Installation and Configuration of UNICOS/mk on a Cray T3E

David C. Holst, Cray Research, A Silicon Graphics Company

ABSTRACT: *The need to support software installations across several platforms on the new GigaRing has necessitated changes for loading media. Serverized Unicos (UNICOS/mk) requires a change in the way that the T3E will be configured. Finally, T3E systems will be shipped with pre-installed software. This document discusses installation and configuration on the Cray T3E and UNICOS/mk.*

1 Introduction

Software installation and configuration under UNICOS/mk has seen considerable changes over the previous process under UNICOS. This document is divided into two major sections. The first section is a high level overview of installation and configuration. The second section contains more detailed information regarding specific configuration areas.

2 Section 1: High Level Overview

There are four main themes regarding system installation and configuration.

- What is different with the Cray T3E and UNICOS/mk compared to previous Cray systems and UNICOS?
- What is Cray doing to my system before shipment?
- What should I do with my Cray system when it arrives?
- How do I upgrade my system.

2.1 What is the Difference Between Cray T3E and UNICOS/mk from Previous Cray Systems and UNICOS?

The primary difference is that kernel configuration has been split from non-kernel configuration. Under UNICOS, kernel configuration items were kept in the CSL Param file on the Operator WorkStation (OWS), in various header files in /usr/src/uts/cf.xxxx/*.h, and in the Install Tool.

For UNICOS/mk, all of the kernel configuration was combined into a file (called the Configuration File) on the SWS.

The UNICOS/mk non-kernel information, consisting of system daemons, networks, tapes, etc. was left in place. The Install Tool has been replaced with a follow-on product, called the Configuration Tool.

2.2 What is Cray doing to my System Before it Ships?

All Cray T3E systems will be pre-installed with a minimal system. The pre-installed system should provide a quick and clean starting point for getting the systems running for production. The customer can be guaranteed that the system has been booted prior to shipment. After the machine has been assembled, all of the Cray T3E systems should boot with minimal changes.

The Systems WorkStation (SWS) will be pre-installed with Solaris 2.5, and the SWS-ION package. The SWS-ION package contains all of the Cray related software that allows the SWS to control and access Cray hardware, such as the Gigaring and the T3E.

The SWS will have two bootable resources defined, one named after the serial number of the T3E (snxxxx), the second is called root_bu. These resources are used to boot the T3E.

The CYRIos package will also be installed on the SWS. The CYRIos package contains UNICOS/mk information required by the SWS to boot the T3E. This includes the Chorus Archive (kernel) located in /opt/CYRIos/1.0/archive. The CYRIos package also includes two configuration files, one named after the T3E serial number (snxxxx.conf) and the other named root_bu.conf.

The configuration files describe all of the kernel related configuration information for the T3E. Each configuration file is also directly referenced by the default bootable resource, mentioned above.

The configuration files describe a minimal hardware and software configuration for all Cray T3E systems. This minimal configuration provides a basis for a system administrator to start with and build upon.

Copyright © 1996. Cray Research, A Silicon Graphics Company. All rights reserved.

2.2.1 *Minimal Hardware Configuration:*

The pre-installed configuration files describe a minimal hardware configuration that will be common to all Cray T3E systems. This minimal hardware configuration consists of a...

- Cray T3E
- SWS
- MPN

The MPN has either an Ethernet board or a FDDI board, and at least one SCSI board. The SCSI board has two 4 Gbyte drives attached.

2.2.2 *Minimal Filesystem Layout*

The configuration files (snxxxx.conf, root_bu) define a minimal set of filesystems spread out on the two 4Gbyte disk drives. These filesystems should provide the minimum required for booting a system to multi-user mode.

We strongly recommend that the customer not modify the root and usr filesystems, since they should provide a system that will always boot.

2.3 What Should I Do with my System Now That I've Got It?

One of the more difficult configuration tasks after a Cray T3E system arrives is to determine where to start. This section will start by quickly discussing a few items that can be completed in preparation of the arrival of the system. The second area to be addressed discusses configuration of a UNICOS/mk kernel, including disk and PE configuration. Finally, this section will discuss configuring non-kernel items.

2.3.1 *Before Your Cray T3E System Arrives...*

The disk and PE configuration can be determined can be determined in advance of the systems arrival. In regards to disk configuration, a complete layout of disks and filesystems can definitely be determined. Determining the layout of the physical disks, physical disk slices, and logical filesystems is very similar to the procedure under UNICOS.

The PE configuration will probably be easier then any initial impressions developed from looking at the PE configuration in the configuration file. There are two major divisions in the PE configuration: Primary PEs and Support PEs.

Primary PEs are the main PEs with which the customer will interact. If a customer buys a machine with x (ex: 128) PEs, then they will have x (ex 128) primary PEs. Primary PEs can be either Application (APP) PEs or Command (CMD) PEs. Application PES (also known as parallel PEs) are used for parallel jobs. Command PEs (also known as serial PEs) are used for single PE tasks, such as login shells, system daemons, etc.

Support PEs are extra PEs that Cray ships with the T3E. The number of support PEs shipped depends on the number of Primary PEs shipped. A 64 PE system may have 4 support PEs, for a total of 68 PEs. A 128 PE system may have 8 support PEs, for a total of 136 PEs (note, these numbers are here as an example, and may not be entirely accurate). Support PEs can be either Operating System PEs (OS PEs) or spare (redundant) PEs.

The final configuration item regarding PE configuration are known as Actor Lists. An Actor list is a list of kernel related servers (such as a packet server, disk server, etc). Each PE has an associated Actor List, describing the various servers that will be running on that particular PE.

PE configuration is mostly a decision of how many Application and Command PEs should be defined on the system. Cray strongly recommends that customers use the default OS PE and actor list configurations.

2.3.2 *Configuring the UNICOS/mk Kernel: Editing the Configuration File.*

2.3.2.1 *The Configuration File Format*

The Configuration File is located on the SWS and contains all of the kernel related configuration of UNICOS/mk. The file can be located in the /opt/CYRIos/config directory. The configuration file usually has a ".conf" ending (snxxxx.conf, root_bu.conf).

The format of the configuration file is a subset of the C programming language, with a few variations. Anyone familiar with C, or any other programming language, should be able to figure out the format.

There have been several requests from the user community to provide a format that is more closely related to the CSL format of the UNICOS param file. A new format, closely related to the CSL format on the UNICOS param file, has been developed. It will be released with UNICOS/mk 1.3.

The formatcs command allows the customer to convert a configuration file from one format to the other. There are only two programs that read the configuration file, pact and csp. Pact is an editor specifically designed for modifying the configuration file. csp is the Configuration Server Proxy, a daemon that runs on the SWS and is responsible for transmitting the configuration file to the T3E during boot. Both Pact and csp will be able to read a configuration file in either format (old and new).

A configuration file can be verified using the csv command.

2.3.2.2 *Pact*

Pact is a GUI tree based editor of the configuration file. Pact resides and runs on the SWS. Pact has been postponed until 11/15/96 for the addition of task-based functions and performance enhancements.

Pact is available on the SWS platforms now, but I suggest only using it for viewing the data organization and simple changes.

2.3.3 *Disk Configuration*

Dynamic disk configuration will be available on the T3E with the UNICOS/mk 1.3 release. The mknod command will be available, and disk configuration will be determined by the nodes defined on the root filesystem. This will provide the same functionality that we currently have on UNICOS systems.

As with UNICOS systems, the nodes on the root filesystem will take precedence over any configuration information in the kernel.

When dynamic configuration is available, the configuration file will only need configuration information about the root filesystem (including the disk, xdd slice, ldd slice, and rootdev entries). Unlike UNICOS systems, a definition for the swap device will not be required.

On the T3E, the fconfig command is available to read all of the disk configuration out of the configuration file and write out all of the respective disk nodes.

2.3.4 PE Configuration

The PE configuration is completely defined in the configuration file. The major decision that has to be made is how many Application versus Command PEs should be available.

Application PEs should always be placed first in the list. For performance reasons, the goal is to maintain as large of a 3D torus as possible.

There is a requirement for a minimum of one command PE. This brings up an unfortunate side-effect. A site with x (ex: 128) PEs can have at most x-1 (ex: 127) Application PEs.

Cray currently has no recommendation regarding how many Command PEs should be configured. This may very considerably from site to site, depending on the usage and work load. Cray is very interested in any data that a customer may wish to provide.

In regards to Support PE configuration, the customer should use the Cray recommended configuration. There is currently a limit of two OS PEs. Developers are working on larger configurations, which should be available soon. Any non-OS Support PEs are spare (redundant) PEs.

The Actor List configuration should not be changed from the defaults that Cray provides. The standard actor list configuration will be required for Cray to read system dumps.

2.4 Non-UNICOS/mk Kernel Configuration

The paragraphs above describe how to configure kernel related configuration items. Non-kernel related items are configured on the T3E using a new tool, called the Config Tool. The Config Tool is used for configuring non-kernel related items, such as system daemons, tapes, networks, etc. It is a follow-on product to the UNICOS Install Tool.

The Config Tool is subsystem oriented. With the removal of all of the kernel related information in the Install Tool, we found that all of the remaining configuration items were separate, having no dependencies or interaction with the other configuration items. Customers familiar with the previous Install Tool should find it easier to navigate through and find the items that they wish to work on.

The Config Tool modifies the configuration files directly. There is no database as was in the previous Install Tool. The database caused difficulties to many sites since it was easy to get it out of sync with the running system.

The Config Tool has a review option that displays a side-by-side comparison of all of the configuration changes. The review is a good secondary check to ensure all of the changes were made correctly.

2.5 How do I Upgrade My System?

The upgrade process has been greatly simplified. System upgrades will employ the “Alternate Root Method”. This method involves making a copy of the current running root and usr onto another root/usr filesystem, then performing the upgrade on the copy. This provides a guaranteed backup (the original filesystem is untouched).

- Copy the current root/usr filesystem to an empty root/usr filesystem.
- Run setup on the UNICOS/mk CD-ROM.
- Common Install Tool (CIT) will start.

CIT has three basic steps.
1: Verify SWS and CRAY T3E communication Paths
2: Select the UNICOS/mk product
3: Click install

While the upgrade process is running, you may have to update some of the SWS configuration in order to get a resource that will boot the new root.

- Copy configuration file /opt/CYRIos/1.0/config/snxxxx.conf
- Copy the SWS T3E Cray Resource (snxxxx)

After the upgrade is complete, shutdown the system and reboot.

2.6 The Common Install Tool (CIT)

CIT is a GUI based tool used to install products on remote machines. CIT currently is used for (or will be used for) installing and upgrading

- SWS-ION Package
- UNICOS/mk
- Async Packages (compilers, etc)
- UNICOS 9.2+

One of the biggest improvements in CIT over previous installation routines is that CIT supports compressed packages. The largest data transference bottleneck is the speed of the CDROM drive. We have been seeing a 70% compression rate on the UNICOS/mk package. WIth only 1/3 of the data to transfer, it is taking about 1/3 of the time.

CIT runs on the SWS and the OWS. With minimal investment, CIT could be expanded to run on just about any platform.

2.7 Summary of High Level Overview

There are several improvements over previous methods of installation and configuration.

- Installation is separate from configuration
- Kernel configuration is in one location
- There is no need to recompile or relink the kernel when changing a configuration item.
- Pre-installation:

  - SWS Solaris 2.5
  - SWS-ION package
  - SWS Cray T3E Resource Configuration

- Cray T3E UNICOS/mk

- Upgrade and Install process simplified (and faster)

3 Technical Stuff

This section contains quite a bit of information and is targeted at system administrators, or anyone else who will be delving into the deep dark world of configuration. The items are loosely related, and loosely presented. The information here is by no means complete, but consists of the most common commands and options that I personally use when working on system installs and configurations. You may notice that this section is less formal then the previous section. I may insert a few opinions on occasion.

As a final caveat, the Cray T3E is quickly evolving and improving. What is true today (or at the time I am writing this) may not be true tomorrow (or at the time you are reading this).

3.1 The MPN and You.

The MPN has a few neat features that administrators might want to take advantage of. First, you can log into it from the SWS using rlogin (rlogin snxxxx-mpn0).

Why would you want to log into the MPN? It shows MPN errors live.

There is one command on the MPN that I use.

- reset: Resets the MPN.

Warning, this may seriously hose any system that is up and using the MPN.

When a MPN is reset, it displays all information regarding all of the devices that is attached to it.

```
SCSIbus 4 - detected
SCSIbus 5 - detected
SCSIbus 6 - detected
SCSIbus 7 - detected
Ethernet 0 - detected
Ethernet 2 - detected
SCSIbus 4 Target 0 LUN 0 [s400], Type = DD314
SCSIbus 4 Target 1 LUN 0 [s410], Type = DD314
SCSIbus 4 Target 2 LUN 0 [s420], Type = DD314
SCSIbus 4 Target 3 LUN 0 [s430], Type = DD314
SCSIbus 5 Target 0 LUN 0 [s500], Type = DD314
SCSIbus 5 Target 1 LUN 0 [s510], Type = DD314
SCSIbus 5 Target 2 LUN 0 [s520], Type = DD314
```

In the preceding example, you may notice that there are two ethernet boards in slot 0 and 2, and that slots 4, 5, 6, and 7 each contain a SCSI board. SCSI slot 4 has 4 DD314 disk drives attached to it.

3.2 More MPN stuff

Each time the MPN resets, a file is created such as /opt/CYRIion/adm/mic-code.snxxxx-mpn0. This file contains a list of all disk devices attached to the MPN. One of the more useful sections of this file is the drive serial number. I recommend making a copy of this file in case the disk drives ever get reshuffled. With this file you can give the engineers the exact drive serial number and drive location in order to unshuffle the drives.

```
File Created 09/17/96, 01:33:24, at MPN load time.
Node sn6301-mpn0, IOP Microcode Rev Level
Cray Research Inc. MPN-1 PROM 2.0 96/06/17 15:11:25 Part #13290600
sn6301-mpn0, SCSIbus 4 - detected
    Product ID = SCS-10 SCSI Adapter
    Microcode Rev Level = 1-17; Initiator Target ID 7
sn6301-mpn0, SCSIbus 5 - detected
    Product ID = SCS-10 SCSI Adapter
    Microcode Rev Level = 1-17; Initiator Target ID 7

sn6301-mpn0  SCSIbus 4 Target 0 LUN 0 [s400], Type = DD314
    Product ID = ST15150W       , Microcode Rev Level = 2100
    Vendor = SEAGATE
    Attributes = Supports Tags, Supports Wide, Supports Auto Flawing
    Serial Number = 01337138

sn6301-mpn0  SCSIbus 4 Target 1 LUN 0 [s410], Type = DD314
    Product ID = ST15150W       , Microcode Rev Level = 2100
    Vendor = SEAGATE
    Attributes = Supports Tags, Supports Wide, Supports Auto Flawing
    Serial Number = 01336054
```

3.3 SWS Configuration : Resource Command

The resource command on the SWS is used to define and control various hardware and software resources. A resource may be defined as a Cray T3E and an associated MPN.

The three most popular ‘resource’ commands are.....

- resource status: Lists all top level resources and their status
- resource start snxxxx: Boots the snxxxx resource.
- resource stop snxxxx: Stops the snxxxx resource

```
sws%  resource status
hubble[SWSm_T3E_System_Resource] is reserved and Functional
sn6546-mpn0[SWSm_ION_System_Resource] is reserved and Functional
hubble-fi[SWSm_T3E_System_Resource] is unreserved
```

3.4 SWS Configuration: swsconfig command

Although there are many different types of resources to configure, the one that you will spend the most time are the T3E related resources.

The t3e_resource is the top level resource for the T3E. It has some data fields in it, and two sub-components called t3e_software and t3e_hardware.

The swsconfig command is used to display, copy, modify, and delete various resources.

3.4.1 Displaying t3e_resource resources

```
swsconfig display t3e_r hubble
autoboot:              false
autodump:              false
hardware_component:    hubble_mf
hostname:              hubble:137.38.228.7:0.0.0
```

```
min_run_time:          1200
name:                  hubble
proxy_port:            470
software_component:    hubble_os
type:                  t3e_resource
```

The example above is a machine installed at Cray, called hubble. The T3E resource called hubble has a software component called hubble_os, and a hardware component called hubble_mf.

3.4.2 Displaying t3e_software resources

```
sws%  swsconfig display t3e_s hubble_os
Supports_Heartbeat:   false
archive:              /opt/CYRIos/1.0/archive/unicosmk.cray-t3e
bootpe:               135
configfile:           /opt/CYRIos/1.0/config/param.conf
console:              mfcon
dump_directory:       /opt/CYRIdump/umk
dump_mode:            all-sws
mainframe_id:         mf[0]
maint_hostspec:       sn6546-mpn0:470
maintenance_boot:     false
maintpe:              65535
mkpal:                /opt/CYRIos/1.0/mkpal/mkpal.cray-t3e
name:                 hubble_os
ram_filesystem:
run_level:            s
skdb_hostspec:        localhost:54321
truthpe:              1
type:                 t3e_software
```

The bootpe is important to check to make sure it matches the entry in the configuration file. The configfile entry is probably the most often changed.

3.4.3 Displaying t3e_hardware resources

```
sws% swsconfig display t3e_h hubble_mf
boot_pal:          /opt/CYRIdiag/t3e/t3esys/tst/hdw_boot.uv
cabinet_type:      0
gigaring_address:  hubble_mf_00.15:00.15
name:              hubble_mf
pe_count:          20
pe_list:           PE000000,PE000001,PE000002,PE000003,PE000004,
PE000005,PE000006,PE000007,PE000008,PE000009,
PE00000a,PE00000b,PE00000c,PE00000d,PE00000e,
PE00000f,PE000010,PE000011,PE000012,PE000013
type:              t3e_hardware
```

The pe_count should match the configuration defined in the configuration file.

3.4.4 Copying Resources

In order to copy a resource, use the copy option of the swsconfig command. It is important to note that the subcomponents of both resources will be THE SAME. Therefore, if the software subcomponent of hubble is hubble_os, then the software component of dh will also be hubble_os.

```
sws%  swsconfig copy t3e_r hubble dh
sws%  swsconfig copy t3e_s hubble_os dh_os
```

3.4.5 Modifying Resources

In order to modify a resource, use the modify option of the swsconfig command. This can be used to change a single data value, or can be used to reset the subcomponents of a specified resource.

```
sws%  swsconfig modify t3e_r -software dh_os dh
```

The above command looks for the t3e_r resource dh, and changes the software subcomponent to dh_os. dh_os must already exist.

```
sws%  swsconfig modify t3e_s -bootpe 225 dh_os
```

The above command changes the bootpe of the dh_os software resource. By the way, I suggest not changing the bootpe unless you really know what you are doing.

In both cases, the component that you want to modify (-bootpe) is a component listed from the swsconfig display command, with a "-" in front of it.

3.5 Boot Process: *mfcon*

You may remember that the resource command is used to boot a T3E.

```
sws%  resource start snxxxx
```

The above command will boot a T3E called snxxxx. During the boot process, an xterm will appear. This xterm will be running a command called mfcon. mfcon provides access from the SWS to the T3E system console (mainframe console). mfcon is analogous to zip on the OWS systems.

```
sws%  mfcon -l 0 snxxxx
```

The above command will open a console connection to a mainframe that has been booted. The -l 0 option specifies which line to use. The snxxxx refers to the resource that was booted (resource start snxxxx).

3.6 Configuration File

The first section of this document ("High Level Overview") described some of the issues regarding configuration and the configuration file. This section will provide more information about many aspects of the configuration file.

3.6.1 Format:

The default format of the configuration file resembles C code. Following is an example...

```
mainframe_t     mf[1];
disk_max_t      diskdriver[1];
diskdriver[0].pddmax     = 32;
diskdriver[0].xddmax     = 256;
diskdriver[0].pddslmax  = 256;
diskdriver[0].rddslmax   = 4;
diskdriver[0].xddslmax  = 256;
mf[0].drivers.diskdriver[0] = diskdriver[0];
```

We (Cray) have received many requests from individuals to have a format similar to the old CSL format on the OWS. There is now an optional format available for viewing and editing the configuration file. After using up most of our creativity coming up with names like "Config Tool", and "Common Install Tool", we decided to take a wild leap and name the new format "The New Format". It looks something like this...

```
mainframe_t mf[1];
mf[0] {
    drivers {
        diskdriver[0] {
                pddmax = 32;
                xddmax = 256;
                pddslmax = 256;
                rddslmax = 4;
                xddslmax = 256;
            }
        }
}
```

To repeat some items in the first section of this document, the formatcs command is used to convert a configuration file between the two formats. Both Pact and the Configuration Server Proxy can read a configuration file in either format.

3.6.2 Physical Disks:

Following is an example of the configuration of two scsi disks.

```
pdisk[0].name          = "dscsi.s0140";     /* s400 */
pdisk[0].ptype         = d_disk;
pdisk[0].stype         = d_x500;
pdisk[0].iopath.ioc    = 0;                 /* ioc = Ring */
pdisk[0].iopath.iop    = 1;                 /* iop = MPN */
pdisk[0].iopath.chan   = 4;                 /* chan = MPN slot */
pdisk[0].unit          = 0;  /* SCSI ID */  /* unit = Unit */
pdisk[0].disk_pe       = "ospe_a";

pdisk[1].name          = "dscsi.s0141";     /* s410 */
pdisk[1].ptype         = d_disk;
pdisk[1].stype         = d_x500;
pdisk[1].iopath.ioc    = 0;
pdisk[1].iopath.iop    = 1;
pdisk[1].iopath.chan   = 4;
pdisk[1].unit          = 1;  /* SCSI ID */
pdisk[1].disk_pe       = "ospe_a";
mf[0].disk_info.phys_stor[0]   = pdisk[0];
mf[0].disk_info.phys_stor[1]   = pdisk[1];
```

There are several items of interest in the above configuration.

1. iopath.ioc is a reference to the Ring that the drive is on.
2. iopath.iop is a reference to the MPN that the drive is on.
3. iopath.chan is a reference to the SLOT that the drive is on.
4. unit is SCSI ID number of the drive.

There is a new field that might be unfamiliar, disk_pe. disk_pe describes the pe that has the disk server on that knows how to access this drive.

The last two items (mf[0].disk_info.phys_stor[0]....) actually create the link between the defined pdisk, and the top level mainframe object. If this link does not exist, then this drive will not be available. Every object in the configuration file eventually has to be linked to the mainframe object, or it will not be available.

Is anyone still here?

3.6.3 XDD Slices

The xdd slice is analogous to the pdd slice under UNICOS. The xdd slice is configured just like pdd slices.

```
xdd[0].minor         = 1;
xdd[0].name          = "root";
xdd[0].start         = 0;
xdd[0].length        = 200000;
xdd[0].length_unit   = blocks;
xdd[0].pstor         = &mf[0].disk_info.phys_stor[0];
xdd[5].minor         = 6;
xdd[5].name          = "dump_s0100";
xdd[5].start         = 1000000;
xdd[5].length        = 102000;
xdd[5].length_unit   = blocks;
xdd[5].pstor         = &mf[0].disk_info.phys_stor[0];
xdd[11].minor        = 15;
xdd[11].name         = "dump.s0101";
xdd[11].start        = 1000000;
xdd[11].length       =  102000;
xdd[11].length_unit  = blocks;
xdd[11].pstor        = &mf[0].disk_info.phys_stor[1];

mf[0].dev.xdd[0]     = xdd[0];
mf[0].dev.xdd[5]     = xdd[5];
mf[0].dev.xdd[11]    = xdd[11];
```

The slices listed above describes a root filesystem, and two slices that will be used to create a dumps filesystem. The most important item to look for is the pstor field. The pstor field must refer to a specific physical disk that has already been linked in. You may note that the root xdd slice has the following pstor...

```
xdd[0].pstor         = &mf[0].disk_info.phys_stor[0];
```

Note that the phys_stor is the same as...

```
mf[0].disk_info.phys_stor[0]   = pdisk[0]
```

Therefore, the root xdd slice is located on SCSI drive dscsi.s0140.

Another item to note is that the two dump slices are on different disks (phys_stor of 0 and 1).

3.6.4 ldd slices

Logical devices are again similar to logical device under UNICOS.

```
ldd[0].name          = "root";
ldd[0].minor         = 1;
ldd[0].nslices       = 1;
ldd[0].member[0].mstor.phys = &mf[0].dev.xdd[0];
ldd[0].member[0].mtype = m_xdd;

ldd[5].name          = "dump";
ldd[5].minor         = 6;
ldd[5].nslices       = 2;
ldd[5].member[0].mstor.phys = &mf[0].dev.xdd[5];
ldd[5].member[0].mtype = m_xdd;
ldd[5].member[1].mstor.phys = &mf[0].dev.xdd[11];
ldd[5].member[1].mtype = m_xdd;
mf[0].dev.dsk[0]     = ldd[0];
mf[0].dev.dsk[5]     = ldd[5];
```

The root filesystem has one member slice which points to xdd 0, the root xdd slice, which again points to dscsi.s0140.

The dump filesystem has two slices, xdd 5 and 11.

3.6.5 System Devices

The configuration file has two system device entries, defining the root and swap that is to be used during the boot process. The swap section may not be necessary, and may disappear soon.

```
/* system devices */
rootdev.name    = “root”;
rootdev.minor   = 1;
swapdev.name    = “swap”;
swapdev.minor   = 5;
```

The minor numbers are actually what is important. You must make sure that the name and the minor number are pointing to the correct LDD device.

3.6.6 Network Configuration

```
/* Network devices */
nw_dev[0].iopath.ioc    = 0;          /*  ioc = Ring */
nw_dev[0].iopath.iop    = 1;          /* iop = MPN */
nw_dev[0].iopath.chan  = 0;          /*  chan = MPN slot */
nw_dev[0].ordinal        = 0;
nw_dev[0].type           = nw_fddi;
nw_dev[0].maxusers     = 1;
nw_dev[0].maxoutputs  = 16;
nw_dev[0].maxinputs    = 16;
nw_dev[0].othreshold    = 0;
nw_dev[0].ithreshold     = 0;
nw_dev[0].mtu              = 4352;
```

Note that the network configuration has the same confusion as the disk configuration by referring to ioc, iop, and chan. The comments in the section above have the correct meanings.

This problem may be corrected some time in the future. I know that there is at least one SPR open on this problem.

3.6.7 Configuring PEs

I will freely admit that there are elements of the PE configuration that I am not overly familiar with. This lack of knowledge is somewhat offset by the System Administrator Rule Number One:

“If you don’t know what it does, don’t change it.”

This rule is a corollary of,

“If it works, don’t fix it.”

There is a third rule, provided by some internal software developers:

“If you want us to read a dump, keep the default configuration.”

3.6.7.1 Server Routes:

```
routes[0].io_contr_lpe = 268;
routes[0].gigaring_node_addr = 15;
routes[0].ringno    = 0;
routes[0].link_type = link_scx;
routes[0].path_type = primary;
routes[0].intr_ps_name = “ospe_b”;
routes[0].send_ps_name = “ospe_b”;
server_routes[0].ringno = 0;
server_routes[0].server_name = “disk”;
server_routes[0].server_pe_name = “ospe_b”;
server_routes[0].ps_pe_name = “ospe_b”;
server_routes[1].ringno = 0;
server_routes[1].server_name = “netdev”;
server_routes[1].server_pe_name = “ospe_b”;
server_routes[1].ps_pe_name = “ospe_b”;
server_routes[2].ringno = 0;
server_routes[2].server_name = “tty”;
server_routes[2].server_pe_name = “ospe_a”;
server_routes[2].ps_pe_name = “ospe_b”;
```

The io_contr_lpe should point to the OS PE that contains the disk server. You may notice that there are a lot of entries for “ospe_b”. If you check the actor list for ospe_b, you will find that one of the actor list entries has the server associated with the entry (disk server, network, and tty)

3.6.7.2 Actor Lists:

There are four actor lists described in the next section. The first one for CMD PEs, the second for APP PEs, and the third and fourth describe OSPEs.

```
/* pe_actors index 0 -- Command PEs */
mf[0].mpp.pe_actors[0].num_actors = 4;
mf[0].mpp.pe_actors[0].actor_name[0] = “kernel”;
mf[0].mpp.pe_actors[0].actor_name[1] = “PM”;
mf[0].mpp.pe_actors[0].actor_name[2] = “fsa”;
mf[0].mpp.pe_actors[0].actor_name[3] = “em”;
/* pe_actors index 1 -- Application PEs */
mf[0].mpp.pe_actors[1].num_actors = 4;
mf[0].mpp.pe_actors[1].actor_name[0] = “kernel”;
mf[0].mpp.pe_actors[1].actor_name[1] = “PM”;
mf[0].mpp.pe_actors[1].actor_name[2] = “fsa”;
mf[0].mpp.pe_actors[1].actor_name[3] = “em”;
```

Two OSPE system, First (boot) OS PE

```
/* pe_actors index 3 -- OS PE -- two-OS-PE system -- first (boot) OS PE */
mf[0].mpp.pe_actors[3].num_actors = 11;
mf[0].mpp.pe_actors[3].actor_name[0] = “kernel”;
mf[0].mpp.pe_actors[3].actor_name[1] = “em”;
mf[0].mpp.pe_actors[3].actor_name[2] = “PM”;
mf[0].mpp.pe_actors[3].actor_name[3] = “config”;
mf[0].mpp.pe_actors[3].actor_name[4] = “info”;
mf[0].mpp.pe_actors[3].actor_name[5] = “log”;
mf[0].mpp.pe_actors[3].actor_name[6] = “grm”;
mf[0].mpp.pe_actors[3].actor_name[7] = “alm”;
mf[0].mpp.pe_actors[3].actor_name[8] = “GPM”;
mf[0].mpp.pe_actors[3].actor_name[9] = “tty”;
mf[0].mpp.pe_actors[3].actor_name[10] = “UP_INIT”;
```

Two OS PE System, Second PE.

```
/* pe_actors index 4 -- OS PE -- two-OS-PE system -- second OS PE */
mf[0].mpp.pe_actors[4].num_actors = 10;
mf[0].mpp.pe_actors[4].actor_name[0] = “kernel”;
mf[0].mpp.pe_actors[4].actor_name[1] = “em”;
mf[0].mpp.pe_actors[4].actor_name[2] = “PM”;
mf[0].mpp.pe_actors[4].actor_name[3] = “packet”;
mf[0].mpp.pe_actors[4].actor_name[4] = “disk”;
mf[0].mpp.pe_actors[4].actor_name[5] = “netdev”;
mf[0].mpp.pe_actors[4].actor_name[6] = “file”;
```

```
mf[0].mpp.pe_actors[4].actor_name[7] = “socket”;
mf[0].mpp.pe_actors[4].actor_name[8] = “sio”;
mf[0].mpp.pe_actors[4].actor_name[9] = “tape”;
```

3.6.7.3 Actual PE Configuration

Each PE has a configuration entry.

Application (APP) PEs:

```
mf[0].mpp.pe[0].flags = 0;
mf[0].mpp.pe[0].pe_index = 0;
mf[0].mpp.pe[0].pe_type = PE_TYPE_APP;
mf[0].mpp.pe[0].actor_list_index = 1;
mf[0].mpp.pe[0].hz = 75000000;
```

Command (CMD) PEs:

```
mf[0].mpp.pe[256].flags = 0;
mf[0].mpp.pe[256].pe_index = 256;
mf[0].mpp.pe[256].pe_type = PE_TYPE_CMD;
mf[0].mpp.pe[256].actor_list_index = 0;
mf[0].mpp.pe[256].hz = 75000000;
```

Note that the actor_list_index of each entry matches the index of ...mpp.pe_actors[0]... as defined under the actor list section.

Operating System (OS) PEs:

```
mf[0].mpp.pe[268].pe_name = “ospe_a”;
mf[0].mpp.pe[268].flags = 0;
mf[0].mpp.pe[268].pe_index = 268;       /* Hi there, I’m an OS PE (0x10c) */
mf[0].mpp.pe[268].pe_type = PE_TYPE_OS;
mf[0].mpp.pe[268].actor_list_index = 3;
mf[0].mpp.pe[268].hz = 75000000;

mf[0].mpp.pe[269].pe_name = “ospe_b”;
mf[0].mpp.pe[269].flags = 0;
mf[0].mpp.pe[269].pe_index = 269;       /* Me too (0x10d) */
mf[0].mpp.pe[269].pe_type = PE_TYPE_OS;
mf[0].mpp.pe[269].actor_list_index = 4;
mf[0].mpp.pe[269].hz = 75000000;
```

The OS PEs each have a different actor list, and a name field with “ospe_a” and “ospe_b”. The values in the name field must match values defined earlier (such as with the physical disk field “pdisk”).

That’s all for now. Enjoy!

David C. Holst (dh@cray.com)