one slave processor to a master processor with N slave processors. This design change was the basis for naming this version of Catamount, CNW—Catamount N-Way.

The implementation phase stretched longer than originally planned due to the lack of quad-core Opterons for testing. This delay was apparent early enough in the project that staffing levels were adjusted to allow continuous progress, but at a slower pace. The first quad-cores that could be booted were available in January 2008. The months of January through March were spent debugging, testing and producing comparison results with dual-core Opterons and CNL. Those results were provided in the first version of this document and are retained herein. In early June, the opportunity arose to run CNW on Jaguar with 7832 quad-core Opterons. A number of large scale runs were done and those results are presented in this second version of the document. The data provides answers to the two key questions associated with the project: 1) What is the effective utilization of the four cores and 2) What is the performance difference between CNW and CNL?

For Sandia applications, the 2006 upgrade from single-core to dual-core Opterons resulted in an effective increase of 70%. That is, on average for Sandia applications, one dual-core Opteron could do the work of 1.7 single core Opterons. While this may initially appear disappointing, it was actually very worthwhile, since it was done at a cost of 10% over the original purchase price. A 70% performance boost was obtained for a 10% cost. For the ten applications run on the four initial Budapest Opterons, one quad-core Opteron could replace between 2.5 and 3.5 equivalent single-core Opterons. At large scale, we only have results for the CTH application. It showed effective utilization numbers ranging from 2.2 to 3.2 cores.

Another important economic question is the performance difference between CNW and CNL on quad-core Opterons. Of the ten applications compared on *four* quad-core Opterons, CNW outperformed CNL by an average of 6%, with the range being between -0.8% and 21%. However, for the applications important to ORNL, the CNW performance improvement averaged 2%. During the large scale test in June, only ORNL applications were compared. For the four ORNL applications tested (GTC, VH1, POP, and AORSA), the CNW performance improvement averaged 3.8%, with the range being from -14% to 44%. In all cases, CNW outperformed CNL for the tests involving the highest core counts.

1. Introduction and Project Overview

The XT4 Catamount Light Weight Kernel (LWK) Risk Mitigation Project began in January 2007 and completed in June 2008. The project's charter was to provide a quad-core version of the Catamount compute node operating system (OS) for ASCR's² flagship Cray XT4 system, called Jaguar, located at Oak Ridge National Laboratory (ORNL). ORNL is running Compute Node Linux (CNL) from Cray, Inc. on Jaguar as part of its upgrade to quad-core Opteron processors. Should Cray's CNL and/or the associated runtime have shown insufficient scalability, stability, or performance, the quad-core Catamount would have been used.

Rather than using the Catamount LWK operating system, the yod job launcher, and the compute processor allocator, (CPA), 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. These new software components have been delivered and have satisfied the goals of schedule, scalability and stability on quad-core Opterons.

This project has met its goal of providing a version of the Catamount OS that will run on quad -core Opterons. This version is called CNW: Catamount N-Way. The name is derived from the fact that the implementation was done assuming N cores per socket. N has been tested for dual and quad cores. Single core sockets have been simulated, but none were available for actual testing. The design and implementation are also believed to support a value of 8 for N, but that could not be tested either, since none exist. Note that the same OS image can support more than one value of N at the same time, should a heterogeneous mix of processors exist on a machine.

This document provides an integrated set of project artifacts. Variations of the text in most sections have been provided in previous status reports and/or documents. We begin with the requirements and a design strategy. We follow with results from application testing. Although less indicative of "real" performance, we provide the results of testing with micro benchmarks and unit tests. We then discuss the impact of the protocol offload version of the Portals networking software.

In this second version of the document, we provide the results of running CNW at full scale on 7832 quad-core Opterons on the Jaguar XT4 system. We ran four applications of interest to ORNL: GTC, VH1, POP, and AORSA. We used the Generic Portals protocol and then reran the same test with Accelerated Portals. We performed a scaling study using Sandia's CTH application. We have considerable experience with this application could therefore structure strong and weak scaling test problems that used a good percentage of memory. Lastly, we ran HPL and the non-HPL portions of the HPCC benchmarks. These results are contains in section 6.

² ASCR is the Office of Advanced Scientific Computing Research; a part of the Department of Energy's Office of Science.

2. Requirements

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. In so far as reasonable, the implementation should be N-way rather than 4-way and be able to run on single or dual core processors without recompilation.

2.1. Regression-less Functionality and Performance

As is typical of most enhancement efforts, regressions are not acceptable. Existing functionality shall be maintained from the Cray's Catamount Virtual Node (CVN) operating system. Likewise, the existing performance characteristics of the applications shall be retained. Performance improvements, of course, are acceptable.

MPI and shared memory (shmem) applications will be supported. Catamount will continue to interface to other system components, such as the Lustre File System, the Compute Processor Allocator, the batch scheduler, and the RAS system.

There was one identified exception to the no-regression requirement. The undocumented "share mode" feature in the Catamount Virtual Node (CVN) implementation will no longer be functional. Share mode allowed a node to simultaneously run up to four independent user processes. Share mode was available in versions of the light weight kernel prior to Catamount. It never proved useful, complicated the load protocol, and hindered independent progress of an application.

Heterogeneous mode, like share mode is a rarely used feature. The "-F <filename>" option of the yod command allows up to 32 different binaries to be loaded onto independent subsets of nodes in a single job. This functionality shall be preserved with N-way Catamount. Unlike the CVN implementation, the number of processes per node can vary for each subset of nodes³. The same binary shall run on each process on each node within the subset. Only the last node of a given subset can have a number of processes that is different than the rest of the group.

2.2. Networking

Enhancements to the original CVN implementation allow each processor to access the Network Interface Chip (NIC) directly when sending messages. This feature shall be retained for N-way. It ensures the more independent progress of the process on each core.

The CVN implementation only supports NIC-sharing on messages being sent. All received messages are initially processed by the QK on the first CPU, who parses the message to determine the ultimate CPU destination. This limitation is due to host-side protocol processing. N-way Catamount supports NIC-side processing of the protocol. This allows messages to be sent to the target CPU/process immediately upon message receipt.

³ The February 2007 version of the requirements document had stated that the number of processes per node must be the same across the entire heterogeneous job. This was not an acceptable requirement, given the early plan to run a mix of dual and quad cores on Jaguar. The error was caught quite early in the implementation and the restriction was eliminated.

2.3. Processes per Node

With CVN, the number of processes per node is specified with either the `-SN` or `-VN` option. If not specified, the file `/etc/xt.conf` provides the system-wide default. With CNW, `-VN2` is a synonym for `-VN`. And `-VN3` and `-VN4` have been added to request three and four processes per node, respectively. There was early discussion of making major changes to how the user specified the total number of application processes and how many processes should run on each node. No consensus was reached, so the interface remains largely unchanged from CVN.

2.4. Scalability

The OS shall be scalable to at least 100,000 nodes. There should be no limit on the number of virtual nodes except the 2^{31} limit on the signed integer value.

Memory usage by the OS itself shall be minimized and not scale with the size of the machine.

2.5. Other Requirements

The initial version of N-way Catamount will not support applications using OpenMP. Some consideration was given to re-introducing OpenMP-style support in Catamount. Prior light weight kernel versions of Catamount did support a simple threaded model. It was removed from Catamount since it was rarely used and had atrophied through the years.

N-way Catamount will retain three design choices made in CVN. After initial job start up, a process is permanently bound to a particular processor. The heap is divided equally among virtual nodes. There is no shared memory between application processes on a node. (The `shmem` library is supported for sharing memory access between any virtual nodes in a job.)

3. Design

From the application perspective, Catamount's virtual node mode makes the processors as equivalent as possible. From the system perspective, much of the behavior is master-slave with `cpu-0` continuing to be master in Catamount N-Way. In CVN, many Catamount system activities are divided between "SYSTEM_SIDE" and "USER_SIDE". While the architectural perspective of one master is retained in CNW, the handling of the other CPU's is modified from "the one user" to "the N others".

3.1. Limit Memory Requirements

The requirement for OS memory to not grow with the number of nodes is not met by CVN. Catamount's Process Control Thread (PCT) has a number of static arrays that are dimensioned by the maximum number of virtual nodes. These are used during the job load process and can be eliminated by borrowing space that the application will ultimately use. The PCT's use of `malloc` after initialization is very restricted since the PCT's heap is used for application memory. Hence, fragmenting the PCT's heap would impact the maximum memory available for the application. The current N-way implementation uses a shared read-only memory region for the application that contains the application's node map and a single executable text section. This space is allocated early in job load and the text portion is used for the various temporary tables the PCT requires to load the job.

3.2. Change "2" to "N"

Conceptually the changes to go from two-way to N-way are quite simple. In CVN there are many places where there are separate paths for handling the two processes or processors. The processing for other than `cpu-0` needs to become a loop over processors. In some cases, it can be combined to a loop over all processors. There were a few places where a loop over "N" was not possible. The logic for each individual node is unique. C preprocessor commands flag these places when "N" is changed to a value greater than 4.

Certain OS structures need expansion to support the increased number of processors. For example, the "other processor" field in the Process Control Block is now dimensioned and references converted to loops, as appropriate.

3.3. PCT – QK Interface

There are generalizations to the interface between the PCT and the QK for virtual node initiation and subsequent job scheduling. Rather than looping over calls into the QK, it is more efficient to modify the QK's API to specify the number of times a request shall be executed.

3.4. Process Migration

In CVN, the two processes on a node both start on `cpu-0` and the second is "migrated" to `cpu-1` by an application system call from the start up library and then the application notifies the PCT. The N-way migration process is made more robust by making the only application system calls for this go through the PCT and allow the PCT to make a single call into the QK to migrate the list of processes.

3.5. QK Multi-CPU Code

There are several places in the CVN QK where there are separate entries or paths for the CPUs. These need to be replicated or generalized. The current N-way implementation does some of each. Where there is hard-coded replication, it is, at present, 4-way. The behind the scenes handling to provide the software with cpu-id information needed slight generalization.

3.6. Portal Process Identifier

In CVN, at the request of Lustre, the local pids, which were also used as portal pids, were being used round robin, instead of first available. Whereas 4-way would use all pids since MAX_NUM_PROCS is not increased, we could prevent early reuse of the portal pid by defining it with a moving bias added to the local pid. Testing proved that this changed was no longer necessary and was not implemented.

3.7. Portals Networking Software

During the development of the Red Storm system, two versions of the Portals communication API were conceived. One version would run on the host, while the second version would run on the NIC. (These are sometimes referred to as “generic” and “accelerated” Portals.) The second version has only recently been fully developed and integrated into CVN V2.1. It has been introduced, tested, and integrated into CNW. Using the NIC to process the protocol should become more important as the number of cores being supported by one NIC grows. See Section 5 for more information on this feature.

4. Results as of March 30, 2008

The CNW implementation is complete. It was tested at large scale on Jaguar in the summer of 2007, when the system consisted of over 10,000 dual-core Opteron nodes. It has only been tested on four quad-core Opteron nodes as that is all we had access to during the planned project timeframe. We provide application performance results from the large scale dual-core test and from the four quad-core nodes. For most tests, we have the corresponding CNL results and those are presented as well. We also provide the results from micro benchmark testing and from unit testing.

The results in this section all used the host-based Portals networking software implementation, called “generic Portals”. Generic Portals (GP) does all protocol processing on the host CPU, thus taking compute time away from the application. Sandia implemented a second version of the Portals protocol called “accelerated” Portals. Accelerated Portals (AP) uses the processing power available on the NIC. This kind of implementation is generally referred to as a protocol offload engine, or POE. Those results are provided in the next section. The discussions are more easily focused when the comparisons are between only two variations at a time (CNL versus CNW and then CNW/GP versus CNW/AP).

4.1. Application Testing

4.1.1. Large Scale Dual-Core Testing on Jaguar

On June 16-17, 2007, Cray booted Jaguar with UNICOS/lc 2.0.5 to test CNL. For our test on June 18th, the CNW binary was applied to the boot image (aka /raw0) and the system rebooted with CNW compute nodes. The goals of the test were to

- Run applications that are important to ORNL. The selected applications were GTC, LSMS, S3D, VH1, and POP
- Compare to CNL results obtained by Don Ferry in May by using the same test problems and node counts.
- Verify that CNW can support nodes with different numbers of cores in one job (i.e. pretend that some of the dual-core nodes were single-core).

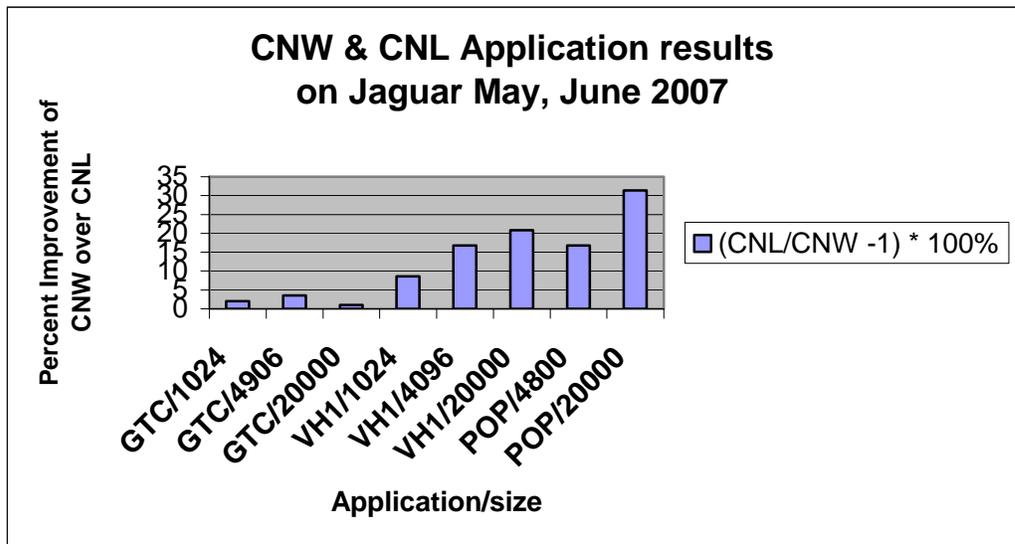
The results for the five applications are itemized in Table 1 below.

	CNL 2.0.03+	CNW 2.0.05+
	PGI 6.1.6	PGI 6.1.3
GTC		
1024 XT3 only	595.6 secs	584.0 secs
20000 XT3/XT4	786.5 secs	778.9 secs
4096 XT3 only	614.6 secs	593.8 secs
LSMS		
23000 XT3/XT4	544.3 secs	Invalid result
S3D		
20000 XT3/XT4	981.9 secs	no results
4096 XT3 only	556.1 secs	no results
VH1		
1024 XT3 only	22.7 secs	20.9 secs
20000 XT3/XT4	1186.0 secs	981.7 secs
4096 XT3 only	137.1 secs	117.4 secs
POP		
4800 XT3 only	90.6 secs	77.6 secs
20000 XT3/XT4	98.8 secs	75.2 secs

Unfortunately the test time ended before we could diagnose the problems with LSMS and S3D. We later determined that our version of LSMS was built with some debug enabled. With 20,000 cores providing diagnostic output one character at a time; the standard output was incomprehensible and looked like garbage. We had millions of lines of print's which caused the results to be unusable.

The S3D tests terminated almost immediately under CNW after reporting file not found on ../data/pressure_wave_test.0000E+00/field.00000. We later determined that S3D was linked with incompatible versions of libsysio.a and liblustre.a.

The July 31st test shot was cancelled and we were unable to obtain results at scale for those two applications. Particularly disappointing is the fact that these were the two applications that had shown better performance with CNL over CVN. Given that caveat, we provide summary graphs of run time comparisons between CNL and CVN for the applications that did run correctly.



During this same test shot, we did run HPL across the entire machine, with some nodes configured as one core per node and some with two cores per nodes. The job ran to completion and provided the expected result.

There were two non-repeatable job launch hangs and one job that hung at termination during the testing. Although the exact cause of these failures is unknown, there have been code changes in these areas that hopefully addressed the issues.

4.1.2. Testing on four Quad-Core Budapest Nodes – Small Pages

Using 2 MB large pages was a performance advantage of the light weight kernels prior to Catamount. However, large pages were often a disadvantage for applications using Catamount on the first and second generations of AMD Opterons. The problem was due to the very small number of Translation Look aside Buffers (TLBs) available for large pages versus small (4 KB) pages. The early generations had 8 large page TLB entries and 512 small page TLB entries. The Budapest generation kept the same number of small page entries and increased the number to 128 for large pages. We ran our test applications with and

without yod's small page option. Most applications run slightly better (less than 3%) with large pages. However, some HPCC benchmarks ran significantly better with large pages. An exhaustive study was not done, but we collect the data points we have in the following table.

Code	Mode	Performance of small pages relative to large pages
HPCC/PTRANS	SN	36.2%
HPCC/HPL	SN	98.5%
HPCC/STREAMS	SN	98.1%
HPCC/Random	SN	87.9%
HPCC/FFT	SN	97.6%
CTH	SN	97-99%
CTH	VN4	103%
GTC	Various	99%
LSMS	VN4	98.9%
POP	Various	97%
S3D	Various	99.4%
VH1	Various	98.6-100.02%

4.1.3. Testing on four Quad-Core Budapest Nodes SN vs VN4

The purpose of this section is to obtain a first-look into how well applications might utilize four CPU cores while sharing memory and NIC access. We do this by running the same problem on one core per node (SN), two cores per node (VN2), and four cores per node (VN4). All problems use four MPI ranks. Since the process count is so small, the results may not prove to be a very reliable forecaster of applications at large scale. Additionally, we are concerned that not all test cases sufficiently taxed processor resources, in particular memory.

The results in this section were run on an XT4 board with four 2.2 GHz Budapest B2 Opterons. Each node has 8 GB of memory. The system did not have a Lustre file system, so all I/O was forced to go through yod to the user's nfs-mounted home directory. PGI 6.1.3 compilers with ACML 3.6.1 were used. Note that PGI 7.1 and ACML 4.1 are needed to utilize the extra SSE instructions.

We begin with the summary. We attempt to quantify the impact of going from one core per node to four cores per node by calculating an effective utilization value.

Application	Utilization of each Core	Cores Effectively Used
CTH	71%	2.84
POP	96%	3.83
GTC	91%	3.66
S3D	64%	2.56
LSMS	97%	3.87
VH1	94%	3.76
PRONTO	79%	3.18
SAGE	74%	2.95
UMT2K	91%	3.62
PARTISN	40%	1.60

By way of example, here is how the calculation is done:

- Assume a test case runs in one hour on all four cores of a single Opteron
- Assume the same test case runs in .85 hours on a single core of four Opterons (.85Q=S)
- Assuming the single core test case is utilizing 100% of its processor, in quad-core mode it is utilizing 85% of each processor
- The quad-core test case is effectively using 3.4 of the 4 processors (.85 x 4 = 3.4)

It is difficult to make predictions based on such problem sizes, but most applications should be able to use 2.6 to 3.9 cores of a quad-core node over a single core per node run.

HPCC⁴ stresses various hardware resources that are important to scientific calculations. The results from key portions of the test are provided in the following table.

HPCC Test	SN mode	VN2 mode	VN4 mode
HPL – GFLOPS	17.90	18.03	17.72
PTRANS – GB/s	1.606	1.551	1.244
STREAMS – GB/s	25.84	18.11	9.95
Random Access – GUPs	0.01182	0.01150	0.011476
FFT – GFLOPS	1.646	1.36	0.959

After the application tests were run, we found a bug in the Portals code that allows intra-node messaging to be done via a memcpy, rather than via the SeaStar NIC. We have corrected the bug, but have not found that results differ significantly from the results that follow in this section. The results in all other sections are from after the bug was corrected.

CTH is a popular shock hydrodynamics code. The selected shaped charge problems use a large number of materials and consume at least 1 GB of memory per MPI rank. The large memory footprint ensures the entire problem does not fit in each cores' private L2 cache. In other words, the effect of sharing the memory bus will be seen in these results.

CTH – time on 4 cores	SN mode	VN2 mode	VN4 mode
Shaped 4	8.385	9.338	11.832
Shaped 2	4.534	5.056	6.901
Meso 2	12.312	13.863	18.023
Meso 1	6.482	7.076	8.953

Timing results in the Parallel Ocean Program (POP) include I/O time. The I/O time dominates the calculation, thus making it a measure of the I/O capability, rather than core utilization. The test problem supplied with POP was run once sending the output the user's nfs-mounted home directory and once to the in-memory /tmp file system. That difference dominates the results, more so that the difference between running on one core per node and all four cores. We are also concerned that the supplied test problem is very small and may well fit in each core's L2 cache, thus showing higher than expected per-core utilization.

POP – time on 4 cores	SN mode	VN2 mode	VN4 mode
------------------------------	----------------	-----------------	-----------------

⁴ HPCC is not an application, but does not fit well in the micro benchmark section of this document either. The analysis was similar to that done for applications, so its results are reported here.

I/O to nfs-mounted /home	25.43	25.87	25.98
I/O to tmpfs /tmp	11.81	12.04	12.34

The Gyrokinetic Toroidal Code (GTC) is a 3-d PIC code for magnetic confinement fusion. The time includes I/O of about 1.2 MB. It took 9.3% longer in VN4 mode than in SN mode. The computation difference is probably longer, but hidden by the I/O time.

GTC - 4 cores	SN mode	VN2 mode	VN4 mode
Time (sec)	593.6	607.7	649.2

S3D is a combustion modeling code. The reported time includes I/O of 125 MB, which is a subset of what of the original source code produced. The program writes one dump file per process. For this environment, the I/O for these core dumps is serialized through yod.

S3D - 4 cores	SN mode	VN2 mode	VN4 mode
Time (sec)	1654	1857	2580

The LSMS electron structure code also includes about 200 KB of output in its timing results. This is a small amount of I/O. It only took 3.3% longer in VN4 mode than in SN mode

LSMS - 4 cores	SN mode	VN2 mode	VN4 mode
Time (sec)	36.7	37.0	37.9

The VH1 multidimensional ideal compressible hydrodynamics code came with a small test problem. It took 6.4% longer in VN4 mode over SN mode.

VH1 - 4 cores	SN mode	VN2 mode	VN4 mode
Time (sec)	115.7	117.0	123.1

Pronto3D is a structured analysis code. The walls problem simulates two sets of brick walls hitting each other and is therefore contact intensive. The VN4 mode takes 26% longer than SN mode.

Pronto3D - 4 cores	SN mode	VN2 mode	VN4 mode
Grind time e-6 (sec)	6.116	6.601	7.699

For the SAGE test (a hydrodynamics code), we used the timing_c problem with 250000 cells/PE. It took 5.8% longer in VN2 mode than SN mode, which is consistent with dual-core results. The VN4 results are 26.3% longer than SN mode. *For the SAGE table, a higher result is better.*

SAGE - 4 cores	SN mode	VN2 mode	VN4 mode
CC/sec/PE	6181.6	5823.6	4556.2

SPPM is a benchmark code for 3-D gas dynamics using a simplified version of the PPM (piecewise parabolic method) code. The test case uses 192 x 192 x 192 cells per processing element. The code ran faster in VN2 and VN4 mode than in SN mode. More study is needed, but an initial guess is communication effects.

SPPM - 4 cores	SN mode	VN2 mode	VN4 mode
Time (sec)	317.1	301.3	305.7

UMT2K is an unstructured mesh radiation transport code. It took 10.5% long in VN4 mode than SN mode.

UMT2K - 4 cores	SN mode	VN2 mode	VN4 mode
Time (minutes)	28.53	29.51	31.52

Partisn is a time-dependent, parallel neutral particle transport code. It reports grind times for the transport and diffusion phases in nanoseconds. It takes 28% to 46% longer in VN2 mode than in SN mode. It takes 2.2 to 2.8 times longer in VN4 versus SN mode. This result is unfortunately consistent with Adolfo Hoise's result in his SC07 paper.

Partisn - 4 cores grind time nsec	SN mode	VN2 mode	VN4 mode
184 MB/core	32.56 / 28.23	47.55 / 39.45	90.78 / 74.07
367 MB/core	41.73 / 29.11	53.52 / 38.51	92.02 / 71.02
1444 MB/core	34.70 / 26.93	48.43 / 35.37	89.45 / 65.53

4.1.4. Testing on four Quad-Core Budapest Nodes CNW vs CNL

The purpose of this section is to look at performance on CNW and compare it to CNL performance. We attempted to make this comparison as head-to-head as possible. The same hardware was used for both tests. The base system was UNICOS 2.0.44 with one XT4 board containing our four quad-core Budapest nodes, each having 8 GB of memory. The same test problems were used for both CNW and CNL. The default page size is used for both systems—4 KB pages for CNL and 2 MB pages for CNW. This system has a 2TB 2-OSS/4-OST Lustre file system. The PGI 6.2.5 compiler was used when building the CNL and CNW binaries. File I/O was to Lustre in both cases.

The system was booted with CVN using vanilla 2.0.44 software. We then swapped out the 2.0.44 CVN binary (aka stage2.sf), the portals module, the SeaStar driver firmware, and vmlinux with the corresponding CNW versions. The stage2.sf contains the Catamount image. The last three files contain modifications for Accelerated Portals. The system was rebooted with CNW and the tests run. We then booted the system using vanilla 2.0.44 software, but this time specifying CNL compute nodes and the tests were rerun.

As in the last section, we begin with the non-application, HPCC. The HPL result is very similar between CNL and CNW, with less than a 1% difference. Large pages give CNW a significant benefit for the Random Access and PTRANS tests. For the following table, larger numbers are better when comparing columns 4 and 5.

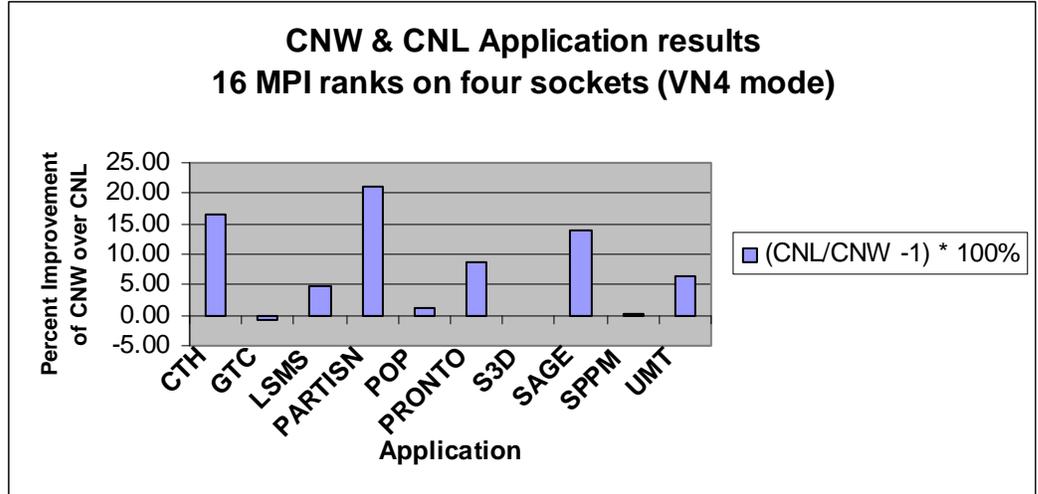
Application Performance	Number MPI Ranks	Cores per Node	CNL	CNW	CNW/CNL
PTRANS	4	1	0.5671	1.606	2.83
HPL	4	1	17.88	17.9	1.00
STREAMS	4	1	25.21	25.84	1.02
Random	4	1	0.006445	0.011823	1.83
FFT	4	1	1.609	1.646	1.02
PTRANS	4	2	0.4878	1.551	3.18
HPL	4	2	17.78	18.03	1.01
STREAMS	4	2	16.45	18.11	1.10
Random	4	2	0.006105	0.011503	1.88
FFT	4	2	1.337	1.36	1.02
PTRANS	4	4	0.2871	1.244	4.33
HPL	4	4	17.59	17.72	1.01
STREAMS	4	4	7.85	9.95	1.27
Random	4	4	0.005984	0.011476	1.92
FFT	4	4	0.902	0.959	1.06

We then worked with eleven different applications. However, VH1 failed on CNL with a message "MEMDPost() failed : PTL_PT_VAL_FAILED". So no CNL results are provided.

In the previous section, we provided a short discussion of each application. This time, we simply provide the tables of results. The first set of data is obtained using all 16 available cores on the four quad-core nodes. Results are in time units, so smaller is better.

Application	Number MPI Ranks	Cores per Node	CNL	CNW	(CNL/CNW -1) * 100%
CTH	16	4	15.131	12.982	16.55
GTC	16	4	664.9	670.6	-0.85
LSMS	16	4	290.1	276.7	4.84
PARTISN	16	4	43.2	35.7	21.01
POP	16	4	153.78	151.93	1.22
PRONTO	16	4	241.5	222	8.78
S3D	16	4	1949.1	1948.9	0.01
SAGE	16	4	267.83	234.94	14.00
SPPM	16	4	847.8	845	0.33
UMT	16	4	8.379	7.872	6.44

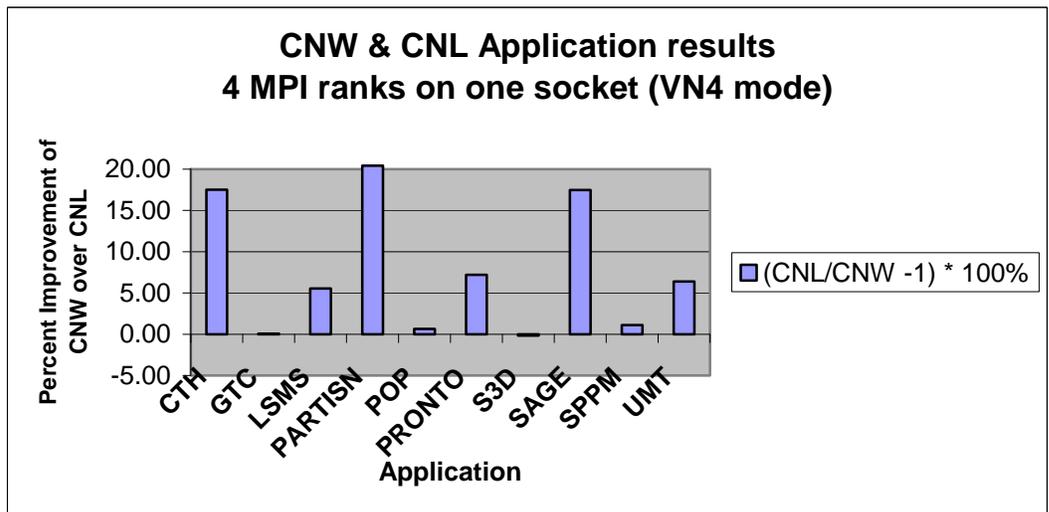
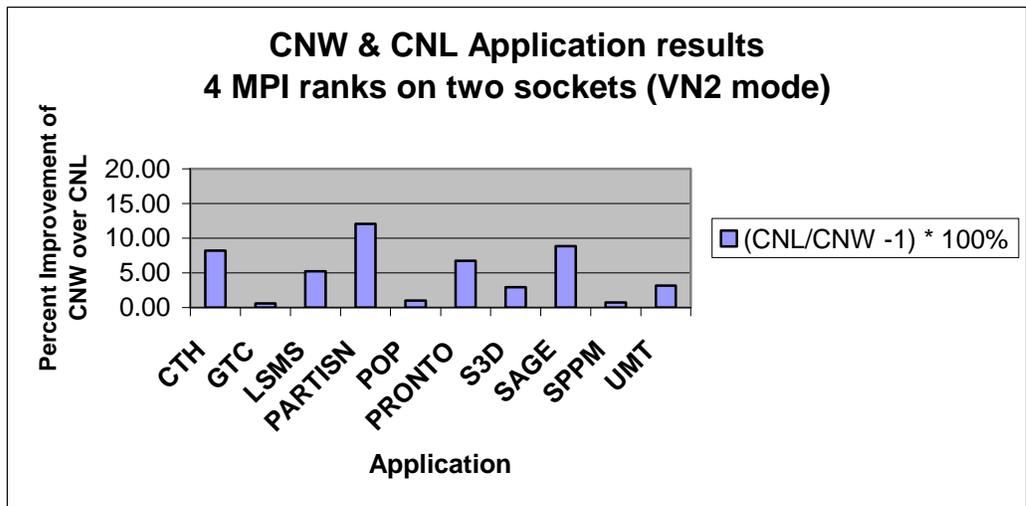
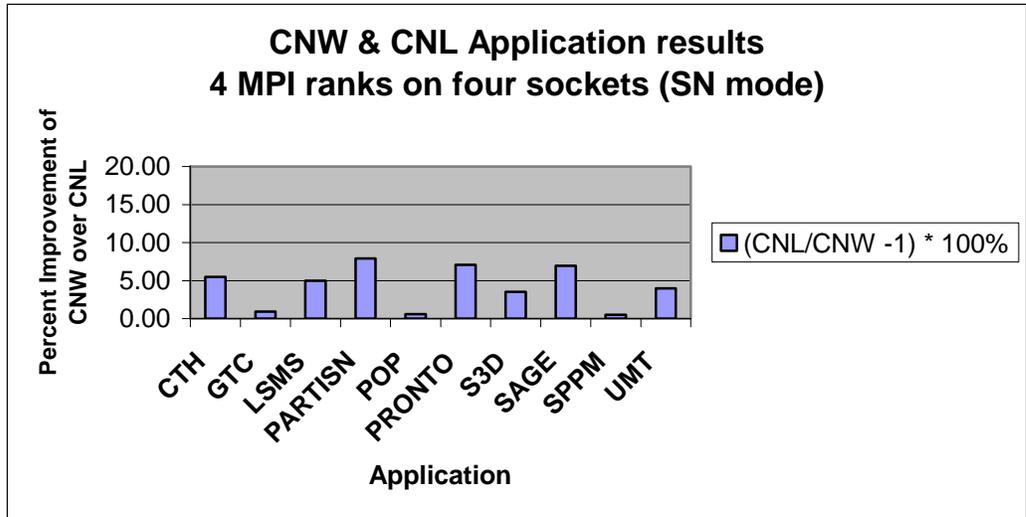
The average increase in CNW performance is approximately 7% for all applications, but only 1.3% for the applications of interest to ORNL (GTC, LSMS, POP and S3D). The following figure shows the percent improvement in graphical form:



We now look at performance on the four sockets, taking into account the number of cores used per socket. CNW appears to have a slightly greater advantage when all four cores are used. Possible reasons are 1) 2MB pages on CNW versus 4KB pages on CNL or 2) the differences (e.g. locking algorithms) in intra-node message passing. In the following table, the results are provided in units of time, so smaller numbers are better.

Application	Number MPI Ranks	Cores per Node	CNL	CNW	(CNL/CNW -1) * 100%
CTH	4	1	8.614	8.167	5.47
GTC	4	1	583.1	577.7	0.93
LSMS	4	1	1160.6	1105.6	4.97
PARTISN	4	1	64	59.3	7.93
POP	4	1	428.04	425.46	0.61
PRONTO	4	1	175.8	164.2	7.06
S3D	4	1	1327.8	1282.5	3.53
SAGE	4	1	169.984	158.949	6.94
SPPM	4	1	294.6	293.1	0.51
UMT	4	1	29.48	28.35	3.99
VH1	4	1		104.3	
CTH	4	2	9.497	8.778	8.19
GTC	4	2	592.9	589.5	0.58
LSMS	4	2	1177.3	1118.6	5.25
PARTISN	4	2	91	81.2	12.07
POP	4	2	440.07	435.67	1.01
PRONTO	4	2	186.8	175	6.74
S3D	4	2	1482.2	1439.7	2.95
SAGE	4	2	179.895	165.276	8.85
SPPM	4	2	297.3	295.2	0.71
UMT	4	2	30.27	29.34	3.17
VH1	4	2		105.7	
CTH	4	4	12.195	10.378	17.51
GTC	4	4	622.8	622.4	0.06
LSMS	4	4	1208.1	1144.6	5.55
PARTISN	4	4	166.8	138.5	20.43
POP	4	4	467.32	464.27	0.66
PRONTO	4	4	209.1	195.1	7.18
S3D	4	4	1937.3	1940.4	-0.16
SAGE	4	4	223.374	190.15	17.47
SPPM	4	4	301.1	297.8	1.11
UMT	4	4	32.41	30.46	6.40
VH1	4	4		108.6	

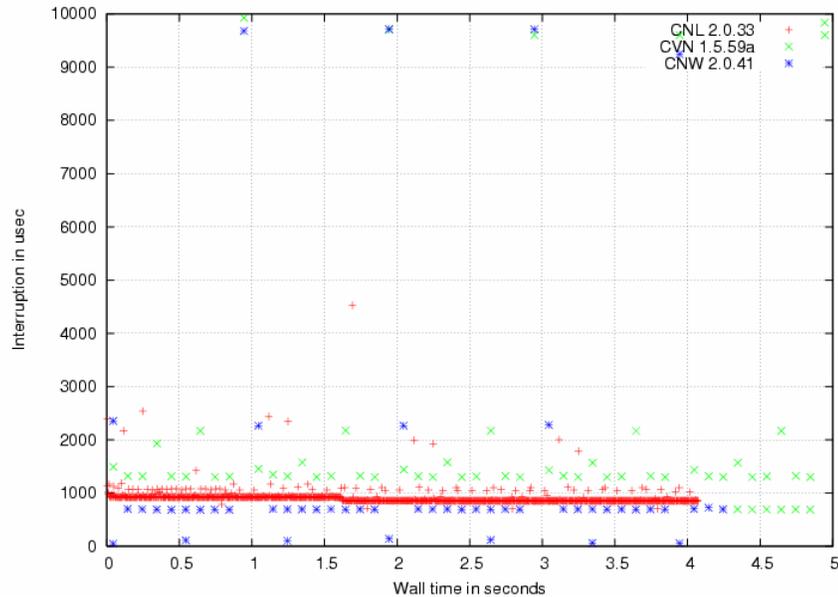
The following graphical views make it apparent that the ORNL applications perform almost the same on CNL and CNW.



4.2. Micro Benchmarks

4.2.1. OS Noise – Selfish

We ran Sandia’s version of the selfish benchmark to compare OS interrupts in CVN, CNW, and CNL. The CVN and CNW results are very similar. CNL interrupts are slightly shorter than CVN interrupts, but there are 250 per second, rather than Catamount’s 10 per second. The 10 microsecond interrupt that occurs once per second in CVN and CNW is the Process Control Thread. This data is reasonable and represents the expected behavior patterns in Catamount and CNL. For completeness, let it be known that one data point was deleted from both the CNL and CNW graphs. Starting with UNICOS/lc releases V2.0 and higher, a long 40 microsecond interrupt is taken 4 seconds after each application starts. Based on our analysis, this is part of BEER initialization. Since it only happens once per job, the data is irrelevant and has been thrown out to preserve the finer scale shown in the graph below. *The y-axis should read nsec, rather than usec.*



4.2.2. Memory – Streams

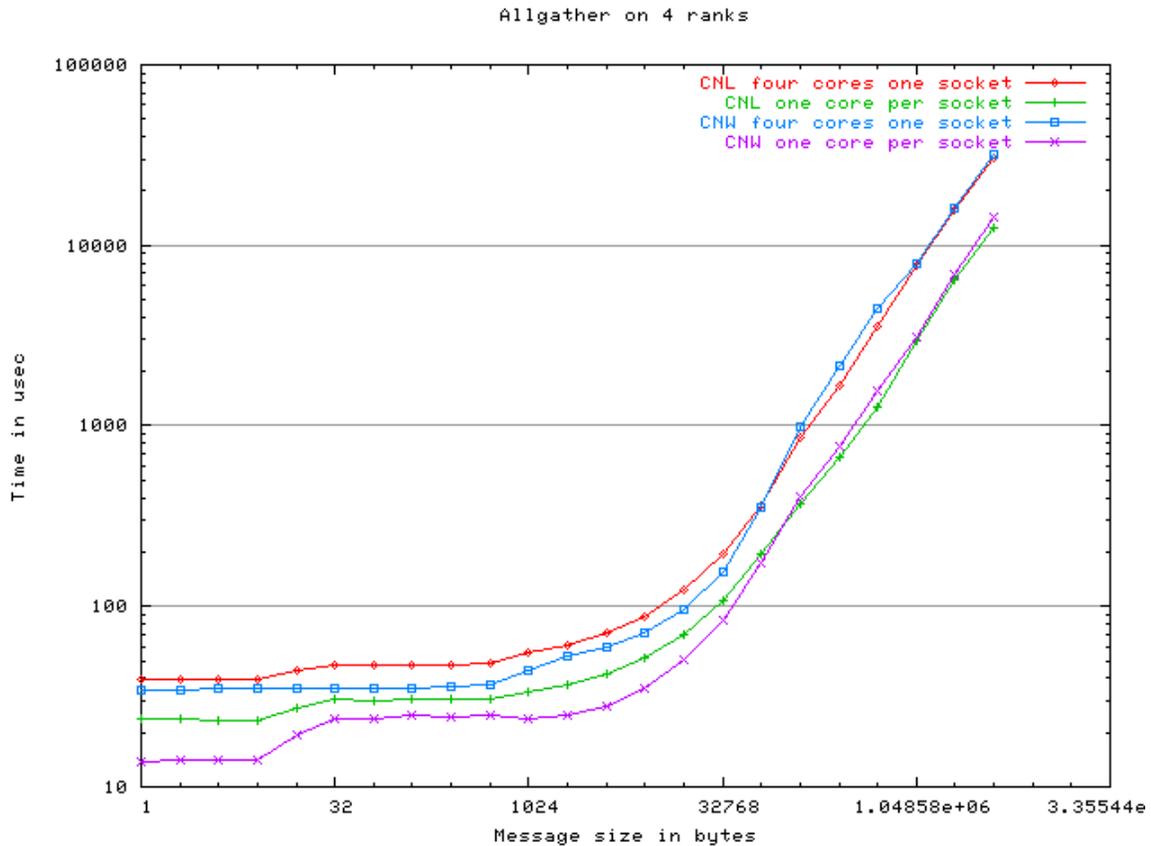
The harshest micro benchmark for a multi-core processor is one that stresses a shared resource. We ran the STREAMS benchmark to confirm the expected poor result for memory bandwidth: STREAMS on 4 cores of one socket achieves 30.7% of the performance obtained when running on one core of four sockets.

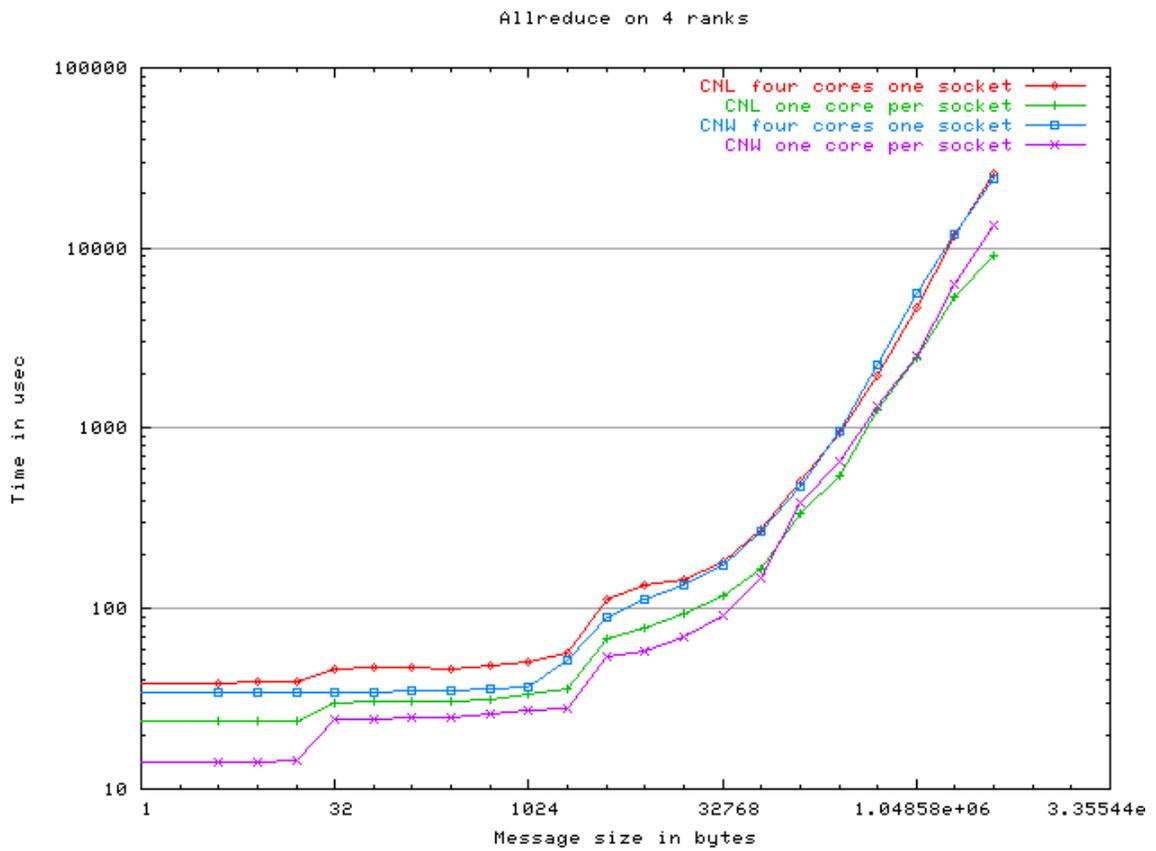
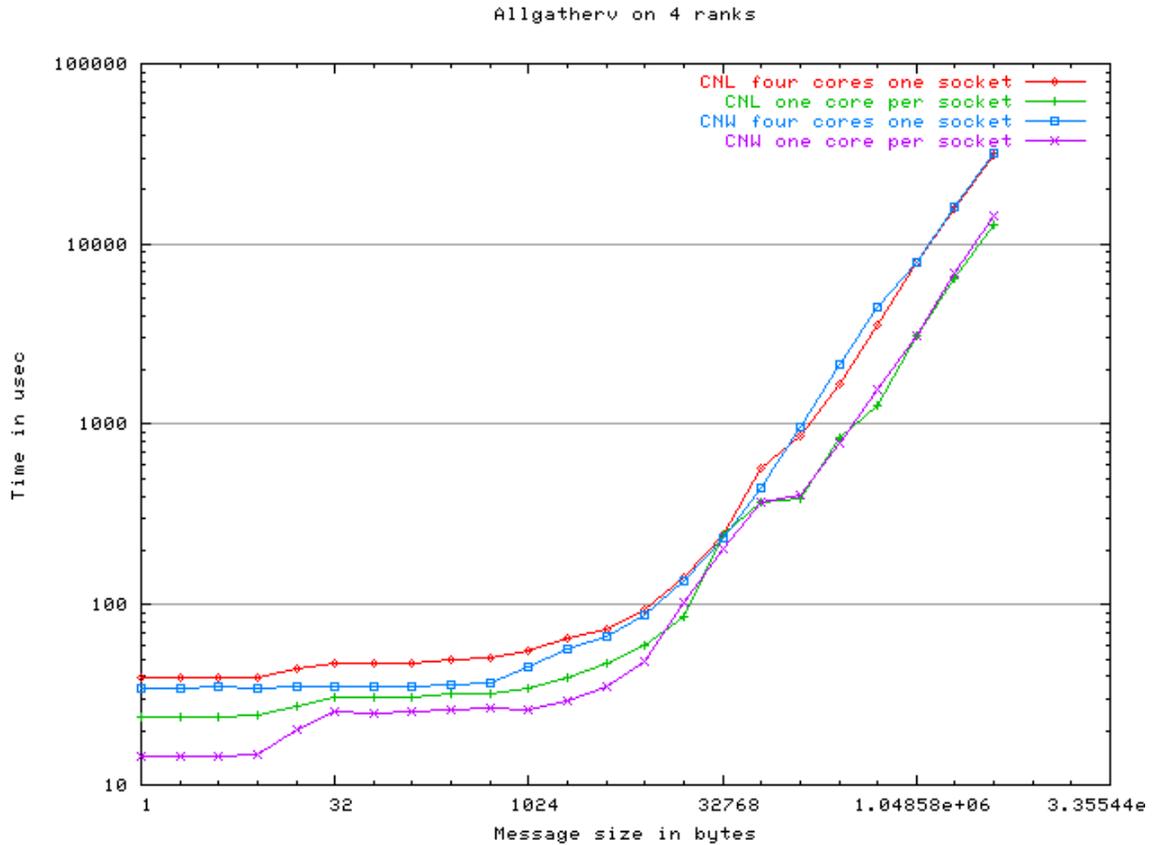
4.2.3. Network – Pallas

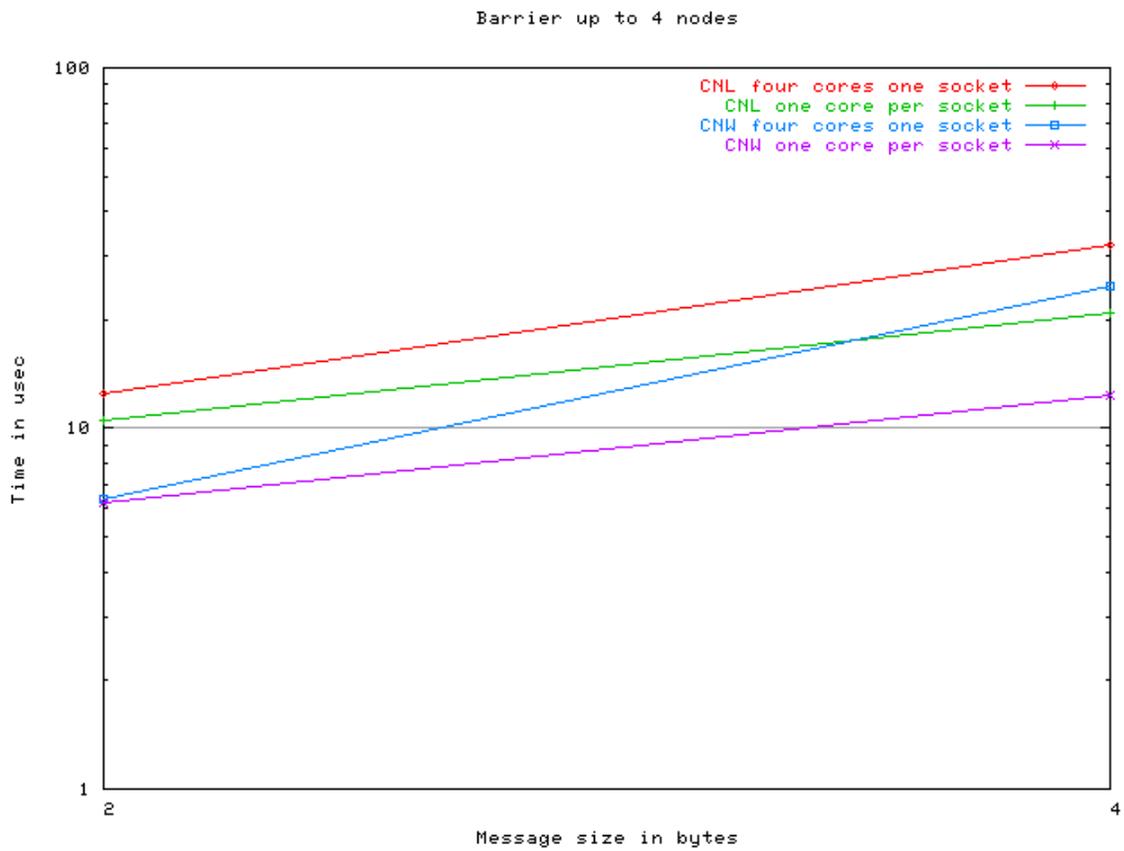
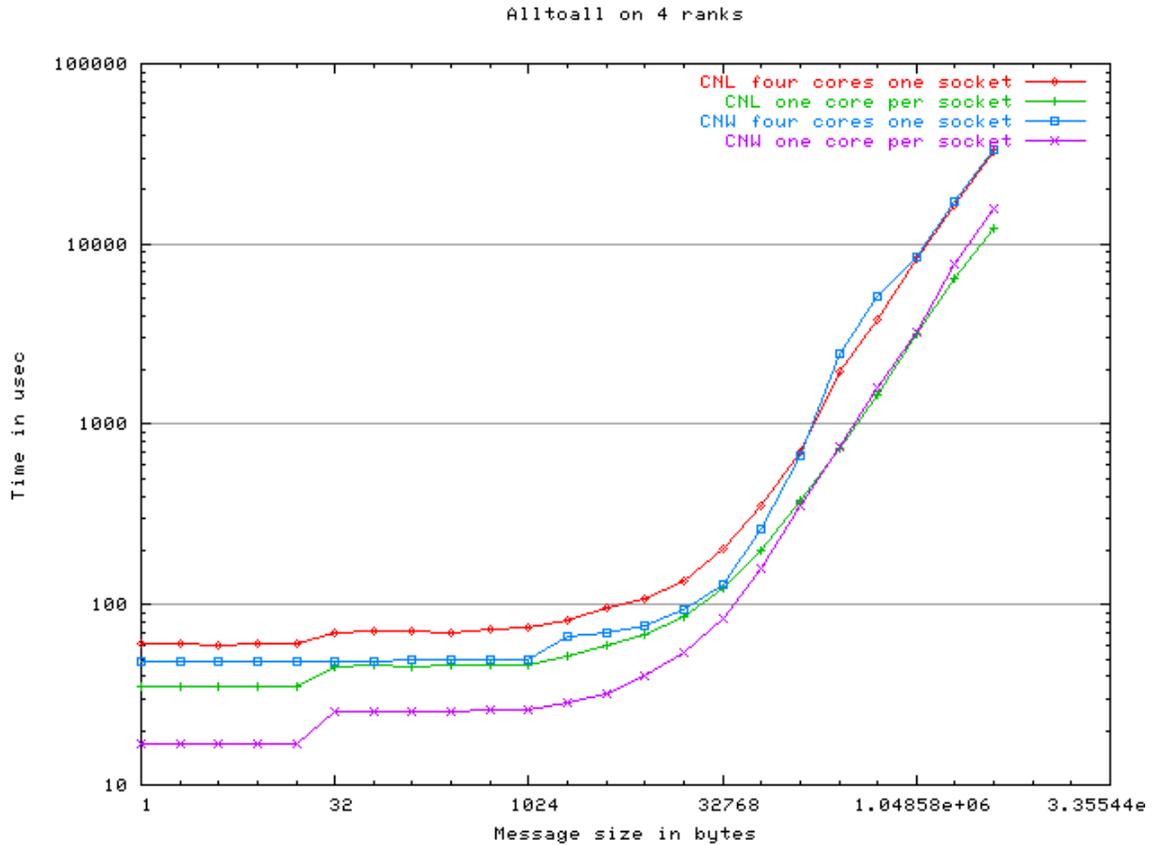
A single SeaStar network interface is shared between the four cores of one Opteron processor. We used V2.2.1 of the Pallas MPI Benchmark to exercise the NIC. The results are as poor as expected. Since we only have four Budapest Opterons, we compared runs using one core per socket with runs using four cores on one socket.

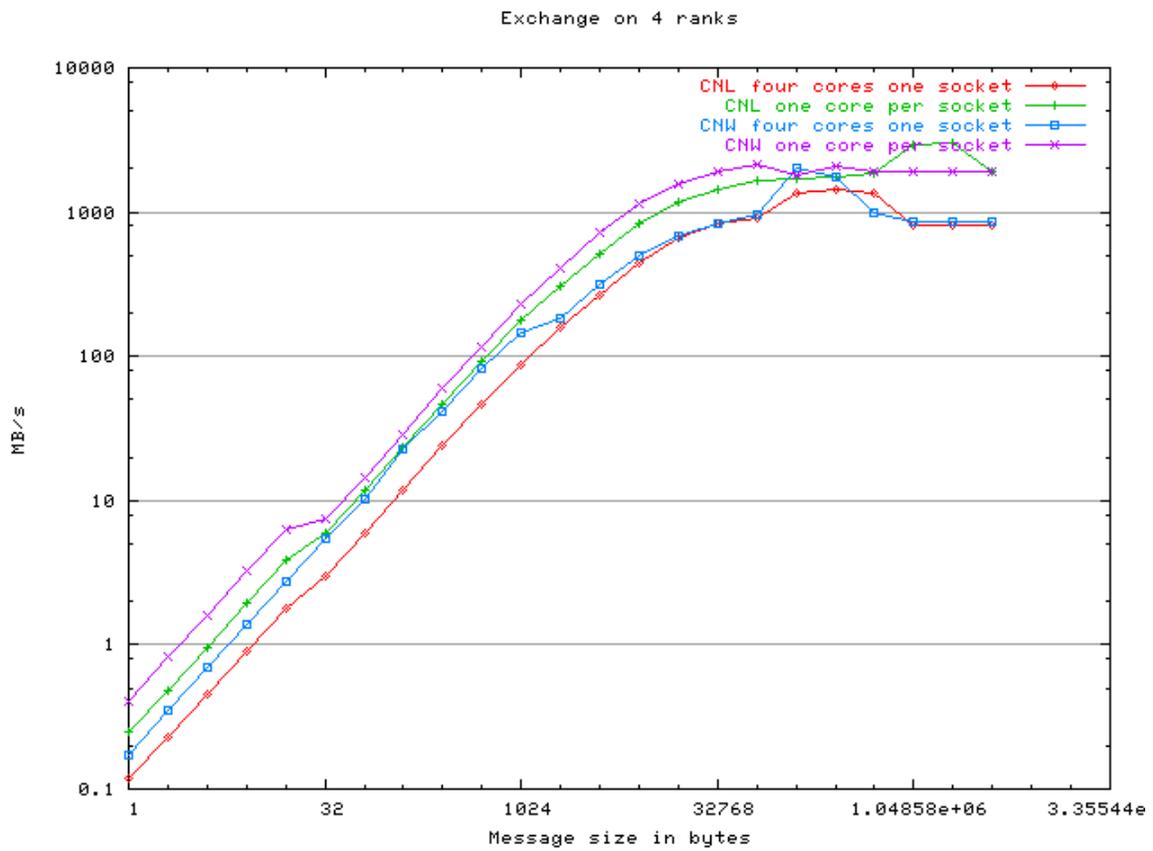
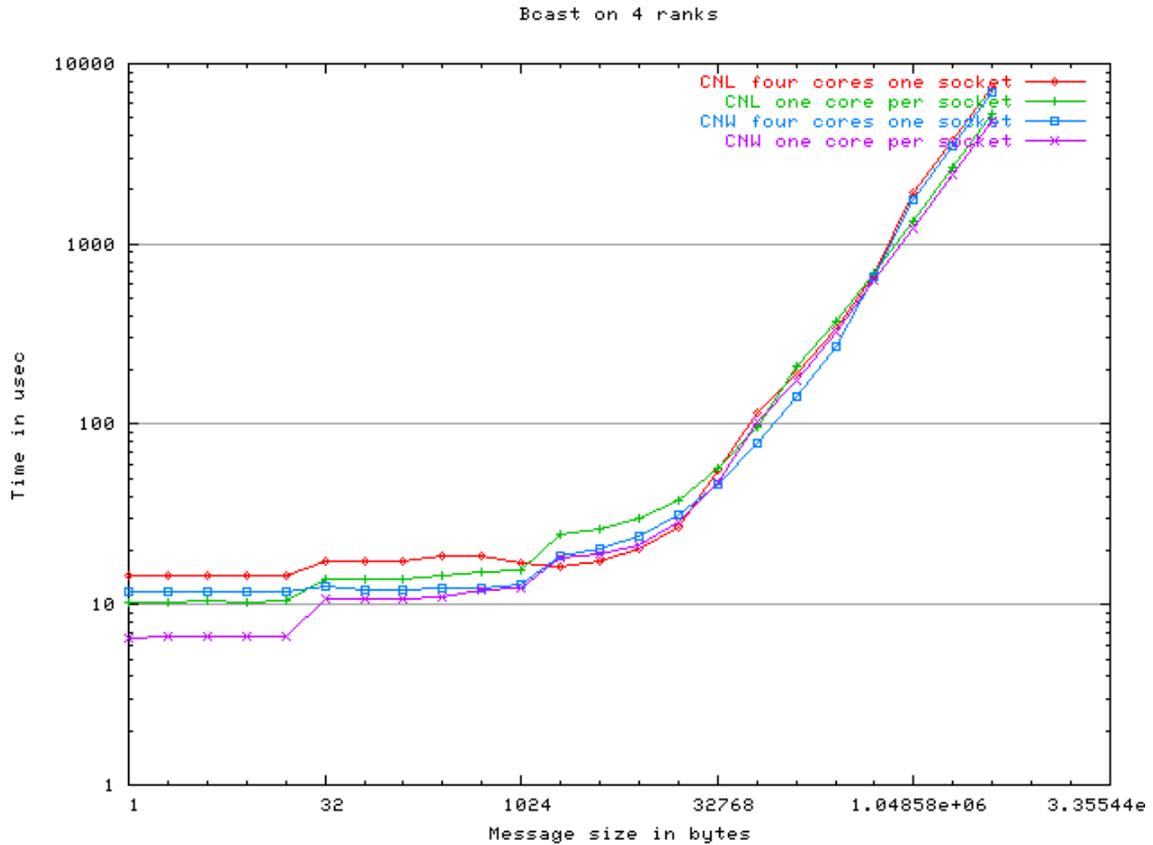
Alltoall operations take 1.5 – 3.4 times longer depending on message size. Allreduce operations take 1.3 to 2.6 times longer, again depending on message size. Allgather operations took 1.6 to 3.2 times longer. The Pallas sendrecv test showed a 50% decrease (on average) in network bandwidth. These results are not particularly insightful since they were run on one socket for the four-core case. There is a Portals optimization that does memory copies for messages less then 512K bytes when the message is destined for a different core on the same physical node. For the cases where it is implemented, any SeaStar hardware sharing will be eliminated. In all cases, there is lock contention, regardless of whether the data passes through the SeaStar or is directly copied in memory.

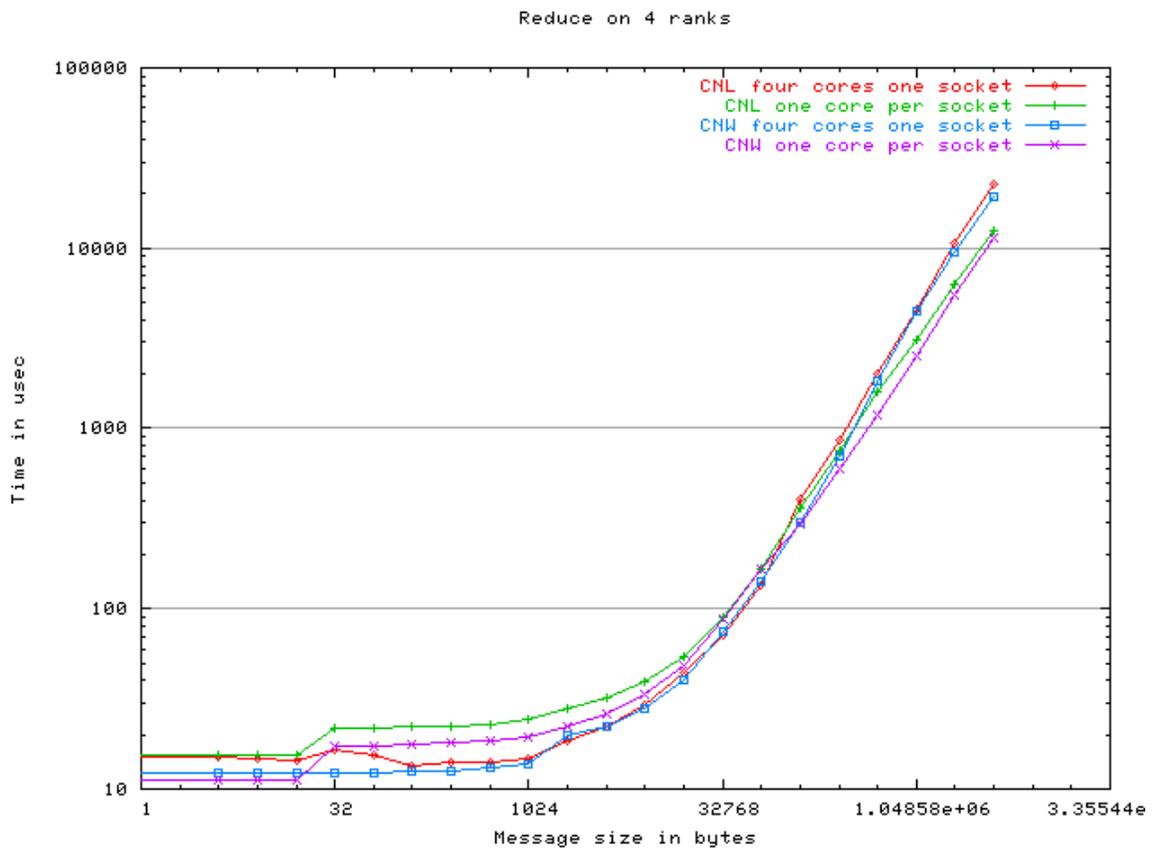
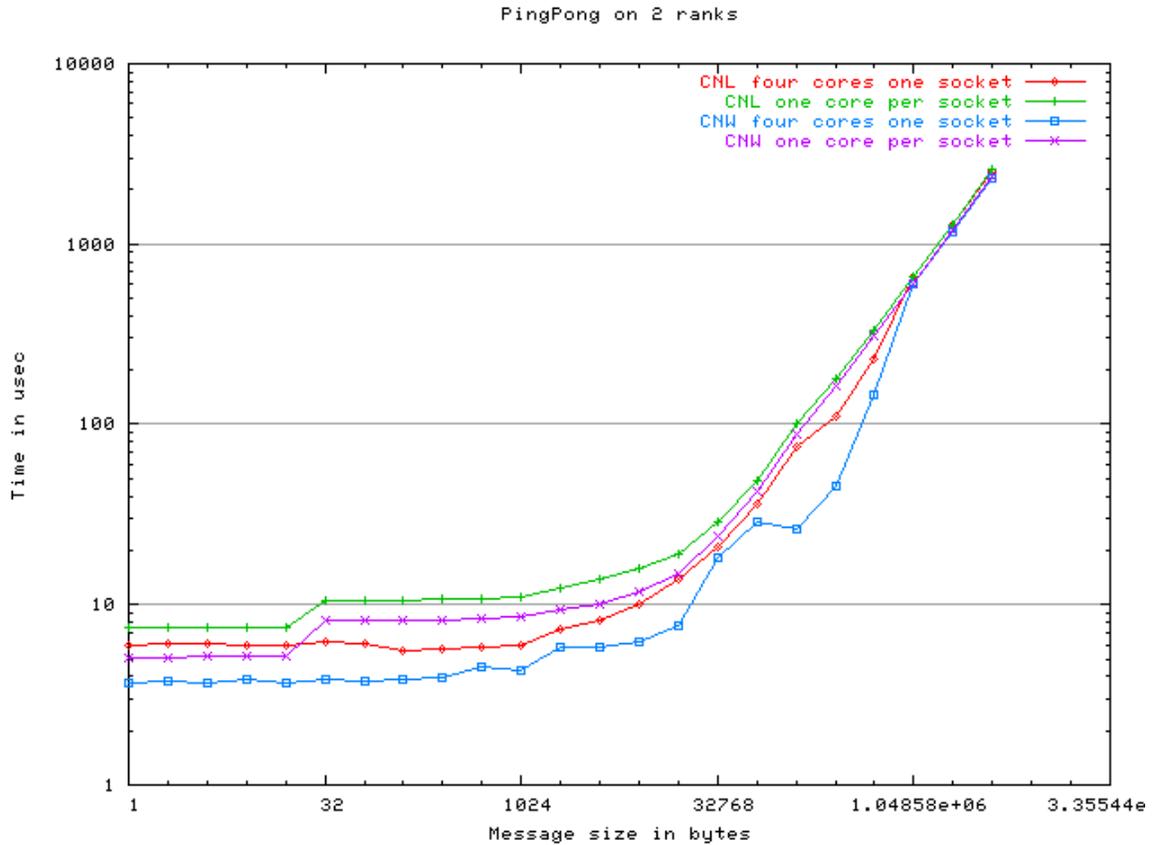
As part of the analysis, we compared CNW Pallas results with CNL results. They are very similar. We had thought that CNL might benefit from the fine grain locking that was implemented for Linux and disabled for Catamount. Upon code review, we found that the compatibility ioctl is being defined for the Linux build. Therefore, the “big kernel lock” is in place, making the fine grain locks only useful during interrupt processing. The interesting result is that the fine-grain locking does add a measurable overhead to latency. The following graphs show the results of all the Pallas tests in graphical form comparing CNW and CNL. Each graph reports CNW using 1) one core on four sockets, 2) CNW using fours cores on one socket, 3) CNL using one core on four sockets, and 4) CNL using four cores on one socket.



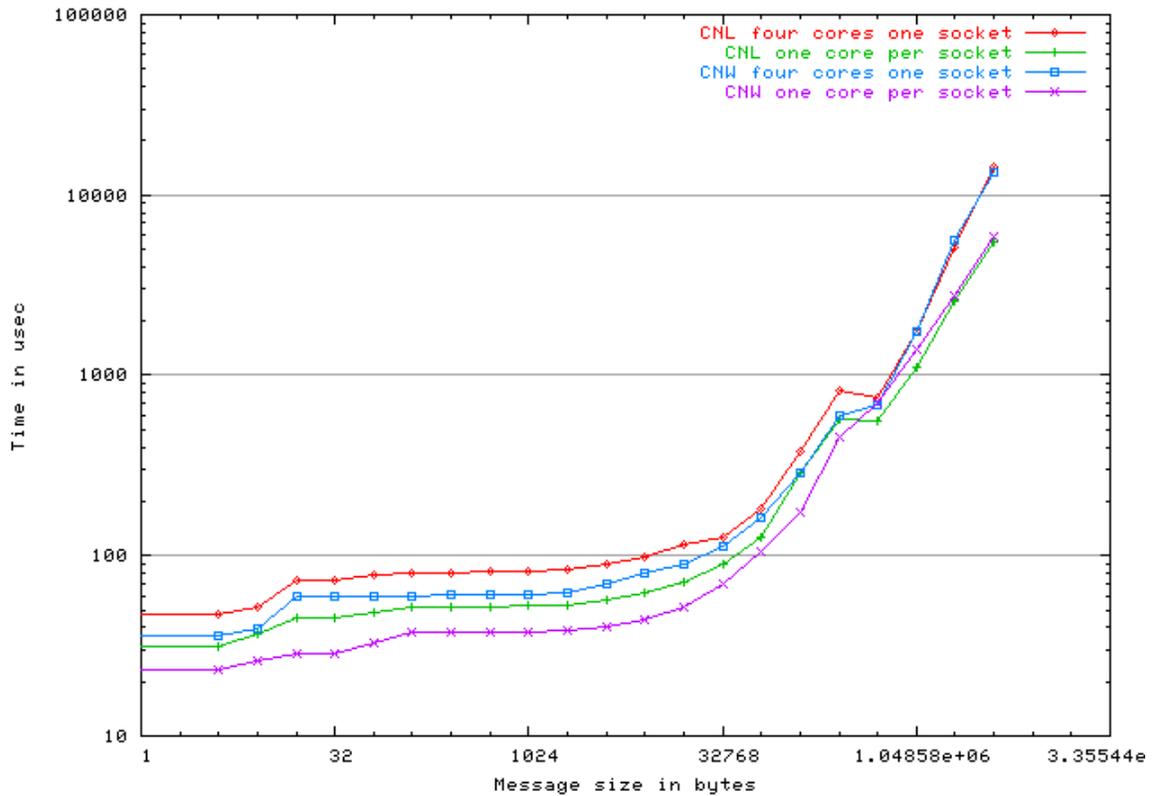




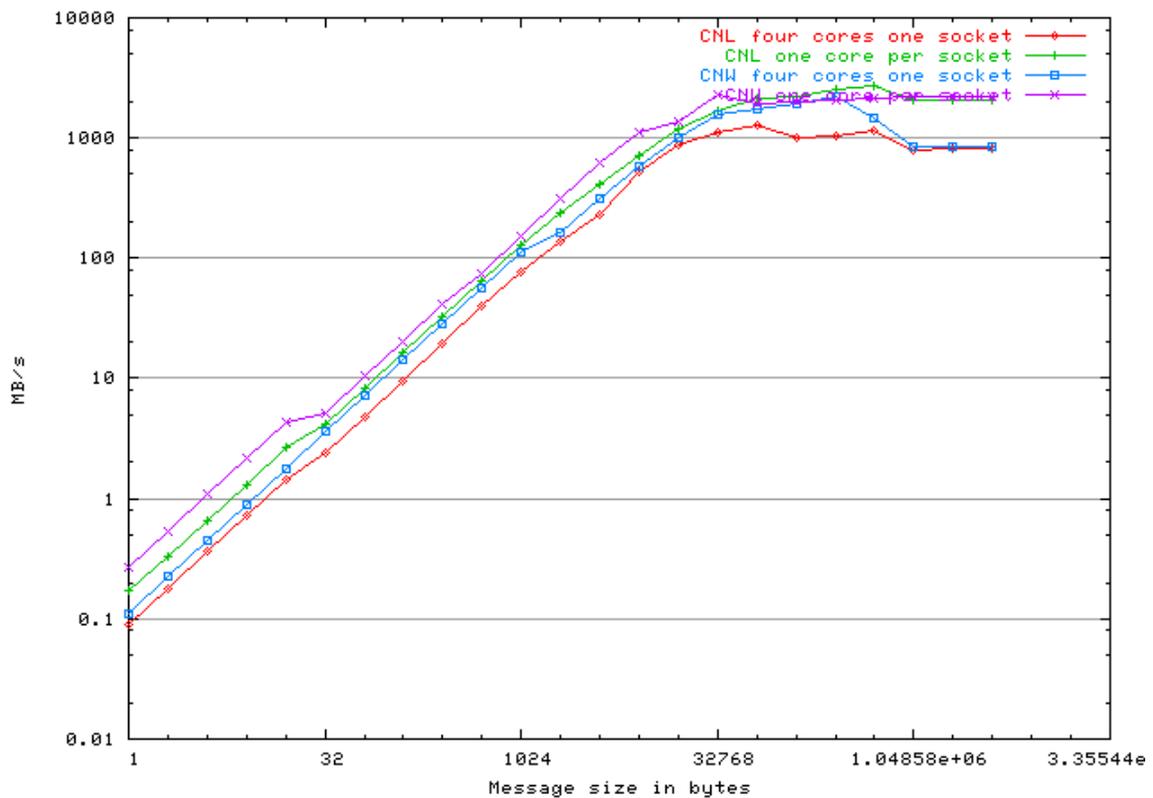




Reduce scatter on 4 ranks

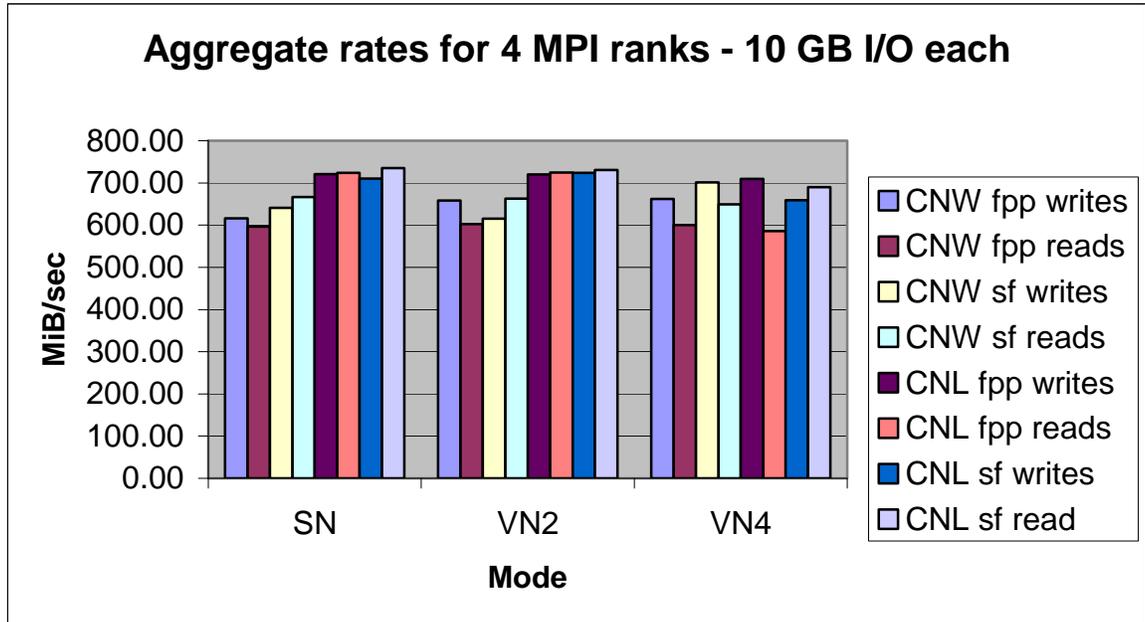


Sendrecv on 4 ranks



4.2.4. File I/O - IOR

IOR is a file I/O benchmark. We ran it specifying 4 MPI ranks using SN mode (one core per node), VN2 mode (two cores per node) and VN4 mode (four cores per node). We ran file per process (fpp) and single shared file (sf) mode. For each test, forty gigabytes total of data was written to a Lustre 2-OSS/4-OST file system and then read back out. File stripe count was 4. Transfer size was 8 MB. The number of cores used per node did not affect the results.



4.3. Unit Testing

We ran a number of Cray’s unit tests on CNW. Where appropriate, we ran them on one core, two cores, and four cores. The catmisc suite exercises a variety of scenarios that had been problematic in the past for Catamount. They look for such things as OS memory leaks and difficult job launch scenarios. The comm tests exercise the network using both MPI and the vanilla Portals protocols. The libc tests ensure that nothing in CNW breaks the supported libc functions. We ran Cray’s versions of the MPICH2 and shmexec test suites. Lastly, we ran the yod interface test suite. There are no unresolved issues or regressions from CVN.

Test Name	SN	VN2	VN4
Catmisc.malloc1	✓	(*)	(*)
Catmisc.badmath1	✓	(*)	(*)
Catmisc.memleak1	✓	(*)	(*)
Catmisc.loadstudy	✓	(*)	(*)
Catmisc.sow611	✓	(*)	(*)
Comm.comtest.cat	✓	✓	✓
Comm..comtest.cat	✓	✓	✓

-loops			
Comm.ptltest.cat	✓	✓	✓
Comm.ptltest.cat-loops	✓	✓	✓
Libc.syscall	✓	(*)	(*)
Libc.gen	✓	(*)	(*)
Libc.locale	✓	(*)	(*)
Libc.stdio	✓	(*)	(*)
Libc.std	✓	(*)	(*)
Libc.string	✓	(*)	(*)
Lustre.IOR (2.0.41)	✓	✓	✓
MPICH2.attr	✓	✓	✓
MPICH2.basic	✓	✓	✓
MPICH2.coll	✓	✓	✓
MPICH2.comm	✓	✓	✓
MPICH2.datatype	✓	✓	✓
MPICH2.errhan	✓	✓	✓
MPICH2.errors	✓	✓	✓
MPICH2.group	✓	✓	✓
MPICH2.info	✓	✓	✓
MPICH2.init	✓	✓	✓
MPICH2.io			
MPICH2.pt2pt	✓	✓	✓
MPICH2.rma	✓	✓	✓
MPICH2.spawn	✓	✓	✓
MPICH2.topo	✓	✓	✓
MPICH2.cxx (as above)	✓	✓	✓
MPICH2.f77	✓	✓	✓
MPICH2.f90	✓	✓	✓
Pallas	✓	✓	✓
Shmem.mpi01	✓	✓	✓
Shmem.sma1	✓	✓	✓
Shmem.sma2	✓	✓	✓
Shmem.smaf	✓	✓	✓
Shmem.smaperf1	✓	✓	✓
Yod.bigexe	✓	(*)	(*)
Yod.checkelf	✓	(*)	(*)
Yod.Checkosversion	✓	(*)	(*)
Yod.cleanup	✓	(*)	(*)
Yod.F	✓	(*)	(*)
Yod.heap	✓	(*)	(*)
Yod.stack	✓	(*)	(*)
Yod.strace	✓	(*)	(*)
Yod.tlimit	✓	(*)	(*)
Yod.YOD_TIME_LIMIT	✓	(*)	(*)
Yod.loadint	✓	(*)	(*)
Yod.prog	✓	(*)	(*)
Yod.progargs	✓	(*)	(*)
Yod.signals.from compute	✓	(*)	(*)
Yod.signals.from service	✓	(*)	(*)

Yod.unixio	✓	(*)	(*)
Yod.validinstall	✓	(*)	(*)

(*)Not applicable – single process tests

5. Accelerated Portals as of March 30, 2008

In this section, we present our test results when using Accelerated Portals, rather than Generic Portals. AP is only implemented for compute node to compute node communication. Therefore, it was not appropriate to run every test in both GP and AP modes. AP had not yet been integrated into CNW when we ran on Jaguar in June, 2007. Therefore, no large scale results are available.

In general, the results contained in this section are disappointing. We had hoped that by offloading Portals protocol processing from the host – and primarily CPU 0, it would improve application run times significantly. This did not turn out to be the case. We offer the following possibilities for why the performance is not more remarkable when compared to the generic, host-based Portals. More study would be needed to understand the impact of each of these ideas.

1. Applications that exchange small messages and are latency sensitive would benefit the most from AP. That is a small number of applications. In 2007, Sandia took dedicated time on Red Storm to run a version of CVN with AP on 2500 nodes. At this scale, Partisn showed a 10% improvement and Pronto showed a 3- 5 percent improvement on 2048 nodes.
2. Applications that overlap computation and communication would benefit the most from AP. It does not help to offload the host processor, if it is idle waiting for a synchronous message from another node.
3. The processor in the SeaStar NIC runs at 500 MHz. In contrast, the host processor used in these tests runs at 2.2 GHz. Protocol processing can therefore be done faster on the host than the NIC if the host is not busy doing application processing.
4. Accelerated Portals does not take advantage of any send-to-self shortcuts using memory copies. Since protocol data structures are stored in the NIC memory for AP, all messages must traverse through the NIC for processing.
5. The intent of AP is to bypass the OS. However, it is critical to have a trusted user id provided in each sent message. To do this, all sends require one trip into the OS to obtain it.
6. Some shared Portals structures remain in host memory. Whenever a CPU core needs to access these structures, it must put a read or write lock on the memory. Like GP, this will force a serialization of messages from each core.
7. AP is relatively new software. It's possible that with future maintenance and enhancement, its performance can be improved. For example, when GP was originally delivered, zero byte MPI latency was 29 usec. It is now ~5 usec.

5.1. Application Testing

5.1.1. Testing on four Quad-Core Budapest Nodes

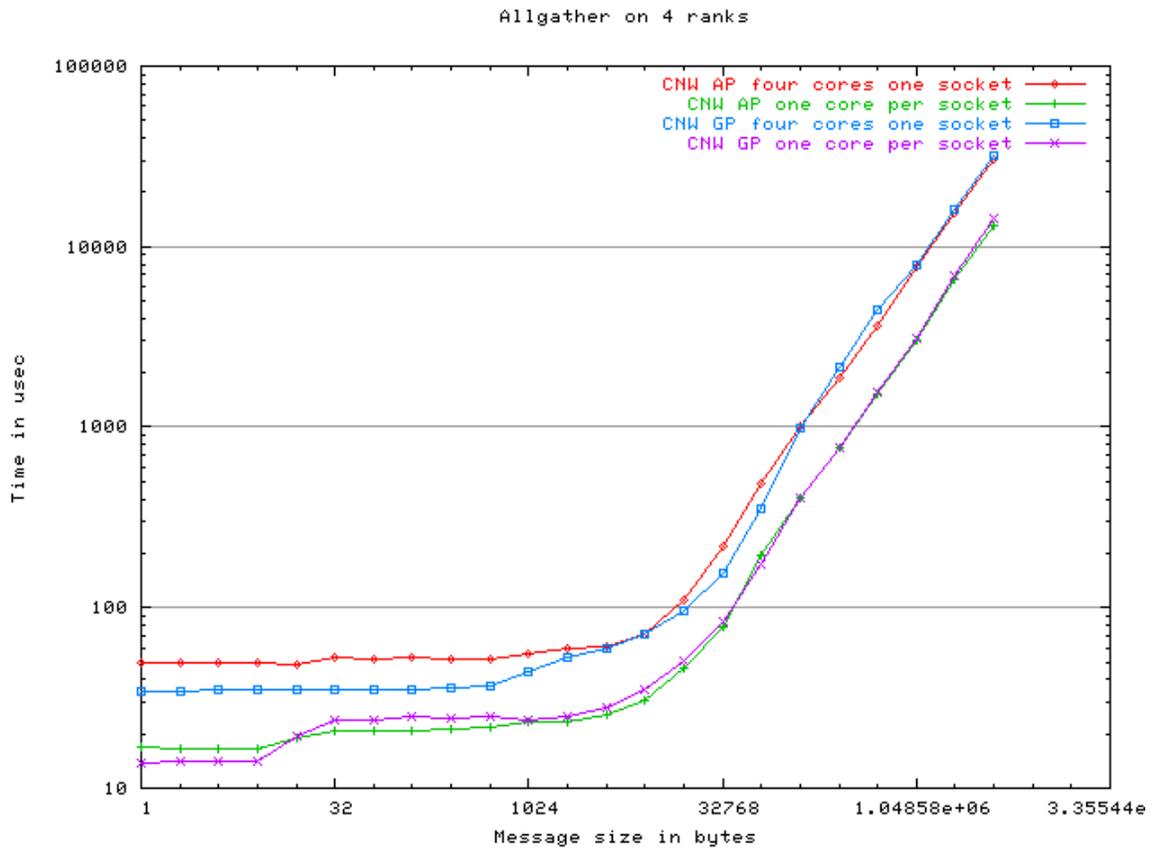
We were able to run all of the applications and HPCC with Accelerated Portals. For HPCC, the results on 16 cores with VN4 mode were encouraging. PTRANS ran 3.4% faster with accelerated, HPL ran 0.4% faster, FFT ran 14% faster, and Random Access ran 43% faster. AP did not make as much of a difference for the applications. We were able to compare 9 of the applications in accelerated mode on 16 cores (the exception was GTC which exposed a system software problem in non-accelerated mode related to Portals send-to-self logic). These showed from a 0.5% decrease in performance to a 1% improvement in performance. On 4 cores (either all in one socket or with 2 or 4 sockets), we see similar results with the spread being a little larger with accelerated being from 1% worse to 3.2% better performance.

5.2. Micro Benchmark

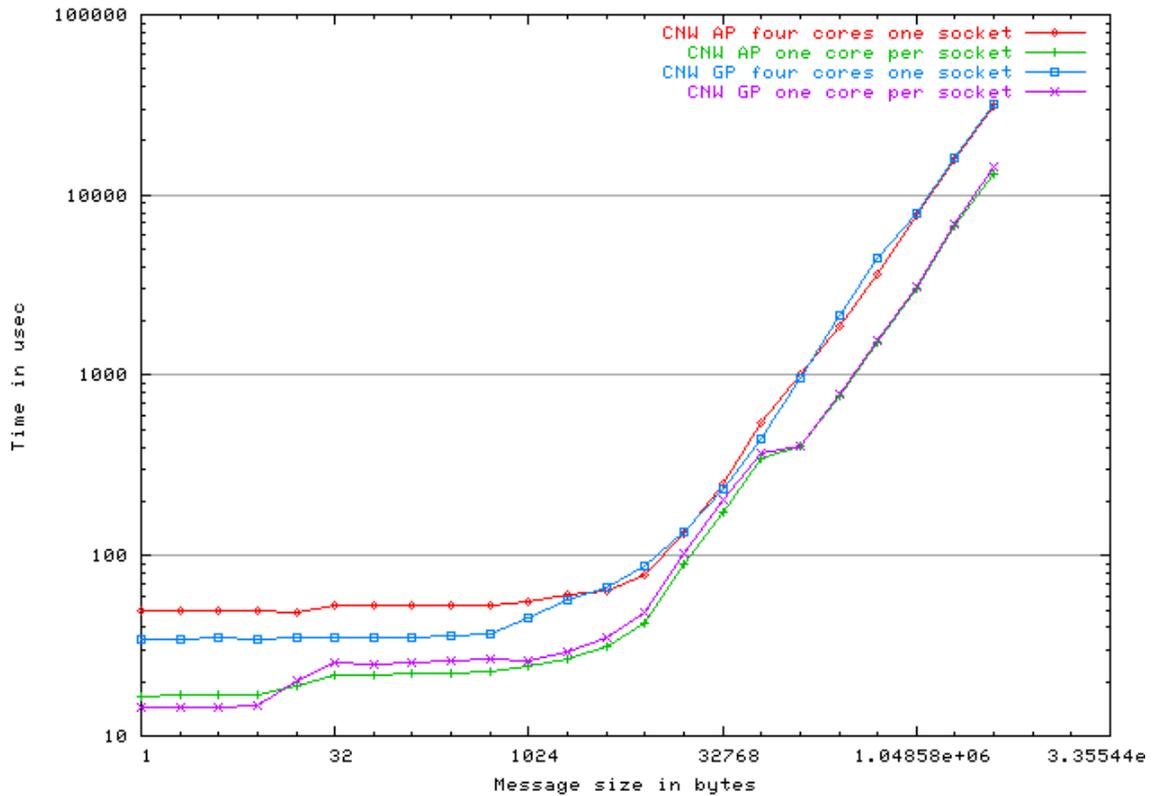
5.2.1. Networking - Pallas

The Pallas results did not show a significant benefit or loss with Accelerated Portals (AP). AP has to suffer a call into the kernel for each message to obtain the user id from a trusted source. Additionally, AP cannot use memory copies for small messages as the necessary protocol data structures are stored on the NIC processor. The following pages show the results of all Pallas tests in graphical form comparing the default (Generic) Portals and Accelerated Portals.

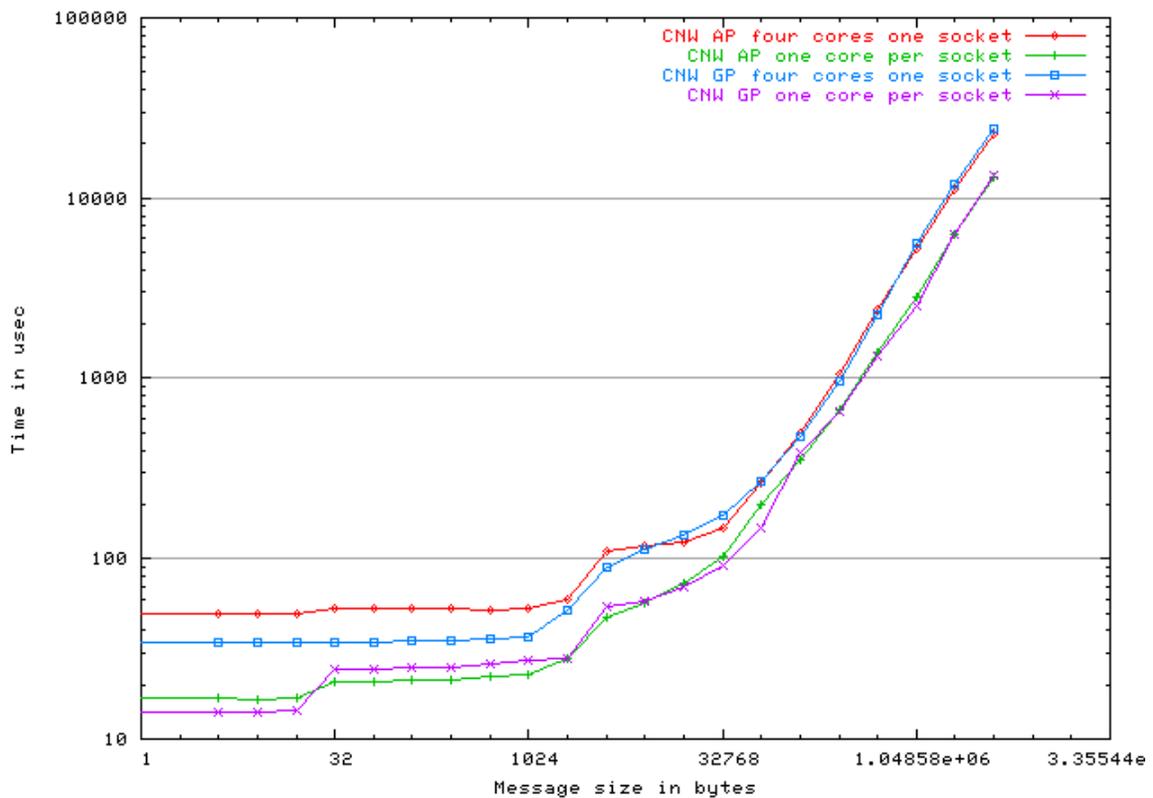
There are times when a tabular list is more easily absorbed than a plethora of graphs. Selected results are provided after the graphs. The tables provide CNL, CNW/GP and CNW/AP results. The table also contains results of 16-rank runs, which used all four cores on all four nodes.



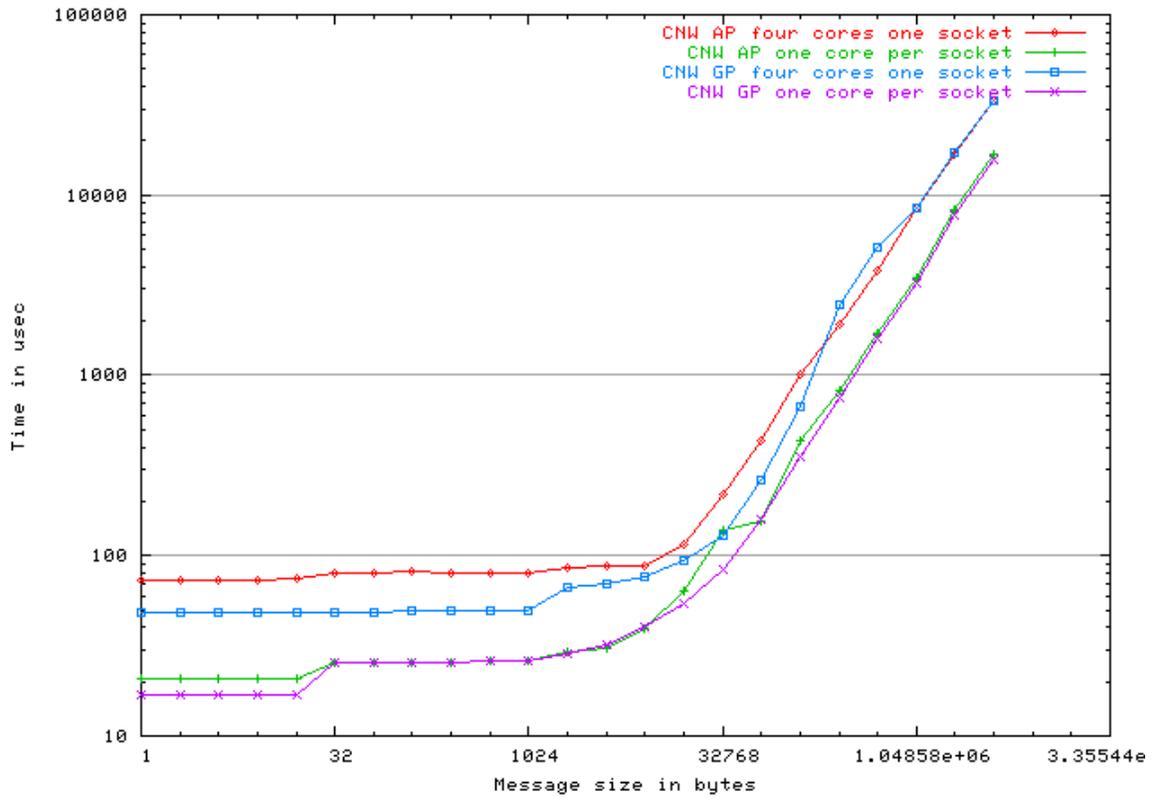
Allgather on 4 ranks



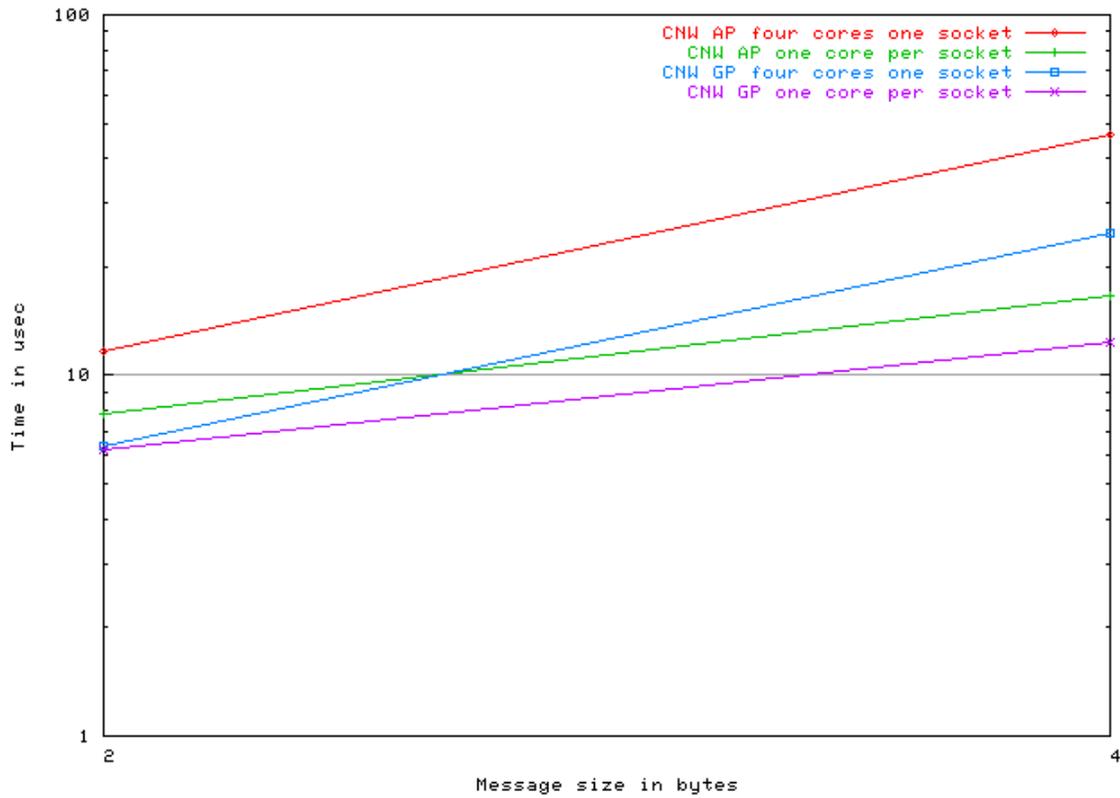
Allreduce on 4 ranks

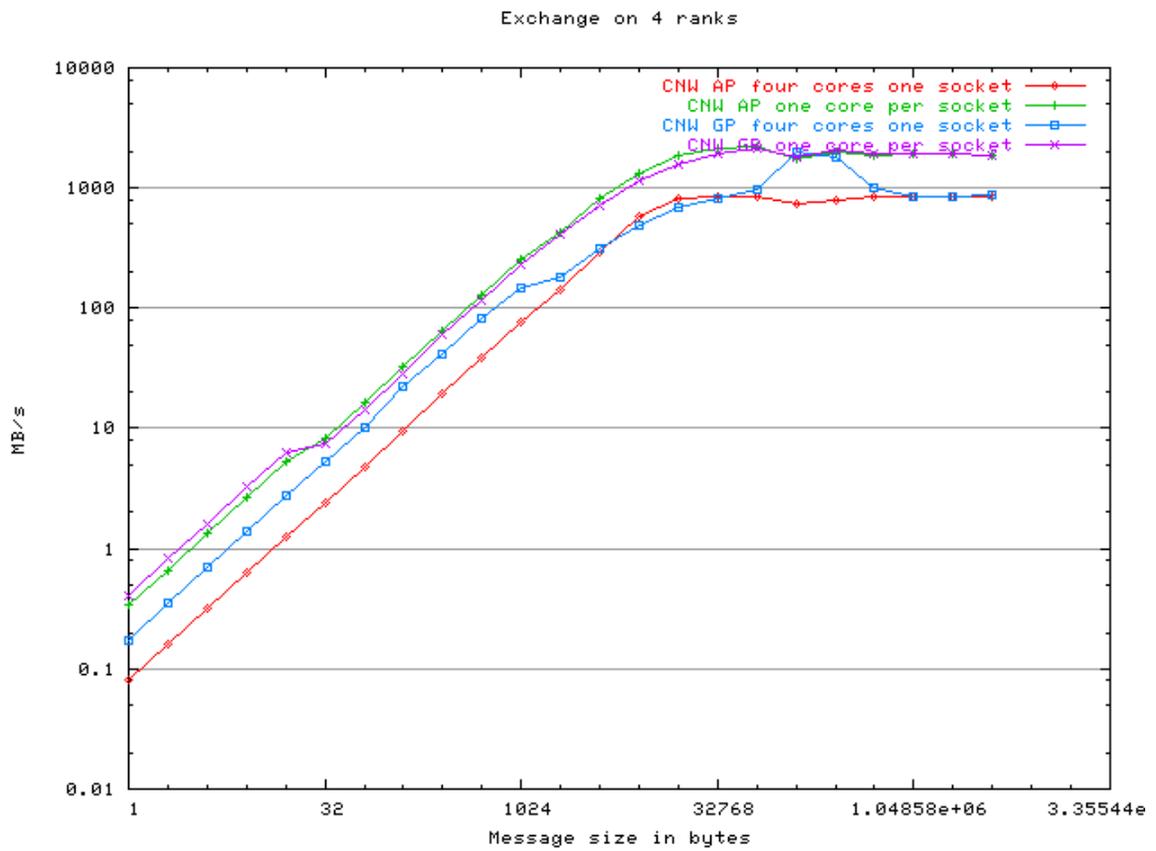
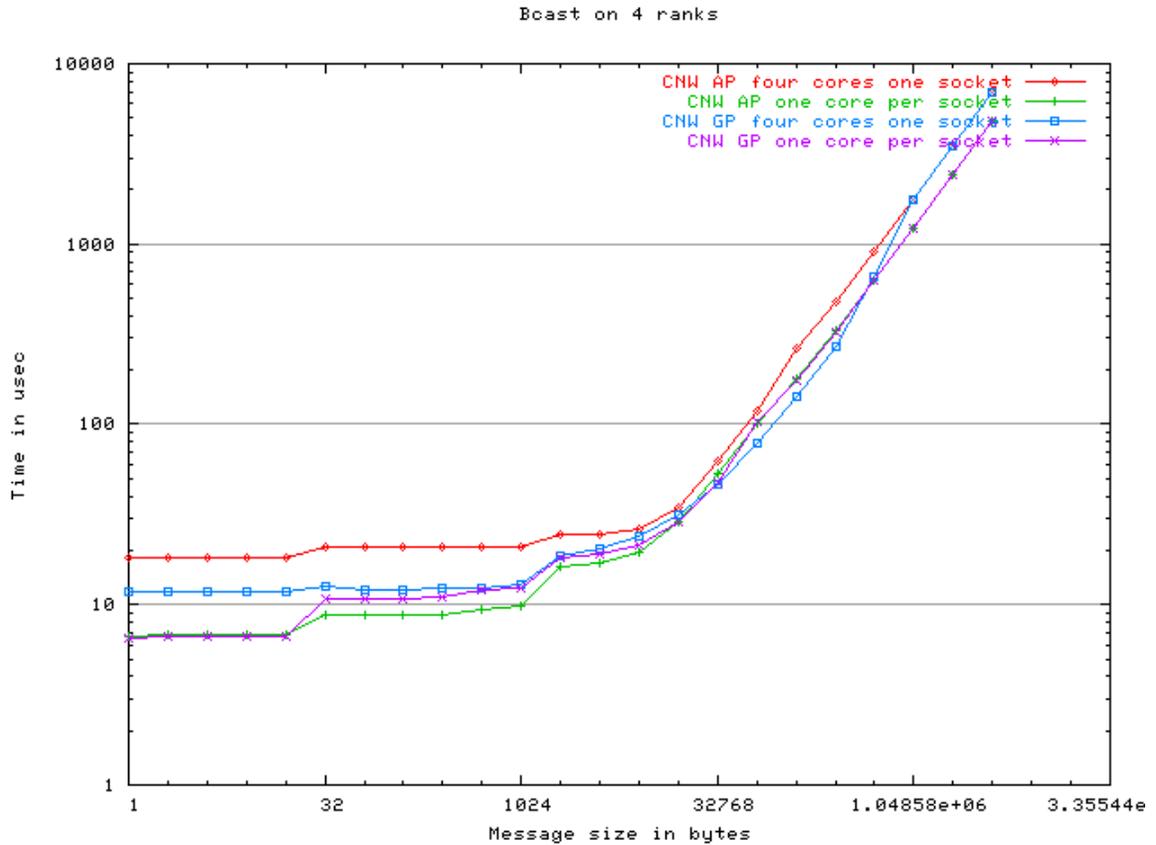


Alltoall on 4 ranks

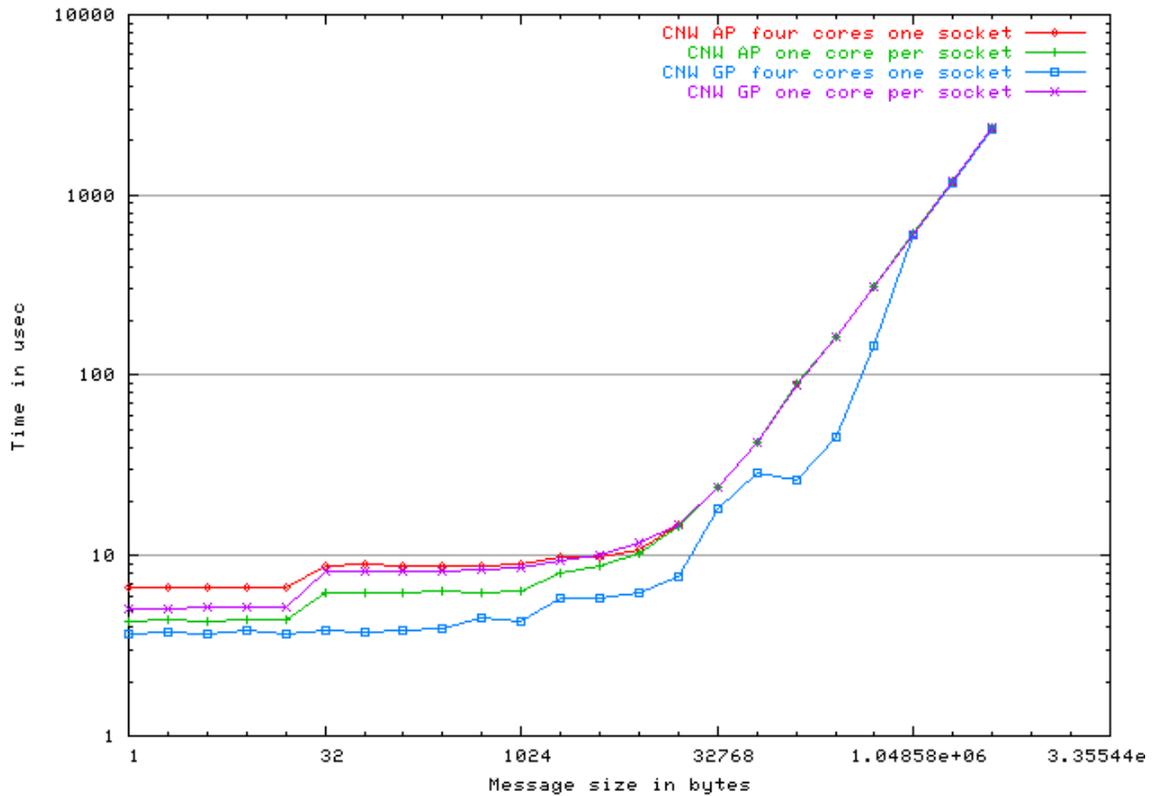


Barrier up to 4 nodes

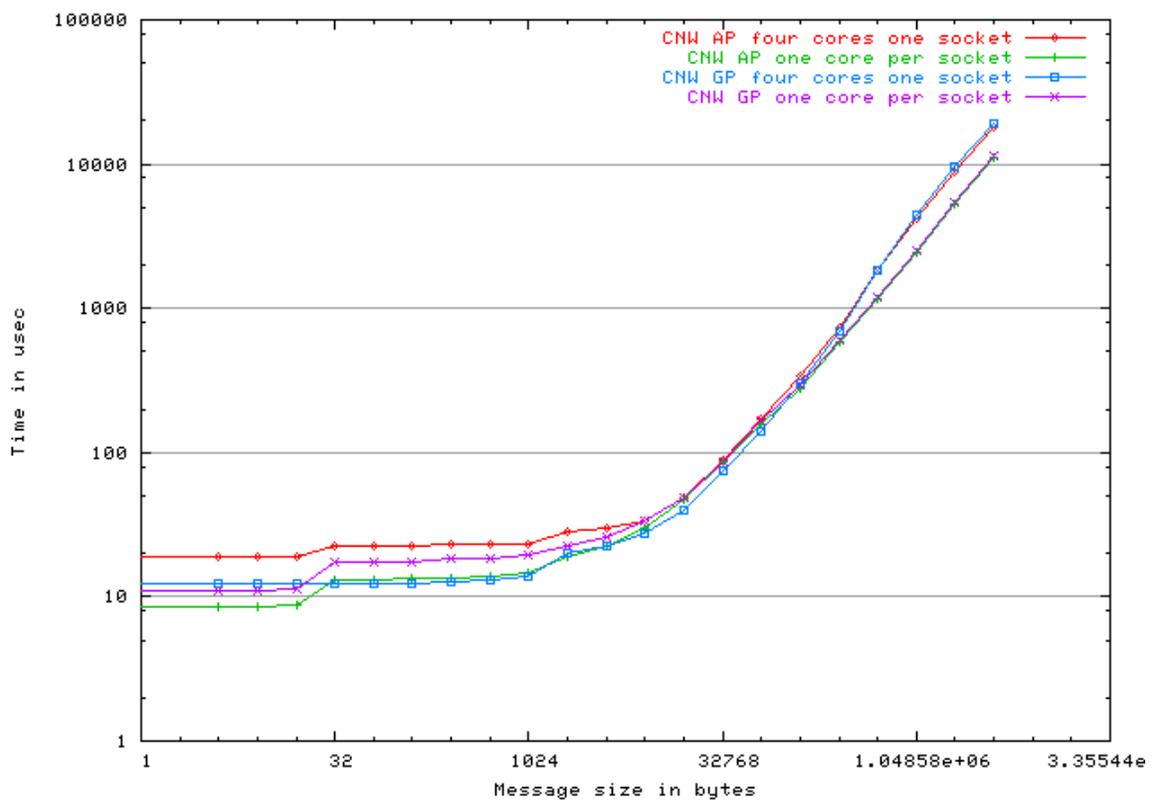




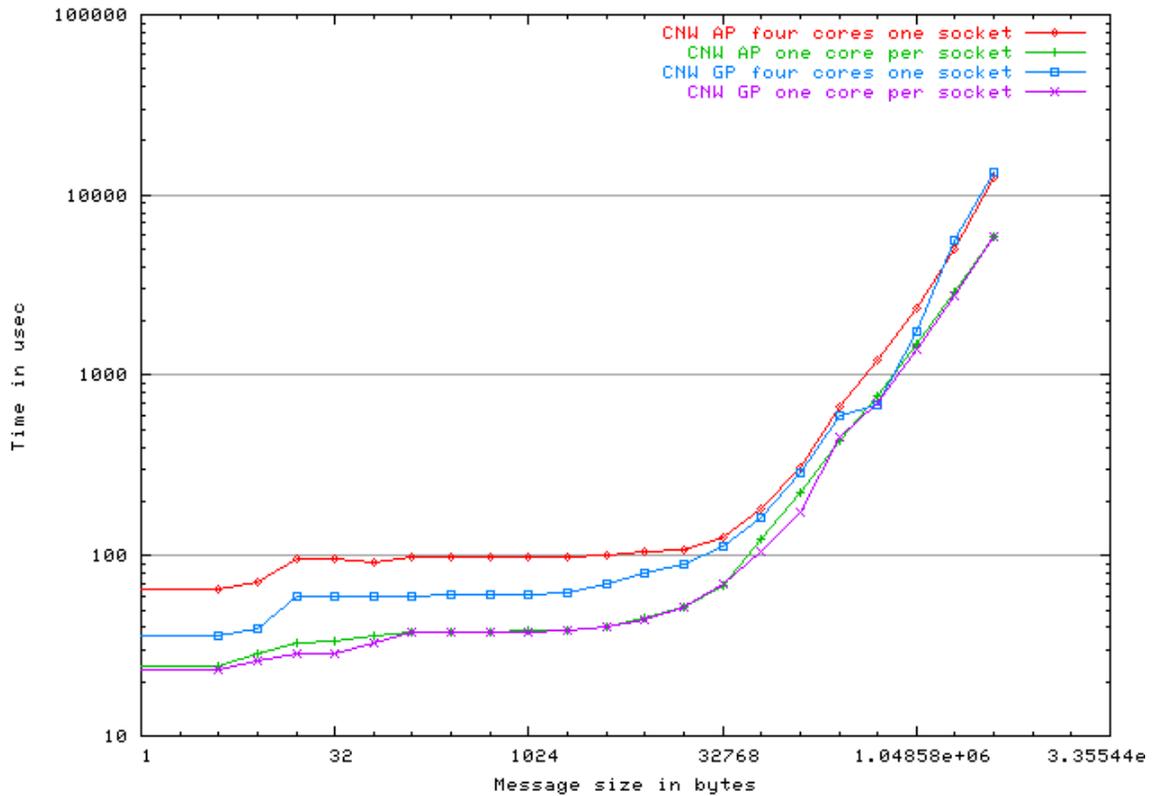
PingPong on 2 ranks



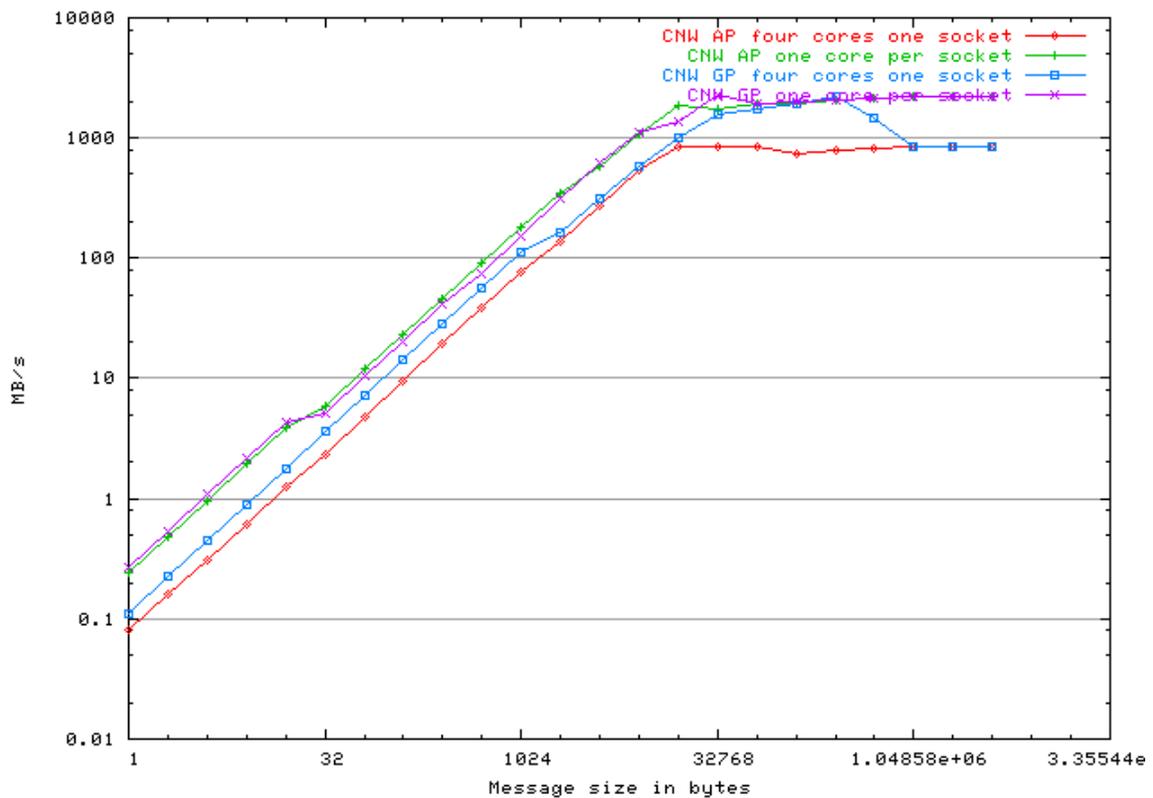
Reduce on 4 ranks



Reduce scatter on 4 ranks



Sendrecv on 4 ranks



Selected Tabular Results from Pallas

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1			CNW	CNW	CNW	CNW	CNW	CNW	CNL	CNL	CNL	CNW	CNW	CNL
2	Test	Msg Size	4 SN	sz4 VN2	sz4 VN4	sz4 SN-AP	sz4 VN2-AP	sz4 VN4-AP	n4 N1	n4 N2	n4 N4	sz16 VN4	sz16 VN4-AP	sz16 N4
3														
4	Barrier		12.41	19.20	25.60	16.54	23.20	46.59	20.95	24.65	32.18	78.37	95.21	85.22
5														
6	PingPong	0	4.82	3.22	3.15	4.27	6.06	6.28	6.91	3.89	3.86	3.25	6.28	3.90
7	t_avg[usec	1	5.18	3.75	3.66	4.37	6.32	6.60	7.48	6.07	5.91	3.72	6.67	5.84
8	lower is be	2	5.17	3.65	3.70	4.38	6.31	6.59	7.43	6.05	6.04	3.74	6.68	5.73
9		4	5.20	3.77	3.79	4.39	6.40	6.59	7.45	6.03	6.08	3.78	6.60	5.85
10		8	5.20	3.67	3.75	4.34	6.39	6.58	7.39	6.03	6.01	3.84	6.61	5.81
11		16	5.26	3.87	3.77	4.43	6.34	6.68	7.44	6.06	5.92	3.85	6.69	6.06
12		32	8.13	3.78	3.76	6.25	8.44	8.87	10.47	6.21	6.28	3.88	8.88	6.33
13		64	8.26	3.80	3.86	6.25	8.42	8.86	10.58	6.34	6.12	3.86	8.87	5.99
14		128	8.15	3.92	3.91	6.22	8.48	8.83	10.57	5.63	5.61	4.00	8.85	5.48
15		256	8.40	3.97	3.96	6.28	8.43	8.87	10.67	5.64	5.63	4.09	8.89	5.47
16		512	8.42	4.22	4.34	6.28	8.45	8.87	10.79	5.78	5.75	4.26	8.87	5.73
17		1024	8.56	4.38	4.41	6.32	8.45	8.87	11.09	6.03	6.01	4.32	8.88	5.97
18		2048	9.45	5.80	5.75	7.99	9.38	9.91	12.25	7.24	7.32	5.77	10.05	7.34
19		4096	10.21	5.83	5.88	8.70	9.91	10.04	13.94	8.13	8.27	5.86	10.16	8.26
20		8192	11.75	6.15	6.41	10.25	10.64	10.79	15.72	9.97	10.05	6.12	11.01	10.10
21		16384	14.81	7.34	9.64	14.55	14.90	14.87	19.18	13.63	13.70	7.46	15.11	13.76
22		32768	23.93	17.98	17.88	23.81	23.88	23.88	28.91	21.06	21.03	17.43	24.12	20.63
23		65536	42.48	28.49	28.97	42.32	41.88	41.89	48.24	33.49	35.89	28.72	42.11	34.33
24		131072	88.37	26.28	26.31	89.99	89.45	90.23	102.25	74.60	74.34	26.13	90.16	73.13
25		262144	162.75	45.18	45.58	164.40	161.57	162.31	179.25	111.08	110.25	45.33	162.37	110.07
26		524288	311.02	146.34	146.68	312.89	305.62	306.55	334.74	230.24	231.84	146.62	306.80	231.43
27		1048576	608.48	595.90	595.76	610.94	593.99	594.98	653.41	652.85	652.59	595.95	595.40	652.06
28		2097152	1202.70	1172.55	1172.30	1203.92	1170.13	1171.12	1292.32	1284.47	1282.02	1172.35	1172.24	1287.80
29		4194304	2392.04	2334.05	2333.65	2393.15	2329.55	2330.74	2583.49	2560.56	2561.15	2333.61	2334.05	2561.10

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1			CNW	CNW	CNW	CNW	CNW	CNW	CNL	CNL	CNL	CNW	CNW	CNL
2	Test	Msg Size	4 SN	sz4 VN2	sz4 VN4	sz4 SN-AP	sz4 VN2-AP	sz4 VN4-AP	n4 N1	n4 N2	n4 N4	sz16 VN4	sz16 VN4-AP	sz16 N4
3														
30	Alltoall	0	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
31	t_avg[usec	1	17.02	30.19	48.12	20.60	36.34	73.54	35.02	45.93	60.19	105.89	98.68	110.13
32	lower is be	2	16.95	30.15	48.02	20.61	36.44	73.54	35.01	45.86	60.26	106.88	99.09	110.79
33		4	16.97	30.17	48.57	20.62	36.46	73.54	35.14	45.81	60.10	131.08	111.09	143.30
34		8	16.97	30.28	48.40	20.58	36.35	73.55	35.04	45.83	60.39	130.78	111.20	143.31
35		16	16.93	30.68	48.29	20.74	36.60	74.00	35.12	46.23	60.62	131.35	111.43	144.04
36		32	25.80	38.57	48.30	25.47	41.45	80.81	45.53	57.71	70.20	132.84	111.92	145.54
37		64	25.83	38.39	47.99	25.46	41.43	80.84	45.84	58.21	70.60	134.72	111.96	147.65
38		128	25.80	38.64	48.77	25.48	41.53	80.96	45.42	58.44	71.14	137.50	112.30	152.32
39		256	25.87	38.74	49.27	25.55	41.66	80.74	46.12	60.89	70.53	162.03	125.92	174.39
40		512	26.00	39.89	49.58	25.69	41.66	80.77	45.99	61.54	72.56	171.88	131.73	206.45
41		1024	26.25	40.60	50.03	25.91	42.04	80.79	46.68	63.06	75.37	432.66	471.51	573.03
42		2048	28.81	47.33	66.77	28.98	47.05	86.52	52.30	71.76	82.63	527.53	537.23	654.48
43		4096	32.54	50.98	71.17	30.54	50.16	87.74	59.32	78.29	95.72	577.63	576.25	750.13
44		8192	42.00	55.32	79.41	39.30	58.61	87.26	67.50	88.25	107.74	745.51	661.94	888.48
45		16384	56.66	72.50	99.18	53.44	94.09	115.01	85.62	115.89	135.95	1135.11	1076.05	1235.53
46		32768	85.02	115.67	135.24	135.13	175.99	220.03	122.43	188.04	203.24	1950.40	2118.25	1869.81
47		65536	159.70	287.36	281.80	155.71	313.87	435.90	197.90	335.76	349.64	2965.65	2553.44	3508.41
48		131072	364.66	666.57	688.24	432.29	693.89	1022.63	381.59	709.10	697.22	7650.95	5937.47	7345.85
49		262144	728.12	1342.28	2464.10	806.83	1420.98	1911.39	729.18	1389.52	1962.10	15973.89	11276.84	14561.60
50		524288	1457.72	3293.41	5091.65	1635.72	2870.74	3769.48	1457.85	2901.36	3812.15	32302.24	21818.67	28586.92
51		1048576	3028.22	6399.33	8522.31	3396.23	6210.85	8553.09	3166.39	5762.09	8266.15	60484.49	42846.60	57817.49
52		2097152	7665.50	13099.72	16814.51	7964.79	12777.65	16887.18	6478.06	11658.84	16479.82	121312.25	84923.13	114988.62
53		4194304	15045.75	25872.08	33770.33	16523.37	25103.60	33676.05	12205.62	24408.60	32972.72	244726.48	168511.72	225886.10

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1			CNW	CNW	CNW	CNW	CNW	CNW	CNL	CNL	CNL	CNW	CNW	CNL
2	Test	Msg Size	4 SN	sz4 VN2	sz4 VN4	sz4 SN-AP	sz4 VN2-AP	sz4 VN4-AP	n4 N1	n4 N2	n4 N4	sz16 VN4	sz16 VN4-AP	sz16 N4
3														
54	AllReduce	0	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.05	0.04	0.04
55	t_avg[usec	4	14.16	21.12	33.86	16.64	24.14	49.66	23.64	29.65	38.28	89.02	98.68	93.27
56	lower is be	8	14.18	21.05	34.18	16.74	24.14	49.65	23.67	29.73	39.09	89.46	98.69	93.41
57		16	14.33	21.01	33.75	16.74	24.22	49.77	23.69	29.77	39.10	89.29	99.36	93.70
58		32	24.77	26.80	34.32	20.63	28.77	52.61	30.21	35.64	46.70	102.04	106.32	119.09
59		64	24.66	26.90	34.57	20.73	30.09	51.79	30.36	36.08	46.78	101.98	107.41	120.10
60		128	24.83	27.31	34.81	21.03	29.02	52.50	30.32	36.44	47.32	102.69	107.24	121.09
61		256	24.96	27.65	34.87	21.24	30.34	52.65	30.48	38.67	46.75	103.56	107.18	121.78
62		512	26.49	28.69	35.64	22.04	30.92	51.59	31.22	39.90	48.82	106.65	108.21	123.90
63		1024	27.66	29.51	36.78	22.84	30.05	52.68	33.23	41.54	50.55	111.12	108.14	128.20
64		2048	27.75	34.09	51.86	24.94	35.71	59.31	36.22	45.42	57.28	134.52	118.85	143.58
65		4096	53.97	71.73	88.58	47.15	65.65	110.55	68.62	86.72	111.98	236.53	220.84	265.00
66		8192	57.73	77.75	110.75	52.86	73.05	118.58	78.60	97.03	134.17	260.27	228.49	289.17
67		16384	69.34	94.86	133.01	61.87	84.34	124.70	92.88	114.15	143.72	297.13	239.23	326.75
68		32768	92.77	130.87	175.42	82.89	117.03	151.87	117.31	146.06	184.27	366.78	278.22	401.49
69		65536	148.65	202.80	269.09	135.36	211.43	260.50	168.07	210.62	276.94	488.83	409.19	540.72
70		131072	389.79	369.11	475.19	237.64	380.01	495.07	336.49	394.85	504.35	767.93	722.42	874.22
71		262144	645.86	748.79	961.45	668.59	806.87	1060.59	551.73	768.96	938.33	1483.15	1492.60	1495.31
72		524288	1255.59	1497.84	2267.13	1210.61	1586.96	2423.50	1276.38	1608.19	1969.21	3213.83	3283.07	3037.97
73		1048576	2621.08	3729.02	5712.22	2368.18	3789.08	5314.94	2457.55	3189.41	4624.68	7327.34	6841.67	6506.37
74		2097152	6232.49	7950.88	12270.92	5776.60	7676.74	11129.46	5369.40	7335.40	11700.22	15363.00	14204.48	15814.23
75		4194304	12564.5	16030.23	24014.87	12006.00	15704.45	22148.12	9088.03	15466.05	25810.15	31056.37	28685.43	33872.36

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1			CNW	CNW	CNW	CNW	CNW	CNW	CNL	CNL	CNL	CNW	CNW	CNL
2	Test	Msg Size	4 SN	sz4 VN2	sz4 VN4	sz4 SN-AP	sz4 VN2-AP	sz4 VN4-AP	n4 N1	n4 N2	n4 N4	sz16 VN4	sz16 VN4-AP	sz16 N4
3														
76	Allgather	0	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
77	t_avg[usec	1	14.02	21.13	33.80	16.67	24.10	49.61	23.59	29.71	39.58	88.10	98.77	93.63
78	lower is be	2	14.00	21.14	34.04	16.51	24.11	49.60	23.58	29.82	39.69	88.74	98.92	93.99
79		4	14.00	21.17	34.01	16.51	24.10	49.63	23.43	29.83	39.74	96.22	95.08	105.10
80		8	14.09	21.22	34.13	16.60	24.14	49.75	23.50	29.76	39.61	102.11	101.15	115.52
81		16	19.58	26.37	33.95	18.85	25.71	48.68	27.06	35.92	43.99	102.04	105.02	119.73
82		32	24.99	26.51	34.07	20.99	28.87	51.20	30.41	35.67	47.05	102.62	108.85	120.94
83		64	24.02	26.48	34.05	20.94	30.72	52.82	30.26	35.79	47.26	104.58	108.96	122.63
84		128	24.29	26.64	34.61	20.98	29.03	52.43	30.43	37.28	47.36	106.39	109.08	125.04
85		256	24.64	26.98	34.84	21.35	29.20	51.29	30.66	39.04	47.52	111.94	111.61	134.81
86		512	25.26	27.55	35.71	21.43	29.35	51.69	30.91	40.40	48.78	120.61	114.87	143.80
87		1024	23.84	30.04	43.73	22.48	31.63	55.55	33.44	43.71	55.17	140.62	123.12	167.61
88		2048	25.16	34.41	52.23	23.10	33.54	59.24	37.12	47.66	61.36	172.17	145.50	210.95
89		4096	28.29	39.35	58.66	25.35	33.88	60.82	42.43	55.71	71.57	234.58	217.67	295.83
90		8192	35.52	50.22	71.36	30.95	48.23	70.35	51.33	69.94	88.52	370.25	420.10	483.69
91		16384	51.13	75.06	97.25	46.32	86.33	111.20	69.80	98.51	123.16	934.13	903.85	956.54
92		32768	83.52	138.12	155.82	79.39	171.93	219.67	107.31	166.36	196.46	1355.02	1468.71	1384.54
93		65536	177.20	271.87	355.22	206.22	448.34	480.76	196.65	343.99	360.50	2665.89	2958.28	2908.83
94		131072	405.89	616.12	986.64	411.10	889.44	1006.32	367.49	790.01	852.19	6274.41	6421.94	5956.98
95		262144	773.58	1179.52	2116.10	773.53	1765.98	1873.08	662.95	1512.14	1661.00	11811.53	12166.62	11439.73
96		524288	1551.70	2837.27	4503.75	1553.02	3547.72	3611.75	1279.84	2815.40	3560.94	22600.97	23753.84	22504.40
97		1048576	3058.06	6742.32	7884.14	3056.92	5131.97	7869.61	2974.81	7003.66	7801.13	47204.72	48068.81	50644.39
98		2097152	7134.52	13533.41	15939.90	7207.92	12353.85	15535.90	6415.25	13905.04	15572.35	93609.82	94947.87	100390.57
99		4194304	14728.80	24585.13	31836.16	14743.15	21220.97	31617.50	12381.45	27300.24	30713.98	187570.17	192495.24	200687.34

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1			CNW	CNW	CNW	CNW	CNW	CNW	CNL	CNL	CNL	CNW	CNW	CNL
2	Test	Msg Size	4 SN	sz4 VN2	sz4 VN4	sz4 SN-AP	sz4 VN2-AP	sz4 VN4-AP	n4 N1	n4 N2	n4 N4	sz16 VN4	sz16 VN4-AP	sz16 N4
3														
100	Sendrecv	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
101	MB/s	1	0.28	0.19	0.11	0.24	0.16	0.08	0.17	0.16	0.09	0.10	0.08	0.09
102	higher is be	2	0.55	0.37	0.22	0.49	0.32	0.16	0.33	0.32	0.18	0.20	0.15	0.17
103		4	1.10	0.75	0.45	0.98	0.63	0.31	0.66	0.63	0.36	0.41	0.31	0.34
104		8	2.19	1.50	0.89	1.95	1.26	0.62	1.32	1.26	0.73	0.81	0.62	0.69
105		16	4.32	2.99	1.78	3.85	2.52	1.23	2.64	2.51	1.46	1.62	1.22	1.34
106		32	5.12	4.59	3.56	5.90	3.99	2.35	4.18	4.07	2.42	2.98	2.26	2.31
107		64	10.32	9.21	7.12	11.85	8.07	4.70	8.22	8.08	4.73	5.93	4.55	4.59
108		128	20.51	18.31	14.16	23.27	15.91	9.88	16.68	16.13	9.59	11.77	8.99	9.03
109		256	41.21	36.24	28.34	46.27	31.65	19.85	32.76	32.03	19.58	23.42	18.05	17.89
110		512	75.10	70.94	55.96	93.99	62.07	39.37	65.11	62.29	39.51	46.20	35.89	35.35
111		1024	152.43	138.25	111.28	182.47	122.85	76.91	128.88	121.22	75.74	90.00	71.40	68.24
112		2048	317.24	250.35	163.59	353.61	219.35	138.49	237.47	216.87	137.09	141.42	130.12	129.86
113		4096	617.00	480.24	315.74	530.60	421.66	277.50	411.81	383.23	232.98	268.42	255.40	228.72
114		8192	1101.74	864.26	577.31	929.94	801.09	555.73	725.26	684.71	520.32	494.76	475.40	385.60
115		16384	1728.13	1051.30	1012.70	1357.75	924.66	836.68	1183.05	975.34	871.78	837.96	632.25	613.23
116		32768	2291.99	1806.72	1579.99	1790.01	1094.95	852.38	1686.91	1549.53	1131.51	1188.87	638.69	861.27
117		65536	2726.56	2196.47	1742.04	1948.94	1443.57	860.64	2115.78	1957.81	1298.66	1414.05	656.58	932.83
118		131072	2347.61	1401.27	1907.72	1944.43	879.31	743.89	2180.43	962.89	999.06	1163.30	601.82	754.78
119		262144	2089.21	1379.19	2211.97	2074.04	911.19	801.37	2500.39	1019.60	1036.48	1378.70	643.97	800.36
120		524288	3111.62	1315.75	1474.54	2145.29	1072.45	833.71	2674.96	1055.69	1143.64	1086.39	618.08	830.45
121		1048576	2700.23	1569.86	848.88	2182.44	1051.95	850.43	2050.44	813.59	801.20	635.86	641.77	583.40
122		2097152	2207.27	1532.83	858.99	2203.25	924.64	859.19	2047.14	840.42	809.37	623.25	655.90	594.72
123		4194304	2218.65	1633.09	862.22	2075.17	895.75	862.11	2052.86	832.91	811.77	663.53	628.49	616.50

5.3. Unit Testing

Only a subset of the unit tests applied to AP. They are no regressions to report.

Test Name	SN Accelerated	VN2 Accelerated	VN4 Accelerated
Comm.comtest.cat	✓	✓	✓
Comm..comtest.cat -loops	✓	✓	✓
Comm.ptltest.cat	✓	✓	✓
Comm.ptltest.cat- loops	✓	✓	✓
MPICH2.attr	✓	✓	✓
MPICH2.basic	✓	✓	✓
MPICH2.coll	✓	✓	✓
MPICH2.comm	✓	✓	✓
MPICH2.datatype	✓	✓	✓
MPICH2.errhan	✓	✓	✓
MPICH2.errors	✓	✓	✓
MPICH2.group	✓	✓	✓
MPICH2.info	✓	✓	✓
MPICH2.init	✓	✓	✓
MPICH2.io	✓	✓	✓
MPICH2.pt2pt	✓	✓	✓
MPICH2.rma	✓	✓	✓
MPICH2.spawn	✓	✓	✓
MPICH2.topo	✓	✓	✓
MPICH2.cxx (as above)	✓	✓	✓
MPICH2.f77	✓	✓	✓
MPICH2.f90	✓	✓	✓
Pallas	✓		✓
Shmem.mpif01	✓		✓
Shmem.sma1	✓		✓
Shmem.sma2	✓		✓
Shmem.smaf	✓		✓
Shmem.smaperf1	✓		✓

6. Results of Running on Jaguar June 9-10, 2008

This section was introduced with version 2.0 of the document. It describes the testing done on the Jaguar XT4 system located at ORNL. Jaguar's 7832 compute nodes are populated with 2.1 GHz revision B2 quad-core Opterons. A 24-hour test period was generously provided. After approximately 17 hours of testing, a severe thunderstorm caused a complete loss of power to the computer facility that houses Jaguar. Once the machine was restored to service, we completed the last 7 hours of testing.

We were very pleased with the machine performance and also delighted with the CNW stability. Prior to this test, CNW had only run on five quad-core Opterons. There were no CNW issues during the booting process. We experienced no load failures or unexpected hangs during the entire test time. Ten nodes faulted during testing. One was a machine check in bank 4 (uncorrectable memory error). We assume this was a hardware problem. The second failed node gave an error message from the function `cpu_wait_for_quiet`. We have seen this error on Red Storm, even with Cray's dual-core Catamount Virtual Node OS. The cause is unknown. While this may be a hardware problem, we suspect this is an OS software issue. Three nodes failed due to an operator procedure that was unrelated to CNW. The last five faults occurred when running the POP application. The node running rank 0 of five POP jobs ran out of portals heap space. This is a known issue and had already been increased in CNL, possibly to accommodate this application. We elected to rebuild CNW with a larger QK heap space and reboot. The POP runs were completed with the larger heap size. The only other failure was caused by the power outage. This stability over a 24-hour test is an excellent result for CNW's first introduction to a capability-sized quad-core machine.

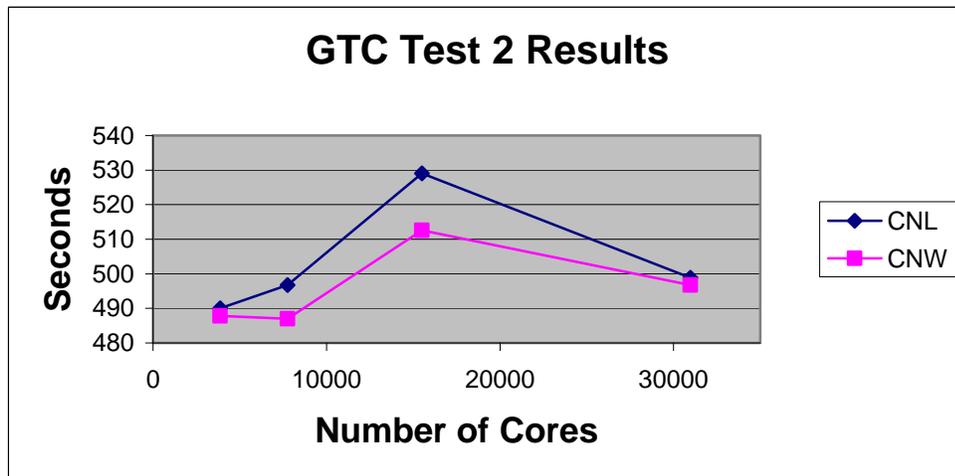
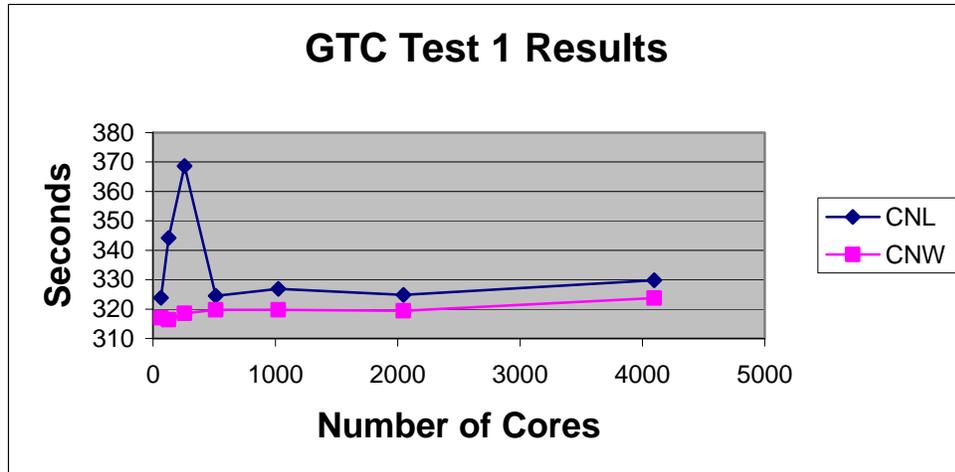
6.1. Application Testing – CNL and CNW Results

ORNL provided four applications and associated test problems. We ran them using CNL during normal Jaguar production time. We could not control job placement for the CNL runs, nor can we estimate that effect on the CNL results. The CNW jobs were placed on contiguous nodes during the dedicated test time. For the small-sized jobs, there were multiple jobs running at the same time. Both the CNW and CNL versions of the application binaries were built using PGI 7.1.6. The same make files were used.

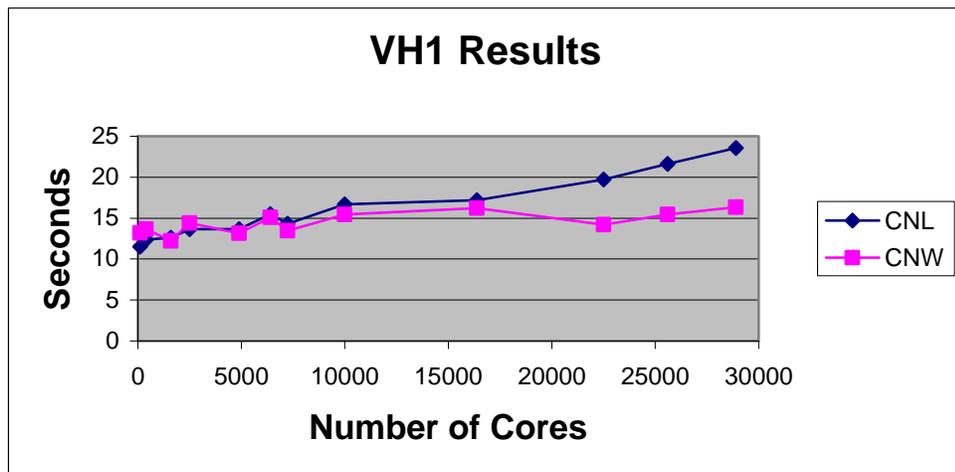
We provide both tabular and graphical versions of the results. All times are converted to seconds, with lower times being better. The "VN" column identifies whether 1, 2, 3, or 4 virtual nodes were specified on the yod command line. The last column shows the percent improvement of CNW over CNL. Note that one POP run used 3 cores per node, while all the other test cases used 4 cores per node. The 3-core test is not included in the graphical results, since it represents a dissimilar test setup. The 3-core test is intriguing as it shows a significant advantage with CNL over CNW. We speculate this may be due to Linux running on the unused core. Catamount always runs the OS and the first application process on CPU 0. Cougar, the predecessor to Catamount, provided the feature to run the application on a different core than the OS. This feature, like thread support (e.g. OpenMP) was removed at Cray's request, when porting Cougar to Catamount for the original single-core Opterons.

App	sockets	cores	VN	(CNL/CNW-1)		
				CNL	CNW	*100%
GTC	16	64	4	323.795	317.072	2.12
	32	128	4	344.157	316.332	8.80
	64	256	4	368.684	318.569	15.73
	128	512	4	324.463	319.737	1.48
	256	1024	4	326.828	319.711	2.23
	512	2048	4	324.788	319.425	1.68
	1024	4096	4	329.805	323.757	1.87
	968	3872	4	490.037	487.813	0.46
	1936	7744	4	496.787	486.999	2.01
	3872	15488	4	529.083	512.593	3.22
VH1	7744	30976	4	498.85	496.777	0.42
	25	100	4	11.5218	13.2072	-12.76
	100	400	4	12.3865	13.6505	-9.26
	400	1600	4	12.5986	12.2138	3.15
	625	2500	4	13.6805	14.3882	-4.92
	1225	4900	4	13.6648	13.1650	3.80
	1600	6400	4	15.5032	15.0973	2.69
	1806	7225	4	14.3358	13.4823	6.33
	2500	10000	4	16.6929	15.4581	7.99
	4096	16384	4	17.1961	16.2242	5.99
	5625	22500	4	19.7081	14.2151	38.64
	6400	25600	4	21.6054	15.4634	39.72
	7225	28900	4	23.5664	16.3595	44.05
	POP	900	3600	4	1537.76	1690.96
1600		6400	4	880.39	1026.65	-14.25
2500		10000	4	779.13	849.30	-8.26
3600		14400	4	628.43	709.24	-11.39
4800		19200	4	661.07	695.61	-4.97
5000		20000	4	729.74	695.44	4.93
6000		18000	3	546.04	598.68	-8.79
AORSA	7200	28800	4	638.18	619.87	2.95
	1024	4096	4	4247.76	3914.70	8.51
	2048	8192	4	4414.98	4119.48	7.17
	4096	16384	4	2395.5	2412.36	-0.70
	5625	22500	4	5791.8	5599.08	3.44
	7056	28224	4	4883.58	4747.32	2.87
AVERAGE						3.83

GTC came with two different test cases. We do not have an explanation for the unusually large difference in the CNL and CNW results for the low core count test cases. It may have been poor job placement when running CNL. That is speculation as we could not determine where node placement information is archived by a system running CNL.



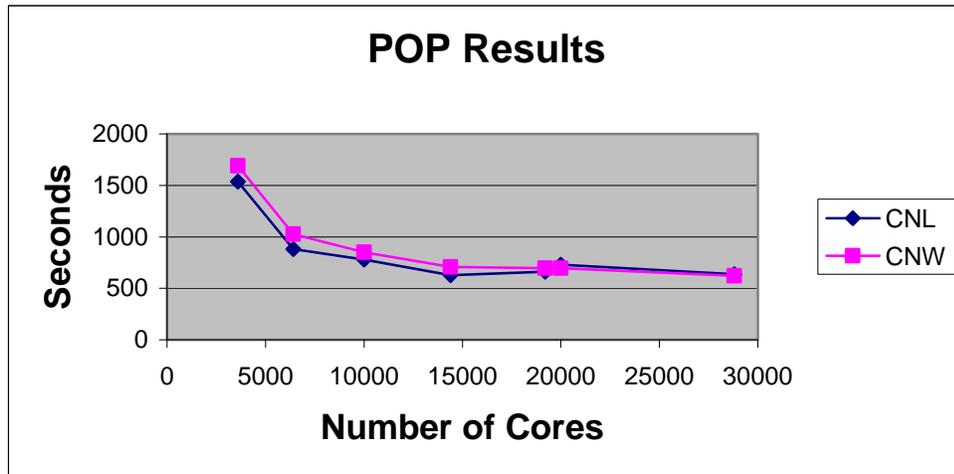
The VH1 test cases ran very quickly. They show the largest disparity with increased node counts.



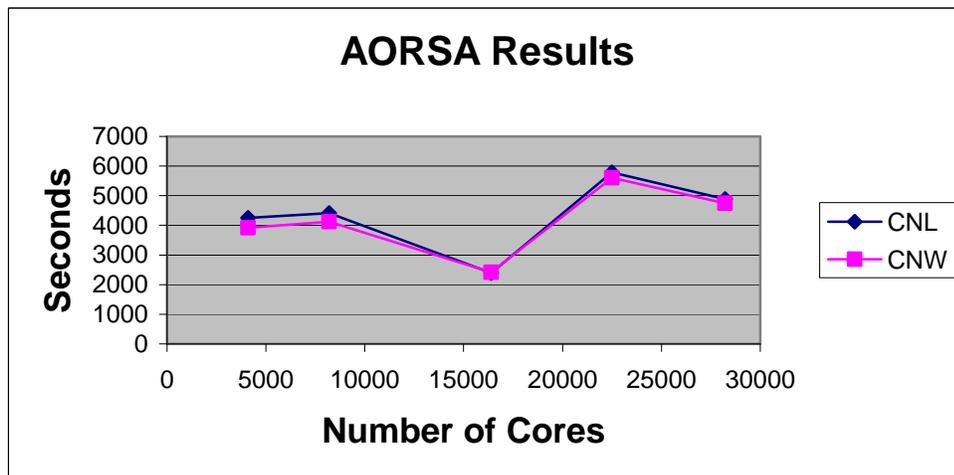
The POP runs show an interesting pattern with CNL outperforming CNW at most core counts. But the advantage reduces as the core count increases. These results are in marked contrast to the

results presented in Section 4.1.1 of this document. Previously on dual cores, CNW significantly outperformed CNL. There have been quite a few changes to POP, CNL, and system libraries since the dual-core test on Jaguar in June, 2007. We conjecture that the recent MPI library change for CNL was an important factor in POP's improved performance on CNL. SGI's MPI library replaced the MPICH2 V1.0.2 that is still provided with Catamount. SGI's library utilizes shared memory features that likely play a key role in collective operations. POP would benefit from an intra-node optimization of MPI_Allreduce(). The benefit would decrease with core count, which is consistent with the trend in the POP results in the following graph. CNL outperforms CNW by 14% at small node counts. By 28,800 cores, CNW outperforms CNL by 3%.

Sandia has introduced shared memory support to CNW. The feature is not yet integrated into system libraries, such as MPI.



The last application tested was AORSA. We had no previous experience with this application and the tests ran for much longer than the other applications.



The CNW tests were rerun using Accelerated portals. Accelerated portals is enabled with a runtime environment variable, so special binaries are not required. The results were quite similar to the small scale results reported in Section 5.1.1. POP was the only application that consistently benefitted from Accelerated portals. As suggested earlier, this new feature would benefit from additional tuning and optimization.

App	sockets	cores	VN	CNL	CNW	CNWacc	(CNL/CNW-1) CNL/CNWacc (CNW/CNWacc-1) *100%		
							*100%	-1)*100%	1) *100%
GTC	16	64	4	323.795	317.072	318.126	2.12	1.78	-0.33
	32	128	4	344.157	316.332	317.562	8.80	8.37	-0.39
	64	256	4	368.684	318.569	317.704	15.73	16.05	0.27
	128	512	4	324.463	319.737	318.600	1.48	1.84	0.36
	256	1024	4	326.828	319.711	319.592	2.23	2.26	0.04
	512	2048	4	324.788	319.425	319.636	1.68	1.61	-0.07
	1024	4096	4	329.805	323.757	320.341	1.87	2.95	1.07
	968	3872	4	490.037	487.813	481.751	0.46	1.72	1.26
	1936	7744	4	496.787	486.999	487.006	2.01	2.01	0.00
	3872	15488	4	529.083	512.593	514.718	3.22	2.79	-0.41
VH1	7744	30976	4	498.85	496.777	487.752	0.42	2.28	1.85
	25	100	4	11.5218	13.2072	13.1627	-12.76	-12.47	0.34
	100	400	4	12.3865	13.6505	13.5551	-9.26	-8.62	0.70
	400	1600	4	12.5986	12.2138	10.9291	3.15	15.28	11.75
	625	2500	4	13.6805	14.3882	13.3367	-4.92	2.58	7.88
	1225	4900	4	13.6648	13.1650	11.8051	3.80	15.75	11.52
	1600	6400	4	15.5032	15.0973	14.4210	2.69	7.50	4.69
	1806	7225	4	14.3358	13.4823	13.5205	6.33	6.03	-0.28
	2500	10000	4	16.6929	15.4581	13.9612	7.99	19.57	10.72
	4096	16384	4	17.1961	16.2242	15.6414	5.99	9.94	3.73
POP	5625	22500	4	19.7081	14.2151	21.5760	38.64	-8.66	-34.12
	6400	25600	4	21.6054	15.4634	22.5660	39.72	-4.26	-31.47
	7225	28900	4	23.5664	16.3595	24.3143	44.05	-3.08	-32.72
	900	3600	4	1537.76	1690.96	1617.23	-9.06	-4.91	4.56
	1600	6400	4	880.39	1026.65	963.36	-14.25	-8.61	6.57
	2500	10000	4	779.13	849.30	772.67	-8.26	0.84	9.92
	3600	14400	4	628.43	709.24	647.16	-11.39	-2.89	9.59
	4800	19200	4	661.07	695.61	616.15	-4.97	7.29	12.90
	5000	20000	4	729.74	695.44	618.59	4.93	17.97	12.42
	6000	18000	3	546.04	598.68	530.12	-8.79	3.00	12.93
AORSA	7200	28800	4	638.18	619.87	558.65	2.95	14.24	10.96
	1024	4096	4	4247.76	3914.70	3918.54	8.51	8.40	-0.10
	2048	8192	4	4414.98	4119.48	4039.68	7.17	9.29	1.98
	4096	16384	4	2395.5	2412.36	2303.04	-0.70	4.01	4.75
	5625	22500	4	5791.8	5599.08		3.44		
	7056	28224	4	4883.58	4747.32		2.87		
AVERAGE							3.83	3.88	1.26

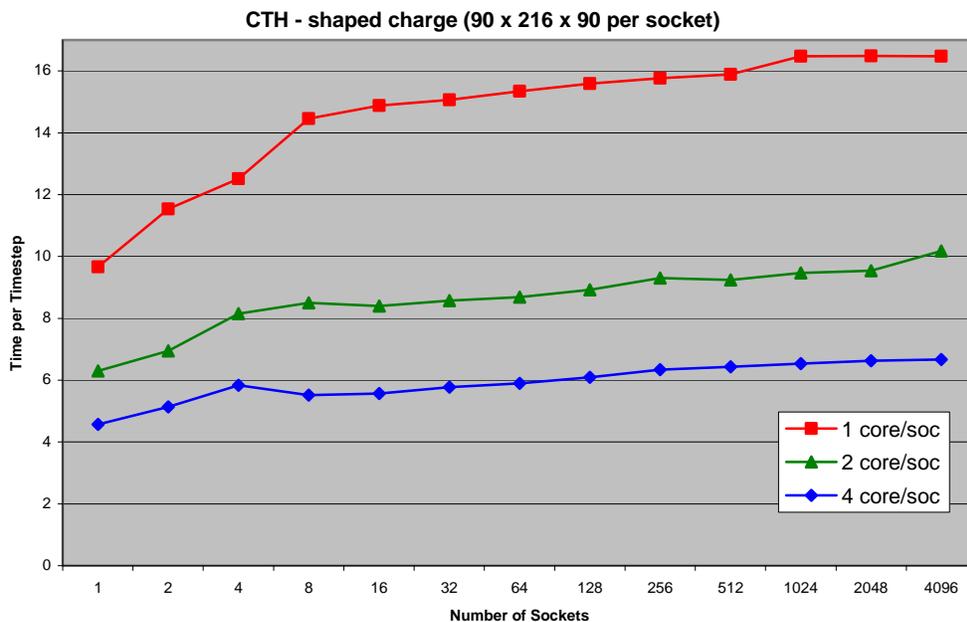
6.2. CTH Scaling Study with CNW

The CTH scaling study provided useful data to study the effective utilization of the additional cores. We began by running the same problem (shaped charge 2000 x 4800 x 2000) three times on 7800 CPU cores. The first time, only one core was used on 7800 sockets (SN mode). The second time, two cores were used on 3900 sockets (VN2 mode). Finally, four cores were used on 1950 sockets (VN4 mode). If there were no resource contention, the timing results should have been the same. Since memory and the SeaStar interface are shared resource, the time per timestep are different:

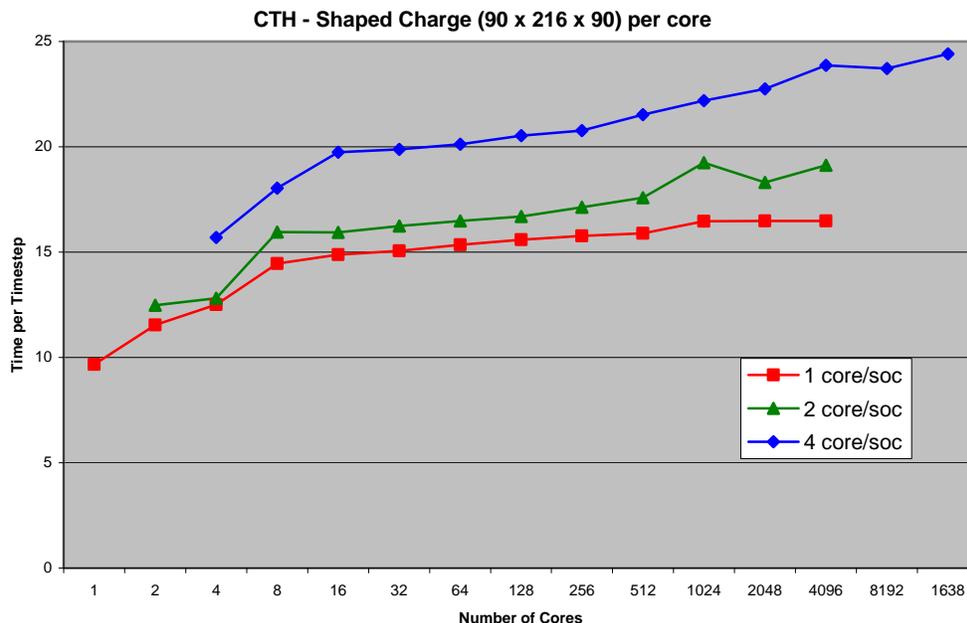
SN (secs)	VN2 (secs)	VN4 (secs)
23.258	25.371	31.340

Using just this one set of data points and the calculation described in Section 4.1.3, we obtain an effective utilization of 2.96 cores. This 7800-core result is very close to the CTH result reported in Section 4.1.3 on 4 cores. We were not able to run this test at large scale on other applications.

We did do two full scaling studies using CTH. The first graph depicts the results of assigning the same amount of work to each socket. The top red line uses only one core per socket to do the work. The middle green line uses two cores to do the same amount of work. Ideally, it would take half the time. The bottom blue line uses four cores to do the work. Again with no contention, the results should be halved. The drop between the 1-core and 2-core results is very nice and most likely shows the advantage of the improved memory subsystem on this Opteron over the previous generation of dual-core Opterons. The 4-core result is not ideal, but is consistent with the previous single data point on effective core utilization. The values range from 2-3 cores effectively utilized.



The second scaling study shows similar results. It compares the same problem on the same number of cores. This time, the number of cores remains constant, but the number of sockets changes. Again, we see a nice, smooth, and relatively flat scaling curve. Two-core utilization is very good and satisfactory on four cores.



6.3. HPL and HPCC Results

We ran HPL on 16,384 cores (4096 sockets) with the N value set to 800,000. This run takes about one hour. CNW showed a 1% improvement, probably due to the use of large (2MB) pages.

	Run time (sec)	GFLOPS
CNL	3817.75	89410
CNW	3777.03	90370

Lastly, we ran HPCC without the HPL portion of the benchmark. This was an optimized (in contrast to baseline) HPCC run. We used Steve Plimpton’s optimized random access algorithm and Mark Sears’ optimized FFT algorithm. (These algorithms have been described in published proceedings and are available from the authors upon request.)

PTRANS produced 1839.22 GB/sec, which is about what we would expect given the number of NICs. STREAMS was 86888.6 GB/sec, which is 11.14 per socket which shows the faster DDR2 memory. Random access produced 22.805 giga-updates/sec. This is an expected result given the number of NICs. Lastly FFT reported 3747.38 GLOPs/sec on 31104 cores – a little less than expected but probably reasonable given the contention on the sockets.

7. Future Plans

This project has satisfied the project goal of providing CNW, a viable light weight operating system for quad-core Opterons on Cray XT4 systems. However, ORNL plans to use Compute Node Linux on its Jaguar system and CNW would be archived.

Due to a fortuitous sequence of events in December, 2007, Sandia was able to negotiate an upgrade to the Red Storm system. Sixty five of its 135 compute cabinets will be upgraded from dual-core to quad-core Opterons during the summer of 2008. Sandia chooses to remain with the Catamount Light Weight Kernel and CNW is now running on Red Storm. In fact, unless CNL is modified to support a mixture of dual and quad-core nodes, CNW is required on Red Storm. Thus, CNW will continue to move forward, based on the work done for this project.

Sandia plans to continue the analysis of CNL performance versus CNW on applications of interest to Sandia. In April, Sandia compared CNL and CNW on Red Storm using 12,960 dual-core Opterons for the CTH and Partisn applications. The difference was more marked than the applications run on Jaguar in June. For example, CTH showed an average 6.7% improvement when using CNW, with the difference being greater at higher core counts.

