Extending Catamount for Multi-Core Processors

John Van Dyke, Courtenay Vaughan, Suzanne Kelly
Sandia National Laboratories
Scalable Computing Systems Department
PO Box 5800, MS 1319
Albuquerque, NM 87185-1319
jpvandy@sandia.gov, ctvaugh@sandia.gov, smkelly@sandia.gov

ABSTRACT
The XT3 Catamount Virtual Node (CVN) implementation was based on the dual processor support in ASCI Red's [1] Cougar Light Weight Kernel Operating System. That solution was limited to no more than 2 virtual nodes per physical node. This paper describes the design for extending Catamount to support more CPUs per node. It identifies the areas needing modification and the selected resolution. Some preliminary performance results are provided.

Keywords
Operating Systems, MPP, Light Weight Kernel, Multi-Core, XT3, XT4, Catamount.

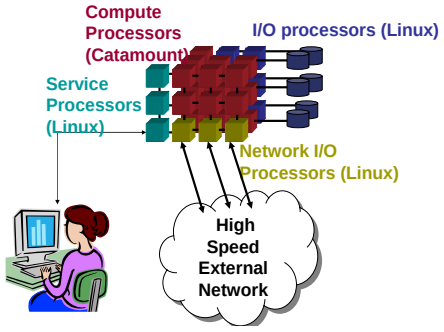
1.0 Introduction

We begin with a brief description of the Catamount Light Weight Kernel Operating System. A fuller description can be found in [2]. Emphasis is placed on areas needing modification for multiple CPU cores. Section 3 describes the requirements for a multiple core solution and we provide the design and implementation in Section 4. While the development is still underway, some early dual-core performance results are available and we present these in Section 5.

2.0 Description of Catamount

Catamount assumes a functionally partitioned MPP [3]. That is, Catamount runs on processors intended for intense computation and relies on other processors within the MPP to perform additional services. These service processors run Linux and provide the interactive development environment, file I/O, and high speed access to external services. Figure 1 depicts the functional partitions.

Figure 1: Functional Partitions of MPP using Catamount



Continue to refer to Figure 1 as we discuss the usage model for Catamount. A user logs into a Linux service processor and performs typical application execution set-up, such as compilation and creation of the problem dataset. The user then requests that the application be run on some number of compute nodes. The user also specifies how many process instances to run on each node, where the number of processes cannot exceed the number of cores. For the current CVN implementation, the maximum is two processes. The user-invoked "yod" program, executing on a service processor, launches the application on the assigned compute nodes. Once started, the application runs under Catamount's control. The application instances on each compute processor pass messages to communicate with each other over the tightly coupled network. File I/O flows between the compute processors and the I/O processors. When the application completes, the user can review the results on the service processors

* 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.

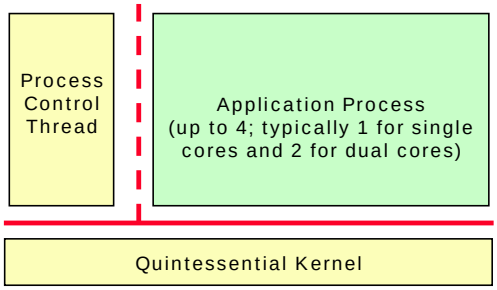
and/or direct that the files be sent to external services for post processing or archival storage.

The Catamount operating system consists of a Quintessential Kernel (QK) and a Process Control Thread (PCT). The PCT and the QK work together to provide the functionality required to run a scientific calculation. The PCT will decide what physical CPU core, and what physical memory a new process is to have.

CVN allows the application to use twice as many nodes with no change to the application executable. Note, however that the number of processors is the only resource that has been doubled. The node memory is split between the two processes and the two processes share network access. It should not be thought of as an SMP since the two processes on a node do not share memory.

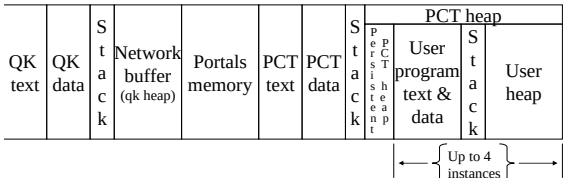
At the behest of the PCT, the QK will set up the virtual addressing structures for the new process that are required by the hardware. The PCT will decide which processes are ready to run on their respective cores and at the behest of the PCT, the QK will flush caches, set up the hardware registers, and run the process(es). The basic structure of these components is shown below:

Figure 2: Compute Processor Components



While Figure 2 provides a useful logical view of the components, the physical layout is quite different and is shown in Figure 3.

Figure 3: Catamount Physical Memory Layout (not to scale)



When the QK installs the PCT, the remainder of physical memory is included in the PCT's heap. When the PCT loads an application, it utilizes the bulk of its heap space for the application's memory. It divides this space equally for each CVN process running on the node.

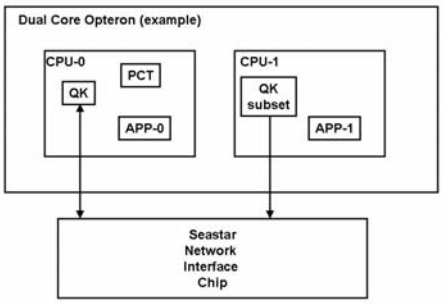
While Catamount utilizes virtual addressing, there is no virtual memory support. This is an important performance, reliability and scalability feature of Catamount. Disks needed to implement virtual memory are very slow in comparison to memory access, have a low mean time to failure, and impede the predictable progress of the application.

Another feature of Catamount's memory management is default support for 2 MB pages. For many applications, larger pages can significantly reduce cache misses and TLB flushes as they cover a larger percentage of memory. Smaller, 4K pages are supported as well for applications that require more random access of memory. The user enables small pages with a command line option when starting the parallel application.

The decision whether to run on the second core in virtual node mode is made at run time. The default can be overridden with a yod command line option. With –SN, only one core is used. With –VN, both cores are used. If an application requires more than one half of the physical memory on a node to do its share of the computation, VN mode cannot be used. There is no virtual memory to extend the address space. Another memory-related consideration is contention. The processors each have their own cache, but the memory bus is shared. The frequency of cache misses will have an impact on how effectively the second processor can be utilized, as they compete for memory bandwidth.

Similarly, the two processes on the same node must share the Network Interface Chip (NIC). A simple locking mechanism allows each process to safely send their messages independently. (This is an enhancement since the description given in [2].) In order to determine process destination, only one processor will receive all incoming messages. These contention issues can impact the effectiveness of CVN.

Figure 4: Dual Core CPU responsibility assignment



Several papers have documented the effectiveness of dual cores, in particular on Catamount and Linux. They can be found in references [4] thru [9].

Given this background on Catamount and its dual-core implementation, we turn now to the requirements for a multi-core version, called, N-way Catamount.

3.0 Requirements and Restrictions

The immediate goal is to create an enhanced N-way Catamount to support 4 processors per node, suitable to run on an XT4 populated with quad-core AMD Opterons (Barcelona). In so far as reasonable, the implementation should be N-way rather than 4-way and *will* be able to run on single core or dual core processors without recompilation.

3.1 Regression-less Functionality and Performance

As is typical of most enhancement efforts, regressions are not acceptable. Existing functionality shall be maintained. 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 Processors Allocator, the batch scheduler, and the RAS system.

There was one identified exception to the no-regression requirement. The undocumented "share mode" feature in CVN 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. As with the CVN implementation, the number of processes per node shall be the same for each subset of nodes. The same binary shall run on each process on the node. Only the last specified binary can request a virtual node count that is not a multiple of the number of processes per node.

3.2 Networking

Enhancements to the original CVN implementation allow each processor to access the 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 will support NIC-side processing of the protocol.[10] This will allow messages to be sent to the target CPU/process immediately upon message receipt.

3.3 Processes per node (ppn)

The current yod options of SN or VN do not lend themselves to expansion to 4 (or more) virtual nodes per node. The current N-way plan is a relatively major API change. The yod command line specification of –SN and –VN will no longer be supported. The yod –sz (or –size or –np) option will now refer to the number of physical nodes to be allocated. With this change, the value in the –sz option will match the number specified on a qsub command. The number of virtual nodes per physical node will be specified with a new option, -ppn, which stands for processes per node. To allow an application to

run on a number of virtual nodes that is not a multiple of the processes per node, a –total-virtual-nodes option is introduced. In UNIX-style notation, the command line format will be:

yod –sz/size/np=#nodes [[-ppn=procs_per_node] -total-virtual-nodes=#vn] …

Should the yod command not include a ppn specification, the file /etc/xt.conf will be consulted for the site default. If none, one will be used.

3.4 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.

3.5 Other Requirement Considerations

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.)

4.0 Design of Required Changes

While these changes are currently being made in the code, presently the only testing is single and dual core to verify no regressions.

4.1 Limit Memory Requirements

The requirement for OS memory to not grow with the number of nodes, is not met by CVN. The 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.

4.2 Change "2" to "N"

Conceptually the changes to go from two-way to N-way are quite simple. In CVN there are many places were 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, will be dimensioned and references will be converted to loops, as appropriate.

4.3 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.

4.4 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.

4.5 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.

4.6 Portal Process Identifier

The file system software cared about not reusing Portals identifiers too rapidly so a rolling bias was added to the compute nodes Portals PID.

4.7 Portals Networking Software

During the development of the Red Storm system, two versions of the Portals protocol were conceived. One version would run on the host, while the second version would run on the NIC. (These are sometimes referred to a "generic" and "accelerated" Portals.) The second version has never been fully developed and integrated into CVN. It will be introduced, tested, and integrated into N-way. Using the NIC to process the protocol becomes more important as the number of cores being supported by one NIC grows.

5.0 Performance Metrics

In order to validate the design requirements of no regressions going from CVN to N-way in either performance or functionality, we have tested on a variety of systems. The largest test to date was run on the mixed XT3/XT4 system at NCCS at Oak Ridge. For that test, we ran two application codes at sizes up to the full machine. The codes were CTH, a shock hydrodynamics code, and PARTISN, a neutron transport code. The performance results from these codes run in VN mode on problems which were scaled with the number of processors are shown in figures 5 and 6. For both of these figures, better performance is indicated by lower values.

Figure 5: CTH VN performance

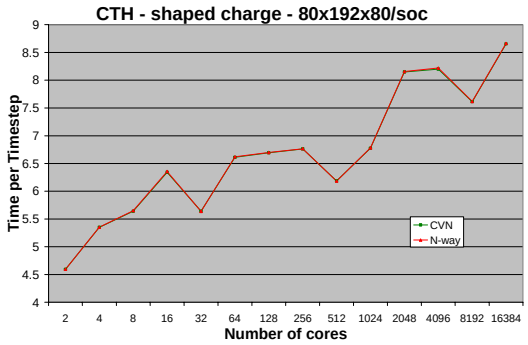
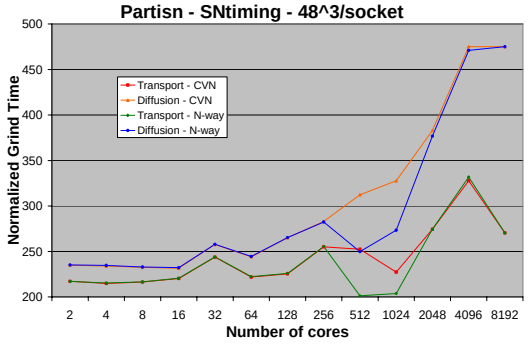


Figure 6: PARTISN VN performance



During this test, we first ran the tests with the machine running the CVN system and then ran the same tests in the same order with the N-way system. In the case of CTH, each of the tests gave the same answers on both operating systems. The difference in performance was between 0.06% faster for N-way and 0.21% faster for CVN, with most tests indicating less than 0.05% difference. This amount of difference is consistent with run to run variation. We also ran CTH in SN mode on 11500 nodes of the machine and saw that CVN was 1.2% faster than N-way. This is more than would have been expected. The other thing that can be noted about the CTH results is the dips in the scaling results for 32, 512, 1024, and 8192 cores. These tests were run after all of the other CTH tests were run and they ran on the XT4 nodes of the machine. All of the other runs, with the exception of the 16384 core runs which ran on a mixture of nodes, ran on the XT3 nodes of the machine. These results seem to indicate that the XT4 nodes were consistently faster than the XT3 nodes.

The PARTISN results show two different data points for each run. This is for the two different phases of the code. We tried to run the tests in the same order, but had a processor failure on the

largest size (16384 cores) for N-way. Not having time to rerun that test, we went ahead and ran the remaining tests. As a result, the tests for 512 and 1024 cores were run on XT4 nodes for the N-way test, while they had been run on XT3 nodes for the CVN tests. The tests for 4096 cores were also run on a different combination of XT3 and XT4 nodes between the two tests. Other than those tests, we have similar performance between CVN and N-way, with the differences being in the same range as those from CTH. We were also not able to compare the results for running on 11500 processors in SN mode due to a processor failure while running in N-way.

6.0 Future Work

This is clearly a "work in progress". This N-way code has never executed on a quad-core processor. The major identified software change remaining is in the QK to accommodate the third memory page size of one gigabyte, which is to be available in quad-core Opteron. Beyond that one must verify (or restore) functionality in quad-core Catamount. Finally we get to do the potentially very interesting scaling studies comparing single-core mode, dual-core mode and quad-core mode.

Sandia is also continuing to research the design, implementation and deployment of operating systems for massively parallel scientific computing platforms. A research project is underway to investigate a framework for building application-specific operating systems [11]. This project is a collaboration between Sandia, the University of New Mexico, and the California Institute of Technology.

Acknowledgement

We are grateful to Fred Johnson in DOE Office of Science to enable special funding for this study. Part of the testing was conducted at the Oak Ridge National Laboratory.

7.0 References

[1] T. G. Mattson, D. Scott, and S. R. Wheat, "A TeraFLOP Supercomputer in 1996: The ASCI TFLOP System," presented at International Parallel Processing Symposium, Honolulu, HI 1996.

[2] S. M. Kelly, R. B. Brightwell, and J. P. Van Dyke, "Catamount Software Architecture with Dual Core Extensions," presented at Cray User Group, Lugano, Switzerland, 2006.

[3] R. Brightwell, W. J. Camp, B. Cole, E. DeBenedictis, R. Leland, J. Tomkins, and A. B. Maccabe, "Architectural Specification for Massively Parallel Computers: An Experience and Measurement-Based Approach," *Concurrency and Computation: Practice and Experience*, vol. 17, pp. 1271-1316, 2005.

[4] R. Brightwell, K. D. Underwood, C. Vaughan, "An Evaluation of the Impacts of Network Bandwidth and Dual-Core Processors on Scalability", 2007 Cray Users' Group Conference, May 2007.

[5] H. Wasserman, "Performance Analysis of the Cray XT4", Cray Technical Workshop USA, February 2007, http://nccs.gov/news/workshops/cray/pdf/Wasserman.pdf

[6] P. Worley, "Cray XT3/XT4 Performance Analysis", Cray Technical Workshop USA, February 2007, http://nccs.gov/news/workshops/cray/pdf/Worley_CrayTech_2_26_07.pdf

[7] J. Kuehn, "HPCC on the Cray XT3 and Cray XT4: Results & Analysis", Cray Technical Workshop USA, February 2007, http://nccs.gov/news/workshops/cray/pdf/xt3_xt4_comparison.pdf

[8] S. R. Alam, R. F. Barrett, J. A. Kuehn, P. C. Roth, J. S. Vetter, "Characterization of Scientific Workloads on Systems with Multi-Core Processors", IEEE International Symposium on Workload Characterization (IISWC) San Jose, October 2006.

[9] D. M. Pase, M. A. Eckl, "A Comparison of Single-Core and Dual-

Core Opteron Processor Performance for HPC", http://www-03.ibm.com/servers/eserver/opteron/pdf/IBM_dualcore_whitepaper.pdf

[10] R. Brightwell, K. T. Pedretti, K. D. Underwood, T. Hudson, "SeaStar Interconnect: Balanced Bandwidth for Scalable Performance", *IEEE Micro*, vol. 26, no. 3, pp 41-57, 2006.

[11] A. B. Maccabe, P. G. Bridges, R. Brightwell, R. Riesen, and T. B. Hudson, "Highly Configurable Operating Systems for Ultrascale Systems," presented at First International Workshop on Operating Systems, Programming Environments and Management Tools for High-Performance Computing on Clusters, St. Malo, France, 2004.