

SGI ProPack™ v 2.1 for Linux® Start Here

007-4558-001

CONTRIBUTORS

Written by Julie Boney

Edited by Susan Wilkening

Graphics by Chrystie Danzer

Production by Glen Traefald

COPYRIGHT

© 2003 Silicon Graphics, Inc. All rights reserved; provided portions may be copyright in third parties, as indicated elsewhere herein. No permission is granted to copy, distribute, or create derivative works from the contents of this electronic documentation in any manner, in whole or in part, without the prior written permission of Silicon Graphics, Inc.

LIMITED RIGHTS LEGEND

The electronic (software) version of this document was developed at private expense; if acquired under an agreement with the USA government or any contractor thereto, it is acquired as "commercial computer software" subject to the provisions of its applicable license agreement, as specified in (a) 48 CFR 12.212 of the FAR; or, if acquired for Department of Defense units, (b) 48 CFR 227-7202 of the DoD FAR Supplement; or sections succeeding thereto. Contractor/manufacturer is Silicon Graphics, Inc., 1600 Amphitheatre Pkwy 2E, Mountain View, CA 94043-1351.

TRADEMARKS AND ATTRIBUTIONS

Silicon Graphics, SGI, the SGI logo, IRIX, Origin, and Onyx are registered trademarks and Altix, NUMAflex, NUMAlink, OpenMP, Performance Co-Pilot, SGI Linux, SGI ProPack, SGIconsole, and XFS are trademarks of Silicon Graphics, Inc., in the U.S. and/or other countries worldwide.

SGI Linux Environment 7.2 is based on Red Hat Linux 7.2, but is not sponsored by or endorsed by Red Hat, Inc. in any way.

Cray is a registered trademark of Cray, Inc. FLEXlm and GLOBEtrout are registered trademarks of GLOBEtrout Software and Macrovision Corporation. Java is a registered trademark of Sun Microsystems, Inc. in the United States and/or other countries. KAP/Pro Toolset and VTune are trademarks and Intel, Itanium, and Pentium are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. Linux is a registered trademark of Linus Torvalds, used with permission by Silicon Graphics, Inc. MIPS is a registered trademark of MIPS Technology, Inc. PostScript is a registered trademark of Adobe Systems, Inc. QLogic is a registered trademark of QLogic Corporation. Red Hat is a registered trademark and Red Hat Linux 7.2 is a trademark of Red Hat, Inc. Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries.

Cover Design By Sarah Bolles, Sarah Bolles Design, and Dany Galgani, SGI Technical Publications.

Record of Revision

Version	Description
001	February 2003 Original publication

Contents

Record of Revision	iii
Figures	ix
Tables	xi
Related Publications	xiv
Intel Compiler Documentation	xiv
Other Intel Documentation	xiv
SGI Documentation	xiv
Obtaining SGI Publications	xvi
Conventions	xvi
Reader Comments	xvii
1. Release Features	1
Software Introduction	1
CD Contents	2
Installation Overview	4
2. Software Planning and Installation	5
Software Planning	5
Table of Disk Partitions	6
File Configurations	6
Installing the SGI Linux Environment 7.2 Base OS	6
Installing SGI ProPack for Linux	9
Installing the SGI Linux Environment 7.2 Updates	10
Installing the System Controller Software 1.1 CD	11
Upgrading Your Software	11
Recovering a Damaged Root Filesystem	12

3.	Product Support 15
	Operating System Enhancements. 19
	CpuMemSets Support. 19
	Cpuset Support 20
	Comprehensive System Accounting (CSA). 20
	Partitioning. 21
	I/O Subsystems. 21
	Persistent PCI-X Bus Numbering 22
	Using the <code>ioconfig</code> Command 23
	Using the <code>ioconfig=</code> Kernel Boot Option 24
	Persistent Naming of Ethernet Devices 24
	XSCSI Subsystem 25
	XSCSI Device Naming 25
	XFS Filesystem 26
	XVM Volume Manager 26
	HPC Application Tools and Support. 26
	Message Passing Toolkit 27
	Performance Co-Pilot (PCP) 27
	PROM Chips 27
	Extensible Firmware Interface (EFI) 28
	FLEXlm 30
	SGIconsole 30
	System Controller Firmware 31
	NUMA Tools 31
	<code>dlook</code> Command 31
	<code>dplace</code> Command 32
4.	Performance Tuning 33
	Determining System Configuration 33
	Single Processor Code Tuning 34
	Getting the Correct Results 35
	Using <code>ddd</code> 36
	Managing Heap Corruption Problems 37

Using Tuned Code	38
Determining Tuning Needs	39
Using Compiler Options Where Possible	39
Tuning the Cache Performance	40
Managing Memory	41
Using <code>chatr</code> to Change Stack and Data Segments	41
Multiprocessor Code Tuning.	42
Parallelizing Your Code	42
Solving Bottlenecks in Code.	44
Fixing False Sharing	44
Using <code>dplace</code> and <code>runon</code>	45
<code>dplace</code> for MPI Codes	45
<code>dplace</code> for OpenMP Codes	46
Environment Variables for Performance Tuning	46
Profilers and Performance Tools.	47
Profiling with <code>pfmon</code>	47
The <code>profile.pl</code> script	48
<code>profile.pl</code> with MPI programs	48
Using VTune for Remote Sampling.	49
Using GuideView.	49
Other Performance Tools	50
Index	53

Figures

Figure 1-1	SGI ProPack for Linux Release CD Contents	3
-------------------	---	---

Tables

Table 2-1	devfs Disk Partitions	6
Table 3-1	SGI ProPack v2.1 for Linux Products	16
Table 3-2	EFI Commands	28

About This Guide

This guide provides information about the SGI ProPack for Linux release. It is divided into the following chapters:

- Chapter 1, "Release Features," describes the major features of this release, the CD contents, and documentation information.
- Chapter 2, "Software Planning and Installation," describes the flowcharts of disk partitions and file configurations and layouts that you need to get started, and provides instructions for installing SGI ProPack for Linux.
- Chapter 3, "Product Support," documents the product components that are supported on SGI Altix 3000 systems.
- Chapter 4, "Performance Tuning," documents tuning issues for single processor and multiprocessor programs.

The information in this guide, other 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 v2.1 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.

Note: The release notes, which contain the latest information about software and documentation in this release, are on the SGI ProPack for Linux Documentation CD in the root directory, in a file named `README.SGI`.

Related Publications

Documents listed in this section contain additional information that might be helpful.

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. 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_prd.pdf`).
- *Intel Fortran Libraries Reference* (`for_lib.pdf`).

Other Intel Documentation

The following references 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 <http://developer.intel.com/design/itanium/manuals>.
- *Intel Itanium Architecture Software Developer's Manual*, available online at <http://developer.intel.com/design/itanium/manuals>.
- *Introduction to Itanium Architecture*, available online at <http://shale.intel.com/softwarecollege/CourseDetails.asp?courseID=13>.

SGI Documentation

The following SGI documentation is available:

- *Linux Device Driver Programmer's Guide*
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.
- *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.
- *Performance Co-pilot for Linux User's and Administrator's Guide*
Describes the Performance Co-Pilot (PCP) software package of advanced performance tools for SGI systems running the Linux operating system.
- *Resource Administration Guide for Linux*
Provides a reference for people who manage the operation of SGI computer systems running the Linux operating system.
- *SGI Altix 3000 User's Guide*
Provides an overview of the architecture and describes the major components of the SGI Altix 3000 computer system. It also describes the standard procedures for powering up and powering down the system, provides basic troubleshooting information, and includes important safety and regulatory specifications.
- *SGI ProPack v2.1 for Linux Release Notes*
Provide 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.SGI`.
- *SGIconsole 1.1 Start Here*
Provides introduction to SGIconsole.
- *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 the SGI Altix 3000 series systems.
- *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.

Obtaining SGI 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.
- If it is installed on your SGI system, you can use InfoSearch, an online tool that provides a more limited set of online books, release notes, and man pages. With an IRIX system, select **Help** from the Toolchest, and then select **InfoSearch**. Or you can type `infosearch` on a command line.
- You can also view man pages by typing `man <title>` on a command line.

Conventions

The following conventions are used throughout this publication:

Convention	Meaning
<code>command</code>	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
<i>variable</i>	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.)
<code>manpage(x)</code>	Man page section identifiers appear in parentheses after man page names.
GUI element	This font denotes the names of graphical user interface (GUI) elements such as windows, screens, dialog boxes, menus, toolbars, icons, buttons, boxes, fields, and lists.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, contact SGI. Be sure to include the title and document number of the manual with your comments. (Online, the document number is located in the front matter of the manual. In printed manuals, 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
1600 Amphitheatre Pkwy, M/S 535
Mountain View, California 94043-1351
- Send a fax to the attention of “Technical Publications” at +1 650 932 0801.

SGI values your comments and will respond to them promptly.

Release Features

This chapter provides an introduction to SGI ProPack for Linux and describes the CD contents, provides documentation information, and provides a brief overview of the installation process.

Software Introduction

The open-source and collaborative development environment of the Linux community provides a new and different model for computer manufacturers to deliver an operating system for a large server or supercomputer. Using Linux as the operating system for very large computer systems has the benefit of providing better software protection and better integration between a system manufacturer's operating system and HPC applications and codes for both independent software vendors and end-users. Also, the collaboration on OS improvements in the Linux community as well as among the various computer manufacturers has enabled Linux to evolve and improve in multiple dimensions faster than would otherwise be possible for a single company working on its own operating system.

The SGI ProPack for Linux product includes capabilities and performance improvements ideal for enabling technical and creative users to solve their big compute and data problems by using the Linux operating system and Itanium processors. This product adds to or enhances features in base Linux distributions based on Red Hat, version 7.2. SGI ProPack for Linux is designed to run on any SGI Altix 3000 series system. SGI hardware platforms and OS configuration settings supported by SGI in this release are documented at the following URL:

<http://support.sgi.com/linux>

CD Contents

The CDs that you receive with SGI ProPack for Linux are as follows (Figure 1-1 lists the contents of each CD):

- SGI Linux Environment 7.2 Installation CD set (2 CDs)
- SGI Linux Environment 7.2 Updates CD
- SGI Linux Environment 7.2 Source Code CD set (3 CDs)
- SGI ProPack v2.1 for Linux Boot CD
- SGI ProPack v2.1 for Linux Open/Free Source Software CD
- SGI ProPack v2.1 for Linux Proprietary Software CD

Note: This CD contains software for use on SGI systems only. Redistribution is not permitted. Please see the shrink-wrap license agreement.

- SGI ProPack v2.1 for Linux Documentation CD
- System Controller Software 1.1 CD (for IRIX and Linux)

Note: This software is licensed for use only on Origin and Onyx 3000 series systems (SGI systems based on MIPS processors) and on SGI Altix 3000 series systems (SGI systems based on Intel Itanium processors). This CD contains software provided under restricted or SGI proprietary licensing terms. The licensing terms for items on this CD allow users to install this software only on SGI systems.

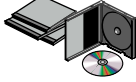
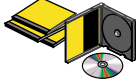
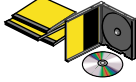
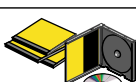
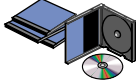
* SGI Linux Environment 7.2		
	SGI Linux Environment 7.2 Installation CD 1 of 2 CD1	Interactive installation procedures
	SGI Linux Environment 7.2 Installation CD 2 of 2 CD2	
	SGI Linux Environment 7.2 Updates CD3	Updates based on Red Hat 7.2 software
	SGI Linux Environment 7.2 Source Code CD 1 of 3 CD4	Source code for base OS and common open source applications
	SGI Linux Environment 7.2 Source Code CD 2 of 3 CD5	
	SGI Linux Environment 7.2 Source Code CD 3 of 3 CD5	
* SGI ProPack v2.1 for Linux		
	SGI ProPack v2.1 for Linux Boot CD CD1	Device drivers and boot kernels
	SGI ProPack v2.1 for Linux Open/Free Source Software CD2	GPL/LGPL and other open source licensed software, latest SGI platform and NUMA support, various kernel performance improvements, CpuMemSets and performance measuring tools, XSCSI infrastructure, QL-SCSI, XFS, CSA, LKCD, kdb
	SGI ProPack v2.1 for Linux Proprietary Software CD3	System PROM, XVM, cpuset and NUMAtools (dlock, dplace, and so on), Array Services, FLEXIm, system partitioning software, MPT
	SGI ProPack v2.1 for Linux Documentation CD CD4	Collection of documentation, FAQs, HOWTOs, and man pages
* System Controller Software 1.1 CD		
	System Controller Software 1.1 CD	System Controller software/firmware (for IRIX and Linux)

Figure 1-1 SGI ProPack for Linux Release CD Contents

For a complete list of the RPMs included on the SGI Linux Environment 7.2 and SGI ProPack v2.1 for Linux CDs listed in Figure 1-1 on page 3, see the release notes, which contain 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.SGI`.

SGI maintains the following website for open source information that describes projects related to its open source efforts:

<http://oss.sgi.com>

You can also access open source information (such as LKCD) from the following website:

<http://sourceforge.net>

Installation Overview

SGI ProPack for Linux will most likely come preinstalled on your SGI platform. If you should need to install it, be aware that you must first install the SGI ProPack v2.1 Boot CD (CD1), then SGI Linux Environment 7.2 Installation CDs (CD1 and CD2), then SGI Linux Environment 7.2 Updates (CD3), and finally the SGI ProPack v2.1 Open/Free Source Software (CD2), and the Proprietary Software (CD3). Installation is described in Chapter 2, “Software Planning and Installation”.

Software Planning and Installation

This chapter describes the planning that you need to do to get started and provides instructions for installing SGI ProPack for Linux.

Your SGI Altix 3000 system comes with a base Linux distribution (SGI Linux Environment 7.2) and the SGI ProPack software preinstalled. This chapter describes how to install the software from the CD if it should ever become necessary to reinstall it.

For security reasons, Linux requires a root password for login. The default password for your preinstalled software is `sgisgi`. After you have logged in, change this root password to a string of your own choice.

The SGI ProPack software works only with the SGI Linux Environment 7.2. Earlier versions of these distributions, or any other distributions, are not compatible with SGI ProPack software.

Before you install or configure your system, please read Chapter 1, “Release Features” so that you understand the features of SGI ProPack for Linux software and how to configure them. You might also want to familiarize yourself with some or all of the documentation listed in “Related Publications” on page xiv.

Software Planning

This section provides disk and file information you need to have about the software that has been shipped to you.

Table of Disk Partitions

Table 2-1 provides mount point and size information about the `devfs` disk partitions.

Table 2-1 `devfs` Disk Partitions

Device	Mount Point	Size
<code>/dev/xscsi/pci01.03.0-1/target1/lun0/part1</code>	<code>/boot/efi</code>	512 MB
<code>/dev/xscsi/pci01.03.0-1/target1/lun0/part2</code>	<code>/</code>	6 GB
<code>/dev/xscsi/pci01.03.0-1/target1/lun0/part3</code>	<code>swap</code>	2 GB
<code>/dev/xscsi/pci01.03.0-1/target1/lun0/part5</code>	<code>/home</code>	8.5 GB

For more information on device naming, see “Persistent PCI-X Bus Numbering” on page 22 and “XSCSI Device Naming” on page 25.

File Configurations

Linux Kernel Crash Dump (LKCD) creates files in `/var/dump`. The `/var` file will grow as Performance Co-Pilot (PCP) and Comprehensive System Accounting (CSA) log to their own directories in `/var`. These files do not usually exceed 100 MB. For default LKCD configuration details, see the `lkcd_config(1)` man page.

Installing the SGI Linux Environment 7.2 Base OS

This section provides steps for installing the SGI ProPack v2.1 for Linux Boot CD (CD1) and the SGI Linux Environment 7.2 CD set.

Note: The installation procedure has buttons that allow you to go back to the previous screen or to quit the installation. To use these buttons, press the `Tab` key to highlight the one you want to use and press `Enter`.

1. Insert the Installation Boot CD (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, Siggl)
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, Siggl0)
```

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 typing **elilo** and pressing Enter. When the ELILO boot prompt appears, press Enter again.

```
fs0:\> elilo This is the boot loader
LoadPe: using PE image entry point
LoadPe: using PE image entry point
ELILO boot: Press ENTER to boot the Linux kernel.
```

5. The CD will then boot and start the installation process.
6. Select the language you would prefer to use for the installation and as the system default. To continue, select **OK**.

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

7. The next screen lets you choose your installation method. Currently, **Local CDROM** is the only choice. To continue, select **OK**.
8. The next screen is the SGI ProPack information screen. To continue, select **OK**.

9. The next screen asks you if you have a mouse installed. For an SGI Altix 3000 system, select **None**. To continue, select **OK**.
10. When asked for the installation type, select **Custom** from the list. To continue, select **OK**.
11. In the partition screen, select the **Autopartition** option and select **OK**. When warning messages appear, select **IGNORE**.
12. When asked which partitioning scheme to use, select the **Remove all partitions on this system** option. Boot devices are selected by default. To continue, select **OK**.
13. When the warning message appears about losing all data on the devices, select **Yes** to confirm the partition layout.
14. The next screen allows you to modify the partition layout as needed for your requirements. If you change the partition layout, ensure that the `/boot/efi` filesystem partition type remains set to `vfat`, the swap partition is set to `swap`, and all other file system types are set to `xfs`. To complete the partition layout, select **OK**.
15. If you use Bootp or DHCP to configure your network interfaces, select **OK**. If you use static IP addresses, fill in the required information and then to continue, select **OK**.

Note: It is important to keep the default IO9 Ethernet interface as `eth0`. If you add other Ethernet interfaces, the interface on the IO9 Ethernet must be the first (default) interface to come up.

16. If you chose to use a static IP address on the previous screen, you are prompted for a hostname for the system. Enter the hostname and to continue, select **OK**.
17. On the next screen you will be asked if you wish to configure a firewall on this system. Select the appropriate option based on your system's need and to continue, select **OK**.
18. The next screen allows you to select additional languages to support on the system.

Note: If you select additional languages, you must select a default language. The default language will be used on your system once installation is complete. If you choose to install other languages, you can change your default language after installation. Console support for Asian languages is not available at this time. SGI recommends that customers in Asia set the default system language to English.

19. The next screen allows you to select which time zone this system is in. Select the appropriate information and to continue, select **OK**.
20. The next screen asks you for the root password for this system. Enter the password you would like to use for root access and to continue, select **OK**.
21. The next screen allows you to create a user for the system. Creating a user allows you to log in to the system as a user other than `root`. If you choose to create a user, fill in the required boxes and to continue, select **OK**. Leaving the boxes blank and selecting **OK** continues without creating another user.
22. If you created a user in the previous screen, the next screen allows you to add additional users. Add other users as necessary. To continue, select **OK**.
23. The next screen lets you choose your authentication configuration. To continue, select **OK**.
24. The **Package Group Selection** screen appears. All packages should be selected for installation. Select **Everything** (located at the bottom of the list). To continue, press the `Tab` key to highlight **OK** and press `ENTER`.
25. The **Installation to begin** screen is informational. To continue, select **OK**.
26. The next several screens are package installation screens, instructing you to insert the remaining CDs, as needed. For each one, to continue, select **OK**.
27. When the installation has finished, select **OK**. The system will reboot. You can remove the CD from the drive.

Installing SGI ProPack for Linux

This section provides steps for installing the SGI ProPack v2.1 for Linux Open/Free Source Software (CD2) and the SGI ProPack v2.1 for Linux Proprietary Software (CD3).

1. Log in to the system as `root`, using the root password you selected in the previous installation process.
2. Insert the SGI ProPack Open/Free Source Software CD (CD2) into the system and use the following command to mount the CD:

```
mount /dev/cdrom /mnt/cdrom
```
3. Enter `/path/INSTALL`
4. The SGI Welcome screen for SGI ProPack will appear. To continue, select **OK**.

5. The next screen lets you choose your installation type. Select **Yes** and select **OK**.
6. The **Package Group Selection** screen appears. This screen allows you to select the type of package group you want to install. You select a package group by using the up and down arrow keys and pressing the space bar to select the one you want. When you select a package group, RPMs for that package group will be installed after you press the Tab key to highlight **OK** and press Enter. All packages are selected by default. If you change the selection, ensure that **SGI Proprietary** remains selected. To continue, select **OK**.
7. The **Installation to Begin** screen appears. It tells you that a log of the installation will be placed in `/tmp/sgi-install.log`. Press the Tab key to highlight **OK** and press Enter.
8. The installation begins. You will see the **Package Installation** screen, which tells you which packages are being installed and logs the time it takes to install them.
9. After the installation is complete, the **Complete** screen appears. Press Enter. You are returned to the root prompt.
10. After the software installation is complete, reboot your system to begin using the newly installed SGI ProPack for Linux software kernel. You can reboot by typing `reboot` and pressing ENTER. You can then remove the CD from the drive.

Installing the SGI Linux Environment 7.2 Updates

This section provides steps for installing the CD that contains SGI Linux Environment 7.2 updates (CD3).

1. Log in to the system as `root`, using the root password you selected in the initial installation process.
2. Insert the SGI Linux Environment 7.2 Updates CD into the system and use the following command to mount the CD:

```
mount /dev/cdrom /mnt/cdrom
```
3. Enter `/path/INSTALL.`
4. The Welcome screen appears. To continue, select **OK**.
5. The next screen lets you choose your installation type. "Upgrade Existing System" is the default. To continue, select **OK**.
6. The next screen allows you to begin the upgrade. To begin the upgrade, select **OK**.

7. The next screen displays the progress of the installation.
8. The next screen is informational. Your SGI Linux Environment 7.2 update is complete. Reboot your system by typing `reboot` and pressing `ENTER`. You can then remove CD from the drive.

Installing the System Controller Software 1.1 CD

This section provides steps for installing the System Controller Software 1.1 (for IRIX and Linux). This software is licensed for SGI Origin and Onyx 3000 series systems based on MIPS processors and for SGI Altix 3000 systems.

1. Log in to the system as `root`, using the root password you used in the previous installation process.
2. Insert the System Controller Software 1.1 CD into the system and use the following command to mount the CD:

```
mount /dev/cdrom /mnt/cdrom
```
3. Change directories to `/mnt/cdrom/RPMS/ia64` and enter the following command:

```
./install
```
4. The installation begins. You will see which packages are being installed. After the installation is complete, you are returned to the `root` prompt.
5. After the software installation is complete, remove the System Controller Software 1.1 CD.

Upgrading Your Software

When you are ready to upgrade your SGI ProPack software, follow the instructions outlined in the section titled “Installing SGI ProPack for Linux” on page 9, using the newer SGI ProPack CD set.

To reinstall all of the software on your system, follow the entire set of installation instructions listed previously, using the latest SGI ProPack CD set.

Recovering a Damaged Root Filesystem

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

1. Insert the Installation Boot CD (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 typing **elilo** and pressing Enter. When the ELILO boot prompt appears, press Enter again.

```
fs0:\> elilo This is the boot loader
LoadPe: using PE image entry point
LoadPe: using PE image entry point
ELILO boot: Press ENTER to boot the Linux kernel.
```

5. The CD will then boot and start the installation process.
6. Select the language you would prefer to use for the installation and as the system default. To continue, select **OK**.

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

7. The next screen lets you choose your installation method. Currently, **Local CDROM** is the only choice. To continue, select **OK**.
8. On the next screen, after seeing a message indicating that Anaconda is starting, you can press `Ctrl-z` at any time to enter the rescue shell. Common system commands are located in `/usr/sbin/`. This directory will be included in your path.

At this point, administrators can inspect and attempt to repair or recover any damaged or missing software.

To repair your XFS root filesystem, use the `xfstool` command. For more information, see *XFS for Linux Administration* and the `xfstool(1M)` man page.

Product Support

This chapter documents the product components that are supported on SGI Altix 3000 systems. (For a list of the products, see Table 3-1 on page 16.)

Descriptions of the product components are grouped in this chapter as follows:

- Operating system enhancements
- I/O subsystems
- HPC application tools and support
- NUMA tools

Software provided by SGI for the SGI ProPack for Linux release consists of a kernel RPM for the SGI Altix 3000 product, SGI Linux Environment 7.2 RPMs, and value-added software developed by SGI to run specifically on SGI systems. Table 3-1 provides a description of the SGI ProPack for Linux products

Table 3-1 SGI ProPack v2.1 for Linux Products

Product	Description
Application performance measuring tools	<p>The following tools perform program optimization:</p> <p>VTune - This tool, developed and supported by Intel, uses the performance measurement facilities of the Itanium processor to take profiles based on elapsed time or other architected events within the processor. These profiles can be used to measure, tune, and improve application performance.</p> <p>pfmon - This tool, available as open source and licensed under the GPL, provides a command line interface to control the performance measurement facilities of the Itanium processor. Data generated by this tool can be post-processed to produce a variety of reports describing application performance. These reports can be used to measure, tune, and improve application performance. The pfmon package also includes a library interface, <code>libpfmon.a</code>, that can be used to create customized performance measurement tools.</p>
Array Services	Provides a set of tools with kernel support that simplify the management of systems and parallel applications for clusters of SGI systems.
CpuMemSets	Provides the kernel support and infrastructure for implementing processor and memory placement.
cpuset	Provides for naming a set of CPUs.
CSA	Provides jobs-based accounting of per-task resources and disk usage for specific login accounts on Linux systems.
FLEXlm	Provides a floating license, run-time environment. Includes daemons suitable for serving floating licenses.

Table 3-1 SGI ProPack v2.1 for Linux Products **(continued)**

Product	Description
IOC4 driver	Driver that supports the Internal IDE CD-ROM, NVRAM, and Real-Time Clock.
kdb	Supports kernel debugging of the running system, either directly from the keyboard or over a serial console.
Kernel partitioning support	Provides the software infrastructure necessary to support a partitioned system, including cross-partition communication support.
Kernel performance improvements	Includes community-based patches such as the O(1) scheduler back-ported from the 2.5.x kernel and included in the SGI kernel, several patches that reduce lock contention on the Big Kernel Lock (BKL), and FRlocks (Fast-Reader Locks), which reduce contention on the <code>xtime_lock</code> (used by <code>gettimeofday</code>).
L1 and L2 System Controller firmware	Provides support for managing and monitoring the power, cooling, and testing functions for a brick and system compute rack.
LKCD	Provides system crash dump analysis tools including <code>lcrash</code> and all user level scripts required for saving and configuring system crash dumps.
MPT	Provides industry-standard message passing libraries optimized for SGI computers.
NUMA tools	Provides a collection of NUMA related tools (<code>dlook</code> , <code>dplace</code> , and so on).
Performance Co-Pilot collector infrastructure	Provides performance monitoring and performance management services targeted at large, complex systems.
PROM	Allows the CPU to boot and allows you to perform system administration and software installations.
runon	Enables running a command on a particular CPU or set of CPUs.

Table 3-1 SGI ProPack v2.1 for Linux Products **(continued)**

Product	Description
SGI Linux Environment 7.2 updates	Provides a set of updated RPMs for the SGI Linux Environment 7.2 base shipped with SGI ProPack.
XFS	Provides a high-performance filesystem for Linux.
XSCSI infrastructure	Provides a disk and ATAPI CD-ROM driver with a QLogic SCSI Host Bus Adapter driver for the QLA12160. Also included is a binary HBA driver for QLogic Fibre Channel QLA2342. The infrastructure includes support for robust error handling, failover, and SAN. It can be configured with or without the Linux SCSI layers. If configured without the Linux SCSI layer, XSCSI will make the familiar disk device names (<code>/dev/sda</code> , etc).
XVM	Provides software volume manager functionality such as disk striping and mirroring.

SGI does not support the following:

- Software obtained from other places (that is, not released by SGI).
- Other releases, updates, or patches from Red Hat.
- Software patches, drivers, or other changes obtained from the Linux community or other vendors.
- Kernels recompiled or reconfigured to run with parameter settings or other modules as not specified by SGI. In particular, the following kernel components are not supported due to quality, functionality, or performance issues: ext3, ReiserFS, LVM, and md. You should use XFS and XVM instead.
- Unsupported hardware configurations and devices.

Operating System Enhancements

Building on the Linux operating system's rapid expansion and improvements for general commercial and enterprise environments, SGI has focused on improving Linux capabilities and performance specifically for high performance computing's (HPC's) big compute and big data environments. Thus, SGI has leveraged its experience with NUMAflex and HPC from its IRIX operating systems and MIPS processor-based systems and concentrated on the Linux kernel improvements specifically important to HPC environments.

CpuMemSets Support

CpuMemSets provides a Linux kernel facility that enables system services and applications to specify on which CPUs they can be scheduled, and from which nodes they can allocate memory. The SGI ProPack kernel and library installation automatically include support for CpuMemSets.

The default configuration makes all CPUs and all system memory available to all applications. You can use the CpuMemSets facility to restrict any process, process family, or process virtual memory region to a specified subset of the system CPUs and memory.

The `runon` command relies on CpuMemSets to enable a user to run a specified command on a specified list of CPUs. Both a C shared library and Python language module are provided to access the CpuMemSets system interface.

In future releases of SGI ProPack for Linux, SGI anticipates providing additional facilities that provide other system services that provide convenient access to the CpuMemSets facility.

For further documentation and details on CpuMemSets support, see the chapter titled "CPU Memory Sets and Scheduling" in the *Resource Administration Guide for Linux* or browse the files that are installed as part of the CpuMemSets RPM, as listed by the following command:

```
rpm -ql CpuMemSets
```

In particular, see the man pages for `cpumemsets` and `runon`.

CpuMemSets is an SGI open source project, also available from the following location:

```
http://oss.sgi.com/projects/cpumemsets
```

Cpuset Support

The Cpuset System is primarily a workload manager tool that permits a system administrator to restrict the number of processors that a process or set of processes can use.

A cpuset is a named set of CPUs, which can be defined to be restricted or open. A restricted cpuset allows only processes that are members of the cpuset to run on the set of CPUs. An open cpuset allows any process to run on its CPUs, but a process that is a member of the cpuset can run only on the CPUs belonging to the cpuset. A cpuset is defined by a cpuset configuration file and a name.

A system administrator can use cpusets to create a division of CPUs within a larger system. Such a divided system allows a set of processes to be contained to specific CPUs, reducing the amount of interaction and contention those processes have with other work on the system. In the case of a restricted cpuset, the processes that are attached to that cpuset will not be affected by other work on the system; only those processes attached to the cpuset can be scheduled to run on the CPUs assigned to the cpuset. An open cpuset can be used to restrict processes to a set of CPUs so that the effect these processes have on the rest of the system is minimized.

For further documentation and details on cpuset support, see the chapter titled “Cpuset System” in the *Resource Administration Guide for Linux*.

Comprehensive System Accounting (CSA)

The port of Comprehensive System Accounting (CSA) software packages from IRIX to Linux is the result of an open source collaboration between SGI and Los Alamos National Laboratory (LANL) to provide jobs-based accounting of per-task resources and disk usage for specific login accounts on Linux systems.

Providing extensive system accounting capabilities is often important for very large systems, especially when the system will be shared or made available for other organizations to use. CSA uses a Job Containers feature, which provides on Linux the notion of a “job.” A job is an inescapable container and a collection of processes that enables CSA to track resources for any point of entry to a machine (for example, interactive login, cron job, remote login, batched workload, and so on).

CSA on Linux is an SGI open source project, also available from the following location:

<http://oss.sgi.com/projects/csa>

For further documentation and details on CSA support, see the chapter titled “Comprehensive System Accounting” in the *Resource Administration Guide for Linux*.

Partitioning

With the SGI Altix 3000 system, SGI provides the ability to divide a larger system into smaller system "partitions," where each partition runs its own copy of the OS kernel. As the demand and need for larger systems continues to increase for solving complex HPC problems, eventually the OS's ability to manage a large pool of CPUs, memory, and I/O resources can limit performance and become a bottleneck for some workloads. Partitioning is an effective way to overcome this problem, especially for parallel workloads such as Message Passing Interface (MPI) jobs. Because each partition is still connected (using the SGI high-performance NUMALink interconnect), partitioning does not sacrifice the low-latency and high-bandwidth benefits for communication between processes when used to scale beyond a single system image.

In addition to improving performance, partitioning also provides higher availability of HPC systems. By reconfiguring a larger system into smaller partitions and using the partitions together as part of a "cluster," a hardware or OS failure within one partition or "cluster node" can be contained to prevent bringing down the rest of the system and software running on the other partitions or cluster nodes. Also, when a partition does crash due to a hardware failure (for example, a C-brick failure), the bad C-brick can be immediately disabled and the partition can be rebooted and put immediately back into service without the C-brick. When a new or replacement C-brick later arrives, you can "warm-swap" the failed C-brick into the system while leaving the system up. Then you can reboot the partition at a convenient time to resume using all of its C-bricks and running at 100% performance again.

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 IRIX 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 XFS filesystem, and the XVM Volume Manager.

Persistent PCI-X Bus Numbering

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. 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.

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.

You can activate persistent bus numbering in one of the following ways:

- Issue the `ioconfig` command, using the `ioconfigfile` configuration file name as an argument.
- Boot the kernel using the `ioconfig=` kernel boot options.

Note: If you use both the `ioconfig` command and the `ioconfig=` kernel boot options, the persistent bus numbering information in `ioconfigfile` will be ignored. The `ioconfig=` kernel boot options take precedence.

Using the `ioconfig` Command

The `ioconfig` command is executed at the Extensible Firmware Interface (EFI) shell. It accepts `ioconfigfile`, the configuration file from the current filesystem, as an argument. Following is an example:

```
EFI> fs0:
EFI> ioconfig ioconfigfile
```

The `ioconfigfile` file contains I/O brick module IDs. Comment lines start with `#`. Following is a sample `ioconfigfile`:

```
# ioconfigfile
101#01
101#02
```

The I/O brick module ID comes from the L2 display, as follows.

```
l2-pumpkin-001-L2>pwrr
001c11:
power appears on
001c27:
power appears on
101#01:
power appears on
101#02:
power appears on
```

As a result of the contents of the `ioconfigfile` file, the following assignments are made:

- I/O module 101#01 is assigned bus numbers 0x1 through 0x6.
- I/O module 101#02 is assigned bus numbers 0x11 through 0x16.

Using the `ioconfig=` Kernel Boot Option

You can also specify the `ioconfig=` kernel boot options at the EFI shell for persistent bus numbering, as in the following example:

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

Notice that the module IDs are specified as `101.01` and `101.02`, instead of `101#01` and `101#02`. This is because to the Linux kernel options parser, `#` indicates comments. Using the `#` would result in `101` instead of `101.01` and `101.02` for the `ioconfig` option when Linux is booted.

Persistent Naming of Ethernet Devices

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 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.

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
```

XSCSI Subsystem

The SGI XSCSI subsystem on Linux leverages from IRIX production quality and commercial proven code to provide more robust error handling, failover, and storage area network (SAN) infrastructure support as well as years of large system performance tuning. XSCSI takes advantage of specific features of the unique SGI architecture that standard open source drivers cannot without rewriting. Naming conventions are described in the following subsection.

XSCSI Device Naming

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.

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.

XFS Filesystem

The SGI XFS filesystem provides a high-performance filesystem for Linux. XFS is 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. While other filesystems are available on Linux, performance tuning and improvements leveraged from IRIX make XFS particularly well suited for large data and I/O workloads commonly found in HPC environments.

For more information on the XFS filesystem, see *Linux XFS Filesystem Administration*.

XVM Volume Manager

The SGI XVM Volume Manager provides a logical organization to disk storage that enables an administrator to combine underlying physical disk storage into a single logical unit, known as a logical volume. Logical volumes behave like standard disk partitions and can be used as arguments anywhere a partition can be specified.

A logical volume allows a filesystem or raw device to be larger than the size of a physical disk. Using logical volumes can also increase disk I/O performance because a volume can be striped across more than one disk. Logical volumes can also be used to mirror data on different disks.

HPC Application Tools and Support

SGI has ported HPC libraries, tools, and software packages from IRIX to Linux to provide a powerful, standards-based system using Linux and Itanium 2-based solutions for HPC environments.

Message Passing Toolkit

The SGI Message Passing Toolkit (MPT) provides industry-standard message passing libraries optimized for SGI computers. On Linux, MPT contains MPI and SHMEM APIs, which transparently utilize and exploit the low-level capabilities within SGI hardware, such as its block transfer engine (BTE) for fast memory-to-memory transfers and the hardware memory controller's fetch operation (fetchop) support. Fetchops enable direct communication and synchronization among multiple MPI processes while eliminating the overhead associated with system calls to the operating system.

Parallel workloads, such as MPI jobs, can be launched, monitored, and controlled across a cluster or partitioned system using the SGI Array Services software. Array Services provides the notion of an array session, which is a set of processes that can be running on different cluster nodes or system partitions. Array Services is implemented using Process Aggregates (PAGGs), which is a kernel module that provides process containers. PAGGs has been open-sourced by SGI for Linux.

For more information on the Message Passing Toolkit, see the *Message Passing Toolkit: MPI Programmer's Manual*.

Performance Co-Pilot (PCP)

The SGI PCP software was ported from IRIX to Linux to provide a collection of performance monitoring and performance management services targeted at large, complex systems. Integrated with the low-level performance hardware counters and with MPT, PCP provides such services as CPU, I/O, and networking statistics, visualization tools, and monitoring tools.

PROM Chips

Programmable read-only memory (PROM) chips are placed in your computer at the factory with software programmed into them that allows the CPU to boot and allows you to perform system administration and software installations. The PROM chips are not part of your disk or your operating system; they are the lowest level of access available for your system. You cannot erase them or bypass them. For more information on PROM, see the `prom(1)` man page.

Extensible Firmware Interface (EFI)

SGI Altix 3000 systems provide the EFI, a supporting platform to provide input to the CPU and to handle its output. In addition, the EFI controls the server's boot configuration, maintaining the boot menu in durable, nonvolatile memory.

From the EFI prompt, you can perform basic file-management tasks (including text editing), make configuration changes, or write scripts that execute at boot time. For a summary of EFI commands, see Table 3-2.

Table 3-2 EFI Commands

EFI Command	Description
alias [-bdv] [<i>sname</i>] [<i>value</i>]	Sets or gets alias settings
attrib [-b] [+/- <i>rhs</i>] [<i>file</i>]	Views or sets file attributes
bcfg	Configures boot driver and load options
cd [<i>path</i>]	Updates the current directory
cls [<i>background color</i>]	Clears screen
comp <i>file1 file2</i>	Compares two files
cp <i>file [file] ... [dest]</i>	Copies files or directories
date [<i>mm/dd/yyyy</i>]	Gets or sets date
dblk <i>device [Lba] [blocks]</i>	Performs hex dump of block I/O devices
dh [-b] [-p <i>prot_id</i>] [<i>handle</i>]	Dumps handle information
dmpstore	Dumps variable store
echo [-on -off] [<i>text</i>]	Echoes text to stdout or toggles script echo
edit [<i>file name</i>]	Edits a file
endfor	Script only: Delimits loop construct
endif	Script-only: Delimits IF THEN construct
err [<i>level</i>]	Sets or displays error level
exit	Exits

Table 3-2 EFI Commands (continued)

EFI Command	Description
<code>for var in set</code>	Script-only: Indicates loop construct
<code>getmtc</code>	Gets next monotonic count
<code>goto label</code>	Script-only: Jumps to label location in script
<code>guid [-b] [sname]</code>	Dumps known guid IDs
<code>help [-b] [internal command]</code>	Displays this help
<code>if [not] condition then</code>	Script-only: Indicates IF THEN construct
<code>load driver_name</code>	Loads a driver
<code>ls [-b] [dir] [dir] ...</code>	Obtains directory listing
<code>map [-bdvr] [sname[:]] [handle]</code>	Maps shortname to device path
<code>mem [address] [size] [;MMIO]</code>	Dumps memory or memory mapped I/O
<code>memmap [-b]</code>	Dumps memory map
<code>mkdir dir [dir] ...</code>	Makes directory
<code>mm address [width] [;type]</code>	Modifies memory: Mem, MMIO, IO, PCI
<code>mode [col row]</code>	Sets or gets current text mode
<code>mount BlkDevice [sname[:]]</code>	Mounts a filesystem on a block device
<code>mv sfile dfile</code>	Moves files
<code>pause</code>	Script-only: Prompts to quit or continue
<code>pci [bus dev] [func]</code>	Displays PCI device(s) info
<code>reset [cold/warm] [reset string]</code>	Indicates cold or warm reset
<code>rm file/dir [file/dir]</code>	Removes file or directories
<code>set [-bdv] [sname] [value]</code>	Sets or gets environment variable
<code>setsize newsize fname</code>	Sets the files size
<code>stall microseconds</code>	Delays for <i>x</i> microseconds

Table 3-2 EFI Commands (**continued**)

EFI Command	Description
<code>time [hh:mm:ss]</code>	Gets or sets time
<code>touch [filename]</code>	Views or sets file attributes
<code>type [-a] [-u] [-b] file</code>	Types file
<code>ver</code>	Displays version information
<code>vol fs [volume label]</code>	Sets or displays volume label

FLEXlm

FLEXlm is a robust and flexible license management system from Globetrotter Software that lets independent software vendors (ISVs) easily license their products and helps system administrators install and manage licenses with minimal overhead. It supports everything from simple node-locked licenses to floating licenses with redundant servers, and allows for a wide range of licensing options.

To build licensed software, ISVs must purchase a set of keys from Globetrotter Software. System administrators can install license servers anywhere. Products purchased from SGI are typically licensed using FLEXlm.

For more information, visit

<http://www.globetrotter.com/flexlm/flexlm.shtml>

SGIconsole

SGIconsole is a combination of hardware and software that provides console management and allows monitoring of multiple SGI servers running the IRIX operating system and SGI ProPack for Linux. These servers include SGI partitioned systems and large, single-system-image servers, including SGI Altix 3000 servers.

SGIconsole consists of an 1U rackmountable SGI server based on the Intel Pentium processor, a serial multiplexer or Ethernet hub, and a software suite that includes the Console Manager package and Performance Co-Pilot, which provides access to common remote management tools for hardware and software.

Console Manager is a graphical user interface for the SGIconsole management and monitoring tool used to control multiple SGI servers. SGIconsole also has a command line interface. For more information on SGIconsole, see the *SGIconsole 1.1 Start Here*.

System Controller Firmware

The L1 and L2 controllers are system controller firmware used in SGI systems.

The L1 controller is embedded in each brick in SGI Origin and Onyx 3000 series systems and in SGI Altix 3000 systems. It provides power and control sequencing, along with temperature and power monitoring for each brick.

The L2 controller is a rack-level controller that monitors and controls the bricks in its rack. All L2 controllers in a system are networked together and they consolidate the control and monitoring information from each brick to provide system-level control and monitoring.

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

NUMA Tools

NUMA tools provides a collection of NUMA related tools. This section describes the commands that are currently provided.

`dlook` Command

The `dlook` command displays the memory map and CPU use for a specified process. The following information is printed for each page in the virtual address space of the process:

- The object that owns the page (file, SYSV shared memory, device driver, and so on)
- Type of page (RAM, FETCHOP, IOSPACE, and so on)
- If RAM memory, the following information is supplied:
 - Memory attributes (SHARED, DIRTY, and so on)

- Node on which that the page is located
- Physical address of page (optional)

Optionally, the amount of elapsed CPU time that the process has executed on each physical CPU in the system is also printed.

dplace Command

The `dplace` command binds a related set of processes to specific CPUs or nodes to prevent process migrations. In some cases, this tool improves performance because of the occurrence of a higher percentage of memory accesses to the local node.

Performance Tuning

This chapter describes tuning issues for single processor and multiprocessor programs. It contains the following sections:

- See “Single Processor Code Tuning” on page 34 for details about performance tuning of single processor programs.
- See “Multiprocessor Code Tuning” on page 42 for details about performance tuning of multiprocessor programs.
- See “Profilers and Performance Tools” on page 47 for details about tools that can be used to tune performance of either type of code.

For a list of available references, see “Related Publications” on page xiv.

Determining System Configuration

To determine the details of the system you are running, you can browse files from the `/proc` pseudo-filesystem (see the `proc(5)` man page for details). Following is some of the information you can obtain:

- `/proc/cpuinfo`: displays processor information, one entry per processor. Use this to determine clock speed and processor stepping.
- `/proc/meminfo`: provides a global view of system memory usage, such as total memory, free memory, swap space, etc.
- `/proc/discontig`: shows memory usage (in pages).
- `/proc/pal/cpu0/cache_info`: provides detailed information about L1, L2, and L3 cache structure, such as size, latency, associativity, line size, etc. Other files in `/proc/pal/cpu0` provide information about the Translation Lookaside Buffer (TLB) structure, clock ratios, and other details.
- `/proc/version`: provides information about the installed kernel.

- `/proc/perfmon`: if this file does not exist in `/proc` (that is, if it has not been exported), performance counters have not been started by the kernel and none of the performance tools that use the counters will work.
- `/proc/mounts`: provides details about the filesystems that are currently mounted.
- `/proc/modules`: contains details about currently installed kernel modules.

You can also use the `uname` command, which returns the kernel version and other machine information. In addition, the `topology` command is a Bourne shell script that uses information in `/dev/hw` to display system configuration information. See the `topology(1)` man page for details.

The currently recommended compilers for SGI Altix 3000 systems are `efc` and `ecc` (the Intel Fortran compiler and Intel C/C++ compiler). Other compilers (such as the GNU set of compilers) can be used; however, `efc` and `ecc` provide the best performance on Linux systems. Examples in this chapter, unless noted otherwise, use the `efc` or `ecc` compiler.

Single Processor Code Tuning

Several basic steps are used to tune performance of single-processor code:

- Get the expected answers and then tune performance. For details, see “Getting the Correct Results” on page 35.
- Use existing tuned code, such as that found in math libraries and scientific library packages. For details, see “Using Tuned Code” on page 38.
- Determine what needs tuning. For details, see “Determining Tuning Needs” on page 39.
- Use the compiler to do the work. For details, see “Using Compiler Options Where Possible” on page 39.
- Consider tuning cache performance. For details, see “Tuning the Cache Performance” on page 40.
- Set environment variables to enable higher-performance memory management mode. For details, see “Managing Memory” on page 41.
- Change stack and data segments. For details, see “Using `chatr` to Change Stack and Data Segments” on page 41.

Getting the Correct Results

One of the first steps in performance tuning is to verify that the correct answers are being obtained. Once the correct answers are obtained, tuning can be done. You can verify answers by initially disabling specific optimizations and limiting default optimizations. This can be accomplished by using specific compiler options and by using debugging tools.

The following compiler options emphasize tracing and porting over performance:

- `-O`: the `-O0` option disables all optimization. The default is `-O2`.
- `-g`: the `-g` option preserves symbols for debugging.
- `-mp`: the `-mp` option limits floating-point optimizations and maintains declared precision.
- `-IPF_fltacc`: the `-IPF_fltacc` option disables optimizations that affect floating-point accuracy.
- `-r`; `-i`: the `-r8` and `-i8` options set default real, integer, and logical sizes to 8 bytes, which are useful for porting Cray codes. **This explicitly declares intrinsic and external library functions.**

Some debugging tools can also be used to verify that correct answers are being obtained. The following debuggers can be used:

- `gdb`: the GNU project debugger. This is useful for debugging programs written in C, C++, and Fortran 95. When compiling with C and C++, include the `-g` option on the compiler command line to produce the `dwarf2` symbols database used by `gdb`.

When using `gdb` for Fortran debugging, include the `-g` and `-O0` options. Do not use `gdb` for Fortran debugging when compiling with `-O1` or higher.

The debugger to be used for Fortran 95 codes can be downloaded from http://sourceforge.net/project/showfiles.php?group_id=56720. (Note that the standard `gdb` compiler does not support Fortran 95 codes.) To verify that you have the correct version of `gdb` installed, use the `gdb -v` command. The output should appear similar to the following:

```
GNU gdb 5.1.1 FORTRAN95-20020628 (RC1)
Copyright 2002 Free Software Foundation, Inc.
```

For a complete list of `gdb` commands, see the `gdb` user guide online at http://sources.redhat.com/gdb/onlinedocs/gdb_toc.html or use the `help` option. Note that current instances of `gdb` do not report `ar.ec` registers correctly. If you are debugging rotating, register-based, software-pipelined loops at the assembly code level, try using `idb` instead.

- `idb`: the Intel debugger. This is a fully symbolic debugger for the Linux platform. The debugger provides extensive support for debugging programs written in C, C++, FORTRAN 77, and Fortran 90. At this time, `idb` cannot be used to debug multithreaded or multiprocessor programs on Itanium systems.

Running `idb` with the `-gdb` option on the shell command line provides `gdb`-like user commands and debugger output.

- `ddd`: a GUI to a command line debugger. It supports `gdb` and `idb`. For details about usage, see “Using `ddd`” on page 36.

Using `ddd`

The `ddd` tool is a GUI to an arbitrary command line debugger. To instantiate `ddd`, use the `--debugger` option to specify the debugger used (for example, `--debugger "idb"`). The default debugger used is `gdb`.

The resulting screen is divided into panes that show:

- Array inspection
- Source code
- Disassembled code
- A command line window to the debugger engine

These panes can be switched on and off from the **View** menu.

Some commonly used commands can be found on the menus. In addition, the following actions can be useful:

- Select an address in the assembly view, click the right mouse button, and select `lookup`. The `gdb` command is executed in the command pane and it shows the corresponding source line.
- Select a variable in the source pane and click the right mouse button. The current value is displayed. Arrays are displayed in the array inspection window. You can print these arrays to PostScript by using the **Menu->Print Graph** option.

- You can view the contents of the register file, including general, floating-point, NaT, predicate, and application registers by selecting **Registers** from the **Status** menu. The **Status** menu also allows you to view stack traces or to switch threads.

Managing Heap Corruption Problems

Two methods can be used to check for heap corruption problems in programs that use `glibc malloc/free` dynamic memory management routines: environment variables and Electric Fence.

Set the `MALLOC_CHECK_` environment variable to 1 to print diagnostic messages or to 2 to abort immediately when heap corruption is detected.

Electric Fence is a `malloc` debugger shipped with Red Hat Linux. It aligns either the start or end of an allocated block with an invalid page, causing segmentation faults to occur on buffer overruns or underruns at the point of error. It can also detect accesses to previously freed regions of memory.

Underruns and overruns cannot be simultaneously detected. The default behavior is to place inaccessible pages immediately after allocated memory, but the complementary case can be enabled by setting the `EF_PROTECT_BELOW` environment variable. To use Electric Fence, link with the `libefence` library, as shown in the following example:

```
% cat foo.c
#include <stdio.h>
#include <stdlib.h>
int main (void)
{
    int i;
    int * a;
    float *b;

    a = (int *)malloc(1000*sizeof (int));
    b = (float *)malloc(1000*sizeof (float));

    a[0]=1;
    for (i=1 ; i<1001;i++)
        {
            a[i]=a[i-1]+1;
        }
    for (i=1 ; i<1000;i++)
        {
            b[i]=a[i-1]*3.14;
        }
}
```

```
    }  
  
    printf("answer is %d %f \n"a[999],b[999]);  
  
}
```

Compile and run the program as follows (note the error when it is compiled with the library call):

```
% ecc foo.c  
% ./a.out  
answer is 1000 3136.860107  
% ecc foo.c -lefence  
% ./a.out
```

```
Electric Fence 2.2.0 Copyright (C) 1987-1999 Bruce Perens  
Segmentation fault  
%
```

To avoid potentially large core files, the recommended method of using Electric Fence is from within a debugger. See the `efence` man page for additional details.

Using Tuned Code

Where possible, use code which has already been tuned for optimum hardware performance.

The following mathematical functions should be used where possible to help obtain best results:

- **MKL:** Intel's Math Kernel Library. This library includes BLAS, LAPACK, and FFT routines.
- **VML:** the Vector Math Library, available as part of the MKL package (`libmkl_vml_itp.so`).
- **Standard Math library:** standard math library functions are provided with the Intel compiler's `libimf.a` file. If the `-lm` option is specified, `glibc libm` routines are linked in first.

Documentation is available for MKL and VML at the following website:

```
http://intel.com/software/products/perflib/  
index.htm?iid=ipp\_home+software\_libraries&.
```


Determining Tuning Needs

Determine what points in your code might benefit from tuning. Use the following tools:

<code>time</code>	Use this command to obtain an overview of user, system, and elapsed time.
<code>gprof</code>	Use this tool to obtain an execution profile of your program (a <code>pcsamp</code> profile). Use the <code>-p</code> compiler option to enable <code>gprof</code> use.
VTune	This Intel performance monitoring tool is a Linux-server, Windows-client application. It supports remote sampling on all Itanium and Linux systems.
<code>pfmon</code>	This performance monitoring tool is designed for Itanium and Linux. It uses the Itanium Performance Monitoring Unit (PMU) to do counting and sampling on unmodified binaries.

For information about VTune and `pfmon`, see “Profilers and Performance Tools” on page 47.

Using Compiler Options Where Possible

Several compiler options can be used to optimize performance. For a short summary of `efc` or `ecc` options, use the `-help` option on the compiler command line. Use the `-dryrun` option to show the driver tool commands that `efc` or `ecc` generate. This option does not execute tools.

Use the following options to help tune performance:

<code>-ftz</code>	Flushes underflow to zero to avoid kernel traps. Enabled by default at <code>-O3</code> optimization.
<code>-fno-alias</code>	Assumes no pointer aliasing. Other aliasing options include <code>-ansi_alias</code> and <code>-fno_fnalias</code> . Note that alias assertions may generate incorrect code.
<code>-ip</code>	Generates single file, interprocedural optimization.
<code>-ipo</code>	Generates multifile, inter-procedural optimization.
<code>-O3</code>	Enables <code>-O2</code> optimizations plus more aggressive optimizations, including loop transformation and prefetching. Level 3 optimization may not improve performance for all programs.

- `-opt_report` Generates an optimization report and places it in the file specified in `-opt_report_file`.
- `-override_limits` This is an undocumented option that sometimes allows the compiler to continue optimizing when it has hit an internal limit.
- `-prof_gen` and `-prof_use` Generates and uses profiling information. These options require a three-step compilation process:
1. Compile with proper instrumentation using `-prof_gen`.
 2. Run the program on one or more training datasets.
 3. Compile with `-prof_use`, which uses the profile information from the training run.
- `-S` Compiles and generates assembly listing in `.s` files and does not link. The assembly listing can be used in conjunction with the output generated by the `-opt_report` option to try to determine how well the compiler is optimizing loops.

Tuning the Cache Performance

There are several actions you can take to help tune cache performance:

- Avoid large power-of-2 strides and dimensions that cause cache thrashing.
- Use strides of 1 wherever possible.
- Cache bank conflicts can occur if there are two accesses to the same 16-byte-wide bank at the same time. Try different padding of arrays if the output from the `pfmon -e L2_OZQ_CANCEL_S1_BANK_CONF` command and the output from the `pfmon -e CPU_CYCLES` command shows a high number of bank conflicts relative to total CPU cycles. These can be combined into one command:

```
% pfmon -e CPU_CYCLES,L2_OZQ_CANCEL_S1_BANK_CONF a.out
```

A maximum of four performance monitoring events can be counted simultaneously.

- Group together data that is used at the same time and do not use vectors in your code, if possible.
- Try to avoid the use of temporary arrays and minimize data copies.

Managing Memory

Codes that frequently allocate and deallocate memory through `glibc malloc/free` calls may accrue significant system time due to memory management overhead. By default, `glibc` strives for system-wide memory efficiency at the expense of performance. A detailed discussion of this issue can be found at <http://www.linuxshowcase.org/ezolt.html>.

To enable the higher-performance memory management mode, set the following environment variables:

```
% setenv MALLOC_TRIM_THRESHOLD_ -1
% setenv MALLOC_MMAP_MAX_ 0
```

Because allocations in `efc` using the `malloc` intrinsic use the `glibc malloc` internally, these environment variables are also applicable in Fortran codes using, for example, Cray pointers with `malloc/free`. But they do not work for Fortran 90 allocatable arrays, which are managed directly through Fortran library calls.

Using `chatr` to Change Stack and Data Segments

The `chatr` command changes the ELF header of a program so that when the program is run, instructions in the stack and data segments are or are not executable.

The default is to allow the stack and data segments to be executable. For certain workloads, a performance improvement can be obtained by turning off the default behavior and making the stack and code segments unexecutable. This typically benefits workloads that do a large number of `fork()` calls. Workloads without a large number of `fork()` calls will probably not see a performance improvement.

You can change this behavior system-wide by using the `sysctl` command. The following command disallows executable stacks and data segments:

```
% sysctl vm.executable_stacks=0
```

The following command allows executable stacks and data segments:

```
% sysctl vm.executable_stacks=1
```

Note that some programs may now fail with a `SEGV` error even though they worked correctly before. Usually, these are programs that store instructions in the stack or data segment and then branch to those instructions (for example, older versions of the X server that load graphics drivers into the data segment, or Java JIT compilers). You can use the `chattr` command to make these programs work correctly, regardless of the value of `vm.executable_stacks`.

For details about usage, see the `chattr(1)` man page.

Multiprocessor Code Tuning

Before beginning any multiprocessor tuning, first perform single processor tuning. This can often obtain good results in multiprocessor codes also. For details, see “Single Processor Code Tuning” on page 34.

Multiprocessor tuning consists of the following major steps:

- Choose the parallelization methodology for your code. For details, see “Parallelizing Your Code” on page 42.
- Analyze your code to make sure it is parallelizing properly. For details, see “Solving Bottlenecks in Code” on page 44.
- Check to determine if false sharing exists. For details, see “Fixing False Sharing” on page 44.
- Tune for data placement. For details, see “Using `dplace` and `runon`” on page 45.
- Use environment variables to assist with tuning. For details, see “Environment Variables for Performance Tuning” on page 46.

Parallelizing Your Code

The first step in multiprocessor performance tuning is to choose the parallelization methodology that you want to use for tuning. You can use the following options:

- Use the Message Passing Interface (MPI) from the SGI Message Passing Toolkit (MPT). MPI is optimized and more scalable for SGI 3000 series systems than generic

MPI libraries. It takes advantage of the SGI Altix 3000 architecture and SGI Linux NUMA features.

Use the `-lmpi` compiler option to use MPI. For a list of environment variables that are supported, see the `mpi` man page. Those variables that are valid for IRIX systems only are so noted on the man page.

`MPIO_DIRECT_READ` and `MPIO_DIRECT_WRITE` are supported under Linux for local XFS filesystems in SGI MPT version 1.6.1 and beyond.

- Use OpenMP. When using C, C++, or Fortran code with OpenMP directives, you can use the following compiler options:

`efc -openmp` or `ecc -openmp`

These options use the Intel OpenMP front-end that is built into the Intel compilers. The resulting executable file makes calls to `libguide.so`, which is the Intel OpenMP run-time library.

`guide` An alternate command to invoke the Intel compilers to use OpenMP code. Use `guidec` (in place of `ecc`), `guideefc` (in place of `efc`), or `guidec++` to translate code with OpenMP directives into code with calls to `libguide`. See “Other Performance Tools” on page 50 for details.

The `-openmp` option to `efc` is the Intel long-term OpenMP compiler for Linux. However, if you have performance problems with this option, using `guide` might provide improved performance.

- Use the compiler to invoke automatic parallelization. Use the `-parallel` and `-par_report` option to the `efc` or `ecc` compiler. These options show which loops were parallelized and the reasons why some loops were not parallelized. If a source file contains many loops, it might be necessary to add the `-override_limits` flag to enable automatic parallelization. The code generated by `-parallel` is based on the OpenMP API, and the standard OpenMP environment variables and Intel extensions apply.

Determine the amount of code that is parallelized. Use the following formula to calculate the amount of code that is parallelized:

$$p = N(T(1) - T(N)) / T(1)(N - 1)$$

In this equation, $T(1)$ is the time the code runs on a single CPU and $T(N)$ is the time it runs on N CPUs. Speedup is defined as $T(1)/T(N)$.

If $speedup/N$ is less than 50% (that is, $N > (2-p) / (1-p)$), stop using more CPUs and tune for better scalability.

CPU activity can be displayed with the `top` or `vmstat` commands or accessed via the PCP tools (for example, `pmval kernel.percpu.cpu.user`) or via PCP visualization tools on a remote IRIX workstation. Hardware configuration information can be displayed using the `/proc/cpuinfo` or `/proc/meminfo` script. Detailed cache (and other) information can be displayed using the `/proc/pal/cpuX` script, where `X` is a CPU number.

Solving Bottlenecks in Code

For details about the different profiling tools that can pinpoint the bottlenecks in your code, see “Profilers and Performance Tools” on page 47.

Fixing False Sharing

If the parallel version of your program is slower than the serial version, false sharing might be occurring. False sharing occurs when two or more data items that appear not to be accessed by different threads in a shared memory application correspond to the same cache line in the processor data caches. If two threads executing on different CPUs modify the same cache line, the cache line cannot remain resident and correct in both CPUs, and the hardware must move the cache line through the memory subsystem to retain coherency. This causes performance degradation and reduction in the scalability of the application. If the data items are only read, not written, the cache line remains in a shared state on all of the CPUs concerned. False sharing can occur when different threads modify adjacent elements in a shared array.

You can use the following methods to verify that false sharing is happening:

- Use the performance monitor to look at output from `pfmon` and the `BUS_MEM_READ_BRILL_SELF` and `BUS_RD_INVALID_ALL_HITM` events.
- Use `pfmon` to check `DEAR` events to track common cache lines.
- Use the PCP `pmshub` utility to monitor cache traffic and CPU utilization.

Note: The `pmshub` utility may not be available for this release. Consult your release notes for information about its availability.

If false sharing is a problem, try the following solutions:

- Use the HW counter to run a profile that monitors storage to shared cache lines. This will show the location of the problem.
- Revise data structures or algorithms.
- Check shared data, static variables, common blocks, and private and public variables in shared objects.
- Use critical regions to identify the part of the code that has the problem.

Using `dplace` and `runon`

The `dplace` command binds processes to specified CPUs in a round-robin fashion. Once bound to a process, they do not migrate. This is similar to `_DSM_MUSTRUN` on IRIX systems. `dplace` numbering is done in the context of the current CPU memory set.

The `runon` command restricts execution to the listed set of CPUs; however, processes are still free to move among listed CPUs.

For more details on these commands, see the following sections.

`dplace` for MPI Codes

For best placement and for CPU-data affinity, use one of the following methods:

- Use `MPI_DSM_MUSTRUN` in a `cpuset`.
- Set the `MPI_DSM_CPULIST` environment variable.
- Use the `dplace -s1` command.

When using SGI MPI-compiled codes, use the `-s1` option and provide the full path to `dplace`, as in the following example:

```
% mpirun -np 32 /usr/bin/dplace -c16-47 -s1 ./a.out
```

Using the `-s1` option directs `dplace` to skip placement of the lightweight MPI “shepherd” process.

dplace for OpenMP Codes

When using OpenMP codes from Intel, use the `-x6` option, as in the following example:

```
% setenv OMP_NUM_THREADS 4  
% dplace -x6 -c5-8 ./a.out
```

The `-x6` option directs `dplace` to skip placement of the two lightweight OpenMP shepherd processes.

Notice the distinction between `dplace` and `runon`:

```
% setenv OMP_NUM_THREADS 4  
% dplace -x6 -c5-8 ./a.out  
% runon 5-8 ./a.out
```

The `dplace` command allows you to skip placement of the shepherd processes, while `runon` does not. `dplace` also locks each thread to its corresponding CPU; `runon` restricts the threads to CPUs 5 through 8, but they are free to migrate among those CPUs.

For information about `dplace` and its usage, see the *Linux Resource Administration Guide*.

Environment Variables for Performance Tuning

You can use several different environment variables to assist in performance tuning. For details about environment variables used to control the behavior of MPI, see the `mpi(1)` man page.

Several OpenMP environment variables can affect the actions of the OpenMP library. For example, some environment variables control the behavior of threads in the application when they have no work to perform or are waiting for other threads to arrive at a synchronization semantic; other variables can specify how the OpenMP library schedules iterations of a loop across threads. The following environment variables are part of the OpenMP standard:

- `OMP_NUM_THREADS` (The default is the number of CPUs in the system.)
- `OMP_SCHEDULE` (The default is `static`.)

- `OMP_DYNAMIC` (The default is `false`.)
- `OMP_NESTED` (The default is `false`.)

In addition to the preceding environment variables, Intel provides several OpenMP extensions, two of which are through the use of the `KMP_LIBRARY` variable.

The `KMP_LIBRARY` variable sets the run-time execution mode, as follows:

- If set to `serial`, single-processor execution is used.
- If set to `throughput`, CPUs yield to other processes when waiting for work. This is the default and is intended to provide good overall system performance in a multiuser environment. This is analogous to the IRIX `_DSM_WAIT=YIELD` variable.
- If set to `turnaround`, worker threads do not yield while waiting for work. This is analogous to the IRIX `_DSM_WAIT=SPIN` variable. Setting `KMP_LIBRARY` to `turnaround` may improve the performance of benchmarks run on dedicated systems, where multiple users are not contending for CPU resources.

If your program gets a segmentation fault immediately upon execution, you may need to increase `KMP_STACKSIZE`. This is the private stack size for threads. The default is 4 MB. You may also need to increase your shell stacksize limit.

Profilers and Performance Tools

Several tools are available to help determine areas for performance improvements. The following sections describe some of those tools.

Profiling with `pfmon`

The `pfmon` tool is a performance monitoring tool designed for Linux. It uses the Itanium Performance Monitoring Unit (PMU) to count and sample on unmodified binaries. In addition, it can be used for the following tasks:

- To monitor unmodified binaries in its per-process mode.
- To run system-wide monitoring sessions. Such sessions are active across all processes executing on a given CPU.
- Launch a system-wide session on a dedicated CPU or a set of CPUs in parallel.

- Monitor activities happening at the user level or at the kernel level.
- Collect basic event counts.
- Sample program or system execution, monitoring up to 4 events at a time.

To see a list of available options, use the `pfmon -help` command.

The `profile.pl` script

The `profile.pl` script handles the entire user program profiling process, including using `dplace`. Typical usage is as follows:

```
% profile.pl -c0-3 -x6 command args
```

This script designates processors 0 through 3. The `-x6` option is necessary only for OpenMP codes.

The result is a profile taken on the `CPU_CYCLES` PMU event and placed into `profile.out`. This script also supports profiling on other events such as `IA64_INST_RETIRED`, `L3_MISSES`, and so on; see `pfmon -l` for a complete list of PMU events. The script handles running the command under the performance monitor, creating a map file of symbol names and addresses from the executable and any associated dynamic libraries, and running the profile analyzer.

See the `profile.pl(1)`, `analyze.pl(1)`, and `makemap.pl(1)` man pages for details.

`profile.pl` with MPI programs

For MPI programs, use the `profile.pl` command with the `-s1` option, as in the following example:

```
% mpirun -np 4 profile.pl -s1 -c0-3 test_prog </dev/null
```

The use of `/dev/null` ensures that MPI programs run in the background without asking for TTY input.

Using VTune for Remote Sampling

The Intel VTune performance analyzer does remote sampling experiments. The VTune data collector runs on the Linux system and an accompanying GUI runs on an IA-32 Windows machine, which is used for analyzing the results.

For details about using VTune, see the following URL:

```
http://developer.intel.com/software/products/vtune/vtune61/  
index.htm.
```

Note: VTune may not be available for this release. Consult your release notes for details about its availability.

Using GuideView

GuideView is a graphical tool that presents a window into the performance details of a program's parallel execution. GuideView is part of the KAP/Pro Toolset, which also includes the Guide OpenMP compiler and the Assure Thread Analyzer. GuideView is not a part of the default software installation with your system.

GuideView uses an intuitive, color-coded display of parallel performance bottlenecks which helps pinpoint performance anomalies. It graphically illustrates each processor's activity at various levels of detail by using a hierarchical summary.

Statistical data is collapsed into relevant summaries that indicate where attention should be focused (for example, regions of the code where improvements in local performance will have the greatest impact on overall performance).

To gather programming statistics, use the `-O3`, `-openmp`, and `-openmp_profile` compiler options. This causes the linker to use `libguide_stats.a` instead of the default `libguide.a`. The following example demonstrates the compiler command line to produce a file named `swim`:

```
% efc -O3 -openmp -openmp_profile -o swim swim.f
```

To obtain profiling data, run the program, as in this example:

```
% export OMP_NUM_THREADS=8  
% ./swim < swim.in
```

When the program finishes, the `swim.gvs` file is produced and it can be used with GuideView. To invoke GuideView with that file, use the following command:

```
% guideview -jpath=your_path_to_Java -mhz=998 ./swim.gvs.
```

The graphical portions of GuideView require the use of Java. Java 1.1.6-8 and Java 1.2.2 are supported and later versions appear to work correctly. Without Java, the functionality is severely limited but text output is still available and is useful, as the following portion of the text file that is produced demonstrates:

```
Program execution time (in seconds):
cpu           :           0.07 sec
elapsed      :          69.48 sec
  serial     :           0.96 sec
  parallel   :          68.52 sec
cpu percent   :           0.10 %
end

Summary over all regions (has 4 threads):
# Thread      :           #0      #1      #2      #3
Sum Parallel  :    68.304    68.230    68.240    68.185
Sum Imbalance :     1.020     0.592     0.892     0.838
Sum Critical Section:  0.011     0.022     0.021     0.024
Sum Sequential :     0.011  4.4e-03  4.6e-03  1.6e-03
Min Parallel  :   -5.1e-04 -5.1e-04  4.2e-04 -5.2e-04
Max Parallel  :     0.090     0.090     0.090     0.090
Max Imbalance :     0.036     0.087     0.087     0.087
Max Critical Section:  4.6e-05  9.8e-04  6.0e-05  9.8e-04
Max Sequential :     9.8e-04  9.8e-04  9.8e-04  9.8e-04
end
```

Other Performance Tools

The following performance tools also can be of benefit when you are trying to optimize your code:

- *Guide OpenMP Compiler* is an OpenMP implementation for C, C++, and Fortran from Intel.
- *Assure Thread Analyzer* from Intel locates programming errors in threaded applications with no recoding required.

For details about these products, see

<http://developer.intel.com/software/products/threadtool.htm>.

Note: These products have not been thoroughly tested on SGI systems. SGI takes no responsibility for the correct operation of third party products described or their suitability for any particular purpose.

Index

A

Array Services, description, 16
Autopartition option, 8

B

Booting the CD, 7, 12
Bus numbering, 22
Buttons for returning, 6

C

Cache performance, 40
CD contents, 2
Changing the ELF header, 41
Compilers, 34
Comprehensive System Accounting (CSA) support, 20
CpuMemSet support, 19
CSA support, 20

D

Disk partition table, 6
dlook command, 31
dplace command, 32
for MPI, 45

for OpenMP, 46

E

EFI commands, 28
EFI support, 28
Electric Fence debugger, 37
Environment variables, 46
Ethernet device naming, 24
Extensible Firmware Interface (EFI) support, 28

F

False sharing, 44
File configurations, 6
Filesystem recovery, 12
Firewall configuration, 8
FLEXlm support, 30

G

gdb tool, 35
GNU debugger, 35
GuideView tool, 49

H

hardware platforms, 1

HPC support

Extensible Firmware Interface (EFI), 28

FLEXlm, 30

libraries and tools, 26

Message Passing Toolkit, 27

Performance Co-Pilot (PCP), 27

PROM chips, 27

SGIconsole, 30

System Controller Firmware, 31

I

idb tool, 36

Installation

additional users, 9

CDs, 9

log, 10

method selection, 7, 13

open/free source software, 9

overview, 4

package group selection, 9, 10

password for access, 9

proprietary software, 9

reboot, 10

SGI Linux Environment 7.2 base OS, 6

SGI Linux Environment 7.2 updates, 10

SGI ProPack for Linux, 9

system controller software, 11

time zone, 9

upgrading, 11

user creation, 9

I/O subsystems

for HPC systems, 21

XFS filesystem support, 26

XSCSI support, 25

XVM support, 26

ioconfig command for bus numbering, 23

ioconfig= kernel boot option, 24

IP address selection, 8

L

L1 and L2 controllers, 31

Language selection, 7, 8, 12

Linux Kernel Crash Dump (LKCD), 6

LKCD for file configuration, 6

M

Message Passing Toolkit

for parallelization, 42

support, 27

using dplace, 45

using profile.pl, 48

N

NUMA tools, 31

dllook command, 31

dplace command, 32

O

Open source website, 4

OpenMP

environment variables, 46

Guide OpenMP Compiler, 50

with dplace, 46

OS configuration settings, 1

OS enhancements

Comprehensive System Accounting (CSA), 20

CpuMemSets, 19

cpuset support, 20
for HPC environments, 19
partitioning, 21

P

Package group selection, 9, 10
Parallelization
 automatic, 43
 using OpenMP, 43
 using MPI, 42
Partition selection, 8
Partitioning support, 21
Password requirement, 5
PCP support, 27
Performance
 Assure Thread Analyzer, 50
 Guide OpenMP Compiler, 50
 GuideView, 49
 measuring, 16
 VTune, 49
Performance Co-Pilot support, 27
Persistent bus numbering, 22
Persistent naming
 bus numbers, 22
 Ethernet devices, 24
 XSCSI devices, 25
pfmon tool, 47
Product list, 15
Product support, 15, 33
profile.pl script, 48
Profiling
 pfmon, 47
 profile.pl, 48
PROM support, 27

R

Release features, 1
Remove partitions, 8
Rescue mode, 12
Root access password, 9

S

Screen
 Complete, 10
 Installation to Begin, 10
 Package Group Selection, 9, 10
 Package Installation, 10
 Welcome, 9, 10
SGIconsole support, 30
Software
 installation, 5
 introduction, 1
 planning, 5
System, 31
System controller firmware support, 31
System Controller Software
 CD, 11

T

Tools
 Assure Thread Analyzer, 50
 Guide OpenMP Compiler, 50
 GuideView, 49
 pfmon, 47
 profile.pl, 48
 VTune, 49
Tuning
 cache performance, 40
 ddd tool, 36

- debugging tools
 - Electric Fence, 37
 - gdb, 35
 - idb, 36
- determining needs, 39
- dplace, 45
- Electric Fence, 37
- environment variables, 46
- false sharing, 44
- heap corruption, 37
- managing memory, 41
- multiprocessor code, 42
- parallelization, 42
- profiling
 - GuideView, 49
 - mpirun command, 48
 - pfmon, 47
 - profile.pl script, 48
 - VTune analyzer, 49
- recommended compilers, 34
- single processor code, 34
- solving bottlenecks, 44
- system configuration, 33
- using chatr, 41
- using compiler options, 39
- using dplace, 45
- using math functions, 38
- using runon, 45
- verifying correct results, 35

U

- Unsupported elements, 18
- Upgrading software, 11

V

- VTune performance analyzer, 49

X

- XFS filesystem support, 26
- XSCSI device naming, 25
- XSCSI support, 25
- XVM support, 26