Linux® Configuration and Operations Guide

007–4633–007

CONTRIBUTORS

Written by Terry Schultz

Illustrated by Chrystie Danzer

Production by Karen Jacobson

Engineering contributions by John Hawkes, Dan Higgins, Robin Holt, Erik Jacobson, Kevin McMahon, Kim McMahon, Dean Roe, Dave Wright

New Features in This Manual

This rewrite of the *Linux Configuration and Operations Guide* supports the SGI ProPack 4 for Linux and SGI ProPack 3 for Linux Service Pack 4 software releases.

Major Documentation Changes

Added information about persistent PCI bus numbering on SGI ProPack 4 for Linux systems in "Persistent PCI Bus Numbering on ProPack 4 Systems" on page 13.

Added information about giving an Ethernet interface a persistent internet address on SGI ProPack 4 for Linux systems in "Persistent IP Addressing of Ethernet Interfaces on ProPack 4 Systems" on page 15.

Added information about partitioning an SGI system running SGI ProPack 4 for Linux in "System Partitioning " on page 22.

Updated information about getting array services operational in "Making Array Services Operational" on page 37.

Removed Chapter 3, “Performance Tuning”. For information on tuning an SGI Altix system, see the *Linux Application Tuning Guide*.

Removed Chapter 4, “NUMA Tools”. For information on NUMA tools, see Chapter 5, "Data Placement Tools" in the *Linux Application Tuning Guide*.

Added information about the location of documentation about Linux kernel tunable parameter for SGI ProPack 4 for Linux systems in Chapter 3, "Kernel Tunable Parameters on SGI ProPack Servers" on page 65.

Removed Appendix A, “Linux Kernel Tunable Parameters”. You can find this information on your system at: `/usr/src/linux/Documentation/sysctl`

Record of Revision

Version	Description
001	October 2003 Original publication.
002	February 2004 Updated to support the SGI ProPack for Linux v2.4 release.
003	May 2004 Updated to support the SGI ProPack 3 for Linux release.
004	August 2004 Updated to support the SGI ProPack 3 for Linux Service Pack 1 release.
005	November 2004 Updated to support the SGI ProPack 3 for Linux Service Pack 2 release.
006	January 2005 Updated to support the SGI ProPack 3 for Linux Service Pack 3 release.
007	February 2005 Updated to support the SGI ProPack 4 for Linux release.

Contents

Figures

Procedures

About This Guide

This guide explains how to perform general system configuration and operations under the Linux operating system used with SGI servers and superclusters. The information in this manual is specific to the SGI Altix 3000 family of servers and superclusters and SGI Altix 350 systems. For information about general Linux system administration, see the “Additional Reading” section of this Preface.

This manual contains the following chapters:

- Chapter 1, "Configuring Your System" on page 1
- Chapter 2, "System Operation" on page 41
- Chapter 3, "Kernel Tunable Parameters on SGI ProPack Servers" on page 65

Related Publications

The following publications contain additional information that may be helpful:

- *SGI ProPack 4 for Linux Start Here* provides information about the SGI ProPack for Linux release including information about major new features, software installation, and product support.
- *SGI ProPack for Linux Start Here* provides information about the SGI ProPack for Linux release including information about major new features, software installation, and product support.
- *SGI ProPack 3 for Linux Release Notes* provides the latest information about software and documentation in this release. The release notes are on the SGI ProPack for Linux Documentation CD in the `root` directory, in a file named `README.TXT`.
- *SGI Altix 350 System User's Guide* provides an overview of the Altix 350 system components, and it describes how to set up and operate this system.
- *SGI Altix 350 Quick Start Guide* guides a knowledgeable user through the installation, setup, and simple configuration of most SGI Altix 350 systems.
- *SGI Altix 3000 User's Guide* provides an overview of the architecture and descriptions of the major components that make up the SGI Altix 3000 computer system. It also describes the standard procedures for powering up and powering

down the system, basic troubleshooting information, and it includes important safety and regulatory specifications.

- *SGI Altix 3700 Bx2 User's Guide* provides an overview of the architecture and descriptions of the major components that compose the SGI Altix 3700 Bx2 family of servers. It also provides the standard procedures for powering on and powering off the system, basic troubleshooting information, and important safety and regulatory specifications.

- *SGI L1 and L2 Controller Software User's Guide* describes how to use the L1 and L2 controller commands at your system console to monitor and manage your system.

- *Console Manager for SGIconsole Administrator's Guide* describes the Console Manager software graphical interface, which allows you to control multiple SGI servers; SGI partitioned systems; and large, single-system image servers.

- *Linux Application Tuning Guide* provides information about tuning application programs on the SGI Altix 3000 family of servers and superclusters and the SGI Altix 350 systems, running the Linux operating system.

- *Linux Resource Administration Guide* is a reference document for people who manage the operation of SGI computer systems running the Linux operating system. It contains information needed in the administration of various system resource management features such as Comprehensive System Accounting (CSA), Array Services, CPU memory sets (CpuMemSets) and scheduling, and the Cpuset System on SGI ProPack 3 systesm and the Cpuset Facility on SGI ProPack 4 systems

- *Performance Co-Pilot for IA-64 Linux User's and Administrator's Guide* documents the Performance Co-Pilot software package running on IA-64 Linux systems. Performance Co-Pilot provides a systems-level suite of tools that cooperate to deliver integrated performance monitoring and performance management services spanning the hardware platforms, operating systems, service layers, database management systems (DBMSs), and user applications.

- *Linux Device Driver Programmer's Guide-Porting to SGI Altix 3000 Systems* provides information on programming, integrating, and controlling drivers.

- *Message Passing Toolkit: MPI Programmer's Manual* describes industry-standard message passing protocol optimized for SGI computers.

- *XFS for Linux Administration* describes XFS, an open-source, fast recovery, journaling filesystem that provides direct I/O support, space preallocation, access control lists, quotas, and other commercial file system features.

- *Origin 2000 and Onyx2 Performance Tuning and Optimization Guide* contains information specific to MIPS/IRIX systems, but the general guidelines in the document are hardware and operating system independent.
- *Event Manager User Guide* provides information about the Event Manger application that collects event information from other applications. This document describes the Event Manager application, the application programming interface that you can use to access it, the procedures that you can use to communicate with it from another application, and the commands that you can use to control it.
- *Embedded Support Partner User Guide* provides information about using the Embedded Support Partner (ESP) software suite to monitor events, set up proactive notification, and generate reports. This revision of the document describes ESP version 3.0, which is the first version of ESP that supports the Linux operating system.

Additional Reading

The following sections describe publications that contain additional information that my be helpful in the administration of your system.

Linux System Administration

Linux system administration information is available on your system at the following location:

`/usr/src/linux/Documentation`

Linux system administration course information is available at:

http://www.sgi.com/support/custeducation/courses/linux/sys_admin.html

Intel Compiler Documentation

Documentation for the Intel compilers is located on your system in the `/docs` directory of the directory tree where your compilers are installed. If you have installed the Intel compilers, the following documentation is available:

- *Intel C++ Compiler User's Guide* (`c_ug_lnx.pdf`)
- *Intel Fortran Compiler User's Guide* (`for_ug_lnx.pdf`)

- *Intel Fortran Programmer's Reference* (`for_prg.pdf`)
- *Intel Fortran Libraries Reference* (`for_lib.pdf`)

Other Intel Documentation

The following documents describe the Itanium (previously called "IA-64") architecture and other topics of interest:

- *Intel Itanium 2 Processor Reference Manual for Software Development and Optimization*, available online at the following location:

http://developer.intel.com/design/itanium2/manuals/251110.htm

- *Intel Itanium Architecture Software Developer's Manual*, available online at the following location:

http://developer.intel.com/design/itanium/manuals/iiasdmanual.htm

- *Introduction to Itanium Architecture*, available online at the following location:

http://shale.intel.com/softwarecollege/CourseDetails.asp?courseID=13

(secure channel required)

Open-Source Documents

The following open-source document may be useful to you.

- *Debugging with DDD User's Guide and Reference Manual* provides information on using the DataDisplayDebugger (DDD). It is available at the following location:

  http://www.gnu.org/manual/ddd/pdf/ddd.pdf

Obtaining Publications

You can obtain SGI documentation in the following ways:

- See the SGI Technical Publications Library at: http://docs.sgi.com. Various formats are available. This library contains the most recent and most comprehensive set of online books, release notes, man pages, and other information.
- SGI ProPack for Linux documentation, and all other documentation included in the RPMs on the distribution CDs can be found on the CD titled "SGI ProPack 3 for Linux - Documentation CD." To access the information on the documentation CD, open the index.html file with a web browser. Because this online file can be updated later in the release cycle than this document, you should check it for the latest information. After installation, all SGI ProPack for Linux documentation (including `README.SGI`) is in `/usr/share/doc/sgi-propack-3.0`.
- You can view man pages by typing man *title* on a command line.

Conventions

The following conventions are used throughout this document:

Convention	**Meaning**
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. (Output is shown in nonbold, fixed-space font.)
[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this publication, contact SGI. Be sure to include the title and document number of the publication with your comments. (Online, the document number is located in the front matter of the publication. In printed publications, the document number is located at the bottom of each page.)

You can contact SGI in any of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

- Use the Feedback option on the Technical Publications Library Web page:

  http://docs.sgi.com

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.

- Send mail to the following address:

  Technical Publications
  SGI
  1500 Crittenden Lane, M/S 535
  Mountain View, California 94043–1351

SGI values your comments and will respond to them promptly.

Chapter 1

Configuring Your System

This chapter provides information on configuring your system and covers the following topics:

- "PCI or PCI-X Card Hot-Plug Software" on page 1
- "I /O Subsystems" on page 9
- "Cloning System Disks on SGI ProPack 3 Systems" on page 18
- "Setting up `quota` on the `root` File System on SGI ProPack 3 Systems" on page 21
- "System Partitioning " on page 22
- "Making Array Services Operational" on page 37
- "Pluggable Authentication Modules on ProPack 3 Systems" on page 38
- "Network File System Configuration on ProPack 3 Systems" on page 38

PCI or PCI-X Card Hot-Plug Software

The Linux PCI/X hot-plug feature supports inserting a PCI or PCI-X card into an empty slot and preparing that card for use or deactivating a PCI or PCI-X card and then removing it from its slot, while the system is running. Hot-plug operations can be initiated using either the `sgihpview` command or a series of shell commands.

This section describes hot-swap operations and covers the following topics:

- "Introduction PCI or PCI-X Card Hot-Plug Operations" on page 2
- "PCI Hot-Plug Virtual File System" on page 2
- "Controlling Hot-Plug Operations" on page 3
- "Using the `sgihpview` GUI To Control a Hot-Plug Operation" on page 5
- "Using Shell Commands To Control a Hot-Plug Operation" on page 8

Introduction PCI or PCI-X Card Hot-Plug Operations

A hot-swap operation is the combination of a remove and insert operation targeting the same slot. Single function cards, multi-function cards, and PCI/X-to-PCI/X bridges are supported.

A hot-plug insert operation consists of attaching a card to an SGI card carrier, inserting the carrier in an empty slot, and using software commands to initiate the software controlled power-up and initialization of the card.

A hot-plug remove operation consists of manually terminating any users of the card, and then using software commands to initiate the remove operation to deactivate and power-down the card.

The Altix system L1 hardware controller has these hot-plug restrictions, as follows:

- A 33 MHz PCI/X card cannot be inserted into an empty PCI bus
- The last card cannot be removed from a bus running at 33 MHz

If these restrictions are detected by the Linux kernel and reported to the user, the requested hot-plug operation fails.

For detailed instructions on how to install or remove a PCI or PCI-X card on the SGI Altix 350 system, see “PCI and PCI-X Cards” in Chapter 6, “Installing and Removing Customer-replaceable Units” in *SGI Altix 350 System User’s Guide*.

For detailed instructions on how to install or remove a PCI or PCI-X card on the SGI Altix 3000 series systems, see “Adding or Replacing a PCI or PCI-X Card” in Chapter 12, “Maintenance and Upgrade Procedures” in *SGI Altix 3000 User’s Guide*.

For more information on the SGI L1 and L2 controller software, see the *SGI L1 and L2 Controller Software User’s Guide*.

PCI Hot-Plug Virtual File System

The Linux PCI hot-plug infrastructure is based on a Linux virtual file system called `pcihpfs`. Each PCI/X slot capable of a hot-plug operation has a directory in this file system with a name based on the hardware location of the slot. Under each directory, is a virtual file called `power` that initiates a hot-plug operation and queries the hot-plug status of the slot.

The PCI hot-plug file system is automatically mounted at system boot at the predefined mount point /proc/bus/pci/slots. If needed, the file system can be manually mounted and unmounted using these commands, as follows:

- To mount the file system, use this command, as follows:

  ```
  % mount -t pcihpfs none /proc/bus/pci/slots
  ```

- To unmount the file system, use this command, as follows:

  ```
  % umount /proc/bus/pci/slots
  ```

An example of a full path name of a power file is, as follows:

```
/proc/bus/pci/slot/module_001i03_bus_2_slot_1/power
```

Writing the character 1 to the power file initiates a hot-plug insert operation. Writing the character 0 to the power file initiates a hot-plug remove operation. Reading the power file returns the character 0 or 1 indicating that the slot is powered-up and operational or powered-down.

Controlling Hot-Plug Operations

This section describes hot-plug operations and the format of a slot name. It covers the following topics:

- "Slot Name Format" on page 3
- "sgihpview Command" on page 4
- "Hot-plug Insert Operation" on page 4
- "Hot-plug Remove Operation" on page 5

Slot Name Format

Hot-plug operations target a particular slot using the name of the slot. All slots that are eligible for a hot-plug operation have a directory in the hot-plug file system that is usually mounted at /proc/bus/pci/slots. The name of the target slot is based on the hardware location of the slot in the system. The format of the slot name is, as follows:

```
module_RRRTPP_bus_B_slot_S
```

where:

- RRR is the rack location of the brick containing the slot
- T is the I/O brick type and can be one of the following:
  - i is for a base I/O brick
  - p is for a general I/O brick
- PP is the position of the brick within the rack
- B is the bus number of the slot printed on the back of the brick
- S is the slot number printed on the back of the brick

For example, the name of the first slot in bus 2 of an IX-brick in position 3 of rack 1 is, as follows:

```
module_001i03_bus_2_slot_1
```

sgihpview Command

Note: The sgihpview command is **only** supported on SGI ProPack 3 systems.

Hot-plug operations can be initiated using either the sgihpview command or a series of shell commands. The sgihpview command is window-based and the shell commands are text-based. Both methods require the name of the target slot when initiating a hot-plug operation.

Before initiating a hot-plug remove operation, the system administrator must manually terminate any processes using the target card. Failure to properly terminate any outstanding accesses to the target card may result in a system failure or data corruption when the hot-plug operation is initiated.

Hot-plug Insert Operation

A hot-plug insert operation first instructs the L1 hardware controller to power-up the slot and reset the card. The L1 controller then checks that the card to be inserted is compatible with the running bus. Compatible is defined, as follows:

- The card must support the same mode as the running bus, for example, PCI or PCI-X

- The card must be able to run at the current bus speed
- That a 33 MHz card is **not** being inserted into an empty bus

Any L1 controller detected incompatibilities or errors are reported to the user and the insert operation fails.

Once the slot has been successfully powered-up by the L1 controller, the Linux hot-plug infrastructure notifies the driver of the card that the card is available and needs to be initialized. After the driver has initialized the card, the hot-plug insert operation is complete and card is ready for use.

Hot-plug Remove Operation

Before initiating a hot-plug remove operation, the system administrator must manually terminate any processes using the target card.

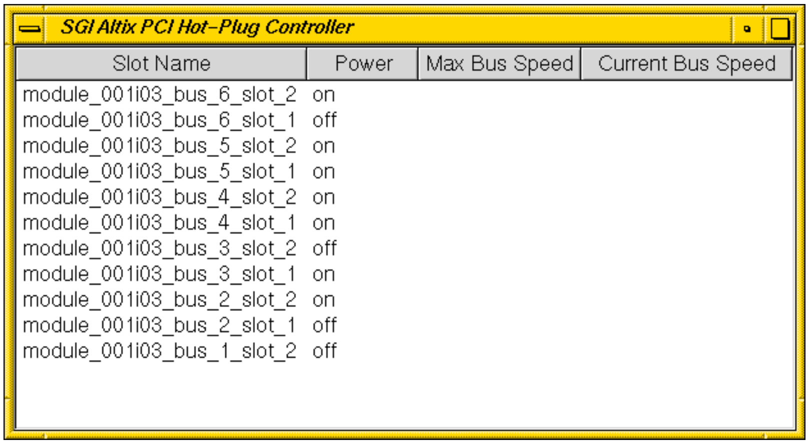
Warning: Failure to properly terminate any outstanding accesses to the target card may result in a system failure or data corruption when the hot-plug operation is initiated.

For a hot-plug remove operation, the hot-plug infrastructure verifies that the target slot is eligible to be powered-down. The L1 hardware controller restrictions do **not** permit the last card to be removed from a bus running at 33 MHz and an attempt to remove the last card fails. The hot-plug infrastructure then notifies the driver of the card of a pending hot-plug remove operation and the driver deactivates the card. The L1 hardware controller is then instructed to power-down the slot.

Attempts to power-down a slot that is already powered-down, or power-up a slot that is already powered-up are ignored.

Using the `sgihpview` GUI To Control a Hot-Plug Operation

Hot-plug operations can be initiated and the hot-plug status of slots queried using the `sgihpview` window-based application shown in Figure 1-1 on page 6.

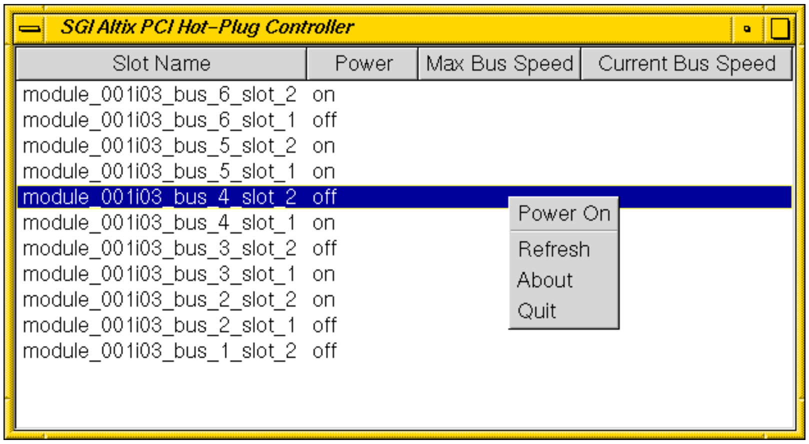


Slot Name	Power	Max Bus Speed	Current Bus Speed
module_001i03_bus_6_slot_2	on		
module_001i03_bus_6_slot_1	off		
module_001i03_bus_5_slot_2	on		
module_001i03_bus_5_slot_1	on		
module_001i03_bus_4_slot_2	on		
module_001i03_bus_4_slot_1	on		
module_001i03_bus_3_slot_2	off		
module_001i03_bus_3_slot_1	on		
module_001i03_bus_2_slot_2	on		
module_001i03_bus_2_slot_1	off		
module_001i03_bus_1_slot_2	off		

Figure 1-1 `sgihpview` Graphical User Interface

The result of a hot-plug operation is displayed in a pop-up window allowing seamless control of hot-plug operations. Every slot in the system that is hot-plug capable is displayed as a row in the main window. Each column presents the name of the slot, power status, maximum bus speed, and current bus speed. The bus speeds are not yet implemented and are blank.

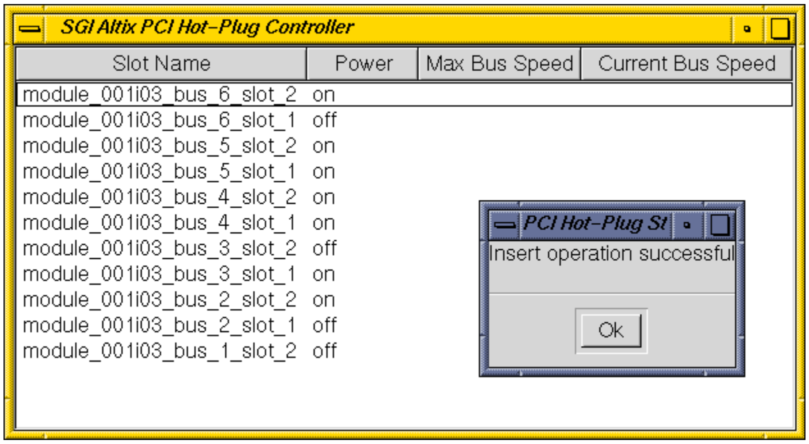
To initiate a hot-plug insert operation, select the target slot by left-clicking its **slot name** in the main window. Then right-click to bring up the action window and select **Power On** as shown in Figure 1-2 on page 7.



Slot Name	Power	Max Bus Speed	Current Bus Speed
module_001i03_bus_6_slot_2	on		
module_001i03_bus_6_slot_1	off		
module_001i03_bus_5_slot_2	on		
module_001i03_bus_5_slot_1	on		
module_001i03_bus_4_slot_2	off		
module_001i03_bus_4_slot_1	on		
module_001i03_bus_3_slot_2	off		
module_001i03_bus_3_slot_1	on		
module_001i03_bus_2_slot_2	on		
module_001i03_bus_2_slot_1	off		
module_001i03_bus_1_slot_2	off		

Figure 1-2 Hot-plug **Power On** Operation

The result of the insert is displayed in a pop-up window and the power status of the slot is updated as shown in Figure 1-3 on page 7.



Slot Name	Power	Max Bus Speed	Current Bus Speed
module_001i03_bus_6_slot_2	on		
module_001i03_bus_6_slot_1	off		
module_001i03_bus_5_slot_2	on		
module_001i03_bus_5_slot_1	on		
module_001i03_bus_4_slot_2	on		
module_001i03_bus_4_slot_1	on		
module_001i03_bus_3_slot_2	off		
module_001i03_bus_3_slot_1	on		
module_001i03_bus_2_slot_2	on		
module_001i03_bus_2_slot_1	off		
module_001i03_bus_1_slot_2	off		

Insert operation successful

Figure 1-3 **PCI Hot-Plug Status** Pop-up Window

To initiate a hot-plug remove operation, select the target slot by left-clicking its **slot name** in the main window. Then right-click to bring up the action window and select

Power Off. The result of the remove operation is displayed in a pop-up window and the power status of the slot is updated.

The current power status of all slots is displayed in the main window and is updated after a hot-plug operation. To manually refresh the power status of a slot, select the slot by left-clicking its **slot name** in the main window. Then right-click to bring up the action menu and select **Refresh**.

Selecting **About** from the action menu displays the `sgihpview` version information and **Quit** terminates `sgihpview`.

Using Shell Commands To Control a Hot-Plug Operation

A hot-plug operation can be initiated by writing to the target `power` file of the slot. After composing the name of the slot based on its location, change into the directory of the slot in the hot-plug virtual file system.

For example, to target hot-plug operations to slot 1 in bus 2 of the IX-brick in position 3 of rack 1, change directory (`cd`), as follows:

```
cd /proc/bus/pci/slots/module_001i03_bus_2_slot_1/
```

To query the current hot-plug status of the slot, read the `power` file of the slot, as follows:

```
% cat power
```

A value of `0` indicates that the slot is powered-down, and a value of `1` indicates that the slot is powered-up.

To initiate an insert operation to a slot that is powered-down, write the character `1` to the `power` file of the slot, as follows:

```
% echo 1 > power
```

Detailed status messages for the insert operation are written to the `syslog` by the hot-plug infrastructure. These messages can be displayed using the Linux `dmesg` user command, as follows:

```
% dmesg
```

A hot-plug remove operation is initiated to a slot that is powered-up by writing the character 0 to the power file of the slot, as follows:

```
% echo 0 > power
```

Detailed status messages for the remove operation are written to the syslog by the hot-plug infrastructure. These messages can be displayed using the Linux dmesg user command, as follows:

```
% dmesg
```

I /O Subsystems

Although some HPC workloads might be mostly CPU bound, others involve processing large amounts of data and require an I/O subsystem capable of moving data between memory and storage quickly, as well as having the ability to manage large storage farms effectively. The SCSI subsystem, XFS filesystem, XVM volume manager, and data migration facilities were leveraged from the IRIX operating system and ported to provide a robust, high-performance, and stable storage I/O subsystem on Linux.

The following sections describe persistent PCI-X bus numbering, persistent naming of Ethernet devices, the XSCSI subsystem, the XSCSI-SCSI subsystem, the XFS filesystem, and the XVM Volume Manager.

This section covers the following topics:

- "Persistent PCI-X Bus Numbering on ProPack 3 Systems" on page 10
- "Persistent Naming of Ethernet Devices on ProPack 3 Systems" on page 11
- "Persistent PCI Bus Numbering on ProPack 4 Systems" on page 13
- "Persistent IP Addressing of Ethernet Interfaces on ProPack 4 Systems" on page 15
- "XSCSI Subsystem on ProPack 3 Systems" on page 15
- "XSCSI-SCSI Subsystem for ProPack 3 Systems" on page 16
- "Tape Stream Driver for ProPack 3 Systems" on page 16
- "Using Standard Linux SCSI Drivers on XSCSI Devices" on page 17

Persistent PCI-X Bus Numbering on ProPack 3 Systems

Note: This section only applies to systems running SGI ProPack 3 for Linux. For SGI ProPack 4 for Linux systems, see "Persistent PCI Bus Numbering on ProPack 4 Systems" on page 13.

Persistent PCI-X bus numbering ensures that bus numbers can remain the same across reboots in case of faulty hardware or reconfiguration. During platform initialization, as buses are discovered, they are assigned a logical bus number. Each logical bus number is unique, systemwide. The default number of buses supported by SGI Altix 3000 systems is 256 (numbered 0 to 255).

By default, bus numbers are allocated starting from the lowest C-brick module ID to which the I/O brick is connected. An I/O brick is either an IX-brick or a PX-brick on SGI Altix 3000 systems. Each I/O brick is allocated 0x10 buses, although the current I/O bricks support only six buses each. Therefore, bus numbers are not contiguous across the system. Bus numbers are sparse and have holes in them.

An I/O brick has six physical buses. These buses are numbered 0x1 through 0x6, left to right, looking at the back of the I/O brick. If there is more than one I/O brick on the system, the buses on the next I/O brick are numbered 0x11 through 0x16. The rationale for this numbering is that the bus numbers of each I/O brick are stamped on the back of the brick and they start from 1. Therefore, the rightmost digit of a bus number corresponds to the actual stamped number on the I/O brick.

f you have only one I/O brick, you do not need persistent bus numbering. However, if you have more than one I/O brick, persistent bus numbering is strongly recommended, so that if an I/O brick fails to boot, your bus numbers are still the same.

To make use of persistent bus numbering, you must supply the `ioconfig=` parameter to the kernel.

The `ioconfig=` parameter takes a comma-separated list of I/O brick numbers as an argument. The following manual boot example uses `elilo`. It tells the kernel that the I/O brick represented by 101.01 is assigned bus numbers 0x1 through 0x6 and the I/O brick represented by `101.02` is assigned bus numbers 0x11 through 0x16, as follows:

```
Shell> elilo vmlinux ioconfig="101.01,101.02" root=/dev/xscsi/pci01.03.0-1/target1/lun0/part3
```

You can find the numbers to use with `ioconfig` by looking at the components of your system from your L2 controller, as in the following example:

```
l2-pumpkin-001-L2>pwr
  001c11:
  power appears on
  001c27:
  power appears on
  101i01:
  power appears on
  101p02:
  power appears on
```

In this example, `101i01` is an IX-brick and represents the `101.01` in the previous `elilo` boot example. The `101p02` notation is a PX-brick and represents `101.02`.

Most customers will have their system set up to boot automatically. This means that you should update your `elilo.conf` file with the `ioconfig` parameter. The `.elilo.conf` file is available in the `/boot/efi/EFI/sgi` directory on an SGI ProPack 3 system. The following is an example `elilo.conf` file:

```
prompt
  timeout=50
  relocatable
  default=sgilinux
  append="ioconfig=101.01,101.02"

  image=vmlinuz-2.4.19-sgi21r4
          label=sgilinux
          read-only
          root=/dev/xscsi/pci01.03.0-1/target1/lun0/part3
```

In the previous example, pay special attention to the `append=` line. That is, notice where the `ioconfig` information goes when using `elilo.conf`.

Persistent Naming of Ethernet Devices on ProPack 3 Systems

Note: This section only applies to systems running SGI ProPack 3 for Linux. For SGI ProPack 4 for Linux systems, see "Persistent IP Addressing of Ethernet Interfaces on ProPack 4 Systems" on page 15.

Persistent naming of Ethernet devices is an SGI proprietary mechanism and is supported on SGI Altix 3000 systems. Persistent naming refers to the mechanism that ensures that the Gigabit Ethernet card on the IO9 interface of an SGI Altix 3000 system is set up to always be eth0. It guarantees that the base Ethernet device number is assigned to the correct media access control (MAC) address on an SGI Altix 3000 system even when multiple Ethernet devices are present in the system.

The /etc/sysconfig/networking/eth0_persist file contains the mapping of Ethernet device numbers to MAC addresses. If the file does not exist, it is created by the /etc/rc.d/init.d/eth_persist script, which is run at boot time. To ensure that eth0 is indeed assigned to the MAC address of the IO9 Ethernet card, it might be necessary to edit the file after the first time an SGI Altix 3000 system has been brought up after a clean install.

Besides ensuring that the mapping of Ethernet device numbers to MAC addresses persists as cards are added to a system, persistent naming also allows system administrators to control the way in which Ethernet devices are numbered. For example, if the Ethernet card with device number ethX is lost and the system administrator tries to recover by using the Ethernet card with device number ethY, it is possible to force the latter card to take on Ethernet device number ethX by editing the /etc/sysconfig/networking/eth0_persist file accordingly.

Following is a sample /etc/sysconfig/networking/eth0_persist file:

```
eth0 08:00:69:13:dc:ec
  eth1 08:00:69:13:72:e8
```

The content of this file results in the following configuration:

```
[root]# ifconfig -a
  eth0 Link encap:Ethernet  HWaddr 08:00:69:13:DC:EC
       inet addr:128.162.246.125 Bcast:128.162.246.255 Mask:255.255.255.0
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:843 errors:0 dropped:0 overruns:0 frame:0
       TX packets:1245 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:100
       RX bytes:386044 (376.9 Kb)  TX bytes:126741 (123.7 Kb)
       Interrupt:59

  eth1 Link encap:Ethernet  HWaddr 08:00:69:13:72:E8
       BROADCAST MULTICAST  MTU:1500  Metric:1
       RX packets:136 errors:0 dropped:0 overruns:0 frame:0
       TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:100
          RX bytes:8850 (8.6 Kb)  TX bytes:0 (0.0 b)
          Interrupt:63
```

If the slot directly to the right of the IO9 is populated by an Ethernet board, that is, bus 1, slot 2, and if you re-install Linux after it has been placed there or if you remove the `eth0_persist` persistent naming file, the IO9 could become `eth1` instead of `eth0`. This is not recommended because it could affect your product licenses. If the Ethernet board is in any other IX-brick slot, you will not encounter this problem.

If it is necessary to have an Ethernet board in IO9 bus 1 slot 2, install the OS, including SGI ProPack, with the Ethernet board removed. After SGI ProPack is installed, you can replace the Ethernet board in bus 1 slot 2.

Persistent PCI Bus Numbering on ProPack 4 Systems

Note: This section only applies to systems running SGI ProPack 4 for Linux. For SGI ProPack 3 for Linux systems, see "Persistent PCI-X Bus Numbering on ProPack 3 Systems" on page 10.

Persistent PCI bus numbering ensures that bus numbers can remain the same across reboots in case of faulty hardware or reconfiguration. During platform initialization, as buses are discovered, they are assigned a logical bus number. Each logical bus number is unique, systemwide. The default number of buses supported by SGI Altix 3000 systems is 256 (numbered 0 to 255).

By default and for IX-bricks and PX-bricks only, bus numbers are allocated starting from the lowest C-brick module ID to which the I/O brick is connected. For the NUMAlink connected PA-bricks it is the lowest module ID of the PA-brick. Since you can have both IX-bricks and PX-bricks on a system along with PA-bricks, it is the lowest of all these combinations.

An I/O brick can be any one of the following:

- IX-brick

  The IX-brick is PX-brick with a BaseIO card (IO9 or IO10) in bus1 slot1 of the brick.

- PX-brick

  The PX-brick is a XIO connected PCI/PCI-X I/O brick. There are 6 pci/pci-x buses, each with 2 pci/pcix slots.

- PA-brick

  The PA-brick, also called the PA expansion module, is a NUMAlink connected PCI/PCI-X I/O Brick. There are 4 pci/pci-x buses, two with 2 slots, and 2 with a single slot. Note that the PA-brick is supported only in NUMAlink-4 Altix configurations

Each I/O brick is allocated 0x10 buses, although the current I/O bricks support only six buses each. Therefore, bus numbers are not contiguous across the system. Bus numbers are sparse and have holes in them.

On an SGI Altix 3000 series systems, the IX-bricks and PX-brick I/O bricks have six physical buses and s PA-brick I/O brick has four physical buses. These buses are numbered 0x1 through 0x6, left to right, looking at the back of the I/O brick. If there is more than one I/O brick on the system, the buses on the next I/O brick are numbered 0x11 through 0x16. The rationale for this numbering is that the bus numbers of each I/O brick are stamped on the back of the brick and they start from 1. Therefore, the rightmost digit of a bus number corresponds to the actual stamped number on the I/O brick An SGI Altix 350 system has 2 buses per module.

If you have only one I/O brick, you do not need persistent bus numbering. However, if you have more than one I/O brick, persistent bus numbering is strongly recommended, so that if an I/O brick fails to boot, your bus numbers are still the same.

To make use of persistent bus numbering, you must set the `PciBusList1`, `PciBusList2`, and `PciBusList3` PROM environment variables. Each variable is a comma separated list of the moduleid of I/O Bricks representing the order you want the persistent bus numbering.

There are three `PciBusList#` variables to support persistent bus numbering for 16 I/O bricks. (This is because PROM variables are limited to 48 characters . That gives us six `moduleid` per variable to support up to 16 I/O bricks). Some examples are, as follows:

```
POD> setallenv PciBusList1 "001i01,001p04,101p10"
```

or

```
POD> setallenv PciBusList1 "001i01,001p04,101p10,010i10,011p12,012p11"
POD> setallenv PciBusList2 "102i01"
```

The PROM `clearallenv PciBusList1` variable is used to clear the persistent bus numbering.

Persistent IP Addressing of Ethernet Interfaces on ProPack 4 Systems

Note: This section only applies to systems running SGI ProPack 4 for Linux. For SGI ProPack 3 for Linux systems, see "Persistent Naming of Ethernet Devices on ProPack 3 Systems" on page 11.

An Ethernet interface can be given a persistent internet addresses by associating its permanent MAC address, such as 08:00:69:13:f1:aa, with an internet protocol (IP) address, for example 192.168.20.1. An interface with a persistent IP address will be given the same IP address each time the system is booted.

ProPack 4 built on top of SLES9. SLES9 has its own implementation of Ethernet persistent naming to allow Ethernet interfaces to use persistent IP addresses, that is, the Media Access Control (MAC) address is embedded in the name of the configuration file of the associated Ethernet interface.

For example, if you want to ensure that the Ethernet interface whose MAC address is `08:00:69:13:f1:aa` always uses the IP address `192.168.20.1`, edit the `/etc/sysconfig/network/ifcfg-eth-id-08:00:69:13:f1:aa` file so that it contains the following:

```
BOOTPROTO="static"
STARTMODE="onboot"
IPADDR="192.168.20.1"
NETMASK="255.255.255.0"
NETWORK="192.168.20.0"
BROADCAST="192.168.20.255"
```

For more information on Ethernet device configuration files and instructions for editing them, see `/usr/share/doc/packages/sysconfig/README`.

XSCSI Subsystem on ProPack 3 Systems

Note: This section only applies to systems running SGI ProPack 3 for Linux. The XSCSI subsystem is not supported on SGI ProPack 4 for Linux systems.

The SGI XSCSI subsystem on Linux leverages from IRIX functionality to provide more robust error handling, failover, and storage area network (SAN) infrastructure support as well as long-term large system performance tuning. XSCSI takes advantage of specific features of SGI architecture that standard open source drivers cannot without rewriting.

The naming convention of XSCSI device names is shown in the following example:

```
/dev/xscsi/pci01.03.0-1/target1/lun0/part1
```

Components of the XSCSI device name are as follows:

`pci01`	System bus number 0x1
`03`	Device number 3 on that bus
`0-1`	Logical unit 0 port 1

Notice that the device number (slot number), logical unit, and port number are fixed. These will never change. However, the system bus number could change because of a hardware problem (such as the I/O brick not booting) or a reconfiguration.

Persistent PCI-X bus numbering (see "Persistent PCI-X Bus Numbering on ProPack 3 Systems" on page 10), if enabled, provides persistent naming to prevent bus number changes even when the hardware fails or is reconfigured. If you use XSCSI names for mounting or locating devices and you also use persistent bus numbering, your XSCSI device names will always be persistent across reboots.

XSCSI-SCSI Subsystem for ProPack 3 Systems

Note: This section only applies to systems running SGI ProPack 3 for Linux. The XSCSI subsystem is not supported on SGI ProPack 4 for Linux systems.

The XSCSI-SCSI subsystem provides SCSI device emulation of XSCSI devices. It allows programs written for the SCSI disk driver, the SCSI tape driver, and the SCSI generic driver to use devices controlled by the XSCSI drivers.

Tape Stream Driver for ProPack 3 Systems

Note: This section only applies to systems running SGI ProPack 3 for Linux. The SGI tape stream driver is not supported on SGI ProPack 4 for Linux systems.

The SGI tape stream (TS) tape driver is low-level software that supports all of tape backup applications on SGI Altix systems. These applications include the Data Migration Facility (DMF), the Tape Management Facility (TMF), OpenVault, and `xfsdump`(1M)/`xfsrestore`(1M), the XFS filesystem incremental dump/restore utilities. The TS tape driver converts application-level commands into a form of communication that the tape drives can understand and vise versa. The TS tape driver is made up of three separate components, none of which are functional without the other two parts. All three packages must be installed on the system for the TM driver to function, as follows:

`ts-driver`

Distributed with SGI ProPack.

The `ts-driver` package contains the loadable TS tape driver (`ts.o`). The driver is kept separate from the TS daemon and commands because the driver is kernel/system specific while the daemon and commands are not. The TS tape driver is used with the SGI XSCSI interface (`xscsi.o`).

`ts-cmd`

Distributed with Asynchronous Personality Daemon (APD) Product CD

The `ts-cmd` package contains the TS daemon and administrator commands as well as the TS man pages.

`apd`

Distributed with Asynchronous Personality Daemon (APD) Product CD

The `apd` package contains a TS-compatible "personality daemon" for each type of tape device supported by SGI. APD is a licensed product and may be obtained upon request from SGI.

Using Standard Linux SCSI Drivers on XSCSI Devices

Note: This section only applies to systems running SGI ProPack 3 for Linux. The XSCSI subsystem is not supported on SGI ProPack 4 for Linux systems.

You can use standard Linux SCSI device drivers on XSCSI devices.

To load drivers for the U320 card and enable SCSI disks on the LSI adapter automatically, add the following lines to /etc/rc.modules:

```
modprobe sd_mod     # SCSI midlayer and SCSI disk driver
modprobe sg         # SCSI-generic driver (optional)
modprobe mptbase    # U320 host driver
modprobe mptscsih   # Fusion MPT SCSI module
```

Cloning System Disks on SGI ProPack 3 Systems

Note: This section only applies to SGI systems running ProPack 3 for Linux.

Hard disks can be divided into one or more logical disks called *partitions*. This division is described in the partition table found in sector 0 of the disk.

You may want to make a second copy or *clone* of the system disks for backup purposes in the event there is a catastrophic failure on the first disk. This makes recovery faster and easier than re-installing the software from CDs.

This sections describes how to clone a disk on an SGI Altix system.

The following assumptions have been made:

- This procedure assumes the target disk is in IX—brick system bay 2 and the source root drive is in IX—brick system bay 1.

  It is not possible to boot from fibre channel disks because there is no PROM support. The boot loader initialized RAM disk (initrd) currently does not support cloning system disks.

- Booting from a TP900 storage system works but the fstab and elilo.conf files need to reflect a different XSCSI path.

  Note: The first drive bay on an Altix system is on the right side when you are facing the drives. The second drive (target 2) is on the left.

Procedure 1-1 Cloning System Disks

To clone a disk on an SGI Altix system, perform the following steps:

1. Halt the system and put the target disk in to drive bay 2 in the IX—brick

2. From the extensible firmware interface (EFI) shell prompt, boot the system into single-user mode, as follows:

   ```
   elilo sgilinux single
   ```

   Use the parted(8) disk partitioning and partition resizing program.

3. From the single-user mode prompt, replicate to the system disk to the disk in the second bay of the IX—brick disk, as follows:

   a. Use the parted(8) disk partitioning and partition resizing program to partition the disk, where the installer makes an EFI GUID Partition (GPT) partition table.

   b. Use the fdisk(8) partition table manipulator for Linux programs to partition the disk and create a master boot record (MBR) partition table.

   ⚠ **Caution:** If you boot the system and EFI cannot find any vfat filesystem on your disk (see fs(1) for filesystem type information), you may have run in to a situation where you tried to make an fdisk/MBR partition table but the GPT label still exists. You can resolve this by running parted on the target disk. If this inconsistency is found when using the parted program, you can fix it when you print the partition table.

   c. If you keep the partition number order the same as the source disk, you do not need to adjust elilo.conf and the fstab files.

4. Partition the disk using the fdisk or parted program to partition the disk. For information on these programs, see the fdisk(8) or parted(8) man page, respectively.

   Make sure the partition disk on the target looks like the partition table on the source. You use the disk- style path with parted or fdisk. For example to start the fdisk program perform the following command:

   ```
   fdisk /dev/xscsi/pci01.03.0-1/target2/lun0/disc
   ```

5. Put the filesystems and swap space on the partitions. Partition 1 is EFI, partition 2 is swap, and partition 3 is root.

6. Make the vfat filesystem for EFI, as follows:

   ```
   mkfs.msdos /dev/xscsi/pci01.03.0-1/target2/lun0/part1
   ```

7. Make the swap partition, as follows:

   ```
   mkswap /dev/xscsi/pci01.03.0-1/target2/lun0/part2
   ```

8. Make the root filesystem, as follows:

   ```
   mkfs.xfs /dev/xscsi/pci01.03.0-1/target2/lun0/part3
   ```

9. Mount the newly created EFI filesystem for replication, as follows:

   ```
   mount /dev/xscsi/pci01.03.0-1/target2/lun0/part1 /mnt
   ```

10. Replicate the EFI filesystem to the target disk, for example:

    ```
    cd /boot/efi; tar cBf - . | (cd /mnt && tar xBf -)
    ```

11. Unmount the target EFI filesystem, as follows:

    ```
    umount /mnt
    ```

12. Mount the target root for replication, as follows:

    ```
    mount /dev/xscsi/pci01.03.0-1/target2/lun0/part3 /mnt
    ```

13. Replicate from the source root drive to the target using the `xfsdump`(8) command piped to `xfsrestore`(8) command, as follows:

    ```
    xfsdump - / | xfsrestore - /mnt
    ```

14. Before you unmount the target root, remove the Ethernet persistent naming file. Failure to do this will result in the target system having no eth0 (eth1 would be appear instead), as follows:

    ```
    rm /mnt/etc/sysconfig/networking/eth0_persist
    ```

15. Unmount the target root filesystem, as follows:

    ```
    umount /mnt
    ```

16. If the source `fstab` uses filesystem labels, you either need to change the `/etc/fstab` of the target disk to **not** use them or you need to label the filesystem. Currently, SGI Altix systems do **not** use filesystem labels by default.

17. You have now replicated the root filesystem. It is a copy of the source. Here are some things to consider when you put the target drive in to its final destination system:

    - The target system may need a PROM upgrade; you may need to flash PROM from EFI.
    - The installer program runs a command to populate the EFI boot menu with a "SGI ProPack" style banner. You have three choices here, as follows:
        - Leave it as is. PROM has a generic non-labeled entry to boot automatically.
        - From Linux update the boot menu used in EFI, as follows:

```
/usr/sbin/efibootmgr -c -w -L "SGI ProPack(TM)" -d /dev/xscsi/pci01.03.0-1/target1/lun0/disc -p 1
```

        - Use the EFI boot manager menus to add boot options.
    - If you intend to use this disk in a different machine (as opposed to just having a backup of the running root available), you will need to configure network settings for the target machine. Configuration files like `/etc/hosts`, `/etc/sysconfig/network`, and `/etc/sysconfig/network-scripts/ifcfg-eth0` need to be updated. You can also run the `netconfig`(8) command.
    - If the target disk is for a different machine (not a backup), it is a good idea to remove the ssh keys from the `/etc/ssh` file to allow the new target to make new keys at the next start up. The filenames of the ssh keys are as follows: `/etc/ssh`: `ssh_host_key.pub`, `ssh_host_key` `ssh_host_rsa_key.pub`, `ssh_host_rsa_key` `ssh_host_dsa_key.pub`, `ssh_host_dsa_key`

Setting up `quota` on the `root` File System on SGI ProPack 3 Systems

Note: This section only applies to SGI systems running ProPack 3 for Linux.

You can use the `quota` command to display the disk usage and limits of a user. By default only the user quotas are printed.

The quota command reports the quotas of all the filesystems listed in the `/etc/mtab` file. For filesystems that are NFS-mounted, a call to the `rpc.rquotad` daemon on the server machine is performed to get the information. For more information on the quota command, see the quota(1) man page.

Procedure 1-2 Setting up quota on the root File System

To set up quota on root filesystem, add the following entry (rootflags=quota) to the append line in the elilo.conf file, as follows:

```
append="root=/dev/xscsi/pci00.01.0-1/target0/lun0/part3 rootflags=quota"
```

You also need to add the quota flag to the fstab entry for the filesystem in /etc/fstab. For more information, see the fstab(5) man page.

The repquota(8) command reports information similar to the following:

```
root@altix root]# repquota /
 *** Report for user quotas on device /dev/xscsi/pci00.01.0-1/target0/lun0/part3
 Block grace time: 7days; Inode grace time: 7days
 Block limits File limits
 User used soft hard grace used soft hard grace
 ----------------------------------------------------------------------
 root -- 2867796 0 0 104034 0 0
 ...
```

The repquota(8) command prints a summary of the disk usage and quotas for the specified file systems. For each user, the current number of files and amount of space (in kilobytes) is printed, along with any quotas created with edquota(8). For more information, see the quota, repquota(8), and edquota(8) man pages.

System Partitioning

This section describes how to partition an SGI ProPack server and contains the following topics:

1. "Overview" on page 23
2. "Advantages of Partitioning" on page 23
3. "Limitations of Partitioning" on page 25
4. "Supported Configurations" on page 25
5. "Installing Partitioning Software and Configuring Partitions" on page 25

6. "Connecting the System Console to the Controller" on page 37

Overview

A single SGI ProPack for Linux server can be divided into multiple distinct systems, each with its own console, root filesystem, and IP network address. Each of these software-defined group of processors are distinct systems referred to as a *partition*. Each partition can be rebooted, loaded with software, powered down, and upgraded independently. The partitions communicate with each other over an SGI NUMAlink connection. Collectively, all of these partitions compose a single, shared-memory cluster.

Direct memory access between partitions, sometimes referred to as global shared memory, is made available by the XPC and XPMEM kernel modules. This allows processes in one partition to access physical memory located on another partition. The benefits of global shared memory are currently available via SGI's Message Passing Toolkit (MPT) software.

It is relatively easy to configure a large SGI Altix system into partitions and reconfigure the machine for specific needs. No cable changes are needed to partition or repartition an SGI Altix machine. Partitioning is accomplished by commands sent to the system controller. For details on system controller commands, see the *SGI L1 and L2 Controller Software User's Guide*.

Advantages of Partitioning

This section describes the advantages of partitioning an SGI ProPack server as follows:

- "Create a Large, Shared-memory Cluster" on page 23
- "Provides Fault Containment" on page 24
- "Allows Variable Partition Sizes" on page 24
- "Provide High Performance Clusters" on page 24

Create a Large, Shared-memory Cluster

You can use SGI's NUMAlink technology and the XPC and XPMEM kernel modules to create a very low latency, very large, shared-memory cluster (currently up to 512 CPUs) for optimized use of Message Passing Interface (MPI) software and logically shared, distributed memory access (SHMEM) routines. The globally addressable, cache coherent, shared memory is exploited by MPI and SHMEM to deliver high performance.

Provides Fault Containment

Another reason for partitioning a system is fault containment. In most cases, a single partition can be brought down (because of a hardware or software failure, or as part of a controlled shutdown) without affecting the rest of the system. Hardware memory protections prevent any unintentional accesses to physical memory on a different partition from reaching and corrupting that physical memory. For current fault containment caveats, see "Limitations of Partitioning" on page 25.

You can power off and “warm swap” a failing C-brick in a down partition while other partitions are powered up and booted. For information see “Adding or Replacing a PCI or PCI-X Card” in chapter 12, “Maintenance and Upgrade Procedures” in SGI Altix 3000 User’s Guide or see “PCI and PCI-X Cards” in Chapter 6, “Installing and Removing Customer-replaceable Units” SGI Altix 350 User’s Guide.

Allows Variable Partition Sizes

Partitions can be of different sizes, and a particular system can be configured in more than one way. For example, a 128-processor system could be configured into four partitions of 32 processors each or configured into two partitions of 64 processors each. (See "Supported Configurations" for a list of supported configurations for system partitioning.)

Your choice of partition size and number of partitions affects both fault containment and scalability. For example, you may want to dedicate all 64 processors of a system to a single large application during the night, but then partition the system in two 32 processor systems for separate and isolated use during the day.

Provide High Performance Clusters

One of the fundamental factors that determines the performance of a high-end computer is the bandwidth and latency of the memory. The SGI NUMAflex technology gives an SGI ProPack partitioned, shared-memory cluster a huge performance advantage over a cluster of commodity Linux machines (white boxes). If a cluster of N white boxes, each with M CPUs is connected via Ethernet or Myrinet or InfinaBand, an SGI ProPack system with N partitions of M CPUs provides superior performance because of the significantly lower latency of the NUMAlink interconnect, which is exploited by the XPNET kernel module.

Limitations of Partitioning

Partitioning can increase the reliability of a system because power failures and other hardware errors can be contained within a particular partition. There are still cases where the whole shared memory cluster is affected; for example, during upgrades of harware which is shared by multiple partitions.

If a partition is sharing its memory with other partitions, the loss of that partition may take down all other partitions that were accessing its memory. This is currently possible when an MPI or SHMEM job is running across partitions using the XPMEM kernel module.

Supported Configurations

See the *SGI Altix 3000 User's Guide* or *SGI Altix 350 User's Guide* for information on configurations that are supported for system partitioning. Currently, the following guidelines are valid:

- Maximum number of partitions supported is 8
- Maximum partition size is 256 processors
- Maximum system size is 512 processors

For additional information about configurations that are supported for system partitioning, see your sales representative. SGI field support personnel may reference the *SGI Altix 3000 Internal Technical Configuration Manual*.

Installing Partitioning Software and Configuring Partitions

To enable or disable partitioning software, see "Partitioning Software" on page 26, to use the system partitioning capabilities, see "Partitioning Guidelines" on page 27 and "Partitioning a System" on page 28.

This section covers the following topics:

- "Partitioning Software" on page 26
- "Partitioning Guidelines" on page 27
- "Partitioning a System" on page 28
- "Determining If a System is Partitioned" on page 34

- "Accessing the Console on a Partitioned System" on page 35
- "Unpartitioning a System" on page 36

Partitioning Software

SGI ProPack for Linux servers have XP, XPC, XPNET, and XPMEM kernel modules installed by default to provide partitioning support. XPC and XPNET are configured off by default in the /etc/sysconfig/sgi-xpc and /etc/sysconfig/sgi-xpnet files, respectively. XPMEM is configured on by default in the /etc/sysconfig/sgi-xpmem file. To enable or disable any of these features, edit the appropriate /etc/sysconfig/ file and execute the /etc/init.d/sgi-xp script.

On SGI ProPack 4 systems, if you intend to use the cross-partition functionality of XPMEM, you will need to add xpc to the line in the /etc/sysconfig/kernel file that begins with MODULES_LOADED_ON_BOOT. Once that is added, you may either reboot the system or issue an modprobe xpc command to get the cross-partition functionality to start working. For more information on using modprobe, see the modprobe(8) man page.

The XP kernel module is a simple module which coordinates activities between XPC, XPMEM, and XPNET. All of the other cross-partition kernel modules require XP to function.

The XPC kernel module provides fault-tolerant, cross-partition communication channels over NUMAlink for use by the XPNET and XPMEM kernel modules.

The XPNET kernel module implements an Internet protocol (IP) interface on top of XPC to provide high-speed network access via NUMAlink. XPNET can be used by applications to communicate between partitions via NUMAlink, to mount file systems across partitions, and so on. The XPNET driver is configured using the ifconfig commands. For more information, see the ifconfig(1M) man page. The procedure for configuring the XPNET kernel module as a network driver is essentially the same as the procedure used to configure the Ethernet driver. You can configure the XPNET driver at boot time like the Ethernet interfaces by using the configuration files in /etc/sysconfig/network-scripts. To configure the XPNET driver as a network driver see the following procedure.

Procedure 1-3 Setting up Networking Between Partitions

The procedure for configuring the XPNET driver as a network driver is essentially the same as the procedure used to configure the Ethernet driver (eth0), as follows:

1. Log in as root.
2. On a SGI ProPack 3 system, configure the xp0 IP address, as follows:

   ```
   netconfig -d xp0
   ```

   For a SGI ProPack 4 system , configure the xp0 IP address using `yast2`. For information on using `yast2`, see *SUSE LINUX Enterprise Server 9 Installation and Administration* manual. The driver's full name inside `yast2` is `SGI Cross Partition Network adapter`.
3. Add the network address for the xp0 interface by editing the `/etc/hosts` file.
4. Reboot your system or restart networking.

The XPMEM kernel module provides direct access to memory located on other partitions. It uses XPC internally to communicate with XPMEM kernel modules on other partitions to accomplish this. XPMEM is currently used by SGI's Message Passing Toolkit (MPT) software (MPI and SHMEM).

Partitioning Guidelines

Follow these guidelines when partitioning your system:

- A partition must be made up of one or more C—bricks (Each C—brick contains four processors). The number of C-bricks in your systems determines the number of partitions you can create. The number of partitions cannot exceed the number of C—bricks your system contains. The first C—brick in each partition must have an IX—brick attached to it via the XIO connection of the C—brick.
- You need at least as many IX—bricks for base IO as partitions you wish to use.
- Each partition needs to have an IX—brick with a valid system disk in it. Since each partition is a separate running system, each system disk should be configured with a different IP address/system name, and so on.
- Each partition must have a unique partition ID number between 1 and 63, inclusively.
- All bricks in a partition must be physically contiguous. The route between any two processors in the same partition must be contained within that partition, and not through any other partition. If the bricks in a partition are not contiguous, the system will not boot.
- Each partition must contain the following components:

- At least one C—brick for system sizes of 64 C—bricks and below, or multiples of 4 C—bricks for system sizes of 65 C—bricks and above (minimum)
- One IX-brick (minimum)
- One root disk
- One console connection

Partitioning a System

This section describes how to partition your system.

Procedure 1-4 Partitioning a System Into Four Partitions

To partition your system, perform the following steps :

1. Make sure your system can be partitioned. See "Partitioning Guidelines" on page 27.
2. You can use the **Connect to System Controller** task of the SGIconsole Console Manager GUI to connect to the L2 controller of the system you want to partition. The L2 controller must appear as a node in the SGIconsole configuration. For information on how to use SGIconsole, see the *Console Manager for SGIconsole Administrator's Guide*.
3. Using the L2 terminal (l2term), connect to the L2 controller of the system you wish to partition. After a connection to the L2 controller, an L2> prompt appears, indicating that the L2 is ready to accept commands, for example:

   ```
   cranberry-192.168.11.92-L2>:
   ```

   If the L2 prompt does not appear, you can type `Ctrl-T`. To remain at the L2 command prompt, type **`l2`** (lowercase letter 'L') at the L2> prompt

Note: Each partition has its own set of the following PROM environment variables: `ConsolePath`, `OSLoadPartition`, `SystemPartition`, `netaddr`, and `root`.

For more information on using the L2 controller, see the *SGI L1 and L2 Controller Software User's Guide*.

You can partition a system from SGIconsole Console Manager system console connection, however, SGIconsole does not include any GUI awareness of partitions (in the node tree view for instance) or in the commands, and there is no way to power down a group of partitions, or get all the logs of a partitioned system, or to actually partition a system. If you partition a node that is managed by SGIconsole, make sure to edit the partition number of the node using the **Modify a Node** task. For more information, see the *Console Manager for SGIconsole Administrator's Guide*.

4. Use the L2 `sel` command to list the available consoles, as follows:

```
cranberry-192.168.11.92-L2>sel
```

Note: If the Linux operating system is currently executing, perform the proper shutdown procedures before you partition your system.

5. This step shows an example of how to partition a system into four separate partitions.

   a. To see the current configuration of your system, use the L2 `cfg` command to display the available bricks, as follows:

```
cranberry-192.168.11.92-L2>cfg
     L2 192.168.11.92: - --- (no rack ID set) (LOCAL)
     L1 192.168.11.92:8:0    - 001c31
     L1 192.168.11.92:8:1    - 001i34
     L1 192.168.11.92:11:0   - 001c31
     L1 192.168.11.92:11:1   - 001i34
     L1 192.168.11.92:6:0    - 001r29
     L1 192.168.11.92:9:0    - 001r27
     L1 192.168.11.92:7:0    - 001c24
     L1 192.168.11.92:7:1    - 101i25
     L1 192.168.11.92:10:0   - 001c24
```

```
L1 192.168.11.92:10:1   - 101i25
L1 192.168.11.92:0:0    - 001r22
L1 192.168.11.92:3:0    - 001r20
L1 192.168.11.92:2:0    - 001c14
L1 192.168.11.92:2:1    - 101i21
L1 192.168.11.92:5:0    - 001c14
L1 192.168.11.92:5:1    - 101i21
L1 192.168.11.92:1:0    - 001c11
L1 192.168.11.92:1:1    - 001i07
L1 192.168.11.92:4:0    - 001c11
L1 192.168.11.92:4:1    - 001i07
```

b. In this step, you need to decide which bricks to put into which partitions.

You can determine which C—bricks are directly attached to IX—bricks by looking at the output from the `cfg` man. Consult the hardware configuration guide for the partitioning layout for your particular system. In the `cfg` output above, you can check the number after the IP address. For example, `001c31` is attached to `001i34` which is indicated by the fact that they both have **`11`** after their respective IP address.

Note: On some systems, you will have a rack ID in place of the IP address. `001c31` is a C–brick (designated by the `c` in `001c31`) and `001i34` is an IX-brick (designated with an `i` in `001i34`).

Another pair is `101i25` and `001c24`. They both have **`10`** after the IP address. The brick names containing an `r` designation are routers. Routers do not need to be designated to a specific partition number.

In this example, the maximum number of partitions this system can have is four. There are only four IX—bricks total: `001i07`, `101i21`, `101i25`, and `001i34`.

Note: Some IX—brick names appear twice. This occurs because some IX—bricks have dual XIO connections.

You do not have to explicitly assign IX-bricks to a partition. The IX—bricks assigned to a partition are inherited from the C—bricks.

---

c. When you specify bricks to L2 commands, you use a *rack.slot* naming convention. To configure the system into four partitions, do not specify the whole brick name (`001c31`) but rather use the designation 1.31 as follows:

```
cranberry-192.168.11.92-L2>1.31 brick part 1
     001c31:
     brick partition set to 1.
     cranberry-192.168.11.92-L2>1.34 brick part 1
     001#34:
     brick partition set to 1.
     cranberry-192.168.11.92-L2>1.24 brick part 2
     001c24:
     brick partition set to 2.
     cranberry-192.168.11.92-L2>101.25 brick part 2
     101#25:
     brick partition set to 2.
     cranberry-192.168.11.92-L2>1.14 brick part 3
     001c14:
     brick partition set to 3.
     cranberry-192.168.11.92-L2>101.21 brick part 3
     101i21:
     brick partition set to 3.
     cranberry-192.168.11.92-L2>1.11 brick part 4
     001c11:
     brick partition set to 4.
     cranberry-192.168.11.92-L2>1.07 brick part 4
     001#07:
     brick partition set to 4.
```

d. To confirm your settings, enter the `cfg` command again, as follows:

Note: This may take up to 30 seconds.

```
cranberry-192.168.11.92-L2>cfg
     L2 192.168.11.92: - --- (no rack ID set) (LOCAL)
     L1 192.168.11.92:8:0    - 001c31.1
     L1 192.168.11.92:8:1    - 001i34.1
     L1 192.168.11.92:11:0   - 001c31.1
     L1 192.168.11.92:11:1   - 001i34.1
     L1 192.168.11.92:6:0    - 001r29
     L1 192.168.11.92:9:0    - 001r27
```

```
L1 192.168.11.92:7:0    - 001c24.2
L1 192.168.11.92:7:1    - 101i25.2
L1 192.168.11.92:10:0   - 001c24.2
L1 192.168.11.92:10:1   - 101i25.2
L1 192.168.11.92:0:0    - 001r22
L1 192.168.11.92:3:0    - 001r20
L1 192.168.11.92:2:0    - 001c14.3
L1 192.168.11.92:2:1    - 101i21.3
L1 192.168.11.92:5:0    - 001c14.3
L1 192.168.11.92:5:1    - 101i21.3
L1 192.168.11.92:1:0    - 001c11.4
L1 192.168.11.92:1:1    - 001i07.4
L1 192.168.11.92:4:0    - 001c11.4
L1 192.168.11.92:4:1    - 001i07.4
```

e. The system is now partitioned. However, you need to reset each partition to complete the configuration, as follows:

```
cranberry-192.168.11.92-L2>p 1,2,3,4 rst
```

Note: You can use a shortcut to reset every partition, as follows:

```
cranberry-192.168.11.92-L2>p * rst
```

f. To get to the individual console of a partition, such as partition 2, enter the following:

```
cranberry-192.168.11.92-L2>sel p 2
```

For more information on accessing the console of a partition, see "Accessing the Console on a Partitioned System" on page 35.

Procedure 1-5 Partitioning a System into Two Partitions

To partition your system, perform the following steps:

1. Perform steps 1 through 5 in Procedure 1-4, page 28.
2. To configure the system into two partitions, enter the following commands:

```
cranberry-192.168.11.92-L2>1.31 brick part 1
     001c31:
```

```
     brick partition set to 1.
     cranberry-192.168.11.92-L2>1.34 brick part 1
     001#34:
     brick partition set to 1.
     cranberry-192.168.11.92-L2>1.24 brick part 1
     001c24:
     brick partition set to 1.
     cranberry-192.168.11.92-L2>101.25 brick part 1
     101#25:
     brick partition set to 1.
     cranberry-192.168.11.92-L2>1.14 brick part 2
     001c14:
     brick partition set to 2.
     cranberry-192.168.11.92-L2>101.21 brick part 2
     101i21:
     brick partition set to 2.
     cranberry-192.168.11.92-L2>1.11 brick part 2
     001c11:
     brick partition set to 2.
     cranberry-192.168.11.92-L2>1.7 brick part 2
     001#07:
     brick partition set to 2.
```

3. To confirm your settings, issue the `cfg` command again, as follows:

   Note: This may take up to 30 seconds.

```
cranberry-192.168.11.92-L2>cfg
     L2 192.168.11.92: - --- (no rack ID set) (LOCAL)
     L1 192.168.11.92:8:0    - 001c31.1
     L1 192.168.11.92:8:1    - 001i34.1
     L1 192.168.11.92:11:0   - 001c31.1
     L1 192.168.11.92:11:1   - 001i34.1
     L1 192.168.11.92:6:0    - 001r29
     L1 192.168.11.92:9:0    - 001r27
     L1 192.168.11.92:7:0    - 001c24.1
     L1 192.168.11.92:7:1    - 101i25.1
     L1 192.168.11.92:10:0   - 001c24.1
     L1 192.168.11.92:10:1   - 101i25.1
     L1 192.168.11.92:0:0    - 001r22
```

```
    L1 192.168.11.92:3:0    - 001r20
    L1 192.168.11.92:2:0    - 001c14.2
    L1 192.168.11.92:2:1    - 101i21.2
    L1 192.168.11.92:5:0    - 001c14.2
    L1 192.168.11.92:5:1    - 101i21.2
    L1 192.168.11.92:1:0    - 001c11.2
    L1 192.168.11.92:1:1    - 001i07.2
    L1 192.168.11.92:4:0    - 001c11.2
    L1 192.168.11.92:4:1    - 001i07.2
```

4. Now the system has two partitions. To complete the configuration, reset the two partitions as follows:

```
cranberry-192.168.11.92-L2>p 1,2 rst
```

Determining If a System is Partitioned

Procedure 1-6 Determing If a System Is Partitioned

To determine whether a system is partitioned or not, perform the following steps:

1. Use the L2term to connect to the L2 controller of the system.

   Note: If you are connected to the L2 controller, but do not have the L2 prompt, try typing the following: CTRL-t.

2. Use the cfg command to determine if the system is partitioned, as follows:

```
cranberry-192.168.11.92-L2>cfg
L2 192.168.11.92: -(no rack ID set) (LOCAL)
L1 192.168.11.92:8:0    - 001c31.1
L1 192.168.11.92:8:1    - 001i34.1
L1 192.168.11.92:11:0   - 001c31.1
L1 192.168.11.92:11:1   - 001i34.1
L1 192.168.11.92:6:0    - 001r29
L1 192.168.11.92:9:0    - 001r27
L1 192.168.11.92:7:0    - 001c24.2
L1 192.168.11.92:7:1    - 101i25.2
L1 192.168.11.92:10:0   - 001c24.2
L1 192.168.11.92:10:1   - 101i25.2
```

```
L1 192.168.11.92:0:0    - 001r22
L1 192.168.11.92:3:0    - 001r20
L1 192.168.11.92:2:0    - 001c14.3
L1 192.168.11.92:2:1    - 101i21.3
L1 192.168.11.92:5:0    - 001c14.3
L1 192.168.11.92:5:1    - 101i21.3
L1 192.168.11.92:1:0    - 001c11.4
L1 192.168.11.92:1:1    - 001i07.4
L1 192.168.11.92:4:0    - 001c11.4
L1 192.168.11.92:4:1    - 001i07.4
```

3. See the explanation of the output from the `cfg` command in Procedure 1-4.

Accessing the Console on a Partitioned System

Procedure 1-7 Access the Console on a Partitioned System

To access the console on a partition, perform the following steps:

1. Use the L2term to connect to the L2 controller of the system.

   Note: If you are connected to the L2 controller, but do not have the L2 prompt, try typing the following: `CTRL-t`.

2. To see output that shows which C-bricks have system consoles, enter the `sel` command without options on a partitioned system as follows:

```
cranberry-192.168.11.92-L2>sel

     known system consoles (partitioned)

             partition  1: 001c31 - L2 detected
             partition  2: 001c24 - L2 detected
             partition  3: 001c14 - L2 detected
             partition  4: 001c11 - L2 detected

     current system console

     console input: not defined
     console output: not filtered
```

   The output from the `sel` command shows that there are four partitions defined.

3. To get to the console of partition 2, for example, enter the following:

   ```
   cranberry-192.168.11.92-L2>sel p 2
   ```

4. To connect to the console of partition 2, enter `Ctrl-d`.

When a system is partitioned, the L2 prompt shows the partition number of the partition you selected, as follows:

```
cranberry-001-L2>sel p 2
console input: partition 2, 001c24 console0
console output: any brick partition 2
cranberry-001-L2:p2>
```

Unpartitioning a System

Procedure 1-8 Unpartitioning a System

To remove the partitions from a system, perform the following steps:

1. Use the L2term to connect to the L2 controller of the system.

   Note: If you are connected to the L2 controller, but do not have the L2 prompt, try typing the following: `CTRL-t`.

2. Shut down the Linux operating system running on each partition before unpartitioning a system.
3. To set the partition ID on all bricks to zero, enter the following command:

   ```
   cranberry-192.168.11.92-L2>r * brick part 0
   ```

4. To confirm that all the partitions on your system have been removed, enter the following command:

   ```
   cranberry-192.168.11.92-L2>cfg
   ```

   The list of bricks no longer have a dot followed by a number in their name (see "Determining If a System is Partitioned" on page 34).

5. To reset all of the bricks, enter the following command:

   ```
   cranberry-192.168.11.92-L2>r * rst
   ```

6. To get to the system console for the newly unpartitioned system, you need to reset the select setting as follows:

   ```
   cranberry-192.168.11.92-L2>sel reset
   ```

7. To get the console (assuming you still have the L2 prompt), enter `Ctrl-d`.

Connecting the System Console to the Controller

System partitioning is an administrative function. The system console is connected to the controller as required by the configuration selected when an SGI ProPack system is installed. For additional information or recabling, contact your service representative.

Making Array Services Operational

This section describes how to get Array Services operational on your system. For detailed information on Array Services, see chapter 3, “Array Sevices”, in the *Linux Resource Administration Guide*.

Procedure 1-9 Making Array Services Operational

To make Array Services operational on your system, perform the following steps:

Note: Most of the steps to install array services is now performed automatically when the array services RPM is installed. To complete installation, perform the steps that follow.

1. Make sure that the setting in the `/usr/lib/array/arrayd.auth` file is appropriate for your site.

   ⚠ **Caution:** Changing the `AUTHENTICATION` parameter from *NOREMOTE* to *NONE* may have a negative security impact on your site.

2. Make sure that the list of machines in your cluster is included in one or more array definitions in `/usr/lib/array/arrayd.conf` file.

3. To determine if Array Services is correctly installed, run the following command:

   ```
   array who
   ```

You should see yourself listed.

Pluggable Authentication Modules on ProPack 3 Systems

Note: This section only applies to SGI systems running ProPack 3 for Linux.

Pluggable Authentication Modules (PAM) are a suite of shared libraries that enable the local system administrator to choose how applications authenticate users. In other words, without rewriting and recompiling a PAM-aware application, it is possible to switch between the authentication mechanism(s) it uses. You may entirely upgrade the local authentication system without touching the applications themselves.

SGI ProPack for Linux is integrated with PAM and PAM is configured on during system installation.

For additional information go to this location:
http://www.kernel.org/pub/linux/libs/pam

Network File System Configuration on ProPack 3 Systems

Note: This section only applies to SGI systems running ProPack 3 for Linux.

This section provides information about configuring network file system (NFS) on an Altix system and covers these topics:

- "Setting NFS Block Size" on page 39
- "Determining the Optimum NFS Block Size" on page 39
- "NFS Error Conditions" on page 39

Setting NFS Block Size

To set NFS block size, you can use the rsize and wsize mount options to the mount(8) command or modify the rsize and wsize parameters in the /etc/fstab file, as follows:

```
# /etc/fstab
my-server:/some/path    /mnt/mountpoint   nfs   noauto,rw,rsize=32768,wsize=32768  0 0
```

Determining the Optimum NFS Block Size

Generally, for better performance you want the largest block size possible. This is especially true if your workload is dominated by large sequential reads and writes. However, the choice of block size is limited by various factors, as follows:

- Normally the limit is silently enforced so you will never know when you have crossed it.
- The minimum useful block size on Altix clients is 16 KB. Note that is is larger than the default.
- The largest block size a Linux client will allow is 32 KB.
- With NFS version 2, the protocol will not support block sizes above 8 KB. Avoid using version 2.
- With NFS version 3, an upper limit on the block size is advertised by the NFS server at mount time. This value depends on the transport protocol and NFS server OS. For SGI ProPack for Linux v3.0 use 32 KB blocks for UDP and TCP transport protocols.
- Linux NFS clients will round the block size to a page size which is 16 KB on an Altix systems.

Generally, SGI recommends using 32–KB block size on Altix systems.

NFS Error Conditions

This section describes typical NFS error conditions you may encounter. It covers these topics:

- "I/O Errors Copying Large Files" on page 40

- "`exportfs` Complains About `sync` Option" on page 40

I/O Errors Copying Large Files

If you encounter I/O errors copying large files, check your `mount`(8) options. If you are using the `soft` `mount` option, try remounting your filesystem without it. The `soft` option does not work properly on Linux. Consider using the `intr` option instead, or if you are using the `soft` option, try increasing the timeout value with the `timeo` `mount` option.

`exportfs` Complains About sync Option

If you run `exportfs` and got a message similar to the following and want to eliminate it:

```
exportfs: /etc/exports [2]: No ’sync’ or ’async’ option specified for export "*:/scratch".
   Assuming default behaviour (’sync’).
   NOTE: this default has changed from previous versions
```

edit the`/etc/exports` file and add the `sync` option to all the export entries which do not have either `sync` or the opposite `async` option. Also, if there are any entries that contain the `async` option, change them to `sync`. The `async` option not only breaks the NFS protocol specification, but also leads to worse performance for many workloads. This is why it is no longer the default.

Chapter 2

System Operation

This chapter describes the operation of an SGI Altix 3000 series computer systems. It covers the following topics:

- "Booting a System" on page 41
- "Halting the System" on page 45
- "Getting Console Access" on page 47
- "Troubleshooting an SGI Altix System" on page 48

Booting a System

This section describes how to boot an SGI Altix series computer system.

Procedure 2-1 Booting a System

To boot an SGI Altix 3000 series computer system, perform the following:

1. Obtain the system console as described in "Getting Console Access" on page 47 if you are using SGIconsole or telnet to the L2 controller as described in "Connecting to the L2 Controller" on page 46.

2. By default, when booting a menu of boot options appears. On a properly configured system (as shipped from the factory), you can boot directly to the Linux operating sytem. A system administrator can change the default using the boot mainenance menus, the extensible firmware interface (EFI) shell> command, or the `efibootmgr` command (see the `efibootmgr` options usage statement for more information).

   Note: To see the boot menus properly on an SGI Altix 3000 system, make sure sure the debug switches are set to **`0`** or **`1`**.

   A screen similar to the following appears after booting your machine:

```
EFI Boot Manager ver 1.02 [12.38]
```

```
Partition  0:                           Enabled Disabled
    CBricks          1         Nodes          2        0
    RBricks          0         CPUs           4        0
    IOBricks         1         Mem(GB)        4        0

Please select a boot option

    UnitedLinux
    ProPack
    Boot option maintenance menu


    Use the arrow keys to change option(s).  Use Enter to select an option
```

One of the menu options appears highlighted. Pressing arrow keys moves the highlight.

An example of selecting the **EFI Boot Maintenance Manager** menu is, as follows:

```
EFI Boot Maintenance Manager ver 1.02 [12.38]

Main Menu. Select an Operation


        Boot from a File
        Add a Boot Option
        Delete Boot Option(s)
        Change Boot Order

        Manage BootNext setting
        Set Auto Boot TimeOut

        Select Active Console Output Devices
        Select Active Console Input Devices
        Select Active Standard Error Devices

        Cold Reset
        Exit
```

An example of selecting the **Boot from a File** option is, as follows:

```
EFI Boot Maintenance Manager ver 1.02 [12.38]

Boot From a File.  Select a Volume


    NO VOLUME LABEL [Pci(1|1)/Scsi(Pun0,Lun1)/HD(Part1,Sig7CFD016D-A
    NO VOLUME LABEL [Pci(1|1)/Scsi(Pun0,Lun2)/HD(Part1,Sig1F9EFFAD-7
    Default Boot [Pci(1|1)/Scsi(Pun0,Lun1)]
    Default Boot [Pci(1|1)/Scsi(Pun0,Lun2)]
    Load File [Pci(4|0)/Mac(08006913DB7D)/NicName(tg0)]
    Load File [EFI Shell [Built-in]]
    Exit
```

An example of selecting Load File **EFI Shell** is, as follows:

```
    Device Path VenHw(D65A6B8C-71E5-4DF0-A909-F0D23000000099B40000002
BB40000005AB4000000A9)
EFI Shell version 1.02 [12.38]
Device mapping table
  fs0  : Pci(1|1)/Scsi(Pun0,Lun1)/HD(Part1,Sigg1)
  fs1  : Pci(1|1)/Scsi(Pun0,Lun2)/HD(Part1,Sigg2)
  blk0 : Pci(1|1)/Scsi(Pun0,Lun1)
  blk1 : Pci(1|1)/Scsi(Pun0,Lun1)/HD(Part1,Sigg1)
  blk2 : Pci(1|1)/Scsi(Pun0,Lun1)/HD(Part2,Sigg3)
  blk3 : Pci(1|1)/Scsi(Pun0,Lun1)/HD(Part3,Sigg4)
  blk4 : Pci(1|1)/Scsi(Pun0,Lun2)
  blk5 : Pci(1|1)/Scsi(Pun0,Lun2)/HD(Part1,Sigg2)
  blk6 : Pci(1|1)/Scsi(Pun0,Lun2)/HD(Part2,Sigg5)
  blk7 : Pci(1|1)/Scsi(Pun0,Lun2)/HD(Part3,Sigg6)
Shell>
```

From the system EFI shell> prompt, you can proceed with booting the system. Optional booting steps follow:

You have the option to select an EFI partition you want to load your kernel as follows. If you choose not to do this, `fs0:` is searched by default and the following prompt appears:

```
fs0:
```

If there are multiple EFI filesystems (for example, `fs1`, `fs2`, and so on), change to the one you from which you wish to load the kernel. Then perform the following:

a. On an SGI ProPack 3 system, to boot the default kernel from the first boot disk, enter the following command at the prompt:

   ```
   Booting disk 1:efi/sgi/elilo
   ```

   On an SGI ProPack 4 system, to boot the default kernel from the first boot disk, enter the following command at the prompt:

   ```
   Booting disk 1:efi/SuSE/elilo
   ```

b. To boot the default kernel from the second root disk, change directory (`cd`) to `efi/sgi` (`efi/SuSE` on SGI ProPack 4) and enter the following command at the prompt:

```
Booting disk 2:elilo sgilinux root=/dev/xscsi/pci01.03.0-1/target2/lun0/part3
```

   The preceding XSCSI path points to the seoncd disk of the primary IX-brick.

3. If the system is at the kernel debugger kdb> prompt or is not responding at all, try resetting the system from the system controller. To get to the system controller prompt, enter Ctrl-T. At the system controller (L1 or L2) prompt, enter `rst`. The EFI shell> promt appears.

4. To give up control of the console, perform one of the following:

   - For k consoles, enter Ctrl-] and then enter Ctrl-D.
   - To exit a session from from SGIconsole, from the Console Manager **File** pulldown menu, choose **Exit**. To exit from the SGIconsole text-based user interface (tscm), enter **8** for quit.
   - To exit a session in in either the graphical or text version of IRISconsole, enter the following: **~x**.
   - To end a `telnet` session to the the L2 controller, enter the following: Ctrl-] and then Ctrl-D.

Halting the System

This section describes how halt the Linux operating sytem and power down your system.

Procedure 2-2 Halting the System

To halt the Linux operating sytem and power down your system, perform the following:

1. Connect to the L2> controller by following the steps in Procedure 2-4 on page 46.
2. Enter Ctrl-D to connect to the system console.
3. Enter the `halt` command.

   You can also reset the system by enter Ctrl-T to get to the L2> prompt and then enter the `rst` command to reset the system.
4. To to power on and power off individual bricks or your entire Altix 3000 series system, see “Powering the System On and Off” in Chapter 1, “Operation Procedures” in the *SGI Altix 3000 User's Guide*.

Connecting to the L1 Controller

You can monitor the L1 controller status and error messages on the L1 controller's liquid crystal display (LCD) located on the front panel of the individual bricks. The L1 controller and L2 controller status and error messages can also be monitored at your system console. The system console allows you to monitor and manage your server or graphics system by entering L1 controller commands. You can also enter L2 controller commands to monitor and manage your system if your system has L2 controller hardware and a system console or if you are using an SGIconsole as your system console. For information on connecting to the system console, see "Getting Console Access" on page 47. For detailed information on using the L2 controller software, see the *SGI L1 and L2 Controller Software User's Guide*

Procedure 2-3 Connecting to the L1 Controller

1. From the `Tasks` pulldown menu of SGIconsole Console Manager GUI, choose **Connect to a System Controller**.
2. To get to the L2> controller prompt, enter Ctrl -T.
3. To get back to the L1> controller prompt, enter Ctrl-D.

Connecting to the L2 Controller

To access the L2 controller firmware, you must connect a system console such as SGIconsole or a dumb terminal, to the L2 controller. The L2 firmware is always running as long as power is supplied to the L2 controller. If you connect a system console to the L2 controller's console port, the L2 prompt appears. For instructions on connecting a console to the L2 controller, see your server or graphics system owner's guide or the *SGIconsole Hardware Connectivity Guide*.

The SGIconsole Console Manager graphical user interface (GUI) or text-based user interface (`tscm`(1)), can be used to securely access a system console and connect to an L2 controller. For information on using Console Manager to access an SGI Altix system or to access an SGI Altix system in secure mode using the `ssh`(1) command, see the *Console Manager for SGIconsole Administrator's Guide*.

Your SGI Altix 3000 system should have an L2 controller on your network that you can access. This section describes how you can connet to an L2 controller if you are not using SGIconsole.

Procedure 2-4 Connecting to the L2 Controller

To connect to a system L2 controller, perform the following steps:

1. From the `Tasks` pulldown menu of SGIconsole Console Manager GUI, choose **Node Tasks -> Get/Steal/Spy**. Follow the instructions in the *Console Manager for SGIconsole Administrator's Guide* to connect to the console. You can also use the `tscm`(1) command line interface to Console Manager. If you do not have SGIconsole installed, proceed to the next step.

2. Use the `telnet`(1) command to connect to the L2 controller as follows:

   ```
   telnet L2-system-name.domain-name.company.com
   ```

   The *system-name* argument is the name of the SGI Altix system to which you want to connect. In some case, you may need to use the full domain such as *system-name.americas.sgi.com*.

3. Once connected, press the Enter key and a prompt similar to the following appears:

   ```
   system_name-001-L2>
   ```

4. To connect to the system console, enter Ctrl–D.

If your system is partitioned, a message similar to the following appears:

```
INFO: ERROR: no system console defined
```

For information on working with partitioned systems, see "System Partitioning " on page 22.

Note: For detailed information on using the L2 controller software, see the *SGI L1 and L2 Controller Software User's Guide* and the *SGI Altix 3000 User's Guide*.

Getting Console Access

This section describes how to access a system console.

Procedure 2-5 Getting Console Access

1. From the `Tasks` pulldown menu of SGIconsole Console Manager GUI, choose **Node Tasks -> Get/Steal/Spy**. Follow the instructions in the *Console Manager for SGIconsole Administrator's Guide* to connect to the console. You can also use the `tscm`(1) command line interface to Console Manager. For information on using Console Manger or `tscm`(1), see *Console Manager for SGIconsole Administrator's Guide*.

   If you do not have Console Manager installed, proceed to the next step.

2. To connect to a system L2 controller, perform the steps in Procedure 2-4 on page 46.

3. Once connected, press the Enter key and a prompt similar to the following appears:

   ```
   system_name-001-L2>
   ```

4. To connect to the system console, enter Ctrl–D.

   If your system is partitioned, a message similar to the following appears:

   ```
   INFO: ERROR: no system console defined
   ```

   For information on working with partitioned systems, see "System Partitioning " on page 22.

5. To return to the L2 controller, enter Ctrl-T.

To return to the telnet prompt, enter Ctrl-] (control -right bracket).

Note: For detailed information on using the L2 controller software, see the *SGI L1 and L2 Controller Software User's Guide*.

Troubleshooting an SGI Altix System

This section describes procedures for troubleshooting an SGI Altix system that appear to have crashed or hung. It is applicable to SGI ProPack 2.4 for Linux and SGI ProPack 3 for Linux.

Procedure 2-6 Debugging Hangs or Crashes on Altix Systems with an L3 Controller

This procedure can be used with SGI Altix systems that have an L3 controller.

1. Connect to the L2 controller via a serial port or Ethernet connection.

   Ensure that you use a tool such as `script`(1) to capture all output from your console session. For example, if you connect from an L3 controller, use the following:

   ```
   # script
    Script started, file is typescript
   # /stand/sysco/bin/l2term --l2 IP address of the Altix L2
   ```

2. Escape to the console by entering `Ctrl-D`. If there is already a [0]kdb> prompt, skip to the next step. If not, enter KDB by typing `Esc-KDB`(note ALL capital letters) at the hung console.

3. Once in kdb, issue the `cpu` command to find CPUs that are hung, as follows:

   ```
   [0]kdb> cpu
   ```

   CPUs that are hung will have a '* 'next to the CPU number.

4. Reset hung CPUs by issuing the `init` command. For example, to reset cpu 19 perform the following:

   ```
   [0]kdb> init 19
   ```

5. Make sure all hung CPUs have been initialized by issuing the cpu command again, as follows:

   ```
   [0]kdb> cpu
   ```

6. Collect data from sn2kdb, as follows:

   ```
   [0]kdb> sn2kdb
   ```

7. Enter POD to collect error information from individual CPUs. Issue the pod command in KDB to get into pod, as follows:

   ```
   [0]kdb> pod
   ```

   Now issue the error a and error commands to obtain the hardware error state, as follows:

   ```
   POD SysCt (RT) Cac> error a
   POD SysCt (RT) Cac> error
   ```

   Exit POD by issuing the exit command, as follows:

   ```
   POD SysCt (RT) Cac> exit
   ```

8. Attempt to obtain a dump by issuing the go command from KDB, as follows:

   ```
   [0]kdb> go
   ```

9. If the system does not start dumping, try issuing the nmi command from the L2 controller, as follows: (Get into the L2 controller by entering Ctrl-t from the console)

   ```
   L2> nmi
   ```

10. If the system does not dump, make sure that the errdmp command running on the L3 has completed and issue a reset, as follows:

    ```
     L2> rst
    ```

11. Continue collecting data until the system is up in multiuser mode and a login prompt is seen at the console. At that time, the script output can be terminated and should be saved and sent to SGI.

If submitting a case to SGI, please provide copies of console output from the above procedures, /var/log/messages, crash dumps from /var/log/dump, SAL records from /var/log/salinfo/decoded, and lkcd dump from /var/log/dump.

For more information on L2 controller commands, see the *SGI L1 and L2 Controller Software User's Guide*.

Procedure 2-7 Debugging Hangs or Crashes on Altix Systems without an L3 Controller

This procedure can be used with SGI Altix systems **without** an L3 controller.

1. Connect to the L2 controller via a serial port or Ethernet connection.

   Ensure that you use a tool such as `script`(1) to capture all output from your console session.

2. Ensure that the console selection is set up correctly, as follows:

   ```
   ctrl-t                (Escape to the L2)
   ?-001-L2>l2           (Select the L2)
   ?-001-L2>sel reset    (Reset selections to defaults)
   console input: 001c11 console0
   console output: not filtered

   ?-001-L2>sel        (ensure that the selection is correct)
                       known system consoles (nonpartitioned)

   001c11 - L2 detected

    current system console

   console input: 001c11 console0
   console output: not filtered
   ```

3. Select the desired partition (for partitioned systems only; for nonpartitioned systems, skip to Step 4), as follows:

   ```
   ?-001-L2>sel p 1    (where "1" is the
                                          desired partition number)
   ```

4. Determine which L1 controllers are functional, as follows:

   ```
   ?-001-L2>cfg

   L2 163.154.17.66: - 001 (LOCAL)
   L1 163.154.17.66:0:0     - 001c11
   ```

```
L1 163.154.17.66:0:1     - 002i01
L1 163.154.17.66:0:5     - 001c14
```

Note: In systems with routers, each C-brick may show up twice. This is normal.

```
?-001-L2>pwr
001c10:
power appears on
001c13:
power appears on
001r16:
power appears on
```

If some L1 controllers are missing, reseat the USB connections between the R-bricks and the L2 controller.

5. Record the LED and port status, as follows:

```
ctrl-t     (escape to the L2)
?-001-L2>l2                   (select the L2)
?-001-L2>leds
001c11:
CPU 0A: 0x3c:   SAL calling OS_INIT
CPU 0C: 0x3c:   SAL calling OS_INIT

CPU 1A: 0x3c:   SAL calling OS_INIT
CPU 1C: 0x3c:   SAL calling OS_INIT

001c14:
CPU 0A: 0x3c:   SAL calling OS_INIT
CPU 0C: 0x3c:   SAL calling OS_INIT

CPU 1A: 0x3c:   SAL calling OS_INIT
CPU 1C: 0x3c:   SAL calling OS_INIT


?-001-L2>port     (record the link LED status;
                              a missing link can cause a hang)
001c11:
Port Stat Remote Pwr Local Pwr  Link LED SW LED
---- ---- ---------- ---------- -------- --------
```

```
A 0x0f       okay       okay       on       on
B 0x0f       okay       okay       on       on
C 0x0f       okay       okay       on       on
D 0x02       none       okay      off      off
001c14:
Port Stat Remote Pwr Local Pwr  Link LED SW LED
---- ---- ---------- ---------- -------- --------
A 0x0f       okay       okay       on       on
B 0x02       none       okay      off      off
C 0x0f       okay       okay       on       on
D 0x02       none       okay      off      off
```

6. Determine whether the system has hung or crashed, as follows:

   Try to ping the system. Also perform the following actions at the console:

   ```
   ctrl-d     (enter console mode)
   ```

   Type '#' followed by Enter a few times and look for a response. If you get a line feed, it means the kernel is still running. If the system is hung, you will typically see the following message:

   ```
   "no response from 001c10 console, system not responding"
   ```

   If the system has dropped to the `kdb>` prompt, then it has crashed and is not hung.

7. Drop to KDB, as follows:

   If the system is not already at the kdb> prompt, drop to KDB by entering `Esc` **`KDB`**. If you see a message such as the following;

   ```
   "127 out of 128 cpus in kdb, waiting for the rest"
   ```

   wait for KDB to respond. It may take up to a minute. If the system does not respond to `Esc` KDB, issue an NMI (nonmaskable interrupt) from the L2 controller, as follows:

   ```
   ctrl-t
   ?-001-L2>nmi
   ```

   If you see a message of the form

   ```
   "1 cpu is not in kdb, its state is unknown"
   ```

issue the **cpu** command to determine which CPU(s) are hung, as follows:

```
[0]kdb> cpu
Currently on cpu 0
Available cpus: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,

14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73*, 74, 75, 76, 77,
78, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110,
111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123,
124, 125, 126, 127

The CPU(s) which are hung will be marked with a ’*’.  Issue an
init to those CPU(s):
 [0]kdb> init 73
```

Note: Issuing an `init` does not remove the ’*’ when you issue the **cpu** command again.

---

8. Capture and print field replaceable unit (FRU) information, as follows:l

```
ctrl-t
?-001-L2>fru capture
354/436 MMRs captured (status 0
ctrl-t
?-001-L2>fru print
```

9. Record information from KDB, as follows:

   At the `kdb>` prompt, enter the following:

```
[0]kdb> sn2kdb
```

   This will produce much output.

10. On an SGI ProPack 3 system, use LKCD to take a crash dump, as follows:

At the kdb> prompt, enter the following:

```
[0]kdb> sr c
```

On an SGI ProPack 4 system, use LKCD to take a crash dump, as follows:

At the kdb> prompt, enter the following:

```
[0]kdb> sr d
```

You should see output similar to the following:

```
Start a Crash Dump (If Configured)
Dumping from interrupt handler !
Uncertain scenario - but will try my best
```

If the dump is successful, the system will reset, and you will see the following message as it boots:

```
Configuring system to save crash dumps [ OK ]
Generating crash report - this may take a few minutes
```

The crash dump will be saved in a numbered directory under `/var/log/dump`.

If submitting a case to SGI, please provide copies of console output from the above procedures, `var/log/messages`, crash dumps from `/var/log/dump`, and SAL records from `/var/log/salinfo`.

For more information on L2 controller commands, see the *SGI L1 and L2 Controller Software User's Guide*.

Procedure 2-8 Debugging Hangs or Crashes on Altix Systems **without** an L2 Controller

Debugging Hangs or Crashes on Altix Systems without an L2 Controllter.

If an L3 is used and the connectivity is via an L2 emulator, instructions in Section I can be used to gather the data. Otherwise, use the following steps, when a terminal is connected directly to the L1 system controller.

1. Connect to the L1 controller via a serial port.

   Ensure that you use a tool such as `script`(1) to capture all output from your console session.

2. Determine which L1 controllers are functional, as follows:

```
001c18-L1>cfg
:0  - 001c18
:1  - 001i28

001c18-L1>* pwr
001c18:
Supply          State Voltage    Margin  Value
--------------  ----- ---------  ------- -----
           48V     on       N/A      N/A
      12V bias     on   12.063V      N/A
          2.5V     on    2.509V   normal   125
          1.8V     on    1.859V   normal   112
          1.2V     on    1.201V   normal    75
     Itanium 0     on       N/A      N/A
     Itanium 1     on       N/A      N/A
     Itanium 2     on       N/A      N/A
     Itanium 3     on       N/A      N/A
      3.3V aux     NC    3.285V      N/A
        5V aux     NC    4.966V      N/A
001i28:
Supply          State Voltage    Margin  Value
--------------  ----- ---------  ------- -----
           48V     on       N/A      N/A
      12V bias     on   12.125V      N/A
          1.8V     on    1.807V   normal    92
          2.5V     on    2.509V   normal   132
          3.3V     on    3.302V   normal   122
            5V     on    5.018V   normal   164
           12V     on   12.000V      N/A
          -12V     on  -12.052V      N/A
      3.3V aux     NC    3.320V      N/A
        5V aux     NC    4.966V      N/A
```

3. Record the LED and port status, as follows:

```
ctrl-t                  (escape to the L1)
001c18-L1>* leds
001c18:
CPU 0A: 0x00:   Kernel: CPU idle
```

```
        0x01:   Kernel: CPU idle
CPU 0C: 0x01:   Kernel: CPU idle
        0x00:   Kernel: CPU idle
        0x02:   Kernel: CPU busy

CPU 1A: 0x00:   Kernel: CPU idle
        0x01:   Kernel: CPU idle
CPU 1C: 0x00:   Kernel: CPU idle
        0x01:   Kernel: CPU idle

 001i28:
 INFO: command not supported on this brick type

001c18-L1>* port        (record the link LED status;
                                    a missing link can cause a hang)
001c18:
Port Stat Remote Pwr Local Pwr  Link LED SW LED
---- ---- ---------- ---------- -------- --------
   A 0x02       none       okay      off      off
   B 0x0f       okay       okay       on       on
   C 0x02       none       okay      off      off
   D 0x02       none       okay      off      off
001i28:
Port Stat Remote Pwr Local Pwr  Link LED SW LED
---- ---- ---------- ---------- -------- --------
   A 0x07       okay       okay       on       on
   B 0x0a       none       okay      off      off
   C ERROR I2C:not present getting status.
   D ERROR I2C:not present getting status.
```

4. Determine whether the system has hung or crashed, as follows:

   Try to ping the system. Also perform the following actions at the console:

   ```
   ctrl-d       (enter console mode)
   ```

   Type '#' followed by `Enter` a few times and look for a response. If you get a line feed, it means the kernel is still running. If the system is hung, you will typically see the following message:

   ```
   "no response from 001c18 console, system not responding"
   ```

If the system has dropped to the `kdb>` prompt, then it has crashed and is not hung.

5. Drop to KDB:

   If the system is not already at the `kdb>` prompt, drop to KDB by typing **`Esc KDB`**.

   If you see a message such as follows:

   ```
   127 out of 128 cpus in kdb, waiting for the rest"
   ```

   Wait for KDB to respond. It may take up to a minute.

   If the system does not respond to KDB, issue an NMI (nonmaskable interrupt) from the L1 controller, as follows:

   ```
   ctrl-t
   001c18-L1>nmi
   ```

   If you see a message of the form `"1 cpu is not in kdb, its state is unknown"`, issue the `cpu` command to determine which CPU(s) are hung, as follows:

   ```
   [0]kdb> cpu
   Currently on cpu 0
   Available cpus: 0, 1, 2*, 3
   ```

   The CPU(s) which are hung will be marked with a ′*′. Issue an `init` to those CPU(s), as follows:

   ```
   [0]kdb> init 2
   ```

   Note: Issuing an init does not remove the ′*′ when you issue the `cpu` command again.

6. Capture and print FRU information, as follows:

   ```
   ctrl-t
   001c18-L1>* fru capture
   001c18:
   354/436 MMRs captured (status 0
   001i28:
   INFO: command not supported on this brick type
   ```

```
ctrl-t
001c18-L1>* fru print
```

7. Record information from KDB, as follows:

   At the kdb> prompt, enter the following:

   ```
   [0]kdb> sn2kdb
   ```

   This produces much output.

8. On an SGI ProPack 3 system, use LKCD to take a crash dump, as follows:

   At the kdb> prompt, enter the following:

   ```
   [0]kdb> sr c
   ```

   On an SGI ProPack 4 system, use LKCD to take a crash dump, as follows:

   At the kdb> prompt, enter the following:

   ```
   [0]kdb> sr d
   ```

   You should see output similar to the following:

   ```
   Start a Crash Dump (If Configured)
   Dumping from interrupt handler !
   Uncertain scenario - but will try my best
   ```

   If the dump is successful, the system will reset, and you will see the following message as it boots:

   ```
   Configuring system to save crash dumps [ OK ]
   Generating crash report - this may take a few minutes
   ```

   The crash dump will be saved in a numbered directory under /var/log/dump.

If submitting a case to SGI, please provide copies of console output from the above procedures, /var/log/messages, crash dumps from /var/log/dump, and SAL records from /var/log/salinfo.

For more information on L2 controller commands, see the *SGI L1 and L2 Controller Software User's Guide*.

Recovering a Damaged Root Filesystem

The following section describes how to recover a damaged root filesystem.

Procedure 2-9 Recovering a Damaged Root Filesystem

If your root filesystem should become damaged, use the following recovery steps:

1. Insert SGI Advanced Linux Environment 2.1.1 CD1 into the system's CD-ROM drive and restart the system.
2. While the system is powering up, check the device mapping table. It should look similar to the following:

```
Device mapping table

fs0  : Pci(2|1)/Ata(Primary, Master)/CDROM(Entry1)This is the CD-ROM
fs1  : Pci(1|1)/Scsi(Pun0/Lun1)/HD(Part1,Sig00000000)
fs2  : Pci(1|1)/Scsi(Pun0/Lun2)/HD(Part1,Sigg1)
blk0 : Pci(2|1)/Ata(Primary, Master)
blk1 : Pci(2|1)/Ata(Primary, Master)/CDROM(Entry1)
blk2 : Pci(1|1)/Scsi(Pun0/Lun1)
blk3 : Pci(1|1)/Scsi(Pun0/Lun1)/HD(Part1,Sig00000000)
blk4 : Pci(1|1)/Scsi(Pun0/Lun1)/HD(Part2,Sig00000000)
blk5 : Pci(1|1)/Scsi(Pun0/Lun1)/HD(Part3,Sig00000000)
blk6 : Pci(1|1)/Scsi(Pun0/Lun1)/HD(Part4,Sig00000000)
blk7 : Pci(1|1)/Scsi(Pun0/Lun2)
blk8 : Pci(1|1)/Scsi(Pun0/Lun2)/HD(Part1,Sigg10)
```

3. At the Shell> prompt, type the CD-ROM device name, as follows:

```
Shell> fs0:        Type this to change to the CD-ROM device
fs0:\>
```

4. Boot the CD by entering **`elilo rescue console=ttyS0`**, as follows:

```
fs5:\> cd efi\boot
fs5:\efi\boot> elilo rescue console=ttyS0   This starts the boot loader
```

   If you are using SGIconsole, Boot the CD by entering **`elilo rescue console=ttyS0`**, as follows:

   ```
   fs5:\> cd efi\boot
   fs5:\efi\boot> elilo rescue console=ttyS0,38400n8
   ```

5. The CD will then boot and start the rescue process.

6. Select the language you prefer to use for the rescue process. To continue, select **OK**.

   Note: Console support for Asian languages is not available at this time. SGI recommends that customers in Asia set the language to English.

7. The next screen lets you choose the media containing the rescue image. Currently, **Local CDROM** is the only choice. To continue, select **OK**.

8. On the next screen, you have the option of mounting your root drive. If your root filesystem has been damaged, it is important to attempt to mount it before running the `xfs_repair` command. This ensures that valuable data from the journal is recovered.

   If you select **Continue** and the installer hangs or crashes, you will have to restart this process and select **Skip**. Then you can attempt to mount the filesystem manually. For information on how to do this, see "Manually Mounting the Root Filesystem" on page 61.

9. If successful, the next screen informs you that your system can be found under `/mnt/sysimage`. To enter the rescue shell, select **OK**.

10. At this point, administrators can inspect and attempt to repair or recover any damaged or missing software. What follows is an example of how you might repair a corrupted root filesystem.

    While it is important to mount your filesystem to recover any data contained in the journal, you must then unmount the filesystem before running the `xfs_repair` command.

The mount command issued without arguments lists currently mounted filesystems, as follows:

```
sh-2.05# mount

rootfs on / type rootfs (rw)
devfs on /dev type devfs (rw)
/dev/root.old on / type ext2 (rw)
none on /dev type devfs (rw)
/proc on /proc type proc (rw)
/dev/pts on /dev/pts type devpts (rw)
/tmp/cdrom on /mnt/source type iso9660 (ro)
/tmp/loop0 on /mnt/runtime type cramfs (ro)
/dev/xscsi/pci01.03.0-1/target1/lun0/part3 on /mnt/sysimage type xfs (rw)
/dev/xscsi/pci01.03.0-1/target1/lun0/part1 on /mnt/sysimage/boot/efi type vfat (rw)
none on /mnt/sysimage/dev/pts type devpts (rw)
none on /mnt/sysimage/proc type proc (rw)
```

The line ending with `/mnt/sysimage` type `xfs` (`rw`) also contains the device corresponding to your root disk. In this case, `/dev/xscsi/pci01.03.0-1/target1/lun0/part3`.

11. Enter the following command to unmount the filesystem:

    ```
    umount /mnt/sysimage
    ```

12. Run `xfs_repair` on the filesystem, as follows:

    ```
    xfs_repair /dev/xscsi/pci01.03.0-1/target1/lun0/part3
    ```

13. After the `xfs_repair` command completes, the filesystem has been repaired and you can reset your system.

Manually Mounting the Root Filesystem

The following section describes how to manually mount the root filesystem.

Procedure 2-10 Manually Mounting the Root Filesystem

If the installer could not locate your root filesystem, you can locate and mount it manually. This section describes that process. For details, see *XFS for Linux Administration*. In rare cases, mounting a corrupted filesystem could cause the mount command to hang or have trouble. If that happens, see *XFS for Linux Administration* for instructions on how to run xfs_repair without mounting first.

1. The following example assumes that your root filesystem is on partition 3 and that it resides on a disk in the first disk bay of the IX-brick. To view the partitions available on the system, issue the following command:

```
sh-2.05# cat /proc/partitions
major minor #blocks name rio rmerge rsect ruse wio wmerge wsect wuse running use aveq
 5     0    125470 xscsi/pci01.01.0/target0/lun0/disc 218 4524 18968 10952 0 0 0 0 0 10952 10952
 4     0  35843686 xscsi/pci01.03.0-1/target1/lun0/disc 32 101 692 69 2 14 512 10 0 80 80
 4     1    513008 xscsi/pci01.03.0-1/target1/lun0/part1 0 0 0 0 0 0 0 0 0 0 0
 4     2   9438208 xscsi/pci01.03.0-1/target1/lun0/part2 0 0 0 0 0 0 0 0 0 0 0
 4     3  25891840 xscsi/pci01.03.0-1/target1/lun0/part3 28 29 584 50 2 14 512 10 0 61 61
 4    16  35843686 xscsi/pci01.03.0-1/target2/lun0/disc 4 72 108 19 0 0 0 0 0 19 19
 4    17    513008 xscsi/pci01.03.0-1/target2/lun0/part1 0 0 0 0 0 0 0 0 0 0 0
 4    18   9438208 xscsi/pci01.03.0-1/target2/lun0/part2 0 0 0 0 0 0 0 0 0 0 0
 4    19  25891840 xscsi/pci01.03.0-1/target2/lun0/part3 0 0 0 0 0 0 0 0 0 0 0
```

2. When the system is running normally, the XSCSI pathname in the following example represents the system disk in bay 1 of the IX-brick. This example assumes that you put the root filesystem on partition 3:

   ```
   /dev/xscsi/pci01.03.0-1/target1/lun0/part3
   ```

3. To manually mount the filesystem, enter the following commands:

```
sh-2.06# mkdir /mnt/sysimage
sh-2.07# mount -t xfs /dev/xscsi/pci01.03.0-1/target1/lun0/part3 /mnt/sysimage
```

Diskless Booting

All of the concepts for diskless booting have been covered thoroughly by Linux HOWTO documents. SGI recommends that you use your favorite web search engine and looking for "linux diskless boot howto". You should find ample information on the concepts and methods available through Linux. For specifics of net booting with elilo, see the documentation provided with the elilo package (/usr/share/doc/elilo*/netbooting.txt).

SGI has provided a device driver with the SGI ProPack for Linux kernels that abstracts the memory-to-memory transfer capabilities of the hardware into a network adapter. The network adapters name is xp0 provided by the xpnet.o module (xpnet.ko module on SGI ProPack 4).

The SGI PROM provides a network adapter compatible with the kernel's adapter. This device supports broadcast and peer-to-peer packets and is sufficient for diskless boot and diskless flash support.

You will need to add a network booting option to the EFI boot menu. To do this, reset the machine to get to the EFI menu and then perform the following:

Procedure 2-11 Adding the Network Booting Option to the EFT Boot Menu

1. Select **Boot Option Maintenance Menu** before the boot timeout expires.
2. Select **Add Boot Option**
3. Select the entry that looks like: **Load File [VenMsg(...**
4. Give it a description such as "NUMAlink Network Boot"

   Note: The MAC address for the adapter is visible at the top of the screen.

5. Press **ENTER** for both the `BootOption` data type and `BootOption Data`.
6. Answer **`Yes`** to the **Save changes to NVRAM**prompt
7. Exit the menus until you get back to **EFI Boot Manager** menu.

You should now be able to select **NUMAlink Network Boot** from the menu and it will begin attempting to acquire a DHCP address over `xp0`.

On the partition that is acting as the boot server, you will need to configure the Dynamic Host Configuration Protocol `dhcpd` server to provide an IP address to the newly initializing partition. The `dhcpd` server provided with the SGI ProPack for Linux is sufficient. For specific information about configuring the `dhcpd` server, see the `dhcpd`(8) and `dhcpd.conf`(5) man pages.

The Trivial File Transfer program `tftp`(1) is used to retrieve the `elilo.efi` EFI executable that in turn retrieves the configuration file (`elilo.conf` or equivalent). The `elilo.conf` file, in turn, specifies the kernel and `initrd` image to retrieve. These steps are documented in detail in the `elilo` documentation (`netbooting.txt` described, above). For additional information, see the `tftp`(1) and `in.tftpd`(8) man pages.

The `initrd` image needs to have the `xp.o`, `xpc.o`, and `xpnet.o` modules loaded (The `xpnet.ko` module is loaded on SGI ProPack 4 systems). An IP address needs to be assigned using either `ifconfig`(8) or a DHCP client. A mount of the root file system and any other file systems needed for the boot process needs to be performed. Finally, a `pivot_root` may be used to make the newly mounted file system tree the default for the actual boot. Many different mechanisms are available to accomplish the steps of this paragraph. Details are not specific to the `xp0` adapter.

Here are some **Notes** to consider, as follows:

- `XPNET STARTUP`

  One problem with the `xp0` adapter is that it occasionally takes a brief second to be able to broadcast to all the other partitions. If you are having periodic booting problems, consider putting a brief delay to the `initrd` script between the `insmod` of the `xpnet.o` module (`xpnet.ko` module on SGI ProPack 4) and the setting of the IP address in the `initrd` script.

- The `nfsroot=` command line option of the kernel is ignored. Since the `xpnet.o` (`xpnet.ko` module on SGI ProPack 4) module is loaded within the `initrd` script, the `nfsroot` options are ignored. A `dhcp` client or a hard-coded IP address must be included with the `initrd` image.

- `BUSYBOX` – The `mount` command that is provided with the SGI ProPack for Linux `mkinitrd`(8) daemon is not able to mount via NFS. There are many other mount utilities available on the Internet that will work with NFS mounts. During our testing of diskless booting, busybox (www.busybox.net) was used as it can perform nearly all functions that typically need to be performed for diskless booting in a single statically linked package. For our testing of diskless booting, we downloaded the `busybox-1.00-pre3.tar.gz` package, selected the options we needed, including a DHCP client and NFS support in the mount command.

  Options selected in `busybox` were, as follows:

  ```
  Netowrking Utilities -> ifupdown
  Networking Utilities -> Use busybox ifconfig and route applets
  Networking Utilities -> udhcp Server/Client -> udhcp Client (udhcpc)
  Linux System Utilities -> mount
  Linux System Utilities -> Support mounting NFS file systems
  ```

Chapter 3

Kernel Tunable Parameters on SGI ProPack Servers

This section identifies and describes the settings for kernel tunable parameters appropriate for large SGI ProPack servers.

This information about Linux kernel tunable parameters is included in your Linux release and can be found in the following file on a SGI ProPack 3 or SGI ProPack 4 system:

`/usr/src/linux/Documentation/sysctl`

This section covers the following topics:

- "CPU Scheduler `/proc/sys/sched` Directory" on page 65
- "`/etc/sysconfig/dump` File" on page 66
- "Resetting System Limits" on page 68
- "Understanding `RSS`, `SIZE`, and `SHARE` Values for MPI Jobs" on page 70
- "Memory (Swap) `sysctl` Parameters" on page 71
- "Virtual Memory `hugetlb` Parameter" on page 77

CPU Scheduler `/proc/sys/sched` Directory

This section describes tunable parameters for CPU scheduling in the `/proc/sys/sched` file.

The contents of the `/proc/sys/sched` directory is similar to the following:

```
[root@profit sched]# ls
busy_node_rebalance_ratio  idle_node_rebalance_ratio_max  min_timeslice
child_penalty              max_loadbal_rejects            sched_exec_threshold
idle_node_rebalance_ratio  max_timeslice                  sched_node_threshold
```

Do not change `min_timeslice` value to be less than 10, which is the current value for `cache_decay_ticks`, or else the scheduler’s load-balancing will be adversely affected on some workloads.

Note: Be very careful in changing any of the values in this directory. You risk adversely affecting CPU scheduling performance.

/etc/sysconfig/dump File

This file contains the configuration variables for the Linux Kernel Crash Dump (LKCD) facility that creates files in the /var/log/dump directory.

The following variables defined in this directory:

- DUMP_ACTIVE

  The DUMP_ACTIVE variable indicates whether the dump process is active or not. If this variable is 0, the dump kernel process is not activated.

- DUMPDEV

  The DUMPDEV variable represents the name of the dump device. It is typically the primary swap partition on the local system, although any disk device can be used.

  Caution: Be careful when defining this value to avoid unintended problems.

- DUMPDIR

  The DUMPDIR variable defines the location where crash dumps are saved. In that directory, a file called bounds is created that is the current index of the last crash dump saved. The bounds file is updated with an incremented index once a new crash dump or crash report is saved.

  If there is an lkcd dump, LKCD could easily exceed multiple gigabytes in /var. This is why the default root filesystem is larger. For this reason, you may wish to make a separate /var/dump filesystem or change the configuration of lkcd. For more information on lkcd, see the lkcd_config(1) man page.

  To save crash dumps to a different location, change the DUMPDIR value in /etc/sysconfig/dump file.

- DUMP_SAVE

The DUMP_SAVE variable defines whether to save the memory image to disk or not. If the value is 1, the vmcore image is stored, and a crash report is created from the saved dump. If it is not set to 1, only a crash report is created and the dump is not saved. Use this option if you do not want your system's disk space consumed by large crash dump images.

- DUMP_LEVEL

  The DUMP_LEVEL variable has a number of possible values, as follows:

DUMP_NONE (0)	Do nothing, just return if called.
DUMP_HEADER (1)	Dump the dump header and first 128K bytes.
DUMP_KERN (2)	Everything in DUMP_HEADER and kernel pages only
DUMP_USED (4)	Everything except the kernel free pages.
DUMP_ALL (8)	All memory is dumped.

Note: You must use the numeric value, not the name of the variable.

- DUMP_COMPRESS

  The DUMP_COMPRESS variable indicates which compression mechanism the kernel should attempt to use for compression. The new method is not to use dump compression unless someone specifically asks for it. There are multiple types of compression available. For now, if you modprobe dump_rle , the dump_rle.o module is installed, that enables RLE compression of the dump pages. The RLE compression algorithm used in the kernel gives (on average) 40% compression of the memory image, which can vary depending on how much memory is used on the system. There are also other compression modules coming (such as gzip). The values for the DUMP_COMPRESS variable are currently, as follows:

DUMP_COMPRESS_NONE(0)	Do not compress this dump.
DUMP_COMPRESS_RLE(1)	Use RLE compression.
DUMP_COMPRESS_GZIP(2)	Use GZIP compression.

- PANIC_TIMEOUT

  The PANIC_TIMEOUT variable represents the timeout (in seconds) before reboot after a panic occurs. Typically, this is set to 0 on the system, which means the kernel sits and spins until someone resets the machine. This is not the preferred action if we want to recover the dump after the reboot.

The following is an example of a /etc/sysconfig/dump file follows:

```
DUMP_ACTIVE=1
DUMPDEV=/dev/vmdump
DUMPDIR=/var/log/dump
DUMP_SAVE=1
DUMP_LEVEL=2
DUMP_FLAGS=0
DUMP_COMPRESS=0
PANIC_TIMEOUT=5
```

Resetting System Limits

To regulate these limits on a per-user basis (for applications that do not rely on limit.h), the limits.conf file can be modified. System limits that can be modified include maximum file size, maximum number of open files, maximum stack size, and so on. You can view this file is, as follows:

```
[user@machine user]# cat /etc/security/limits.conf
# /etc/security/limits.conf
#
#Each line describes a limit for a user in the form:
#
#            #
#Where:
# can be:
#        - an user name
#        - a group name, with @group syntax
#        - the wildcard *, for default entry
#
# can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits
#
# can be one of the following:
#        - core - limits the core file size (KB)
#        - data - max data size (KB)
#        - fsize - maximum filesize (KB)
#        - memlock - max locked-in-memory address space (KB)
#        - nofile - max number of open files
#        - rss - max resident set size (KB)
#        - stack - max stack size (KB)
#        - cpu - max CPU time (MIN)
#        - nproc - max number of processes
#        - as - address space limit
#        - maxlogins - max number of logins for this user
#        - priority - the priority to run user process with
#        - locks - max number of file locks the user can hold
#
#                #

#*               soft    core            0
#*               hard    rss             10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#@student        -       maxlogins       4

# End of file
```

For instructions on how to change these limits, follow the procedure in "File Descriptor Limits for MPI Jobs" on page 69.

File Descriptor Limits for MPI Jobs

Because of the large number of file descriptors that MPI jobs require, you might need to increase the system-wide limit on the number of open files on your Altix system. The default value for the file limit resource is 1024. You can change the file descriptor limit by editing the `/etc/security/limits.conf` file.

Procedure 3-1 Increasing File Descriptor Limits for MPI Jobs

To change the default value for all users to 8196 file descriptors, perform the following:

1. Add the following line to `/etc/pam.d/login` file:

   ```
   session    required     /lib/security/pam_limits.so
   ```

2. Add the following lines to `/etc/security/limits.conf` file:

   ```
   *     soft    nofile      8196
   *     hard    nofile      8196
   ```

The default 1024 file descriptors allows for approximately 199 MPI processes per host. Increasing the file descriptor value to 8196, allows for more than 512 MPI processes per host.

Understanding RSS, SIZE, and SHARE Values for MPI Jobs

You can use the top(1) and ps(1) command to view the RSS, SIZE, and SHARE values when running MPI jobs on an Altix system. In particular the RSS number for an MPI job can seem reasonable, while the SIZE and SHARE values are much higher.

The RSS value reflects the amount of memory that has been accessed and is currently resident in memory (faulted in). If swapping occurs, this value should go down. This tracks real memory usage of an application.

The SIZE value includes the faulted in memory plus adds up all the possible pages the application _could_ fault in (pages that have been previously allocated by the mmap function). This value includes the length of memory-mapped (mmap) regions, even if they are never touched.

The SHARE value includes the faulted in memory plus adds up all the possible pages the application _could_ fault in that are marked with the MAP_SHARED attribute. This value includes the length of MAP_SHARED memory-mapped (mmap) regions, even if they are never touched.

The reason that the SIZE and SHARE values are so high for MPI jobs, is that MPI cross-maps a significant amount of memory from each MPI process onto every other MPI process via XPMEM. This is done to allow single-copy transfers, fast MPI-2 one-sided transfers, and SHMEM capability. MPI programs currently cross-map the static region, stack, and a good portion of the private heap. All of these mappings use the MAP_SHARED attribute.

MPI programs cross-map these regions at init time, but none of the memory is actually touched or faulted in until the particular application accesses these mapped pages. The only "resource" the MPI programs are consuming is using up some virtual address space.

A high SIZE and/or SHARE value should not indicate any additional need for swapping, since these pages are not faulted in.

The RSS value should reasonably reflect the application memory usage, except it does not have any way to indicate shared resources.

Memory (Swap) sysctl Parameters

Note: This section applies to SGI ProPack 3 systems only.

The following kernel parameters can be modified at runtime using the sysctl(8) command to affect kernel swap behavior, as follows:

- "vm.min_swap_page_calls" on page 71
- "vm.oom_killer_nap_jiffies" on page 71
- "vm.swap_watch_interval" on page 72
- "vm.min_jiffies_out" on page 72
- "vm.print_get_swap_page" on page 72
- "vm.min_free_swap_pages" on page 72

vm.min_swap_page_calls

Minimum number of swap calls in a swap watch interval before a decision is made to determine if the system is out of swap space or not.

Defaults to 1000

vm.oom_killer_nap_jiffies

How long the oom_killer_thread naps after it is first woken up and before it kills some process; also the time it naps after killing a process and before killing the next process. This is done to make sure that the system does not kill a process unless it has been out of swap space for a while. It also gives the system time to react (and perhaps stop swapping so much) after a processes has been killed and before it decides whether or not to kill another process. (Note that oom_killer() function checks to make sure it is still out of swap space every it time it wakes up from a "nap". If we are not out of swap space, it goes back to long term sleep waiting until `start_oom_killer`() is called again.

Defaults to 10*HZ

vm.swap_watch_interval

How long between resets of the swap out statistics collected by the `get_swap_page`() function.

Defaults to 10*HZ

vm.min_jiffies_out

Minimum time that the `try_to_free_pages_zone`() function has to have consistently failed before the `out_of_memory`() function will start up the `oom_killer`() function.

Defaults to 5*HZ

vm.print_get_swap_page

If set to 1, `vm.print_get_swap_page` parameter will cause the system to log swap out statistics every `swap_watch_interval`, provided that swapping is active. This may be removed in a future release.

Defaults to 0

vm.min_free_swap_pages

If this much swap space is free, the system will decide it is no longer out of swap (out of memory) on the next pass through the `oom_killer`() function. While settable using the `sysctl`(8) command, this is reset after each `swapon`() call to 1% of available swap pages. If some other default is desired, there has to be another `sysctl` call.

Defaults to 1% of total swap pages

sched.child_penalty

The sched. child_penalty parameter controls how much or how little a forking child process inherits of one of the scheduling characteristics of the parent process, that is; its "interactivity" assessment. A forking child typically inherits only a fraction of the parent's "interactivity" assessment in order to avoid a potential denial-of-service attack on the system's CPU resource.

Each process is regularly assessed with a quantitative "interactivity" level and is assigned a value in a numerical continuum that ranges between the extremes of "totally computebound" and "executes for brief periods of time on rare occasions." If a process is deemed to be more and more "interactive," the scheduler gives it more and more of a transitory boost in priority when the process wakes up and wants the CPU resource. That is, a process that appears to be "interactive," such as a shell that responds to user keyboard inputs, is given more timely access to the CPU than a process which appears to be computebound.

This is a very heuristic assessment, and as such it is prone to approximation, confusion, and errors. One of the potential problems is the denial-of-service effect of having an interactive parent (which executes with that priority boost) being able to fork numerous children that would inherit the same high-priority "interactive" label as the parent and would themselves also preempt other lower-priority less-interactive processes.

The remedy for this potential problem is to not allow a forked child to inherit the exact same "interactive" quantitative value as the parent. Instead, a forked child is assessed a child_penalty, which is a percentage of the parent's "interactive" assessment. The default child_penalty is 50, or 50% of the parent's value.

Load-balancing Algorithms sysctl Parameters

Note: This section applies to SGI ProPack 3 systems only.

The kernel parameters that can be modified at runtime using the sysctl(8) command to affect kernel load-balancing algorithms, are as follows:

```
sched.sched_loadbal_max = 1
sched.sched_exec_threshold = 1
sched.max_loadbal_rejects = 100
sched.sched_node_threshold = 125
sched.busy_node_rebalance_ratio = 10
sched.idle_node_rebalance_ratio_max = 50
sched.idle_node_rebalance_ratio = 10
```

These CPU Scheduler parameters affect the behavior of the load-balancing algorithms that attempt to equalize the runqueue lengths. This section describes these parameters.

The `sched_exec_threshold` parameter affects the aggressiveness of load-balancing at process `exec` time. When a parent process forks, the resulting child process is by default assigned to the same CPU as the parent. This is often the optimal behavior, for example, due to shared virtual address spaces. However, if the child process issues an `exec` call itself, it is reasonable to assume that the child process typically gains little advantage in executing on the CPU of the parent process. Instead, the child process should be migrated to a lesser loaded CPU. The `sched_exec_threshold` parameter is the runqueue length above which the child issuing an `exec` call will search for a lesser-loaded CPU. The default value of one (1) means that the child process will always search. This is the most aggressive behavior. Raising the value makes the search less aggressive, which trades off a slight decrease in `exec` overhead for a less load-balanced set of CPUs.

The remaining parameters described in this section control behavior of the load-balancing algorithms that periodically execute on each CPU. During each 1024Hz timer tick, the CPU Scheduler decides whether or not to spend effort to compare its runqueue length against the runqueue lengths of the other CPUs. This is to determine if one or more processes should be pull-migrated from the runqueue of another CPU onto the runqueue of this CPU. This runqueue examination can be an expensive operation in terms of system resources, especially with large numbers of CPUs. Therefore, a CPU must trade off the cost of executing it too frequently versus the inefficiency of executing it too infrequently and having the CPU remain under utilized.

At each timer tick, the first decision made is whether or not to execute the basic load-balancing algorithm at all. The more frequently a CPU performs this load-balancing scan, the more evenly balanced are the runqueues of each and every CPU relative to the other runqueues. However, there are two significant downsides to performing overly frequent load-balancing. The first is that frequent load-balancing is invasive and causes contention on the busiest CPUs' runqueues's spinlocks. High contention levels will affect context-switching performance and may in fact produce so much contention (especially at high CPU counts) that the system "livelocks" on the busiest CPU's runqueue lock. The second downside to frequent load-balancing is that processes may be migrated away from local physical memory and thus may suffer substantially longer memory access latencies. The trade-off is giving a process access to more CPU cycles at the cost of having those CPU cycles be less efficient because of longer latencies. In some cases, a process is much better off remaining on a more busy CPU because the process remains close to the physical memory it can most efficiently access.

An idle CPU is more tempted to perform this relatively costly load-balancing scan than a non-idle (“busy") CPU, since the system would generally benefit (ignoring issues of NUMA memory locality) from migrating a not-currently-executing process from another CPU into this idle CPU. Every 1024Hz tick (roughly every millisecond) an idle CPU performs the load-balance scan within the node, that is , examining only the other CPU in the two-CPU Altix node. Every `idle_node_rebalance_ratio` ticks (current a default value of 10, or roughly every ten milliseconds) an idle CPU performs a load-balance scan of all nodes in the system. Therefore, increasing the `idle_node_rebalance_ratio` value makes the idle CPU full system rebalancing less frequent. Decreasing the value makes it more frequent.

If an idle CPU finds no process to pull-migrate from a busier CPU, then the delay (the "infrequency") of these idle scans is dynamically increased by one, up to a maximum value of `idle_node_rebalance_ratio_max`. Therefore, with a default maximum of 50, an idle CPU does an all-CPU load-balance scan after 10 milliseconds. If no pull-migrate occurs, the next scan occurs 11 milliseconds later, then 12 milliseconds later, and so on, up to a maximum of 50 milliseconds. When one of the scans finds a process to pull-migrate, the delay is reset to the basic `idle_node_rebalance_ratio` value, which defaults to 10. The higher the value of the `idle_node_rebalance_ratio_max` parameter, the longer it will likely be between all-CPU load-balancing scan. This means that when a "busier" CPU does emerge, the slower the other idle CPUs will be to recognize it and to off-load that "busier" CPU. Systems with larger CPU counts may well benefit from higher `idle_node_rebalance_ratio` and `idle_node_rebalance_ratio_max` values. Each individual idle CPU may be slower to see a suddenly overloaded CPU, but because there are likely to be many idle CPUs, then some idle CPU will likely recognize the overloaded CPU and perform load-balancing. Never specify an `idle_node_rebalance_ratio_max` value less than `idle_node_rebalance_ratio`.

A non-idle "busy" CPU performs the same within-the-local-node load-balancing scan at 1/100th the frequency of an idle CPU, or about every 100 milliseconds, and it performs the all-CPUs scan at a multiplier of `busy_node_rebalance_ratio` of that. With a `busy_node_rebalance_ratio` value of 10, this means an all-CPUs scan about once per second. These scan rates are definitely less frequent for a non-idle CPU than for an idle CPU. Once again, the CPU Scheduler is reluctant to migrate processes between nodes and potentially away from low-latency, local physical memory.

Once a CPU decides to perform the load-balancing scan, there are more tuning parameters that can affect behavior. The first is the `sched_node_threshold` parameter, which is the threshold ratio of "imbalance" (relative to 100) that determines whether to begin pulling processes. The default value of 125 defines an imbalance of 25%. An alternative value of 150 would define an imbalance of 50%. You should never use a value less than 100 and should avoid using values less than the default 125.

Once "busier" CPUs are identified that have processes that can be pull-migrated to this less-loaded CPU, the `sched_loadbal_max` is the number of processes that may be pull-migrated. The default value of one means that each load-balance scan will pull-migrate at most one process. Raising this value, increases the number of processes that may be migrated. The higher the value, the greater the likelihood that the load-balancing migrations may become overly aggressive. However, this is a heuristic algorithm, and some workloads might benefit from a value greater than the default of one.

Consider a system with 128 CPUs, when the runqueue workload of one of the CPUs suddenly skyrockets to 128 processes, while the other 127 CPUs are idle. With a `sched_loadbal_max` value of one, as each idle CPU executes the load-balancing scan at whatever frequency that scan occurs (as noted earlier, depending upon both fixed constants and by tuning parameters), each will pick off one process at a time from this overloaded runqueue until each CPU has a runqueue of one.

If, however, the `sched_loadbal_max` parameter is a high value, the first idle CPU to execute the load-balance algorithm would pull half of the processes of the busy CPUs; the 128–CPU system will have two CPUs with equal runqueue lengths of 64-64. The next CPU to execute the load-balance algorithm would pull 32 processes from one of these busy CPUs to equalize that one bilateral imbalance, producing runqueue lengths of 32-64-32, and then would pull 16 processes from the second CPU to equalize that bilateral imbalance, thus producing final runqueue lengths of 32-48-48. Note that most of the migrated processes will not have actually executed, but will merely have moved from waiting on one runqueue to waiting in a second runqueue. A third idle CPU does its load-balancing and produces another readjustment of 32-24-36-36. Once again, processes are migrated around in groups, from CPU to CPU, likely before actually executing. Thus, a higher `sched_loadbal_max` value may result in more active migrations and certainly in a different pattern of change of the runqueue lengths, but it is unclear whether higher values are more, or less, effective than lower value.

Finally, the `max_loadbal_rejects` parameter puts a limit the number of pull-migration candidate processes a load-balancing CPU will examine; and reject as being unsuitable, before releasing runqueue spinlocks and giving up the search. A candidate is rejected if it has not been in the runqueue of the busy CPU for a long enough time and thus is deemed as being "cache hot"; or is rejected because the candidate process's cpus_allowed mask (as set by CpuMemSets or by the sys_sched_setaffinity() system call) does not allow that process to execute on the load-balancing CPU. The higher the max_loadbal_rejects value, the more effort the searching CPU makes to find a process to pull-migrate. Once again, this is another trade-off; the cost of acquiring and holding valuable runqueue spinlocks for potentially longer and longer periods of time, versus the benefit of successfully load-balancing the CPU runqueues.

Virtual Memory hugetlb Parameter

The /usr/src/linux/Documentation/vm/hugetlbpage.txt file on your system provides a brief summary of hugetlb page support in the Linux kernel. The Intel Itanium architecture supports multiple page sizes 4K, 8K, 64K, 256K, 1M, 4M, 16M, 256M and so on. A translation lookaside buffer (TLB) is a cache of virtual-to-physical translations. Typically, this is a very scarce resource on a processor. Operating systems try to make best use of limited number of TLB resources. This optimization is more critical now as larger physical memories (several GBs) are more available.

You can use the huge page support in Linux kernel by either using the mmap(2) system call or standard shared memory system calls shmget(2) and shmat(2) (see the shmop(2) man page).

For information on using hugetlb, see /usr/src/linux/Documentation/vm/hugetlbpage.txt.

You can also boot your system with the kernel hugepages= *X* parameter set, where *X* is a number of pages. Setting the hugepages parameter allows the kernel to allocate as many of the huge pages as possible early on. After the system is booted, it gets harder to find contiguous memory. You can also set huge pages early in the init scripts in /proc/sys/vm/nr_hugepages as described in the /usr/src/linux/Documentation/vm/hugetlbpage.txt file.

Some considerations on using the hugetlb parameter on your Altix system are, as follows:

- Starting with the SGI ProPack 3 for Linux Service Pack 1 release, allocation of hugetlb pages is NUMA aware. That is, pages are allocated either on the node, or as close as possible to the node where the mmap or shmget(2) system call is executed. The hugetlb pages are allocated and mapped into the requesting address space at mmap() or shmat() time; the hugetlb pages are not demand faulted into the address space as are regular pages. The hugetlb pages are allocated and zeroed by the thread that calls mmap() or shmget().

- Starting with the SGI Propack 3, Service Pack 1 release, hugetlb pages are allocated on first touch, much like how regular pages are allocated into an address space by a call to the mmap routine. The hugetlb page allocation is NUMA aware; this means that the huge page is allocated on (or as close as possible to) the node where the thread that first touched the page is executing. In previous SGI ProPack releases, tasks that allocated more hugetlb pages than were available at the time of the mmap() or shmget() call, caused the mmap() or shmget() call to fail. To keep this behavior the same in the ProPack 3, Service Pack 1 release, the system implements a "reservation" algorithm to ensure that once the mmap() or shmget() call has completed successfully, the system guarantees that there are sufficient pages at first-touch time to satisfy the (delayge) storage allocation request that occurs at first-touch time. Page reservation is not subject to NUMA allocation; that is, you cannot reserve pages on a particular node, nor does the node on which the mmap() or shmget() request executes have any influence on where the hugetlb pages are finally placed. Placement of hugetlb pages is determined by the node where the first touch occurs and the locations of the available hugetlb pages available at that time. The number of hugetlb pages currently reserved can be found in the /proc/meminfo file, as shown in the following example:

```
[root@revenue3 proc]# cat meminfo
        total:    used:    free:  shared: buffers:  cached:
Mem:  32301940736 4025974784 28275965952        0    98304 2333163520
Swap: 10737319936  9928704 10727391232
MemTotal:     31544864 kB
MemFree:      27613248 kB
MemShared:           0 kB
Buffers:            96 kB
Cached:        2271280 kB
SwapCached:       7200 kB
Active:         459360 kB
Inactive:      1845840 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:     31544864 kB
LowFree:      27613248 kB
SwapTotal:    10485664 kB
SwapFree:     10475968 kB
```

```
HugePages_Total:     0
HugePages_Free:      0
Hugepagesize:    262144 kB
```

- You can change the hugetlb page size can at system boot time by specifying the hugepagesz=NNNN parameter to the ELILO boot command prompt (this parameter can also be specified using the append command in the/boot/efi/efi/sgi/elilo.conf file). The NNNN parameter is the size of the hugetlb pages in bytes and this parameter must have a value that is a supported hugetlb page size for the hardware platform where the kernel is booted.

Index