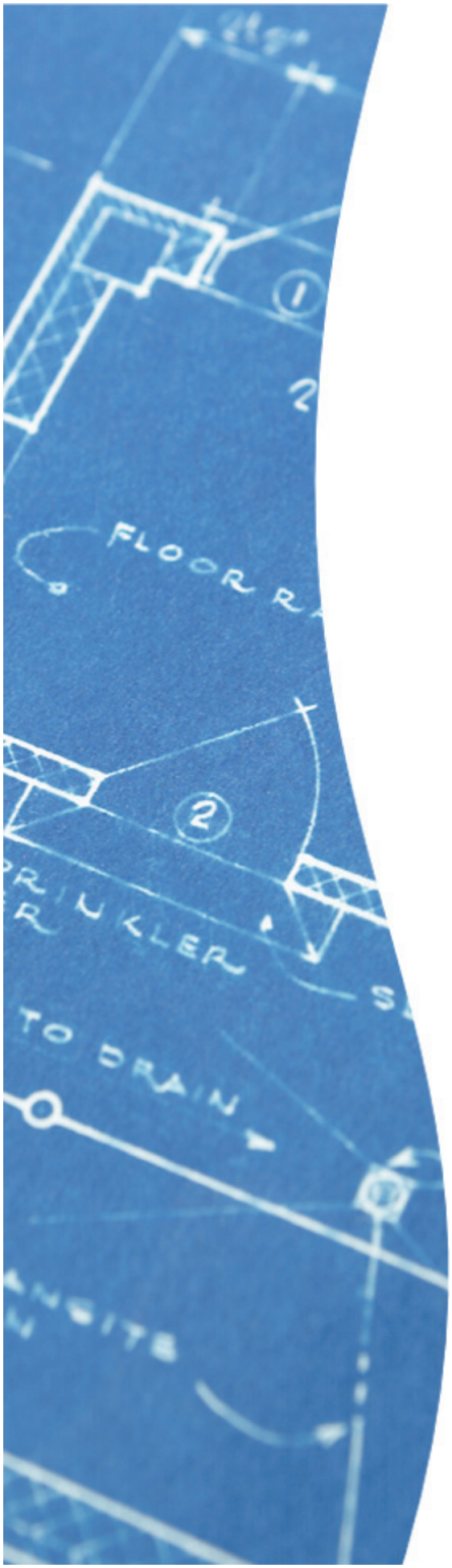

LDOMS I/O BEST PRACTICES
DATA RELIABILITY WITH LOGICAL DOMAINS

Peter A. Wilson, Systems Technical Marketing

Sun BluePrints™ On-line

Part No 820-5089-10
Revision 1.0, 6/20/08

Table of Contents

Data Reliability With Logical Domains 1
About This Article 2

Data Availability And Reliability Overview 4
Internal Storage Reliability With Logical Domains 4
Hardware RAID 5
Reliability In I/O Domains 5
Reliability In Guest Domains 7

I/O Reliability And Availability On Sun Platforms 8
Example: Sun SPARC Enterprise T2000 Server I/O Architecture 8
I/O Architecture Summary 9

Installing And Configuring Logical Domains 11

Configuring Hardware RAID 12
About Hardware RAID Disk Sets 12
Implementing Hardware RAID 13
Creating A Logical Domain 15

Configuring ZFS In The I/O Domain 18
Implementing ZFS And Cloning Guest Domain Volumes 18
Create A ZFS Pool 19
Clone And Unconfigure 20
Clone And Run The Unconfigured System 22
Save Your Configuration 23

Configuring Volume Management In Guest Domains 24
Setting Up Volume Management Through Network Install 24
Set Up The Guest Domain 24
Rules File Setup 25
Profile File Setup 26
Sysidcfg File Setup 26
Add Install Client 27
Install 27
Housekeeping 27
Test 28

Conclusion 30
About The Author 31
Acknowledgments 31
References 31
Ordering Sun Documents 31

Chapter 1

Data Reliability With Logical Domains

Datacenter best practices for disk and network I/O are relatively well defined. Configure multiple data paths to network and storage resources to increase availability. For data reliability, use media redundancy including hardware and software RAID and sophisticated filesystems such as the Zettabyte File System (ZFS). These rules of thumb drive the way in which real-world datacenter servers are configured and cabled — but how do these rules translate into the virtual world? What does it mean to have redundant virtual disks and networks? How should they be leveraged to provide availability and reliability benefits similar to those in the physical world? This Sun BluePrints™ Series article begins to address the answers to these questions in the context of Logical Domains.

Logical Domains is a virtualization technology supported on Sun servers with CoolThreads™ technology — those powered by UltraSPARC® T1, T2, and T2 Plus processors. Logical Domains, or LDoms, allow server resources to be partitioned and allocated to virtual machines. Resources can be allocated down to the CPU thread, cryptographic processor, memory, and PCI bus. The Logical Domains architecture consists of four classes of virtual machines that run on a thin hypervisor layer (Figure 1): a single *control domain* manages the virtualized environment; one or more *I/O domains* own real I/O devices that *service domains* use to provide virtual I/O services to *guest domains*. Each domain runs in its own dedicated partition of the server hardware. A single domain can take on multiple roles: for example I/O and service domains are usually combined into a single domain that handles physical and virtual I/O. In many configurations, the control domain is combined with an I/O and service domain. In this paper, we refer to a combined I/O and service domain as simply an I/O domain.

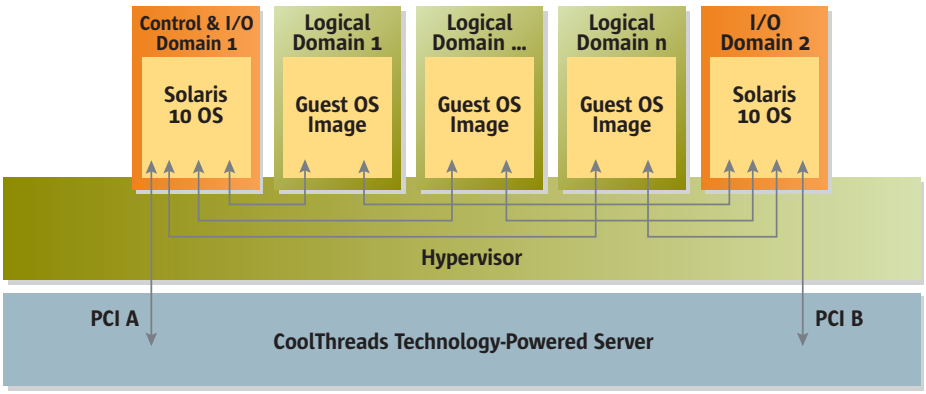

Figure 1. Logical Domains supports multiple guest domains each with their own secure partition of server hardware. I/O is handled by I/O domains that 'own' one or more PCI buses.

Logical Domains allow individual PCI root nexus nodes (referred to as PCI buses in this article) to be allocated to I/O domains so that each one 'owns' a PCI bus and any devices connected to it. On servers with multiple PCI buses, multiple I/O domains can be configured to provide multiple paths from guest domains to I/O resources. If an I/O domain, its PCI bus, or its peripherals fail, or the domain needs to be rebooted, a guest domain can continue to access its I/O resources through the second I/O domain given an appropriate configuration in the guest domain.

In the realm of *data reliability*, Sun servers with CoolThreads technology, combined with the Solaris™ 10 Operating System, offer several tools to build reliability through redundancy. These tools can be used at three different architectural levels in order to increase data reliability using the server's on-board disk drives and disk controller:

- *Hardware-level redundancy* can be established using the RAID capabilities of the server's built-in disk controller.
- An I/O *domain* can support I/O redundancy through software features including the Zettabyte File System (ZFS) and Solaris Volume Manager software. With reliability established in the I/O domain, reliable storage can be provided to guest domains.
- *Guest domains* can use the same software features to implement redundancy in the guest itself: ZFS and Solaris Volume Manager software can be configured within a logical domain using virtual disks.

This Sun BluePrints article discusses the approaches and trade-offs for establishing data reliability through redundancy using the resources of the server itself, namely its own PCI bus(es), built-in disk controller, and disk drives. Logical Domains opens up a realm of possibilities for implementing I/O availability and reliability techniques in the virtual world, and this article is the first in a series to address this broad topic. Perhaps not surprisingly, the techniques discussed in this article all have their benefits and drawbacks. The choice of which ones to implement is one that has to be made after carefully comparing each solution's benefits to your datacenter requirements.

About This Article

This Sun BluePrints Series article discusses the following topics:

- Chapter 2, "Data Availability And Reliability Overview" on page 4 provides an overview of both availability and reliability techniques that can be used with Logical Domains.
- Chapter 3, "I/O Reliability And Availability On Sun Platforms" on page 8 discusses specifically what techniques can be used on different Sun servers with CoolThreads technology depending on their logical architecture.
- Chapter 4, "Installing And Configuring Logical Domains" on page 11 gives a brief overview of how to set up a working Logical Domains environment.
- Chapter 5, "Configuring Hardware RAID" on page 12 provides step-by-step instructions for configuring reliable disk storage using the server's disk controller.

- Chapter 6, “Configuring ZFS In The I/O Domain” on page 18 shows the benefits of establishing reliability using ZFS, in particular using snapshots and clones with Logical Domains.
- Chapter 7, “Configuring Volume Management In Guest Domains” on page 24 shows how to set up Solaris Volume Manager software in a guest domain using JumpStart™ software.

Chapter 2

Data Availability And Reliability Overview

Although the goal of this article is to address data reliability using on-board disk storage with Logical Domains, reliability and availability are closely related. This relationship is important to understand because different approaches to reliability and availability may, or may not be, applicable depending on the underlying server architecture. For the purpose of this article:

- *Reliability* is what makes sure that no data is lost. We do this by storing redundant copies of data stored on disk using hardware RAID, software RAID implemented in volume managers (such as Solaris Volume Manager software), or filesystems (such as ZFS). There are three levels at which we can establish this redundancy using LDoms and internal disk storage: in the disk controller itself, in the I/O domain, and in the guest domain.
- *Availability* is what makes the data accessible as much as possible. We usually increase availability by establishing at least two of every path to data so the failure of one does not affect data accessibility. With Logical Domains:
  - We can configure multiple network connections to network resources and multiple PCI buses to disk storage.
  - We can establish dual virtual data paths from guest domains to I/O resources by setting up two I/O domains. An I/O domain owns all of the devices on each PCI bus dedicated to it, so the use of multiple I/O domains is limited to servers having multiple PCI buses.
  - Having multiple I/O domains handling I/O for guest domains increases availability by allowing administrators to reboot an I/O domain — for example after its operating system has been patched — without causing the guest domains to lose connectivity to their data.
  - Using multiple I/O domains also isolates the resources used by one I/O domain from another — so that a badly-behaved guest domain using all of an I/O domain's resources could not prevent other guest domains from accessing their data.

The next section discusses the various levels at which you can establish reliable disk storage with Logical Domains; the section following that discusses the architecture of Sun's CoolThreads technology-enabled servers and how they support various availability approaches.

Internal Storage Reliability With Logical Domains

There are three approaches to establishing reliable disk storage using internal disk drives on Sun servers with CoolThreads technology: hardware RAID, software RAID in the I/O domain, and software RAID in the guest domain. These approaches are

discussed here, and configuration examples are provided later in the article. In general, the lower in the stack you go, the more efficient the RAID implementation should be. Hardware RAID, implemented in the disk controller, offloads the CPU from implementing RAID, and should be the most efficient. Software RAID implemented in a guest domain must replicate every write to at least two virtual disks, imposing the overhead of traversing through an I/O domain for every write.

All Sun servers with CoolThreads technology have a single controller for internal disk drives, so only a single I/O domain can be used to access internal disks. Given this fact, for access to *internal* storage, reliability can be considered independently from availability.

Hardware RAID

Every Sun server with CoolThreads technology comes with an internal RAID controller that supports RAID 0 (striping) and RAID 1 (mirroring). RAID 1 allows pairs of disks to be configured into RAID groups where the disk mirroring is handled in hardware. This approach is easy to manage, and is transparent to both the I/O and the guest domains.

With hardware RAID mirroring disks, there are three ways in which this reliable storage can be provided to guest domains:

1. An entire mirrored volume can be provided as a virtual disk to a guest domain. This disk can be partitioned and used as a boot disk.
2. A partition of a mirrored disk can be provided to a guest domain. Partitions provided to guest domains cannot be partitioned as virtual disks can.
3. A flat file created on a mirrored disk can be provided as a virtual disk to a guest domain. This disk can be partitioned and used as a boot disk.

This is true through Logical Domains 1.0.3, but may change in the future

The most common technique is Option 3, using a flat file provided to a guest domain as a virtual disk because many such virtual disks can be created on a single mirrored disk, and these virtual disks can be partitioned and booted from the guest domain. Option 1, allocating a mirrored disk set per guest domain, is an option that would significantly limit the number of available domains on most systems. Option 2, allocating a partition of a mirrored disk, limits the number of virtual disks that can be created because only a fixed number of partitions can be created on a physical disk. These virtual disks also cannot be used as boot disks in Logical Domains.

Reliability In I/O Domains

Disk storage reliability can be established in I/O domains with software RAID, where reliable virtual disks are provided to guest domains. As with hardware RAID, this approach provides reliable disk storage to guest domains with the details hidden from them. The reliable storage is managed from a single point: the I/O domain itself. There are two approaches that use software included with the Solaris 10 OS:

- *Solaris Volume Manager* software can be used to create RAID sets that can be provided as virtual disks to guest domains in two ways: entire volumes can be exported as virtual disks, and flat files can be created on a volume and then exported as a virtual disk. Again, the most common approach would be to create flat files that provide partitionable virtual disks to guest domains.
- *The Zettabyte File System (ZFS)* incorporates volume management and filesystem technology into a single interface that can manage both. Disks can be allocated to a ZFS pool with the ZFS volume management capabilities establishing replication. ZFS supports mirroring and RAID-Z, which is similar to RAID 5. The most common approach using ZFS is to allocate flat files to guest domains so that they can be partitioned and used as boot disks. In LDoms version 1.0.3 you can provide a ZFS filesystem to a guest domain.

If the choice is between Solaris Volume Manager software and hardware RAID, hardware RAID is the most transparent of the choices. Still, if an IT organization has standardized on Solaris Volume Manager software, the organization's current best practices can be directly translated into a Logical Domains environment.

If the choice is between either RAID technology and ZFS, there are some important ZFS features that help in implementing Logical Domains. Using ZFS snapshots and cloning, a flat file containing a guest domain's virtual disk can be frozen in time and used as a backup. One or more snapshots can be taken and used as points in time to which a Logical Domain can be rolled back. Snapshots can also be used as point-in-time volume copies and transferred to magnetic tape. Unlike volume copies, snapshots are instant and space efficient. The only space used by a snapshot is the volume's metadata and any blocks changed in the primary volume since the snapshot was taken.

ZFS allows read-only snapshots to be turned into clones. A clone is a read/write volume created from a snapshot that is equally instant and space efficient. Clones can be used to create virtual disks for a number of Logical Domains and for a number of purposes. For example:

- A golden master environment can be created and then cloned for use by multiple Logical Domains. Horizontally scaled applications (such as Web servers) are prime candidates for using this technology.
- A single environment can be cloned and used for various project phases. Clones of virtual disks can be used for development, test, and production environments.
- An existing production environment can be cloned and used for testing operating system and application patches before they are put into production.

In all cases, the benefits of ZFS clones are the same: they can be created nearly instantly, and they use storage efficiently, helping to raise return on investment.

Reliability In Guest Domains

Just as reliable disk storage can be configured in an I/O domain, Solaris Volume Manager software and ZFS can be used in guest domains as well. Using internal storage, multiple physical disks, physical disk partitions, or flat files can be provided to guest domains through a single I/O domain. Solaris Volume Manager software or ZFS can be configured in the guest domain to use the virtual volumes. Where a single I/O domain is used, this approach has no reliability or availability advantage over configuring the same software in the I/O domain. This approach is most useful if an IT organization's current best practices dictate using OS-level replication. The ability to use volume management from within a Logical Domain allows these standard and best practices to be extended into the virtual world.

Configuring reliability and availability with external storage will be considered in a future article.

Establishing reliability in guest domains is a highly viable solution when multiple interfaces to *external* storage are configured. This allows guest domains to access storage through two I/O domains. This increases availability by using two redundant paths to storage so that the failure or rebooting of an I/O domain (or failure of any component in the path to storage) does not interrupt access.

Chapter 3

I/O Reliability And Availability On Sun Platforms

The architecture of Sun servers with CoolThreads technology dictates whether multiple I/O domains can be used to extend parallel redundant channels to I/O. The configuration steps provided in this article focus on internal disk storage which, for all Sun servers with CoolThreads technology, are connected via a single PCI bus and disk controller. Internal storage is thus available via one I/O domain. As background information for future articles, we outline the relationship between Sun servers with CoolThreads technology and the ability to use multiple I/O domains for network and disk I/O.

Example: Sun SPARC Enterprise T2000 Server I/O Architecture

Figure 2 illustrates how I/O domains can access devices on the two PCI buses in a Sun SPARC Enterprise T2000 server. If two I/O domains are created, one domain can have access to PCI bus A, along with all of its peripherals, and one domain can have access to PCI bus B, along with all of its peripherals.

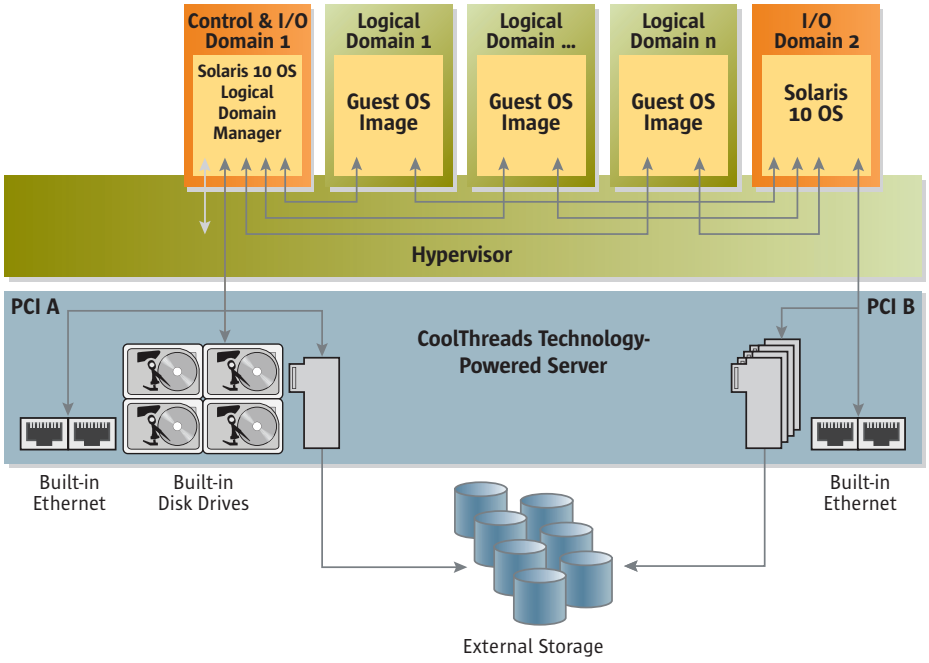

Figure 2. The Sun SPARC Enterprise T2000 server's I/O configuration illustrates how internal and external I/O resources can interface to guest domains through two I/O domains.

The illustration highlights the fact that all internal drives are connected to a single PCI bus, so all internal storage is accessed via one I/O domain. If the server's two PCI buses are allocated to each of two I/O domains, however, two host bus adapters can be configured to provide access to external storage through two PCI Express slots via two I/O domains. Redundancy and resiliency can then be established in the guest domain with software mirroring such as that provided by Solaris Volume Manager software. The Sun SPARC Enterprise T2000 server is used as an interesting example here because it is the only server in this series with its built-in Ethernet ports split between two PCI buses. This architecture allows IP Multipathing (IPMP) to be used for high availability network access through two I/O domains using internal ports.

I/O Architecture Summary

Table 1 details the I/O architecture for all Sun servers with CoolThreads technology as of the date of this article. It allows us to assess the server-imposed constraints on using internal and external redundant paths to storage and network resources:

Table 1. Sun server configurations affecting I/O domain configurations with Logical Domains.

			Number of interfaces connected to each PCI bus		
Sun Server	PCI Buses	PCI Bus	Disk Controllers	Built-In Ethernet Ports	Expansion Slots
Sun SPARC Enterprise T5240 Server	2	A	1	0	3
		B	0	4	3
Sun SPARC Enterprise T5140 Server	2	A	1	0	1
		B	0	4	2
Sun SPARC Enterprise T5220 Server	1	A	1	4	6
Sun SPARC Enterprise T5120 Server	1	A	1	4	3
Sun Fire / Sun SPARC Enterprise T2000 Server	2	A	1	2	1
		B	0	2	4
Sun Fire / Sun SPARC Enterprise T1000 Server	2	A	0	0	1
		B	1	4	0

- The number of PCI buses indicates the maximum number of I/O domains that can be configured. By default, the first I/O domain that you create owns all PCI buses; creating a second I/O domain requires some additional steps.
- Servers with a single PCI bus are limited to using a single I/O domain for all I/O.
- Servers with two PCI buses can have two I/O domains. For example, if a second I/O domain is created on a Sun SPARC Enterprise 5240 server, and PCI bus B is allocated to it, the domain can access four built-in Ethernet ports and three PCI Express expansion slots.
- All of the servers having two PCI buses connect to at least one expansion slot per bus. This allows PCI Express host bus adapters and/or network interface controllers to be

installed for redundant access to storage and/or networks through two I/O domains. This includes the Sun SPARC Enterprise T5240, T5140, and T2000 servers.

- The Sun SPARC Enterprise T2000 server is the only product having built-in Ethernet ports configured across both PCI buses. This server can support two I/O domains, each connected to two Ethernet ports, allowing IP Multipathing (IPMP) to be configured in guest domains with no additional hardware required.

Chapter 4

Installing And Configuring Logical Domains

An introduction to Logical Domains 1.0 is available in the Sun BluePrints Series article: *Beginner's Guide to LDoms: Understanding and Deploying Logical Domains*, 820-0832, available at *www.sun.com/blueprints*. Since this document describes LDoms 1.0, please also refer to the *Logical Domains Administration Guide*, available from *docs.sun.com*.

A brief but essential guide to setting up a network install server is provided in the Sun BluePrints Series article: *Building a JumpStart Infrastructure*, 816-0428, available at *www.sun.com/blueprints*. See also the *Solaris OS Installation Guide: Network-Based Installations* (820-0177) and *Custom JumpStart and Advanced Installations* (820–0179) from *docs.sun.com*.

In order to set up Logical Domains on Sun servers with CoolThreads technology, you'll need to install the required set of Logical Domains software, upgrade your server firmware (if necessary), and reboot your server with Logical Domains enabled. These procedures are well-described in external documents, and the recommended steps are outlined here:

1. It's a good idea to set up a network install server. With a network install server, you'll be able to quickly and easily install Solaris software into guest domains using JumpStart™ software, or install your own OS and application combinations using Solaris Flash software. For Logical Domains version 1.0.2 and earlier, a network install server is a requirement, as these versions do not provide access from guest domains to built-in DVD drives. For Logical Domains 1.0.3 and later, you can boot a guest domain from a DVD and install the Solaris OS interactively.
2. If you plan to use hardware RAID, update the firmware in your disk controller if necessary, and set up hardware RAID as described in the next chapter before you proceed to the next step. To obtain the firmware, consult the *Product Notes* document for your server available at: *http://docs.sun.com/app/docs /prod/coolthreads.srvr#hic*. You'll find a patch that contains instructions for determining your current firmware version and the latest firmware to install. You'll need to update firmware while booted from the network or from the install CD since this process re-partitions the disks that you configure into RAID sets.
3. Install the Solaris 10 8/07 release or later. This OS version is required on some of the newer CoolThreads technology-enabled servers, and it incorporates patches that are needed with earlier versions. You can check the *Product Notes* issued with each version of the Solaris OS and LDoms software for the required minimum supported versions of each component.
4. Visit *http://www.sun.com/servers/coolthreads/ldoms/get.jsp* to obtain the latest Logical Domain Manager software, links to the latest Administration Guide, and the latest Release Notes. LDoms version 1.0.2 works with the Solaris 10 8/07 OS.
5. The *Logical Domains Administration Guide* describes how to update the system firmware. This is important because the LDoms hypervisor runs in firmware and must be synchronized with the version configured in the OS. Note that the firmware revision number is not related to the Logical Domains version number.
6. Install the Logical Domains Manager following the instructions contained in the package that you downloaded.
7. Set up your control and I/O domain as described in the Sun BluePrints article cited in the sidebar, or as described in the *Logical Domains Administration Guide*.

Chapter 5

Configuring Hardware RAID

More details on configuring hardware RAID can be found in the *Server Administration Guide* for your server, available at *docs.sun.com*.

A typical Logical Domains configuration uses one domain to act as the server's control and first service and I/O domain. Hardware RAID, available on all Sun servers with CoolThreads technology, is an excellent way to provide a highly reliable boot disk for the combined control, service, and I/O domain. This domain's reliability is of critical importance: if it fails, all I/O to on-board disk drives for all guest domains fails as well.

This chapter describes how to configure disk mirroring for the first two disk drives on a four-disk server. These configuration steps were performed on a Sun SPARC Enterprise T2000 server, however the commands for other platforms are the same.

About Hardware RAID Disk Sets

When you set up mirroring using the hardware RAID controller, it takes two identical disk devices and presents a new, single disk drive to the OpenBoot™ software and to the operating system. You can use the `format` command to show the available disk drives on your server. When it initializes, it presents a list of the available disk drives. For example, this server has two 72 GB and two 146 GB drives installed:

```
AVAILABLE DISK SELECTIONS:
       0. c1t0d0 <SUN72G cyl 14087 alt 2 hd 24 sec 424>
          /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@0,0
       1. c1t1d0 <SUN72G cyl 14087 alt 2 hd 24 sec 424>
          /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@1,0
       2. c1t2d0 <SUN146G cyl 14087 alt 2 hd 24 sec 848>
          /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@2,0
       3. c1t3d0 <SUN146G cyl 14087 alt 2 hd 24 sec 848>
          /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@3,0
```

In this example, we will combine disks 0 and 1, both Sun 72 GB drives, into a single mirrored RAID set. After the sequence of steps is complete, the `format` command will display the following choice of drives, making it appear that the server has one fewer disk drive; namely, the `c1t1d0` drive no longer appears:

```
AVAILABLE DISK SELECTIONS:
       0. c1t0d0 <LSILOGIC-LogicalVolume-3000 cyl 65533 alt 2 hd 16 sec 136>
          /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@0,0
       1. c1t2d0 <SUN146G cyl 14087 alt 2 hd 24 sec 848>
          /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@2,0
       2. c1t3d0 <SUN146G cyl 14087 alt 2 hd 24 sec 848>
          /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@3,0
```

In contrast to Solaris Volume Manager software, hardware RAID mirrors only whole disks at a time, so regardless of how the disk is partitioned, all of its data is mirrored. The drive appearing at `c1t0d0` should be partitioned and the OS installed on it only after the RAID set is configured by the following steps. Any data previously stored on the mirrored disks will be lost.

Implementing Hardware RAID

The `raidctl` command provides a controller-independent way to set up hardware RAID. The act of creating the RAID set invalidates any data stored on the drives that you configure into the set, so it's best to execute the command from an install environment booted from DVD or from a network install server. If you have set up a network install server, simply boot into single-user mode from the network:

```
{0} ok boot net -s
```

The `raidctl` command allows disks to be specified using the standard *controller-target-disk* format (such as `c1t0d0`), or in an *x.y.z* format from which you can learn the possible *x*, *y*, and *z* values by typing the `raidctl` command with no arguments. Since we already know the controller-target-disk way of specifying disks, we use this form to create a RAID 1 set that mirrors disks *c1t0d0* and *c1t1d0*:

```
# raidctl -c -r 1 c1t0d0 c1t1d0
```

After asking you to confirm this destructive operation, the `raidctl` command will provide status reports as it configures the RAID set, concluding by telling you the identify of the new volume that it has created.

```
Creating RAID volume will destroy all data on spare space of member
disks, proceed (yes/no)? y
/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2 (mpt0):
        Physical disk 0 created.
/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2 (mpt0):
        Physical disk 1 created.
/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2 (mpt0):
        Volume 0 created.
/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2 (mpt0):
        Volume 0 is |enabled||optimal|
/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2 (mpt0):
        Volume 0 is |enabled||optimal|
/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2 (mpt0):
        Physical disk (target 1) is |out of sync||online|
/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2 (mpt0):
        Volume 0 is |enabled||degraded|
/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2 (mpt0):
        Volume 0 is |enabled||resyncing||degraded|
Volume c1t0d0 is created successfully!
```

When you format the disk, it's important to let the `format` command auto-configure the disk type before you partition and label the drive. Type the `format` command and select `disk 0`, assuming that you have combined drives `0` and `1` into a single volume. Use the `type` command to display the selection of possible disk types:

```
format> type


AVAILABLE DRIVE TYPES:
        0. Auto configure
        1. Quantum ProDrive 80S
        2. Quantum ProDrive 105S
        3. CDC Wren IV 94171-344
        4. SUN0104
        ...
```

Select `0` to auto-configure the drive, and you will see that the `format` command recognizes the newly-created RAID set. Then label the disk:

```
Specify disk type (enter its number)[19]: 0
c1t0d0: configured with capacity of 68.00GB
<LSILOGIC-LogicalVolume-3000 cyl 65533 alt 2 hd 16 sec 136>
selecting c1t0d0
[disk formatted]
format>
format> label
Ready to label disk, continue? y
```

Note that the disk size has reduced slightly to allow the RAID controller room on the disk to store its metadata. Now you can partition the disk as part of the OS install process, and at any time you can type `raidctl -l` to see the status of the RAID set. The output from this command, issued just after the RAID set was created, indicates that the controller is synchronizing the two drives. When the synchronization is complete, you'll see the status change to `OPTIMAL`.

```
# raidctl -l c1t0d0
Volume                  Size    Stripe  Status   Cache  RAID
        Sub                     Size                    Level
                Disk
----------------------------------------------------------------
c1t0d0                  68.3G   N/A     SYNC     N/A    RAID1
                0.0.0   68.3G           GOOD
                0.1.0   68.3G           GOOD
```

As this sequence of instructions has demonstrated, setting up hardware RAID is straightforward. Now any raw partition or flat file that you provide to a logical domain has redundant storage backing it up.

Creating A Logical Domain

For instructions on creating Logical Domains, please refer to the resources noted in the sidebar at the top of page 11.

After you've installed the Solaris 10 OS on the RAID set, you need to install and configure the Logical Domains software. You create a virtual disk service and virtual switch that provide storage and networking to guest domains. You can create flat files whose contents are mirrored, connect them as boot disks to guest domains, and then use your network install server to install the Solaris OS in each domain. This example illustrates creating a guest domain `guest2` with 2 virtual CPUs, 2 GB of main memory, and an 8 GB boot disk. The guest domain's network is connected to the existing virtual switch named `primary-vsw0`. The domain is set not to automatically boot.

Because the server used to create these examples is powered with CoolThreads technology, its chosen host name is `fraser` in honor of one of Colorado's coldest towns.

```
fraser# mkfile 8g /domains/s2image.img
fraser# ldm create guest2
fraser# ldm set-vcpu 2 guest2
fraser# ldm set-mem 2g guest2
fraser# ldm add-vnet vnet0 primary-vsw0 guest2
fraser# ldm add-vdsdev /domains/s2image.img vol2@primary-vds0
fraser# ldm add-vdisk vdisk0 vol2@primary-vds0 guest2
fraser# ldm bind guest2
fraser# ldm set-variable auto-boot\?=false guest2
```

Assuming that you plan to do a network install into this domain, one of the key pieces of information that you'll need to set up JumpStart software is the domain's network MAC address on the virtual switch connecting the server to the physical network:

```
fraser# ldm list-bindings guest2
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL
UPTIME
guest2           bound    -----   5001    2     2G

MAC
    00:14:4f:fb:08:50

VCPU
    VID    PID    UTIL STRAND
    0      8             100%
    1      9             100%

MEMORY
    RA               PA               SIZE
    0x8000000        0x188000000      2G

VARIABLES
    auto-boot?=false

NETWORK
    NAME             SERVICE                     DEVICE     MAC
    vnet0            primary-vsw0@primary        network@0  00:14:4f:fa:2b:e4
        PEER                         MAC
        primary-vsw0@primary         00:14:4f:fb:ae:95
        vnet0@guest1                 00:14:4f:fb:58:90

DISK
    NAME             VOLUME                      TOUT DEVICE  SERVER
    vdisk0           vol2@primary-vds0                disk@0  primary

VCONS
    NAME             SERVICE                     PORT
    guest2           primary-vc0@primary         5001
```

The MAC address you'll need to use is `00:14:4f:fa:2b:e4`, noted in orange above. This is the MAC address for the server that appears on the virtual network `vnet0`. Note that this is *not* the MAC address listed under MAC.

The `list-bindings` command indicates that the domain's console is located on port 5001. You can reach the console on that port via `telnet localhost 5001`, and begin the network install:

```
fraser# telnet localhost 5001
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Connecting to console "guest2" in group "guest2" ....
Press ~? for control options ..


{0} ok boot vnet0 - install
```

The network install proceeds just as it does on a physical server; examples of JumpStart software configuration files are beginning with "Rules File Setup" on page 25.

Chapter 6

Configuring ZFS In The I/O Domain

For complete details on ZFS, please refer to the *Solaris ZFS Administration Guide*, 819-5461, available at *docs.sun.com*

Software volume management, using products such as Solaris Volume Manager software and ZFS, can be used to configure reliable storage for use in guest domains. With reliable disk storage configured in the I/O domain, the storage software is managed at a single point, and the reliability aspects are transparent to the guest domains. ZFS is a particularly useful technology for establishing redundancy in I/O domains, as it has features beyond what simple volume management and hardware RAID can accomplish. Snapshots can create point-in-time copies of guest domain virtual disks, and clones can make permanent copies of them for use in other virtual machines. This is an excellent way to make multiple copies of guest domains for various development phases (develop, test, production), and for horizontal scaling (Web servers, for example). For whatever the purpose, ZFS snapshots and clones are practically instant, and their copy-on-write semantics makes efficient use of storage.

Implementing ZFS And Cloning Guest Domain Volumes

This section works through a hypothetical example of how you might make a copy of a boot disk for an existing application and then clone multiple instances of it for use by multiple Logical Domains. This process is useful, for example, if you need to make multiple instances of a horizontally-scaled application. The sequence of snapshots and clones is illustrated in Figure 3:

1. Copy the boot disk image for a working configuration into a ZFS filesystem.

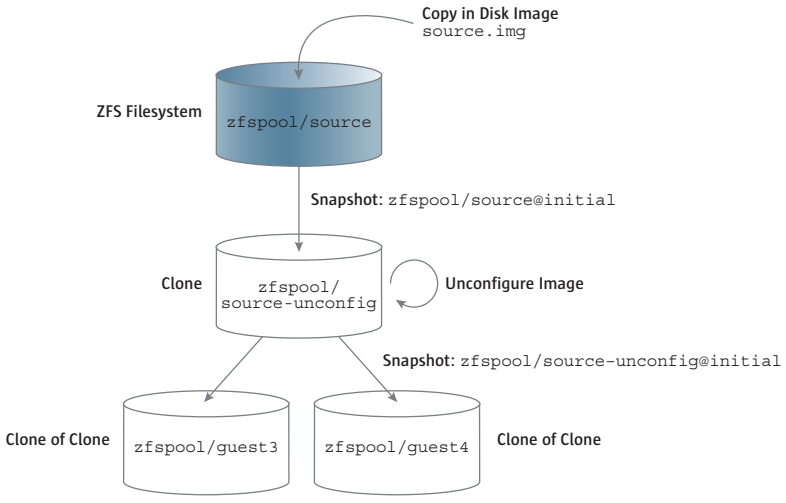

Figure 3. Sequence of snapshots and clones to create multiple instances of an application environment for use by multiple Logical Domains. The original ZFS filesystem shaded in blue is the only one that consumes significant amounts of disk space.

2. Create a snapshot and a clone that you then unconfigure. This step preserves the original boot disk image so that you can use it again for other purposes or roll back to it in case you make a mistake.
3. Create a snapshot of the filesystem containing the unconfigured image and make multiple clones of it, illustrating that you can make clones of clones.

Create A ZFS Pool

First we create the ZFS pool. Since we have used the first two of four drives on our Sun Fire T2000 server to demonstrate hardware RAID, we'll allocate the second two drives to a ZFS pool with software mirroring enabled. The simplest way to do this would be to simply allocate the two entire drives to the ZFS pool with the command:

```
fraser# zpool create zfspool mirror c1t2d0 c1t3d0
```

In our case, since we want to save some physical disk partitions for future use, we allocate two disk *partitions* to a zfs pool. We first use the `format` command to print the partition table on drive 3, `c1t2d0`:

```
Part      Tag    Flag     Cylinders         Size            Blocks
  0       root    wm       0 -  1030       10.01GB    (1031/0/0)   20982912
  1       swap    wu    1031 -  1237        2.01GB    (207/0/0)     4212864
  2     backup    wu       0 - 14086      136.71GB    (14087/0/0) 286698624
  3 unassigned    wm       0                 0         (0/0/0)             0
  4 unassigned    wm       0                 0         (0/0/0)             0
  5 unassigned    wm       0                 0         (0/0/0)             0
  6        usr    wm    1238 - 14086      124.69GB    (12849/0/0) 261502848
  7 unassigned    wm       0                 0         (0/0/0)             0
```

Just to make sure that the fourth drive is partitioned exactly the same, we use a powerful (but potentially dangerous) set of commands to copy drive 3's partitioning to drive 4. This set of commands makes the task easy, but be sure that your arguments are correct, as it's a very fast way to make existing data inaccessible if you accidentally partition the wrong drive:

```
fraser# prtvtoc /dev/rdsk/c1t2d0s2 | fmthard -s - /dev/rdsk/c1t3d0s2
```

Now we create a ZFS pool called `zfspool` that uses the `usr` slice of each drive:

```
fraser# zpool create zfspool mirror c1t2d0s6 c1t3d0s6
```

Now we create a ZFS filesystem named `source` in which we can store files that are then provided to guest domains as virtual disks:

```
fraser# zfs create zfspool/source
```

We copy a flat file that contains the boot disk for another guest domain into the source filesystem. The source file we'll work with from here on is `source.img`.

```
fraser# cp /domains/s2image.img /zfspool/source/source.img
```

Using the `zfs list`, command note that we currently are using 8 GB for the image, which is the only file stored in the ZFS pool:

```
fraser# zfs list
NAME             USED  AVAIL  REFER  MOUNTPOINT
zfspool         8.00G   114G  25.5K  /zfspool
zfspool/source  8.00G   114G  8.00G  /zfspool/source
```

Clone And Unconfigure

Since the file `source.img` in the ZFS filesystem `zfspool/source` contains a working image, we preserve the ability to roll back to that image at any time by making a clone of it before we go further. The steps in this section aren't necessary; they're provided just to give an example of what you might want to do in a real-world situation.

To create a clone, you first snapshot a ZFS filesystem, and then make a clone of it. The cloning process makes a permanent copy of the snapshot. Note that snapshots operate on filesystems, not files, so a good practice is not to load lots of disk images into a single filesystem, rather create a filesystem for a single image, or a collection of related images. In this example, we create a snapshot of the source filesystem named `initial`, and then we clone the snapshot and create a new filesystem `zfspool/source-unconfig` at the same time. Now you see that we have two filesystems in the ZFS pool:

```
fraser# zfs snapshot zfspool/source@initial
fraser# zfs clone zfspool/source@initial zfspool/source-unconfig
fraser# ls
source           source-unconfig
```

Note that the clone of the 8 GB disk image uses 22.5 KB of storage space since all of its blocks simply reference the existing `zfspool/source` filesystem:

```
fraser# zfs list
NAME                      USED  AVAIL  REFER  MOUNTPOINT
zfspool                  8.00G   114G  27.5K  /zfspool
zfspool/source           8.00G   114G  8.00G  /zfspool/source
zfspool/source@initial       0      -  8.00G  -
zfspool/source-unconfig  22.5K   114G  8.00G  /zfspool/source-unconfig
```

The clone of `source.img` is the disk image that we'll unconfigure so that it no longer has a system identity as defined by host name, IP address, time zone, etc. Since we'll only be using this file temporarily, let's hijack an existing Logical Domain by attaching this file as its boot disk just while we do the unconfigure step. In this example, we'll first detach the guest's existing virtual disk device named `vol2@primary-vds0`, and add a new virtual disk device, the `source.img` file from our cloned filesystem, as the same name, `vol2@primary-vds0`. We start with `guest2` in the `stopped` state and then `start` it once we've switched its disks:

```
fraser# ldm unbind-domain guest2
fraser# ldm remove-vdsdev vol2@primary-vds0
fraser# ldm add-vdsdev /zfspool/source-unconfig/source.img
        vol2@primary-vds0
fraser# ldm bind guest2
fraser# ldm start guest2
LDom guest2 started
```

Now we `telnet` to the correct port number on `localhost` for reaching `guest2`'s console, boot the domain if necessary, and run the `sys-unconfig` command. If we do anything unintended, we can always delete the clone and revert back to the copy of `source.img` in the `zfspool/source filesystem`.

```
# sys-unconfig
                        WARNING

This program will unconfigure your system.  It will cause it
to revert to a "blank" system - it will not have a name or know
about other systems or networks.

This program will also halt the system.

Do you want to continue (y/n) ? y
```

The `sys-unconfig` command shuts down the system, and now we can restore `guest2`'s original disk image:

```
fraser# ldm unbind-domain guest2
fraser# ldm remove-vdsdev vol2@primary-vds0
fraser# ls /domains
guest1image  s2image.img
fraser# ldm add-vdsdev /domains/s2image.img vol2@primary-vds0
fraser# ldm bind guest2
```

Clone And Run The Unconfigured System

Now that we have an unconfigured system, we can clone the filesystem containing its boot disk multiple times. Then we can attach each cloned filesystem's `source.img` file to a new Logical Domain. In this example, we create a snapshot called `initial` of the filesystem containing the unconfigured disk image. We create two clones, `guest3` and `guest4`, and note that we have still only used approximately 8 GB of storage in the ZFS pool:

```
fraser# zfs snapshot zfspool/source-unconfig@initial
fraser# zfs clone zfspool/source-unconfig@initial zfspool/guest3
fraser# zfs clone zfspool/source-unconfig@initial zfspool/guest4
fraser# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
zfspool                          8.04G   114G  29.5K  /zfspool
zfspool/guest3                       0   114G  8.00G  /zfspool/guest3
zfspool/guest4                       0   114G  8.00G  /zfspool/guest4
zfspool/source                   8.00G   114G  8.00G  /zfspool/source
zfspool/source@initial               0      -  8.00G  -
zfspool/source-unconfig          31.3M   114G  8.00G  /zfspool/source-
unconfig
zfspool/source-unconfig@initial      0      -  8.00G
```

Here are the steps to create `guest3`, a Logical Domain with two virtual CPUs, 2 GB of main memory, and a boot disk whose image is `source.img` contained in the ZFS filesystem `zfspool/guest3`. Since we aren't running a production environment, we set the domain not to boot automatically.

```
fraser# ldm create guest3
fraser# ldm set-vcpu 2 guest3
fraser# ldm set-mem 2g guest3
fraser# ldm add-vnet vnet0 primary-vsw0 guest3
fraser# ldm add-vdsdev /zfspool/guest3/source.img vol3@primary-vds0
fraser# ldm add-vdisk vdisk0 vol3@primary-vds0 guest3
fraser# ldm set-variable auto-boot\?=false guest3
fraser# ldm set-variable boot-device=/virtual-devices@100/channel-
devices@200/disk@0:a guest3
fraser# ldm bind guest3
```

Now you can use `ldm start` to start the logical domain. If `guest3` is the third logical domain you've created, you can `telnet` to `localhost` port `5002` to connect to the console and boot the domain. Once booted, the system configuration process starts, and you will be asked to provide system identity information just as if you opened up a new server out of the box. In a real environment, you would have a new instance of a "golden master" application instance that you had configured in the source Logical Domain.

The space savings of using clones is enormous. The `ldm list` command shows that, after creating two new domains and configuring them, we've used only approximately 50 MB per domain:

```
fraser# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
zfspool                          8.13G   114G  29.5K  /zfspool
zfspool/guest3                   50.4M   114G  8.00G  /zfspool/guest3
zfspool/guest4                   51.7M   114G  8.00G  /zfspool/guest4
zfspool/source                   8.00G   114G  8.00G  /zfspool/source
zfspool/source@initial               0      -  8.00G  -
zfspool/source-unconfig          31.3M   114G  8.00G  /zfspool/source-
unconfig
zfspool/source-unconfig@initial      0      -  8.00G  -
```

Save Your Configuration

After you make changes to your Logical Domains configuration, always remember to save the state in the service processor so that it remembers your configuration after a power cycle. This set of commands removes the existing configuration `fourguests` and replaces it with the current configuration:

```
fraser# ldm remove-spconfig fourguests
fraser# ldm add-spconfig fourguests
```

Chapter 7

Configuring Volume Management In Guest Domains

Please refer to the *Solaris Volume Manager Administration Guide* at *docs.sun.com* for complete details about the software.

The third architectural level at which I/O redundancy can be configured is within the I/O domain itself. Volume management software, such as Solaris Volume Manager software or ZFS, manages multiple virtual disks and performs the necessary replication from the guest domain. Of the two choices, Solaris Volume Manager software is the one that supports mirrored boot disks and is the one discussed here.

In cases where two I/O domains are used to support multiple paths to external storage, volume management in the guest domain is the preferred approach because there is no single point of failure along the communication path other than the server itself. In the case of a single I/O domain accessing internal disk storage, the topic of this article, replicating storage within the guest domain is less efficient because it imposes the overhead of dual virtual-to-physical paths in the guest. Every disk write must traverse the path to the virtual device twice, whereas configuring the replication in the I/O domain puts the I/O closer to the physical device, which is more efficient.

Setting Up Volume Management Through Network Install

A brief but essential guide to setting up a network install server is provided in the Sun BluePrints Series article: *Building a JumpStart Infrastructure*, 816-0428, available at *www.sun.com/blueprints*. See also the *Solaris OS Installation Guide: Network-Based Installations* (820-0177) and *Custom JumpStart and Advanced Installations* (820–0179) from *docs.sun.com*.

Nevertheless, if your existing best practices dictate using volume management in the guest OS, or if you need to create a development environment that re-creates a real one, you can use Solaris Volume Manager software and ZFS in guest domains just as you do on a physical system. One datacenter best practice is to establish disk mirroring during an install with JumpStart software, so this section illustrates how to set up root disk mirroring with Solaris Volume Manager software as part of the network install process. This section assumes that you're familiar with the process of setting up JumpStart software, so the process we describe is reduced to the essentials.

Set Up The Guest Domain

First set up a guest domain with two virtual disks on which you wish to mirror using Solaris Volume Manager software. Solaris Volume Manager software uses a quorum algorithm that requires half plus one of its state databases to be available and valid in order for the mirrored set of disks to be used. This means that if a single disk fails on a two-disk system where each disk holds half of the state databases, you'll have to manually delete the failed disk's state databases in order to boot. Because we have the benefit of being able to use virtual disks, they're cheap, and we can create a small, 16 MB, third disk that acts as a tie breaker.

In the real world, you should create three virtual disks whose contents are stored on three different physical disks. In this example, since we have already allocated half the server's disks to a RAID set and half to a ZFS pool, we create virtual disks from space

that is already mirrored. This is an inefficient thing to do, however for the purposes of this example, the commands are correct.

We use Logical Domain `guest4` that already is set up using a cloned disk from the previous section. We create a second flat file as a mirror, and a third flat file as the quorum disk for Solaris Volume Manager software. Then we save the configuration so that it will survive a power cycle of the server:

```
fraser# mkfile 8g /zfspool/guest4/mirror.img
fraser# mkfile 16M /domains/guest4quorum.img
fraser# ldm stop guest4
fraser# ldm unbind guest4
fraser# ldm add-vdsdev /zfspool/guest4/mirror.img vol5@primary-vds0
fraser# ldm add-vdisk vdisk1 vol5@primary-vds0 guest4
fraser# ldm add-vdsdev /domains/guest4quorum.img vol6@primary-vds0
fraser# ldm add-vdisk vdisk2 vol6@primary-vds0 guest4
fraser# ldm bind guest4
fraser# ldm remove-spconfig fourguests
fraser# ldm add-spconfig fourguests
```

Connect to the domain's console and verify that it has three disks attached:

```
{0} ok show-disks
a) /virtual-devices@100/channel-devices@200/disk@2
b) /virtual-devices@100/channel-devices@200/disk@1
c) /virtual-devices@100/channel-devices@200/disk@0
q) NO SELECTION
Enter Selection, q to quit: q
```

Rules File Setup

The rules file on the network install server detects the type of install client and causes the install process to follow the specific profile file that it specifies. The following rules file line specifies the `profile.mirror.s4` profile for all installs onto servers with CoolThreads technology (sun4v architecture), and specifies no pre-install or post-install scripts to run:

```
arch sun4v - Profiles/profile.mirror.s4 -
```

Profile File Setup

Following is the profile file that we used to install an existing Solaris Flash archive:

```
#
# Jumpstart Profile to install a Solaris Flash archive in a LDom
# Use a third, small virtual disk as a quorum disk for SVM
#
install_type            flash_install
archive_location        nfs js:/jumpstart/Archives/s4mirror.flar
#
partitioning    explicit
#
# Mirror root and swap
#
filesys mirror:d10 c0d0s0 c0d1s0 free /
filesys mirror:d20 c0d0s1 c0d1s1 1024 swap
#
# Create three sets of state databases so there should
# always be a quorum of databases if a single disk fails.
# Two databases per volume for media redundancy.
#
metadb c0d0s7 size 8192 count 2
metadb c0d1s7 size 8192 count 2
metadb c0d2s2 size 8192 count 2
```

Sysidcfg File Setup

If there is any black magic to setting up JumpStart software, it's creating a `sysidcfg` file that enables an unattended install. The requirements change between Solaris OS releases, so they tend to create a moving target. The following `sysidcfg` file works for installations of the Solaris 10 8/07 operating system. The particular trick in this file is that the time server must be specified as `localhost` to prevent dropping into an interactive install:

```
timezone=US/Mountain
timeserver=localhost
root_password=678B3hGfBa01U
name_service=DNS {domain_name=loneagle.com search=loneagle.com
    name_server=ins(192.168.0.134)}
network_interface=vnet0 {protocol_ipv6=no hostname=s4
    ip_address=192.168.0.58 default_route=192.168.0.1
    netmask=255.255.255.192}
security_policy=NONE
system_locale=en_US
terminal=xterms
keyboard=US-English
service_profile=open
nfs4_domain=loneagle.com
```

Add Install Client

Make sure that you have set up the guest domain's IP address in your network install server's `/etc/hosts` file. Determine the guest domain's MAC address using the procedure described in "Creating A Logical Domain" on page 15. Then you can issue an `add_install_client` command that will set up the boot block, correct parameters in `/etc/bootparams`, and the MAC address in `/etc/ethers`:

```
js# ./add_install_client -e 00:14:4f:f8:2b:7e -s
js:/jumpstart/OS/Solaris10-2007-08 -c js:/jumpstart -p
js:/jumpstart/Sysidcfg.ldom-mirror s4 sun4v
Adding Ethernet number for s4 to /etc/ethers
updating /etc/bootparams
```

Install

Install the Solaris OS, or in this case the Solaris Flash archive, by attaching to the logical domain's console and issuing the command:

```
#{0} ok boot vnet0 - install
```

The installation process will set up the mirrored `root` and `swap` filesystems. During the install, you may see an error message while the state replicas are created:

```
metadb: s4: network/rpc/meta:default: failed to enable/disable SVM
service
```

And you may also see multiple errors at boot time similar to this:

```
NOTICE: mddb: unable to get devid for 'vdc', 0x7
```

These relate to known issues and do not interfere with Solaris Volume Manager software's operation.

Housekeeping

You will need to do the normal housekeeping that you should do when setting up a mirrored `root` partition with Solaris Volume Manager software, including setting up a boot block on the second disk in the mirror, and setting up alternate boot devices in OpenBoot software.

Install a boot block in the second disk of the mirror from within the guest domain so that you can boot from the second disk in the event that the first drive fails:

```
s4# cd /usr/platform/`/usr/bin/uname -i`/lib/fs/ufs
s4# /usr/sbin/installboot ./bootblk /dev/rdsk/c0d1s0
```

Then, at the OpenBoot software prompt, create device aliases for your primary and secondary boot disks and set your boot-device to boot from them in order:

```
{0} ok nvalias primary_root /virtual-devices@100/channel-devices@200/disk@0
{0} ok nvalias backup_root /virtual-devices@100/channel-devices@200/disk@1
{0} ok setenv boot-device primary_root backup_root
boot-device =           primary_root backup_root
{0} ok nvstore
```

Test

You can test your configuration by removing disks virtually; shut down the guest domain and remove one of the disks. In this example, we remove `disk1`:

```
fraser# ldm unbind guest4
fraser# ldm remove-vdisk vdisk1 guest4
fraser# ldm bind guest4
fraser# ldm start guest4
```

You can verify from the OpenBoot software that `disk1` is missing:

```
{0} ok show-disks
a) /virtual-devices@100/channel-devices@200/disk@2
b) /virtual-devices@100/channel-devices@200/disk@0
```

The domain will boot normally because it has a quorum of state databases. After testing, you can replace the ‘failed’ disk and reboot with both disks available:

```
fraser# ldm unbind guest4
fraser# ldm add-vdisk vdisk1 vol5@primary-vds0 guest4
fraser# ldm bind guest4
fraser# ldm start guest4
LDom guest4 started
```

Now that you've restored the drive whose failure was simulated, the `metastat` command executed in the guest domain indicates that you need to resync the mirrors. You can do so with the `metareplace` command:

```
s4# metareplace -e d10 c0d1s0
s4# metareplace -e d20 c0d1s1
```

Chapter 8
Conclusion

The virtual world enabled by Logical Domains on Sun servers with CoolThreads technology opens up a broad range of possibilities for organizations needing to implement data reliability and availability mechanisms to match those on physical servers. Data availability can be increased by configuring multiple paths to network or disk I/O resources. Data reliability can be increased by configuring redundant storage media through mirroring and RAID techniques.

The purpose of this Sun BluePrints series article is to address reliability techniques using Logical Domains and the internal disk resources of the Sun servers on which they run. Because every Sun server with CoolThreads technology is configured with a single controller for on-board disk storage, the techniques available boil down to implementing mirroring and RAID at one of three architectural levels:

- In the hardware, using the disk controller itself,
- In the I/O domain, using Solaris Volume Manager or ZFS software, or
- In the guest domain, using Solaris Volume Manager or ZFS software.

Although our recommendations are to implement redundancy at as low a level as possible given the limitation of using a single I/O domain, there are plenty of reasons to use higher architectural levels, including the following:

- The Zettabyte File System integrates volume management and filesystems that support mirroring and RAID, and also snapshots and clones. These mechanisms can be used to create new Logical Domains for development and testing, or for implementing a number of identical Logical Domains for horizontal scaling. Snapshots can be used for backups, and they can be used to create point-in-time copies that can be moved to remote locations for disaster-recovery purposes. What's best about ZFS is that snapshots and clones are created almost instantly, and their copy-on-write semantics make efficient use of storage resources.
- Using Solaris Volume Manager or ZFS software in guest domains allows existing physical configurations to be re-created almost exactly in a virtual world. Instead of these volume management mechanisms accessing physical disks, they access virtual disks that are supported underneath with flat files, physical drives, or disk partitions. The ability to re-create the world of physical servers so faithfully in the virtual world allows IT organizations to continue using their existing best practices. It also provides a low-cost, low-impact test bed for creating disk failure scenarios and validating the procedures for handling them.

This article is the first in a series designed to address the issues of translating I/O reliability and availability practices from the physical world into the world of Logical Domains. Future articles may address high availability network and disk I/O

configurations using multiple PCI buses and the trade-offs of implementing the techniques at various architectural levels.

About The Author

Peter A. Wilson has more than 16 years of industry experience, 12 of which have been with Sun, serving in a wide variety of hardware, software, systems and product marketing roles. Peter moved from the United Kingdom to the US in 2000 to lead the customer tests of Sun's Netra™ and fault-tolerant servers. Peter is currently a technical marketing engineer responsible for all aspects of Sun's servers with CoolThreads technology. Peter holds a M.Eng (Master of Engineering) degree in Microelectronics and Software Engineering from the University of Newcastle-upon-Tyne, U.K.

Acknowledgments

The author would like to thank Steve Gaede, an independent technical writer and engineer, for working through the issues and configurations described in this paper and developing this article based on them. Thanks also to Maciej Browarski, Alexandre Chartre, James MacFarlane, and Narayan Venkat for their help in understanding some of the nuances of using Solaris Volume Manager software.

References

References to relevant Sun publications are provided in sidebars throughout the article.

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is

`http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at:

`http://www.sun.com/blueprints/online.html`

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN (9786) **Web** sun.com

Printed in USA 6/2008