White Paper

Linux Scalability for the Altix 3000
A Software Environment for HPC Applications

Table of Contents

1 Executive Summary **3**

2 Introduction **3**

3 Enhancements to the Linux Kernel **4**

3.1 System Scalability 4

3.1.1 Big Kernel Lock 4

3.1.2 Multiqueue CPU Scheduler 4

3.1.3 Translation Lookaside Buffer (TLB) Flush Bottlenecks 4

3.1.4 Directory Cache Contention 5

3.1.5 Least Recently Used List 5

3.1.6 System Clock Contention 5

3.1.7 Discontiguous Memory 5

3.1.8 Datastructures in Local Memory 5

3.2 Additional Tools for Achieving HPC I/O Performance 5

3.2.1 XSCSI 5

3.2.2 XFS® File System 6

3.2.3 The Linux Device File System 6

4 A Complete Environment for HPC **6**

4.1 Advanced Storage Environments 6

4.1.1 CXFS™ 6

4.1.2 SGI® Data Migration Facility (DMF) 6

4.2 Resource Management Tools 6

4.2.1 CPUMemSets 7

4.2.2 System Accounting 7

4.2.3 System Partitioning 7

4.2.4 Clustering Software 7

4.3 Development Environment 7

4.3.1 Message Passing Toolkit 7

5 Conclusion **8**

6 For More Information **8**

1 Executive Summary

The popularity of Linux OS-based compute clusters has catapulted Linux into the HPC market in recent years. Commodity and open-source economies make them attractive to even the most cash-strapped researchers, and they work very well for a variety of applications. But they lack features that have long been enjoyed by users of traditional SMP machines, which employ tuned scalable operating systems on hardware designed to support big computing as well as big I/O. It seemed that while clusters could solve some HPC problems, there was room in the world's research labs for large scalable systems.

The SGI® Altix™ system was designed to bring the performance advantages of a robust shared-memory architecture into an industry-standard, open-source Linux environment. Built around SGI's well-proven NUMAflex™ architecture, the Altix is based on Intel's Itanium® 2 processors and a Linux implementation tuned specifically for applications with big compute and I/O requirements. With hardware support for transparent cache-coherency for up to 512 processors, the engineering team was faced with the challenge of providing the industry's most scalable Linux environment. Additionally, the team sought to allow users to create "superclusters" by combining those large Single System Image (SSI) nodes across a high-speed interconnect and supporting shared memory across them.

Of course, clusters can physically grow to include virtually infinite numbers of processors; since each small node runs its own copy of Linux, large-scale OS scaling isn't an issue. Whether or not a user gets desirable performance depends on the application: can the algorithm and data effectively be split up into chunks small enough to fit on the cluster's 2 or 4 processor nodes, and do they require lots of bottleneck-producing communication between the nodes? In a shared memory architecture, no decomposition of the problem or data is necessary. All of the processors operate on all of the memory, so there is no need to move data or instructions across a slow interconnect. And the environment is carefully tuned to reduce bottlenecks within the node, with constructs ranging from the cache algorithms to the TLBs examined for opportunities to streamline multiprocessor performance.

In order to achieve these goals, SGI's engineering team worked closely with others in the open-source community to identify or develop possible solutions for all of the challenges presented by HPC applications. These challenges go beyond the problem of scaling Linux to large processor counts. Research labs today face dramatically growing storage requirements along with larger and more complex systems management issues, and must often tune home-grown codes to run well in increasingly complex and flexible architectures.

In addition to the tools available in the open source community, SGI was also able to draw on its own extensive experience with HPC installations, and a number of system management, storage management and development tools used in those installations were ported to complete the Altix HPC environment.

The purpose of this paper is to describe the features, scalability enhancements, and additional tools implemented in SGI's Linux Environment for Altix. These include scalability enhancements to the kernel, I/O enhancements, storage and system management enhancements, and development tools.

2 Introduction

By the late 1990s, computer scientists and engineers had begun to take Linux seriously as a platform for supercomputing. The fact that Linux and many of its associated applications and tools are available in the open source community, combined with the economics of commodity hardware, made "Beowulf" clusters of inexpensive systems attractive test beds for deploying Linux and open source software in HPC.

While these first generation clusters were well-suited for highly parallel algorithms with minimal inter-node communication, they offered limited capabilities for complex problems requiring large data sets or extensive interprocess communication. During that time period, SGI began development work with Linux and released some early Linux OS-based clusters. This early experience with clusters quickly revealed the limitations of the Linux cluster approach for complex HPC applications.

In addition to the known issues surrounding commodity clusters, Linux in the late-1990's also lacked a variety of features that have proven to be critical for productivity in big-data HPC, such as native 64-bit journaled file system support, tuned programming libraries, and data management tools.

Despite these limitations, it was clear that the open-source, collaborative nature of the Linux operating environment could be of great benefit to scientific users. SGI thus began to work to deploy scalable Linux on its advanced NUMAflex architecture. As a member of the Linux community, SGI has undertaken a significant development effort and worked closely with the open-source community to make Linux a viable operating environment for high performance computing and to help provide the functionality required by HPC users. SGI is contributing to key community projects focused on scaling the Linux operating system, such as the “Linux on Large Systems Effort” and the “Linux Scalability Effort”. Key goals of these activities include:

- Enhancing Linux for improved usability and performance across a broad spectrum of HPC applications, while utilizing and contributing to ongoing efforts within the Linux community
- Contributing technology and resources back to the Linux community
- Maintaining binary compatibility with existing Linux standards

The results of this development effort are showcased in the SGI® Altix™ 3000 family of servers and superclusters. SGI leveraged its extensive experience with scalable, shared-memory NUMA (non-uniform memory access) supercomputers to enhance Linux capabilities for the Altix platform.

3 Enhancements to the Linux Kernel

The Linux kernel for the Altix platform consists of the standard 2.4.21 kernel for Itanium processors (http://kernel.org) with a number of enhancements drawn from the open source community and from SGI’s own development efforts.

The resulting operating system maintains binary compatibility with existing 64-bit Linux software, enabling Altix users to take immediate advantage of existing commercial Linux applications and open source software. The following subsections detail the specific enhancements that were made to increase CPU scalability and improve I/O.

3.1 System Scalability

“Perfect scaling” – a linear one-to-one relationship between CPU count and throughput for all CPU counts – is rarely achieved because one or more bottlenecks introduce serial constraints into the otherwise independently parallel CPU execution streams. The most common Linux bottlenecks, involving contention among processors for access to shared resources like the kernel or cache lines, apply to simple two-processor commodity nodes as well as to NUMA systems such as the Altix 3000 system.They are usually insignificant on smaller systems, but high processor counts make the bottlenecks increasingly visible and problematic.

Much of the work required to scale Linux to 64 processors involved improvements to reduce or eliminate lock contention—a state in which CPUs sit idle waiting for resources that are locked by another CPU. Many of the changes required to improve CPU scaling were available from the Linux community and used by SGI with only minor enhancements.

3.1.1 Big Kernel Lock

Any multiprocessor implementation in which resources are shared needs a way to lock those resources while they’re in use in order to guarantee that only one CPU at a time can update critical shared data. A simple, coarse-grained approach would be to have a single lock on the kernel, and require that any processor accessing the kernel first acquire the lock. Early Linux multiprocessor releases relied on a single lock, the Big Kernel Lock (BKL), as the primary synchronization and serialization lock for the kernel. While this was not a significant problem for workloads on a 2-CPU system, a single coarse-granularity lock is a major scaling bottleneck for systems with larger CPU counts.

Recently, a number of open-source projects have attempted to address the BKL bottleneck. SGI applied several of them in its implementation for Altix. One approach has been the preferential use of the XFS file system, which uses scalable fine-grained locking and largely avoids using the BKL altogether. The 2.5/2.6 kernel's algorithm for process accounting, which uses a new locking mechanism instead of the BKL, was backported, and a large set of changes was applied from the Linux Scalability Effort rollup patch. All together, these changes resulted in a tenfold reduction in some benchmarks in the fraction of CPU cycles spent waiting for the Big Kernel Lock.

3.1.2 Multiqueue CPU Scheduler

The CPU scheduler in the 2.4 and earlier kernels is simple and efficient for uniprocessor and small multiprocessor platforms, but is inefficient for large CPU counts and large thread counts. One scaling bottleneck is due to the use of a single, heavily contended lock to protect the global runqueue. This scheduler has been replaced on Altix with the O(1) scheduler of the Linux 2.5/2.6 kernel, which partitions the single global runqueue of the standard scheduler into multiple runqueues. SGI has added some "NUMA-aware" enhancements and other changes that provide additional performance improvements for typical Altix workloads. SGI continues to track enhancements to the scheduler, and contributes SGI developments back to the Linux community.

3.1.3 Translation Lookaside Buffer (TLB) Flush Bottlenecks

A translation lookaside buffer (TLB) is a CPU hardware structure that maps virtual memory addresses to real physical addresses for recently referenced memory pages; it serves as a quick-reference index to the pages the system is most likely to need. When memory changes make the TLB entries invalid, its contents are flushed and reloaded with current information. Since all processors have shared memory access, a TLB flush on one processor must be propagated across the system. This can be an expensive operation, particularly on systems with large processor counts.

Several changes have been incorporated into the Linux kernel for Altix to reduce the impact of the TLB flush bottleneck. First, the memory management system is designed to minimize the frequency of TLB flushes. Once a TLB flush does happen, the system is able to determine which processors have executed code that makes their TLBs invalid, and only include those TLBs in the flush routine. This can substantially reduce the time required to complete the system-wide flush operation.

3.1.4 Directory Cache Contention

The directory cache is used to cache information on frequently accessed filesystem directories in system memory. A lock is used to synchronize access to this global system resource. In earlier versions of Linux the contention on this lock was minimal, but the 2.4.18 and later versions of the Linux kernel use this lock in a frequently used subroutine that made it a major CPU cycle consumer. This problem was solved by backporting a version of that subroutine from the 2.5/2.6 kernel that employed a finer-grained locking strategy. This change returned contention on the directory cache lock to acceptable levels.

3.1.5 Least Recently Used List

The Least Recently Used list stores an ordered list of memory pages for the virtual memory system. Access to this resource is controlled through a highly contended spinlock. SGI has made a minor but measurably effective optimization to some of the VM subroutines to address this contention. This optimization takes advantage of the fact that for a highly-contended resource, it is often better to double the lock's hold-time than to release the lock and soon thereafter have to contend for ownership a second time. The change produced a 2-3% improvement in AIM7 peak throughput.

3.1.6 System Clock Contention

As with other resources, access to system time is controlled by a lock in the Linux kernel, and this lock is a severe bottleneck to scalability. A mere handful of concurrent user programs accessing the system timers can result in serious performance degradation. This problem was eliminated by incorporating an open-source patch that handles system time locking differently. Under the new scheme, programs can read the timing information without acquiring the lock.

3.1.7 Discontiguous Memory

A significant feature of NUMA architectures like Altix is that each group of processors has local memory plus direct, high speed access to memory in other processor groups. This allows every processor access to all system memory, but also means that memory is physically discontinuous. An initial step in adapting Linux to the Altix platform was the inclusion of an open-source patch called the "discontig" patch, which was developed as part of the Linux on Large Systems Foundry. This patch allows Linux to support NUMA systems such as Altix with discontiguous physical memory.

3.1.8 Data Structures in Local Memory

Because access latencies in a NUMA architecture to remote memory are greater than for local memory, SGI has taken steps to ensure that various data structures used by each CPU are allocated in local memory. The addition of the CPUMemSets package of library and kernel enhancements permits applications to control process placement and memory allocation to obtain similar benefits. This package allows a process to be pinned to a particular CPU or set of CPUs. This is particularly useful on NUMA systems with high processor counts because it allows an application to scale without consuming excessive communications bandwidth between nodes.

3.2 Additional Tools for Achieving HPC I/O Performance

Given the very large datasets common in research environments, efficient data handling and I/O capabilities are critical to overall HPC productivity; indeed, this is a common failing of many commodity clusters. Large-scale shared-memory architectures are designed to handle those requirements well. With Altix, the challenge was to bring those capabilities into a Linux os-based environment.

To allow Linux to achieve the I/O performance required by data-intensive HPC applications, SGI needed to address issues related to SCSI, the file system, and device handling.

3.2.1 XSCSI

The SCSI subsystem in the 2.4 kernel series has been a significant topic of discussion in the Linux development community, as its performance is limited on an even moderately scaling system. Early tests of the Linux® 2.4 SCSI I/O subsystem showed that the demanding I/O needs of HPC could not be met without a major overhaul in this area. While this issue was being actively worked on within the community and has been addressed in the 2.6 kernel, SGI needed an immediate fix for its 2.4-based kernel. SGI applied its XSCSI infrastructure and drivers from the SGI® IRIX® operating system as an interim solution to deliver immediate performance gains.

3.2.2 XFS File System

XFS, an extent-based, 64-bit journaling filesystem that is extremely well suited to the I/O requirements of HPC customers, has been implemented in Linux and is the default filesystem on Altix. It provides enhanced performance and robustness, and its fine-grained locking structure eliminates many of the scaling problems associated with the Big Kernel Lock. XFS combines the ability to support exceptionally large disk farms (many petabytes) with rapid failure recovery and exceptional I/O throughput capabilities.

SGI introduced XFS in 1997, and made a Linux port available to the open-source community in 1999. In both environments, it has been widely adopted for research and commercial use, and has been used for many years in some of the most demanding HPC environments in the world. XFS configurations routinely achieve I/O rates of multiple gigabytes per second.

3.2.3 The Linux Device File System

The Linux Device File System helps Altix handle the large numbers of disks and I/O buses typical of Altix systems. An optional enhancement to the Linux 2.4 kernel, this routine ensures that device pathnames remain persistent across reboots even after disks or controllers are added or removed. This makes the system administrator's job much less complicated, and it is particularly important for systems with many devices.

4 A Complete Environment for HPC

While scaling the capacity and I/O capabilities of Linux are important for HPC, they are not sufficient to create a complete and productive HPC environment. A full suite of development tools, advanced storage capabilities and other services is required. SGI has focused significant attention on ensuring that it can deliver a complete environment to its Linux customers to enable them to be immediately and maximally productive.

4.1 Advanced Storage Environments

HPC environments often require both huge storage capacity and high performance shared access to important data. Through many years of experience with high-performance IRIX system installations, SGI developed a powerful set of tools for data-intensive environments. These have now been ported to Altix, and provide unique capabilities for HPC environments running Linux.

4.1.1 CXFS

Most HPC environments share the same data between multiple systems. For instance, scientists and engineers may perform some aspects of data preparation and analysis using desktop workstations while using supercomputers for detailed computations. Because network file systems such as NFS lack the bandwidth to efficiently handle the huge volumes of data involved, many organizations resort to painful and time consuming copying to get data where it is needed.

CXFS provides data sharing over a storage area network (SAN), allowing multiple computers simultaneous direct access to a common shared filesystem with local filesystem performance. Multiple systems thus share a single data file, and a single copy of the data is maintained. This saves disk space, eliminates the need for expensive network-based file transfers, and reduces version-control problems. CXFS clients are available for systems running the Linux, IRIX, Windows NT®, Windows® 2000 and Solaris™ operating systems.

4.1.2 Data Migration Facility (DMF)

Meeting the growing storage requirements of modern HPC facilities is a significant challenge that in many cases limits the size of problems that can be addressed. Keeping all data on disk can be prohibitively expensive, but archiving data to tape creates inevitable problems with data access and management. SGI's Data Migration Facility (DMF) is a mature and proven solution to this problem.

DMF automatically and transparently migrates data from online storage to less expensive near-line storage according to user-defined criteria. Files are automatically recalled to online storage as they are accessed without user or system administrator intervention, and they always appear as local regardless of media location.

HPC sites that use DMF typically have a pool of online disk storage capable of storing the active data set required by their largest problems. This online storage is backed by DMF to provide a virtually limitless storage pool. Already in use in hundreds of data centers using SGI and Cray® systems, DMF was ported to Altix in 2003.

4.2 Resource Management Tools

System administrators and programmers have found value in controlling and tracking the execution of their jobs since complex multiprocessing systems were first introduced. SGI began providing these capabilities on its IRIX OS-based systems in the late 1980s. To support these requirements for Altix users, SGI leveraged work originally done for IRIX, and also supported popular third party tools such as LSF® from Platform Computing.

This combination of tools gives administrators and programmers flexibility over how and on which system resources their jobs execute. Programmers who want to insert calls directly into their applications can do so, and users with black box binary applications can use runtime tools to control where individual threads execute. Third party applications like Platform LSF provide a global system environment for managing how the system's CPUs and memory are allocated to different sets of users and applications.

4.2.1 CPUMemSets
This kernel facility provides support to SGI provided commands and libraries to specify on which CPUs processes may be scheduled and from which nodes processes may allocate memory. Programmers can make explicit calls to the SGI provided commands and libraries cpuset, runon, and dplace to ensure a particular workload or batch job can efficiently use the available CPU and memory resources, and to help ensure multiple jobs can allocate the resources they require without interfering with each other. This gives users maximum flexibility in resource allocation and can help deliver fast, repeatable run times on mission-critical jobs.

4.2.2 System Accounting
System accounting capabilities are often important for large systems, especially when the system is shared between organizations. Since no standard solution existed within Linux, the Comprehensive System Accounting (CSA) software package was added to the Altix system software to provide this functionality. CSA is an open source collaboration between SGI and Los Alamos National Laboratory that provides jobs-based accounting services. CSA provides methods for collecting per-process resource usage data, monitoring disk usage, and charging fees to specific login accounts according to configurable parameters.

4.2.3 System Partitioning
This capability allows users to partition the system into smaller CPU/memory subsets, each running its own single system image. Using partitions, administrators can effectively divide and isolate resources to maximize resilience and eliminate single points of failure.

4.2.4 Clustering Software
Parallel workloads, such as MPI jobs, can be launched, monitored, and controlled across a cluster using SGI's Array Services software. The Array Services software package contains a library, a system daemon, and a set of commands that enable developers to define and administer cluster configurations and manage the set of jobs running the cluster.

4.3 Development Environment
The development environment for the SGI Altix 3000 system is designed to ensure that users can get the most out of the unique capabilities of Linux and the Altix platform. These tools provide access to global shared memory across cluster nodes, to the unique capabilities of NUMAflex, and to high-performance I/O. Existing Linux software will run on Altix without modification, although a modest optimization effort will allow software to get optimal performance from the underlying hardware.

Development Environment	
Compilers	Intel® compilers for Linux: C, C++, and Fortran GNU compiler collection: C and Fortran 77
Libraries	SGI® Message Passing Toolkit (MPT) Scientific Computing Software Library (SCSL) Flexible File Input/Output (FFIO) CPU sets and memory placement (CpuMemSets) Intel® Math Kernel Library (MKL) Intel® Integrated Performance Primitives (IPP)
Code Development Tools	Etnus® TotalView® Advanced tool GNU GDB Intel® Debugger (IDB)l
Performance and Application Analysis Tools	Performance Co-Pilot™ Pfmon performance tuning tool Intel® VTune™ Performance Analyzer Pallas Vampir® and Vampirtrace

Table 1. Components of SGI's Development Environment for Linux.

SGI supports the major programming models for Linux, including OpenMP™, MPI, SHMEM, and optimized shared memory/MPI hybrid programming models. Both MPI and SHMEM parallel programming models are offered as part of the Message Passing Toolkit (MPT).

4.3.1 Message Passing Toolkit
MPT provides industry-standard message passing libraries optimized for the unique capabilities of the Altix hardware. MPT contains both MPI and SHMEM APIs, which transparently utilize and exploit the low-level capabilities within Altix hardware. These libraries implement an innovative “global pointer” construct that allows jobs to address both local and remote memory regions, crossing node boundaries without a performance penalty.

5 Conclusion

The result of these efforts is a high-performance implementation of Linux that—in combination with the Altix hardware—has demonstrated exceptional scalability and performance, including:

- Scalability to 128 processors in a single system image
- Scalability to 512 processors in a shared memory cluster configuration
- Demonstrated I/O throughput in excess of 2GBs/second
- World record SPECfp® benchmark results for a 64-processor system
- World record STREAM benchmark results for a microprocessor based system
- Leading performance on a wide variety of application-specific benchmarks

To create this implementation, SGI leveraged work from the open source community, its own IRIX toolset, and new developments. SGI will continue to partner with the Linux Community to enhance Linux for the most demanding HPC requirements. At the same time, SGI will continue to extend the Altix system by scaling Linux further, increasing supported supercluster configurations, doubling interconnect performance, and adding graphics capabilities.

As SGI proceeds with additional enhancements, we resolve to ensure binary compatibility with industry standards, and to contribute critical software technologies back to the open source community whenever appropriate. Ultimately, robust HPC solutions should be available to all Linux users.

6 For More Information

Bryant, Ray and Hawkes, John; *Linux Scalability for Large NUMA Systems*; from the Proceedings of the Linux Symposium, July 23-26, 2003.

Neuner, Steve; *Behind the Altix 3000: SGI's New 64-processor, 64-bit NUMA System*, Linux Journal, February, 2003.

Woodacre, Michael; Robb, Derek; et. al. *The SGI® Altix™ 3000 Global Shared Memory Architecture*. An SGI White Paper.

Corporate Office
1600 Amphitheatre Pkwy.
Mountain View, CA 94043
(650) 960-1980
www.sgi.com

North America 1 (800) 800.7441
Latin America (52)5267.1300
Europe (44)118.925.75.00
Japan (81)3.5488.1811
Asia Pacific (67)6771.0290