Cray XR1™ User and Administration Guide

S–2486–10

Version 1.0 Published July 2009 Supports general availability (GA) release of the Cray Linux Environment (CLE) 2.2 operating system running on Cray XT systems.

Contents

Page

Figures

Cray XR1 Hardware Overview [1]

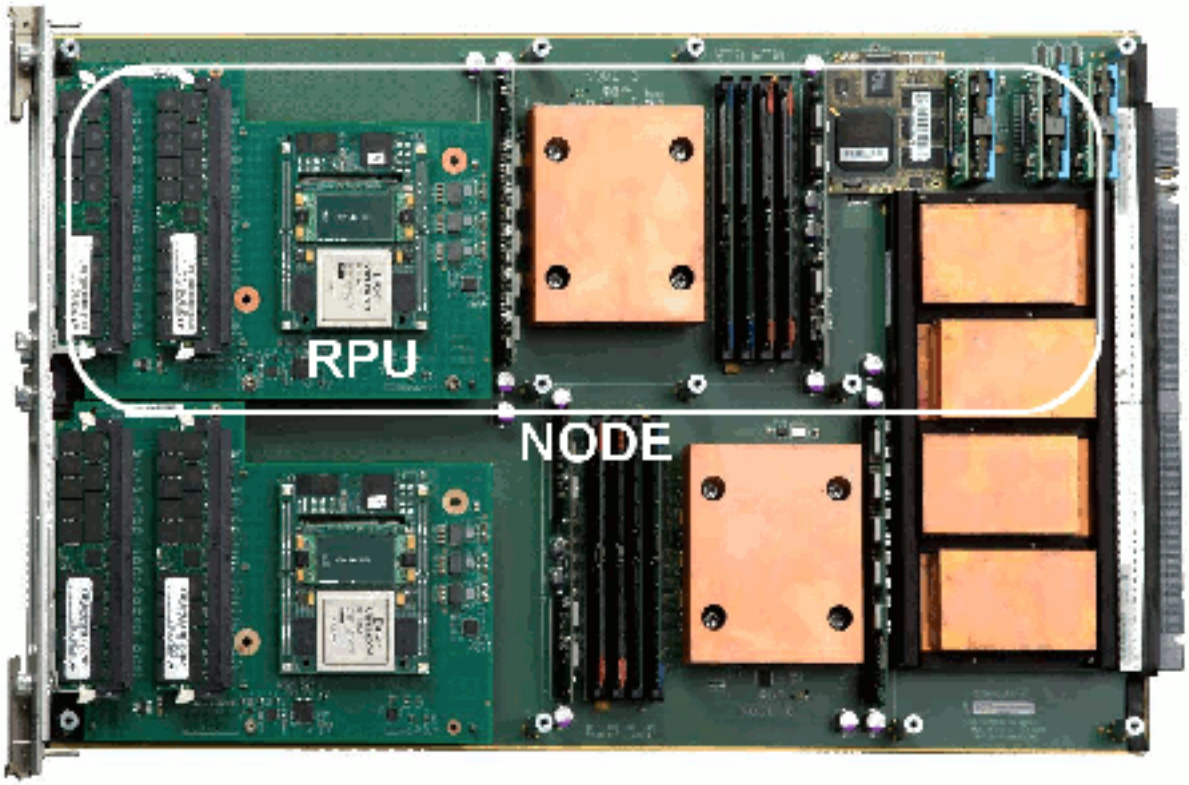
The Cray XR1 blade comes from the integration of the Cray XT service blade and a pair of DRC Computer Reconfigurable Processing Units (RPU). This innovation allows the user to write a program for multiple Opteron processors, and accelerate portions of the code by using tightly integrated RPUs.

Users of the Cray XR1 will use DRC development systems to develop applications before moving them to a Cray XR1 system. This hardware overview also includes information on the development system available from DRC, and how applications will map to the Cray XR1.

1.1 The Cray XT System with Cray XR1 Blades

The Cray XR1 blades are installed in a standard Cray XT system cabinet. Each Cray XR1 blade comprises two Cray XR1 nodes, with each node having one Opteron processor and two DRC RPU110-L200/LP modules. Figure 1 shows a picture of the Cray XR1 board. Each board comprises two nodes with four RPU. The RPU are stacked so only the top pair can be seen in the figure.

Figure 1. Cray XR1 Blade



The Cray XR1 node is connected to the SeaStar network via a 6.4 GB/s bi-directional HyperTransport link. Figure 2 shows the Cray XR1 node's architecture with the Opteron connected to the SeaStar network and two RPUs daisy chained off of the Opteron. Note that there is memory associated with the Opteron and both RPUs. Data is moved from the SeaStar network to the Opteron on the Cray XR1 node and then can be forwarded on to the attached DRC RPU via a 3.2 GB/s bi-directional HyperTransport link. Each DRC RPU is composed of a Xilinx Virtex4 LX200 with RLDRAM on the RPU and DRAM DIMMs on riser cards.

Figure 2. Cray XR1 Node Architecture

1.2 DRC Development System DS2008

The DRC Development System is a complete development platform for modifying your application subroutines to run on hardware. The system is a standard PC workstation enhanced with DRC Reconfigurable Processor Units (RPU). DRC offers three models in the DS2000 family, all with DDR memory, disk drives, and a graphics controller, but only the DS2008 should be used to develop for systems using Cray XR1 blades. The DS2008 is a 8-way system with six dual core Opterons and two DRC RPUs. It has a 4-way mother board with an additional 4-socket CPU board. The DS2008 system should be used when developing code for the Cray XR1, since the HT links are identical and the bit stream is binary compatible between the two systems.

1.3 DRC RPU110-L200 and RPU110-L200/LP

DRC offers a family of RPUs, each a complete hardware and software package that includes a Xilinx Field Programmable Gate Array (FPGA) and the functional elements that allow the module to plug into an AMD Opteron socket and interface with the HyperTransport bus, the DDR memory and other motherboard resources.

These modules come with Xilinx's Virtex-4 LX200, LX160, or LX100. The modules come in either a standard format with DRAM on the RPU (Figure 3) or a low-profile format (Figure 4) without DRAM on the RPU. The DRC RPU110 features an innovative logic array and exceptional bus and memory bandwidth that combine to provide sustainable acceleration for your applications. Implementation requires no modifications to a workstation or server motherboards. The RPU inserts directly into an open 940 socket in a standard multiprocessor AMD Opteron system and enjoys direct access to adjacent DDR memory and Opteron processors at HyperTransport speed and nanosecond latency. Tight coupling between CPU and memory means that traditional bandwidth and latency bottlenecks are virtually eliminated.

Figure 3. DRC RPU110-L200

Figure 4. DRC RPU110-L200/LP (Used in Cray XR1)

The DRC RPU110 family include several advanced features: the largest FPGAs on the market, on-board RPU memory for total bandwidth of 14.4 GB/second, up to 3 HyperTransport bus interfaces per RPU, enabling double the bandwidth while using 2 HT buses between 2 CPUs, and supporting expansion to 4-way or larger.

1.4 HyperTransport Links Used in DRC and Cray Systems

To make the bit streams between a development system and a Cray XR1 system binary compatible, both systems need to use the same HT links. The DRC DS2008 HT link configuration is the same as that on an Cray XR1 blade. Both of the architectures are shown in Figure 5 and Figure 6.

Figure 5. HT Links in DRC DS2008 Development System

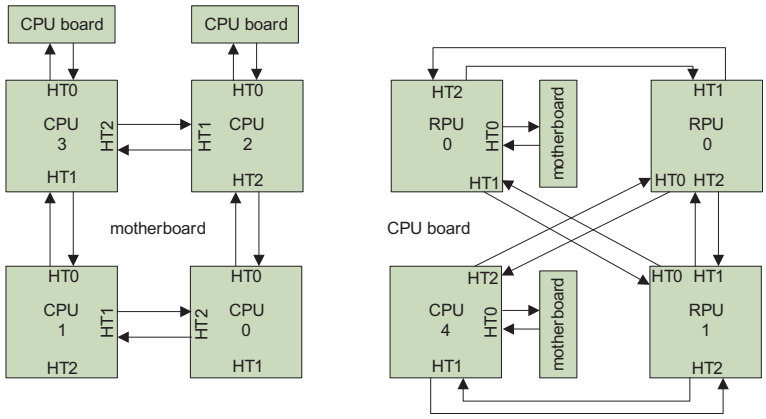
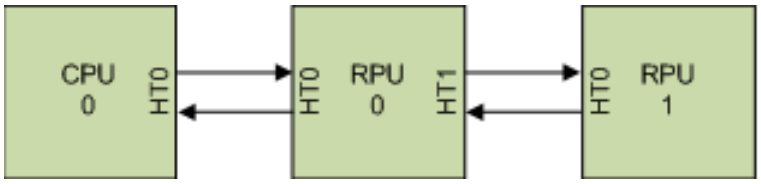


Figure 6. HT Links in a Cray XR1 Blade



1.5 Xilinx Virtex-4 Chip Resources

Figure 7 lists the resources available for all the Xilinx Virtex FPGAs.

Figure 7. Xilinx Virtex-4 Resources

Device	Configurable Logic Blocks (CLBs)				Xtreme DSP RAM	Block RAM		DCMs	PMCDs	PowerPC Processor Blocks	Total I/O Banks	Max User I/O
	Array Row x Column	Logic Cells	Slices	Max Distributed RAM (Kb)		18Kb Blocks	Max Block RAM (Kb)					
XC4VLX15	64 x 24	13,824	6,114	96	32	48	864	4	0	N/A	9	320
XC4VLX25	96 x 28	24,192	10,752	168	48	72	1296	8	4	N/A	11	448
XC4VLX40	128 x 36	41,472	18,432	288	64	96	1728	8	4	N/A	13	640
XC4VLX60	128 x 52	59,904	26,624	416	64	160	2880	8	4	N/A	13	640
XC4VLX80	160 x 56	80,640	35,840	560	80	200	3600	12	8	N/A	15	768
XC4VLX100	192 x 64	110,592	49,152	768	96	240	4320	12	8	N/A	17	960
XC4VLX160	192 x 88	152,064	67,584	1056	96	288	5184	12	8	N/A	17	960
XC4VLX200	192 x 116	200,448	89,088	1392	96	336	6048	12	8	N/A	17	960

Building and Running an Application [2]

Note: Applications must be fully debugged on a DRC development workstation before installation on an Cray XR1 system.

2.1 Loading Modules

The DRC libraries are built into the programming environment. The only additional module that may need to be loaded is the Portable Batch System (PBS). To load the PBS module, use the following script:

```
% module load pbs
```

The default compiler is PGI. If you wish to use the GNU compiler, you must switch programming environments. To switch programming environments, run the following script:

```
% module swap PrgEnv-pgi PrgEnv-gnu
```

2.2 Compiling Applications

Once the program has been debugged and tested on a DRC workstation, the source code and bit streams can be moved to the Cray XR1. The bit streams are binary compatible, but the Opteron binary must be rebuilt in order to create a static binary and the Cray XT MPI libraries. To compile a program use the following script:

% cc o a.out *your_program.c*

2.3 Running Applications

FPGA programs must be run from within the PBS batch system. The Cray XR1 software relies on PBS to grab and hold a set of nodes, such that a bit stream can be loaded and the node or nodes warm booted without losing the PBS session.

For running applications, use the following four FPGA commands:

- `load_bitstream`
- `boot_bitstream`
- `reset_bitstream`
- `inventory`

2.3.1 Loading New Bit Streams

The example below shows how to load a new bitstream into the RPU's.

```
load_bitstream <bit_stream0> <bit_stream1>
```

In the above example, `bit_stream0` loads on the first RPU (RPU0) and `bit_stream1` will load on the second RPU (RPU1). Bit streams are loaded onto the SRAM for each RPU and the "boot from SRAM" bit is set on the RPU. Then, on the next warm boot, these bit streams will be loaded into the FPGAs and become active.

2.3.2 Setting New Bit Streams

To set the RPUs to boot from the default bit stream in the RPU flash memory, run the following command:

```
reset_bitstream
```

The RPUs have a flash memory containing a default bit stream. On cold boot, the default bit stream loads into the FPGAs, providing an initial configuration. If a bad bit stream is loaded into the RPUs, this command can be used to bring the RPU back to its initial state. Again, the design is not loaded, only the "boot from flash" bit is set such that it will boot from the flash on the next warm boot.

2.3.3 Making Bit Streams Active

To make a bit stream in either the SRAM or flash active on the FPGA, the node(s) must be warm booted by running the following command:

```
% boot_bitstream
```

This command interacts with the batch system and reboots (warm boots) all the nodes within the PBS batch group. When the RPU reboots, it looks at either the "boot from SRAM" bit or the "boot from flash" bit on the RPU and boots from the appropriate memory. The command may take up to two minutes to complete. When the command finishes, appropriate bit streams will be active in the FPGAs.

2.3.4 Using the `inventory` Command

Use the `inventory` command to verify whether the RPUs on a node are active and to determine the version of the RPSysCore that was used to build the bit stream. This command will return a line for each node and will list the RPSysCore version for both RPUs.

```
% inventory
Application 35067 resources: utime 0, stime 0
c0-0c0s2n0  8 0: 0x107 ID 0x9600007244d0238d  1: 0x107 ID 0x720d01ad468d2eaf
c0-0c0s2n3 11 0: 0x107 ID 0x720e01ad468d4a04  1: 0x107 ID 0x410c009d444a881f
```

Once the bit streams are loaded, run the bit streams binary with the standard `aprun` command:

```
aprun -n 4 -N 1 a.out <a.out options>
```

Since there are two RPUs per node, and the Opteron has two cores, the user should use the `N 1` setting to guarantee that only one binary runs per node. Since the job is run under the PBS batch system, the user must specify the appropriate PBS resources for `mppwidth` and `mppnppn`.

Note: Once the RPUs are loaded with a bit stream, that design and any data left in the RLDRAM and DRAM will remain in the device. If the data should be kept private, reload the RPU with the default design to destroy any data left on the device. When a new bit stream is loaded, the memory controllers are reinitiated, making the data on the RLDRAM and DRAM from the last job unavailable. This can be done by using the following commands:

- `reset_bitstream`
- `boot_bitstream`

Example Cray XR1 Run [3]

The `user_app` program demonstrates how to compile and run a multiple Cray XR1 node program that passes data from node to node and through all the RPUs.

Entering the command **`cc -o user_app user_app.c`** in the `Test_App` directory will create a binary. This binary can be run in such a way as to execute on the Opteron processor alone, or on one or both RPUs. This application runs the tutorial example that is supplied by DRC computer, using the default bit stream that is loaded at boot time.

The basic structure of the program creates an array of random data and passes it from node to node, and from RPU to RPU within each node. The arrays transverse all the nodes and all the RPUs for the job and return back to the original sending node. The values in the array are checked at several points long the way.

Below is an example batch job to run the program on twelve nodes.

```
#!/bin/bash
#PBS -l mppwidth=12
#PBS -l mppnppn=1
#PBS -N user_app
#PBS -l walltime=1:00:00
#PBS -j oe
set -x
cd $PBS_O_WORKDIR
load_bitstream rpware_ht0_117.bit rpware_ht0_117.bit
boot_bitstream
inventory
aprun -n 12 -N 1 ./user_app -v -n 512 -l 10000 RPU0 RPU1
```

Administration Guide [4]

The Cray XR1 software is a defined as a set of RPM package managers applied to the Cray XT OS version 2.2. These RPMs are applied to both the SMW software and boot node software.

4.1 Software Management Workstation Packages

- `hss-coldstart`
- `hss-crms`
- `hss-crms-mc`
- `hss-crms-diag`

4.2 Boot Node Changes

To build the Cray XR1 boot node, run the following commands:

```
cp 2.0.41XR1-xtpackages.tar.gz ~crayadm/install.xt2.0.41XR1/craydist/xt-packages
cd /home/crayadm/install.xt2.0.41XR1/craydist/xt-packages
rm /home/crayadm/install.xt2.0.41XR1/craydist/xt-packages/*
tar xfz 2.0.41XR1-xtpackages.tar.gz
```

After running the above commands, run install as instructed in *Cray System Management Workstation (SMW) Software Installation Guide* (S–2480)

Additionally, the following settings must also be manually installed:

```
crayadm@rst17:~/install.xt2.0.41XR1/craydist/xt-packages> scp
*drc*rpm root@boot:/rr/current/software/.
Password:
xt-drc-adm-2.0.41XR1-12701.x86_64.rpm                          100%
6329KB   6.2MB/s   00:00
xt-drc-boot-cmd-2.0.41XR1-12701.x86_64.rpm                     100%
4669     4.6KB/s   00:00
xt-drc-boot-utils-2.0.41XR1-12701.x86_64.rpm                   100%
3124     3.1KB/s   00:00
crayadm@rst17:~/install.xt2.0.41XR1/craydist/xt-packages> ssh boot l root
Password:
Last login: Wed Aug 20 23:00:16 2008 from smw
2.0.41XR1 release
boot001:~ # xtopview -x /etc/node_classes
default/:/ # cd /software/
default/:/software # rpm -ivh *drc*
error: failed to stat /rr: No such file or directory
Preparing...
########################################### [100%]
   1:xt-drc-boot-utils
########################################### [ 33%]
   2:xt-drc-adm
########################################### [ 67%]
   3:xt-drc-boot-cmd
########################################### [100%]
```

Cray XR1 RPU Configuration and the Boot Process [A]

A.1 Cray XR1: Cold Boot and Warm Boot

Cold boot involves removing power from the Cray XR1 blade, Opteron socket, riser, and RPU. The cold boot operation destroys the integrity of the High Speed Network (HSN) that connects all the nodes in a Cray XT system. Cold boot should only be performed when booting the entire system and normal operation is expected after the boot. *Warm boot* involves rebooting while maintaining power to the Cray XR1 blade, Opteron socket, riser, and RPU.

Note: The procedures for warm boot and cold boot are nearly identical. Steps that pertain to cold boots will be noted accordingly.

Warm boot and cold boot procedures include the following terms:

- High Speed Network (HSN)
- Software Management Workstation (SMW): The workstation connected to the L1 and L0 microprocessors in the Cray XR1 via an Ethernet local network.
- L1: The L1 microprocessor is in every cabinet and controls all the blades in the system via the L0 on every blade.
- L0: The L0 microprocessor is on every blade in the system and controls the components on that blade.
- Boot Node: The boot node is the head node for the system and is connected to all the other nodes in the system via the HSN.
- SeaStar: The SeaStar custom router chip used to make the HSN. The SeaStar ASIC also stores code during the Opteron's initial BIOS bring up.
- Cray XR1 Node: The Cray XR1 node is made up of a SeaStar router, an Opteron, Opteron memory, a riser card with an RPU and memory, and a daughter card with RPU and memory. The RPUs are daisy-chained to the Opteron via a HyperTransport link.
- Reconfigurable Processing Unit (RPU): Each Cray XR1 node has two RPUs, each comprising several components: a user FPGA, a control CPLD, flash memory for holding default bit streams, SRAM for holding user bit streams, and RLDRAM for user access.

- RPU0: RPU0 is the first RPU in the chain and sometimes referred to as the tunnel RPU.
- RPU1: RPU1 is the second RPU in the chain and sometimes referred to as the cave RPU.
- HyperTransport (HT): HT is the point-to-point link technology used on Cray XR1 blades. This can be set to fast/wide (16-bits wide @ 400 MHz) or slow/narrow (8-bits wide @ 200 MHz.)
- no-booter: A node is defined as a no-booter when it finishes coldstart (BIOS) but fails to complete the Linux boot.
- no-chainer: A node is defined as a no-chainer when it has successfully booted but did not report the RPUs that were expected to be part of the nodes configuration. A no-chainer is identified by running the `inventory` command, which returns "unknown" as the RPU version number.

A.1.1 Initial State of the Machine

For all procedures in this document, the following preconditions apply:

- Power is supplied to the machine
- The L1 microprocessor in the cabinet has valid code in flash memory
- The L0 microprocessor on all blades has valid code in flash memory
- The RPU flash memory has a valid bit stream

When power is applied to the machine, the L1 microprocessor will power up first and read code from its flash memory. L1 will then command the L0 processors to power up. The L0s will read the code from their flash memories and go into a spin loop, waiting for actions from the SMW.

A.1.2 RPU Reconfiguration

To perform RPU reconfiguration, the user must have a file with a new bit stream in the Lustre shared file system.

When the `load_bitstream` command is issued, the nodes that are part of the PBS group are compiled into a list. Then the `aprun` command is issued, which invokes a pair of DRC `rpu_reconfig` commands for each node in the PBS group. The `rpu_reconfig` reads the new bit stream from the file and writes it to the SRAM on both RPUs attached to that node. The `rpu_reconfig` commands follow the following format:

```
rpu_reconfig --device /dev/drcmod1 bit-stream

rpu_reconfig --device /dev/drcmod0 bit-stream
```

This command changes the RPU configuration of the complex programmable logic device (CPLD) so that, on the next warm boot, the CPLD will use the bit stream image from the SRAM, instead of rebooting from the default bit stream in the flash.

Note: Both RPUs continue to run the old, previously used bit streams. At the end of the `rpu_reconfig` command, the new bit streams are loaded into the SRAM, but are inactive until the next warmboot.

To check if the RPU is ready for reconfiguration, run the following command:

```
rpu_reconfig -i
```

The output from this command will show the "boot from SRAM" bit as set to true with a new bit stream loaded

A.1.3 Warm Boot – Shutdown

To initiate a warm boot with the bit streams loaded into the SRAM, issue the `boot_bitstream` command. This command generates a list of nodes in the PBS group that are to be rebooted. Then, the `rr.py` script runs on that list of nodes. The `rr.py` process communicates with the server process on the SMW, which controls the reboot of the given nodes.

On the SMW, `rr.py` uses the `xtgenevent` command to issue a shutdown request to all the nodes on the generated list. Each node receives this request, and proceeds through a Linux shutdown process modified for Cray XT compute nodes. This process includes a DRC specific script. This script controls any reconfiguration indicated by the RPU's configuration. If indicated, the script will change PCI configuration space variables to change the HT link configuration at the next reset. The SeaStar HT link will remain unchanged.

The server then waits for all the nodes in the given list to complete the shutdown by monitoring the HSS events coming from all the nodes.

A.1.4 Warm Boot – Bring Up

After all the listed nodes have completed shutdown, the `rr.py` server then starts to bring up the nodes by issuing the `xtbounce -s` command to all on the nodes in the requested set. The `xtbounce` command is a standard part of the Cray XT system administration and is covered in *Cray System Management Workstation (SMW) Software Installation Guide* (S–2480). The process is similar to the BIOS boot on a standard microprocessor system.

Note: In the case of a cold boot, the `xtbounce` command power cycles the blade and all the components on that blade, including the Opteron processor, riser, daughter, and RPUs. If this is a warm boot, there is no power cycle of any of the components on the blade.

The bounce process starts with the L0 microprocessor writing code in the form of applets to the SeaStar's RAM. It then tags the Opteron processor to look at the SeaStar memory and execute those instructions. Once the processor finishes an applet, it returns control to the L0. The L0 then loads another applet into the SeaStar's memory and tells the Opteron to continue executing from the SeaStar memory.

There are many applets, and the L0 serves them up one at a time. The console will show which applet is currently being executed and when it has finished.

If the bus configuration was changed during shutdown, the HyperTransport (HT) link will be reconfigured to slow/narrow during the first reset issued by the L0 microprocessor. HT init is the second applet that is loaded during the bounce process. This applet determines how devices are linked, assigns the appropriate device IDs, and determines which HT will supply the BIOS (applets). As the link is established between the Opteron processor and the first RPU, the applet determines the current HT configuration. If the HT is set as slow/narrow, HT init reconfigures the link to fast/wide. At the conclusion of the HT init process, the applet informs the L0 that it has changed the configuration of the HT link and L0 initiates another warm boot.

During the follow-up warm boot process, the HT init will see the fast/wide configuration between the Opteron processor and the first RPU and the warm boot will continue. The HT init applet also checks to see if there are one or two RPUs attached to the HT link. Sometimes the second RPU is late coming up and the Opteron does not detect it. This scenario is called a "no-chainer" and the HT init applet will instruct L0 to perform another warm boot upon exit.

A.1.5 Warm Boot – RPU Details

During HT init, the Opteron processor tells the L0 microprocessor to toggle the lines that reset the RPU's. When the CPLD sees the reset become active, it reconfigures the FPGA that is using the data previously written into SRAM. The HT interface code in the FPGA then waits for the Opteron to begin link initialization.

Note: One known issue with the warm boot process is related to the HT init applet. On some occasions, the Opteron processor initiates HT init, but never completes the applet. The current theory to explain this phenomenon is that noise in the HT lines causes the Opteron to receive confused signals and hang up.

The HT powerOK line is only connected to the CPLD. The reset# is connected to both the CPLD and the FPGA. During cold boot, the system de-asserts powerOK and asserts the reset# lines. During warm boot, powerOK is in the assert state but only reset# is asserted.

A.1.6 Warm Boot – Linux Startup

On the SMW, the `rr.py` script used the `xtbounce` command on all the nodes in the requested group to reset those nodes. The message file on the L0 will show the different applets being sent to each node and a message as they finish. The `rr.py` server parses the output from `xtbounce`, drops any nodes that fail the reset process, and takes copies of certain node specific data from each failing node.

At this point, all the nodes will have finished loading their BIOS and will standby.

Once the `xtbounce` process completes, the surviving nodes are booted using the `xtcli` command's `boot` subcommand, which arranges for the Linux kernel and initramfs to be transferred to the booting node and instructs the kernel to start booting. The Linux kernel does not distinguish from a cold boot or a warm boot.

During the normal CNL boot, the DRC device driver is loaded. If the driver detects RPUs, then the load succeeds and device nodes are created in `/dev` to allow user interaction with the RPUs. It also initiates Application Level Placement Scheduler (ALPS) and Lustre file system.

Once the above actions have taken place, Linux is up and ready for binary to use the new bit stream.

Glossary

blade

1) A field-replaceable physical entity. A Cray XT service blade consists of AMD Opteron sockets, memory, Cray SeaStar chips, PCI-X or PCIe cards, and a blade control processor. A Cray XT compute blade consists of AMD Opteron sockets, memory, Cray SeaStar chips, and a blade control processor. A Cray X2 compute blade consists of eight Cray X2 chips (CPU and network access links), two voltage regulator modules (VRM) per CPU, 32 memory daughter cards, a blade controller for supervision, and a back panel connector. 2) From a system management perspective, a logical grouping of nodes and blade control processor that monitors the nodes on that blade.

CNL

CNL is the Cray XT and Cray X2 compute node kernel. CNL provides a set of supported system calls. CNL provides many of the operating system functions available through the service nodes, although some functionality has been removed to improve performance and reduce memory usage by the system.

node

For CLE systems, the logical group of processor(s), memory, and network components acting as a network end point on the system interconnection network.