HPC Linux Project Update

Lynne M. Johnson, SGI, lynne@sgi.com

ABSTRACT: This paper addresses current progress and future plans for SGIs Linux for large systems. It addresses HPC Linux architecture and engineering efforts that are underway including Open Source development and collaboration.

1 SGI and Linux

Future generations of SGI Scalable node platforms will be based on commodity Intel chips. SGI will use Linux as the primary operating system for Intel based products. The benefits of this strategy are many. We will run a commodity OS. We can focus engineering efforts on our ability to innovate. Running a standard distribution of Linux enables us to expand the applications catalog available to our customers. We will be the IA64 Linux reference port platform for many important ISVs.

By relying on the standard Linux kernel, we can use collaborative development efforts, and gain all the benefits of an open source solution including more people contributing to the code, and more people running and testing the code.

SGI is placing an emphasis on the IA-64 architecture, and working with Intel and others on the Trillian project.

To support our HPC customers, we will place a strong emphasis on scaling the SNia platforms. The SNia will be the most scalable IA-64 platform on the market. We'll be leveraging relevant IRIX technologies for the IA-64 platform especially in the following areas:

- Scalability (ccNUMA architecture, CPU scalability, memory scalability)
- High-bandwidth I/O
- High performance scalable file systems
- High performance compilers and MPI
- Resource Management
- Storage Management

.

1.1 Clusters and Partitions

A cluster is a group of machines connected by a network fabric. There are three primary types of clusters.

The first type is the "availability" cluster, which is used primarily for non-stop applications - for example web serving. These applications run well on commodity IA hardware, and are usually enterprise-class applications.

The second type of cluster is the "throughput" or "capacity" cluster. This type of cluster is used to increase the compute power of a system, while not necessarily having applications that take advantage of the whole cluster. Most applications that run on throughput clusters run only on one node, i.e. they are not "cluster-aware applications." More work can be done through the whole systems by scheduling jobs across the cluster. Throughput clusters can often be created quite effectively using commodity hardware. There are many third party applications of this type that are currently available.

The third type of cluster is the "capability cluster" which allows you to use the aggregate performance of all the nodes in the cluster for a single application. Applications that run on capability clusters are HPC type applications, that are "cluster" aware (generally MPI or parallel database apps). They require most if not all of the nodes on a system to reach a desired performance level. Applications of this type are a mix of ISV code, and in-house development.

A partitioned system is an SNia system that is connected by NUMAlink fabric which provides RAS features of a capability cluster, and extremely high-performance interconnect and scalability. The SN-IA system uses partitioning software to run multiple images of the Linux system within a single system, and uses cluster and partitioning software to tightly couple the systems together.

1.2 HPC Linux

The goal of the HPC Linux project is to deliver the capability performance of the SN-IA platform

while providing Linux application compatibility, and a robust operating environment. The partitioned SN-IA system represents the peak of the capability cluster performance.

1.3 HPC Linux Technical Approach

HPC Linux will build on standard Linux. SGI will contribute to standard Linux where our expertise can help, and we will work with partners on mainstream technology. We will then add high performance differentiators with the goal of supporting extremely high performance parallel applications. We will not fragment Linux. Our programming and operating environments will make it easy for applications to take advantage of our low-latency interconnect – NUMAlink. We will focus on providing high-performance applications and highly reliable systems for first ship. Then we will provide additional features later.

1.4 HPC Linux System Architecture

Linux was originally developed for smaller machines, (1P!) and will scale up in time but its scalability is increasing. We will run core Linux on the SN-ia. Our SN-IA hardware will scale well beyond the capabilities of Linux, and we need to support customers that require more compute power than is available in a single Linux system.

Software for the SN-IA system will be modular as is the hardware. As SN-IA system sizes increase beyond the current effective system size for Linux, we will run the SN-IA as a partitioned Linux system. We will contribute technology to Linux to increase its scalability. We will focus on specific HPC application performance and scale and tune our Linux systems to provide peak performance for specific applications. For example, the first SGI Linux release was focused on database application performance. Details on features for performance and scaling follow.

1.5 Core Linux for High Performance Technical Computing workloads

The fundamental building block for HPC Linux running on the SNia systems is core Linux. SGI has made several advances in core Linux in the areas of kernel development tools and scaling.

Kernel development tools include several features, which are outlined below. All of these features are available on the SGI open source web site, http://oss.sgi.com and have been contributed to Linux.

1.5.1 Linux kernel lock metering .

The Linux SMP kernel uses spinlocks to protect data structures from concurrent, potentially conflicting accesses. The lock-metering feature performs simple "metering", or record keeping, of spinlock usage in the Linux kernel. The **lockstat** command turns metering on and off, and retrieves the metering data from the kernel and displays it in a human-readable format.

This is an essential first step towards identifying which areas are potential bottlenecks for scaling up the Linux kernel.

1.5.2 Kernel profiling

Kernprof is a set of facilities for profiling the Linux kernel. It consists of a kernel patch that implements a number of profiling data collection mechanisms and a device driver for controlling them, plus the user level command **kernprof** that allows a user to configure and control the kernel profiling facilities.

1.5.3 Post/wait synchronization

Post/Wait is an efficient, fast, lightweight sleep/wakeup/timer mechanism used between a cooperating group of processes.

1.5.4 Kernel debugging

The SGI Linux kernel team has also developed tools for dump collection and analysis that help to debug potential kernel problems.

KDB is a built-in kernel debugger for Linux. It is part of the Linux kernel and provides a means of examining kernel memory and data structures while the system is operational. Additional commands may be easily added to format and display essential system data structures given an identifier or address of the data structure. Current command set allows complete control of kernel operations.

kGDB provides a mechanism to debug the Linux kernel using the popular debugger gdb. kGDB is an extension to the kernel that allows a user running gdb on a remote host to connect to a machine running the kGDB-extended kernel.

The user can then "break" into the kernel, set breakpoints, examine data, etc. very similarly to how one would use gdb on an application program. One of the primary features of this patch is that the remote host running gdb connects to the target machine (running the kernel to be debugged) during the boot process. This allows debugging to begin as early as possible.

1.5.5 Kernel Crash Dumps (lcrash)

The Linux Kernel Crash Dump project is designed to meet the needs of customers wanting a more reliable method of examining system failures after the machine recovers. This project contains kernel and user level code designed to:

- Save the kernel memory image when the system dies due to a software failure;
- Recover the kernel memory image when the system is rebooted;
- Analyze the memory image to determine what happened when the failure occurred.

The dump is recovered with an application called **lcrash** (Linux Crash) once the system boots back up, before the swap partitions are mounted.

1.5.6 ST

The Scheduled Transfer Protocol (STP) is an ANSI specified connection-oriented data transfer protocol. The protocol supports flow-controlled Read and Write sequences and non-flow-controlled, persistent-memory Put, Get and FetchOp sequences. For all sequences, small control messages are used to pre-allocate buffers at the data Destination before the data movement begins, thus allowing the data to be moved immediately from the physical network into the end device's memory. This characteristic of the protocol results in very high-bandwidth data transfer with minimal host CPU usage via the flow-controlled Read Write sequences available through the kernel, and/or very low latency messaging using non-flow-controlled persistent-memory Put, Get and FetchOp sequences available directly from user space.

1.5.7 NUMA support

Non-Uniform Memory access NUMA is an architecture where the memory access times for different regions of memory from a given processor varies according to the "distance" of the memory region from the processor. Each region of memory to which access times are the same from any CPU, is called a node. On such architectures, it is beneficial if the kernel tries to minimize inter-node communications. Schemes for this range from kernel text and read-only data replication across nodes, and trying to house all the data structures that key components of the kernel need on memory on that node.

In 2.2 and 2.3, Linux has support for handling memory holes by using PG_skip and freeing unused mem map entries. This still does not release the unused space held by the page allocation bitmaps. There is much more work to do in this area.

1.5.8 Bigmem

Linux is currently limited on IA-32 platforms by a 4Gb virtual address space size, that it carves up between user address space and kernel address space, leading to the limitation that it can only support maximum 2Gb physical memory on an ia32 system. This patch allows Linux to use approximately 3.8Gb of physical memory on ia32 platforms, while preserving 3Gb user space.

1.5.9 Raw I/O enhancements

Raw I/O enhancements provide high-bandwidth low-overhead SCSI disk I/O capabilities. The asynchronous I/O (AIO) facility implements interfaces defined by the POSIX standard, although it has not been through formal compliance certification. This version of AIO is implemented with support from kernel modifications, and hence is called KAIO to distinguish it from AIO facilities available from newer versions of glibc/librt. KAIO is integrated to work well with Raw I/O.

Preliminary experience with KAIO has shown over 35% improvement in database performance tests. Unit tests (which only perform I/O) using KAIO and Raw I/O have been successful in achieving 93% saturation with 12 disks hung off 2 X 40 MB/s Ultra-Wide SCSI channels. We believe that these encouraging results are a direct result of implementing a significant part of KAIO in the kernel using split-phase I/O while avoiding or minimizing the use of any globally contented locks.

1.6 Partitioning

Partitioning provides the ability to run a large SNia system as a partitioned system with extremely low-latency NUMAlink as the interconnect. Partitioned systems are designed so a failure in one of the partitions will not affect the operation of the other partitions. If a partition goes down, the system can be reconfigured to remove the failing node, and reboot the failing partition while that node is repaired or replaced. The other partitions remain running.

Basic partitioning support will provide the ability to install and configure a partitioned system, independently boot or shutdown one of the partitions, and to reconfigure partitions through simple reboot. The partitioned system provides a very flexible machine, which can easily handle configuration changes, rolling upgrades, or support different sizes of SSIs in a single system.

Partitioning will provide a number of different mechanisms for high-speed communication between processes on different partitions: a network driver, a block transfer engine (BTE), fetchops, and shared memory. All the mechanisms will be implemented as device drivers or loadable modules, to minimize if not eliminate their modifications to the standard Linux kernel.

The partition network device driver is a driver that provides standard network access to other partitions. All communication utilizes the underlying NUMAlink fabric instead of a physical network device.

The Block Transfer Engine (BTE) is a SN-IA low latency high bandwidth hardware memory copy mechanism. The partitioning software allows a process to register its memory so that the BTE can work across partitions. The user will initiate the BTE via a new lightweight Linux system call. SN-IA Fetchops operations will initially also be supported via a light-weight Linux system call.

A Shared Memory Device Driver (SMDD) will implement hardware memory sharing across partitions. Using an export, import, and memory map mechanism, cooperating processes on different partitions can directly access memory between partitions. Once this is set up, the memory is accessed with load/store operations like any other memory.

The SMDD will have an interface similar to System V Shared Memory segments.

Because the SMDD requires lowering memory firewalls between partitions, and accesses are directly from user space, the fault containment capabilities of the partitioning software may be compromised.

While the ultimate goal is to provide both the SMDD and the complete fault containment capability, initially sites with rigorous fault containment goals may want to limit access to the SMDD. The high performance BTE and Fetchop mechanisms do not have this initial limitation.

1.7 System Administration

Efficient system administration is a key requirement for partitioned systems. Our System Administration effort will be a combination of collaborations for basic system administration (including user admin, system setup and configuration) and in-house tools developed to support our advanced features such as partitioning

and CXFS, the clustered file system. Our goal is to provide a single system view with easy monitoring, tracking, system upgrades, and an easy mechanism for the user to track jobs, status, etc.

1.8 Resource Management

Effective resource management is critical to providing efficient utilization of partitioned SNia systems. System resources, such as CPUs and memory, need to be scheduled and managed in a way that promotes satisfactory job throughput and system utilization. Often, there is a trade-off between these two goals. For example, job throughput can be increased by over allocating resources, but this waste of resources decreases system utilization. To further complicate things, a general-purpose system requires a resource scheduler that is extremely flexible and highly configurable.

The initial implementation of a partitioned SNia system will be based on the beowulf cluster. Users will access the system via the headnode. The head node controls the cluster. All jobs will be submitted to the compute nodes through a Workload Management (WLM) system. The WLM will schedule and track jobs. Users will not have direct access to the compute nodes.

Our current work in the resource management arena is focused on three areas – Accounting, introducing the "job" concept, and porting array services from IRIX (called cluster sessions for Linux. We are also working with third party workload management systems to ensure that our solutions are well integrated.

1.8.1 Accounting

Accounting is very important for many HPC customers. Linux accounting is currently quite limited. SGI has entered into a collaboration with the ACL group at Los Alamos National Laboratories to port SGI Comprehensive System Accounting (based on Cray System Accounting) from IRIX to Linux. CSA performs job level accounting, as opposed to the more familiar process level accounting.

1.8.2 Jobs

A job is a group of related processes all descended from a point of entry process and identified by a unique job ID. A job can contain multiple process groups, sessions, or array sessions and all processes in one of these subgroups are always contained within one job. The job acts as a unit of containment for resource limits and accounting information.

The job, as defined on IRIX (as of version 6.5.7) has the following characteristics:

- A job is an inescapable container.
- Each new process inherits the job ID and limits from its parent process.
- All point of entry processes (job initiators) create a new job and set the job limits appropriately.
- Users can raise and lower their own job limits within maximum values specified by the system administrator.
- The job initiator performs authentication and security checks.
- The process control initialization process (init(1M)) and startup scripts called by init are not part of a job and have a job ID of zero.

Future work on resource limitation capabilities on clusters requires a job container that is cluster-aware. A job container that is capable of operating in a cluster environment provides the means for improved application monitoring and control.

We will provide a generic solution for implementing process containers in Linux.

1.8.3 Cluster sessions

Mechanisms that are typically used to manage multiple related processes (e.g. process groups, terminal sessions) are limited in scope to a single machine. As a result, mundane tasks such as killing a job or accounting for all of its resource usage can become very difficult when the job runs across several machines. Some means of correlating related processes on different machines is required. We will provide this function in with the notion of a cluster session.

In formal terms, a cluster session is a set of processes all related to each other by a single unique identifier, the cluster session handle (csess).

Cluster session handles will be unique across the system. Each machine that is to run part of the job has a local cluster session id which gets "upgraded" to a global csessid. The process on each machine that started the new session is free to fork off any number of children to do the required work. These children will all have the same cluster session id and can be correlated with each other for administrative tasks such as job control or accounting.

The cluster sessions daemon needs to find all of those related processes and act upon them. The cluster sessions daemon performs several different tasks:

- It allocates global array session handles
- It knows the current cluster configuration and can provide that information to other commands and programs
- It can determine which processes belong to a particular cluster session and provide that information to other commands and programs
- It can forward commands to all of the machines in a cluster.

1.8.4 Workload Management Systems

There are many Workload Management (WLM) products available for Linux to today.

PBS is a batch software processing system developed at NASA Ames Research Center by MRJ technologies. It operates on networked, multi-platform UNIX environments, including heterogeneous clusters of workstations, supercomputers, and massively parallel systems. The PBS Job Scheduler module was designed to be highly configurable. PBS is currently provided as a source code distribution, enabling further customization of PBS if desired.

LSF is a commercially available WLM that has support for the Linux operating system. It provides solutions for parallel job management and has a customer support infrastructure in place. It also supports multicluster (cluster of clusters) configurations; which is an issue if one wants make a Linux-based cluster to appear more like a single system when it is part of a larger group of systems.

We will work with WLM vendors to effectively support our partitioned SN-IA systems.

1.9 Storage Management

SGI has several efforts going in the storage management arena. XFS (the high-performance journalled file system from IRIX) is being ported to Linux, and is now available in open source. XFS is a highly scalable true 64-bit filesystem. It allows for large file systems, scalable I/O, and quick recovery.

We are also porting CXFS and XVM. CXFS is a clustered file system that allows a file system to be shared by several hosts that are attached to the same storage. CXFS provides tokens for file access, and once the client system has access to the file, I/O can be done at approximately local file system access speeds for many operations.

The Data Migration Facility (DMF) is an extremely scalable hierarchical storage manager that was originally developed for UNICOS systems. DMF provides virtually limitless file system space by automatically saving less frequently user, larger files to tape, and freeing up on-line disk space. DMF’s storage management is transparent to the user. The Data Migration Facility (DMF) 2.6 will be ported in 2001, and the client server version will be supported as part of the DMF 3.0 project.

The Tape Management Facility (TMF) provides and efficient utility for reading and writing tapes. DMF uses TMF for tape mounting. TMF is being ported and will be available.

1.10 Application support

We will provide the most powerful development tools for HPC applications for HPC Linux. We are developing open source compilers for IA-64 including C, C++ and Fortran.

The partition system Shared Memory Device Driver allows for unique "custom shared memory" applications to be built that can apply the entire SN-IA machine to a single shared memory application. Some custom coding is necessary at startup time, but once the processes and shared

memory are set up, the memory mappings have the same access and performance characteristics as any other memory mapping.

We are porting SGI's MPI to the Intel platform. We currently have an IA-32 port done as a proof of concept (with no plans to productize). The port to IA-64 is also underway, and supports all the optimizations and scaling features that are expected of SGI's MPI including MPI2 one-sided communication. We are also working on optimizations for better performance across Myrinet, and NUMAlink including shared memory support, use of the BTE for fast transfers, and a shared memory driver.

In the tools area, there are several excellent third party alternatives available include Vampir for performance tuning, Totalview for parallel debugging, and some other stuff too.

2 Delivering the solution

SGI's Advanced Cluster Environment (ACE) provides a framework in which to develop a software cluster infrastructure for throughput and capability clusters. It is based on the Beowulf architecture. ACE is a fully tested and supported solution which will provide a consistent cluster environment as the SGI IA product line scales from IA-32 to IA-64 and the SN-IA.

2.1 ACE Strategy

The ACE strategy is to release approximately every three months. ACE contains a mix of open source products, internally developed products, and commercial products. We also provide web download of the ACE package open source components at http://oss.sgi.com/projects/ace.

2.2 ACE 1.0

ACE 1.0 was released in February of 2000. It included the following components:

- MPICH (MPI from Argonne)
- MPICH support for GM from Myricom
- PBS (Job scheduler from MRJ/Veridian)
- Installace (internally developed for parallel software installation)
- Interconnect drivers for Myrinet and ethernet
- Linux console (internally developed for something)
- Performance Co-Pilot (PCP) and SGI product that does system performance monitoring
- Test scripts to ensure that the cluster was installed correctly
- Documentation

2.3 ACE 1.2

ACE 1.2 was released in early May of 2000. It includes updates of all the pieces listed above, and evaluation copies of the following third party software products:

- Etnus Totalview
- Enlighten/DSM (for system administration)
- Platform LSF
- Pallas Vampir

It also includes improved user administration guides, improved installation procedures, and Hoover which is a GUI for the VACM (Cluster Manager) product with usability improvements for large node-count systems .

2.4 Future ACE Plans

We plan on including new features and functionality into the ACE package as they become available. Features will include IA-64 support, System administration features, SGI MPI, Storage products, partitioning support, shared memory support for the SNia system, and more.

3 Summary

We have a great roadmap moving ahead. Our plans and projects for HPC Linux will allow us to provide a highly scalable HPC Linux system based on the SN-IA architecture.