Compute Node OS for the XT3

Jim Harrell
Cray Inc

ABSTRACT: *Cray is working on different kernels for the compute node operating system. This talk will describe the rationale, requirements, and progress.*

KEYWORDS: Light Weight Kernel, Linux, XT

1. Introduction

The talk upon which this paper is based was designed to discuss the purpose and issues around a small project working on common Compute Node Operating Systems, CNOS, for Several Cray products. The goal is to apprise the audience of the status of the work and next steps, but also to ensure that there is an understanding of the complexities that this project faces. For the purpose of this discussion we will use the CNOS term for the kernel that runs on a compute node. There are a variety of other terms like Light Weight Kernel, LWK, that would be equally useful – but rather than use multiple terms in a single discussion we will use just the CNOS term to avoid confusion. This talk was broken into a number of sections. We will provide an overview of each section.

The basis for a discussion about Compute Node Operating Systems, CNOS, is built around the software Architecture of MPP style systems. The first segment of this paper will review the Architecture to re-establish the importance of the CNOS in the System and help to define the functionality that is required of the CNOS.

The next section will describe a rationale and set of requirements for a CNOS. This project was started because there is a set of new hardware requirements in the future that will be difficult to meet with the current CNOS. There is also a set of software features that are requested to provide a broader scope of applications for XT. These software features are considered difficult to add to the current CNOS. Together these new requirements started a review of options for a CNOS in the near future.

A review of the current CNOS, Catamount, provides the basis for judging alternatives. Catamount is well understood and has set the standard on XT for what a CNOS can be expected to do.

The current work on an alternative CNOS has focused on a Linux kernel. This has caused a lot of discussion both positive and negative. The use of a Linux kernel for this project is not to be construed as an intention to run full Linux on compute nodes. The Linux kernel is modified to only support application processes. This differs sharply from what a Linux user would expect on a standard server. In general, talking about CNOS and Linux appears to be less productive than talking about a different CNOS without explaining its derivation. We may actually pursue this approach in the future as it helps focus on the purpose of the CNOS.

There are some other alternative CNOSs that could be used. Several are either under development or currently more prototypes than a kernel Cray could use in a production environment. However, there are good reasons to track the research and development in this area. Promising developments might be useful in the future and should not be ignored. We will review a few of these CNOS possibilities.

Finally, we will discuss the current status of the project and the next steps that are being pursued. The project does have some commitments and these will be ddiscussed.

2. A Review of XT System Architecture

The basic organization of XT is very familiar to anyone who has worked on an MPP in the past decades. An MPP is a form of distributed system where the operating system services are provided from a set of nodes that have complete systems installed including the servers for all the distributed services. These nodes are called Service Nodes in the XT Architecture. These nodes are specialized to provide specific functions and to scale specific functions, such as the number of login nodes to scale the access for individual users or the number of I/O nodes to scale the access and quantity of disks for the system. The Compute Nodes, where all the user application processes are run, have a minimal kernel that provides a small set of functions and uses the Service Nodes for all the application processes' service requests – such as I/O.

3. Compute Node Definition

A definition of a Compute Node is often based on describing what the node does not do as opposed to what the node does. This is because a Compute Node is designed to provide support only for an application process and not for all the services that a normal operating system might provide. For instance, a Compute Node will support virtual memory because all processors today implement memory support using Virtual memory techniques. However, Compute Node Operating Systems will not support Demand Paging because the cost of the I/O and the performance lost for a distributed application because of the time involved in managing the demand paging.

A Compute Node has the following characteristics:

- Provides support for the machine dependent aspects of the node
- Supports the High Speed network (HSN) connection – HSN is "allocated" to application usage
- Provides support for application processes allocated to the node
- Provides limited set of system call functions – most system calls forwarded to Service nodes
- Small memory footprint and CPU utilization (when not servicing requests)

Overall, a Compute Node Operating System takes little of the processor time available and performs the minimum set of functions required to support application process(es) assigned to the node. All of this ensure the forward progress of a distributed application – which is the purpose of the system.

4. Rationale for a Different Compute Node Operating System

There are three basic components of the rationale for a different CNOS – changes in processor/node technology, new features and functions required in the software, and the ability to leverage a different CNOS between multiple Cray products.

There are changes in Socket technology that are adding more and more cores per socket. The ability to take advantage of these additional cores does factor into a decision about a new CNOS.

There are a number of applications that would benefit from some additional CNOS functionality. These include, networking access, support for new or different file systems, support for Global Arrays, and support for OpenMP. Support for OpenMP is obviously related to the number of cores available in a socket. OpenMP support would be limited to within a node.

Cray has other products that use Compute Node style Operating Systems. It would be a benefit to Cray to be able to share the development of a CNOS across several products. It would also help move Cray software towards the longer-term goal of Adaptive SuperComputing if the CNOS could be operated and managed in the same way across products.

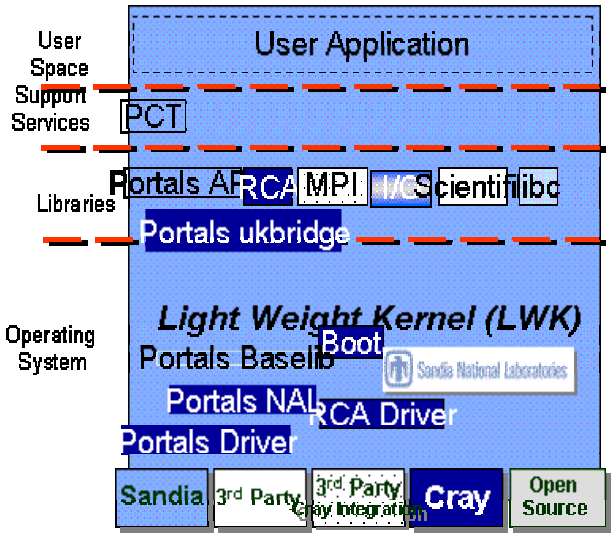
5. Cray Requirements for a New CNOS

The requirements for a new CNOS are based on the current CNOS, Catamount. Catamount has been successful at setting the standard by which a new CNOS will be judged. The areas Catamount has most influenced are boot time, application start time, and I/O performance. Boot time is an area of influence because Catamount can boot on several thousand nodes in seconds. The second major area of influence is application start time, because the start up time of an application is also a matter of small numbers of seconds for thousands of nodes. The area of I/O performance is a bar set by Catamount that any new CNOS must clear. Catamount support for I/O is sufficient for most applications and scales well. Catamount has also had influence in the number of nodes a CNOS can be expected to support. By supporting more than 10 thousand nodes the expectation is that a compute node operating system should be able to support 30 thousand nodes.

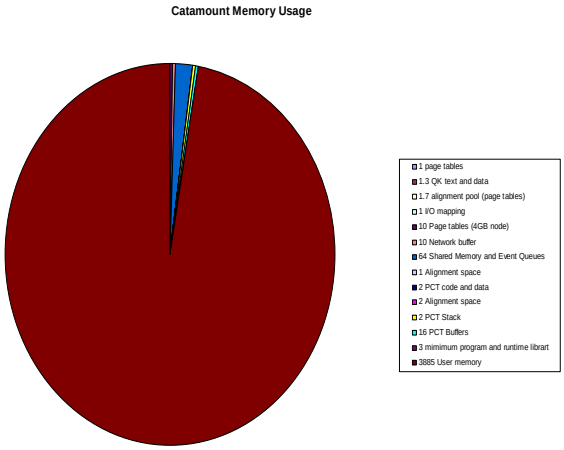
There are features that are required or desired by customers for their applications that add to the capabilities now supplied by Catamount. These are support for sockets or networking, support for SHMEM, UPC, CAF, and Global arrays, signal support, and as already mentioned support for OpenMP. Each of these additional features brings some issues as well. Support for networking means that each Compute Node will have to have an IP address if the implementation adds a network stack to each compute node.

6. Catamount Review

The Catamount kernel is currently used on all XT3 systems. The basic structure of Catamount is similar to many operating systems. There is a low level machine dependent kernel which supports the hardware and system call functionality. There is a set of libraries and support services. In the case of Catamount the support services are reduced to just the Process Control Thread (PCT) which provides support for the application process. The following diagram shows the structure of Catamount.



Catamount memory footprint is minimal. Much of the space not left for user process(es) is buffer space for Portals. The following graph shows a 4GB node and the Catamount kernel with all components broken down to show space used. Obviously the majority of the space is left to the user process(es).



7. Catamount Positives

Catamount obviously has positive attributes. Just to ensure they get

mentioned together – they are – scaling, CPU usage, small memory foot print, support for MPI, and a "mature" code base. There are several existing systems with 5 thousand and 10 thousand nodes. This is a proof of existence of scaling. The CPU utilization of Catamount has been documented to be less than .0008%. This is the percentage of time used by the operating system when the application is making no system calls. This is substantially less than many other systems that use a CNOS. The size of the memory footprint is about 115 Megabytes. This includes all the Portals buffers. The Portals buffers are over half the total size of the Catamoount kernel. In the past microkernels had a suggested size of about 10% of memory. Catamount is at this size even for a single Gigabyte memory size. Catamount does benefit from the growth in memory sizes. Microkernels had memory sizes of 128 Megabytes, which made 10% a much smaller number. Catamount supports MPI applications that are the greater proportion of HPC applications currently. The Catamount code base is maturing rapidly as the systems provide it with more run time.

8. Catamount Issues

All software has issues. Catamount is no exception. The following list describes the current issues that are being pursued.

- Catamount was developed to support a set of MPI applications. In order to broaden the application set a number of features need to be considered.

- Sandia and Cray are sole sources for Catamount development and support. The design and development of any new feature or fix is done by one or the other organization.

- Support for new processors and devices have to be added after the hardware becomes available – Support for other software is available at processor or device release.

- Fully productizing Catamount is substantial work. Logging and debugging are still difficult problems

- Catamount design has some limitations that will be difficult to overcome:
  - File System support
  - Application Network Support
  - Many system calls missing
    - User/system time
  - Debugging support (kernel)
  - Signal support – for exit
  - Multiple threads in the kernel
  - Lack of support for multi-core or SMP
  - Robustness problems

- Sandia had not been planning future versions of Catamount to support 4 cores

- The support for more than 1 or 2 processors/cores is required. *Productizing dual-core Catamount has proven problematic. The asymmetry of the design results in unpredictable performance.

9. Catamount Extensions

Any time there is a discussion of issues with software, such as with Catamount, it is necessary to acknowledge that some of the issues could be mitigated by adding the functionality. There have been some discussions about adding some features to Catamount and this section describes a few of those features.

First among the features to be added to Catamount is support for multiple cores, beyond just two cores. A project is underway at Sandia to use the same Master/Slave implementation as exists with Catamount Virtual Node (CVN) and extend it to more than two cores. There are obviously positive and negative points that could be made about the approach, but the work is underway and will provide a benchmark for any other implementation.

It is also possible to implement other features in Catamount such as OpenMP. This method of implementing OpenMP might provide a lighter weight implementation than a full operating

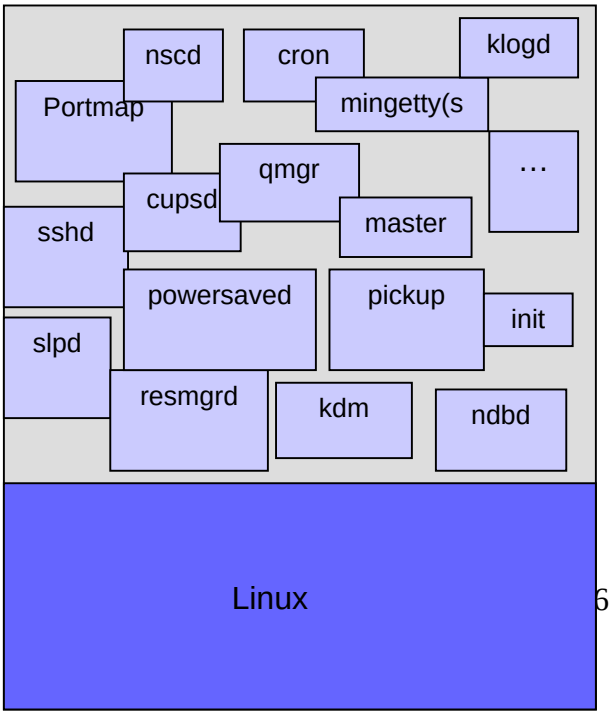
system support for OpenMP. Similarly, it may be possible to support Global Arrays by adding the feature to Catamount.

There is a reason for concern in adding many features to Catamount. First, new code takes time to mature and the lightweight nature of Catamount could easily be lost if too many features are added. It is also disheartening how often new features take longer to complete than expected, create unexpected problems, and don't perform as well as an existing implementation.

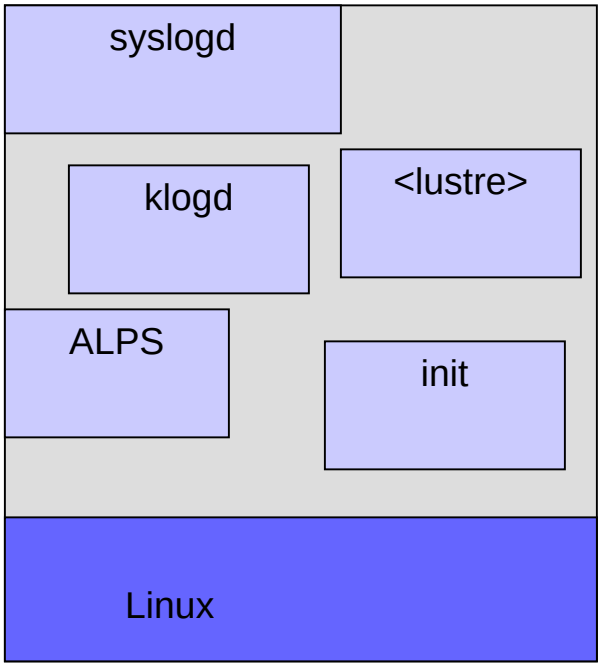
10. Linux Review

The current plans for a Linux based compute node operating system is not Linux as is available for use on a server. The Linux kernel for a compute node is configured and modified to support just the application processes sent to run on the node. This fits with the description of a compute node operating system but does not fit with expectations for all the services that might be available on a Linux system. It would be better, to not discuss this CNOS as Linux, but perhaps as Linux based.

A standard server from SUSE has a number of demons and services that are started by default. The following is an example picture – that is not completely accurate – which demonstrates the kinds of services.



The current version of compute node Linux has only the processes needed to run application processes, a few test services to help with debugging, a few logging services, and Lustre. Lustre is a special problem, as it adds more services and demons than would be expected. However, to date there have not been substantial problems reported as a result of Lustre services.



This is not proof that Linux can be used for a compute node operating system, but it is proof of Linux's configurability.

11. Linux Positives

There are reasons why Linux would be attractive as a compute node operating system. A number of the reasons are listed here.

•Linux is an accepted standard for x86 architectures (for Unix)

• Almost all of HPC is using Linux or a Unix variant

•Development is "Shared" among the Linux community
  •HPC does have a limited following and limited influence

•Linux supports a broad variety of applications and application models
•New hardware and new features are available in Linux before other systems

•Linux is still a lightweight architecture

12. Linux Issues

There are a number of concerns and issues with using Linux as a compute node operating system. None of the problems appear insurmountable, but will require some work and changes to make the Linux kernel a suitable alternative. The following list of concerns are currently being pursued and work is going on to understand what changes may be required.

•Linux has no concrete examples of its use on an HPC system at 5K or 10K nodes

•Linux as a compute node will not provide all the services that a standard Linux system would provide
  •Most services and demons would be removed to ensure proper performance

•Changes to Linux to provide a better compute node LWK will have to be supported outside of the distributions

•Linux memory management does not handle alternate page sizes as well as would be desired
  •Some "patches" from other vendors may help, but the patches would then be a support burden

•Linux scheduler and clock tick features will likely need to be modified to optimize the performance

•Portals performance issues in Linux need to be investigated and fixed

[Portals is a hold over from Catmount – it seems that this imposed requirement]

•Root file system is required for Linux – but
  •Use of a RAM file system takes memory
  •RAM file system would have to be created and maintained
  •RAM file system would limit shared library usage
  •Shared root – like NFS or Lustre may simply not work at 30K nodes

•Work in Programming Environment – MPI, libraries, and tools
  •Although much of this work would make us more standard

•Networking adds complexity
  •Each node needs an IP address
  •ARP and other noise needs to be eliminated from the HSN network
  •NAT, RSIP, or alternative for routing outside the machine

•Memory footprint is larger. How to constrain it?

•User credentials – Need to support some mechanisms

•File Systems – Need to aid Lustre client scaling to 30K nodes

13. Linux – Decisions and Investigations

Using Linux as a compute node operating system does not change the fundamental architecture of the system. The use of service sand compute nodes remains the same. A goal of the current project is to be able to use the CRMS infrastructure without changes. This has been successful so far and enables easy testing of Compute Node Linux, CNL. The initial test vehicle has more functionality than Catamount, but substantially less than a full Linux distribution.

The ALPS scheduling package is being used instead of CPA/YOD/PCT. ALPS is the placement scheduler for Linux compute node operating system support on Black Widow. Job scheduling is still supported using PBS. LSF and MOAB are planned.

Lustre is the file system supported in the prototype. The Lustre organization is similar to the standard Linux cluster configuration, except that the compute nodes only support clients. There is work to scale and tune Lustre. If Lustre clients prove untenable it would be possible to use libLustre as an alternative.

There is a standard IP based network set up on the High Speed Network, HSN. The hosts and IP addresses are fixed and generated automatically. This simplifies configuration and administration. The routes are static and there is no use of ARP. This ensures than there will not be unnecessary traffic on the HSN. Currently we are looking at RSIP for a way of providing gateway support to external networks.

The kernel boots on a compute node in 15 seconds. This needs some attention. We will be looking at a number of changes including reducing the flow of console messages at boot.

The prototype uses a RAM based root file system. The contents are currently limited to init, apinit (for ALPS), the RCA kernel module, sshd, and a few debugging tools. There has been some experimenting with shared libraries, but currently the plan is to force applications to load all the libraries during the application build.

User credentials, user ids etc. are currently managed by ALPS. This limits the use of some library requests to change users and groups, but allows the prototype to avoid the use of LDAP and NIS.

There is work to do to improve the performance of Portals specifically with the copying of data and page management. There is some potential to use GART and MRT to help with the memory access.

Scaling studies have just begun. There has been a little testing at about two thousand nodes. A number of measurements need to be taken including:

- Job start
- I/O performance
- Network performance
- OS jitter/noise on nodes
- Boot times
- Console/CRMS load
- Memory usage/availability
- Syslog loading
- Lustre client cpu and memory utilization.

As these measurements are completed the next set of project requirements will be put into place. At this point the initial numbers look promising, but are not complete.

14. Other Alternatives

There are other alternatives to Catamount and Linux. There is a growing community of researchers and developers who are looking at compute node operating systems specifically for use with large scalable systems. This is a change from the past and is quite welcome. A few examples of the work going on are virtualisation layers such as Xen. This is quite popular in a number of research communities, but may take some time to mature. DOE supports a group called FastOS. The FastOS group is looking at several ideas including library based runtimes and some versions of Linux.

One alternative to a single Linux compute node operating environment would be to have several Linux environments with different capabilities. “Fatter and thinner” Linux compute node environments could offer a variety of services at different scales for different application requirements.

One potential in the future is to have a variety of compute node operating systems running on the same distributed

system. This would require careful support for configuration and scheduling but would enable alternatives to be used as needed or required by different applications. In the long run the purpose of a compute node operating system is the support of application processes – and perhaps no one operating environment is sufficient.

About the Author

Jim Harrell is Director of the Software Architecture Group. He can be reached at ejh@cray.com.