sgi®

SGI® UV™ Systems Linux® Configuration and Operations Guide

007–5629–004

Record of Revision

Version	Description
001	June 2010 Original publication.
002	October 2010 Updated to support the SGI Performance Suite 1.0 release.
003	February 2011 Updated to support the SGI Performance Suite 1.1 release.
004	May 2012 Updated to support the SGI Performance Suite 1.4 release.

Contents

Procedures

About This Guide

This guide is a reference document for people who manage the operation of SGI® UV™ 100, SGI® UV™ 1000 or SGI® UV™ 2000 systems. It explains how to perform general system configuration and operations under the Linux operating system used with SGI UV 100, SGI UV 1000, and SGI UV 2000 systems.

This manual applies to SGI Altix UV 100, UV 1000 and UV 2000 systems. For the SGI Altix UV 10 system, see the *SGI Altix UV 10 System User's Guide.* For SGI Altix 4000 series systems, see the *Linux Configuration and Operations Guide.* For more information on SGI® ICE™ 8200 and SGI® ICE™ 8400 systems, see the *SGI Management Center for ICE*, and their respective hardware guides. For SGI® ICE X™ systems, see the *SGI Management Center for ICE X* and the *SGI ICE X System Hardware User Guide.*

SGI Management Center (SMC) software running on the system management node (SMN) provides a robust graphical interface for system configuration, operation, and monitoring. For more information on the SMC, see *SGI Management Center System Administrator Guide.*

This manual contains the following chapters:

- Chapter 1, "SGI LK License Facility" on page 1
- Chapter 2, "Configuring Your System" on page 7
- Chapter 3, "System Operation" on page 23

Related Publications

For a list of manuals supporting SGI Linux releases and SGI online resources, see the *SGI Performance Suite 1.4 Start Here.*

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.

- You can view man pages by typing man *title* on a command line.

Conventions

The following conventions are used throughout this document:

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

Reader Comments

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

You can contact SGI in any of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.
- Send mail to the following address:

SGI
Technical Publications
46600 Landing Parkway
Fremont, CA 94538

SGI values your comments and will respond to them promptly.

Chapter 1

SGI LK License Facility

SGI Foundation Software 2.6 contains a licensing mechanism called LK. LK was developed by SGI for SGI products only. LK and the LK floating licenses facilities are described in this chapter.

Overview

The `lkd` program is the SGI LK license manager. It serves counted licenses for use by `localhost` and `remotehost`. Applications linked with LK and using counted licenses use the local `lkd` daemon to proxy requests out to the final daemon. Counted licenses are known by the `ssn` (system serial number) serving that key. The daemon maintains an `[ssn,ip or name,port]` pair so it can process LKN (LK network) requests with the remote daemon. The applications never talk to the remote daemon; they only send requests to the local daemon via a UNIX socket. The request is processed locally by the local daemon if it is its own `ssn` or proxy sent out by the request to the remote daemon. For usage information, see the `lkd`(8) man page.

This chapter covers the following topics:

- "LK License Considerations" on page 1
- "Obtaining Host Information" on page 2
- "Obtaining the License Keys from SGI" on page 3
- "Installing the License Keys" on page 3
- "Verifying the License Keys " on page 3
- "Setting Up the LK Floating Licenses `lkd` Daemon" on page 4
- "Using the LK Floating Licenses Facilities" on page 4

LK License Considerations

The LK licensing daemon (`lkd`) manages counted licenses. Therefore, it manages licenses with a count value greater than or equal to one (1). The LK license references the server serial number (`ssn`) of the server; this is the `lk_hostid` of the LK license manager. See the `lk`(5) man page for more information. For example:

```
# A counted ( 5 seats ) nodeless license served by ssn 0011432bc7b3
product=acme, version=1.0, \
   count=5@0011432bc7b3, begDate=0, \
   expDate=0, licenseID=0, \
   key=0iKOmyC0F9vFgVJqJBsIBBU/ZpwTYIoG
```

From an application license perspective using managed licenses, can run on the following:

- Every host

  This is called a nodeless managed license. This is the most commonly issued license.

- One host

  This is called a nodelock managed license.

In all cases, the server is locked to a specific `ssn` as indicated by the `count=xxx@ssn` notation.

Obtaining Host Information

When you order SGI Foundation Software 2.6, you will receive an entitlement ID. You must submit the system host ID, host name, and entitlement ID when requesting your permanent LK license key.

To obtain the host information for a server-capable administration node, execute the following command (you must have the `lkSGI` RPM from the SGI Foundation Software 2.6 installed):

```
/usr/sbin/lk_hostid
```

For example, the following shows that the serial number is `N0000302` and the license ID is `e000012e`:

```
server-admin % /usr/sbin/lk_hostid
N0000302 e000012e socket=8 core=16 processor=16
```

Obtaining the License Keys from SGI

To obtain your license keys, see information provided in your customer letter and the following web page:

http://www.sgi.com/support/licensing

Installing the License Keys

To install the license keys, copy them into the `/etc/lk/keys.dat` file. A sample `keys.dat` file is, as follows:

```
server-admin% cat keys.dat
product=XVM_PLEX_LINUX, version=5.000, count=0, \
begDate=1216648987, expDate=0, licenseID=e000012e, \
key=Ea5Yc+1vxqykBXc931CGMHv5cmOCSfns, \
info='XVM PLEX STD 5 M',attr='16 CPU', \
vendor='Silicon Graphics, Inc.',ref_id='070706'
```

Verifying the License Keys

You can use the `lk_verify -A` command to verify LK licenses. To see more output, use the `-v` option. For example:

```
server-admin% lk_verify -A -vvv
lk_check        All     All : total found=1

  1 /etc/lk/keys.dat:005        product=XVM_PLEX_LINUX, version=5.000, count=0, be
         expDate=0, licenseID=e000012e, key=Ea5Yc+1vxqykBXc931CGMHv5cmOCSfns, \
         info='XVM PLEX STD 5 M',attr='16 CPU', vendor='Silicon Graphics, Inc.', \
        ref_id='070706'
                Verdict:         SUCCESS. Nodelock.
                                 Available since 372 days on 21-Jul-2008 09:03:07.
                                 No End Date.
                Attribute 1 of 4 : info=XVM PLEX STD 5 M
                Attribute 2 of 4 : attr=16 CPU
                Attribute 3 of 4 : vendor=Silicon Graphics, Inc.
                Attribute 4 of 4 : ref_id=070706

lk_check        All     All : total matched=1
```

Setting Up the LK Floating Licenses `lkd` Daemon

In order for the LK floating licenses facilities to operate correctly, both the client and server need to have this common setup:

- The following configuration setting: `chkconfig lkd on`
- Share the same `/etc/lk/keys.dat` (or equivalent) either via distribution mechanism (`rsync`) or NFS, or similar
- Allow TCP/IP port 11106 traffic between the server and client hosts

The client must also define the license server "`ssn host_or_ip`" pair in either one of these locations:

- `/etc/lkd.map` (see `lkd.map`(5)
- And/or embedded in the license file itself in the form of:

  ```
  #! 080069010203 192.168.1.1
  ```

- And/or an NIS `lkd.map`. See `/usr/share/doc/lk-2.0/nis/readme.txt` for how to setup this map. For more information, see `lkd.map`(5) for more information.

Using the LK Floating Licenses Facilities

In order to use LK floating licenses facilities, the `lkd` daemon has to run in two places:

- The license server

  The host acting as the license server

- The license client(s)

  All the hosts where applications utilizing LK floating licenses capabilities

Upon start-up, `lkd` determines the licenses to serve by using the standard lookup LK mechanism, as described in `lk`(5) man page. The daemon will only serve licenses belonging to its local `hostid` as determined by `lk_hostid`(1) program. LK client applications use the same mechanism. However, the daemon does not monitor license file changes. This means `lkd` need to be restarted to account for license file changes.

Also, note that the license file may contain floating licenses for several server. For example:

```
# floating license server by ssn=00a0d1e4071d for product ’a’
product=a, version=1, count=5@00a0d1e4071d, begDate=0, expDate=0, \
     licenseID=0, key=/sHK7L45x8aiHy9VRt+YWx/c8KTYxWgd, \
     vendor="Silicon Graphics, Inc.",info="vendor_info"

# floating license server by ssn=00a0d1e4080b for product ’b’
product=b, version=1, count=5@00a0d1e4080b, begDate=0, expDate=0, \
     licenseID=0, key=fs5rUFeD3nBdzAGw6UDsmb5yw2wioNOh, \
      vendor="Silicon Graphics, Inc.",info="vendor_info"

# a nodeless license for product ’c’
product=c, version=1, count=5, begDate=0, expDate=0, licenseID=0, \
      key=Pl8UFMVeNBo/SiiG6cbBb4CDchO5R2z8, \
      vendor="Silicon Graphics, Inc.",info="vendor_info"

# a nodelock license for product ’d’
product=d, version=1, count=, begDate=1248886680, \
  expDate=0, licenseID=d1e4071d, \
  key=f5DvhYYFuvHoX2FVoI/ixvcBITbnE9pn
```

The LK licensing daemon (`lkd`) exchanges LK license requests of clients using TCP/IP official `sgi-lk` port `11106`. You can use the `-L` option to specify a different port. When using this option, make sure this is reflected in the `ssn` mapping. See the `lkd.map`(5) man page for more information. The `lkd` daemon also satisfies local LK license request using a UNIX TCP socket file in `/var/run/lkd/socket.SGI.server`. Client applications never talk to remote license server via TCP/IP. Instead, all requests are made though the local daemon who will either satisfy the request locally or act as a proxy, and handle the request to the remote license server via TCP/IP. Consequently, it means there are no direct LK exchange between the application and remote LK daemon over TCP/IP.

When an LK request is granted to a client application via a `lk_get`(3) call, the daemon decrements the count for the granted key. The license server then starts monitoring (pings), at periodic interval (`-P` option), the client daemon to make sure the client process ID (PID) is still present and not a "zombie" process. See the `lkdc`(8) man page for more information.

Similarly, when the application release the key using `lk_free`(3), the daemon increments the original granted count for the application. Should a client application hang or crash, the license server acts as if the application crashed, hence "taking back" the granted count.

The `lkd` daemon can also report information about granted keys, and so on. This is enabled by the `-c` option. This is normally specified (by default) in `/etc/sysconfig/lkd` file. See the `lkdc`(8) man page for more information.

The `lkd` daemon also maintains a journal of license activities (`-j` option). This file should be cleaned and/or rotated appropriately from time to time. To turn off the journal option, just specify `-j /dev/null`.

Configuring Your System

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

- "CPU Frequency Scaling" on page 7
- "System Partitioning " on page 8
- "Making Array Services Operational" on page 20

CPU Frequency Scaling

CPU frequency scaling is disabled by default on your SGI UV 100 , SGI UV 1000, or SGI UV 2000 system. This is accomplished by adding the `acpi-cpufreg` file to the `/etc/modprobe.d` directory.

An example is, as follows:

```
admin:/etc/modprobe.d # cat acpi-cpufreq

# comment out the following line to enable CPU frequency scaling

install acpi-cpufreq /bin/true
```

To enable CPU frequency scaling, log into your SGI UV system (`ssh root@hostname`) and remove the `acpi-cpufreg` file in the `/etc/modprobe.d` directory. If your system is partitioned, you need to perform this on each partition.

If you decide to enable CPU frequency scaling on your system, SGI highly recommends that you set the default scaling governor to `performance` using the following script:

```
maxcpu=‘grep processor /proc/cpuinfo | awk ’{print $3}’ | tail -1‘

for cpu in ‘seq 0 $maxcpu‘

do

   cpufreq-set -c $cpu -g performance
```

```
done
```

Note: In order to enable the Intel processor's Turbo boost feature, CPU frequency scaling has to be enabled.

System Partitioning

This section describes how to partition an SGI UV 100, SGI UV 1000 or SGI UV 2000 server and contains the following topics:

- "Overview" on page 8
- "Advantages of Partitioning" on page 10
- "Limitations of Partitioning" on page 11
- "Supported SSI" on page 11
- "Installing Partitioning Software and Configuring Partitions" on page 14

Overview

A single SGI UV 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 processor cores 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 called *cross-partition communication*. XPNET is the TCP/IP interface for cross-partition communication over NUMAlink. 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. For more information on MPT, see the *Message Passing Toolkit (MPT) User Guide*.

Partition discovery software allows all of the partitions to know about each other.

Partition firewalls provide memory protection for each partition. XPMEM software uses firewall code to open up a portion of memory so that is can be accessed by CPU cores in other partitions. Firewalls have kernel, BIOS, and hardware components. The kernel component is XPMEM. XPMEM allocates some memory and makes a BIOS call passing BIOS the memory address. The BIOS calls allow the UV hub hardware to change the memory protections on each of the cache lines in the block of memory. Hardware memory directory bits indicate which nodes have access to that cache line. Nodes in the partition always have access to memory in their own partition. Nodes outside the partition do not have access, unless the memory is opened up by XPMEM.

A *coherence domain* is the extent to which a CPU core is able to coherently load and store cacheable memory. Usually, a coherence domain is the same as an operating system partition or single-system image. With XPMEM, the coherence domain can be expanded to include memory in other partitions. *Expanding the coherence domain* is part of the process of opening up the memory in a remote partition.

A *heartbeat* mechanism allows each partition to determine the state of all partitions in the system. Each partition increments its own heartbeat which is read by other partitions. As long as the local partition keeps incrementing its heartbeat, the other partitions know that it is still operational.

Each partition has a *partition page* that stores its heartbeat. A partition page is a page of physical memory that contains information about the local partition and whose address is known by the other partitions in the system.

Reset fences are special hardware mechanisms built into the NUMAlink ports of hubs and routers that prevent resets from propagating into a partition. Reset fences are set at the partition boundaries. BIOS sets up the reset fences after it does NUMAlink discovery.

The global reference unit (GRU) no-fault code allows a partition to accesses a remote partition safely. If the remote access fails, the GRU no-fault code cleans up the GRU so it can be reused.

All of the partitions in a partitioned system have the same *system serial number*. The system serial number is stored in the system controller.

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

Advantages of Partitioning

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

- "Create a Large, Shared-memory Cluster" on page 10
- "Provides Fault Containment" on page 10
- "Allows Variable Partition Sizes" on page 10
- "Provide High Performance Clusters" on page 11

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 for optimized use of Message Passing Interface (MPI) software and logically shared, distributed memory access (SHMEM) routines. The globally addressable, cache coherent, shared memory is exploited by MPI and SHMEM to deliver high performance.

Provides Fault Containment

Another reason for partitioning a system is fault containment. In most cases, a single partition can be brought down (because of a hardware or software failure, or as part of a controlled shutdown) without affecting the rest of the system. Hardware memory protections prevent any unintentional accesses to physical memory on a different partition from reaching and corrupting that physical memory. For current fault containment caveats, see "Limitations of Partitioning" on page 11.

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 CPU cores each or configured into two partitions of 64 CPU cores 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 CPU cores 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 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 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.

Failures can usually be contained within a partition even when memory is being shared with other partitions. XPC is invoked using normal shutdown commands such as `reboot`(8) and `halt`(8) to ensure that all memory shared between partitions is revoked before the partition resets. This is also done if you remove the XPC kernel modules using the `rmmod`(8) command. Unexpected failures such as kernel panics or hardware failures almost always force the affected partition into the KDB kernel debugger or the LKCD crash dump utility. These tools also invoke XPC to revoke all memory shared between partitions before the partition resets. XPC cannot be invoked for unexpected failures such as power failures and spontaneous resets (not generated by the operating system), and thus all partitions sharing memory with the partition may also reset.

Supported SSI

The SGI UV 1000 system sizes range from 2 to 128 blades (16 to 2048 cores) in a single system image (SSI). . The SGI UV 100 series is a family of multiprocessor distributed shared memory (DSM) computer systems that initially scale from 16 to 768 Intel processor cores as a cache-coherent single system image (SSI).

For SGI UV 1000 systems, the maximum number of processor cores in an SSI is 2048. The following describe the minimum and maximum metrics within an SSI:

- one partition
- one to four racks
- one to eight individual rack units (IRUs) with maximum of two IRUs per rack
- one to eight Base I/O (only one Base I/O has the capability to boot the system)
- two to 128 compute blades
- two to 128 SGI UV Hubs (one Hub on each compute blade)
- two to 256 processor sockets (one socket on memory expansion blade, two sockets on compute blade)
- 16 to 2048 processor cores (up to 4096 threads with Hyper-Threading enabled)
- eight to 2048 DDR3 memory DIMMs (16 DIMMs maximum per compute blade)
- Up to 16 terabytes (TBs) with up to 4 TB per rack (using 8 GB DIMMs)

Currently, the Linux operating system only supports 2048 cores/threads.

Note: The terms single system image (SSI) and partition can be used interchangeably.

See the *SGI Altix UV 1000 System User’s Guide* for information on configurations that are supported for system partitioning.

The SGI UV 2000 system is a large densely packed, blade-based, cache-coherent non-uniform memory access (ccNUMA), computer system that is based on the Intel® Xeon® processor E5 family. The basic building block of the UV system is the individual rack unit (IRU). The IRU is a 10U high enclosure that supports the following:

- Eight compute blades
- One chassis management controller
- Three power supplies
- Nine cooling fans

The UV 2000 system scales, as follows:

- From 2 to 128 compute blades in a single system image (SSI)
- A maximum of 2048 processor cores with hyper-threading turned off
- A maximum of 4096 processor threads (2048 processor cores) with hyper-threading turned on

  Each processor core supports two threads.

The following describe the minimum and maximum metrics within an SSI for SGI UV 2000 systems:

- No one partition can be larger than one-half the system size. Exception: A three IRU system can be partitioned as two IRUs and one IRU.
- The minimum granularity for a partition is two compute blades.
- Each partition must have the infrastructure to run as a standalone system. This infrastructure includes a system disk and console connection.
- An I/O blade belongs to the partition that the attached IRU belongs to. I/O blades cannot be shared by two partitions.
- Peripherals, such as dual-ported disks, can be shared the same way two nodes in a cluster can share peripherals.
- Partitions must be contiguous in the topology (for example, the route between any two nodes in the same partition must be contained within that partition - and not route through any other partition). This allows intra-partition communication to be independent of other partitions.
- Partitions must be fully interconnected. That is to say, for any two partitions, there is a direct route between those partitions without passing through a third. This is required to fulfill true isolation of a hardware or software fault to the partition in which it occurs.
- If the system is unpartitioned, then routerless systems can have any number of blades 1-32, but the missing blades should be at the end (highest IRU highest blade number). Unpartitioned routered systems of size 3 or 4 IRUs should be multiples of blade pairs, missing at the end. Unpartitioned routered systems of size 5 and above IRUs need to be in multiples of 4 blades, missing at the end.

See the *SGI UV 2000 System User Guide* for information on configurations that are supported for system partitioning.

For additional information about configurations that are supported for system partitioning, see your sales representative.

Installing Partitioning Software and Configuring Partitions

To enable or disable partitioning software, see "Partitioning Software" on page 14, to use the system partitioning capabilities, see "Partitioning a System" on page 15 and "Partitioning a System" on page 15.

This section covers the following topics:

- "Partitioning Software" on page 14
- "Partitioning a System" on page 15

Partitioning Software

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

On SGI systems running SLES11, if you intend to use the cross-partition functionality of XPMEM, you will need to add `xpc` to the line in the `/etc/sysconfig/kernel` file that begins with `MODULES_LOADED_ON_BOOT`. Once that is added, you may either reboot the system or issue an `modprobe xpc` command to get the cross-partition functionality to start working. For more information on using `modprobe`, see the `modprobe`(8) man page.

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

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

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

Procedure 2-1 Setting up Networking Between Partitions

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

1. Log in as root.
2. For SGI systems, configure the `xp0` IP address using `yast2`. For information on using `yast2`, see *SUSE Linux Enterprise Server 11 Administration Guide* . The driver's full name inside `yast2` is `SGI Cross Partition Network adapter`.
3. Add the network address for the `xp0` interface by editing the `/etc/hosts` file.
4. Reboot your system or restart networking.

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

Partitioning a System

This section describes how to partition your system. The following example shows how to use chassis manager controller (CMC) software to partition a two rack system containing four IRUs into four distinct systems, use the `uvcon` command to open a console and boot each partition and repartiton it back to a single system.

Note: Each partition must have one base I/O blade and one disk blade for booting. `001i01b00` refers to rack 1, IRU 0, and blade00. `r001i01b01` refers to rack 1, IRU 0, and blade01.

Base I/O and the boot disk are displayed by the `config -v` command, similar to the following:

```
r001i01b00 IP93-BASEIO
r001i01b01 IP93-DISK
```

Procedure 2-2 Partitioning a System Into Four Partitions

To partition your system, perform the following steps :

1. Use the `hwcfg` command to create four system partitions, as follows:

   ```
   CMC:r1i1c>hwcfg partition=1 "r1i1b*"
   CMC:r1i1c>hwcfg partition=2 "r1i2b*"
   CMC:r1i1c>hwcfg partition=3 "r2i1b*"
   CMC:r1i1c>hwcfg partition=4 "r2i2b*"
   ```

2. Use the `config -v` command to show the four partitions, as follows:

   ```
   CMC:r1i1c> config -v

   CMCs:           4
           r001i01c UV1000 SMN
           r001i02c UV1000
           r002i01c UV1000
           r002i02c UV1000

   BMCs:          64
           r001i01b00 IP93-BASEIO P001
           r001i01b01 IP93-DISK P001
           r001i01b02 IP93-INTPCIE P001
           r001i01b03 IP93 P001
           r001i01b04 IP93 P001
           r001i01b05 IP93 P001
           r001i01b06 IP93 P001
           r001i01b07 IP93 P001
           r001i01b08 IP93 P001
           r001i01b09 IP93-INTPCIE P001
           r001i01b10 IP93-INTPCIE P001
           r001i01b11 IP93-INTPCIE P001
           r001i01b12 IP93-INTPCIE P001
           r001i01b13 IP93 P001
           r001i01b14 IP93 P001
           r001i01b15 IP93 P001
           r001i02b00 IP93-BASEIO P002
           r001i02b01 IP93-DISK P002
           r001i02b02 IP93-INTPCIE P002
           r001i02b03 IP93 P002
           r001i02b04 IP93 P002
   ```

```
r001i02b05 IP93 P002
r001i02b06 IP93 P002
r001i02b07 IP93 P002
r001i02b08 IP93 P002
r001i02b09 IP93 P002
r001i02b10 IP93 P002
r001i02b11 IP93 P002
r001i02b12 IP93 P002
r001i02b13 IP93 P002
r001i02b14 IP93 P002
r001i02b15 IP93 P002
r002i01b00 IP93-BASEIO P003
r002i01b01 IP93-DISK P003
r002i01b02 IP93 P003
r002i01b03 IP93 P003
r002i01b04 IP93 P003
r002i01b05 IP93 P003
r002i01b06 IP93 P003
r002i01b07 IP93 P003
r002i01b08 IP93 P003
r002i01b09 IP93 P003
r002i01b10 IP93 P003
r002i01b11 IP93 P003
r002i01b12 IP93 P003
r002i01b13 IP93 P003
r002i01b14 IP93 P003
r002i01b15 IP93 P003
r002i02b00 IP93-BASEIO P004
r002i02b01 IP93-DISK P004
r002i02b02 IP93 P004
r002i02b03 IP93 P004
r002i02b04 IP93 P004
r002i02b05 IP93 P004
r002i02b06 IP93 P004
r002i02b07 IP93 P004
r002i02b08 IP93 P004
r002i02b09 IP93 P004
r002i02b10 IP93 P004
r002i02b11 IP93 P004
r002i02b12 IP93 P004
r002i02b13 IP93 P004
```

```
        r002i02b14 IP93 P004
        r002i02b15 IP93 P004

Partitions:      4
        partition001 BMCs:   16
        partition002 BMCs:   16
        partition003 BMCs:   16
        partition004 BMCs:   16
```

3. Use can also use the `hwcfg` command to display the four partitions, as follows:

```
CMC:r1i1c> hwcfg
NL5_RATE=5.0
PARTITION=1 ............................................... 16/64 BMC(s)
PARTITION=2 ............................................... 16/64 BMC(s)
PARTITION=3 ............................................... 16/64 BMC(s)
PARTITION=4 ............................................... 16/64 BMC(s)
```

4. To reset the system and boot the four partitions, use the following commands:

```
CMC:r1i1c> power on
CMC:r1i1c> power reset "p*"
```

Note: In the `power reset "p*"` command, above, quotes are required to prevent shell expansion.

5. Use the `uvcon` command to open consoles to each partition and boot the partitions. Open a console to partition one, as follows:

```
CMC:r1i1c> uvcon p1
uvcon: attempting connection to localhost...
uvcon: connection to SMN/CMC (localhost) established.
uvcon: requesting baseio console access at partition 1 (r001i01b00)...
uvcon: tty mode enabled, use 'CTRL-]' 'q' to exit
uvcon: console access established (OWNER)
uvcon: CMC <--> BASEIO connection active
************************************************
*******  START OF CACHED CONSOLE OUTPUT  *******
************************************************

******** [20100513.215944] BMC r001i01b15: Cold Reset via NL broadcast reset
******** [20100513.215944] BMC r001i01b07: Cold Reset via NL broadcast reset
```

```
******** [20100513.215945] BMC r001i01b13: Cold Reset via NL broadcast reset
******** [20100513.215945] BMC r001i01b05: Cold Reset via NL broadcast reset
******** [20100513.215945] BMC r001i01b06: Cold Reset via NL broadcast reset
******** [20100513.215946] BMC r001i01b10: Cold Reset via NL broadcast reset
******** [20100513.215946] BMC r001i01b09: Cold Reset via NL broadcast reset
******** [20100513.215945] BMC r001i01b11: Cold Reset via NL broadcast reset
******** [20100513.215945] BMC r001i01b12: Cold Reset via NL broadcast reset
******** [20100513.215945] BMC r001i01b04: Cold Reset via NL broadcast reset
******** [20100513.215945] BMC r001i01b08: Cold Reset via NL broadcast reset
******** [20100513.215946] BMC r001i01b02: Cold Reset via NL broadcast reset
******** [20100513.215945] BMC r001i01b00: Cold Reset via NL broadcast reset
******** [20100513.215945] BMC r001i01b14: Cold Reset via NL broadcast reset
******** [20100513.215947] BMC r001i01b09: Cold Reset via ICH
******** [20100513.215946] BMC r001i01b12: Cold Reset via ICH
******** [20100513.215947] BMC r001i01b10: Cold Reset via ICH
******** [20100513.215947] BMC r001i01b11: Cold Reset via ICH
******** [20100513.215947] BMC r001i01b02: Cold Reset via ICH
******** [20100513.215947] BMC r001i01b00: Cold Reset via ICH
******** [20100513.215953] BMC r001i01b03: Cold Reset via NL broadcast reset
******** [20100513.220011] BMC r001i01b01: Cold Reset via NL broadcast reset
******** [20100513.220012] BMC r001i01b08: Cold Reset via NL broadcast reset
******** [20100513.220012] BMC r001i01b07: Cold Reset via NL broadcast reset
******** [20100513.220011] BMC r001i01b15: Cold Reset via NL broadcast reset
******** [20100513.220012] BMC r001i01b06: Cold Reset via NL broadcast reset
******** [20100513.220012] BMC r001i01b05: Cold Reset via NL broadcast reset
******** [20100513.220012] BMC r001i01b14: Cold Reset via NL broadcast reset
******** [20100513.220012] BMC r001i01b13: Cold Reset via NL broadcast reset
******** [20100513.220011] BMC r001i01b04: Cold Reset via NL broadcast reset
******** [20100513.220012] BMC r001i01b03: Cold Reset via NL broadcast reset
******** [20100513.220013] BMC r001i01b09: Cold Reset via NL broadcast reset
******** [20100513.220013] BMC r001i01b10: Cold Reset via NL broadcast reset
******** [20100513.220013] BMC r001i01b11: Cold Reset via NL broadcast reset
******** [20100513.220012] BMC r001i01b12: Cold Reset via NL broadcast reset
******** [20100513.220012] BMC r001i01b02: Cold Reset via NL broadcast reset
******** [20100513.220012] BMC r001i01b00: Cold Reset via NL broadcast reset
******** [20100513.220014] BMC r001i01b09: Cold Reset via ICH
******** [20100513.220014] BMC r001i01b10: Cold Reset via ICH
******** [20100513.220014] BMC r001i01b11: Cold Reset via ICH
******** [20100513.220013] BMC r001i01b12: Cold Reset via ICH
******** [20100513.220013] BMC r001i01b02: Cold Reset via ICH
******** [20100513.220016] BMC r001i01b00: Cold Reset via ICH
```

```
******** [20100513.220035] BMC r001i01b14: Cold Reset via NL broadcast reset
******** [20100513.220035] BMC r001i01b06: Cold Reset via NL broadcast reset
******** [20100513.220034] BMC r001i01b15: Cold Reset via NL broadcast reset
******** [20100513.220035] BMC r001i01b05: Cold Reset via NL broadcast reset
******** [20100513.220034] BMC r001i01b01: Cold Reset via NL broadcast reset
******** [20100513.220035] BMC r001i01b07: Cold Reset via NL broadcast reset
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  ....
Hit [Space] for Boot Menu.
ELILO boot:
```

Note: Use the `uvcon` command to open consoles on the other three partitions and boot them. The system will then have four single system images.

6. Use the `hwcfg -c` partition command to clear the four partitions, as follows:

```
CMC:r1i1c> hwcfg -c partition
PARTITION=0
PARTITION=0
```

Note: This will take several minutes on large systems.

7. To reset the system and boot it as a single system image (one partition), use the following command:

```
CMC:r1i1c> power reset "p*"
```

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*.

Standard Array Services is installed by default on an SGI Performance Suite 1.4 system. To install Secure Array Services, use the YaST Software Management and use the **Filter->search** function to search for secure array services by name (`sarraysvcs`).

Procedure 2-3 Making Array Services Operational

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

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

---

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

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

   ---

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

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

   ```
   array who
   ```

   You should see yourself listed.

System Operation

This chapter describes some basic system operations using the chassis manager controller (CMC) commands and covers the following topics:

SGI UV System Control Network

SGI Management Center (SMC) software running on the system management node (SMN) provides a robust graphical interface for system configuration, operation, and monitoring. For more information, see *SGI Management Center System Administrator's Guide*.

The SGI UV 2000, SGI UV 1000, or UV 100 system control network is a private, closed network. It is not to be reconfigured in any way different from the standard UV installation, nor is it to be directly connected to any other network. The SGI UV system control network does not accommodate additional network traffic, routing, address naming other than its own schema, and DCHP controls other than its own configuration. The system control network also is not security hardened, nor is it tolerant of heavy network traffic, and is vulnerable to Denial of Service attacks. The system management node acts as a gateway between the UV system control network and any other networks. For more information on the SGI UV system control network, see the *SGI Altix UV 100 System User's Guide*, the *SGI Altix UV 1000 System User's Guide* or the *SGI UV 2000 System User Guide*.

Connecting to the UV System Controller Network Using the CMC

The SGI UV software controller is designed to manage and monitor the individual blades in SGI UV systems. Depending on your system configuration, you can monitor and operate the system from the system management node (SMN) or on smaller systems, such as, the SGI UV 100 from the CMC itself. UV 1000 systems up to 16 racks (four building blocks, also called one super block) can also be controlled and monitored from a CMC in the system. For more information, see the “Connecting to the UV System Controller Network” in the *SGI UV CMC Controller Software User Guide*.

Power on and Booting an SGI UV System from Complete Power Off

To boot an SGI UV system from complete power off, perform the following steps:

1. Make sure the power breakers are on.
2. Establish a serial connection to the CONSOLE on the CMC or establish a network connection to the CMC. For a network connection, your PC must be connected to the CMC (via the network connection) and have its `/etc/hosts` file setup to include the CMCs. See "Connecting to the UV System Controller Network Using the CMC" on page 24.
3.

```
ssh root@hostname-cmc
SGI Chassis Manager Controller, Firmware Rev. 0.0.22

CMC:r1i1c>
```

   Typically, the default password set out of the factory is **`root`**. The CMC prompt appears. **CMC:r1i1c** refers to rack 1, IRU 1, CMC.

   If the host name is not set up in the PC/workstation’s hosts file, you can simply use the IP address of the CMC, as follows:

```
ssh root@<IP-ADDRESS>
```

4. Power up your SGI UV system using the `power on` command, as follows:

```
CMC:r1i1c> power on
```

Note: You can open a second window on the CMC, `ssh root@hostname-cmc` and use the `console` command to open a console and watch the system power on.

5. Open a second console to the CMC using the `console` command, formerly, `uvcon` command to see the system power on, as follows:

```
ssh root@hostname-cmc
SGI Chassis Manager Controller, Firmware Rev. 0.0.22

CMC:r1i1c> console
console: attempting connection to localhost...
console: connection to SMN/CMC (localhost) established.
console: requesting baseio console access at r001i01b00...
console: tty mode enabled, use 'CTRL-]' 'q' to exit
console: console access established
console: CMC <--> BASEIO connection active
************************************************
*******  START OF CACHED CONSOLE OUTPUT  *******
************************************************

******** [20100512.143541] BMC r001i01b10: Cold Reset via NL broadcast reset
******** [20100512.143541] BMC r001i01b07: Cold Reset via NL broadcast reset
******** [20100512.143540] BMC r001i01b08: Cold Reset via NL broadcast reset
******** [20100512.143540] BMC r001i01b12: Cold Reset via NL broadcast reset
******** [20100512.143541] BMC r001i01b14: Cold Reset via NL broadcast reset
******** [20100512.143541] BMC r001i01b04: Cold Reset via NL
```

Note: Use `CTRL-] q` to exit the console.

6. Depending upon the size of your system, in can take 5 to 10 minutes for the SGI UV system to power on. When the **shell>** prompt appears, enter **`fs0:`**, as follows:

```
shell> fs0:
```

7. At the **fs0:** prompt, enter **`boot`**, as follows:

```
fs0:> boot
```

ELILO Linux Boot loader is called and various SGI configuration scripts are run and the SUSE Linux Enterprise Server 11 SP2 installation program appears.

Power off an SGI UV System

To power down the SGI UV sytem, use the `power off` command, as follows:

```
CMC:r1i1c> power off
==== r001i01c (PRI) ====
```

You can use the `power status` command, to check the power status of your system, as follows:

```
CMC:r1i1c> power status
==== r001i01c (PRI) ====
on: 0, off: 32, unknown: 0, disabled: 0
```

Power NMI to Drop into KDB

To send a nonmaskable interrupt (NMI) signal from the power command to the CMC to drop into the kernel debugger (KDB), use the `power nmi` command, as follows:

```
CMC:r1i1c> power nmi

Entering kdb (current=0xffff8aa3fe11c040, pid 0) on processor 7 due to NonMaskable
     r15 = 0x0000000000000000      r14 = 0x0000000000000000
     r13 = 0x0000000000000000      r12 = 0x0000000000000000
      bp = 0xffffffff81927380       bx = 0xffff8ac1ff11dfd8
     r11 = 0xffffffff8101a2c0      r10 = 0xffff88000beefd18
      r9 = 0x00000000ffffffff       r8 = 0x0000000000000000
      ax = 0x0000000000000000       cx = 0x0000000000000000
      dx = 0x0000000000000000       si = 0xffff8ac1ff11dfd8
      di = 0xffffffff81a2b308  orig_ax = 0xffffffffffffffff
      ip = 0xffffffff8100ad42       cs = 0x0000000000000010
   flags = 0x0000000000000246       sp = 0xffff88000bee7ff0
      ss = 0x0000000000000018 &regs = 0xffff88000bee7f58
[7]kdb>
```

Upgrading System BIOS

To upgrade the compute blade PROM, perform the following steps:

1. From the CMC prompt, to show the current PROM level, perform the folllowing command:

   ```
   CMC:r1i1c> showbios
   Flashed on Sat May  1 14:14:45 UTC 2010 was bios.latest.fd (20100429_1603)
   ```

2. Get the newest PROM image from SupportFolio Online at http://support.sgi.com/.

3. Copy the latest BIOS to a directory on the CMC in `/work/bmc/common/` An example directory is, as follows:

   ```
   CMC:r1i1c> ls
   bios.latest.fd flashbios
   ```

4. Use the `flashbios` command to flash the compute blade BIOS, as follows:

   ```
   CMC:r1i1c> flashbios
   Using default bios: bios.latest.fd
   Checking processor status on all nodes....
   Done. System is read for BIOS flash update
   Flashing bios bios.lastest.fd (20100429_1603) This will take several minutes.
   ...
   ```

For more information on using CMC software, see the *SGI UV CMC Controller Software User Guide*.

For information on updating firmware on your SGI UV system from the system management node (SMN) and new firmware update commands, see the *SGI UV System Management Node Administrator Guide*.

Index