3 Oracle VM Server for SPARC

Logical Domains (now Oracle VM Server for SPARC) is a virtualization technology that creates SPARC virtual machines, also called domains. This new style of hypervisor permits operation of virtual machines with less overhead than traditional designs by changing the way guests access physical CPU, memory, and I/O resources. It is ideal for consolidating multiple complete Oracle Solaris systems onto a modern powerful, low-cost, energy-efficient SPARC server, especially when the virtualized systems require the capability to have different kernel levels.

The Logical Domains technology is available on systems based on SPARC chip multithreading technology (CMT) processors. These include the Sun SPARC Enterprise T5x20/T5x40 servers, Sun Blade T6320/T6340 server modules, and Sun Fire T1000/T2000 systems. The chip technology is integral to Logical Domains because it leverages the large number of CPU threads available on these servers. At this writing, that number can be as many as 128 threads in a single-rack unit server and as many as 256 threads in a four-rack unit server. Logical Domains is available on all CMT processors without additional license or hardware cost.

3.1 Overview of Logical Domains Features

Logical Domains creates virtual machines, usually called *domains*. Each appears to have its own SPARC server. A domain has the following resources:

- CPUs
- RAM

- Network devices
- Disks
- Console
- OpenBoot environment
- Cryptographic accelerators (optional)

The next several sections describe properties of Logical Domains and explain how they are implemented.

3.1.1 Isolation

Each domain runs its own instance of Oracle Solaris 10 or OpenSolaris with its own accounts, passwords, and patch levels, just as if each had its own separate physical server. Different Solaris patch and update levels run at the same time on the same server without conflict. Some Linux distributions can also run in domains. Logical Domains support was added to the Linux source tree at the 2.6.23 level.

Domains are isolated from one another. Thus each domain is individually and independently started and stopped. As a consequence, a failure in one domain—even a kernel panic or CPU thread failure—has no effect on other domains, just as would be the case for Solaris running on multiple servers.

3.1.2 Compatibility

Oracle Solaris and applications in a domain are highly compatible with Solaris running on a physical server. Solaris has long had a binary compatibility guarantee; this guarantee has been extended to Logical Domains, making no distinction between running as a guest or on bare metal. Solaris functions essentially the same in a domain as on a nonvirtualized system.

3.1.3 Real and Virtual CPUs

One of the distinguishing features of Logical Domains compared to other hypervisors is the assignment of CPUs to individual domains. This approach has a dramatic benefit in terms of increasing simplicity and reducing the overhead commonly encountered with hypervisor systems.

Traditional hypervisors time-slice physical CPUs among multiple virtual machines in an effort to provide CPU resources. Time-slicing was necessary because the number of physical CPUs was relatively small compared to the desired number of virtual machines. The hypervisor also intercepted and emulated privileged instructions that would change the shared physical machine's state (such as interrupt masks, memory maps, and other parts of the system environment), thereby violating the integrity of separation between guests. This process is complex and creates CPU overhead. Context switches between virtual machines can require hundreds or even thousands of clock cycles. Each context switch to a different virtual machine requires purging cache and translation lookaside buffer (TLB) contents because identical virtual memory addresses refer to different physical locations. This scheme increases memory latency until the caches become filled with fresh content, only to be discarded when the next time slice occurs.

In contrast, Logical Domains is designed for and leverages the chip multithreading (CMT) UltraSPARC T1, T2, and T2 Plus processors. These processors provide many CPU threads, also called *strands*, on a single processor chip. Specifically, the UltraSPARC T1 processor provides 8 processor cores with 4 threads per core, for a total of 32 threads on a single processor. The UltraSPARC T2 and T2 Plus processors provide 8 cores with 8 threads per core, for a total of 64 threads per chip. From the Oracle Solaris perspective, each thread is a CPU. This arrangement creates systems that are rich in dispatchable CPUs, which can be allocated to domains for their exclusive use.

Logical Domains technology assigns each domain its own CPUs, which are used with native performance. This design eliminates the frequent context switches that traditional hypervisors must implement to run multiple guests on a CPU and to intercept privileged operations. Because each domain has dedicated hardware circuitry, a domain can change its state—for example, by enabling or disabling interrupts—without causing a trap and emulation. The assignment of strands to domains can save thousands of context switches per second, especially for workloads with high network or disk I/O activity. Context switching still occurs within a domain when Solaris dispatches different processes onto a CPU, but this is identical to the way Solaris runs on a non-virtualized server.

One mechanism that CMT systems use to enhance processing throughput is detection of a cache miss, followed by a hardware context switch. Modern CPUs use onboard memory called a *cache*—a very high-speed memory that can be accessed in just a few clock cycles. If the needed data is present in memory but is not in this CPU's cache, a *cache miss* occurs and the CPU must wait dozens or hundreds of clock cycles on any system architecture. In essence, the CPU affected by the cache miss stalls until the data is fetched from RAM to cache. On most systems, the CPU sits idle, not performing any useful work. On those systems, switching to a different process would require a software context switch that consumes hundreds or thousands of cycles.

In contrast, CMT processors avoid this idle waiting by switching execution to another CPU strand on the same core. This hardware context switch happens in a single clock cycle because each hardware strand has its own private hardware context. In this way, CMT processors use what is wasted (stall) time on other processors to continue doing useful work.

This feature is highly effective whether Logical Domains are in use or not. Nonetheless, a recommendation for Logical Domains is to reduce cache misses by allocating domains so they do not share per-core L1 caches. The simplest way to do so is to allocate domains with a multiple of the CPU threads per core—for example, in units of 8 threads on T2-based systems. This approach ensures that all domains have CPUs allocated on a core boundary and not shared with another domain. Actual savings depend on the system's workload, and may be of minor consideration when consolidating old, slow servers with low utilization.

3.2 Logical Domains Implementation

Logical Domains are implemented using a very small hypervisor that resides in firmware and keeps track of the assignment of logical CPUs, RAM locations, and I/O devices to each domain. It also provides logical channels for communication between domains and between domains and the hypervisor.

The Logical Domains hypervisor is intentionally kept as small as possible for simplicity and robustness. Many tasks traditionally performed within a hypervisor kernel (such as the management interface and performing I/O for guests) are offloaded to special-purpose domains, as described in the next section.

This scheme has several benefits. Notably, a small hypervisor is easier to develop, manage, and deliver as part of a firmware solution embedded in the platform, and its tight focus helps security and reliability. This design also adds redundancy: Shifting functions from a monolithic hypervisor to privileged domains insulates the system from a single point of failure. As a result, Logical Domains have a level of resiliency that is not available in traditional hypervisors of the VM/370, z/VM, or VMware ESX style. Also, this design makes it possible to leverage capabilities already available in Oracle Solaris, providing access to features for reliability, performance, scale, diagnostics, development tools, and a large API set. It has proven to be an extremely effective alternative to developing all these features from scratch.

3.2.1 Domain Roles

Domains are used for different roles, and may be used for Logical Domain infrastructure or applications. The *control domain* is an administrative control point that runs Solaris or OpenSolaris and the Logical Domain Manager services. It has a privileged interface to the hypervisor, and can create, configure, start, stop, and destroy other domains. *Service domains* provide virtualized disk and network devices for other domains. *I/O domains* have direct access to physical I/O devices and are typically used as service domains to provide access to these devices. The control domain also is an I/O domain and can be used as a service domain. Applications generally run in *guest domains,* which are non-I/O domains using virtual devices provided by service domains. The domain structure and the assignment of CPUs are shown in Figure 3.1.

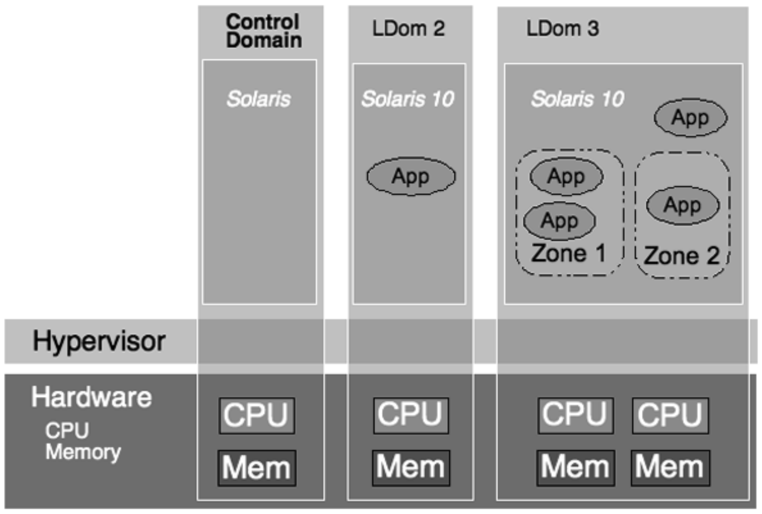


Figure 3.1 Control and Guest Domains

The definition of a domain includes its name, amount of RAM and number of CPUs, its I/O devices, and any optional hardware cryptographic accelerators. Domain definitions are made by using the command-line interface in the control domain, using the Oracle Enterprise Manager Ops Center product, or for the initial configuration, using the Logical Domains Configuration Assistant.

3.2.1.1 Domain Relationships

Each server has exactly one control domain, found on the instance of Solaris that was first installed on the system. It runs Logical Domain Manager services, which are accessed by a command-line interface provided by the `ldm` command. These Logical Domain Manager services include a "constraint manager" that decides how to assign physical resources to satisfy the specified requirements (the "constraints") of each domain.

There can be as many I/O domains as there are physical PCI buses on the system. An I/O domain is often used as a service domain to run virtual disk services and virtual network switch services that provide guest domain virtual I/O devices.

Finally, there can be as many guest domains as are needed for applications, subject to the limits associated with the installed capacity of the server. At the time of this writing, the maximum number of domains on a CMT system was 128, including control and service domains, even on servers with 256 threads such as the T5440. While it is possible to run applications in control or service domains, it is highly recommended, for stability reasons, to run applications only in guest domains. Applications that require optimal I/O performance can be run in an I/O domain to avoid virtual I/O overhead, but it is recommended that such an I/O domain not be used as a service domain.

A simple configuration consists of a single control domain also acting as a service domain, and some number of guest domains. A more complex configuration could use redundant service domains to provide failover in case of a domain failure or loss of a path to an I/O device.

3.2.2 Dynamic Reconfiguration

CPUs and virtual I/O devices can be dynamically added to or removed from a Logical Domain without requiring a reboot. An Oracle Solaris instance running in a guest domain can immediately make use of a dynamically added CPU for additional capacity and can also handle the removal of all but one of its CPUs. Virtual disk and network resources can also be nondisruptively added to or removed from a domain, and a guest domain can make use of a newly added virtual disk or network device without a reboot.

3.2.3 Virtual I/O

Logical Domains technology abstracts underlying I/O resources to virtual I/O. It is not always possible to give each domain direct access to a bus, an I/O memory mapping unit (IOMMU), or devices, so Logical Domains provides a virtual I/O (VIO) infrastructure to provide access to these resources.

Virtual network and disk I/O is provided to Logical Domains by service domains. A service domain runs Solaris and usually has direct connections to a PCI bus connected to physical network and disk devices. In that configuration, it is also an I/O domain. Likewise, the control domain is typically configured as a service domain. It is also an I/O domain, because it requires access to I/O buses and devices to boot up.

The virtual I/O framework allows service domains to export virtual network and disk devices to other domains. Guest domains use these devices exactly as if they were dedicated physical resources. Guest domains perform virtual I/O to virtual devices provided by service domains. Service domains then proxy guests' virtual I/O by performing I/O to back-end devices, which are usually physical devices. Virtual device characteristics are described in detail later in this chapter.

Guest domains have network and device drivers that communicate with I/O domains through Logical Domain Channels (LDCs) provided by the hypervisor. The addition of device drivers that use LDCs rather than physical I/O is one of the areas in which Solaris has been modified to run in a logical domain—, an example of paravirtualization discussed in Chapter 1, "Introduction to Virtualization." LDCs provide communications channels between guests, and an API for enqueuing and dequeuing messages that contain service requests and responses. Figure 3.2 shows the relationship between guest and service domains and the path of I/O requests and responses.

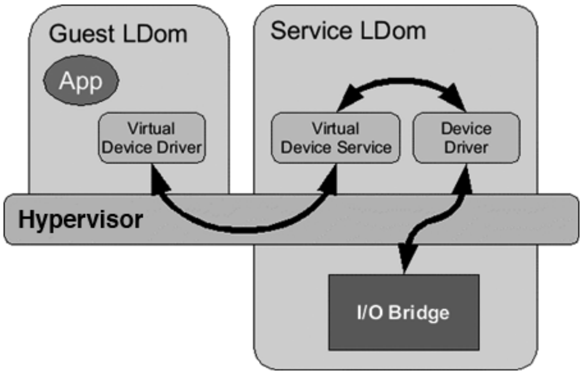


Figure 3.2 Service Domains Provide Virtual I/O

Shared memory eliminates the overhead associated with copying buffers between domains. The processor's memory mapping unit (MMU) is used to map shared buffers in physical memory into the address spaces of a guest and an I/O domain. This strategy helps implement virtual I/O efficiently: Instead of copying the results of a disk read from its own memory to a guest domain's memory, an I/O domain can read directly into a buffer it shares with the guest. This highly secure mechanism is controlled by hypervisor management of memory maps.

I/O domains are designed for high availability. Redundant I/O domains can be set up so that system and guest operation can continue if a path fails, or if an I/O domain fails or is taken down for service. Logical Domains provides virtual disk multipathing, thereby ensuring that a virtual disk can remain accessible even if a service domain fails. Domains can use IP network multipathing (IPMP) for redundant network availability.

3.3 Details of Domain Resources

Logical Domains technology provides flexible assignment of hardware resources to domains, with options for specifying physical resources for a corresponding virtual resource.

3.3.1 Virtual CPUs

As mentioned in the section "Real and Virtual CPUs," each domain is assigned exclusive use of a number of CPUs, also called *threads* or *strands*. Within a domain, these are called *virtual CPUs* (vCPUs).

The granularity of assignment is a single vCPU. A domain can have from one vCPU up to all the vCPUs on the server. On UltraSPARC T1 systems (T1000 and T2000), the maximum is 8 cores with 4 threads, for a total of 32 vCPUs. On UltraSPARC T2 and T2 Plus systems, the maximum is 8 cores with 8 threads each, for a total of 64 vCPUs per chip. Systems with the T2 Plus chip can have multiple chips per server: The T5140 and T5240 servers have 2 T2 Plus chips for a total of 16 cores and 128 vCPUs, while the T5440 has 4 T2 Plus chips with 32 cores and 256 vCPUs.

Virtual CPUs should be assigned to domains on core boundaries. This strategy prevents "false cache sharing," which can reduce performance when multiple domains share a CMT core and compete for the same L1 cache. To avoid this problem, vCPU quantities equivalent to entire cores to each domain should be allocated. For example, you should allocate vCPUs in units of 8 vCPUs on T2 and T2 Plus servers. Of course, this tactic may be overkill for some workloads, and administrators need not excessively concern themselves when defining domains to accommodate the light CPU requirements needed to consolidate small, old, or low utilization servers. Figure 3.3 is a simplified diagram of the threads, cores, and caches in a SPARC CMT chip.

The number of CPUs in a domain can be dynamically and nondisruptively changed while the domain is running. Oracle Solaris commands such as `vmstat` and `mpstat` can be used within the domain to monitor its CPU utilization, just as on a dedicated server. The `ldm list` command can be used in the control domain to display each domain's CPU utilization. A change in the quantity of vCPUs in a running domain takes effect immediately. The number of CPUs can be managed automatically with the Logical Domains Dynamic Resource Manager, which is discussed later in this chapter.

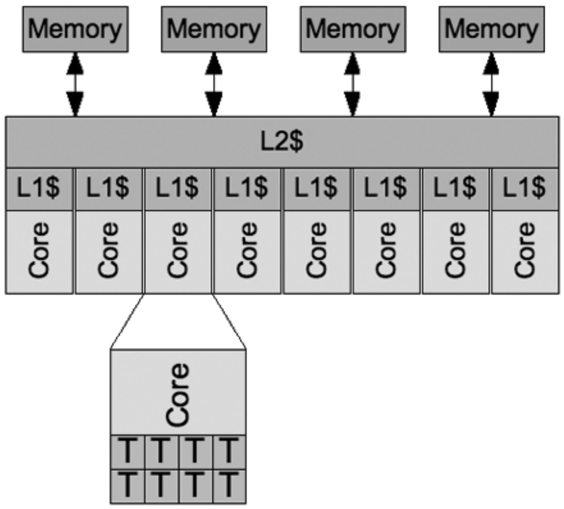


Figure 3.3 CMT Cores, Threads, and Caches

3.3.2 Virtual Network Devices

Guests have one or more virtual network devices connected to virtual Layer 2 network switches provided by service domains. Virtual network devices can be on the same or different virtual switches so as to connect a domain to multiple networks, provide increased availability using IPMP (IP Multipathing), or increase the bandwidth available to a guest domain.

From the guest perspective, virtual network interfaces are named `vnet`*N*, where *N* is an integer starting from 0 for the first virtual network device defined for a domain. In fact, the simplest way to determine if an Oracle Solaris instance is running in a guest domain (specifically, a domain that is not an I/O domain) is to issue the command `ifconfig -a` and see if the network interfaces are `vnet0`, `vnet1`, and so on, rather than real devices like `nxge0` or `e1000g0`. Virtual network devices can be assigned static IP or dynamic IP addresses, just as with physical network devices.

3.3.2.1 MAC Addresses

Every virtual network device has its own MAC address. This is different from Oracle Solaris Containers, where a single MAC address is usually shared by all Containers in a Solaris instance. MAC addresses can be assigned manually or automatically from the reserved address range of `00:14:4F:F8:00:00` to `00:14:4F:FF:FF:FF`. The bottom half of the address range is used for automatic assignments; the other 256K addresses can be used for manual assignment.

The Logical Domains manager implements duplicate MAC address detection by sending multicast messages with the address it wants to assign and listening for a possible response from another machine's Logical Domains manager saying the address is in use. If such a message comes back, it randomly picks another address and tries again. The message time-to-live (TTL) is set to 1, and can be changed by the SMF property `ldmd/hops`. Recently freed MAC addresses from removed domains are used first, to help prevent DHCP servers from exhausting the number of addresses available.

3.3.2.2 Network Connectivity

Virtual switches are usually assigned to a physical network device, permitting traffic between guest domains and the network segment to which the device is connected. Network traffic between domains on the same virtual switch does not travel to the virtual switch or to the physical network, but rather is implemented by a fast memory-to-memory transfer between source and destination domains using dedicated LDCs. Virtual switches can also be established without a connection to a physical network device, which creates a private secure network not accessible from any other server. Virtual switches can be configured for securely isolated VLANs, and can exploit features such as VLAN tagging and jumbo frames.

3.3.2.3 Hybrid I/O

Network Interface Unit (NIU) Hybrid I/O is an optimization feature available on servers based on the UltraSPARC T2 chip, the T5120 and T5220 servers, and the Sun Blade T6320 server module. It is an exception to the normal Logical Domains virtual I/O model, and provides higher performance network I/O. In hybrid mode, DMA resources for network devices are loaned to a guest domain so it can perform network I/O without going through an I/O domain. In this mode, a network device in a guest domain can transmit unicast traffic to and from external networks at essentially native performance. Multicast traffic, and network traffic to other domains on the same virtual switch are handled as described above.

In current implementations, there are two 10 GbE NIU `nxge`*`N`* devices per T2-based server. Each can support three hybrid I/O virtual network devices, for a total of six.

3.3.3 Virtual Disk

Service domains can have virtual disk services that export virtual block devices to guest domains. Virtual disks are based on back-end disk resources, which may be physical disks, disk slices, volumes, or files residing in ZFS or UFS file systems. These resources could include any of the following:

- A physical block device (disk or LUN)—for example, `/dev/dsk/c1t48d0s2`
- A slice of a physical device or LUN—for example, `/dev/dsk/c1t48d0s0`
- A disk image file residing in UFS or ZFS—for example, `/path-to-filename`
- A ZFS volume—for example,

  `zfs create -V 100m ldoms/domain/test/zdisk0`

  creates the back-end `/dev/zvol/dsk/ldoms/domain/test/zdisk0`
- A volume created by Solaris Volume Manager (SVM) or Veritas Volume Manager (VxVM)
- A CD ROM/DVD or a file containing an ISO image

A virtual disk may be marked as read-only. It also can be made exclusive, meaning that it can be given to only one domain at a time. This setting is available for disks based on physical devices rather than files, but the same effect can be provided for file back-ends by using ZFS clones. The advantages of ZFS—such as advanced mirroring, checksummed data integrity, snapshots, and clones—can be applied to both ZFS volumes and disk image files residing in ZFS. ZFS volumes generally provide better performance, whereas disk image files provide simpler management, including renaming, copying, or transmission to other servers.

In general, the best performance is provided by virtual disks backed by physical disks or LUNs, and the best flexibility is provided by file-based virtual disks or volumes, which can be easily copied, backed up, and, when using ZFS, cloned from a snapshot. Different kinds of disk back-ends can be used in the same domain: The system volume for a domain can use a ZFS or UFS file system back-end, while disks used for databases or other I/O intensive applications can use physical disks.

Redundancy can be provided by using virtual disk multipathing in the guest domain, with the same virtual disk back-end presented to the guest by different service domains. This provides fault tolerance for service domain failure. A time-out interval can be used for I/O failover if the service domain becomes unavailable. The `ldm` command syntax for creating a virtual volume lets you specify an MPXIO group. The following commands illustrate the process of creating a disk volume back-end served by both a control domain and an alternate service domain:

```
# ldm add-vdsdev mpgroup=foo \
  /path-to-backend-from-primary/domain ha-disk@primary-vds0
# ldm add-vdsdev mpgroup=foo \
  /path-to-backend-from-alternate/domain ha-disk@alternate-vds0
# ldm add-vdisk ha-disk ha-disk@primary-vds0 myguest
```

Multipathing can also be provided from a single I/O domain with multiplexed I/O (MPXIO), by ensuring that the domain has multiple paths to the same device—for example, two FC-AL HBAs to the same SAN array. You can enable MPxIO in the control domain by running the command `stmsboot -e`. That command creates a single, but redundant path to the same device. The single device is then configured into the virtual disk service. Perhaps most simply, insulation from a path or media failure can be provided by using a ZFS file pool with mirror or RAID-Z redundancy. These methods offer resiliency in case of a path failure to a device, but do not insulate the system from failure of a service domain.

3.3.4 Console and OpenBoot

Every domain has a console, which is provided by a virtual console concentrator (vcc). The vcc is usually assigned to the control domain, which then runs the Virtual Network Terminal Server daemon (*`vntsd`*) service.

By default, the daemon listens for localhost connections using the Telnet protocol, with a different port number being assigned for each domain. A guest domain operator connecting to a domain's console first logs into the control domain via the `ssh` command so that no passwords are transmitted in cleartext over the network; the `telnet` command can then be used to connect to the console.

Optionally, user domain console authorization can be implemented to restrict which users can connect to a domain's console. Normally, only system and guest domain operators should have login access to a control domain.

3.3.5 Cryptographic Accelerator

The processors in CMT systems are equipped with on-chip hardware cryptographic accelerators that dramatically speed up cryptographic operations. This technique improves security by reducing the CPU consumption needed for encrypted transmissions, and makes it possible to transmit secure traffic at wire speed. Each CMT processor core has its own hardware accelerator unit, making it possible to run multiple concurrent hardware-assisted cryptographic transmissions.

In the T1 processor used on the T1000 and T2000 servers, the accelerator performs modular exponentiation and multiplication, which are normally CPU-intensive portions of cryptographic algorithms. The accelerator, called the *Modular Arithmetic Unit* (MAU), speeds up public key cryptography (i.e., RSA, DSA, and Diffie-Hellman algorithms).

Although the T2 and T2 Plus chips include this function, the accelerator has additional functionality. This cipher/hash unit accelerates bulk encryption (RC4, DES, 3DES, AES), secure hash (MD5, SHA-1, SHA-256), other public key algorithms (elliptical curve cryptography), and error-checking codes (ECC, CRC32).

At this time, a cryptographic accelerator can be allocated only to domains that have at least one virtual CPU on the same core as the accelerator.

3.3.6 Memory

The Logical Domains technology dedicates real memory to each domain, instead of using virtual memory for guest address spaces and swapping them between RAM and disk, as some hypervisors do. This approach limits the number and memory size of domains on a single CMT processor to the amount that fits in RAM, rather than oversubscribing memory and swapping. As a consequence, it eliminates problems such as thrashing and double paging, which are experienced by hypervisors that run virtual machines in virtual memory environments.

RAM can be allocated to a domain in highly granular units—the minimum unit that can be allocated is 4 MB. The memory requirements of a domain running the Oracle Solaris OS are no different from running Solaris on a physical machine. If a workload needs 8 GB of RAM to run efficiently on a dedicated server, it will need the same amount when running in a domain.

3.3.7 Binding Resources to Domains

The Logical Domains administrator uses the `ldm` command to specify the resources required by each domain: the amount of RAM, the number of CPUs, and so forth. These parameters are sometimes referred to as the domain's constraints.

A domain that has been defined is said to be inactive until resources are bound to it by the `ldm bind` command. When this command is issued, the system selects the physical resources required by the domain's constraints and associates them with the domain. For example, if a domain requires 8 CPUs, the domain manager selects 8 CPUs from the set of online and unassigned CPUs on the system and gives them to the domain.

Until a domain is bound, the sum of the constraints of all domains can exceed the physical resources available on the server. For example, one could define 10 domains, each of which requires 8 CPUs and 8 GB of RAM on a machine with 64 CPUs and 64 GB of RAM. Only the domains whose constraints are met can be bound and started. In this example, the first 8 domains to be bound would boot. Additional domains can be defined for occasional or emergency purposes, such as a disaster recovery domain defined on a server normally used for testing purposes.

3.4 Installing Logical Domains and Building a Guest Domain

This section walks through the process of installing Logical Domains on a CMT server, creating a guest domain, and installing Oracle Solaris on it. The installed domain will then be set up as a master image for cloning further domains.

3.4.1 Verifying and Installing Firmware

All CMT servers and Blade Modules ship with firmware support for Logical Domains. Nevertheless, it is important to ensure that the current firmware for the server is installed, matching it to the version of Solaris running in the control domain and the version of Logical Domains management software to be installed. Current information can be obtained from `http://docs.sun.com`. The Release Notes, Reference Manual, and Administration Guide for Logical Domains software list the firmware levels needed on each server model. Use the instructions in the server's Installation Guide or Administration Guide for verifying and installing the firmware.

3.4.2 Installing Logical Domains Software

Examples in this section are taken from Logical Domains version 1.2. (Note: File names, command syntax, and screen output may vary slightly in future releases.) Logical Domains software can be downloaded from `http://www.sun.com/servers/coolthreads/ldoms/`. The software is delivered as a `.zip` file, which can be unpacked by issuing the following command (or something similar):

```
# unzip Ldoms_Manager-1_2.zip
```

Before installing the software, read the `README` file contained in the opened `.zip` file. It will describe any prerequisite patches to install. If any patches are required, download them from `http://sunsolve.sun.com` and install them using `patchadd`(1M) as directed.

At that point, you can run the install script, responding to any prompts as needed.

```
# cd /ldoms/LDoms1.2/LDoms_Manager-1_2/Install
# ./install-ldm

Welcome to the LDoms installer.

You are about to install the Logical Domains Manager package that will
enable you to create, destroy and control other domains on your
system.  Given the capabilities of the LDoms domain manager, you can
now change the security configuration of this Solaris instance using
the Solaris Security Toolkit.

Select a security profile from this list:

a) Hardened Solaris configuration for LDoms (recommended)
```

```
b) Standard Solaris configuration
c) Your custom-defined Solaris security configuration profile

Enter a, b, or c [a]:
...
...
LOGICAL DOMAINS CONFIGURATION

Once installed, you may configure your system for a basic Logical
Domains deployment. If you select "y" for the following question, the
Logical Domains Configuration Assistant will be launched following a
successful installation of packages.

(You may launch the LDoms Configuration Assistant at a later time with
the command /usr/sbin/ldmconfig, or use the GUI Configuration Assistant
which is bundled in the LDoms zip file - see README.GUI for more
details)

Select an option for configuration:

y) Yes, launch the LDoms Configuration Assistant after install
n) No thanks, I will configure LDoms myself later

Enter y or n [y]: n
Installing LDoms and Solaris Security Toolkit packages.
pkgadd -n -d "/ldoms/LDoms1.2/LDoms_Manager-1_2/Product" -a pkg_admin SUNWldm.v
Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Installation of <SUNWldm> was successful.
pkgadd -n -d "/ldoms/LDoms1.2/LDoms_Manager-1_2/Product" -a pkg_admin SUNWjass
Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Installation of <SUNWjass> was successful.

Verifying that all packages are fully installed.  OK.
Enabling services: svc:/ldoms/ldmd:default
Solaris Security Toolkit was not applied. Bypassing the use of the
Solaris Security Toolkit is _not_ recommended and should only be
performed when alternative hardening steps are to be taken.
```

You can then verify that the software is installed and that the Logical Domains daemon is running. The command-line interface is accessed by issuing the `ldm` command. At this time, a single domain will be running on the server that owns all of the installed memory and CPUs.

The following example is taken from a small (6-core) T1000. The warning that the system is in configuration mode is displayed after each `ldm` command until the system is rebooted with the new configuration. For brevity, we will omit the subsequent identical warning messages.

```
# pkginfo -x SUNWldomr
SUNWldomr  Solaris Logical Domains (Root)
           (sparc.sun4v) 11.10.0,REV=2006.10.04.00.26
# pkginfo -x SUNWldomu
SUNWldomu  Solaris Logical Domains (Usr)
           (sparc.sun4v) 11.10.0,REV=2006.08.08.12.13

# ldm -V
-----------------------------------------------------------------------
Notice: the LDom Manager is running in configuration mode. Configuration and resource
information is displayed for the configuration under construction; not the current
active configuration. The configuration being constructed will only take effect after it
is downloaded to the system controller and the host is reset.
-----------------------------------------------------------------------
Logical Domain Manager (v 1.2)
        Hypervisor control protocol v 1.3
        Using Hypervisor MD v 1.1
System PROM:
        Hypervisor v. 1.7.3   @(#)Hypervisor 1.7.3 2009/06/08 18:00:15
        OpenBoot   v. 4.30.3  @(#)OBP 4.30.3 2009/06/08 13:27
# ldm list
Name      State   Flags  Cons   VCPU  Memory  Util  Uptime
primary   active -n-c--  SP     24    8064M   0.1%  12m
```

3.4.3 Configuring the Control Domain

Because the control domain initially owns all hardware resources, it must be resized to provide the RAM and CPUs to be allocated to guest domains. Essential services for providing virtual disk, network, and consoles should be defined now as well. When the domain is rebooted, it will have the specified number of CPUs and amount of RAM. The remainder of those resources will be available for allocation to guest domains. The sequence will look similar to this sequence:

```
1     # ldm add-vdiskserver primary-vds0 primary
2     # ldm add-vconscon port-range=5000-5100 primary-vcc0 primary
3     # ldm add-vswitch net-dev=e1000g0 primary-vsw0 primary
```

```
4     # ldm set-mau 1 primary
5     # ldm set-vcpu 4 primary
6     # ldm set-memory 3g primary
7     # ldm add-config initial
8     # ldm list-config
9     factory-default
10    initial [current]
11
12    # shutdown -y  -g0 -i6
```

Line 1 defines a virtual disk server. Notice the naming convention: The name of the server is `primary-vds0`, indicating that the service operates in the domain named `primary` (the control domain) and is the initial virtual disk server (`vds0`). The last token on the line indicates which domain will run this service. While following this convention is not necessary, it is highly recommended, as the name of the service makes its function self-documenting. Note also that this naming convention is separate from Solaris device naming.

Line 2 defines a vcc that will listen for local connections on ports 5000 to 5100. Line 3 defines a virtual Layer 2 switch, `primary-vsw0`, which is associated with the physical NIC device `e1000g0`. Multiple virtual switches can be defined and attached to different NIC devices or to no NIC device at all. Lines 4, 5, and 6 describe the control domain: It has a single cryptographic accelerator, 4 CPUs, and 3 GB of RAM. The remaining lines save this initial configuration in firmware so it will persist after a power cycle and then reboot the server.

3.4.4 Network Connectivity Between Primary and Guest Domains

By default, networking on a virtual switch connecting the control domain and guest domains is disabled. This approach provides an additional layer of security by isolating the control domain from guest domains' network traffic. If this situation is not desired—for example, if the control domain is to be used as a JumpStart server for guest domains—the virtual switch can be configured as a network device and may then be used as the primary interface instead of the physical device to which the switch is assigned.

To configure the virtual switch as a network device, first issue the command `ifconfig -a` to get all the network parameters for the physical device (in the following example, `e1000g0`). Then unplumb the device, and replumb the virtual switch (in this example, `vsw0`) with the same information. When this procedure is complete, guest domains will be able to communicate with the control domain via this network connection.

Important Note!

Before you attempt to reconfigure the virtual switch, you *must* log in from the control domain's console or from a different interface than the interface being unplumbed; otherwise, you will abruptly terminate your session! Terminating the network interface you are using for your login session can be embarrassing.

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232
        index 1 inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 3
        inet 10.6.160.125 netmask fffffc00 broadcast 10.6.163.255
        ether 0:14:4f:2b:65:4e
# ifconfig e1000g0 down          #(See important note!)
# ifconfig e1000g0 unplumb
# ifconfig vsw0 plumb
# ifconfig vsw0 10.6.160.125 netmask 255.255.252.0 up
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
vsw0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 5
        inet 10.6.160.125 netmask fffffc00 broadcast 10.6.163.255
        ether 0:14:4f:fa:85:bb
```

This change does not persist over a control domain reboot, so you must update `/etc/hostname.*` and `/etc/dhcp/*` to make it permanent. In this example, you would rename the configuration files for `e1000g0` to the equivalent files for `vsw0` by issuing the following commands:

```
# mv /etc/hostname.e1000g0 /etc/hostname.vsw0
# mv /etc/dhcp.e1000g0 /etc/dhcp.vsw0
```

3.4.5 Creating a Domain and Installing Oracle Solaris

In the following example, we create a guest domain imaginatively named `ldom1`, with no cryptographic accelerator, a single virtual network device, and a virtual disk residing in a ZFS file system. The following commands define a ZFS file system and allocate within it an empty 10 GB file that will be used for the Solaris system disk:

```
# zfs create rpool/ldoms
# zfs set mountpoint=/ldoms rpool/ldoms
# zfs set compression=on rpool/ldoms
# zfs create rpool/ldoms/ldom1
# mkfile -n 10g /ldoms/ldom1/disk0.img
```

In the preceding lines, ZFS compression is turned on to save disk space. The option to `mkfile` creates an empty file: No disk blocks are allocated until data is written to them. The file takes up no disk space, even though it is apparently 10 GB in size:

```
# ls -l /ldoms/ldom1/disk0.img
-rw------T 1 root root 10737418240 2009-10-12 19:32 /ldoms/ldom1/disk0.img
# zfs list rpool/ldoms/ldom1
NAME                USED  AVAIL  REFER  MOUNTPOINT
rpool/ldoms/ldom1    19K  21.4G    19K  /ldoms/ldom1
```

The virtual disk can also be allocated from other back-ends, such as a physical disk, but ZFS provides operational flexibility.

The following commands create the domain. Lines 2 and 3 set the number of CPUs and cryptographic accelerators. Line 4 sets the amount of RAM assigned to the domain. Line 5 creates the virtual network device using the virtual Layer 2 switch defined previously. Additional network devices could be defined if we wanted the domain to reside on separate networks. Line 6 exports the empty disk image as a virtual volume `vol10@primary-vds0` from the virtual disk service. Line 7 imports this volume as a virtual disk `vdisk10` into the guest domain. The commands for adding a virtual disk are a bit more complicated than the others; they can be interpreted as first defining a resource exported by the virtual disk server and then importing that resource into the domain that uses it. Finally, lines 8 and 9 do the same for a file containing the ISO format image of an Oracle Solaris installation DVD.

```
1   # ldm add-domain ldom1
2   # ldm set-vcpu 4 ldom1
3   # ldm set-mau 0 ldom1
4   # ldm set-mem 4g ldom1
5   # ldm add-vnet vnet1 primary-vsw0 ldom1
6   # ldm add-vdsdev /ldoms/ldom1/disk0.img vol10@primary-vds0
7   # ldm add-vdisk vdisk10 vol10@primary-vds0 ldom1
8   # ldm add-vdsdev /DVD/S10u7/solarisdvd.iso s10u7iso@primary-vds0
9   # ldm add-vdisk vdisk_iso s10u7iso@primary-vds0 ldom1
```

At this point, the domain's definition is complete. We set OpenBoot Prom (OBP) variables to force the domain to come to the `ok` prompt instead of booting an OS by setting the `autoboot?` property so as to demonstrate OBP commands. The "\" in the command line is an escape character, so we can enter the "?" character as a literal value. Then we bind the domain, which assigns the specified resources to the domain. This includes assigning the port used by the virtual console concentrator—`5000` in the example. Finally, we start the domain, which has an effect similar to performing a "power on" operation for a real server: It loads the OBP, which then displays its `ok` prompt.

```
# ldm set-variable autoboot\?=false ldom1
# ldm bind ldom1
# ldm list   #(NOTE: this was done after bind, before start)
NAME         STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary      active     -n-cv-  SP      4     3G       1.2%  38m
ldom1        bound      ------  5000    4     1G
# ldm start ldom1
```

It's helpful to bring up a second terminal window to watch this process. The `telnet` command can be issued after the `ldm bind` command to do so. At first, no output follows the output coming from the `telnet` command itself (the line beginning with "`Press ~?`"). When `ldm start ldom1` is issued, OpenBoot is loaded and outputs `{0} ok`.

```
# telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost....
Escape character is '^}'.
Connecting to console "ldom1" in group "ldom1" ....
Press ~? for control options ..

{0} ok
```

3.4.6 Viewing a Domain

From the `ok` prompt, we can issue OpenBoot commands and view the domain's devices:

```
Sun Fire(TM) T1000, No Keyboard
Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
OpenBoot 4.30.3, 1024 MB memory available, Serial #xxxxxxxx.
Ethernet address x:xx:xx:xx:xx:xx, Host ID: xxxxxxxx
{0} ok banner
Sun Fire(TM) T1000, No Keyboard
Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
OpenBoot 4.30.3, 1024 MB memory available, Serial #xxxxxxxx.
Ethernet address x:xx:xx:xx:xx:xx, Host ID: xxxxxxxx.
{0} ok show-disks
```

```
a) /virtual-devices@100/channel-devices@200/disk@1
b) /virtual-devices@100/channel-devices@200/disk@0
Enter Selection, q to quit:q
{0} ok show-nets
b) /virtual-devices@100/channel-devices@200/network@0
q) NO SELECTION
Enter Selection, q to quit:q
{0} ok devalias
vdisk_iso       /virtual-devices@100/channel-devices@200/disk@1
vdisk10         /virtual-devices@100/channel-devices@200/disk@0
vnet10          /virtual-devices@100/channel-devices@200/network@0
net             /virtual-devices@100/channel-devices@200/network@0
disk            /virtual-devices@100/channel-devices@200/disk@0
virtual-console /virtual-devices/console@1
name            aliases
```

This output shows that the domain is a virtual machine with its own devices—notice how the device aliases are derived from the `ldm` commands that defined them.

3.4.7 Installing Oracle Solaris into a Domain

Even the thrill of issuing OpenBoot commands at the `ok` prompt can pall after a while, so we will install Solaris 10 in the domain. Installation can be done over the network if a JumpStart or Ops Center infrastructure is available, and works just as on physical machines. For simplicity, we illustrate booting from a Solaris installation DVD image. Except for the device name alias in the boot command, this process is identical to that used when installing from a DVD on a physical machine. Notice the name of the discovered network interface: `vnet0`.

```
{0} ok boot vdisk_iso
Boot device: /virtual-devices@100/channel-devices@200/disk@1:f  File and args:
SunOS Release 5.10 Version Generic_139555-08 64-bit
Copyright 1983-2009 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Using RPC Bootparams for network configuration information.
Attempting to configure interface vnet0...
Configured interface vnet0
Reading ZFS config: done.
Setting up Java. Please wait...
Serial console, reverting to text install
Beginning system identification...
Searching for configuration file(s)...
Search complete.
Discovering additional network configuration...
```

The rest of the Solaris installation is as usual, and is not shown here.

The disk space occupied by virtual disks may be smaller than its apparent size. As demonstrated by the following lines, a 10 GB virtual disk containing a newly installed copy of Oracle Solaris takes up only a little more than 2.5 GB of disk space.

```
# ls -l /ldoms/ldom1/
total 5292263
-rw------T   1 root     root     10737418240 Aug 14 16:45 disk0.img
# zfs list rpool/ldoms/ldom1
NAME                USED  AVAIL  REFER  MOUNTPOINT
rpool/ldoms/ldom1  2.52G  61.6G  2.52G  /ldoms/ldom1
```

The disk space consumed will depend on how much of the virtual disk has been populated by the guest operating system and how well the data is compressed if ZFS compression is being used.

3.4.8 Observing Guest Domains from the Control Domain

From the control domain, you can use the `ldm` command to observe the activity of the domain, and to see which CPU, memory, network disk, cryptographic accelerator, and console resources are bound to it. The following example shows the short and long forms of `ldm list`. The control domain's console is `SP`, indicating that its console is accessed via the service processor, which is typical for Solaris on a SPARC server. The line for the guest domain shows the virtual console service port number used to access the domain's console.

In this example, you can see the number of virtual CPUs and the physical CPUs to which they are bound. For instance, `ldom1`'s virtual CPU 0 is on physical CPU 4. The long listing format shows the utilization of each virtual CPU.

```
# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     3G       1.2%  1h 3m
ldom1            active     -n----  5000    4     1G        29%  22m
# ldm list -l ldom1
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldom1            active     -n----  5000    4     1G        17%  22m

SOFTSTATE
Solaris running

MAC
    00:14:4f:f9:fe:c8
```

```
HOSTID
    0x84f9fec8

CONTROL
    failure-policy=ignore

DEPENDENCY
    master=

VCPU
    VID    PID    UTIL STRAND
    0      4       38%   100%
    1      5       51%   100%
    2      6       13%   100%
    3      7      2.6%   100%

MEMORY
    RA                PA                SIZE
    0x8000000         0xc8000000        1G

VARIABLES
    auto-boot?=false

NETWORK
    NAME              SERVICE                     DEVICE     MAC                 MODE
PVID VID                   MTU
    vnet10            primary-vsw0@primary        network@0
00:14:4f:fb:f8:a4        1                          1500

DISK
    NAME              VOLUME                      TOUT DEVICE  SERVER
MPGROUP
    vdisk10           vol10@primary-vds0               disk@0  primary
    vdisk_iso         s10u7iso@primary-vds0            disk@1  primary

VCONS
    NAME              SERVICE                     PORT
    ldom1             primary-vcc0@primary        5000
```

The `STRAND` column indicates the percentage of the physical CPU strand owned by that domain. In the current implementation, that value is always 100% because threads are dedicated to domains. The field is provided so scripts that process command output will continue to work if partial thread allocation is added later. The `ldm list -p` command option can be used to produce parsable output for easier script writing. In the following listing, `ldom2` is a four-vCPU domain running at 25% average CPU utilization. The parsable format output, with or without the `awk` command, makes it clear that one CPU is fully used while the others are idle.

```
# ldm list ldom2
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldom2            active     -n----  5001    4     1G        25%  1h 27m
# ldm list -p -l ldom2|grep strand
|vid=0|pid=12|util=0.4%|strand=100
|vid=1|pid=13|util=0.0%|strand=100
|vid=2|pid=14|util=0.0%|strand=100
|vid=3|pid=15|util=100%|strand=100
# ldm list -p -l ldom2|grep strand\
     |awk -F'|' '{print $4}'|sed 's/util=//; s/%//'
0.4
0.0
0.1
100
```

3.4.9 Viewing a Domain from the Inside

Once Oracle Solaris is installed in a domain, you can use normal commands to view its configuration. Notice that the domain has four CPUs and a network interface named `vnet0` with its own MAC address. A pleasant side effect of running in a domain is that the boot process is very fast, because there is no need to perform a power-on self-test (POST) or probe physical devices.

```
{0} ok boot disk
Boot device: /virtual-devices@100/channel-devices@200/disk@0  File and args:
SunOS Release 5.10 Version Generic_139555-08 64-bit
Copyright 1983-2009 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Hostname: t1ldom1
Reading ZFS config: done.
Mounting ZFS filesystems: (5/5)

t1ldom1 console login: root
Password:
Aug 14 16:42:06 t1ldom1 login: ROOT LOGIN /dev/console
Last login: Fri Aug 14 16:35:45 from xxx.xxx.xx.xxx
Sun Microsystems Inc.   SunOS 5.10      Generic January 2005
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
vnet0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 192.168.2.101 netmask ffffff00 broadcast 192.168.2.255
        ether 0:14:4f:fb:f8:a4
```

```
# hostid
84f9fec8
# psrinfo
0       on-line   since 08/14/2009 16:40:26
1       on-line   since 08/14/2009 16:40:28
2       on-line   since 08/14/2009 16:40:28
3       on-line   since 08/14/2009 16:40:28
```

At this point, the administrator can install application software and start using the domain.

3.4.10 Dynamic Reconfiguration

The Logical Domains technology lets the system administrator change the CPU and virtual I/O resources made available to guest domains. For example, the domain in the preceding example can be changed to use eight CPUs by issuing either of these commands: `ldm add-vcpu 4 ldom2` or `ldm set-vcpu 8 ldom2`. Each command takes effect immediately without rebooting the guest, and the new CPU resources can be used immediately. The `psrinfo` command in the guest domain shows the new CPU configuration, including the date and time shown for the added CPUs.

```
# psrinfo
0       on-line   since 10/25/2009 11:17:44
1       on-line   since 10/25/2009 11:17:45
2       on-line   since 10/25/2009 11:17:45
3       on-line   since 10/25/2009 11:17:45
4       on-line   since 10/28/2009 09:21:36
5       on-line   since 10/28/2009 09:21:36
6       on-line   since 10/28/2009 09:21:36
7       on-line   since 10/28/2009 09:21:36
```

Virtual I/O devices can also be added via the `ldm add-vnet` and `ldm add-vdisk` commands, and both I/O and CPU resources can be removed using complementary `ldm rm-vnet` and `ldm rm-vdisk` commands. Similarly, cryptographic accelerators can be dynamically added or removed. These commands can easily be scripted to ensure their execution at particular times or as part of a resource manager. At this writing, changing the amount of RAM in a domain requires that the domain be rebooted before the Solaris instance will recognize the change.

3.4.11 Dynamic Resource Management

The Logical Domains technology provides a policy-based resource manager that automatically adds or removes CPUs from a running domain based on its utilization and relative priority. Policies can be prioritized to ensure that important domains obtain preferential access to resources. They can also be enabled or disabled manually or based on time of day for different prime shift and off-hours policies. For example, one domain may have the highest resource needs and priority during the daytime, while a domain running batch work may operate in a more resource-intensive manner at night.

Policy rules specify the number of CPUs that a domain has, bounded by minimum and maximum values, and based on their utilization:

- The number of CPUs is adjusted between `vcpu-min` and `vcpu-max` based on `util-upper` and `util-lower` CPU busy percentages. (All of these variables are property values associated with the policy.)
- If CPU utilization exceeds the value of `util-upper`, virtual CPUs are added to the domain until the number is between `vcpu-min` and `vcpu-max`.
- If the utilization drops below `util-lower`, virtual CPUs are removed from the domain until the number is between `vcpu-min` and `vcpu-max`.
- If `vcpu-min` is reached, no more virtual CPUs can be dynamically removed. If `vcpu-max` is reached, no more virtual CPUs can be dynamically added. Manual changes to the number of CPUs can still be made using the `ldm` commands shown previously.
- Multiple policies can be in effect, and are optionally controlled by `tod-begin` and `tod-end` (time of day) values.

The resource manager includes ramp-up (attack) and ramp-down (decay) controls to adjust the system's response to workload changes, specifying the number of CPUs to add or remove based on changes in utilization, and how quickly the resource manager responds. Resource management is disabled in elastic power management mode, in which unused CPUs are powered down to reduce power consumption. The following is an example of a command creating a policy:

```
# ldm add-policy tod-begin=09:00 tod-end=18:00 util-lower=25 \
      util-upper=75 vcpu-min=2 vcpu-max=16 attack=1 \
      decay=1 priority=1 name=high-usage ldom1
```

This policy controls the number of CPUs for domain `ldom1`, is named `high-usage`, and is in effect between 9 A.M. and 6 P.M. The lower and upper CPU utilization settings are 25% and 75% CPU busy, respectively. The number of CPUs is adjusted between 2 and 16: One CPU is added or removed at a time (the attack and decay values). For example, if the CPU utilization exceeds 75%, a CPU is added unless `ldom1` already has 16 CPUs.

The resource manager provides flexible and powerful policy-driven dynamic CPU resource management for Logical Domains that can automatically adjust CPU assignments based on CPU resource requirements and domain priorities.

3.4.12 Cloning a Domain

The Logical Domains technology makes it easy to clone systems, especially when using virtual disks residing in ZFS. A golden image instance of Oracle Solaris can be installed, patched, and customized, and then used as a master copy for multiple domains.

ZFS makes this process efficient by letting the administrator take a snapshot of a virtual disk and then create clones from it. Snapshots are read-only images of data in a ZFS file system at the time the `zfs snapshot` command is executed, while clones are read/write images based on the snapshot. ZFS saves disk space because only changed data takes up additional space on disk. For example, if a snapshot is made of a ZFS file system with 100 GB of data and 5 MB of data is changed, only 5 MB of additional disk space is consumed by the snapshot. Both a snapshot and the file system it is based on use common disk locations for data that is common to both. The same is true with clones: Space is consumed only for changed disk contents. At first, the disk footprint of the new domain will be negligible, as shown in the following example, but eventually it may increase if the contents of the Solaris instance diverge from the master image.

Virtual disks based in a different file system, such as UFS, can be easily replicated by using the low-tech `cp` command. With this approach, however, each copy of a disk will require the same disk space as the original. Given that ZFS is much more space efficient, it is the recommended technique for cloning a domain.

Before cloning a domain, be sure to shut it down to ensure that all buffers have been written and that its disk contents are stable. You may also wish to unbind the domain if its purpose is to be a template for other domains.

Then, assuming use of the same domain as in the previous example, we could issue the following commands:

```
# zfs snapshot tank/ldoms/ldom1@initial
# zfs list
NAME                            USED  AVAIL  REFER  MOUNTPOINT
…
rpool/ldoms/ldom1              2.52G  61.6G  2.52G  /ldoms/ldom1
rpool/ldoms/ldom1@initial          0      -  2.52G  -
```

continues

```
# zfs clone rpool/ldoms/ldom1@initial  rpool/ldoms/ldom2
# zfs list
NAME                            USED  AVAIL  REFER  MOUNTPOINT
...
rpool/ldoms/ldom1              2.52G  61.6G  2.52G  /ldoms/ldom1
rpool/ldoms/ldom2                  0  61.6G  2.52G  /ldoms/ldom2
rpool/ldoms/ldom1@initial          0      -  2.52G  -
# ls -l /ldoms/ldom2
-rw------T   1 root     root     10737418240 Jan  2 23:19 disk_ldom.img
{Issue ldm commands to define the domain.}
{The OS is already installed; just boot it.}
{When it has finished booting: }
# zfs list
...
rpool/ldoms/ldom2              11.2M  60.6G  2.25G  /ldoms/ldom2
```

Once you've issued the `ldm` commands needed to define domain `ldom2`, you can simply bind and boot the guest domain—Oracle Solaris is already installed on the virtual disk.

A cloned Solaris instance has the same IP address and host name as the system it was cloned from. You can choose from several methods to give it a unique identity. One strategy is to configure the guest to use DHCP at boot. Another technique is to issue the `/usr/sbin/sys-unconfig` command before shutting down and cloning the golden image. The first boot of the cloned image will then prompt (at the guest console) for system identification data, such as the time zone, IP address, and so on. Alternatively, you can configure the golden image domain with a unique host name and IP address that will not be given to any other domain. After booting a domain cloned from it, simply log into the clone via SSH, and change its host name and IP address via standard Solaris administrator commands. An advantage of this method is that it avoids the use of the guest console.

3.5 Domain Mobility

Domains can be installed on one CMT server (the source host) and migrated to a different, compatible CMT server (the target host) for planned workload migration. Doing so can free up memory and CPUs on a server, or vacate it for planned maintenance. At the time of this writing, Logical Domains supported two forms of domain migration, both invoked by the `ldm migrate` command. The following command migrates a domain to the host at the specified address. The command will prompt for the root password of the control domain on the target system.

```
# ldm migrate ldg1 root@192.168.1.12
```

The `ldm migrate` command can be issued with the option `-n` to request a dry run that tests whether the migration is possible but does not actually migrate the domain.

In *cold* migration, the domain must be stopped. It must also be in the *bound* or *inactive* state on the source machine. Cold migration consists of verifying access to the domain's I/O resources. In particular, the virtual disks must be accessible to both the source and the target. This step also ensures there are sufficient resources to bind the domain after migration, and then transmits the domain's description to the target machine. Once the domain is migrated, the domain on the source machine is unbound (if it is currently bound to resources) and removed from the source domain configuration. Cold migration applies only to domains that are not running, but can be very helpful for planned migration of workloads that can tolerate an outage with a shutdown and reboot. It is very fast, as only descriptive information is transmitted between servers.

In *warm* migration, a running domain is suspended on the source machine and resumed on the target machine without a reboot. Domain execution is suspended while it is being migrated. This process may take a number of seconds or a few minutes, depending on domain memory size and network speed, and on whether the control domain has access to a cryptographic accelerator.

First, the domain managers on both source and target machines verify that sufficient capacity to bind the domain on the target is available. Then, the target system creates and binds a single-CPU version of the guest domain, and the source machine removes all but one CPU from the running domain. Next, the source machine suspends the domain and removes its last CPU. The domain's memory contents and state are then compressed, encrypted (using the cryptographic accelerator if one is associated with the control domain), and transmitted from the source to the target.

Processing is multithreaded and takes advantage of the CPU threads in the control domain; it also exploits the cryptographic accelerator to ensure better performance. Domain memory contents are always encrypted before transmission, as it would be a significant security exposure to transmit a domain's memory contents (which might include passwords or private information) as cleartext. The domain is then restored on the target system and its remaining CPUs added. The Oracle Solaris instance continues execution on the target machine with the same disk storage, IP addresses, and MAC addresses as before, and applications resume processing. Finally, the domain is unbound and removed from the source host's domain configuration.

Warm migration is suitable for planned migration of any workload that can be paused for a number of seconds.

A *live* migration capability is anticipated, in which a migrating domain will be unresponsive for only a very small interval.

3.6 Physical to Virtual Conversion

Logical Domains provides a Physical to Virtual (P2V) tool to automate conversion of physical Oracle Solaris systems into guest Logical Domains. This tool moves file system contents from a physical server to a Logical Domain on the CMT server, and if necessary, replaces packages geared toward the `sun4u` architecture platform (all UltraSPARC and SPARC64 systems) with packages for the `sun4v` architecture provided by the CMT platform. The physical system can be running Solaris 8 or later on a `sun4u` system, or Solaris 10 running outside of a domain.

P2V migration consists of three phases, which are carried out under the control of the `ldmp2v` command:

1. *Collection*: Runs on the physical machine and collects a file system image and configuration data using `ufsdump` or `flarcreate`. The resulting file can be transmitted to the target system's control domain or stored on an NFS server that is available to both the physical system and the control domain.
2. *Preparation*: Runs on the control domain of the target platform. It creates a guest domain, and restores the contents of the collected file system into virtual disks.
3. *Conversion*: Runs on the control domain of the target platform. It upgrades the guest domain to prepare it to run as a domain. This process removes packages, replacing `sun4u` packages with their corresponding `sun4v` versions.

If the physical system is running Solaris 8 or Solaris 9, the P2V process installs the system image in a Solaris Container using a Solaris 8 Container or Solaris 9 Container under the guest domain's Solaris 10 kernel. That practice lets the migrated image appear to be the same Solaris version as on the physical machine. The P2V process can optionally preserve the physical system's network identity by reusing its MAC address.

The P2V process requires Solaris 10 system images to be available in either Solaris installation DVD or network install format, with the package `SUNWldmp2v` being installed in the control domain. The file `/etc/ldmp2v.conf` must be populated with variables indicating the names of the default virtual switch and virtual disk servers, and the type of disk back-ends to use for virtual disks. The P2V command must be made available to the physical system, either by NFS mount or by copying the command to a local disk.

Once these setup tasks are complete, a set of commands can be executed like those shown in the following example. In this case, `sparcules` indicates the login session on the physical server, and `primary` indicates the control domain login session. The collect phase writes a collected image to an NFS-mounted directory that is also accessed by the control domain. The preparation phase creates a domain and imports the collected file system image into virtual disks. The conversion phase boots the guest domain with an Oracle Solaris installation image to upgrade the system image to a software level that is compatible with the Logical Domains technology.

```
sparcules# ldmp2v collect -x /export/home -d /home/p2v
Collecting system configuration …
Archiving file systems ...
  DUMP: Date of this level 0 dump: Sun Oct 25 11:46:32 2009
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/rdsk/c0d0s0 (sparcules:/) to /home/p2v/ufsdump.0.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Writing 63 Kilobyte records
  DUMP: Estimated 5576806 blocks (2723.05MB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
  DUMP: 64.06% done, finished in 0:05
  DUMP: 5576758 blocks (2723.03MB) on 1 volume at 3064 KB/sec
  DUMP: DUMP IS DONE
sparcules#
{...now we move to the control domain...}

primary# ldmp2v prepare -d /home/p2v -o keep-mac sparcules
Creating vdisks ...
Creating file systems ...
Populating file systems ...
Modifying guest domain OS image ...
Removing SVM configuration ...
Unmounting guest file systems ...
Creating domain sparcules ...
Attaching vdisks to domain sparcules
primary # ldmp2v convert -i /DVD/S10u7/solarisdvd.iso \
     -d /home/p2v sparcules
Testing original system status ...
LDom sparcules started
Waiting for Solaris to come up ...
Select 'Upgrade' (F2) when prompted for the installation type.
Disconnect from the console after the Upgrade has finished.
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.
Connecting to console "sparcules" in group "sparcules" ....
Press ~? for control options ..
Configuring devices.
Using RPC Bootparams for network configuration information.
Attempting to configure interface vnet0...
{… many lines of a typical Solaris upgrade dialogue omitted.}
```

Once this process is complete, the guest domain has the same system identity and applications as it did on the original physical server. The amount of time to complete the process depends on the size of the file systems being copied and the network bandwidth available for its transmission to the control domain. The amount of administrator effort is dramatically reduced, with only a few commands needed to carry out a complete system migration.

3.7 Ease-of-Use Enhancements

The preceding examples manage Logical Domains with a command-line interface. This strategy is traditional and scriptable—but also requires prerequisite skills, can be error-prone, and is intimidating for the occasional user. It also does not scale well when many machines are involved.

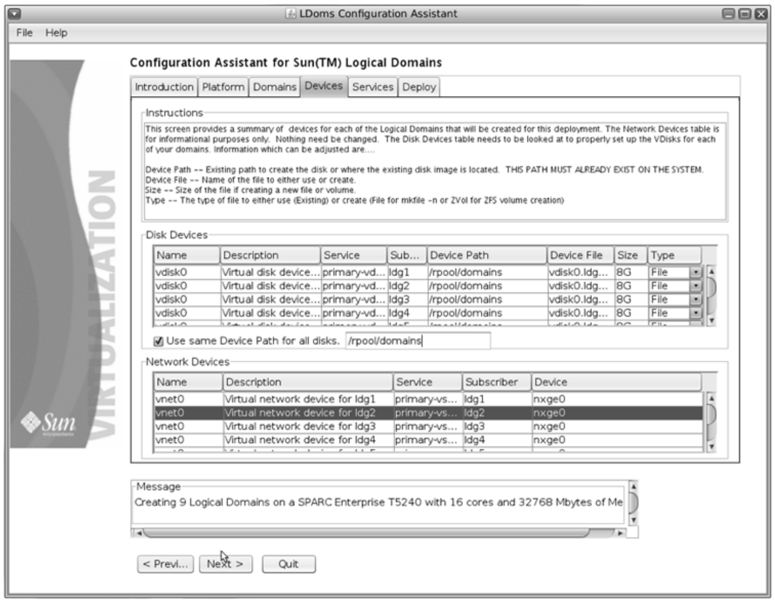
These limitations can be addressed by using scripts and by making appropriate use of standards and automation in an enterprise. They are also addressed by the Logical Domains Configuration Assistant, which provides a graphical user interface (GUI) for the initial installation and configuration of Logical Domains software and guest domains on a server. The Configuration Assistant is a Java application that is provided with the Logical Domains install software. To start it, issue the following command:

```
$ java -jar /path/Configurator.jar
```

This command launches a GUI that steps the user through a set of panels, such as the one shown in Figure 3.4.

The GUI steps through a series of panels that let the user specify the number of domains to be created and the paths to their virtual disks. The last step lets the user save a script to run at a later time on one or more servers or, if the appropriate host name and password are provided, to run on a target CMT server. The Logical Domains package also includes the `ldmconfig` command, a text-mode tool that prompts the administrator for information about the domains and then generates the commands to build those domains according to best-practice naming conventions and a template of where the virtual disk back-ends are stored.

A more comprehensive solution is based on the Oracle Enterprise Manager Ops Center licensed product, described at `http://www.sun.com/software/products/opscenter/`. This product provides full life-cycle support for provisioning, and managing systems in a data center from a single user interface and screen, for both real and virtual systems. Ops Center can create, delete, configure, boot, and shut down Logical Domains. It also provides monitoring and management, showing utilization charts for CPU, memory, and file systems. In addition, the Ops Center manages resource pools, permitting dynamic reallocation of resources based on policies, and provides a graphical interface for controlling domain migration. Figure 3.5 shows several physical nodes that are visible in the assets tab of the Ops Center, with details illustrating resources used by several domains.



Configuration Assistant for Sun(TM) Logical Domains

Instructions

This screen provides a summary of devices for each of the Logical Domains that will be created for this deployment. The Network Devices table is for informational purposes only. Nothing need be changed. The Disk Devices table needs to be looked at to properly set up the VDisks for each of your domains. Information which can be adjusted are....

Device Path -- Existing path to create the disk or where the existing disk image is located. THIS PATH MUST ALREADY EXIST ON THE SYSTEM.
Device File -- Name of the file to either use or create.
Size -- Size of the file if creating a new file or volume.
Type -- The type of file to either use (Existing) or create (File for mkfile -n or ZVol for ZFS volume creation)

Disk Devices

Name	Description	Service	Sub...	Device Path	Device File	Size	Type
vdisk0	Virtual disk device...	primary-vd...	ldg1	/rpool/domains	vdisk0.ldg...	8G	File
vdisk0	Virtual disk device...	primary-vd...	ldg2	/rpool/domains	vdisk0.ldg...	8G	File
vdisk0	Virtual disk device...	primary-vd...	ldg3	/rpool/domains	vdisk0.ldg...	8G	File
vdisk0	Virtual disk device...	primary-vd...	ldg4	/rpool/domains	vdisk0.ldg...	8G	File

☑ Use same Device Path for all disks. /rpool/domains

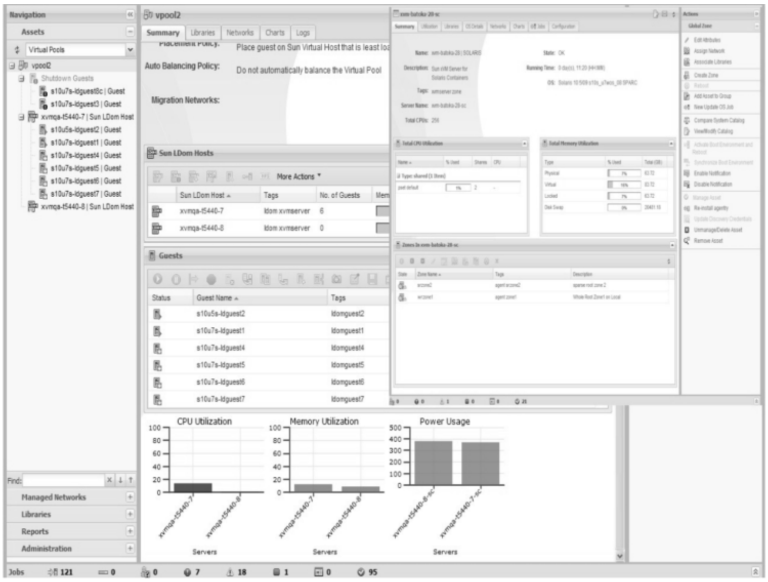
Network Devices

Name	Description	Service	Subscriber	Device
vnet0	Virtual network device for ldg1	primary-vs...	ldg1	nxge0
vnet0	Virtual network device for ldg2	primary-vs...	ldg2	nxge0
vnet0	Virtual network device for ldg3	primary-vs...	ldg3	nxge0
vnet0	Virtual network device for ldg4	primary-vs...	ldg4	nxge0

Message

Creating 9 Logical Domains on a SPARC Enterprise T5240 with 16 cores and 32768 Mbytes of Me

Figure 3.4 Logical Domains Configuration Assistant



vpool2

Placement Policy: Place guest on Sun Virtual Host that is least loa

Auto Balancing Policy: Do not automatically balance the Virtual Pool

Migration Networks:

Sun LDom Hosts

Sun LDom Host	Tags	No. of Guests
xvmqa-t5440-7	ldom xvmserver	6
xvmqa-t5440-8	ldom xvmserver	0

Guests

Status	Guest Name	Tags
	s10u5s-ldguest2	ldomguest2
	s10u7s-ldguest1	ldomguest1
	s10u7s-ldguest4	ldomguest4
	s10u7s-ldguest5	ldomguest5
	s10u7s-ldguest6	ldomguest6
	s10u7s-ldguest7	ldomguest7

Figure 3.5 Oracle Enterprise Manager Ops Center Display for Logical Domains

3.8 Comparison with Oracle Solaris Containers

Oracle Solaris Containers and Logical Domains provide alternative and complementary methods to virtualize Solaris systems, so it is natural to compare them.

Solaris Containers provide ultra-lightweight OS virtualization with excellent isolation, security, observability, fast deployment, resource granularity, and native performance. However, because many Containers run on a single Solaris instance, they do not permit multiple kernel patch levels to be used. Solaris Containers also have a few restrictions, such as the inability to run an NFS server.

In contrast, Logical Domains are more like "business as usual" for system administrators and application planners. They are individually installed, patched, used, and managed, much like physical servers. Oracle Solaris can be installed in a domain via a JumpStart or by booting from a DVD device, just like on a physical server. Like Containers, domains can also be cloned from previously installed and customized instances. However, unlike Containers, a running domain can be migrated from one server to another even without rebooting the domain.

Logical Domains are available only on CMT servers, whereas Containers are available on any platform that supports Solaris. Also, because domains host full OS instances on dedicated CPUs and RAM, they have a larger resource footprint than Containers. Many more virtualized instances can be hosted on the same platform by using Containers than by using domains.

Solaris Containers and Logical Domains are complementary virtualization technologies. They can be combined without adding overhead by running Containers within domains to achieve the highest degree of flexible virtualization. Separate OS instances can be configured when different OS kernel levels are required, with each OS instance hosting many lightweight, virtualized Container environments.

3.9 Summary

Logical Domains provide low-cost, efficient virtualization on UltraSPARC T1, T2, and T2 Plus processors. Each domain is a separate, independent virtual machine with its own OS instance. Hardware resources are dedicated to each domain through a highly granular and dynamic resource allocation scheme.

Compared to other virtualization technologies, Logical Domains provide an extremely efficient hypervisor implementation that avoids the overhead inherent in traditional hypervisors, and avoids the license costs of commercial virtualization products.

Logical Domains is an ideal technology for migrating from older platforms to current hardware in a consolidated, energy-efficient platform. Oracle Solaris systems can be easily moved to a domain using the Logical Domains P2V tool, thereby leveraging Solaris and SPARC binary compatibility. Workloads from other operating systems and platforms can also be migrated to provide an efficient virtualized solution that reduces both energy costs and server sprawl.