Partitioning The Silicon Graphics Origin 2000(TM)

Allan J Christie
Silicon Graphics Computer Systems
2011 N. Shorline Blvd.
Mountain View, California 94043-1389

ajc@sgi.com

Cray User Group Conference
June 15 - 19, 1998

Abstract

Partitioning a machine provides a mechanism to divide a single Origin 2000 into multiple distinct hosts, each with their own resources. This paper provides a brief overview of the hardware and software support for partitioning, along with an overview of valid partitioned configurations and a description of how to partition a system.

1 Introduction

The Origin 2000 hardware contains several new features to support partitioning. First, consider the requirements of partitioning one physical machine into two or more logical machines. Each machine should be unto itself, a stand alone entity. It must not require the assistance, or even presence of any other partition to function. Each partition must also be isolated from any other partition in the system in several areas:

1. Software failures in one partition must not affect the operation of another partition.
2. Hardware failures in one partition must not affect operation in another partition.
3. A reset or power cycle in one partition must not affect another partition.

These requirements lead to three fundamental design decisions:

1. Cache lines can not be owned or shared outside the owning partition.
2. Intra-partition Craylink traffic must be routed entirely within the partition.
3. Inter-partition Craylink traffic must be routed entirely within the two partitions containing the end-points.

Design decisions 2 and 3 have the added benefit that Craylink traffic in a partition is either related to accesses from within the partition, or related to an explicit inter-partition transfer. Thus, programs running in 2 different partitions do not interfere with each other in terms of memory latency, memory bandwidth or Craylink bandwidth.

2 Hardware Support

The basic building block of the Origin 2000 system is the *node* board. A node board is made up of 2 processors with their associated caches, memory and its associated directory, an I/O interface, and a craylink network interface.

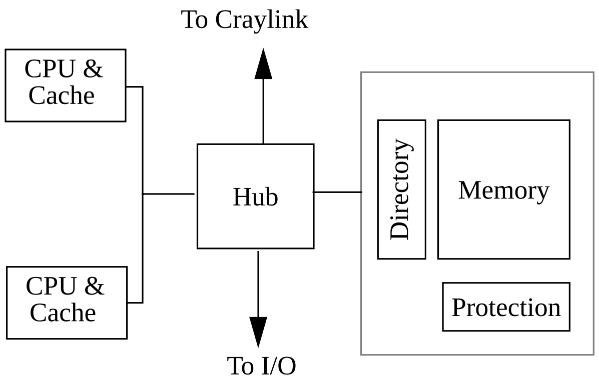


Figure 1: Origin 2000 Node

The Hub is a Silicon Graphics ASIC that controls the interfaces to each of the components. Each of these nodes is assigned a Named Address Space ID or *NASID*. NASIDs are assigned on a system wide basis and not a partition wide basis. The NASID uniquely identifies the node within the Craylink interconnect.

2.1 Regions

Partitioning the Origin 2000 requires a means to specify protections. A *Region* is the building block for all of the protection mechanisms used. A region is a group of NASIDs (and thus a group of nodes). Origin hardware supports 64 regions, and on systems up to 64 nodes (128 processors), regions map one-to-one with the nodes. Beyond 64 nodes, a region corresponds to 8 nodes (or 16 processors). For example, region 0 corresponds to nodes 0 through 7, region 1 corresponds to nodes 8 through 15 etc.

2.2 Memory

All memory in an Origin 2000 system is globally addressable. For a CPU to access memory on a given node, all that is required is to generate the appropriate address. However, as indicated in Figure 1: *Origin 2000 Node*, all physical memory is accompanied by an associated set of protection memory. Each 4KB segment of physical memory includes 64 protection entries, one for each region. Each region’s access to a specific part of memory is described by these protection entries and can be set to No Access or Read/Write access. Some Origin 2000 models support separate R/W permissions for I/O and processors accesses but these are currently set to the same value.

2.3 Craylink

There are two important aspects to the Craylink network specific to partitions: reset propagation and routing. Reset signals are propagated across Craylink to ensure that all modules in a system are reset when a single module is reset. The Origin routers have the ability to block incoming reset signals from propagating.

Craylink routing in a partitioned system is set up the same as in a non-partitioned system. Each partition is provided a route to every other partition but hardware protections restrict accesses to resources by remote partitions.

2.4 Hub

The HUB contains a series of configuration and status registers that are part of the global address space. These registers are part of the global address space and can be accessed by any node in the system. For this reason, hardware protections similar to those used to protect memory, protect against un-authorized access from nodes outside the partition. The only exception is the *Cross Call Interrupt Register*, described in *Section 2.5 Cross Call Mechanism*.

2.5 Cross Call Mechanism

A special cross partition interrupt mechanism is provided by the hardware to allow a remote partition to interrupt a specific CPU. This is accomplished by writing to a specific HUB register known as the *Cross Call Interrupt Register.* The HUB contains two such registers, one for each of the processors attached to it. Unlike all other registers in the HUB, this register is always writable by all processors in all partitions. A write to this register causes an interrupt to the corresponding local processor, and the contents of the write data are ignored. The HUB contains three additional registers related to the cross call mechanism that are protected by the protection mechanism described in *Section 2.4 Hub*: the *Cross Call Interrupt Mask* register, the *Cross Call Interrupt Pendin*g register, and the *Cross Call Interrupt Pending Clear* register. The cross call interrupt pending register contains a mask of the regions of all the processors that have written the Cross Call Interrupt register. The Cross Call Interrupt Mask register contains a mask of regions that are permitted to generate interrupts. This register is ANDed with the cross call interrupt pending register, and if the result is non-zero, an interrupt is signaled to the processor. Bits in the Cross Call interrupt pending register are cleared by writing to the corresponding cross call interrupt pending clear register.

2.6 Block Transfer Engine

Although the block transfer engine (BTE) is generally not required to partition a system, it is used for cross partition communication. The BTE is capable of copying a cache coherent block of data from one address to another. Each node on the system contains a BTE as part of the HUB. The origin hardware supports a special type of read operation for the BTE known as a *BTE read shared operation*. A BTE data pull is accomplished by sending a *BTE read shared request* to the node that owns the targeted physical memory. That node pulls any modified data from processor’s caches (which are by definition within the same partition as the targeted node), and returns a copy of the data to the requesting BTE. Since the BTE is making a copy of the data, the node that owns the memory does **not** update the directory for the targeted cache lines (the requesting node is never a shared or exclusive owner of the cache lines in question). This read operation is also allowed to bypass the normal memory permissions allowing a partition to pull a copy of data from another partition without having permission to read or write the memory.

3 Configurations

Conceptually, Origin 2000 configurations can be viewed as part of a cube (in the smallest configurations) or a series of connected cubes. A common visualization of this configuration is shown in figure Figure 2: *Origin 2000 Representation*, which shows a single module (8 processors).

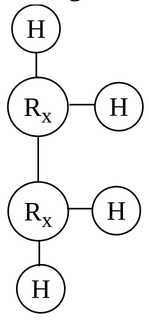


Figure 2: Origin 2000 Representation

In this figure, the vertices of the cube labeled with R_x represent routers. Each router is connected to two nodes, shown as circled H's (representing HUBs). The subscript (X) on the router designation is used to identify routers that are part of the same module. The two R_x's indicate the routers are in the same module. To simplify the diagrams, often the HUBs are omitted. The lines connecting the routers represent Craylink connections. The smallest Origin 2000 is 1 module which can support up to 4 nodes and 2 routers (although it need not be fully populated).

Several restrictions are placed on the configurations allowed in a partitioned system, primarily to maintain the independence of the participating partitions.

1 All partitions must be totally self contained. No intra-partition network traffic can travel outside the partition.

2 All partitions must be fully interconnected - the route between any node in one partition to any node in another partition must be contained within those two partitions.

3 Partition boundaries must coincide with routers.

Restriction (1) is required to ensure the independence of each partition. If network traffic between to nodes in the same partition (A) was routed through another partition (B), then partition A could be adversely affected by partition B. If partition B was powered off for example, a cache read in partition A could cause coherency traffic to be routed through partition B, and never be satisfied. This type of scenario is represented by grouping module 1 and module 4 as partition A and module 2 and module 3 as partition B in Figure 3: *32-processor Configuration*. The dotted arrow shows the path a coherency request between a node in module 2 and a node in module 3 may take. This path requires a correctly functioning and powered on router in module 4.

A similar problem leads to requirement (2). If a partition (A) is communicating to another partition (C) through a third (B), and partition B fails or is powered down, partitions A and C may not be able to communicate with each other. Applications on partitions A and C that are using cross partition communication will see a failure caused by a third partition. Such a failure will also cause problems when partitions are being discovered. Partition A and C will never discover each other if the traffic between them is routed though a third non-functional partition. Again using Figure 3: *32-processor Configuration*, assume each module forms a partition where partition A, B, C, and D are comprised of modules 1, 2, 3, and 4 respectively. If partition B and D are not operational, any traffic being routed from partition A to partition C will fail.

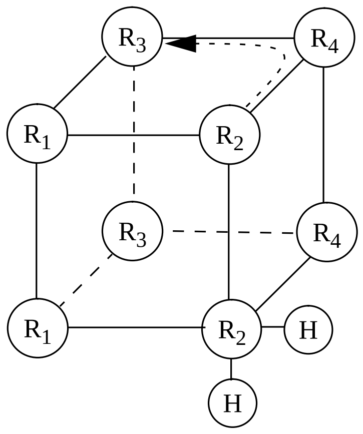


Figure 3: 32-processor Configuration

The final requirement (3) is dictated by the reset barriers. As described in section *Section 2.3 Craylink*, the reset signal is propagated across the Craylink and can only be fenced off at the incoming router port and NOT the Network Interface on the HUB. All nodes and routers in a module are reset when a non-fenced reset signal is detected.

It is possible to configure a 32 processor system as four partitions by fully interconnecting the partitions. By adding Craylink connections between module 2 and 3, and between module 1 and 4, all partitions can be connected to all other partitions. One of the standard Origin 2000 32 processor configurations includes *Express Links*, which provide these extra connections. Figure 4: *Express-Link 32-Processor* shows the logical connectivity of such a system, with the express links shown as double lines.

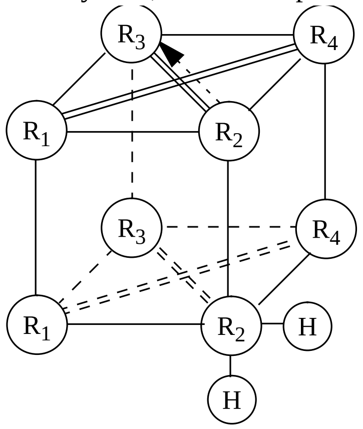


Figure 4: Express-Link 32-Processor Configuration

The requirements listed restrict the configurations that are valid, while other configurations are not supported for configuration testing reasons. Table 1 *Supported Hardware Configurations* lists the set of supported configurations.

Modules	# Partitions	Modules per Partition	CPUs per Partition
1	1	1	1 to 8
2	1	2	2 to 16
	2	1	1 to 8
4	1	4	4 to 32
	2	2	2 to 16
	4[a]	1	1 to 8
8	1	8	8 to 64
	2	4	4 to 32
	4[b]	2	2 to 16
16	1[b]	16	16 to 128
	4[b]	4	4 to 32

Table 1: Supported Hardware Configurations

a. Express links required
b. Meta-router required

Sparse configurations are allowed but it is important to remember that a partition is part of an entire system. For proper Craylink routing and fault isolation, the system must be cabled appropriately. The system must be configured using table Table 1: *Supported Hardware Configurations*, and locating the *Modules per Partition* number that is greater than or equal to the largest partition to be formed -That dictates the logical configuration. For example, to create a partitioned system with 3 modules in one partition, and

1 module in another partition, the system must be set up and cabled as if it were a sparsely populated 8 module system. Figure 5: *32-Processor Sparse Configuration* demonstrates the logical cabling for such a configuration.

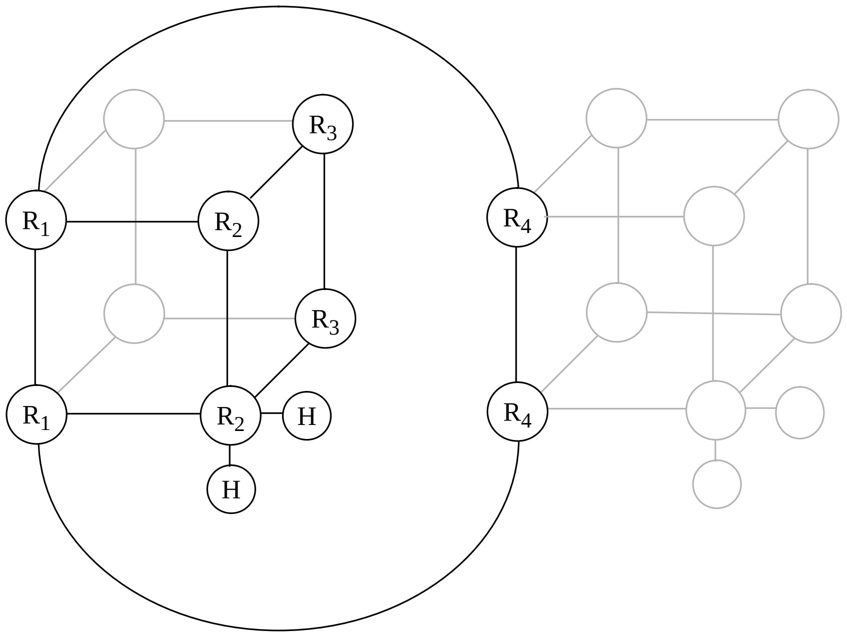


Figure 5: 32-Processor Sparse Configuration

Cray User Group Conference
June 15 - 19, 1998

4 Software Support

IRIX$_{(TM)}$ 6.5 contains several new software modules to deal with partitions and the communication between them. Two terms, *activate* and *de-activate*, are used to describe the detection of a new partition or the termination of a partition. When a remote partition is noticed and is running IRIX, it is said to be *activated*. Activation of a remote partition denotes noticing the remote partition, initializing all required data structures, spawning the required partition threads, and creating devices associated with that partition in the hardware graph. It does not mean that the remote partition has just booted (although that is usually the case), only that the local partition has noticed it. Likewise, de-activation of a remote partition means all threads associated with it are terminated and all associated data structures freed. It does not necessarily mean the remote partition has been powered down or reset. Only that for whatever reason, the remote partition is considered non-existent or un-reachable.

There are four main software modules used to support partitioning on IRIX: partition, XPC, if_cl, and cl. Other software components include the mkpd daemon and the mkpart command.

4.1 Partition

The partition module is primarily responsible for protecting the local partition by raising fire-walls around memory and other resources. It is also responsible for maintaining a map of all partitions in the entire system. When a remote partition is booted, the partition module *activates* it, meaning all other partition modules (such as XPC, if_cl, and cl) are notified of its existence. It also maintains a heart beat that allows remote partitions to know when the local partition has failed, and periodically scans all remote partitions heart-beats making sure they are operating. When a remote partition’s heartbeat terminates, the remote partition is *deactivated*, that is to say, all associated modules are notified it is no longer functioning, and all communication with it is terminated.

4.2 XPC

XPC, or the Cross Partition Communication module implements a low level message passing interface between partitions using the BTE (see *Section 2.6 Block Transfer Engine*). All cross partition communication done in the if_cl network driver (*Section 4.3 if_cl*) and the cl raw device driver (*Section 4.4 cl*) use this module for actual data transfer. The details of the implementation of XPC are beyond the scope of this paper, but it provides a reliable message based protocol using strictly BTE Pull operations and Cross call interrupts. This mechanism assures no cross partition cache pollution. XPC messages can contain a fixed amount of data in the message itself (immediate data), and/or a combination of immediate data and pointers to out-of-line data. The XPC interface also provides a message rendezvous facility, allowing a sender and receiver (located on different partitions) to rendezvous in the XPC subsystem. This allows upper layer drivers, such as cl, to avoid copying user data into intermediate buffers.

4.3 if_cl

The if_cl module provides a network driver interface to XPC. All normal IP network operations are supported over Craylink such as TCP/IP sockets, UDP, and NFS. Although XPC only supports point-to-point communication channels, the if_cl driver opens channels to all activated partitions, and routes packets appropriately.

4.4 cl

The cl module provides a raw device interface to XPC. These devices take advantage of the XPC rendezvous capabilities to allow DMA transfers directly from the source to the destination without any intermediate copying IF possible. The DMA operation is done using the BTE which supports transfers of multiple cache lines on cache line boundaries. Therefore, in order to avoid the overhead of the cl driver copying data to an intermediate buffer, the source and destination must be 128 byte aligned. If the transfer is NOT a multiple of 128 bytes, only the last part cache line will be copied through an intermediate buffer.

4.5 mkpd

The daemon mkpd runs on all partitions of a partitioned system. They communicate with each other sharing partition information and exchanging configuration information when the mkpart (see *Section 6.1 mkpart*) command is run. This daemon can be found running on Origin 2000 systems running IRIX 6.5 and provides a back end for the mkpart command. It generally is not directly accessible by the users or system administrator.

5 Software Configuration

Before using a partitioned environment, several partitioning modules must be configured into the kernel. IRIX 6.5 is shipped with these modules as part of the standard distribution CDs, which are installed on the root disk in */var/sysgen/boot*. The file names are partition.o, xpc.o, cl.o and if_cl.o. The default system generation file located in */var/sysgen/system/irix.sm* **EXCLUDE**s all four of these subsystems. To configure partitioning, the EXCLUDE statements in the irix.sm file must be changed to INCLUDE as seen below.

```
*
* Partition and cross partition communication subsystems.
*
EXCLUDE: partition
EXCLUDE: xpc
EXCLUDE: cl
EXCLUDE: if_cl
```

Change irix.sm to INCLUDE partitioning subsystems

```
*
* Partition and cross partition communication subsystems.
*
INCLUDE: partition
INCLUDE: xpc
INCLUDE: cl
INCLUDE: if_cl
```

To enable the memory protections between partitions, only the partition module is required; xpc, cl, and if_cl need not be included. However, to enable communication over the Craylink interfaces, the other modules are required (see *Section 4 Software Support* for a functional description of each of the modules). In general, all 4 of the partition related modules are desired and should be included.

Both the partition and xpc module are not generally visible at the application level. They provide the low level functionality used by the upper layers of software in cl and if_cl. To check if the partitioning software is installed, once IRIX is running execute the mkpart command as root. If partitioning is working, partition information is output; otherwise, the message following is displayed:

```
>mkpart
This system is not partitioned.
```

If the partition software is installed, the following is displayed:

```
>mkpart
Partition id = X
```

where X is the partition ID of the current partition. Partition ID 0 generally indicates the system is NOT partitioned, but the partitioning subsystem is installed.

5.1 Console Messages

Although other IRIX subsystems that use the partition and cross partition communication support may print their own messages, Table 2 *Partition Related Console Messages* lists the console messages from the partition and XPC subsystems and their meaning.

Prefix	Message	Description
	Partition Subsystem	
WARNING	Partition *X* not responding: Access(HB)	Partition heartbeat mechanism failed to access one of the required values in partition *X*. All of these messages are functionally equivalent, and generally mean the remote partition has been reset, powered off, or the Craylink connection to it has failed. XPC and related subsystems such as if_cl and cl are no longer usable to that partition.
	Partition *X* not responding: Access(PM)	
	Partition X not responding: Access(STATE)	
	Partition *X* not responding: Timeout(HB)	
	Partition *X* not responding: Removed(PM)	Partition *X* has failed to recognize the current partition, and has stopped heart-beating. XPC and related sub-systems such as if_cl and cl are no longer usable to that partition.
	XP for partition *X* is in partition *Y*	Indicates partition *X* has attempted to communicate but is broadcasting an invalid setup. The current partition will fail to recognize partition *X* until the situation is resolved. This message indicates a software failure in Partition *X*.
NOTICE	partition: Local Partition version *A.B* does not support remote version *C.D*	To allow for changes in partition support, all partitions broadcast their Partition support version number. If a remote partition is running software that is incompatible, this message will appear. The current partition will continue to function, but will not recognize the remote partition.
WARNING	NASID *N* partition[*X*] no longer responding	Node *N* previously belonged to partition *X*, but has stopped responding. It is similar in nature to the heart beat failures but can only occur while remote partitions are being discovered.
	XPC Subsystem	
NOTICE	Partition *X* does not support XPC messaging	Indicates partition *X* contains hardware that does not support XPC. The local partition will continue to run and protect itself from partition *X*, but will be unable to communicate with it.
	xpc: Remote partition *X* requested connection xpc: Local version *A.B* does not support Remote version *C.D*	The locally executing version of XPC is *A.B*, which is incompatible with the remote version *C.D*. The local partition will continue to run and protect itself from partition *X*, but will be unable to communicate with it.

Table 2: Partition Related Console Messages

6 User Interfaces

Partitioning support has three main user interfaces, mkpart to configure partitions, the raw devices which support a open/read/write interface, and a network interface that supports TPC/IP and UDP type sockets. A PROM interface to some of the partition support exists, but use of it is discouraged because it is not able to perform all the checks required to disallow invalid configurations. More information on configuring a partitioned system and the commands available can be found in the *IRIX Admin: System Configuration and Operation* manual in the chapter *Configuring the IRIX Operating System*.

6.1 mkpart

The mkpart command is used to configure and query the partitioning information. mkpart is shipped with IRIX 6.5 and includes a comprehensive man page. For detailed information on options to this command consult the manual page - mkpart (1M). The mkpart command provides 3 basic functions

1. Setting up or changing a partitioned configuration
2. Listing current partitions
3. Explicitly activating or deactivating partitions

mkpart provides a simple command line interface to set up partitions. Consider a system with 4 modules numbered 1, 2, 3, and 4. To partition the machine into two partitions with partition 1 being modules 1 and 2, and partition 2 being modules 3 and 4:

```
mkpart -p 1 -m 1 2 -p 2 -m 3 4
```

When complete, you will be prompted if you wish to reset the system. Partitioning changes only take affect on the next re-boot. At the next re-boot, the system will come up partitioned. To list the currently running configuration:

```
mkpart -l
```

Before the re-boot, the currently running configuration is returned. After the re-boot, if both partitions are running IRIX you will see:

```
partition: 1 = module: 1 2
partition: 2 = module: 3 4
```

To coalesce all partitions back into one, the -init option may be used. It is important to note that mkpart communicates with mkpd, which in turn uses configuration information from other mkpd's running in the other partitions. All partitions must be running IRIX for this command to provide the desired results.

```
mkpart -init
```

will undo the above mkpart command.

Finally, the mkpart command can be used to explicitly de-activate or activate partitions. When a partition is deactivated, all communication with it stops, meaning cross partition raw devices and network devices will not longer work. Using the previous example of 2 partitions, if the following command is issued in partition 1:

```
mkpart -d 2
```

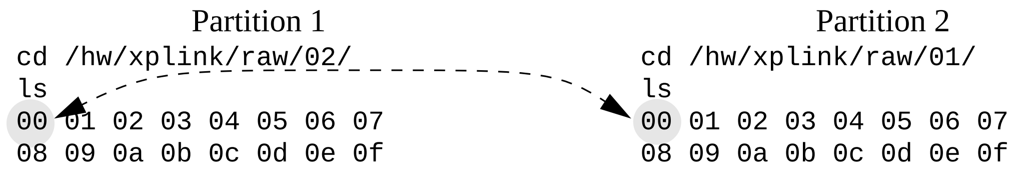
then partition 1 and partition 2 will no longer know about each other. There are a variety of reasons that the partition sub-system may look for remote partitions at different times. As long as partition 1 and 2 are running, they can, at any time, re-discover each other - although it is also possible that they will never rediscover each other unless explicitly told to check. The command

```
        mkpart -a
```

requests the partition subsystem attempt to re-establish communication with all possible partitions. It is not possible to re-establish communication with a specific partition. This command will find all partitions that are running IRIX.

6.2 Raw Devices

The cross partition raw devices are located in the directory /hw/xplink/raw/*remote_partition*/*device_number*, where *remote_partition* is the partition ID to which a device should be opened, and the *device_number* specifies which of the 16 possible devices to open. These devices appear in the hardware graph the first time a remote partition is discovered (or activated). They are never removed. For example, consider two partitions 1 and 2. Once both partitions are activated, devices will appear as follows:



```
cd /hw/xplink/raw/02/               cd /hw/xplink/raw/01/
ls                                  ls
00 01 02 03 04 05 06 07             00 01 02 03 04 05 06 07
08 09 0a 0b 0c 0d 0e 0f             08 09 0a 0b 0c 0d 0e 0f
```

Each partition contains 16 raw devices which can be opened to form point to point channels to remote partitions. If partition 1 opens /hw/xplink/raw/02/00 and partition 2 opens /hw/xplink/raw/01/00, the file descriptors may be used to exchange data with the remote partitions using read, write, and select. The devices provide a byte stream type interface.

6.3 Network Devices

If the if_cl network driver is installed, one network interface (cl0) is created. Unlike the raw devices, the network devices do not appear as point-to-point devices. That is, the network device has routes to each of the other partitions that are active and must be configured in the same manner as any other network device, using the file /etc/config/netif.options. For example, if the host foobar is one partition in a partitioned system, the network interface gate2-foobar can be configured to use the Craylink network interface by changing the netif.options file to:

```
#  To override the name and/or address of the first gateway interface,
#  change the value part and remove the leading : character.

if2name=cl0
if2addr=gate2-$HOSTNAME
```

The IP address for the cl interface must be assigned by adding it to the /etc/hosts file.

6.4 PROM Interface

The PROM interface can be accessed by selecting option 5 (*Enter Command Monitor*) in the system maintenance menu. In general, this interface should be used only as a last resort to recover from a failure in hardware or higher level partitioning software. A complete description of commands and their affects can be found in the *IRIX Admin: System Configuration and Operation* manual.

7 Performance

The performance numbers provided are run on a pre-release version of IRIX 6.5 software. The hardware configuration consisted of a 2 module, two partition Origin 2000. Each partition contained 4 node boards (8 processors) running at 195 MHz, with 4MB secondary caches. **THESE PERFORMANCE NUMBERS ARE PRELIMINARY AND FOR INFORMATIONAL USE ONLY. THEY ARE SUBJECT TO CHANGE AND ARE NOT A COMMITMENT.**

At first glance, the performance numbers may seem confusing as the raw device bandwidth is lower than the network devices. This can be explained by the fact the operations on a specific raw device will use only one BTE to transfer data at a given point in time, while operations on network devices can use multiple BTEs to transfer independent packets simultaneously. Using multiple raw devices, or multiple TCP/IP sockets can increase the aggregate bandwidth substantially.

7.1 Raw Device

The raw device performance numbers were obtained using a simply test program that opened the appropriate device, and then continuously issued 1000 reads in one partition and a 1000 writes in the other partition of the specified size. The bandwidth numbers listed in Table 3 *Raw Device Performance* is computed by dividing the total number of bytes sent by the total elapsed time (wall clock time). All transfers were cache-line aligned on the source (write end) and the destination (read end).

Read/Write Size (bytes)	MB/s
32	.8
64	1.5
128	2.8
256	1.2
512	2.4
1024	4.8
2048	9.5
4096	18.5
8192	34
16384	60
32768	95
65536	135

Table 3: Raw Device Performance

Read/Write Size (bytes)	MB/s
131072	170
262144	195
1048576	210

Table 3: Raw Device Performance

7.2 TCP/IP

TCP/IP performance runs were done using the TTCP command. The shaded row in the table indicates the default ttcp buffer size (i.e. no -b parameter was issued on the command line).

Receiver Command	ttcp -r -s -l49152 -b <socket buffer size> -n 126976
Sender Command	ttcp -t -s -l49152 -b <socket buffer size>-n 126976 <ip-address>

Socket Buffer Size (Bytes)	Transmit	Receive
1048576	200+ MB/s	200+ MB/s
524288	200+ MB/s	200+ MB/s
262144	170+ MB/s	170+ MB/s
131072	95+ MB/s	95+ MB/s
65536	55+ MB/s	55+ MB/s

Table 4: TCP/IP Performance

7.3 UDP/IP

UDP/IP performance runs were done using the TTCP command. The shaded row in the table indicates the default ttcp buffer size (i.e. no -b parameter was issued on the command line).

Receiver Command	ttcp -u -r -s -l49152 -b <socket buffer size> -n 126976
Sender Command	ttcp -u -t -s -l49152 -b <socket buffer size>-n 126976 <ip-address>

Socket Buffer Size (Bytes)	Transmit	Receive	% data dropped
1048576	270+ MB/s	270+ MB/s	< 1%
524288	266+ MB/s	263+ MB/s	< 1%

Table 5: UDP/IP Performance

Socket Buffer Size (Bytes)	Transmit	Receive	% data dropped
262144	270+ MB/s	255+ MB/s	5.5%
131072	275+ MB/s	220+ MB/s	20%
65536	275+ MB/s	160+ MB/s	45%

Table 5: UDP/IP Performance