Compute Node Linux: New Frontiers in Compute Node Operating Systems

Dave Wallace
Cray Inc

ABSTRACT: *This is a discussion of Compute Node Linux performance and the work underway to bring that performance in line with Catamount.*

KEYWORDS: Light Weight Kernel, Linux

1. Introduction

The talk upon which this paper is based was designed to discuss the performance related work going on within Cray Software Development. There has been a lot of interest in the performance on Compute Node Linux, CNL. The original goal for CNL was to be within 10% of Catamount performance for a set of applications that are important to current customers. Because of the interest it seemed like a good idea to give an update on the current performance and what we are learning about Linux as a compute node operating system. The results presented here are early. There is still work to be completed, and we are learning more about system interactions everyday.

The talk that this paper is based on covered five areas involved in the current performance investigations. The areas are: metrics – what is being measured, Jitter – What we are learning about Jitter, Portals – probably the key area so far in our performance investigations, I/O – our baseline data about I/O performance, and finally application results – a look at some current results relative to Catamount.

It is critical to remember that while Cray is moving to use CNL this does not imply that Catamount is bad or undesirable. Catamount is the standard in compute node operating systems. Cray continues to sell and service Catamount.

2. Metrics

The measures for CNL are primarily based around application performance. However, there are other interesting metrics for a compute node operating system. I/O is important and the ability to handle I/O requests across many compute nodes is important. I/O measures can be taken for applications and by using I/O benchmarks.

Another interesting metric is the time it takes to start and stop applications. This is often a long time on clusters and Catamount has set the standard for starting an application in seconds even on multiple thousands of nodes. There is some room for CNL to show better performance – but mostly when there are application failures – as Lustre lock recovery is much faster in Linux than Catamount.

Early analysis of differences between Catamount and Linux showed little variation in on node performance. The similarities in obvious things like hardware, compilers, and libraries meant that much of the computation would be the same under either operating system. The differences would be focused in areas of communication, the messaging between nodes and in operating system overhead differences between Catamount and Linux.

3. Development Work in Progress

There are several areas of work underway in development. Each subproject has a set of tasks that are in progress and have some initial results. The first area of interest is

Jitter. Jitter is described in a number of papers over the past few years. Basically this is the impact of operating system overhead on application performance.

There are several approaches that have been taken to control Jitter. Cray used a synchronized scheduler on the XD1. The approach works well when there are many services being used on the compute node. Since Catamount did not use this approach and scaled well. And Catamount "compatibility" is the initial objective of the CNL project, the decision was made to not use the XD1 synchronized scheduler and instead remove all the services not absolutely required for operation.

Portals performance with Linux is another focus of the project. This area was known to need tuning to support applications. Changes have been made to focus on the Portals driver to reduce the overhead of this driver as much as possible. Work in the driver included making multi-threading locking enhancements and changing how memory is managed within the driver. Some of this work is complete and it appears to be making a positive (in terms of results – negative in the time spent in the driver) impact.

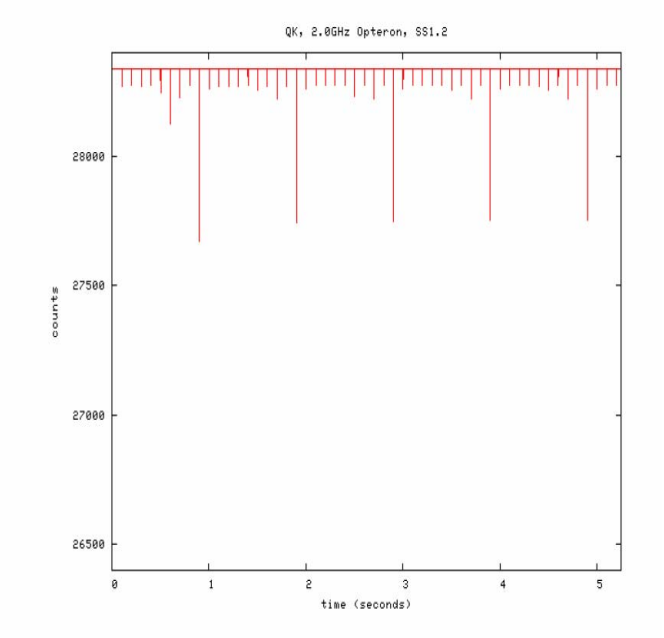
In the area of I/O there are changes required for both Jitter – Lustre heartbeats that are essentially Jitter. And there is the problem of too many console messages which has been a problem with the Linux clients. These are being worked on and are fixed respectively.

In the Programming Environment area there are some important efforts underway. First, "send to self" a feature that uses Linux shared memory as a fast transport for MPI messages between processes on the same node. We expect to get somewhere in the vicinity of 0.5 microsecond for zero byte latency for this sort of MPI messaging. This is substantially higher than thee mechanism available under Catamount. S

The second Programming Environment change is the use of mixed MPI and OpenMP models within an application. The use of OpenMP on the node makes sharing easier. MPI continues to be the messaging mechanism for off node communication.

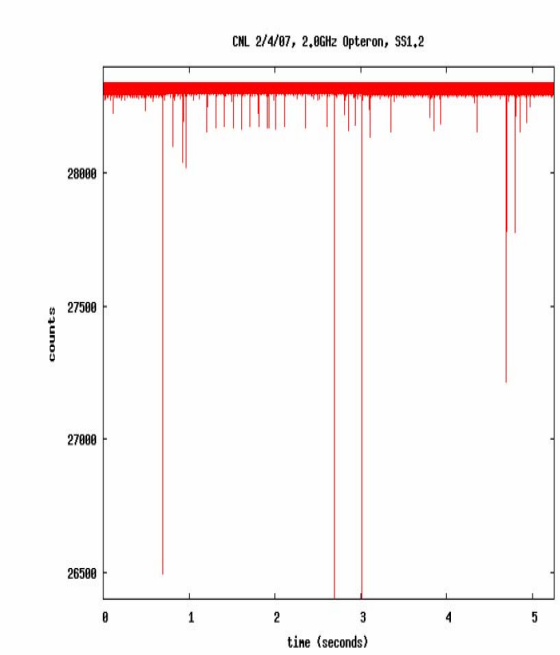
4. Jitter

The best measure of a Jitterless compute node operating system is Catamount. We used the FTQ benchmark because it has been used to measure Jitter on a compute node by other researchers. The following is a graph of the output of FTQ on a Catamount node.



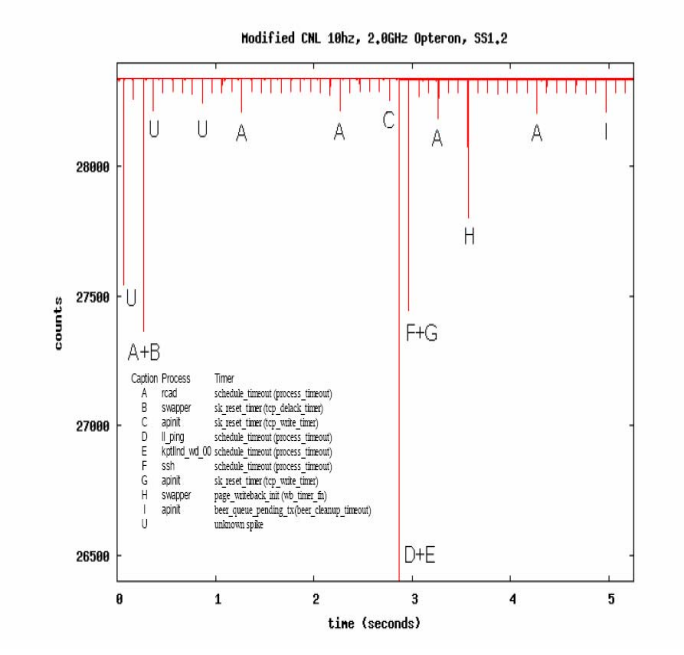
What this graph shows is the regular 10Hz tick of the Catamount clock and the 1 second interval of the Process Control Thread, PCT, doing it's cleanup work. There is very little other indication of an application losing cpu cycles to the Catamount OS.

In contrast to this you can see a Linux operating system that has been stripped of most services still has plenty of activity that interrupts an application.

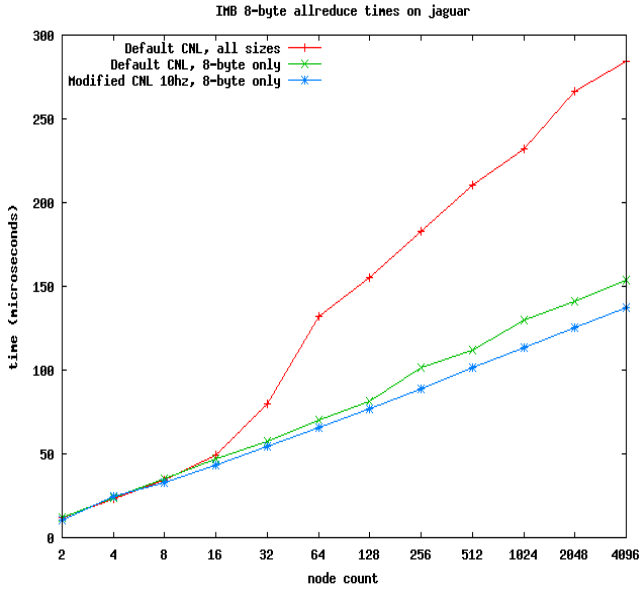


The Linux clock in a 2.6.16 kernel is still ticking at 250Hz. This causes the thick red slab at the top of the graph. There are a number of other interrupts and the spikes of the time this takes from the application vary widely, but clearly many cycles are not available to the application and the potential for barriers and messages to arrive during this time and forcing delays in responses from the application were an early indication of issues with Jitter.

We have been looking at each interrupt and deciding which interrupts to delete or modify in a compute node environment. An early view of this work is shown below.



This Linux 2.6.26 kernel has a 10Hz clock and has many of the longer interrupts removed. There is still work to be done, but the effects of this work can now be seen at scale in some of the microbenchmarks.



This slide shows a clear straight line in blue below – taking less cpu time on the benchmark – than the default CNL kernel. Interestingly the difference is not apparent until between 128 and 256 nodes and become a marked difference at 1024 nodes. This is a clear indication

that the effects of scale are difficult to discern without sufficient number of nodes.

5. Portals

A major focus of the CNL performance work has been in Portals. As previously mentioned the Linux driver was not tuned for supporting applications. This is not as odd as it might seem. The Linux driver is used on Service I/O nodes to connect to disk devices which have extremely long latencies and are mostly reading and writing very large blocks of data. An application node needs to have a driver that is extremely fast at all message sizes and is tuned to be ready for a new message as quickly as possible. This tuning work is expected to help improve Service I/O node performance as devices and other connections are added to those nodes.

All the new compute nodes are multi-core. The expectation of a quad core compute node in 2007 makes it extremely important that the Portals driver be threaded so that multiple requests and interrupts can be handled simultaneously. This work is mostly complete and we can already see improvements in the dual core performance of CNL.

Memory management is very different in Linux than Catamount. This combined with differences in message management have made this an important area of change and optimization. The changes in this area to date are probably among the most important and most visible.

Catamount tuning of latency and bandwidth over the past several years have not been matched by the Linux Portals driver. The Linux driver was about 20% slower on zero byte MPI messages and gave away 15% of bandwidth to Catamount. Initial work on memory management immediately retrieved 50% of the difference in zero byte latency. This work continues to focus on improving the driver performance and comparing measures to the Catamount driver. As with any tuning effort the changes begin to come more slowly and in smaller sizes as the tuning continues. We expect to work for every 100 nanoseconds going forward.

6. I/O

Our initial look at I/O performance shows that CNL has a very large advantage in small, less than 1MB I/O requests. CNL also shows rather dramatic improvements in the directory and metadata operations. These improvements are likely due to caching and the Linux Lustre client support. There is a small advantage that Catamount has with larger I/O requests at the 1MB and 4MB I/O request size for a single width file system. A file system striped across 4 OSTs shows a Catamount advantage only at 4MB. These differences will bear some investigation as the project moves forward.

7. Application Results

We have been running applications at scale with some of our recent changes for a short time. We are quite encouraged at the improvements from our first encounters with applications at scale.

Some simple results at a moderate scale on GTC and MILC run in mid April show modest differences in performance – within the acceptable range of 10% slower than Catamount, which is our initial target.

Application	# Processes	% difference SC	% difference DC
GTC	512	-2	-2
-	1024	-2	-2
-	2048	-	+1
MILC	512	-10	-4
-	1024	-10	-5
-	2048	-	-4

One of the applications that had shown a 2x slowdown with CNL is POP. POP is running much closer t
o Catamount now. We are still looking at differences in performance, but at less than a 15% difference we are feeling that we have a much better chance of improving – perhaps getting faster than Catamount.

Application	# Processes	% difference SC	% difference DC
POP Step/Total	1000	-3	-9
Baroclinic	1000	0	-13
Barotropic	1000	-7	-2
POP Step/Total	2000	-10	-7
Baroclinic	2000	-1	-15
Barotropic	2000	-16	-3
POP Step/Total	4800	-1	-14
Baroclinic	4800	-4	-10
Barotropic	4800	0	-9
POP Step/Total	8000	-	-13
Baroclinic	8000	-	-12
Barotropic	8000	-	-13

Application	# Processes	% difference SC	% difference DC
POP Step/Total	10000	-	-14
Baroclinic	10000	-	-16
Barotropic	10000	-	-14

The application LSMS is a often used code on XT systems. This code shows very little loss of performance on CNL.

Application	# Processes	% difference SC	% difference DC
LSMS bcc_Fe_1024	1024	-4	-4
bcc_Fe_2048	2048	-	-2
bcc_Fe_4096	4096	-	-2
bcc_Fe_8192	8192	-1	-1

The S3D application was chosen because it has some I/O requirements. This application shows some improvements but a complete understanding of the performance of this application is not complete.

Application	# Processes	% difference SC	% difference DC
S3D	**1024**	**0**	**+6**
-	**2048**	**+13**	**-4**
-	**4096**	**-2**	**+11**
-	**8192**	-	**X**

About the Author(s)

Dave Wallace is Technical Project Leader of XT. He can be reached at dbw@cray.com. Dave sometimes makes the mistake of letting Jim Harrell work on his talks. Jim Harrell is Director of the Software Architecture Group. He can be reached at ejh@cray.com.