Linux® Configuration and Operations Guide

007–4633–001

CONTRIBUTORS

Written by Terry Schultz

Illustrated by Chrystie Danzer

Production by Karen Jacobson

Engineering contributions by John Hawkes, Dan Higgins, Erik Jacobson, Kevin McMahon, Dean Roe

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

Record of Revision

Version	Description
001	October 2003 Original publication.

Contents

Figures

Procedures

About This Guide

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

This manual contains the following chapters:

- Chapter 1, "Configuring Your System", page 1
- Chapter 2, "System Operation", page 21
- Chapter 3, "Performance Tuning", page 35
- Chapter 4, "NUMA Tools", page 59
- Chapter 5, "Kernel Tunable Parameters on SGI ProPack Servers", page 73
- Appendix A, "Linux Kernel Tunable Parameters", page 77

Related Publications

The following publications contain additional information that may be helpful:

- *SGI ProPack for Linux Start Here* provides information about the SGI ProPack for Linux release including information about major new features, software installation, product support, and performance tuning.
- *SGI ProPack v2.3 for Linux Release Notes* provides the latest information about software and documentation in this release. The release notes are on the SGI ProPack for Linux Documentation CD in the `root` directory, in a file named `README.TXT`.
- *SGI Altix 3000 User's Guide* provides an overview of the architecture and descriptions of the major components that make up the SGI Altix 3000 computer system. It also describes the standard procedures for powering up and powering down the system, basic troubleshooting information, and it includes important safety and regulatory specifications.

- *SGI L1 and L2 Controller Software User's Guide* describes how to use the L1 and L2 controller commands at your system console to monitor and manage your system.
- *Console Manager for SGIconsole Administrator's Guide* describes the Console Manager software graphical interface, which allows you to control multiple SGI servers; SGI partitioned systems; and large, single-system image servers.
- *Linux Resource Administration Guide* is a reference document for people who manage the operation of SGI computer systems running the Linux operating system. It contains information needed in the administration of various system resource management features such as Comprehensive System Accounting (CSA), Array Services, CPU memory sets (CpuMemSets) and scheduling, and the Cpuset System.
- *Performance Co-Pilot for IA-64 Linux User's and Administrator's Guide* documents the Performance Co-Pilot software package running on IA-64 Linux systems. Performance Co-Pilot provides a systems-level suite of tools that cooperate to deliver integrated performance monitoring and performance management services spanning the hardware platforms, operating systems, service layers, database management systems (DBMSs), and user applications.
- *Linux Device Driver Programmer's Guide-Porting to SGI Altix 3000 Systems* provides information on programming, integrating, and controlling drivers.
- *Message Passing Toolkit: MPI Programmer's Manual* describes industry-standard message passing protocol optimized for SGI computers.
- *XFS for Linux Administration* describes XFS, an open-source, fast recovery, journaling filesystem that provides direct I/O support, space preallocation, access control lists, quotas, and other commercial file system features.
- *Origin 2000 and Onyx2 Performance Tuning and Optimization Guide* contains information specific to MIPS/IRIX systems, but the general guidelines in the document are hardware and operating system independent.
- *Event Manager User Guide* provides information about the Event Manger application that collects event information from other applications. This document describes the Event Manager application, the application programming interface that you can use to access it, the procedures that you can use to communicate with it from another application, and the commands that you can use to control it.
- *Embedded Support Partner User Guide* provides information about using the Embedded Support Partner (ESP) software suite to monitor events, set up proactive notification, and generate reports. This revision of the document

describes ESP version 3.0, which is the first version of ESP that supports the Linux operating system.

Additional Reading

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

Linux System Administration

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

`/usr/src/linux-2.4.20/Documentation`

Linux system administration course information is available at:

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

Intel Compiler Documentation

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

- *Intel C++ Compiler User's Guide* (`c_ug_lnx.pdf`)
- *Intel Fortran Compiler User's Guide* (`for_ug_lnx.pdf`)
- *Intel Fortran Programmer's Reference* (`for_prg.pdf`)
- *Intel Fortran Libraries Reference* (`for_lib.pdf`)

Other Intel Documentation

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

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

`http://developer.intel.com/design/itanium/manuals`

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

`http://developer.intel.com/design/itanium/manuals`

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

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

(secure channel required)

Open-Source Documents

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

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

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

Obtaining Publications

You can obtain SGI documentation in the following ways:

- See the SGI Technical Publications Library at: `http://docs.sgi.com`. Various formats are available. This library contains the most recent and most comprehensive set of online books, release notes, man pages, and other information.
- SGI ProPack for Linux documentation, and all other documentation included in the RPMs on the distribution CDs can be found on the CD titled "SGI ProPack V.2.3 for Linux - Documentation CD." To access the information on the documentation CD, open the index.html file with a web browser. Because this online file can be updated later in the release cycle than this document, you should check it for the latest information. After installation, all SGI ProPack for Linux documentation (including `README.SGI`) is in `/usr/share/doc/sgi-propack-2.3`.
- You can view man pages by typing man *title* on a command line.

Conventions

The following conventions are used throughout this document:

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

Reader Comments

If you have comments about the technical accuracy, content, or organization of this 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 Parkway, 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.

Chapter 1

Configuring Your System

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

- "Cloning System Disks", page 1
- "Setting up `quota` on the `root` File System", page 4
- "System Partitioning ", page 5
- "Making Array Services Operational", page 19
- "Pluggable Authentication Modules", page 20

Cloning System Disks

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

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

This sections describes how to clone a disk on an SGI Altix 3000 system.

The following assumptions have been made:

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

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

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

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

Procedure 1-1 Cloning System Disks

To clone a disk on an SGI Altix 3000 system, perform the following steps:

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

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

   ```
   elilo sgilinux single
   ```

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

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

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

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

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

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

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

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

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

5. Put the filesystems and swap space on the partitions. Partition 1 is EFI, partition 2 is swap, and partition 3 is root.
6. Make the `vfat` filesystem for EFI, as follows:

   ```
   mkfs.msdos /dev/xscsi/pci01.03.0-1/target2/lun0/part1
   ```
7. Make the swap partition, as follows:

   ```
   mkswap /dev/xscsi/pci01.03.0-1/target2/lun0/part2
   ```
8. Make the root filesystem, as follows:

   ```
   mkfs.xfs /dev/xscsi/pci01.03.0-1/target2/lun0/part3
   ```
9. Mount the newly created EFI filesystem for replication, as follows:

   ```
   mount /dev/xscsi/pci01.03.0-1/target2/lun0/part1 /mnt
   ```
10. Replicate the EFI filesystem to the target disk, for example:

    ```
    cd /boot/efi; tar cBf - . | (cd /mnt && tar xBf -)
    ```
11. Unmount the target EFI filesystem, as follows:

    ```
    umount /mnt
    ```
12. Mount the target root for replication, as follows:

    ```
    mount /dev/xscsi/pci01.03.0-1/target2/lun0/part3 /mnt
    ```
13. Replicate from the source root drive to the target using the `xfsdump`(8) command piped to `xfsrestore`(8) command, as follows:

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

    ```
    rm /mnt/etc/sysconfig/networking/eth0_persist
    ```
15. Unmount the target root filesystem, as follows:

    ```
    umount /mnt
    ```

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

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

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

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

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

Setting up `quota` on the `root` File System

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

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

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

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

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

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

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

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

System Partitioning

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

1. "Overview", page 6
2. "Advantages of Partitioning", page 6
3. "Limitations of Partitioning", page 8
4. "Supported Configurations", page 8
5. "Installing Partitioning Software and Configuring Partitions", page 8
6. "Connecting the System Console to the Controller", page 19

Overview

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

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

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

Advantages of Partitioning

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

- "Create a Large, Shared-memory Cluster ", page 6
- "Provides Fault Containment", page 7
- "Allows Variable Partition Sizes", page 7
- "Provide High Performance Clusters", page 7

Create a Large, Shared-memory Cluster

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

Provides Fault Containment

Another reason for partitioning a system is fault containment. In most cases, a single partition can be brought down (because of a hardware or software failure, or as part of a controlled shutdown) without affecting the rest of the system. Hardware memory protections prevent any unintentional accesses to physical memory on a different partition from reaching and corrupting that physical memory. For current fault containment caveats, see "Limitations of Partitioning", page 8.

You can power off and "warm swap" a failing C-brick in a down partition while other partitions are powered up and booted. For information see "Adding or Replacing a PCI or PCI-X Card" in chapter 12, "Maintenance and Upgrade Procedures" in `SGI Altix 3000 User's Guide`.

Allows Variable Partition Sizes

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

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

Provide High Performance Clusters

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

Limitations of Partitioning

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

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

Supported Configurations

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

- Maximum number of partitions supported is 8
- Maximum partition size is 64 processors
- Maximum system size is 512 processors

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

Installing Partitioning Software and Configuring Partitions

To enable or disable partitioning software, see "Partitioning Software", page 9, to use the system partitioning capabilities, see "Partitioning Guidelines", page 9 and "Partitioning a System", page 10.

This section covers the following topics:

- "Partitioning Software", page 9
- "Partitioning Guidelines", page 9
- "Partitioning a System", page 10
- "Determining If a System is Partitioned", page 16
- "Accessing the Console on a Partitioned System", page 17

- "Unpartitioning a System", page 18

Partitioning Software

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

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

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

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

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

Partitioning Guidelines

Follow these guidelines when partitioning your system:

- A partition must be made up of one or more C—bricks (Each C—brick contains four processors). The number of C-bricks in your systems determines the number of partitions you can create. The number of partitions cannot exceed the number of C—bricks your system contains. The first C—brick in each partition must have an IX—brick attached to it via the XIO connection of the C—brick.

- You need at least as many IX—bricks for base IO as partitions you wish to use.
- Each partition needs to have an IX—brick with a valid system disk in it. Since each partition is a separate running system, each system disk should be configured with a different IP address/system name, and so on.
- Each partition must have a unique partition ID number between 1 and 63, inclusively.
- All bricks in a partition must be physically contiguous. The route between any two processors in the same partition must be contained within that partition, and not through any other partition. If the bricks in a partition are not contiguous, the system will not boot.
- Each partition must contain the following components:
  - At least one C—brick for system sizes of 64 C—bricks and below, or multiples of 4 C—bricks for system sizes of 65 C—bricks and above (minimum)
  - One IX-brick (minimum)
  - One root disk
  - One console connection

Partitioning a System

This section describes how to partition your system.

Procedure 1-3 Partitioning a System Into Four Partitions

To partition your system, perform the following steps :

1. Make sure your system can be partitioned. See "Partitioning Guidelines", page 9.
2. You can use the **Connect to System Controller** task of the SGIconsole Console Manager GUI to connect to the L2 controller of the system you want to partition. The L2 controller must appear as a node in the SGIconsole configuration. For information on how to use SGIconsole, see the *Console Manager for SGIconsole Administrator's Guide*.
3. Using the L2 terminal (l2term), connect to the L2 controller of the system you wish to partition. After a connection to the L2 controller, an L2> prompt appears, indicating that the L2 is ready to accept commands, for example:

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

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

---

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

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

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

---

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

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

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

---

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

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

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

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

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

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

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

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

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

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

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

---

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

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

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

Note: This may take up to 30 seconds.

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

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

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

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

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

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

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

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

For more information on accessing the console of a partition, see "Accessing the Console on a Partitioned System", page 17.

Procedure 1-4 Partitioning a System into Two Partitions

To partition your system, perform the following steps:

1. Perform steps 1 through 5 in Procedure 1-3, page 10.
2. To configure the system into two partitions, enter the following commands:

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

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

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

Note: This may take up to 30 seconds.

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

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

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

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

Determining If a System is Partitioned

Procedure 1-5 Determing If a System Is Partitioned

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

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

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

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

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

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

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

Accessing the Console on a Partitioned System

Procedure 1-6 Access the Console on a Partitioned System

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

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

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

   ---

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

   ```
   cranberry-192.168.11.92-L2>sel

        known system consoles (partitioned)

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

        current system console

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

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

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

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

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

Unpartitioning a System

Procedure 1-7 Unpartitioning a System

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

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

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

   ---

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

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

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

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

   The list of bricks no longer have a dot followed by a number in their name (see "Determining If a System is Partitioned", page 16).
5. To reset all of the bricks, enter the following command:

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

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

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

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

Connecting the System Console to the Controller

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

Making Array Services Operational

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

Procedure 1-8 Making Array Services Operational

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

1. To enable Array Services by default, perform the following:

   ```
   chkconfig array on
   ```

2. Add a user called guest, as follows:

```
adduser -u 81 -g 99 -s /bin/tcsh -c "array services guest" -d /dev/null guest
```

3. Add a guest group, as follows:

   Note: User guest does not have to be a member of group guest.

   ```
   groupadd -g 999 guest
   ```

4. Add the following line to the `/etc/services` file:

   ```
   sgi-arrayd 5434/tcp # SGI Array Services daemon
   ```

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

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

6. Make the following changes to your networking files as needed.

- Add `localhost` to your `/etc/hosts.equiv` file.
- Add every node in the cluster to each `/etc/hosts.equiv` file.

Note: If you have NIS installed, you can force a map lookup from `/etc/hosts.equiv` listing all the names if you prefer (for example, `+@equiv`).

In general, keeping the `hosts.equiv` file up-to-date helps users because they do not need to update their individual `.rhosts` files.

Manually, start Array Services as follows:

Note: Since you configured Array Services on in step 1, Array Services will be automatically started in the future.

```
/etc/init.d/array start
```

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

```
array who
```

You should see yourself listed.

Pluggable Authentication Modules

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

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

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

Chapter 2

System Operation

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

- "Booting a System", page 21
- "Halting the System", page 24
- "Getting Console Access", page 27
- "Handling a System Hang or Crash on Panic", page 28

Booting a System

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

Procedure 2-1 Booting a System

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

1. Obtain the system console as described in "Getting Console Access", page 27 if you are using SGIconsole or telnet to the L2 controller as described in "Connecting to the L2 Controller", page 26.
2. By default, when booting a menu of boot options appears. On a properly configured system (as shipped from the factory), you can boot directly to the Linux operating sytem. A system administrator can change the default using the boot mainenance menus, the extensible firmware interface (EFI) shell> command, or the `efibootmgr` command (see the `efibootmgr` options usage statement for more information).

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

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

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

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

Please select a boot option

    UnitedLinux
    ProPack
    Boot option maintenance menu


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

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

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

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

Main Menu. Select an Operation


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

        Manage BootNext setting
        Set Auto Boot TimeOut

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

        Cold Reset
        Exit
```

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

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

Boot From a File.  Select a Volume


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

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

```
    Device Path VenHw(D65A6B8C-71E5-4DF0-A909-F0D23000000099B40000002BB40000005AB4
EFI Shell version 1.02 [12.38]
Device mapping table
  fs0  : Pci(1|1)/Scsi(Pun0,Lun1)/HD(Part1,Sigg1)
  fs1  : Pci(1|1)/Scsi(Pun0,Lun2)/HD(Part1,Sigg2)
  blk0 : Pci(1|1)/Scsi(Pun0,Lun1)
  blk1 : Pci(1|1)/Scsi(Pun0,Lun1)/HD(Part1,Sigg1)
  blk2 : Pci(1|1)/Scsi(Pun0,Lun1)/HD(Part2,Sigg3)
  blk3 : Pci(1|1)/Scsi(Pun0,Lun1)/HD(Part3,Sigg4)
  blk4 : Pci(1|1)/Scsi(Pun0,Lun2)
  blk5 : Pci(1|1)/Scsi(Pun0,Lun2)/HD(Part1,Sigg2)
  blk6 : Pci(1|1)/Scsi(Pun0,Lun2)/HD(Part2,Sigg5)
  blk7 : Pci(1|1)/Scsi(Pun0,Lun2)/HD(Part3,Sigg6)
Shell>
```

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

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

```
fs0:
```

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

a. To boot the default kernel from the first boot disk, enter the following command at the prompt:

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

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

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

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

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

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

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

Halting the System

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

Procedure 2-2 Halting the System

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

1. Connect to the L2> controller by following the steps in Procedure 2-4, page 26.
2. Enter Ctrl-D to connect to the system console.
3. Enter the `halt` command.

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

Connecting to the L1 Controller

You can monitor the L1 controller status and error messages on the L1 controller's liquid crystal display (LCD) located on the front panel of the individual bricks. The L1 controller and L2 controller status and error messages can also be monitored at your system console. The system console allows you to monitor and manage your server or graphics system by entering L1 controller commands. You can also enter L2 controller commands to monitor and manage your system if your system has L2 controller hardware and a system console or if you are using an SGIconsole as your system console. For information on connecting to the system console, see "Getting Console Access", page 27. For detailed information on using the L2 controller software, see the *SGI L1 and L2 Controller Software User's Guide*

Procedure 2-3 Connecting to the L1 Controller

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

Connecting to the L2 Controller

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

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

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

Procedure 2-4 Connecting to the L2 Controller

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

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

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

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

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

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

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

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

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

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

For information on working with partitioned systems, see "System Partitioning ", page 5.

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

Getting Console Access

This section describes how to access a system console.

Procedure 2-5 Getting Console Access

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

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

2. To connect to a system L2 controller, perform the steps in Procedure 2-4, page 26.

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

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

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

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

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

   For information on working with partitioned systems, see "System Partitioning ", page 5.

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

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

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

Handling a System Hang or Crash on Panic

If the system panics or drops into the kernel debugger, a kdb> prompt appears on the system console. This section describes how to troubleshoot a system that hangs, that is, is unresponsive and does not respond to commands from the controller prompt or a system that appears to have crashed on a system panic.

Procedure 2-6 Handling a System Hang or Crash

1. Connect to the L2 controller via a serial port or Ethernet connection or SGIconsole. (see "Getting Console Access", page 27)

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

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

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

   a. Escape to the L2 controller via serial port or Ethernet connection by entering Ctrl-t.

   b. Select the L2 controller, as follows:

   ```
   ?-001-L2>l2
   ```

   c. Reset selections to defaults, as follows:

   ```
   ?-001-L2>sel reset
   console input: 001c11 console0
   console output: not filtered
   ```

   d. Make sure that the selection is correct, as follows:

   ```
   ?-001-L2>sel
   known system consoles (nonpartitioned)
   ```

```
   001c11 - L2 detected

   current system console

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

3. For partitioned systems only, select the desired partition, for example `1`, as follows (for nonpartitioned systems, go to step 4):

   ```
   ?-001-L2>sel p 1
   ```

4. Make sure that the destination is set up correctly by setting it to default, as follows:

   ```
   ?-001-L2>dest reset
   ```

5. For partitioned systems only, set the destination to the desired partition, for example `1`, as folllows (for nonpartitioned systems, go to step 6):

   ```
   ?-001-L2>p 1 dest
   ```

6. Determine which L1 controllers are configured and functional by using the `cfg` and `pwr` commands, as follows:

   ```
   ?-001-L2>cfg

           L2 163.154.17.66: - 001 (LOCAL)
           L1 163.154.17.66:0:0     - 001c11
           L1 163.154.17.66:0:1     - 002i01
           L1 163.154.17.66:0:5     - 001c14
   ```

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

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

If some L1 controllers are missing, reseat the Universal Serial Bus (USB) connections between the R-bricks and the L2 controller.

7. Determine whether the system has hung or crashed on panic, as follows:

   a. Use the `ping`(8) command to try to ping the system.

   b. Enter console mode by entering Ctrl-d.

      Press the Enter key several times and look for a response from the system. If you get a line feed, it means the kernel is still running. If the system is hung, you will typically see a message similar to the following:

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

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

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

   a. Escape to the L2 controller by entering Ctrl-t:

   b. Select the L2 controller, as follows:

      ```
      ?-001-L2>l2
      ```

   c. Record LED status, as follows:

      ```
      ?-001-L2>leds
              001c11:
              CPU 0A: 0x3c:   SAL calling OS_INIT
              CPU 0C: 0x3c:   SAL calling OS_INIT

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

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

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

      Record the link LED status; a missing link can cause a hang.

d. Record the port status, as follows:

```
?-001-L2>port
 001c11:
        Port Stat Remote Pwr Local Pwr  Link LED SW LED
        ---- ---- ---------- ---------- -------- --------
           A 0x0f       okay       okay       on       on
           B 0x0f       okay       okay       on       on
           C 0x0f       okay       okay       on       on
           D 0x02       none       okay      off      off
        001c14:
        Port Stat Remote Pwr Local Pwr  Link LED SW LED
        ---- ---- ---------- ---------- -------- --------
           A 0x0f       okay       okay       on       on
           B 0x02       none       okay      off      off
           C 0x0f       okay       okay       on       on
           D 0x02       none       okay      off      off
```

9. Record information from KDB, as follows:

   a. Return to console mode by entering Ctrl-d.

   b. If the system is not already at the kdb> prompt, enter the system debugger by entering the following command at the system console (Hit the escape key (Esc) and then enter KDB all in caps).

   ```
   l2> Esc KDB
   ```

   c. At the kdb> prompt, enter the following:

   ```
   [0]kdb> sn2kdb
           [0]kdb> pod
           0 000: POD SysCt (RT) Cac> error a
           0 000: POD SysCt (RT) Cac> exit
   ```

   This produces much output.

10. Generate an error dump from the L3 controller, as follows:

   a. From the L3 controller, enter the following:

   errdmp -m *system name*e

b. Alternatively, if the name of the L2 is not of the form l2-*systemname* enter the following:

   `errdmp --l2` *L2 IP address*

   A file is created in the `/usr/people/sgidiag/sn2/dumps` directory.

11. Use the Linux Kernel Crash Dump (LKCD) comand to take a crash dump, as follows:

    a. Leave KDB as follows:

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

    b. Enter Esc SYSc (the Escape key followed by "SYS" and then "c" within 5 seconds) to initiate a system dump. You should see output similar to the following output:

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

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

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

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

12. If the system does not respond to the SysRq escape sequence, perform the following:

    a. Perform an NMI (nonmaskable interrupt), as follows:

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

    b. Capture the field replaceable unit (FRU) information, as follows:

    ```
    l2> Ctrl-T fru capture
    ```

c. Print the FRUinformation, as follows:

```
Ctrl-t
?-001-L2>fru print
```

d. If the system dropped into KDB when the NMI was performed, follow the instructions in Step 9 and then peform a system reset, as follows:

```
?-001-L2>reset
```

When the system comes back up, the error records are saved in `/var/log/messages` and written to the console.

Chapter 3

Performance Tuning

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

For a list of available references, see “Related Publications” in the preface of this manual.

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). Some of the information you can obtain is, as follows:

- `/proc/cpuinfo`

  Displays processor information, one entry per processor. You can 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, and so on.

- `/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, and so on. 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 are not 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`(1) command, which returns the kernel version and other machine information similar to the following:

```
[user@ferrari ~]$ uname -a
Linux ferrari.americas.sgi.com 2.4.19-sgi212r2 #1 SMP Mon Mar 31 23:59:51 PST 2003 ia64 unknown
```

In addition, the `topology` command is a Bourne shell script that uses information in `/dev/hw` to display system configuration information, similar to the following:

```
[user@ferrari ~]$ topology

Machine ferrari.americas.sgi.com has:
8 cpu’s
4 memory nodes

The cpus are:
cpu 0 is /dev/hw/module/001c05/slab/0/node/cpubus/0/a
cpu 1 is /dev/hw/module/001c05/slab/0/node/cpubus/0/c
cpu 2 is /dev/hw/module/001c05/slab/1/node/cpubus/0/a
cpu 3 is /dev/hw/module/001c05/slab/1/node/cpubus/0/c
cpu 4 is /dev/hw/module/001c08/slab/0/node/cpubus/0/a
cpu 5 is /dev/hw/module/001c08/slab/0/node/cpubus/0/c
cpu 6 is /dev/hw/module/001c08/slab/1/node/cpubus/0/a
cpu 7 is /dev/hw/module/001c08/slab/1/node/cpubus/0/c
```

```
The nodes are:
node 0 is /dev/hw/module/001c05/slab/0/node
node 1 is /dev/hw/module/001c05/slab/1/node
node 2 is /dev/hw/module/001c08/slab/0/node
node 3 is /dev/hw/module/001c08/slab/1/node

The topology is defined by:
/dev/hw/module/001c05/slab/0/node/link/1 is /dev/hw/module/001c05/slab/1/node
/dev/hw/module/001c05/slab/0/node/link/2 is /dev/hw/module/001c08/slab/0/node
/dev/hw/module/001c05/slab/1/node/link/1 is /dev/hw/module/001c05/slab/0/node
/dev/hw/module/001c05/slab/1/node/link/2 is /dev/hw/module/001c08/slab/1/node
/dev/hw/module/001c08/slab/0/node/link/1 is /dev/hw/module/001c08/slab/1/node
/dev/hw/module/001c08/slab/0/node/link/2 is /dev/hw/module/001c05/slab/0/node
/dev/hw/module/001c08/slab/1/node/link/1 is /dev/hw/module/001c08/slab/0/node
/dev/hw/module/001c08/slab/1/node/link/2 is /dev/hw/module/001c05/slab/1/node

/dev/hw/module/001c05/slab/0/node/xtalk/0/link -> /dev/hw/module/001c05/slab/0/IXbrick
```

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.`ecc` compiler.

Single Processor Code Tuning

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

- Get the expected answers and then tune performance. For details, see "Getting the Correct Results", page 38.
- Use existing tuned code, such as that found in math libraries and scientific library packages. For details, see "Using Tuned Code", page 44.
- Determine what needs tuning. For details, see "Determining Tuning Needs", page 45.
- Use the compiler to do the work. For details, see "Using Compiler Options Where Possible", page 45.

- Tune cache performance. For details, see "Tuning the Cache Performance", page 46.
- Set environment variables to enable higher-performance memory management mode. For details, see "Managing Memory", page 47.
- Change stack and data segments. For details, see "Using the `chatr` Command to Change Stack and Data Segments", page 48.

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`(1).

  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: 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: The 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 `idb` program is the the Intel debugger. This is a fully symbolic debugger for Linux platforms. 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 systems with Itanium processors.

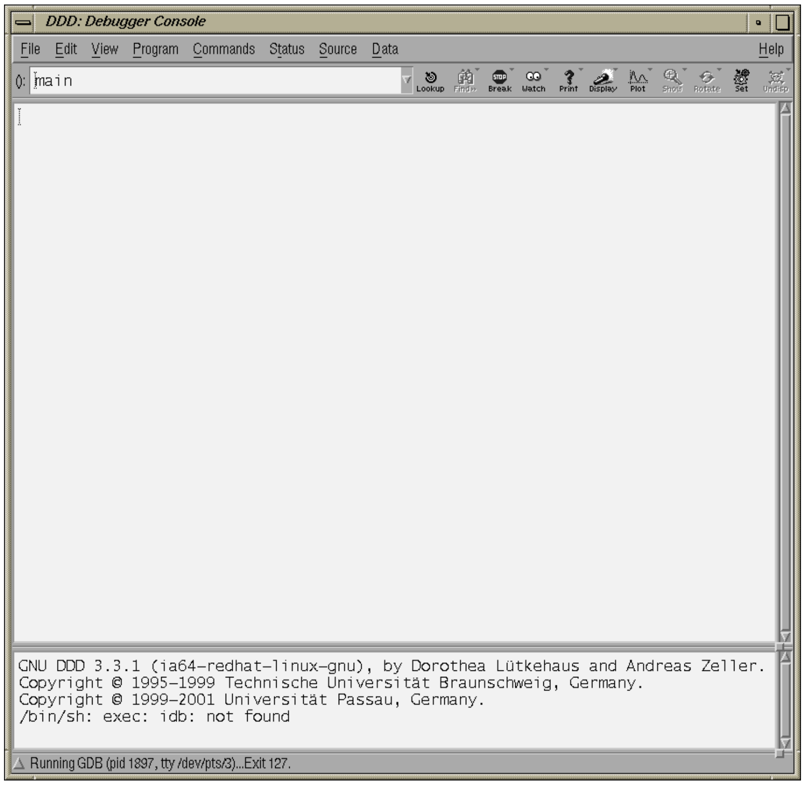
  Running `idb` with the `-gdb` option on the shell command line provides `gdb`-like user commands and debugger output.

- `ddd`

A graphical user interface (GUI) to a command line debugger. It supports `gdb` and `idb`. For details about usage, see "Using the `ddd` Tool".

Using the ddd Tool

The DataDisplayDebugger `ddd`(1) tool is a GUI to an arbitrary command line debugger as shown in Figure 3-1, page 40. To instantiate `ddd`, use the `--debugger` option to specify the debugger used (for example, `--debugger "idb"`). The default debugger used is `gdb`.



```
GNU DDD 3.3.1 (ia64-redhat-linux-gnu), by Dorothea Lütkehaus and Andreas Zeller.
Copyright © 1995-1999 Technische Universität Braunschweig, Germany.
Copyright © 1999-2001 Universität Passau, Germany.
/bin/sh: exec: idb: not found
```

Running GDB (pid 1897, tty /dev/pts/3)...Exit 127.

Figure 3-1 DataDisplayDebugger(`ddd`)(1)

When the debugger is loaded the DataDisplayDebugger screen appears divided into panes that show the following information:

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

You can use the **View** menu to switch these panes on and off.

Some commonly used commands, such as run, step, next, continue, kill are found off the **Program** pull-down menu and command history, previous, next, find forward, find backward, quit search, and so on are found off the **Commands** pull-down menu. 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`(1) 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 a PostScript file by using the **File > 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.

Procedure 3-1 Using `ddd`(1) with MPI Jobs

To use the `ddd` tool with an MPI job, perform the following:

1. Compile your program `-g` option similar to the following example:

   ```
   % cc -g -o hello hello.c -lmpi
   ```

2. In a window, set the start and pause environment variables, as follows:

   ```
   % setenv MPI_SLAVE_DEBUG_ATTACH 0
   ```

   The 0 argument is is the *rank* used by MPI to refer to the logical process number in the job. A 64 processor job has ranks 0-63. A program can query what its rank and perform particular tasks, accordingly.

   Then, run a command similar to the following:

   ```
   % mpirun -v -np 2 hello
   ```

The output similar to the following appears:

```
MPI: libxmpi.so 'SGI MPI 4.4pre MPT 1.9pre  08/04/03 11:37:02'
MPI: libmpi.so  'SGI MPI 4.4pre MPT 1.9pre  08/04/03 11:32:58'
MPI: MPI_MSGS_MAX = 524288
MPI: MPI_BUFS_PER_PROC= 32
MPI rank 0 sleeping for 20 seconds while you attach the debugger.
```

In this example, the `hello` program simply runs N copies, each reporting their own rank. It happens that rank 1 was the first to start up and report the instructions for how to attach the debugger to the job.

Use can use one or the other of the following debugger tools, as follows:

```
% gdb /proc/29911/exe 29911
```

or

```
% idb -pid 29911 /proc/29911/ex
```

3. In a second window, set your DISPLAY environment variable and then run the `ddd` command with the name of your program and the program's PID, as follow:

   `% setenv DISPLAY` ***display:0.0***

   then

   ```
   % ddd hello 29911
   ```

4. From the **View** pull-down menu, select **Machine Code Window**.

For detailed information on using the DataDisplayDebugger (DDD), see the *Debugging with DDD User's Guide and Reference Manual* available at the following location:

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

For more information on `gdb` and `ddd`, see the `gdb`(1) and `ddd`(1) man pages, respectively.

Managing Heap Corruption Problems

Two methods can be used to check for heap corruption problems in programs that use the `glibc` library `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. For more information on the malloc and free memory management routines, see the malloc(3) and free(1) man pages, respectively.

Electric Fence is a malloc debugger for Linux and UNIX. It stops your program on the exact instruction that overruns (or underruns) a malloc() memory buffer and displays the source-code line that causes the bug. 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. For more information, see the `efence`(3) man page.

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, the `glibc libm` routines are linked in first.

You can find documentation for MKL and VML at the following website:

`http://intel.com/software/products/perflib/index.htm?iid=ipp_home+software_libraries&`

Determining Tuning Needs

To determine what points in your code might benefit from tuning, use the following tools:

`time`	Use this command to obtain an overview of user, system, and elapsed time. For more information, see the `time`(1) man page.
`gprof`	Use this tool to obtain an execution profile of your program (a `pcsamp` profile). Use the `-p` compiler option to enable `gprof` use. For more information, see the `gprof`(1) man page.
VTune	This Intel performance monitoring tool is a Linux-server, Windows-client application. It supports remote sampling on all Itanium and Linux systems.
`pfmon`	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 more information, see the `pfmon`(1) man page.

For information about VTune and `pfmon`, see "Profilers and Performance Tools".

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:

`-ftz`	Flushes underflow to zero to avoid kernel traps. Enabled by default at `-O3` optimization.
`-fno-alias`	Assumes no pointer aliasing. Other aliasing options include `-ansi_alias` and `-fno_fnalias`. Note that alias assertions may generate incorrect code.
`-ip`	Generates single file, interprocedural optimization.
`-ipo`	Generates multifile, interprocedural optimization.

`-O3`	Enables `-O2` 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

The `pfmon` software is a performance monitoring tool that provides a user-level interface to the Performance Monitoring Unit (PMU) built into the Itanium processor chip. `pfmon` can be used to count or sample PMU events for unmodified binaries in its per-process mode and it can also be used to run system-wide monitoring sessions. The `pfmon` tool can monitor activities happening at the user and/or kernel level for both type of sessions. These counts and samples can be used to optimize program performance on the IPF processor. For more information on `pfmon`, see the `pfmon`(1) man page.

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

- 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_CANCELS1_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, as follows:

```
% pfmon -e CPU_CYCLES,L2_OZQ_CANCELS1_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 the `glibc` library `malloc`(3)/`free`(1) 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 the following location:

`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` library `malloc` command internally, these environment variables are also applicable in Fortran codes using, for example, Cray pointers with `malloc`(3)/`free`(1). But they do not work for Fortran 90 allocatable arrays, which are managed directly through Fortran library calls.

Using the `chatr` Command to Change Stack and Data Segments

The `chatr`(1) 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 `chatr` command to make these programs work correctly, regardless of the value of `vm.executable_stacks`.

For more detailed information about usage, see the `chatr`(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", page 37.

Multiprocessor tuning consists of the following major steps:

- Choose the parallelization methodology for your code. For details, see "Parallelizing Your Code", page 49.
- Analyze your code to make sure it is parallelizing properly. For details, see "Solving Bottlenecks in Code", page 50.

- Check to determine if false sharing exists. For details, see "Fixing False Sharing", page 51.
- Tune for data placement. For details, see "Using `dplace` and `runon`", page 52.
- Use environment variables to assist with tuning. For details, see "Environment Variables for Performance Tuning", page 53.

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

- Use the Message Passing Interface (MPI) from the SGI Message Passing Toolkit (MPT). MPI is optimized and more scalable for SGI Altix 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`(1) 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" 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 by using the PCP tools (for example, `pmval kernel.percpu.cpu.user`) or by using Performance Co-Pilot 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/cpu`*X* 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".

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.

Determining the Occurence of False Sharing

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

- Use the performance monitor to look at output from `pfmon`(1) and the `BUS_MEM_READ_BRIL_SELF` and `BUS_RD_INVAL_ALL_HITM` events.
- Use `pfmon` to check `DEAR` events to track common cache lines.
- Use the Performance Co-Pilot `pmshub` utility to monitor cache traffic and CPU utilization. For more information, see the `pmshub`(1) man page.

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

Fixing False Sharing

If false sharing is a problem on your system, 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. For more information, see the `dplace`(1) man page.

The `runon` command restricts execution to the listed set of CPUs; however, processes are still free to move among listed CPUs. For more information, see the `runon`(1) man page.

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. For more information on the `dplace` command, see the `dplace`(1) man page.

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

For information about the `dplace` command and its usage, see the `dplace`(1) man page and "`dplace`", page 65 in Chapter 4, "NUMA Tools", page 59.

Environment Variables for Performance Tuning

You can use several different 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.

f 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 stack size limit.

Profiling and Analyzing Your Code

Several tools are available to help determine areas for performance improvements. The following sections describe some of those tools and covers the following topics:

- "Profling with `pfmon`", page 54
- "Using the `profile.pl` Script", page 55
- "Using VTune for Remote Sampling ", page 56
- "Using GuideView", page 56

Profling 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 session 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 four events at a time.

To see a list of available options, use the `pfmon -help` command or see the `pfmon`(1) man page.

Using 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; enter `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.

For more information on these program profiling tools, see `profile.pl`(1), `analyze.pl`(1), and `makemap.pl`(1) man pages.

`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: The VTune tool may not be available for this release. Consult your release notes to its availability.

Using GuideView

`GuideView` is a graphical tool that presents a window into the performance details of the parallel execution of a program. `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 on your system.

`GuideView` uses an intuitive, color-coded display of parallel performance bottlenecks which helps pinpoint performance anomalies. It graphically illustrates the activity of each processor 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 used 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
pcpu 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:l

- 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.

You can find more detailed information about these products, at the following location:

```
http://developer.intel.com/software/products/threading
```

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.

NUMA Tools

This chapter describes the `dlook`(1) and `dplace`(1) tools that you can use to improve the performance of processes running on your SGI nonuniform memory access (NUMA) machine. You can use `dlook`(1) to find out where in memory the operating system is placing your application's pages and how much system and user CPU time it is consuming. You can use the `dplace`(1) command to bind a related set of processes to specific CPUs or nodes to prevent process migration. This can improve the performance of your application since it increases the percentage of memory accesses that are local.

This chapter covers the following topics:

`dlook`

The `dlook`(1) command allows you to display the memory map and CPU usage for a specified process as follows:

```
dlook [-a] [-c] [-h] [-l] [-o outfile] [-s secs] command [command-args]
dlook [-a] [-c] [-h] [-l] [-o outfile] [-s secs] pid
```

For each page in the virtual address space of the process, `dlook`(1) prints the following information:

- The object that owns the page, such as a file, SYSV shared memory, a device driver, and so on.
- The type of page, such as random access memory (RAM), FETCHOP, IOSPACE, and so on.
- If the page type is RAM memory, the following information is displayed:
  - Memory attributes, such as, SHARED, DIRTY, and so on

   - The node on which the page is located
   - The physical address of the page (optional)

- Optionally, the `dlook`(1) command also prints the amount of elapsed CPU time that the process has executed on each physical CPU in the system.

Two forms of the `dlook`(1) command are provided. In one form, `dlook` prints information about an existing process that is identified by a process ID (PID). To use this form of the command, you must be the owner of the process or be running with root privilege. In the other form, you use `dlook` on a command you are launching and thus are the owner.

The `dlook`(1) command accepts the following options:

`-a`	Shows the physical addresses of each page in the address space.
`-c`	Shows the elapsed CPU time, that is how long the process has executed on each CPU.
`-h`	Explicitly lists holes in the address space.
`-l`	Shows libraries.
`-o`	Outputs the file name. If not specified, output is written to stdout.
`-s`	Specifies a sample interval in seconds. Information about the process is displayed every second (`secs`) of CPU usage by the process.

An example for the `sleep` process with a PID of 4702 is as follows:

Note: The output has been abbreviated to shorten the example and bold headings added for easier reading.

```
dlook 4702

Peek:  sleep
Pid: 4702       Thu Aug 22 10:45:34 2002

Cputime by cpu (in seconds):
                  user    system
  TOTAL          0.002     0.033
  cpu1           0.002     0.033

Process memory map:
  2000000000000000-2000000000030000 r-xp 0000000000000000 04:03 4479 /lib/ld-2.2.4.so
```

```
      [2000000000000000-200000000002c000]        11 pages on node   1  MEMORY|SHARED

2000000000030000-200000000003c000 rw-p 0000000000000000 00:00 0
      [2000000000030000-200000000003c000]         3 pages on node   0  MEMORY|DIRTY

                                    ...

2000000000128000-2000000000370000 r-xp 0000000000000000 04:03 4672       /lib/libc-2.2.4.so
      [2000000000128000-2000000000164000]        15 pages on node   1  MEMORY|SHARED
      [2000000000174000-2000000000188000]         5 pages on node   2  MEMORY|SHARED
      [2000000000188000-2000000000190000]         2 pages on node   1  MEMORY|SHARED
      [200000000019c000-20000000001a8000]         3 pages on node   1  MEMORY|SHARED
      [20000000001c8000-20000000001d0000]         2 pages on node   1  MEMORY|SHARED
      [20000000001fc000-2000000000204000]         2 pages on node   1  MEMORY|SHARED
      [200000000020c000-2000000000230000]         9 pages on node   1  MEMORY|SHARED
      [200000000026c000-2000000000270000]         1 page  on node   1  MEMORY|SHARED
      [2000000000284000-2000000000288000]         1 page  on node   1  MEMORY|SHARED
      [20000000002b4000-20000000002b8000]         1 page  on node   1  MEMORY|SHARED
      [20000000002c4000-20000000002c8000]         1 page  on node   1  MEMORY|SHARED
      [20000000002d0000-20000000002d8000]         2 pages on node   1  MEMORY|SHARED
      [20000000002dc000-20000000002e0000]         1 page  on node   1  MEMORY|SHARED
      [2000000000340000-2000000000344000]         1 page  on node   1  MEMORY|SHARED
      [200000000034c000-2000000000358000]         3 pages on node   2  MEMORY|SHARED

                                         ....

20000000003c8000-20000000003d0000 rw-p 0000000000000000 00:00 0
      [20000000003c8000-20000000003d0000]         2 pages on node   0  MEMORY|DIRTY
```

The `dlook` command gives the name of the process (`Peek:  sleep`), the process ID, and time and date it was invoked. It provides total user and system CPU time in seconds for the process.

Under the heading **Process memory map**, the `dlook` command prints information about a process from the `/proc/`*pid*`/cpu` and `/proc/`*pid*`/maps` files. On the left, it shows the memory segment with the offsets below in decimal. In the middle of the output page, it shows the type of access, time of execution, the PID, and the object that owns the memory (in this case, `/lib/ld-2.2.4.so`). The characters `s` or `p` indicate whether the page is mapped as sharable (`s`) with other processes or is private (`p`). The right side of the output page shows the number of pages of memory consumed and on which nodes the pages reside. *Dirty memory* means that the memory has been modified by a user.

In the second form of the `dlook` command, you specify a command and optional command arguments. The `dlook` command issues an `exec` call on the command and passes the command arguments. When the process terminates, `dlook` prints information about the process, as shown in the following example:

`dlook` *date*

```
Thu Aug 22 10:39:20 CDT 2002
_______________________________________________________________________________
Exit:  date
Pid: 4680       Thu Aug 22 10:39:20 2002


Process memory map:
  2000000000030000-200000000003c000 rw-p 0000000000000000 00:00 0
        [2000000000030000-200000000003c000]         3 pages on node   3  MEMORY|DIRTY

  20000000002dc000-20000000002e4000 rw-p 0000000000000000 00:00 0
        [20000000002dc000-20000000002e4000]         2 pages on node   3  MEMORY|DIRTY

  2000000000324000-2000000000334000 rw-p 0000000000000000 00:00 0
        [2000000000324000-2000000000328000]         1 page  on node   3  MEMORY|DIRTY

  4000000000000000-400000000000c000 r-xp 0000000000000000 04:03 9657220    /bin/date
        [4000000000000000-400000000000c000]         3 pages on node   1  MEMORY|SHARED

  6000000000008000-6000000000010000 rw-p 0000000000008000 04:03 9657220    /bin/date
        [600000000000c000-6000000000010000]         1 page  on node   3  MEMORY|DIRTY

  6000000000010000-6000000000014000 rwxp 0000000000000000 00:00 0
        [6000000000010000-6000000000014000]         1 page  on node   3  MEMORY|DIRTY

  60000fff80000000-60000fff80004000 rw-p 0000000000000000 00:00 0
        [60000fff80000000-60000fff80004000]         1 page  on node   3  MEMORY|DIRTY

  60000fffffff4000-60000fffffffc000 rwxp ffffffffffffc000 00:00 0
        [60000fffffff4000-60000fffffffc000]         2 pages on node   3  MEMORY|DIRTY
```

If you use the `dlook` command with the `-s` *secs* option, the information is sampled at regular internals. The output for the command **`dlook -s 5 sleep 50`** is as follows:

```
Exit:  sleep
Pid: 5617       Thu Aug 22 11:16:05 2002


Process memory map:
  2000000000030000-200000000003c000 rw-p 0000000000000000 00:00 0
        [2000000000030000-200000000003c000]            3 pages on node   3  MEMORY|DIRTY

  2000000000134000-2000000000140000 rw-p 0000000000000000 00:00 0

  20000000003a4000-20000000003a8000 rw-p 0000000000000000 00:00 0
        [20000000003a4000-20000000003a8000]            1 page  on node   3  MEMORY|DIRTY

  20000000003e0000-20000000003ec000 rw-p 0000000000000000 00:00 0
        [20000000003e0000-20000000003ec000]            3 pages on node   3  MEMORY|DIRTY

  4000000000000000-4000000000008000 r-xp 0000000000000000 04:03 9657225    /bin/sleep
        [4000000000000000-4000000000008000]            2 pages on node   3  MEMORY|SHARED

  6000000000004000-6000000000008000 rw-p 0000000000004000 04:03 9657225    /bin/sleep
        [6000000000004000-6000000000008000]            1 page  on node   3  MEMORY|DIRTY

  6000000000008000-600000000000c000 rwxp 0000000000000000 00:00 0
        [6000000000008000-600000000000c000]            1 page  on node   3  MEMORY|DIRTY

  60000fff80000000-60000fff80004000 rw-p 0000000000000000 00:00 0
        [60000fff80000000-60000fff80004000]            1 page  on node   3  MEMORY|DIRTY

  60000ffffff4000-60000fffffffc000 rwxp ffffffffffffc000 00:00 0
        [60000ffffff4000-60000fffffffc000]            2 pages on node   3  MEMORY|DIRTY
```

You can run an message passing interface (MPI) job using the `mpirun` command and print the memory map for each thread, or redirect the ouput to a file, as follows:

Note: The output has been abbreviated to shorten the example and bold headings added for easier reading.

```
mpirun -np 8 dlook -o dlook.out ft.C.8

Contents of dlook.out:
_______________________________________________________________________________
Exit:  ft.C.8
Pid: 2306       Fri Aug 30 14:33:37 2002


Process memory map:
  2000000000030000-200000000003c000 rw-p 0000000000000000 00:00 0
        [2000000000030000-2000000000034000]           1 page  on node  21  MEMORY|DIRTY
        [2000000000034000-200000000003c000]           2 pages on node  12  MEMORY|DIRTY|SHARED

  2000000000044000-2000000000060000 rw-p 0000000000000000 00:00 0
        [2000000000044000-2000000000050000]           3 pages on node  12  MEMORY|DIRTY|SHARED
                                         ...
_______________________________________________________________________________
_______________________________________________________________________________
Exit:  ft.C.8
Pid: 2310       Fri Aug 30 14:33:37 2002


Process memory map:
  2000000000030000-200000000003c000 rw-p 0000000000000000 00:00 0
        [2000000000030000-2000000000034000]           1 page  on node  25  MEMORY|DIRTY
        [2000000000034000-200000000003c000]           2 pages on node  12  MEMORY|DIRTY|SHARED

  2000000000044000-2000000000060000 rw-p 0000000000000000 00:00 0
        [2000000000044000-2000000000050000]           3 pages on node  12  MEMORY|DIRTY|SHARED
        [2000000000050000-2000000000054000]           1 page  on node  25  MEMORY|DIRTY

                                           ...
_______________________________________________________________________________
_______________________________________________________________________________
Exit:  ft.C.8
Pid: 2307       Fri Aug 30 14:33:37 2002
```

```
Process memory map:
  2000000000030000-200000000003c000 rw-p 0000000000000000 00:00 0
        [2000000000030000-2000000000034000]            1 page  on node  30  MEMORY|DIRTY
        [2000000000034000-200000000003c000]            2 pages on node  12  MEMORY|DIRTY|SHARED

  2000000000044000-2000000000060000 rw-p 0000000000000000 00:00 0
        [2000000000044000-2000000000050000]            3 pages on node  12  MEMORY|DIRTY|SHARED
        [2000000000050000-2000000000054000]            1 page  on node  30  MEMORY|DIRTY
                                                    ...
_______________________________________________________________________________
_______________________________________________________________________________
Exit:  ft.C.8
Pid: 2308       Fri Aug 30 14:33:37 2002


Process memory map:
  2000000000030000-200000000003c000 rw-p 0000000000000000 00:00 0
        [2000000000030000-2000000000034000]            1 page  on node   0  MEMORY|DIRTY
        [2000000000034000-200000000003c000]            2 pages on node  12  MEMORY|DIRTY|SHARED

  2000000000044000-2000000000060000 rw-p 0000000000000000 00:00 0
        [2000000000044000-2000000000050000]            3 pages on node  12  MEMORY|DIRTY|SHARED
        [2000000000050000-2000000000054000]            1 page  on node   0  MEMORY|DIRTY
                                                   ...
```

For more information on the `dlook` command, see the `dlook` man page.

dplace

The `dplace` command allow you to control the placement of a process onto specified CPUs as follows:

`dplace [-c` *cpu_numbers*`] [-s` *skip_count*`] [-n` *process_name*`]`
`[-x` *skip_mask*`] [-p` *placement_file*`] command [`*command-args*`]`

`dplace -q`

Scheduling and memory placement policies for the process are set up according to `dplace` command line arguments.

By default, memory is allocated to a process on the node on which the process is executing. If a process moves from node to node while it running, a higher percentage of memory references are made to remote nodes. Because remote accesses typically have higher access times, process performance can be diminished.

You can use the `dplace` command to bind a related set of processes to specific CPUs or nodes to prevent process migrations. In some cases, this improves performance since a higher percentage of memory accesses are made to local nodes.

Processes always execute within a CpuMemSet. The CpuMemSet specifies the CPUs on which a process can execute. By default, processes usually execute in a CpuMemSet that contains all the CPUs in the system (for detailed information on CpuMemSets, see the *Linux Resource Administration Guide*).

The `dplace` command invokes a kernel hook (that is, a process aggregate or PAGG) to create a placement container consisting of all the CPUs (or a or a subset of CPUs) of the CpuMemSet. The `dplace` process is placed in this container and by default is bound to the first CPU of the CpuMemSet associated with the container. Then `dplace` invokes `exec` to execute the command.

The command executes within this placement container and remains bound to the first CPU of the container. As the command forks child processes, they inherit the container and are bound to the next available CPU of the container.

If you do not specify a placement file, `dplace` binds processes sequentially in a round-robin fashion to CPUs of the placement container. For example, if the current CpuMemSet consists of physical CPUs 2, 3, 8, and 9, the first process launched by `dplace` is bound to CPU 2. The first child process forked by this process is bound to CPU 3, the next process (regardless whether it is forked by parent or child) to 8, and so on. If more processes are forked than there are CPUs in the CpuMemSet, binding starts over with the first CPU in the CpuMemSet.

For more information on `dplace`(1) and examples of how to use the command, see the `dplace`(1) man page.

The `dplace`(1) command accepts the following options:

`-c` *cpu_numbers*	The *cpu_numbers* variable specifies a list of CPU ranges, for example: "-c1", "-c2-4", "-c1, 4-8, 3". CPU numbers are **not** physical CPU numbers. They are logical CPU numbers that are relative to the CPUs that are in the set of allowed CPUs as specified by the current CpuMemSet or `runon`(1) command. CPU numbers start at 0. If this option is not specified, all CPUs of the

	current CpuMemSet are available. Note that a previous `runon` command may be used to restrict the available CPUs.
`-s` *skip_count*	Skips the first *skip_count* processes before starting to place processes onto CPUs. This option is useful if the first *skip_count* processes are “shepherd" processes that are used only for launching the application. If *skip_count* is not specified, a default value of 0 is used.
`-n` *process_name*	Only processes named *process_name* are placed. Other processes are ignored and are not explicitly bound to CPUs. **Note:** The *process_name* argument is the basename of the executable.
`-x` *skip_mask*	Provides the ability to skip placement of processes. The `skip_mask` argument is a bitmask. If bit N of `skip_mask` is set, then the N+1th process that is forked is not placed. For example, setting the mask to 6 causes the second and third processes from being placed. The first process (the process named by the `command`) will be assigned to the first CPU. The second amd third processes are not placed. The fourth process is assigned to the second CPU, and so on. This option is useful for certain classes of threaded applications that spawn a few helper processes that typically do not use much CPU time. **Note:** Intel OpenMP applications currently should be placed using the `-x` option with a `skip_mask` of *6* (`-x`*6*). This could change in future versions of OpenMP.
`-p` *placement_file*	Specifies a placement file that contains additional directives that are used to control process placement. (Not yet implemented).
`command` [*command-args*]	Specifies the command you want to place and its arguments.

`-q`	Lists the global count of the number of active processes that have been placed (by `dplace`) on each CPU in the current cpuset. Note that CPU numbers are logical CPU numbers within the cpuset, **not** physical CPU numbers.

Example 4-1 Using `dplace` command with MPI Programs

You can use the `dplace` command to improve placement of MPI programs on NUMA systems and verify placement of certain data structures of a long running MPI program by running a command such as the following:

```
mpirun -np 64 /usr/bin/dplace -s1 -c 0-63 ./a.out
```

You can then use the `dlook`(1) command to verify placement of certain data structures of long running MPI program by using the `dlook` command in another window on one of the slave thread PIDs to verify placement. For more information on using the `dlook` command, see "`dlook`", page 59 and the `dlook`(1) man page.

Example 4-2 Using `dplace` command with OpenMP Programs

To run an OpenMP program on logical CPUs 4 through 7 within the current CpuMemSet, perform the following:

```
efc -o prog -openmp -O3 program.f
setenv OMP_NUM_THREADS 4
dplace -x6 -c4-7 ./prog
```

The `dplace`(1) command has a static load balancing feature so that you do not necessarily have to supply a CPU list. To place `prog1` on logical CPUs 0 through 3 and `prog2` on logical CPUs 4 through 7, perform the following:

```
setenv OMP_NUM_THREADS 4
dplace -x6 ./prog1 &
dplace -x6 ./prog2 &
```

You can use the `dplace -q` command to display the static load information.

Example 4-3 Using `dplace` command with Linux Commands

The following examples assume that the command is executed from a shell running in a CpuMemSet consisting of physical CPUs 8 through 15.

Command	Run Location
`dplace -c2 date`	Runs the `date` command on physical CPU 10.

`dplace make linux`	Runs `gcc` and related processes on physical CPUs 8 through 15.
`dplace -c0-4,6 make linux`	Runs `gcc` and related processes on physical CPUs 8 through 12 or 14.
`runon 4-7 dplace app`	The `runon` command restricts execution to physical CPUs 12 through 15. The `dplace` sequentially binds processes to CPUs 12 through 15.

topology

The `topology`(1) command provides topology information about your system. Topology information is extracted from information in the`/dev/hw` directory. Unlike IRIX, in Linux the hardware topology information is implemented on a `devfs` filesystem rather than on a `hwgraph` filesystem. The `devfs` filesystem represents the collection of all significant hardware connected to a system, such as CPUs, memory nodes, routers, repeater routers, disk drives, disk partitions, serial ports, Ethernet ports, and so on. The `devfs` filesystem is maintained by system software and is mounted at `/hw` by the Linux kernel at system boot.

Applications programmers can use the `topology` command to help execution layout for their applications. For more information, see the `topology`(1) man page.

Output from the `topology` command is similar to the following: (Note that the following output has been abbreviated.)

```
% topology
Machine parrot.americas.sgi.com has:
64 cpu's
32 memory nodes
8 routers
8 repeaterrouters

The cpus are:
cpu 0 is /dev/hw/module/001c07/slab/0/node/cpubus/0/a
cpu 1 is /dev/hw/module/001c07/slab/0/node/cpubus/0/c
cpu 2 is /dev/hw/module/001c07/slab/1/node/cpubus/0/a
cpu 3 is /dev/hw/module/001c07/slab/1/node/cpubus/0/c
cpu 4 is /dev/hw/module/001c10/slab/0/node/cpubus/0/a
                         ...
The nodes are:
```

```
node 0 is /dev/hw/module/001c07/slab/0/node
node 1 is /dev/hw/module/001c07/slab/1/node
node 2 is /dev/hw/module/001c10/slab/0/node
node 3 is /dev/hw/module/001c10/slab/1/node
node 4 is /dev/hw/module/001c17/slab/0/node
                        ...
The routers are:
/dev/hw/module/002r15/slab/0/router
/dev/hw/module/002r17/slab/0/router
/dev/hw/module/002r19/slab/0/router
/dev/hw/module/002r21/slab/0/router
                        ...
The repeaterrouters are:
/dev/hw/module/001r13/slab/0/repeaterrouter
/dev/hw/module/001r15/slab/0/repeaterrouter
/dev/hw/module/001r29/slab/0/repeaterrouter
/dev/hw/module/001r31/slab/0/repeaterrouter
                       ...
The topology is defined by:
/dev/hw/module/001c07/slab/0/node/link/1 is /dev/hw/module/001c07/slab/1/node
/dev/hw/module/001c07/slab/0/node/link/2 is /dev/hw/module/001r13/slab/0/repeaterrouter
/dev/hw/module/001c07/slab/1/node/link/1 is /dev/hw/module/001c07/slab/0/node
/dev/hw/module/001c07/slab/1/node/link/2 is /dev/hw/module/001r13/slab/0/repeaterrouter
/dev/hw/module/001c10/slab/0/node/link/1 is /dev/hw/module/001c10/slab/1/node
/dev/hw/module/001c10/slab/0/node/link/2 is /dev/hw/module/001r13/slab/0/repeaterrouter
```

Installing NUMA Tools

To use the `dlook`(1), `dplace`(1), and `topology`(1) commands, you must load the `numatools` kernel module. Perform the following steps:

1. Configure the `numatools` kernel module on across system reboots by using the `chkconfig`(8) utility as follows:

   ```
   chkconfig --add numatools
   ```

2. To turn on `numatools`, enter the following command:

   ```
   /etc/rc.d/init.d/numatools start
   ```

This step will be done automatically for subsequent system reboots when `numatools` are configured on by using the `chkconfig`(8) utility.

The following steps are required to disable `numatools`:

1. To turn off `numatools`, enter the following:

   ```
   /etc/rc.d/init.d/numatools stop
   ```

2. To stop `numatools` from initiating after a system reboot, use the `chkconfig`(8) command as follows:

   ```
   chkconfig --del numatools
   ```

Chapter 5

Kernel Tunable Parameters on SGI ProPack Servers

This section identifies and describes the settings for kernel tunable parameters appropriate for large SGI ProPack servers. For a general description of Linux kernel tunable parameters, see Appendix A, "Linux Kernel Tunable Parameters", page 77.

CPU Scheduler `/proc/sys/sched` Directory

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

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

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

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

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

`/etc/sysconfig/dump` Directory

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

The following variables defined in this directory:

- `DUMP_ACTIVE`

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

- `DUMPDEV`

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

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

- `DUMPDIR`

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

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

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

- `DUMP_SAVE`

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

- `DUMP_LEVEL`

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

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

  Currently, the `DUMP_NONE`, `DUMP_HEADER` or `DUMP_ALL` variables are valid.

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

- DUMP_COMPRESS

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

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

- PANIC_TIMEOUT

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

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

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

Linux Kernel Tunable Parameters

The `sysctl`(8) command is used to modify kernel parameters at runtime. These parameters are those listed under the `/proc/sys/****` file system. You can use the `sysctl`(8) command to both read and write system control data.

This information about Linux kernel tunable parameters is included in your Linux release and can be found in the following file: `/usr/src/linux-2.4.19/Documentation/sysctl` file.

This chapter describes the following topics:

- "`/proc/sys/kernel` Parameters", page 77
- "`/proc/sys/fs` Parameters", page 81
- "`/proc/sys/vm` Parameters", page 84

`/proc/sys/kernel` Parameters

This section describes the files in the `/proc/sys/kernel` directory that can be used to tune and monitor filesystem performance of the Linux operating system.

Note: Changing the values of the parameters in these files can adversely affect the performance of your system. Make sure you have read the appropriate documentation before applying these adjustements.

This section describes the following kernel parameters contained in the following files:

- "`acct`", page 78
- "`ctrl-alt-del`", page 78
- "`domainname` and `hostname`", page 79
- "`osrelease`, `ostype`, and `version`", page 79
- "`overflowgid` and `overflowuid`", page 79
- "`panic`", page 80

`acct`

The `acct` file defines the high-water and low-water frequency.

If BSD-style process accounting is enabled these values control its behavior. If free space on filesystem where the log lives goes below the *lowater* percentage, the accounting suspends. If free space gets above the *highwater* percentage accounting resumes. The *frequency* parameter determines how often do we check the amount of free space (value is in seconds).

The **default:** is 4 2 30

That is, suspend accounting if there is less than or equal to 2% free space; resume accounting if there is greater than or equal to 4% free space; consider information about amount of free space valid for 30 seconds.

`ctrl-alt-del`

When the value in this file is 0, `ctrl-alt-del` is trapped and sent to the `init`(1) program to handle a graceful restart. When, however, the value is greater than 0, the operating system's reaction to a Vulcan Nerve Pinch (The keyboard combination that forces a soft boot or jump to ROM monitor on machines that support such a feature) will be an immediate reboot, without even synchronizing its dirty buffers.

Note: When a program (like `dosemu`) has the keyboard in **raw** mode, the `ctrl-alt-del` is intercepted by the program before it ever reaches the kernel tty layer, and it is up to the program to decide what to do with it.

`domainname` and `hostname`

These files can be used to set the NIS/YP domainname and the hostname of your system in exactly the same way as the commands `domainname` and `hostname`, for example:

```
# echo "darkstar" > /proc/sys/kernel/hostname
# echo "mydomain" > /proc/sys/kernel/domainname
```

This has the same effect as the following:

```
# hostname "darkstar" > /proc/sys/kernel/hostname
# domainname "mydomain" > /proc/sys/kernel/domainname
```

Note, however, that the classic darkstar.frop.org has the hostname `darkstar` and DNS (Internet Domain Name Server) domainname `frop.org`, not to be confused with the NIS (Network Information Service) or YP (Yellow Pages) domainname. These two domain names are in general different. For a detailed discussion see the `hostname`(1) man page.

`osrelease`, `ostype`, and `version`

The files `osrelease` and `ostype` parameters displayed below are self-evident. The version parameter needs a little more clarification, however. The number 5 means that this is the fifth kernel built from this source base and the date behind it indicates the time the kernel was built. The only way to tune these values is to rebuild the kernel.

```
# cat osrelease
2.1.88
# cat ostype
Linux
# cat version
#5 Wed Feb 25 21:49:24 MET 1998
```

`overflowgid` and `overflowuid`

These parameters allow you to change the value of the fixed UID and GID.

The **default** is 65534.

panic

The value in this file represents the number of seconds the kernel waits before rebooting on a panic. When you use the software watchdog, the recommended setting is 60 seconds.

printk

The four values in `printk` denote: `console_loglevel`, `default_message_loglevel`, `minimum_console_level`, and `default_console_loglevel`, respectively.

These values influence `printk()` behavior when printing or logging error messages.

- `console_loglevel`: messages with a higher priority than this are printed to the console.
- `default_message_level`: messages without an explicit priority are printed with this priority.
- `minimum_console_loglevel`: minimum (highest) value to which `console_loglevel` can be set.
- `default_console_loglevel`: default value for `console_loglevel`.

Note: The code in the `linux/kernel/printk.c` executable reveals that these variables are not put inside a structure, so their order in-core is not formally guaranteed and garbage values might occur when the compiler changes.

For more information on log levels, see the `syslog`(2) man page.

`rtsig-max` and `rtsig-nr`

The `rtsig-max` file can be used to tune the maximum number of POSIX real–time (queued) signals that can be outstanding in the system.

The `rtsig-nr` file shows the number of real-time signals currently queued.

`sg-big-buff`

This file shows the size of the generic SCSI (sg) buffer. You can not tune it just yet, but you could change it on compile time by editing the `/scsi/sg.h` include file and changing the value of `SG_BIG_BUFF`.

Typically, the value of this parameter is not changed.

`shmmax`

This value can be used to query and set the run time limit on the maximum shared memory segment size that can be created. Shared memory segments up to 1GBs are now supported in the kernel. This value defaults to `SHMMAX`

`tainted`

The tainted parameter is nonzero if the kernel has been tainted. These are numeric values which can be ORed together as follows:

1	A module with a non-General Public License (non-GPL) license has been loaded, this includes modules with no license. Set by `modutils` equal to or greater than the 2.4.9 release.
2	A module was force loaded by `insmod -f`. Set by modutils equal to or greater than the 2.4.9 release.

`/proc/sys/fs` Parameters

This section describes the files in the `/proc/sys/fs` directory that can be used to tune and monitor general system performance controlled by the Linux kernel.

Note: Changing the values of the parameters in these files can adversely affect the performance of your system. Make sure you have read the appropriate documentation before applying these adjustements.

This section describes the following kernel parameters:

- "`dentry-state`", page 82

- "dquot-max and dquot-nr", page 82
- "file-max and file-nr", page 82
- "inode-max, inode-nr, and inode-state", page 83
- "overflowgid and overflowuid", page 83
- "super-max and super-nr", page 84

dentry-state

The entries in the dentry-state are dynamically allocated and deallocated and nr_dentry is typically 0 all the time. Consequently, it is safe to assume that only nr_unused, age_limit and want_pages parameters are used. The nr_unused is not used. The age_limit parameter is the age in seconds after which dcache entries can be reclaimed when memory is short and the want_pages parameter is nonzero when shrink_dcache_pages() has been called and the dcache is not pruned yet.

The following structure is from /fs/dentry.c:

```
struct {
        int nr_dentry;
        int nr_unused;
        int age_limit;         /* age in seconds */
        int want_pages;        /* pages requested by system */
        int dummy[2];
} dentry_stat = {0, 0, 45, 0,};
```

dquot-max and dquot-nr

The dquot-max file shows the maximum number of cached disk quota entries. The file dquot-nr file shows the number of allocated disk quota entries and the number of free disk quota entries. If the number of free cached disk quotas is very low and you have an extremely large number of simultaneous system users, you might want to raise the limit.

file-max and file-nr

The kernel allocates file handles dynamically but, as of yet, it does not free them again.

The value in the `file-max` file denotes the maximum number of file handles that the kernel will allocate. If your receive man error messages about running out of file handles, you may want to increase this limit.

`inode-max`, `inode-nr`, and `inode-state`

Like file handles, the kernel allocates the inode structures dynamically, but, as of yet, it does not free them again.

The value in the `inode-max` file denotes the maximum number of inode handlers. This value should be 3 to 4 times larger than the value in the `file-max` file , since `stdin`, `stdout` and network sockets also need an inode structure to handle them. If you regularly run out of inodes, you need to increase this value.

The `inode-nr` file contains the first two items from `inode-state` file described in the following paragraphs.

The `inode-state` file contains three actual numbers and four placeholder numbers that currently do not affect inode structures. The actual numbers are, in order of appearance, `nr_inodes`, `nr_free_inodes` and `preshrink`.

The `nr_inodes` parameter stands for the number of inodes the system has allocated. This can be slightly more than `inode-max` because Linux allocates them a whole page at a time.

The `nr_free_inodes` parameter represents the number of free inodes and the preshrink `parameter` is nonzero when the `nr_inodes` value is greater than `inode-max` and the system needs to shorten the inode list instead of allocating more inodes.

`overflowgid` and `overflowuid`

Some filesystems only support 16-bit UIDs and GIDs, although in Linux UIDs and GIDs are 32 bits. When one of these filesystems is mounted with writes enabled, any UID or GID that would exceed 65535 is translated to a fixed value before being written to disk.

These `sysctl` parameters allow you to change the value of the fixed UID and GID.

The **default** is 65534.

super-max and super-nr

The super-max and super-nr numbers control the maximum number of superblocks and thus the maximum number of mounted filesystems the kernel can have. You only need to increase the super-max value if you need to mount more filesystems than the current value in the super-max parameter allows you to.

/proc/sys/vm Parameters

This section describes the files in the /proc/sys/vm directory that can be used to tune the operation of the virtual memory (VM) subsystem of the Linux kernel and the bdflush file also has some influence on disk usage.

Default values and initialization routines for most of these files can be found in mm/swap.c.

This section describes the following kernel parameters:

- "bdflush", page 84
- "buffermem", page 86
- "freepages", page 86
- "kswapd", page 86
- "overcommit_memory", page 87
- "max_map_count", page 88
- "page-cluster", page 88
- "pagecache", page 88
- "pagetable_cache", page 89

bdflush

This file controls the operation of the bdflush kernel daemon that flush dirty buffers back to disk. For more information on bdflush daemon, see the bdflush(8) man page. The source code to this structure can be found in linux/fs/buffer.c. It currently contains 9 integer values, of which 4 are actually used by the kernel.

The bdflush_param stucture from the /fs/buffer.c executable follows:

```
union bdflush_param {
        struct {
                int nfract;     /* Percentage of buffer cache dirty to
                                   activate bdflush */
                int dummy1;     /* old "ndirty" */
                int dummy2;     /* old "nrefill" */
                int dummy3;     /* unused */
                int interval;   /* jiffies delay between kupdate flushes */
                int age_buffer; /* Time for normal buffer to age */
                int nfract_sync;/* Percentage of buffer cache dirty to
                                   activate bdflush synchronously */
                int dummy4;     /* unused */
                int dummy5;     /* unused */
        } b_un;
        unsigned int data[N_PARAM];
} bdf_prm = {{30, 64, 64, 256, 5*HZ, 30*HZ, 60, 0, 0}};
```

The nfract parameter governs the maximum number of dirty buffers in the buffer cache. *Dirty* means that the contents of the buffer still have to be written to disk (as opposed to a clean buffer, which can just be forgotten about). Setting this to a high value means that the operating system can delay disk writes for a long time but it also means that it will have to do many I/O operations when memory becomes short. A low value will spread out disk I/O more evenly, at the cost of more frequent I/O operations.

The **default** value is 30%, the minimum is 0%, and the maximum is 100%.

The interval parameter is the minimum rate at which the kupdate routine will wake and flush the buffer. The value is expressed in *jiffies* (clock ticks), the number of jiffies per second is normally 100 (Alpha is 1024). Thus, x*HZ is *x* seconds.

The **default** value is 5 seconds, the minimum is 0 seconds, and the maximum is 600 seconds.

The age_buffer parameter governs the maximum time Linux waits before writing out a dirty buffer to disk. The value is in jiffies.

The **default** value is 30 seconds, the minimum is 1 second, and the maximum 6,000 seconds.

The `nfract_sync` parameter governs the percentage of buffer cache that is dirty before `bdflush` activates synchronously. This can be viewed as the hard limit before `bdflush` forces buffers to disk.

The **default** is 60%, the minimum is 0%, and the maximum is 100%.

buffermem

The three values in this file correspond to the values in the `buffer_mem` structure. It controls how much memory should be used for buffer memory. The percentage is calculated as a percentage of total system memory.

The values are as follows:

`min_percent`	This is the minimum percentage of memory that should be spent on buffer memory.
`borrow_percent`	Not used.
`max_percent`	Not used.

freepages

This file contains the values in the `freepages` structure. This structure contains three members: `min`, `low`, and `high`.

The values are as follows:

`freepages.min`	When the number of free pages in the system reaches this number, only the kernel can allocate more memory.
`freepages.low`	If the number of free pages gets below this point, the kernel starts swapping aggressively.
`freepages.high`	The kernel tries to keep up to this amount of memory free; if memory comes below this point, the kernel gently starts swapping in the hopes that it never has to do real aggressive swapping.

kswapd

The kernel swapout daemon is `kswapd`.

The numbers in this page correspond to the numbers in thefollowing structure:

```
struct pager_daemon {tries_base, tries_min, swap_cluster};
```

The `tries_base` and `swap_cluster` probably have the largest influence on system performance.

The values are as follows:

`tries_base`	The maximum number of pages that the `kswapd` daemon tries to free in one round is calculated from this number. Usually this number will be divided by 4 or 8 (see `mm/vmscan.c`), so it is not as big as it appears. When you need to increase the bandwidth to or from swap, you will want to increase this number.
`tries_min`	This is the minimum number of times that the `kswapd` daemon tries to free a page each time it is called. Basically, it is just there to make sure that the `kswapd` daemon frees some pages even when it is being called with minimum priority.
`swap_cluster`	This is the number of pages that the `kswapd` daemon writes in one turn. You want this large so that kswapd does it's I/O in large chunks and the disk does not have to seek often, but you do not want it to be too large since that would flood the request queue.

overcommit_memory

This value contains a flag that enables memory overcommitment. When this flag is 0, the kernel checks before each `malloc()` request to determine if there is enough remaining memory. If the flag is nonzero, the system pretends there is always enough memory.

This feature can be very useful because there are many programs that use the `malloc`() routine huge amounts of memory just-in-case it is needed and do not use much of it.

For more information, look at `vm_enough_memory`() routine in the `mm/mmap.c` file.

max_map_count

This file contains the maximum number of memory map areas a process may have. Memory map areas are used as a side-effect of calling the `malloc` routine, directly by `mmap` and `mprotect`, and also when loading shared libraries.

While most applications need less than a thousand maps, certain programs, particularly the malloc routine debuggers, may consume lots of them, for example, up to one or two maps per allocation.

page-cluster

The Linux virtual memory (VM) subsystem avoids excessive disk seeks by reading multiple pages on a page fault. The number of pages it reads is dependent on the amount of memory in your machine.

The number of pages the kernel reads in at once is equal to 2 ^ page-cluster. Values above 2 ^ 5 do not make much sense for swap because we only cluster swap data in 32-page groups.

pagecache

This file does exactly the same as the `buffermem` file, only this file controls the `page_cache` structure, and thus controls the amount of memory used for the page cache.

The page cache is used for three main purposes as follows:

- Caching `read`() data from files.
- Caching `mmap`() data and executable files.
- Swapping cache.

When your system is both deep in swap and high on cache, it probably means that a lot of the swapped data is being cached, making for more efficient swapping than possible with prior kernels.

`pagetable_cache`

The kernel keeps a number of page tables in a per-processor cache (this helps a lot on shared-memory multiprocessor (SMP) systems). The cache size for each processor will be between the low and the high value.

On a low-memory, single CPU system you can safely set these values to 0 so you do not waste the memory. On SMP systems, it is used so that the system can do fast page table allocations without having to acquire the kernel memory lock.

For large systems, the settings are probably OK. For normal systems, they will not have negative effects. For small systems, (less than 16MB RAM) it might be advantageous to set both values to 0.

Index