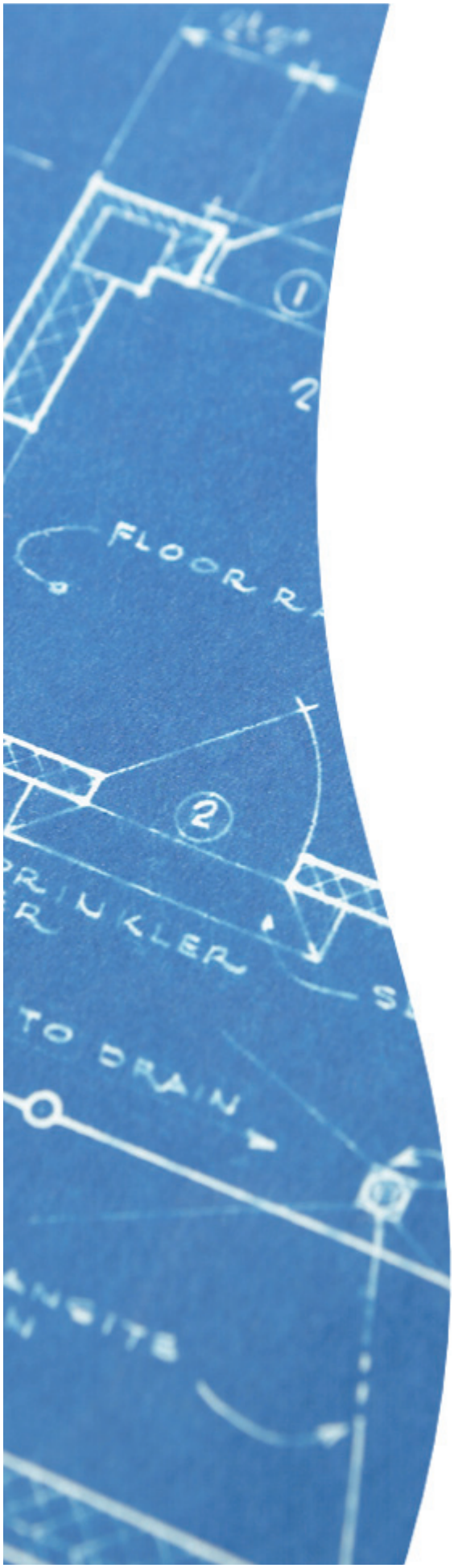

UNDERSTANDING THE SUN™ xVM HYPERVISOR ARCHITECTURE

Michael Haines
David Edmonson
Sun Microsystems

Sun BluePrints™ Online

Part No 820-3089-10
Revision 1.0, 5/20/08

Table of Contents

Chapter 1. Introduction 1

Chapter 2. Creating Efficient Datacenters 2
Virtualization is the Key 2
Types of Virtualization 2
Dynamic System Domains 3
Operating System Virtualization 3
Resource Management 4
Virtual Machines 4
Sun xVM Hypervisor Architecture Overview 5
Virtual CPUs 6
Memory 6
Running the Solaris OS in the Control Domain 7
Running Multiple Operating Systems 7

Chapter 3. Basic Control Domain Verification 9
Preparing for Guest Domain Installation 9
Checking the Sun xVM Hypervisor Packages and the Control Domain 9
Creating the Right domU Management Environment 11
Using Solaris ZFS 11
Setting Up Files 12
Basic Installation Using virt-install 13
Installation via the Command Line 15
Installation with Solaris JumpStart™ 15
Configuring Domains to Automatically Start at Boot Time 15

Chapter 4. Virtual Machine Management 17

Chapter 5. Advanced Installation and Configuration 18
Creating the Configuration File 18
Directives Used in the Solaris x86 xVM Domain Configuration File 18
Example Configuration File 18
Installing the First Guest Domain 19
Creating a Guest Domain Using the Command Line 36
Creating a Guest Domain Using Interactive Mode 37
Accessing Virtual Block Devices and Block Storage 37
Dynamically Allocating and Deallocating Virtual CPUs 38
Sun xVM Hypervisor Networking Infrastructure 41
Inter-Domain Networking 42
General Facilities 42
Guest Domains—domU 42
The Control Domain—dom0 43
The Back-end GLDv3 Driver—xnbu 43

Layering Over a GLDv3 Driver—xnbo 43
Integration into the Sun xVM Hypervisor Toolset 44

Chapter 6. Migration of Virtual Machine Instances 46
Migration 46
Live Migration 47
Benefits of Live Migration 47
The Live Migration Process 47
Observing the Live Migration Process 48

Chapter 7. Troubleshooting 49
Differences from a Standard Environment 49
Understanding Resource Virtualization 49
Crash Dumps 50
Observability 50
Dynamic Tracing 51

Chapter 8. Hardware-Assisted Virtualization 53
Installing Unmodified Operating Systems 53

Chapter 9. For More Information 56
About the Author 56
Acknowledgments 56
References 56
Ordering Sun Documents 57
Accessing Sun Documentation Online 57

Appendix A. Glossary 58

Chapter 1
Introduction

Over the course of the last decade, the nature of computing changed in fundamental ways. The explosive growth of corporate intranets and the Internet — including the need for more bandwidth, larger networks, and innovative digital devices — created new and challenging demands. As the number of users and devices accessing services over the network grows, organizations are being forced to rethink how they create, manage, extend, and ultimately deliver information technology (IT) services with greater functionality and reduced cost. At the same time, these changes are creating massive opportunities for innovations in service and functionality.

Note – The content in this document reflects the Solaris™ Express Developer Edition 1/08 for x86 and x64 platform release. This document may be updated periodically to reflect modifications and additional functionality provided in subsequent software releases.

Today, organizations are focused on obtaining additional availability or scalability in the most efficient way possible, with users taking for granted that IT services will deliver the performance and predictability needed. However, as systems are replicated throughout IT infrastructures for greater resiliency and throughput, the result is often a sprawling, complex network of systems that are costly and difficult to manage. As a result, many enterprises are turning to consolidation and virtualization strategies in order to maximize resource utilization and simplify datacenter complexity.

This Sun BluePrints™ article discusses the Sun™ xVM hypervisor architecture, a new approach to virtualization for x86 and x64 systems that makes it possible to run multiple disparate operating systems and applications on a single server.

- Chapter 2, “Creating Efficient Datacenters”, discusses approaches to virtualization and introduces the Sun xVM hypervisor software architecture.
- Chapter 3, “Basic Control Domain Verification”, describes how to create paravirtualized domains on x86 and x64 systems using Sun xVM hypervisor software.
- Chapter 4, “Virtual Machine Management”, describes the command line and graphical tools available to manage guest domains.
- Chapter 5, “Advanced Installation and Configuration”, takes a detailed look at the command line methods and advanced options available for configuring domains.
- Chapter 6, “Migration of Virtual Machine Instances”, explains the live migration process.
- Chapter 7, “Troubleshooting”, provides an overview of topics and techniques that may be useful when diagnosing domain related problems.
- Chapter 8, “Hardware-Assisted Virtualization”, explains how to install unmodified guest operating systems in hardware virtual machines.
- Chapter 9, “For More Information”, provides links to references.
- Appendix A, “Glossary”, defines key terms used throughout this document.

Chapter 2
Creating Efficient Datacenters

Enterprises continually adjust business plans, priorities, and work to create new opportunities, and often deploy increasing numbers of services to satisfy demand. With business success at stake, many organizations avoid possible system security, availability, and performance conflicts by deploying only one application per server. Corresponding servers are then installed to support development, test, and disaster recovery requirements.

Such common datacenter management practices quickly create a sprawling complex of compute resources that are difficult to manage and leave systems largely under utilized. In addition, IT managers are startled to find datacenters quickly reach physical and budgetary constraints in the areas of power, cooling, and real estate. The resulting inefficient IT infrastructure carries cost burdens that can limit business growth, leading organizations to seek more effective hosting strategies.

Virtualization is the Key

Many IT organizations launch server consolidation initiatives in an effort to raise datacenter efficiency levels. While consolidation gives enterprises an opportunity to better utilize resources, simply deploying multiple applications on a single server often results in poor performance and even application failure. Tuning and maintenance requirements may not easily align across a number of software programs, poorly behaved applications can starve other co-located software services of resources, and sensitive data may not be fully protected from unwanted access.

Virtualization technologies enhance consolidation strategies by enabling organizations to create administrative and resource boundaries between applications within a system. The ability to isolate software programs running within a consolidated server helps IT organizations deliver on application performance and security requirements, as well as meet custom tuning needs. Combining workloads and using virtualization techniques can help enterprises maximize the use of compute platforms, simplify an IT infrastructure, and bring new levels of efficiency, manageability, and agility to a growing enterprise.

Types of Virtualization

A number of technologies enable platform virtualization — resource management, hard partitioning, operating system virtualization, and virtual machine technology — each providing varying degrees of flexibility, availability, and security (Figure 2-1). In some cases, organizations can benefit by utilizing a hybrid of virtualization technologies on a single server deployment.

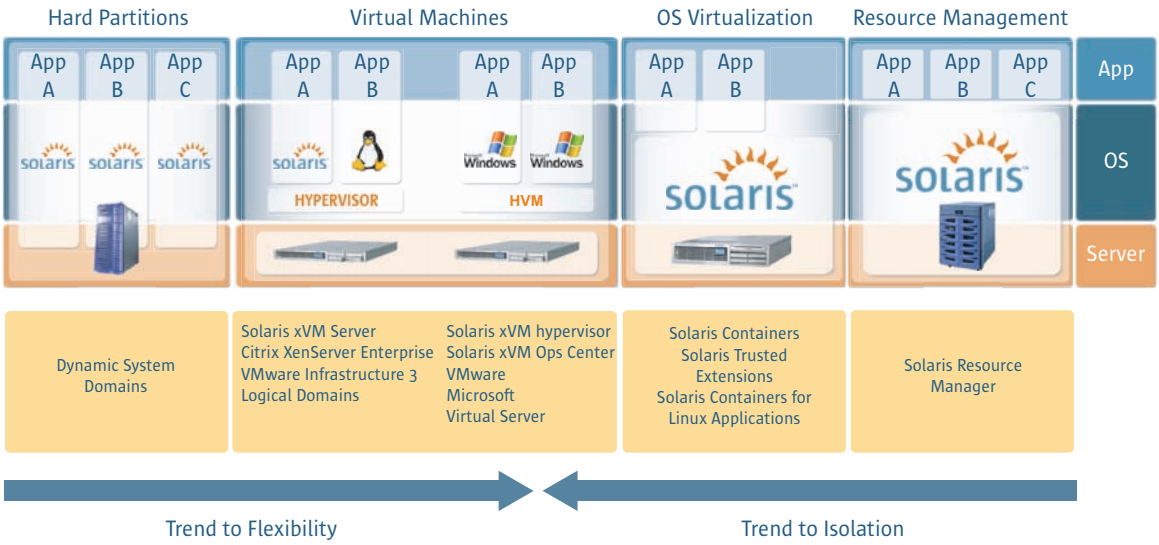

Figure 2-1. A variety of server virtualization technologies are available, each with different levels of flexibility, availability and security

Dynamic System Domains

Introduced by Sun in 1996, Dynamic System Domains is a proven technology for partitioning hardware resources. Each Dynamic System Domain enable a single system to be logically divided into multiple stand-alone domains, each running its own instance of the Solaris Operating System. Resources—including physical CPU, memory, and I/O resources—are assigned to specific domains, and can be allocated dynamically as needs dictate. In addition, failures in one domain do not affect applications running in other domains, increasing availability and providing a reliable, secure platform for running multiple applications simultaneously.

Operating System Virtualization

Operating system virtualization technology creates many private execution environments within a single instance of an operating system. With flexible, software-defined boundaries, virtual operating system environments are independent of the hardware layer and available for all platforms that support the operating system. As an example, Solaris Containers technology uses Solaris Zones and resource management capabilities to help IT organization harness and provision compute power into a secure, isolated runtime environment for individual applications. Each environment holds a unique identity and maintains resource and name space isolation. In addition, administrators can configure separate LAN or virtual local area network (VLAN) connections with exclusive IP stacks for individual Solaris Containers, enabling secure separation of network traffic. By taking advantage of the capabilities of Solaris Containers, administrators gain the ability to exert fine-grained control over rights and resources within a consolidated server without increasing the number of operating system instances to manage.

Resource Management

Resource management tools address the needs of consolidation efforts which require soft resource boundaries between applications. With no privileges to access underlying hardware, resource management software leverages operating system controls to govern utilization of CPU, memory, and I/O. For example, Solaris Resource Manager software enables system administrators to set and enforce policies that guarantee a share of CPU cycles and virtual memory space to individual applications. Administrators can also set upper limits on process count, number of logins, and connect time for each system user ID. In addition, Solaris Resource Manager software can be used along with other virtualization technologies to further define resource rights for each virtualized boundary. In fact, Solaris Resource Manager software enables dynamic allocation of processors and individual processor cores to a Solaris Container. The power to define and readily adjust compute resource levels within virtualized environments helps enterprises improve hardware utilization and better guarantee the quality of service for individual applications.

Virtual Machines

A virtual machine monitor enables enterprises to run multiple, different operating systems concurrently on a single physical machine. Virtual machines make use of a hypervisor to enable partitioning of resources on conventional hardware in a safe and effective manner without sacrificing performance or function. Hypervisors in virtual machine solutions which use full virtualization emulate a machine architecture down to the register level, enabling execution of unmodified guest operating systems. Some implementations, such as VMware ESX Server, dynamically rewrite portions of the hosted machine code to insert traps wherever virtual machine monitor intervention might be required. This translation also is applied to the entire guest OS kernel, resulting in added translation, execution, and caching delays for core operations. As such, the full virtualization approach incurs a high cost for update-intensive operations, potentially introducing significant processing overhead.

Rather than rely upon translation, paravirtualization virtual machine architectures, such as Sun xVM hypervisor and Citrix XenServer, rely upon the hypervisor exporting a modified version of the underlying physical hardware. Using paravirtualization increases efficiency (such as performance and isolation) by providing a virtual machine architecture on which guest operating systems run. Using paravirtualization, a guest operating system explicitly calls support functions implemented by the hypervisor code rather than trying to access system registers directly. Much of the code that traditionally exists in the lowest layers of an operating system is moved from the operating system and placed in the hypervisor. Sun Logical Domains optimize performance and security by implementing the hypervisor in firmware, setting Logical Domains technology apart from other virtual machine technology.

Sun xVM Hypervisor Architecture Overview

The Sun xVM hypervisor is a virtual machine monitor for the Solaris OS running on x86 and x64 platforms. With the Sun xVM hypervisor software, organizations can run multiple virtual machines simultaneously on a single physical system. Each virtual machine, known as a *domain*, runs its own complete and distinct operating system and has its own I/O devices.

The Sun xVM hypervisor is a Type 1 hypervisor that runs directly on the hardware platform. In essence, it sits between the x86 and x64 hardware and the operating system, and virtualizes the system's hardware in order to transparently share and partition system resources — including processors, memory, network interface cards and more — among guest domains (Figure 2-2). The hypervisor supports multiple operating system instances simultaneously. Each instance is called a *domain* and runs a full instance of an operating system. Two types of domains are supported: the control domain (typically called dom0) and guest domains (called domUs). While the hypervisor performs the low-level plumbing needed to provide a virtualized platform for operating systems, it relies on the control domain for most other tasks. For example, the control domain decides which domUs are created, which resources domUs can access, how much memory is allocated to a domU, and more. In addition, the control domain performs all device access.

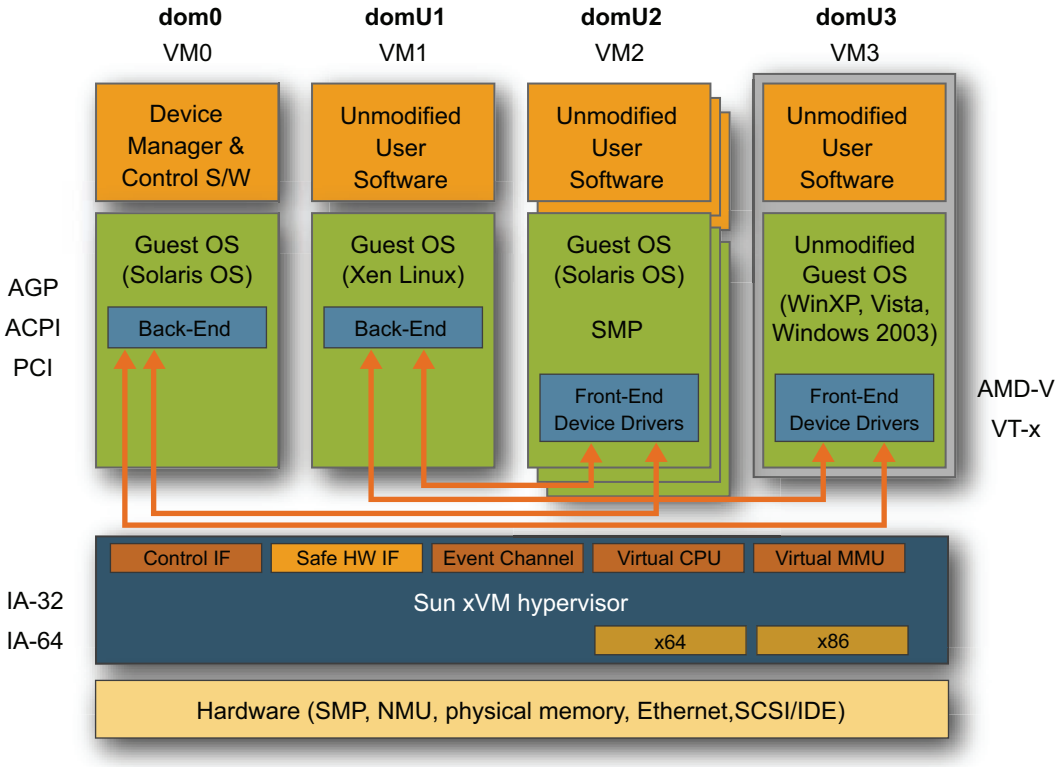

Figure 2-2. The Sun xVM hypervisor architecture

Virtual CPUs

The hypervisor includes a domain scheduler that schedules domains onto processors. Each domain, including dom0, has a set of virtual CPUs (VCPUs) that it owns and does not share with other domains. Each VCPU can be scheduled onto a real (running), blocked, or runable CPU. Scheduling decisions are made by a combination of the hypervisor credit scheduler, which distributes work among the set of real CPUs, and the configuration for each domain, which can restrict the CPUs a VCPU can run on, or bind a VCPU to a real CPU.

The credit-based CPU scheduler included in Sun xVM hypervisor is a fair share scheduler that balances virtual CPUs across available physical CPUs based on an assigned weight and the maximum amount of CPU resources that can be consumed by a domain, expressed as a percentage of one processor. Each CPU maintains a queue of runable VCPUs and keeps track of the amount of CPU resources consumed by the VCPU. When an event occurs that requires the assignment of a VCPU, the next available VCPU in the queue with sufficient resources is used. If a VCPU is not found in the queue, other CPUs are searched for VCPUs with sufficient access to resources. Note that:

- VCPUs must be allocated from dom0.
- Administrators must declare the number of VCPUs to be allocated to both dom0 and guest domains.
- The maximum number of VCPUs to be allocated must be specified during the domU boot process. However, the `virsh setvcpus` and `virsh vcpupin` commands can be used to dynamically set and pin VCPUs to processors.

Memory

When dom0 is booted, Sun xVM hypervisor reserves a small amount of memory for itself and assigns the remainder to dom0. When a domU is created, Sun xVM hypervisor takes memory from dom0 and assigns it to the new domU. Several commands can be used to manage memory assignments:

- The amount of memory each domain is assigned, called the memory reservation, can displayed using the `virsh dominfo` command.
- The reservation for a domain can be changed by using the `virsh setmem` command.
- The amount of memory assigned to dom0 can be limited with the `dom0_mem` options on the `grub` command line.
- The `virsh setmaxmem` command sets the maximum reservation for a domain, and cannot be exceeded.

To set the memory usage of a guest domain, the control tools create a xenstore entry specifying the target memory size. This triggers the balloon thread to adjust the domain's memory usage to the newly specified target.

- If the domain's reservation is shrinking, the balloon thread starts at the highest address and works down, collecting free pages. When the thread finds enough to satisfy the change in reservation, it unassigns the memory page, saves the page structures for possible future use, and gives the addresses back to the Sun xVM hypervisor.
- If the domain's reservation is growing, the balloon thread asks the Sun xVM hypervisor to provide the memory address. If page structures remain from a previous decrease in the reservation, the thread matches the new addresses with the page structures and releases them back into the system for general use. If there are no page structures left, the balloon thread carves out the first few pages for various structures the system needs, initializes them, and releases the rest into the system.

Running the Solaris OS in the Control Domain

In a virtualized environment including the Sun xVM hypervisor, an instance of the Solaris OS runs in the control domain on top of the hypervisor. This Solaris OS instance runs and behaves just like a standard instance of the operating system.

Running Multiple Operating Systems

In a fully virtualized environment, the operating system is unaware that it is running in a virtualized environment. With paravirtualization, the operating system is aware of the virtualization layer and works with the hypervisor to achieve higher performance. The Sun xVM hypervisor supports both full and paravirtualization. Since the control domain must work closely with the hypervisor, it is always paravirtualized. However, domUs may be fully virtualized or paravirtualized, and a system may have both varieties running simultaneously.

Full virtualization permits most x86 operating systems to run in a guest domain, including the Solaris OS, Linux, or Microsoft Windows environments. As a result, full virtualization requires the hypervisor to transparently intercept many operations that an operating system typically performs directly on the hardware. This interception enables the hypervisor to ensure that a domain cannot read or modify another domain's memory, interfere with its device access, or shut down the processors it is using for tasks. To implement full virtualization, the Sun xVM hypervisor requires special hardware support. Specifically, Intel processors with VT-x support, such as the Intel Core Duo, or AMD processors with AMD-V technology must be used.

Paravirtualization is built on top of a hypercall interface. Analogous to system calls, hypercalls are made by the operating system into the hypervisor layer. Hypercalls enable domains to initiate the creation of new domains, request the creation of address mappings, pass a buffer from one domain to another, send an interrupt between processors, and more. The hypervisor layer lets the operating system explicitly initiate the operations that are transparently intercepted in a fully virtualized environment. Since no interception is required in paravirtualization, specialized hardware support is not needed, enabling paravirtualization to run on any processor type. For paravirtualization, changes to the operating system are required—only operating systems incorporating those changes can be hosted in a paravirtualized domU. Currently, only the Solaris OS, Linux, FreeBSD, and NetBSD operating systems are suitably modified.

Chapter 3
Basic Control Domain Verification

This chapter describes how to verify that the Sun xVM hypervisor packages are installed, and the Service Management Facility (SMF) services are started correctly.

Preparing for Guest Domain Installation

Preparing the system for guest domain installation requires the Sun xVM hypervisor software packages be installed, as well as selecting and configuring a dom0 management environment.

Checking the Sun xVM Hypervisor Packages and the Control Domain

The following steps outline the process for verifying the installation of the Sun xVM hypervisor software packages, as well as updating the system and creating the control domain (dom0).

1. Verify that the following Sun xVM hypervisor software packages are installed on the system. These packages are installed as part of the core Solaris Operating System.

```
# pkginfo | grep xvm
system   SUNWxvmdomr      Hypervisor Domain Tools (Root)
system   SUNWxvmdomu      Hypervisor Domain Tools (Usr)
system   SUNWxvmh         Hypervisor Header Files
system   SUNWxvmhvm       Hypervisor HVM
system   SUNWxvmpv        xVM paravirtualized Drivers
system   SUNWxvmr         Hypervisor (Root)
system   SUNWxvmu         Hypervisor (Usr)

# pkginfo | grep virt
system   SUNWlibvirt      libvirt
system   SUNWvirtinst     virt-install

# pkginfo | grep urlgrabber
system   SUNWurlgrabber   urlgrabber
```

2. Confirm the Sun xVM hypervisor is listed as one of the installed operating system instances in the GRUB menu.

```
# bootadm list-menu

The location for the active GRUB menu is: /boot/grub/menu.lst
default 0
timeout 10
0 Solaris Express Community Edition snv85-mx3.1 X86
1 Solaris xVM
2 Solaris failsafe
3 Diagnostic Partition
```

If `Solaris xVM` does not appear in the above list, run the following command as the root user.

```
# bootadm -m upgrade
```

3. Change the default boot entry from 0 to 1 so that Sun xVM boots the next time the system is rebooted.

```
# bootadm set-menu default=1

The location for the active GRUB menu is: /boot/grub/menu.lst
default 1
timeout 10
0 Solaris Express Community Edition snv85-mx3.1 X86
1 Solaris xVM
2 Solaris failsafe
3 Diagnostic Partition
```

4. Reboot the system

```
# init 6
```

5. After confirming the above packages are installed, reboot the system and select `Solaris xVM` from the GRUB menu (`i86xpv`).
6. Confirm that the Sun xVM hypervisor services are started.

```
# svcs | grep xvm
online     0:40:15   svc:/system/xvm/console:default
online     0:40:15   svc:/system/xvm/domains:default
online     0:42:43   svc:/system/xvm/xend:default
online     0:42:43   svc:/system/xvm/store:default
```

7. Once the Sun xVM hypervisor is booted, check to see if the services are running. If the Sun xVM hypervisor services are not running, restart them using the following sequence of commands. Note that the order in which the services are started is important.

```
# svcadm enable xvm/store
# svcadm enable xvm/xend
# svcadm enable xvm/console
# svcadm enable xvm/domains
```

8. If the system is booted into `i86xpv` and the services are not enabled or started, start the services in the following order.

```
# svcadm enable xvm/store
# svcadm enable xvm/xend
# svcadm enable xvm/console
# svcadm enable xvm/domains
```

9. Run the `virsh list` command to ensure the software is running. A single domain named Domain-0 should be running. This domain is the control domain (dom0) which controls all guest operating systems that run on the machine.

```
# virsh list --all
Name       ID       MEM     CPUs    State   Time(s)
Domain-0   0        7932    4       r-----  4251.4
```

Creating the Right domU Management Environment

Prior to installing guest domains (domUs) and guest operating systems, a decision must be made regarding how best to manage the domU environment. Two methods are available: Solaris ZFS and a files-based approach.

Using Solaris ZFS

The Solaris ZFS approach uses the Solaris ZFS file system and volume datasets to enable the rapid provisioning of domUs. A revolutionary file system that is part of the Solaris 10 OS, Solaris ZFS is a scalable, high-performance and reliable file system that is based on the concept of storage pools. A storage pool may consist of part or all of one disk, and may span multiple disks. Multiple disks may be mirrored for redundancy and maximum availability, or striped for increased performance. Disks can be added to the storage pool at any time if more space is required. Multiple mount points can be defined within a storage pool. Each mount point shares the total amount of storage in the storage pool, and does not need to pre-define how much space is needed for the volume and file system. Additional space can be allocated on an as-needed basis.

More information on Solaris ZFS can be found *http://sun.com/solaris* and *http://opensolaris.org/os/community/zfs*.

Configuring the Solaris ZFS for the Sun xVM hypervisor requires the following steps.

1. Create a new storage pool using the `zpool(1M)` command. Note that `xvm` is the name of the pool and `c1t1d0` is the name of the disk to include in the storage pool. The name of the disk may be specified as a full device path, such as `/dev/dsk/c0t2d0s0`, or as the filename, such as `c1t1d0`. Multiple disks can be specified using space separated disk names, if disk striping is desired.

```
# zpool create xvm c1t1d0
```

2. Verify that the storage pool is created using either the `zpool status` command or the `zpool list` command.

```
# zpool status
pool: xvm
state: ONLINE
scrub: none requested
config:
NAME        STATE   READ    WRITE   CKSUM
xvm         ONLINE  0       0       0
c1t1d0      ONLINE  0       0       0
errors: No known data errors

# zpool list
NAME        SIZE    USED    AVAIL   CAP     HEALTH    ALTROOT
xvm         34G     156K    34.0G   0%      ONLINE    -
```

3. Use the `zpool iostat` command to view information about the I/O throughput of the newly created storage pool.

```
# zpool iostat
capacity  operations              bandwidth
pool      used    available       read    write   read    write
xvm       156K    34.0G           0       0       166     155
```

4. Create a Solaris ZFS volume that can be used to store the domU master image. Note that the Solaris ZFS volume is a dataset, however, it is presented as a block device and can be used like traditional UNIX block devices. The following example creates an 8 GB Solaris ZFS volume. The `–V` option specifies the volume size.

```
# zfs create -V 8g xvm/domU-master
```

5. List the Solaris ZFS volume block device nodes using the `ls` command. Note that Solaris ZFS volumes are identified as devices in the `/dev/zvol/dsk` and `/dev/zvol/rdsk` directories.

```
# ls -l /dev/zvol/dsk/xvm
total 2
lrwxrwxrwx  1     root      35        Apr 19 10:24 domu.master ->
../../ll/ll/devices/pseudo/zfs@0:1c
```

Setting Up Files

A files-based approach uses normal files to store domU disk images, and enables live migrations to take place when used in conjunction with the Network File System (NFS). The `virt-install` command can be used to create a file for the DomU, if it does not exist. The following example automatically creates a 10 GB file for use by the Solaris OS domU disk image.

```
# virt-install -n domU-solnvb82-PV --nographics -p \
-f /xvm_pool/guest-files/domU-solnvb82-PV.img -r 1024 -s 10 \
-l /xvm_pool/iso/nv/solarisdvd.iso

Starting install...
Creating storage file... 100% |=========================| 10 GB 00:00
Creating domain... 0 B 00:10
v3.0.4-1-xvm reno 'Wed Mar 19 11:43:59 2008 -0800 13229:fc49a937b8f7'
SunOS Release 5.11 Version snv_82 64-bit
Copyright 1983-2007 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
Configuring /dev
Solaris Interactive Text (Console session)
Using install cd in /dev/dsk/c0d1p0
Using RPC Bootparams for network configuration information.
Attempting to configure interface xnf0...
Skipped interface xnf0
Reading ZFS config: done.
Setting up Java. Please wait...
Beginning system identification...
Searching for configuration file(s)...
Search complete.
Discovering additional network configuration...
...
```

Basic Installation Using virt-install

Guest domains can be installed using the `virt-install` command. Options to the `virt-install` command enable guest domains to be installed using the command, interactively, or via Netinstall or Solaris JumpStart™. Table 3-1 describes the key options to the `virt-install` command.

Table 3-1. Command line options

Option	Description
–accelerate	Instructs the command to use kernel acceleration capabilities.
–arch	Identifies the CPU architecture to simulate. Allowed values are x86 and sparc.
–autocf=AUTOCF	Identifies the path that contains Kickstart or JumpStart information.
–bridge	Specified the bridge to which to connect the network interface.
–cdrom	Identifies the file to use as a virtual CD-ROM device for fully virtualized guests. This option works with HVM domains, and it ignored with paravirtualized domains.
–check-cpu	Check that the number of VCPUs does not exceed the number of physical CPUs, and issue a warning if the count is exceeded.
–connect	Connect to a hypervisor located at the specified URI.
-d or –debug	Display debugging information

Option	Description
-f or --file	Specifies the file to use as a disk image.
-h or --help	Display a list of virt-install options.
-k or --keymap	Set up a keymap for a graphical console.
-l or --location	Identifies the installation source for a paravirtualized guest. The location can be specified as nfs:host/path, http://host/path, ftp://host/path, or as a file system path in the control domain.
-m or --mac	Specifies a fixed MAC address for the guest.
-n or --name	The name or handle of the guest instance. This name is typically used by the control domain to reference the domU and its hostname.
–noacpi	Disables the Advanced Configuration and Power Interface (ACPI) for a fully virtualized guest.
–noapic	Disables the Advanced Programmable Interrupt Controller (APIC) for a fully virtualized guest.
–noconsole	Specifies that a graphics console is not to be set up for a guest.
–nographics	Indicates no graphical output. The console is serial. This option is always used with the -p option.
–nonsparse	Specifies that spare files are not to be used for disks. Use of this option can slow the guest creation process.
-os-type	Specifies the operating system type for a fully virtualized guest. Valid values include solaris, unix, linux, and windows.
os-variant	Specifies the operating system variant for a fully virtualized guest. Valid values include fedora6, rhel5, solaris10, win2k, and vista.
-p or --paravirt	Indicates a paravirtualized guest. This option is always used in combination with the –nographics option. A paravirtualized install cannot be done with a graphics console.
-r or --ram	Specifies the amount of memory to use for a guest, in megabytes. A minimum of 1 GB is recommended.
-s	Specifies the size of the disk image (if it does not exist), in gigabytes.
-u or –uuid	Specifies the UUID for the guest as a 32-digit hexadecimal number. If a UUID it not specified, the system generates a random UUID.
-v or –hvm	Identifies the guest as a hardware-assisted virtual machine.
–vcpus	Specifies the number of virtual CPUs to configure for the guest.
–vnc	Specifies the use of virtual network computing for graphics support. This option only works with HVM domains.

Option	Description
–vncport	Identifies the port to use for virtual network computing.
-w or –network	Connects the guest to a virtual networking and forwards to a physical network with Network Address Translation (NAT).
-x or –extra-args	Specifies additional arguments to pass to the installer for paravirtualized guests.

Installation via the Command Line

The following steps describe how to install a guest domain from the command line using an ISO image.

1. Create and install the guest domain.

```
$ virt-install --nographics -n domu-x16 --paravirt \
-f /xvm/domu-x16.img -r 1011 --mac=aa:ff:bb:aa:28:16 -s 18\
-l /net/inst.server/export/xVMDomU/x_iso/63-0419-nd.iso
Starting install...
Creating domain...
SunOS Release 5.11 Version 64-bit
...
```

2. Once the domain is created, `sysidcfg` setup is initiated. Answer the configuration questions asked, including selection of a language, type of terminal in use, etc.

Installation with Solaris JumpStart™

The following steps describe how to install a guest domain using Solaris JumpStart.

1. Create and install the guest domain.

```
$ virt-install --name solaris1 --ram 1024 --nographics \
--file /dev/zvol/dsk/pool/solaris1-disk \
--location nfs:install.domain.com:/export/solaris/nv75 \
--autocf nfs:install.domain.com:/export/jumpstart/solaris1
```

2. Once the domain is created, `sysidcfg` setup is initiated. Answer the configuration questions asked, including selection of a language, type of terminal in use, etc.

Configuring Domains to Automatically Start at Boot Time

Whether to run the Solaris OS as a virtualized dom0 for as a standalone operating system is a boot time decision.

- To run the Solaris OS as a standalone operating system, continue to use the same GRUB menu entries currently in use. For example:

```
title Solaris Nevada snv_86 X86
kernel$ /platform/i86pc/kernel/$ISADIR/unix
module$ /platform/i86pc/$ISADIR/boot_archive
```

- To run the Solaris OS as a control domain, instruct GRUB to boot the Sun xVM hypervisor. The Sun xVM hypervisor loads and boots the Solaris OS instance. An example GRUB entry to accomplish this task follows.

```
title Solaris xVM
root (hd0,0,d)
kernel$ /boot/$ISADIR/xen.gz console=com1 com1=9600,8n1
module$ /platform/i86xpv/kernel/$ISADIR/unix \
        /platform/i86xpv/kernel/$ISADIR/unix
module$ /platform/i86pc/$ISADIR/boot_archive
```

Alternatively, the `bootadm` command can be used.

- List the default boot configuration and available command options.

```
# bootadm list-menu
```

- Change the `bootadm` options using the following command.

```
# bootadm set-menu default=menu_list_number
```

Chapter 4
Virtual Machine Management

The Sun xVM hypervisor includes both command-line and graphical user interface tools which can be used to create, install, and manage domains (Table 4-1). Note that the following commands must be run as the `root` user on the host operating system. The commands cannot be run on the guest operating system running in the virtual machine.

Table 4-1. Some command line utilities available for managing guest domains

Command	Description
virsh	A simpler, cross-platform alternative to the xm command that queries the state of the hypervisor, creates the control domain, manipulates configurations, and more.
virsh autostart	Configure a domain to automatically start at boot time.
virsh connect	Connect to the hypervisor.
virsh create *file*	Create a domain from a specified XML file.
virsh define *file*	Define a domain from a specified XML file.
virsh destroy *domain-id*	Terminate a domain immediately.
virsh dominfo *domain-id*	Return basic information about a domain.
virsh domstate *domain-id*	Return the state of a running domain.
virsh list	Print information about one or more domains on the system.
virsh rebooot *domain-id*	Reboot a domain.
virsh restore *state-file*	Restore a domain from a saved state file.
virsh resume *domain-id*	Resume a suspended domain.
virsh save *state-file*	Save a running domain to a state file so that it can be restored at a later time.
virsh shutdown *domain-id*	Shut down a domain gracefully.
virsh suspend *domain-id*	Suspend a running domain.
virsh undefine *domain-id*	Undefine the configuration for an inactive domain.
virt-install	A command line utility that creates guest domains and installs virtual machine instances. Installation can be done via the command line or performed interactively.
virt-manager	A graphical control tool for the hypervisor and control domain that makes it easier to manage virtual machine instances.
xm	A hypervisor-specific command line utility for managing domains.

Chapter 5
Advanced Installation and Configuration

Once the control domain (dom0) is configured and operational, guest domains (domUs) can be configured and installed. The following sections describe how to create a Solaris OS domain, which can be used as a template for future domU domains. Steps include:

- Configuring naming and storage
- Installing the guest domain
- Booting, migrating and halting the guest domain

Creating the Configuration File

The Solaris OS guest domain needs a configuration file (Python script) associated with it that describes to the guest domain how to boot.

Directives Used in the Solaris x86 xVM Domain Configuration File

Several directives are used by the domU in the Solaris x86 xVM domain configuration file. While not an exhaustive list, the directives identified in Table 5-1 explain what the domU needs for a minimal guest configuration file.

Table 5-1. Directives used in the Sun xVM domain configuration file

Directive	Description
name=”solaris”	A unique name given to the guest domain name.
memory=”1024”	The initial memory allocation for the guest, expressed in megabytes.
extra=”k”	
root=”/dev/dsk/c0d0s0”	The root device used for the guest domain (domU).
disk=[‘file:/xen/guests/solaris-b57/disk.img,0,w’]	
vif=[“]	A network interface directive that may contain an entry for each network device. The default networks is specified as shown.
on_shutdown=”destroy”	
on_reboot=”restart”	
on_crash=”destroy”	

Example Configuration File

The following code listing details an example configuration file. While values are defined, it is important to modify the file to reflect a given system and network configuration.

```
# The name of this guest domain:
#name = "solaris-pv"

# The amount of memory assigned to this domain at the time it boots
# (in megabytes):
#memory = "1024"

# This is necessary after install to avoid a fatal failure to
# boot with the message "Cannot mount /devices."
#root = "/dev/dsk/c0d0s0"

# Disk devices:
#
# - Guest domain using a zvol (see zfs(1M)) as a disk image:
#disk = ['phy:/dev/zvol/dsk/xen/domU-master,0,w']
#
# - Guest domain using a zvol (see zfs(1M)) as a disk image and
#   installing Solaris from an ISO image file:
#disk = ['file:/xen/isos/solaris-pv/82-nd.iso,6:cdrom,r',
#       'phy:/dev/zvol/dsk/xen/domU-master,0,w']
#
# - Guest domain using a file as a disk image:
#disk = ['file:/xen/isos/solaris-pv.img,0,w']

# Network devices:
#
# - A single network device connected via the default NIC with an
#   automatically allocated MAC address:
#vif = ['']
#
# - A single network device connected via the 'bge1' NIC with a
#   particular MAC address:
#vif = ['bridge=bge1,mac=aa:00:11:22:33:44']

# Default values for behaviour on guest domain restart:
#on_shutdown = "destroy"
#
# During an install, set this to "destroy" to avoid booting from the CD
# after the installation.
#on_reboot = "restart"
#on_crash = "destroy"
```

Installing the First Guest Domain

The following steps outline how to create the virtual machine instance.

1. Create the virtual machine instance using the `xm` command.

```
$ xm create -c solaris-pv.py
Using config file "./solaris-pv.py".
SunOS Release 5.11 Version snv_82 64-bit
Copyright 1983-2007 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Configuring /dev
Solaris Interactive Text (Console session)
Using install cd in /dev/dsk/c0d6p0
Using RPC Bootparams for network configuration information.
Attempting to configure interface xnf0...
Skipped interface xnf0
Setting up Java. Please wainning system identification...
Searching for configuration file(s)...
Search complete.
Discovering additional network configuration...
```

2. Select a language for the installation.

```
Select a Language

   1. English
   2. French
   3. German
   4. Italian
   5. Japanese
   6. Korean
   7. Simplified Chinese
   8. Spanish
   9. Swedish
  10. Traditional Chinese

Please make a choice (1 - 10), or press h or ? for help: 1
```

3. Specify the terminal type in use.

```
What type of terminal are you using?
 1) ANSI Standard CRT
 2) DEC VT52
 3) DEC VT100
 4) Heathkit 19
 5) Lear Siegler ADM31
 6) PC Console
 7) Sun Command Tool
 8) Sun Workstation
 9) Televideo 910
 10) Televideo 925
 11) Wyse Model 50
 12) X Terminal Emulator (xterms)
 13) CDE Terminal Emulator (dtterm)
 14) Other
Type the number of your choice and press Return: 12
Completing system identification...
in.rdisc: No interfaces up
```

4. Proceed through the Solaris Installation Program.

```
- The Solaris Installation Program --------------------------------
  The Solaris installation program is divided into a series of short
  sections where you'll be prompted to provide information for the
  installation. At the end of each section, you'll be able to change
  the selections you've made before continuing.

  About navigation...
        - The mouse cannot be used
        - If your keyboard does not have function keys, or they do
          not respond, press ESC; the legend at the bottom of the
          screen will change to show the ESC keys to use for
          navigation.
-------------------------------------------------------------------
    F2_Continue    F6_Help

- Identify This System -------------------------------------------

  On the next screens, you must identify this system as networked or
  non-networked, and set the default time zone and date/time.

  If this system is networked, the software will try to find the
  information it needs to identify your system; you will be prompted
  to supply any information it cannot find.

  > To begin identifying this system, press F2.

- Network Connectivity -------------------------------------------
  Specify Yes if the system is connected to the network by one of the
  Solaris or vendor network/communication Ethernet cards that are
  supported on the Solaris CD. See your hardware documentation for
  the current list of supported cards.

  Specify No if the system is connected to a network/communication
  card that is not supported on the Solaris CD, and follow the
  instructions listed under Help.
      Networked
      ---------
      [X] Yes
      [ ] No
-------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- DHCP for xnf0 --------------------------------------------------

  Specify whether or not this network interface should use DHCP to
  configure itself.  Choose Yes if DHCP is to be used, or No if the
  network interface is to be configured manually.

  NOTE: DHCP support will not be enabled, if selected, until after
  the system reboots.

      Use DHCP for xnf0
      -----------------
      [ ] Yes
      [X] No
-------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help
```

```
- Host Name for xnf0 -------------------------------------------------

  Enter the host name which identifies this system on the network.  The
  name must be unique within your domain; creating a duplicate host name
  will cause problems on the network after you install Solaris.

  A host name must have at least one character; it can contain letters,
  digits, and minus signs (-).

      Host name for xnf0 winchester
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- IP Address for xnf0 ------------------------------------------------

  Enter the Internet Protocol (IP) address for this network interface.
  It must be unique and follow your site's address conventions, or a
  system/network failure could result.

  IP addresses contain four sets of numbers separated by periods (for
  example 129.200.9.1).

    IP address for xnf0 129.156.103.144
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- Subnet for xnf0 ----------------------------------------------------

  On this screen you must specify whether this system is part of a
  subnet.  If you specify incorrectly, the system will have problems
  communicating on the network after you reboot.

    > To make a selection, use the arrow keys to highlight the option and
      press Return to mark it [X].

        System part of a subnet
        -----------------------
        [X] Yes
        [ ] No

  -------------------------------------------------------------------------
      Esc-2_Continue    Esc-6_Help

- Netmask for xnf0 ---------------------------------------------------

  On this screen you must specify the netmask of your subnet.  A default
  netmask is shown; do not accept the default unless you are sure it is
  correct for your subnet.  A netmask must contain four sets of numbers
  separated by periods (for example 255.255.255.0).

Netmask for xnf0 255.255.255.0

-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help
```

```
- IPv6 for xnf0 ------------------------------------------------------

  Specify whether or not you want to enable IPv6, the next generation
  Internet Protocol, on this network interface.  Enabling IPv6 will have
  no effect if this machine is not on a network that provides IPv6
  service.  IPv4 service will not be affected if IPv6 is enabled.

    > To make a selection, use the arrow keys to highlight the option and
      press Return to mark it [X].

      Enable IPv6 for xnf0
      --------------------
      [ ] Yes
      [X] No

-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- Set the Default Route for xnf0 ---------------------------------------

  To specify the default route, you can let the software try to detect
  one upon reboot, you can specify the IP address of the router, or you
  can choose None.  Choose None if you do not have a router on your
  subnet.

  > To make a selection, use the arrow keys to select your choice and
  press Return to mark it [X].

      Default Route for xnf0
      --------------------------
      [X] Detect one upon reboot
      [ ] Specify one
      [ ] None
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- Confirm Information for xnf0 ----------------------------------------

  > Confirm the following information.  If it is correct, press F2;
    to change any information, press F4.

                  Networked: Yes
                   Use DHCP: No
                  Host name: winchester
                 IP address: 129.156.103.144
    System part of a subnet: Yes
                    Netmask: 255.255.255.0
                Enable IPv6: No
              Default Route: Detect one upon reboot
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-4_Change    Esc-6_Help
```

```
- Configure Security Policy: ----------------------------------------

  Specify Yes if the system will use the Kerberos security mechanism.

  Specify No if this system will use standard UNIX security.

      Configure Kerberos Security
      ---------------------------
      [ ] Yes
      [X] No
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- Confirm Information ------------------------------------------------

  > Confirm the following information.  If it is correct, press F2;
    to change any information, press F4.

    Configure Kerberos Security: No

-------------------------------------------------------------------------
    Esc-2_Continue    Esc-4_Change    Esc-6_Help

- Name Service ------------------------------------------------------

  On this screen you must provide name service information.  Select the
  name service that will be used by this system, or None if your system
  will either not use a name service at all, or if it will use a name
  service not listed here.

  > To make a selection, use the arrow keys to highlight the option
    and press Return to mark it [X].

      Name service
      ------------
      [ ] NIS+
      [X] NIS
      [ ] DNS
      [ ] LDAP
      [ ] None
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- Domain Name -------------------------------------------------------

  On this screen you must specify the domain where this system resides.
  Make sure you enter the name correctly including capitalization and
  punctuation.

    Domain name: niscafe.uk.sun.com
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help
```

```
- Name Server ------------------------------------------------------------

  On this screen you must specify how to find a name server for this
  system. You can let the software try to find one, or you can specify
  one.  The software can find a name server only if it is on your local
  subnet.

  > To make a selection, use the arrow keys to highlight the option and
    press Return to mark it [X].

      Name server
      ---------------
      [ ] Find one
      [X] Specify one

------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- Name Server Information ------------------------------------------------

  On this screen you must enter the host name and IP address of your name
  server.  Host names must be at least two characters, and may contain
  letters, digits, and minus signs (-).  IP addresses must contain four
  sets of numbers separated by periods (for example 129.200.9.1).

     Server's host name: nis-uk
    Server's IP address: 129.156.85.41
------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- Confirm Information ----------------------------------------------------

  > Confirm the following information.  If it is correct, press F2;
    to change any information, press F4.

           Name service: NIS
            Domain name: niscafe.uk.sun.com
            Name server: Specify one
     Server's host name: nis-uk
    Server's IP address: 129.156.85.41

------------------------------------------------------------------------
    Esc-2_Continue    Esc-4_Change    Esc-6_Help

- NFSv4 Domain Name ------------------------------------------------------

  NFS version 4 uses a domain name that is automatically derived from the
  system's naming services. The derived domain name is sufficient for
  most configurations. In a few cases, mounts that cross domain
  boundaries might cause files to appear to be owned by "nobody" due to
  the lack of a common domain name.

The current NFSv4 default domain is: "uk.sun.com"
      NFSv4 Domain Configuration
      ------------------------------------------
      [X] Use the NFSv4 domain derived by the system
      [ ] Specify a different NFSv4 domain
```

```
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- Confirm Information for NFSv4 Domain --------------------------------

  > Confirm the following information.  If it is correct, press F2;
    to change any information, press F4.

    NFSv4 Domain Name:  << Value to be derived dynamically >>
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-4_Change    Esc-6_Help

- Time Zone -----------------------------------------------------------

  On this screen you must specify your default time zone.  You can
  specify a time zone in three ways:  select one of the continents or
  oceans from the list, select other - offset from GMT, or other -
  specify time zone file.

  > To make a selection, use the arrow keys to highlight the option and
    press Return to mark it [X].

      Continents and Oceans
      ---------------------------------
  -   [ ] Africa
  ?   [ ] Americas
  ?   [ ] Antarctica
  ?   [ ] Arctic Ocean
  ?   [ ] Asia
  ?   [ ] Atlantic Ocean
  ?   [ ] Australia
  ?   [X] Europe
  v   [ ] Indian Ocean
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- Country or Region ---------------------------------------------------
  > To make a selection, use the arrow keys to highlight the option and
    press Return to mark it [X].

      Countries and Regions
      ----------------------
  -   [ ] Aaland Islands
  ?   [ ] Albania
  ?   [ ] Andorra
  ?   [ ] Austria
  ?   [ ] Belarus
  ?   [ ] Belgium
  ?   [ ] Bosnia & Herzegovina
  ?   [X] Britain (UK)
  ?   [ ] Bulgaria
  ?   [ ] Croatia
  ?   [ ] Czech Republic
  ?   [ ] Denmark
  v   [ ] Estonia
-------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help
```

```
- Date and Time ------------------------------------------------------

  > Accept the default date and time or enter new values.

  Date and time: 2007-05-21 10:20

    Year   (4 digits) : 2007
    Month  (1-12)     : 05
    Day    (1-31)     : 21
    Hour   (0-23)     : 10
    Minute (0-59)     : 20

------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

- Confirm Information ------------------------------------------------

  > Confirm the following information.  If it is correct, press F2;
    to change any information, press F4.

        Time zone: GB
    Date and time: 2007-05-21 10:20:00

------------------------------------------------------------------------
    Esc-2_Continue    Esc-4_Change    Esc-6_Help

- Root Password ------------------------------------------------------

  Please enter the root password for this system.

  The root password may contain alphanumeric and special characters. For
  security, the password will not be displayed on the screen as you type
  it.

  > If you do not want a root password, leave both entries blank.

    Root password:  ******
    Root password:  ******
------------------------------------------------------------------------
    Esc-2_Continue    Esc-6_Help

System identification is completed.
- Solaris Interactive Installation ----------------------------------

  On the following screens, you can accept the defaults or you can
  customize how Solaris software will be installed by:
        - Selecting the type of Solaris software to install
        - Selecting disks to hold software you've selected
        - Selecting unbundled products to be installed with Solaris
        - Specifying how file systems are laid out on the disks

After completing these tasks, a summary of your selections (called a
profile) will be displayed.
```

```
 There are two ways to install your Solaris software:

  - "Standard" installs your system from a standard Solaris
    Distribution. Selecting "Standard" allows you to choose between
    initial install and upgrade, if your system is upgradable.
  - "Flash" installs your system from one or more Flash Archives.
-------------------------------------------------------------------------
     F2_Standard    F4_Flash    F5_Exit    F6_Help

- Eject a CD/DVD Automatically? ---------------------------------------

  During the installation of Solaris software, you may be using one or
  moreCDs/DVDs. With the exception of the currently booted CD/DVD, you
  can choose to have the system eject each CD/DVD automatically after it
  is installed or you can choose to manually eject each CD/DVD.

  Note: The currently booted CD/DVD must be manually ejected during
  system reboot.

            [X] Automatically eject CD/DVD
            [ ] Manually eject CD/DVD
-------------------------------------------------------------------------
     F2_Continue    F3_Go Back    F5_Exit

- Reboot After Installation? ------------------------------------------

  After Solaris software is installed, the system must be rebooted. You
  can choose to have the system automatically reboot, or you can choose
  to manually reboot the system if you want to run scripts or do other
  customizations before the reboot.  You can manually reboot a system by
  using the reboot(1M) command.

            [X] Auto Reboot
            [ ] Manual Reboot
-------------------------------------------------------------------------
     F2_Continue    F3_Go Back    F5_Exit

- Information ---------------------------------------------------------

        You may need to manually eject the CD/DVD or select a different
        boot device after reboot to avoid repeating the installation
        process.
-------------------------------------------------------------------------
     Esc-2_OK

- Choose Media --------------------------------------------------------

  Please specify the media from which you will install the Solaris
  Operating System.

  Media:
      [X]  CD/DVD
      [ ]  Network File System
-------------------------------------------------------------------------
     F2_Continue    F3_Go Back    F5_Exit
```

```
- Initializing ----------------------------------------------------------

  The system is being initialized.

  Loading install media, please wait...

- License ---------------------------------------------------------------
 -      Sun Microsystems, Inc. ("Sun")
 -      SOFTWARE LICENSE AGREEMENT
 -
 -      READ THE TERMS OF THIS AGREEMENT ("AGREEMENT") CAREFULLY BEFORE
 -      OPENING SOFTWARE MEDIA PACKAGE. BY OPENING SOFTWARE MEDIA
 -      PACKAGE, YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE
 -      ACCESSING SOFTWARE ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF
 -      THESE TERMS BY SELECTING THE "ACCEPT"(OR EQUIVALENT) BUTTON AT
 -      THE END OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THE
 -      TERMS, PROMPTLY RETURN THE UNUSED SOFTWARE TO YOUR PLACE OF
 -     PURCHASE FOR A REFUND OR, IF SOFTWARE IS ACCESSED ELECTRONICALLY,
 -      SELECT THE "DECLINE" (OR EQUIVALENT) BUTTON AT THE END OF THIS
 -      AGREEMENT. IF YOU HAVE SEPARATELY AGREED TO LICENSE TERMS
 -     ("MASTER TERMS") FOR YOUR LICENSE TO THIS SOFTWARE, THEN SECTIONS
 -      1-6 OF THIS AGREEMENT ("SUPPLEMENTAL LICENSE TERMS") SHALL
 -      SUPPLEMENT AND SUPERSEDE THE MASTER TERMS IN RELATION TO THIS
 -      SOFTWARE.
 -
 v      1. Definitions.

-------------------------------------------------------------------------
     Esc-2_Accept License    F5_Exit

- Select Geographic Regions ---------------------------------------------

  Select the geographic regions for which support should be installed.

  > [ ] Australasia
  > [ ] Asia
  > [ ] Eastern Europe
  > [ ] Northern Europe
  > [ ] Northern Africa
  > [ ] Middle East
  > [ ] Southern Europe
  > [ ] South America
  > [ ] Central America
  > [/] Central Europe
  > [/] North America
  > [X] Western Europe

Move left, right, up, down using the arrow keys
-------------------------------------------------------------------------
     Esc-2_Continue    F3_Go Back    F5_Exit    F6_Help
```

```
- Select System Locale -------------------------------------------------

  Select the initial locale to be used after the system has been
  installed.

    [X]     POSIX C ( C )
            Western Europe
    [ ]     Belgium-Flemish (ISO8859-1) ( nl_BE.ISO8859-1 )
    [ ]     Belgium-Flemish (ISO8859-15 - Euro) ( nl_BE.ISO8859-15 )
    [ ]     Belgium-Flemish (UTF-8) ( nl_BE.UTF-8 )
    [ ]     Belgium-French (ISO8859-1) ( fr_BE.ISO8859-1 )
    [ ]     Belgium-French (ISO8859-15 - Euro) ( fr_BE.ISO8859-15 )
    [ ]     Belgium-French (UTF-8) ( fr_BE.UTF-8 )
    [ ]     France (UTF-8) ( fr_FR.UTF-8 )
    [ ]     France (ISO8859-1) ( fr_FR.ISO8859-1 )
    [ ]     France (ISO8859-15 - Euro) ( fr_FR.ISO8859-15 )
    [ ]     Great Britain (ISO8859-1) ( en_GB.ISO8859-1 )
    [ ]     Great Britain (ISO8859-15 - Euro) ( en_GB.ISO8859-15 )
    [ ]     Great Britain (UTF-8) ( en_GB.UTF-8 )
    [ ]     Ireland (ISO8859-1) ( en_IE.ISO8859-1 )
    [ ]     Ireland (ISO8859-15 - Euro) ( en_IE.ISO8859-15 )

----------------------------------------------------------------------
     Esc-2_Continue    F3_Go Back    F5_Exit    F6_Help

- Additional Products --------------------------------------------------

  To scan for additional products, select the location you wish to scan.
  Products found at the selected location that are in a Web Start Ready
  install form will be added to the Products list.

  Web Start Ready product scan location:

      [X]  None
      [ ]  CD/DVD
      [ ]  Network File System
----------------------------------------------------------------------
     Esc-2_Continue    F3_Go Back    F5_Exit

- Select Software ------------------------------------------------------

  Select the Solaris software to install on the system.

  NOTE: After selecting a software group, you can add or remove software
  by customizing it. However, this requires understanding of software
  dependencies and how Solaris software is packaged.

      [ ]  Entire Distribution plus OEM support ....... 6106.00 MB
      [X]  Entire Distribution ........................ 6105.00 MB
      [ ]  Developer System Support ................... 5914.00 MB
      [ ]  End User System Support .................... 4765.00 MB
      [ ]  Core System Support ........................ 1058.00 MB
      [ ]  Reduced Networking Core System Support ..... 1013.00 MB
----------------------------------------------------------------------
     Esc-2_Continue    F3_Go Back    F4_Customize    F5_Exit    F6_Help
```

```
- Select Disks ---------------------------------------------------------

  On this screen you must select the disks for installing Solaris
  software. Start by looking at the Suggested Minimum field; this value
  is the approximate space needed to install the software you've
  selected. Keep selecting disks until the Total Selected value exceeds
  the Suggested Minimum value.

NOTE: ** denotes current boot disk

  Disk Device                                            Available Space

=============================================================================
  [ ]    c0d0                                                     0 MB

                                     Total Selected:      0 MB
                                  Suggested Minimum:   4755 MB
-----------------------------------------------------------------------------
     Esc-2_Continue    F3_Go Back    F4_Edit    F5_Exit    F6_Help

- No Solaris fdisk Partition -------------------------------------------

There is no Solaris fdisk partition on this disk. You must create a
Solaris fdisk partition if you want to use it to install Solaris
software.
-----------------------------------------------------------------------------
     Esc-2_OK    F5_Cancel

- Create Solaris fdisk Partition ---------------------------------------

  There is no Solaris fdisk partition on this disk. You must create a
  Solaris fdisk partition if you want to use this disk to install
  Solaris software.

  One or more of the following methods are available: have the software
  install a boot partition and a Solaris partition that will fill the
  entire fdisk, install just a Solaris partition that will fill the
  entire fdisk (both of these options will overwrite any existing fdisk
  partitions), install a Solaris partition on the remainder of the disk,
  install a boot partition on the disk, or manually lay out the Solaris
  fdisk partition.

    [X]  Use entire disk for Solaris partition (10228 MB)
    [ ]  Manually create fdisk partitions

-----------------------------------------------------------------------------
     Esc-2_OK    F5_Cancel    F6_Help

- Select Disks ---------------------------------------------------------

  On this screen you must select the disks for installing Solaris
  software. Start by looking at the Suggested Minimum field; this value
  is the approximate space needed to install the software you've
  selected. Keep selecting disks until the Total Selected value exceeds
  the Suggested Minimum value.

  NOTE: ** denotes current boot disk
```

```
 Disk Device                                          Available Space
=============================================================================
  [X]    c0d0                                        10189 MB  (F4 to edit)

                                     Total Selected:  10189 MB
                                  Suggested Minimum:   4755 MB
-----------------------------------------------------------------------------
     Esc-2_Continue    F3_Go Back    F4_Edit    F5_Exit    F6_Help

- Automatically Layout File Systems? ---------------------------------

  Do you want to use auto-layout to automatically layout file systems?
  Manually laying out file systems requires advanced system
  administration skills.

-----------------------------------------------------------------------------
F2_Auto Layout    F3_Go Back    F4_Manual Layout    F5_Exit    F6_Help

- Automatically Layout File Systems ---------------------------------

  On this screen you must select all the file systems you want auto-
  layout to create, or accept the default file systems shown.

  NOTE: For small disks, it may be necessary for auto-layout to break up
  some of the file systems you request into smaller file systems to fit
  the available disk space. So, after auto-layout completes, you may
  find file systems in the layout that you did not select from the list
  below.
            File Systems for Auto-layout
            ========================================
            [X]  /
            [ ]  /opt
            [ ]  /usr
            [ ]  /usr/openwin
            [ ]  /var
            [X]  swap
-----------------------------------------------------------------------------
     Esc-2_Continue    F5_Cancel    F6_Help

- File System and Disk Layout ---------------------------------------

  The summary below is your current file system and disk layout, based
  on the information you've supplied.

  NOTE: If you choose to customize, you should understand file systems,
  their intended purpose on the disk, and how changing them may affect
  the operation of the system.

      File sys/Mnt point      Disk/Slice        Size

=============================================================================
      /                       c0d0s0            5789 MB
      swap                    c0d0s1            596 MB
      overlap                  c0d0s2           10213 MB
      /export/home            c0d0s7            3804 MB
-----------------------------------------------------------------------------
     Esc-2_Continue    F3_Go Back    F4_Customize    F5_Exit    F6_Help
```

```
- Mount Remote File Systems? ---------------------------------------

Do you want to mount software from a remote file server? This may be
necessary if you had to remove software because of disk space problems.

------------------------------------------------------------------------
Esc-2_Continue    F3_Go Back    F4_Remote Mounts    F5_Exit    F6_Help

- Profile ----------------------------------------------------------

The information shown below is your profile for installing Solaris
software. It reflects the choices you've made on previous screens.

NOTE:  You must change the BIOS because you have changed the default
boot device.
========================================================================

 -              Installation Option: Initial
 -                      Boot Device: c0d0
 -                  Client Services: None
 -
 -                          Regions: Western Europe
 -                    System Locale: C ( C )
 -
 -                         Software: Solaris 11, Entire Distribution
 -
 -      File System and Disk Layout: /                   c0d0s0 5789 MB
 v                                   swap                c0d0s1  596 MB

------------------------------------------------------------------------
     Esc-2_Begin Installation    F4_Change    F5_Exit    F6_Help

Preparing system for Solaris install

Configuring disk (c0d0)
        - Creating Fdisk partition table
        - Creating Solaris disk label (VTOC)

Creating and checking UFS file systems
        - Creating / (c0d0s0)
        - Creating /export/home (c0d0s7)

Beginning Solaris software installation
Solaris Initial Install

      MBytes Installed:   253.98
      MBytes Remaining:  3652.17

            Installing: Office core module for StarOffice 8

|           |           |           |           |           |
0          20          40          60          80         100

Solaris 11 software installation succeeded
```

```
Customizing system files
        - Mount points table (/etc/vfstab)
        - Network host addresses (/etc/hosts)
        - Environment variables (/etc/default/init)

Cleaning devices

Customizing system devices
        - Physical devices (/devices)
        - Logical devices (/dev)

Installing boot information

Installation log location
        - /a/var/sadm/system/logs/install_log (before reboot)
        - /var/sadm/system/logs/install_log (after reboot)

Installation complete
Executing SolStart postinstall phase...
Executing finish script "patch_finish"...
Finish script patch_finish execution completed.

Executing JumpStart postinstall phase...

The begin script log 'begin.log'
        is located in /var/sadm/system/logs after reboot.

The finish script log 'finish.log'
        is located in /var/sadm/system/logs after reboot.

Launching installer. Please Wait...

Installing Additional Software
|-1%----------25%--------------50%--------------75%----------100%|

Pausing for 30 seconds at the "Summary" screen. The wizard will continue
to the next step unless you select "Pause". Enter 'p' to pause. Enter 'c'
to continue. [c]

Creating ram disk for /a
updating /a/platform/i86pc/amd64/boot_archive...this may take a minute
updating /a/platform/i86pc/boot_archive...this may take a minute

syncing file systems... done
rebooting...
```

5. Modify the `solaris-pv.py` configuration file to avoid having to run the Solaris Installation Process again. Start by uncommenting the `root=` line.

```
# This is necessary after install to avoid a fatal failure to
# boot with the message "Cannot mount /devices."
root = "/dev/dsk/c0d0s0"
```

6. Comment out the `root=` , and `file:` lines as shown below.

```
# - Guest domain using a zvol (see zfs(1M)) as a disk image and
#   installing Solaris from an ISO image file:
#disk = ['file:/xen/isos/82-nd.iso,6:cdrom,r'
        #'phy:/dev/zvol/dsk/xen/domU-master,0,w']
        #'file:/xen/isos/solaris-pv.img,0,w']
```

7. Uncomment the `disk=` line as shown below.

```
# - Guest domain using a file as a disk image:
disk = ['file:/xen/isos/solaris-pv.img,0,w']
```

8. Start the virtual machine instance and answer the questions as shown below.

```
$ xm create -c solaris-pv.py
Using config file "./solaris-pv.py".
Started domain solaris-pv
SunOS Release 5.11 Version snv_82 64-bit
Copyright 1983-2007 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Hostname: winchester
Configuring devices.
Loading smf(5) service descriptions: 105/105
/dev/rdsk/c0d0s7 is clean

- Configure Keyboard Layout ---------------------------------------

  Please specify the keyboard layout from the list below.

  > To make a selection, use the arrow keys to highlight the option
    and press Return to mark it [X].

      Keyboard Layout
      -----------------------
  ^   [ ] Slovenian
  -   [ ] Slovakian
  -   [ ] Spanish
  -   [ ] Swedish
  -   [ ] Swiss-French
  -   [ ] Swiss-German
  -   [ ] Taiwanese
  -   [ ] TurkishQ
  -   [ ] TurkishF
  -   [X] UK-English
  -   [ ] US-English

----------------------------------------------------------------------
    F2_Continue    F6_Help
```

9. Login to the system when the login prompt appears.

```
winchester console login: root
Password:

Last login: Mon May 21 14:50:31 on console
May 21 15:00:46 winchester login: ROOT LOGIN /dev/console
Sun Microsystems Inc.   SunOS 5.11      matrix-build-2007-04-26
October 2007
#
```

Creating a Guest Domain Using the Command Line

The following example describes how to install Microsoft Windows in a guest domain using a local file as a root disk.

1. Run the `virt-install` command and specify that a guest domain running Microsoft Windows is to be created.

```
$ virt-install --name windowsxp-guest-HVM \
--ram 1024 --hvm --file /guest/windowsxp-guest-HVM-disk \
--cdrom /en_winxp_pro_with_sp2.iso --file-size 10 --vnc
```

   Note that:

   - `--name` is the name of the guest instance
   - `-- ram` specifies the amount of memory to allocate for the guest instance, in megabyte
   - `--hvm` identifies the guest as a hardware-assisted virtual machine
   - `--file` identifies the file to use as the disk image
   - `--cdrom` identifies the file to use as a virtual CD-ROM device for fully virtualized guests
   - `--file-size` specifies the size of the disk image (if it does not exist), in gigabytes
   - `--vnc` indicates that Virtual Network Computing (VNC) is to be used for graphics support

2. Start the guest domain using the `virsh start` command.

```
 $ virsh start windowsxp-guest-HVM
```

3. List the domains and verify the new guest domain is running.

```
$ virsh list --all
Id Name         State
----------------------------
0 Domain-0      running
1 windowsxp-guest-HVM running
```

Creating a Guest Domain Using Interactive Mode

The `virt-install` command can be executed to run in interactive mode. Information about a guest domain is entered as a series of responses to prompts.

1. Run the `virt-install` command.

```
$ virt-install
```

2. Respond to the series of prompts provided. The following information is needed in order to create the guest domain.

Information	Description
Name of guest domain	The name serves as the label of the guest operating system, and will be the name of the file that stores the guest domain's configuration.
Amount of RAM to allocate	Size must be specified in MB.
Path to disk image of the guest operating system	The path is exported as an entire disk to the guest domain. It resides on the control domain and can be a physical disk, a file, or a Solaris ZFS volume. If the path does not exist, specify the size (in GB) for the virtual disk for the guest domain.
URI for the location of the installation software for the guest operating system	The URI can be an NFS path, HTTP or FTP URL, or the path to a local directory or ISO.

3. Once the information is entered, installation begins. If graphics capabilities are enabled, a Virtual Network Computing (VNC) window opens and presents the graphical installer. If graphics capabilities are not enabled, the text installer displays.
4. Start the guest domain using the `virsh start` command.

```
$ virsh start windows1
```

5. List the domains and verify the new guest domain is running.

```
$ virsh list --all
Id Name         State
----------------------------
0 Domain-0      running
1 windows1      running
```

Accessing Virtual Block Devices and Block Storage

Device drivers for control domains in the Solaris OS are identical to drivers that run on bare metal hardware. As long as device drivers are written to standard Device Driver Interfaces (DDI), they should function identically on the control domain and standard

releases of the Solaris OS. Device drivers that work around the DDI may encounter issues when running on the control domain. For example, if a device driver attempts to use physical addresses directly rather than using the `ddi_dma_*_bind_handle()` interface, physical address to hardware address translations can be missed, resulting in the device being instructed to perform I/O to or from the wrong location. Such behavior may result in a crash of the control domain, or silent data corruption.

Device drivers may work in two different ways in the guest domain. In a fully virtualized guest domain, the Sun xVM hypervisor traps writes to I/O spaces and DMA operations. These requests are forwarded transparently to the appropriate device in the control domain. Such an environment can result in reduced performance, as trapping and forwarding requests can be expensive operations.

In a paravirtualized guest domain, each device driver consists of a front-end which resides in the guest domain, and a back-end which runs in the control domain. The front-end driver takes standard Solaris OS requests and forwards them to the back-end device driver, which executes the request on the physical hardware and passes results to the front-end driver. In turn, the front-end driver notifies the operating system of the completion of the request. Since the driver is aware of the hypervisor, the driver can work with the hypervisor and back-end driver to deliver faster performance than a driver in a fully virtualized environment.

The Sun xVM hypervisor includes a virtual block device and virtual network device. These two device drivers present the domU with standard disk and network interfaces.

Dynamically Allocating and Deallocating Virtual CPUs

The Sun xVM hypervisor enables virtual CPUs (VCPUs) to be dynamically allocated to, and deallocated from, virtual machines. Note that VCPUs can be dynamically allocated to both the dom0 and domUs, and a domU can have up to 32 VCPUs regardless of the number of real CPUs. Let's start by looking at how to dynamically add VCPUs to the control domain.

1.

```
# virsh setvcpus solaris-pv 10
```

2. Determine how many VCPUs are available to the virtual machine instance. In the following example, `solaris-pv` is the virtual machine instance name.

```
# virsh list
Name          ID    Mem   VCPUs   State   Time(s)
Domain-0      0     2247  2       r-----  1589.9
solaris-pv    4     1023  10      -b----  21.5
```

3. Obtain more detailed information about the current status of the VCPUs in the virtual machine instance by using the `virsh` command.

```
# virsh vcpuinfo solaris-pv
Name          ID    VCPU  CPU     State   Time(s)   CPU    Affinity
solaris-pv    8     0     1       -b-     39.6      any    cpu
solaris-pv    8     1     0       -b-     23.9      any    cpu
solaris-pv    8     2     0       -b-     24.2      any    cpu
solaris-pv    8     3     0       -b-     22.4      any    cpu
solaris-pv    8     4     0       -b-     22.6      any    cpu
solaris-pv    8     5     0       -b-     23.7      any    cpu
solaris-pv    8     6     1       -b-     23.7      any    cpu
solaris-pv    8     7     0       -b-     23.9      any    cpu
solaris-pv    8     8     0       -b-     23.8      any    cpu
solaris-pv    8     9     1       -b-     22.5      any    cpu
```

4. Check the status of the newly created virtual machine instance.

```
# uname -a
SunOS winchester 5.11 snv_82 i86pc i386 i86xpv

# psrinfo
0       on-line     since       05/21/2007      14:56:18
1       on-line     since       05/21/2007      14:56:26
2       on-line     since       05/21/2007      14:56:26
3       on-line     since       05/21/2007      14:56:26
4       on-line     since       05/21/2007      14:56:26
5       on-line     since       05/21/2007      14:56:27
6       on-line     since       05/21/2007      14:56:27
7       on-line     since       05/21/2007      14:56:27
8       on-line     since       05/21/2007      14:56:28
9       on-line     since       05/21/2007      14:56:28
```

The preceding steps configured 10 VCPUs for the virtual machine instance. The following steps explain how to reduce the number available to eight VCPUs.

1. Set the number of VCPUs to eight.

```
# virsh setvcpus solaris-pv 8
```

2. Check the current status of the VCPUs available to the virtual machine instance. Note that the two deallocated VCPUs report a powered off state.

```
# virsh vcpuinfo solaris-pv
Name          ID    VCPU  CPU     State   Time(s)   CPU    Affinity
solaris-pv    8     0     1       -b-     41.4      any    cpu
solaris-pv    8     1     0       -b-     24.2      any    cpu
solaris-pv    8     2     1       -b-     24.5      any    cpu
solaris-pv    8     3     1       -b-     22.6      any    cpu
solaris-pv    8     4     1       -b-     23.5      any    cpu
solaris-pv    8     5     0       -b-     24.0      any    cpu
solaris-pv    8     6     0       -b-     24.4      any    cpu
solaris-pv    8     7     0       -b-     24.1      any    cpu
solaris-pv    8     8     -       --p     23.9      any    cpu
solaris-pv    8     9     -       --p     22.6      any    cpu
```

3. Check the status of the virtual machine instance.

```
# uname -a
SunOS winchester 5.11 snv_82 i86pc i386 i86xpv

# psrinfo
0       on-line       since     05/21/2007      14:56:18
1       on-line       since     05/21/2007      14:56:26
2       on-line       since     05/21/2007      14:56:26
3       on-line       since     05/21/2007      14:56:26
4       on-line       since     05/21/2007      14:56:26
5       on-line       since     05/21/2007      14:56:27
6       on-line       since     05/21/2007      14:56:27
7       on-line       since     05/21/2007      14:56:27
8       powered-off   since     05/21/2007      15:14:38
9       powered-off   since     05/21/2007      15:14:38
```

4. If a domain is rebooted or saved/restored, access to the powered off VCPUs is lost. The powered off VCPUs do not appear on the list. Use the `virsh` command to revert to the original 10 VCPU state.

```
# virsh setvcpus solaris-pv 10
```

5. Check the status of the virtual machine to see that all 10 VCPUs are configured.

```
# virsh vcpuinfo solaris-pv
Name          ID    VCPU  CPU     State   Time(s)   CPU    Affinity
solaris-pv    8     0     0       -b-     42.6      any    cpu
solaris-pv    8     1     1       -b-     24.5      any    cpu
solaris-pv    8     2     0       -b-     24.8      any    cpu
solaris-pv    8     3     1       -b-     22.9      any    cpu
solaris-pv    8     4     1       -b-     23.8      any    cpu
solaris-pv    8     5     0       -b-     24.3      any    cpu
solaris-pv    8     6     1       -b-     25.2      any    cpu
solaris-pv    8     7     0       -b-     24.3      any    cpu
solaris-pv    8     8     1       -b-     24.1      any    cpu
solaris-pv    8     9     0       -b-     22.6      any    cpu
```

Setting up the VCPUs for dom0 works differently. By default, a dom0 VCPU is assigned for each CPU on the system. This assignment can be changed via the configuration and `dom0-cpus` on `xend`.

1. Check the status of the virtual machine to see the VCPUs automatically assigned to dom0.

```
# virsh vcpuinfo 0
Name          ID    VCPU  CPU     State   Time(s)   CPU    Affinity
Domain-0      0     0     0       -b-     371.7     any    cpu
Domain-0      0     1     1       r--     280.4     any    cpu
```

2. Set the number of VCPUs to one and restart the control domain in order to make the new configuration active.

```
# svccfg -s system/xctl/xend setprop config/dom0-cpus = 1
# svcadm refresh system/xctl/xend
# svcadm restart system/xctl/xend
# virsh vcpuinfo 0
```

3. Verify the status of the VCPUs for the control domain.

```
# virsh vcpuinfo 0
Name          ID    VCPU  CPU     State   Time(s)   CPU    Affinity
Domain-0      0     0     1       r--     388.8     any    cpu
Domain-0      0     1     -       --p     290.3     any    cpu
```

Sun xVM Hypervisor Networking Infrastructure

The Sun xVM hypervisor provides three basic mechanisms that are used to implement inter-domain communication paths on a single physical system:

- The sharing of memory between two domains, where both domains may read and write to the shared area
- The transfer of ownership of memory between domains
- An event channel mechanism that is used to send and receive alert style notifications between domains

The Sun xVM hypervisor defines a set of inter-domain protocols built on these three mechanisms to provide the emulation of traditional disk, network, and console hardware interfaces. Currently, these inter-domain protocols operate between a single pair of domains (point-to-point). Connecting more than two domains is not an aim or feature of the protocols defined, and is considered a task for another component. The following sections discuss the inter-domain network protocol, its implementation in the Solaris 10 OS, and integration with other Solaris software.

Inter-Domain Networking

For a particular inter-domain network path, the Sun xVM hypervisor separates the peers into a back-end and a front-end. The back-end domain typically provides the front-end domain with access to some shared infrastructure. The shared infrastructure is managed by the back-end domain, and usually takes the form a physical network interface. In a typical deployment, the back-end of an inter-domain network path runs in domain 0, which has access to physical hardware. The front-end of the path runs in an unprivileged domain. As a result, domain 0 provides shared access to a physical network interface to the guest domains, which have no direct access to physical devices.

To provide an inter-domain path, the underlying facilities of the hypervisor are used to implement:

- A ring of transmit descriptors, stored in a shared page
- A ring of receive descriptors, stored in a shared page
- The exchange of packetized data through the transfer of page ownership
- An interrupt initiation and delivery mechanism for notification of changes to the transmit or receive descriptor rings

The transmit and receive descriptors should be familiar to those who studied the driver for a physical network card. When combined, the facilities provided by the hypervisor are used to model an Ethernet network. The network includes only two participants: the back-end and front-end domains. All packets transmitted by one domain are received by the peer.

General Facilities

When running on the Sun xVM hypervisor, the Solaris OS makes functions available to drivers that enable the creation, destruction, and management of the underlying hypervisor features and capabilities. For example, the inter-domain event channel mechanism is mapped into the existing Solaris OS interrupt model. Device drivers that need to respond to inter-domain event channel notifications do so by providing an interrupt handler. Sun continues to work on software packages, device drivers, and other network infrastructure software.

Guest Domains—domU

Unprivileged domains typically require access to shared infrastructure via a privileged domain. Therefore, these domains usually participate in the inter-domain protocol as a front-end. Given the desire to model an Ethernet network, it is natural to implement a driver for unprivileged domains using GLDv3 (PSARC/2004/571 PSARC/2005/396). The `xnf` driver does so, and provides a standard DLPI interface to the inter-domain communication channel. The driver implements the inter-domain protocol defined in the hypervisor using the facilities provided by the remainder of the operating system

kernel. Packets passed to the interface by higher layers (such as IP) are sent to the peer domain and are transmitted over the underlying network. Packets sent by the peer domain are passed to higher layers and are received by the network and processed accordingly. Note that the `xnf` driver can be used like other network drivers in the Solaris OS, and can be plumbed under IP, snooped, and more.

The Control Domain—dom0

The control domain, dom0, can access physical network hardware. As a result, dom0 usually participates in the inter-domain protocol as a back-end and provides access to some shared infrastructure for unprivileged domains. The back-end interface is modeled in two distinct ways by the Solaris OS: as a GLDv3 driver (`xnbu`), and as a layer over another GLDv3 driver (`xnbo`). While both models share the same underlying implementation of the inter-domain protocol (`xnb` misc module), they differ in how packets received from peers are treated, and the source of packets that are transmitted to peers.

The Back-end GLDv3 Driver—xnbu

Similar to the `xnf` front-end driver, `xnbu` provides a GLDv3 driver interface on top of the inter-domain communication channel. Packets passed to the interface by higher layers are sent to the peer domain via transmission over the underlying network. Packets sent by the peer domain are passed to higher layers and are received by the network and processed accordingly. As a result, the back-end and front-end pair can be used to emulate a private Ethernet segment between two domains. In such a configuration, multiple private Ethernet segments between multiple domain pairs can be used to provide a routed IP network.

If a software Ethernet bridge implementation is available, it can be used to join two or more of the distinct Ethernet segments into a larger bridged ethernet network. If this larger ethernet network is bridged together with one or more physical network interfaces, the unprivileged domains appear to have direct access to the physical network to remote systems. Configurations using a software Ethernet bridge are commonly deployed on Linux dom0 systems.

Layering Over a GLDv3 Driver—xnbo

Given the general aim of providing unprivileged domains access to back-end accessible infrastructure, an alternative approach is made available by the `xnbo` driver. In this case, access to a network interface that is present in the back-end domain is extended into the unprivileged domain. To do this, the `xnbo` driver opens a GLDv3 driver in the back-end domain and passes any packets received by that driver to the peer (the front-end domain). Similarly, any packets received from the front-end domain are passed to the GLDv3 driver for transmission. In effect, the front-end domain owns the network interface present in the back-end domain.

- *Dedicated devices*
  If the GLDv3 driver opened by `xnbo` corresponds directly to a physical network device, the configuration of the `xnbo` instance and underlying device is termed *dedicated*. The network device is not available to traditional users in the back-end domain, with the exception of snoop. The MAC address of the physical network device is set to the MAC address assigned to the front-end domain.

- *Shared devices*
  Requiring a dedicated physical network interface for each guest domain is undesirable. Therefore, the Sun xVM hypervisor takes advantage of the concepts and implementation of network interface card virtualization from Project Crossbow. One of many endeavors undertaken via the OpenSolaris™ Project, Project Crossbow brings bandwidth resource control and virtualization into the operating system architecture, instead of adding on layered functionality with heavy overhead and undue complexity. Project Crossbow provides the building blocks for network virtualization and resource control by virtualizing the network stack and network interface card around any service protocol, such as HTTP, HTTPS, FTP, and NFS, or any virtual machine, including the Sun xVM hypervisor. Each virtual stack can be assigned its own priority and bandwidth on a shared NIC without degrading performance. The architecture dynamically manages priority and bandwidth resources, and can provide a better defense against denial of service attacks directed at a particular service or virtual machine by isolating the impact to that entity. All virtual stacks are separated by a hardware classification engine that ensures traffic for one stack does not impact other virtual stacks.

  A layer is added to the control path, enabling the creation of multiple virtual NICs on top of the same physical MAC device. A VNIC acts like a regular MAC device—it can be assigned to a Solaris Zone or stack instance, IP interfaces can be plumbed on it, and it can be assigned to a virtual machine with a separate address space co-existing on the host. The implementation of VNICs provides the ability to segment a single physical network interface into multiple virtual network interfaces, where all virtual network interfaces share access to the underlying network medium. The VNICs provide a GLDv3 interface to consumers, enabling the `xnbo` driver to use them without modification.

  More information on the network virtualization and resource control capabilities of Project Crossbow can be found on the OpenSolaris Project Web site located at *http://opensolaris.org/os/project/crossbow*

Integration into the Sun xVM Hypervisor Toolset

The Sun xVM hypervisor toolset enables an application to be run to prepare network interfaces in back-end domains before they are made available to front-end domains. Similarly, when the front-end portion of a network connection disappears (because the domain failed or was migrated), an application is run that can free assigned resources.

This mechanism is used by the Solaris implementation to ensure that VNIC creation and destruction take place as appropriate. When the shared device model is in use, VNICs are created and destroyed as required.

Chapter 6
Migration of Virtual Machine Instances

The Sun xVM hypervisor allows the migration of virtual machine instances between servers with a similar hardware architecture. A running operating system instance and its applications can be migrated as a single unit. When migrating a domain to another host machine, the following conditions must be mat on the target host machine:

- The target host must be running the same version of the Sun xVM hypervisor.
- The migration TCP port must be open and accepting connections from the source host.
- There must be sufficient resources, such as memory and disk space, for the domain to run.

Note – Note: Use of the `migrate --live` command migrates the domain between hosts without shutting down the domain.

Migration

Migrating a domain to a separate machine is a simple extension of the normal save and restore process. The save operation replaces machine-specific frame values with pseudo-physical frames, enabling the restore to take place on a remote machine—even though the actual hardware pages given to the domU are different. In order to do so, the Sun xVM hypervisor daemon on the remote machine must be listening on the HTTP port. The following command is used to perform the migration operation. Prior to the migration, the `xend` daemon stops the domain. The domain remains down during the migration process.

```
# xm migrate xvm-1 remotehost
```

Instead of writing the content of each page to a file, the software transmits the content across HTTP to the Sun xVM hypervisor daemon running on a remote machine. The restore is performed on the remote machine in the same manner—the data for each page is written into one of the machine addresses reserved for the new domain. When a saved page table is written, the memory page value for each PTE is replaced with the new machine address value used by the new instance of the domain.

Eventually, control is given back to the restored domain and it comes out of the `HYPERVISOR_suspend()` call. At this point, the event channel setup is rebuilt, and anything torn down before suspending is resumed. Finally, the software returns from the suspend handler and operation continues.

Live Migration

Live migration enables a domain to be kept running during most of the migration process. The live migration process relies on the assumption that an operating system instance is unlikely to modify a large percentage of its pages within a certain time frame. By iteratively copying over modified domain pages, it is possible to reach a point where the remaining data to be copied is small enough that the actual downtime required to migrate a domU is minimal. The following command is used to perform a live migration.

```
# xm migrate --live solarisxvm1 solarisxvm2
```

Benefits of Live Migration

The ability to move a domain and its applications from one physical machine to another with minimal downtime provides a host of benefits, including:

- Eliminates the need for users and clients to re-connect to the new virtual machine instance, simplifying the user experience and maintaining uptime
- Enables virtual machines to be moved as needed, facilitating load balancing
- Moves data and services quickly, easing system management for the originating host once migration is complete

The Live Migration Process

In operation, the domain is switched to use a modified form of the shadow page tables, known as *log dirty* mode. Typically, full virtualization uses a method known as *shadow page tables*, in which two sets of page tables are maintained. Guest domain page tables are not visible to the hardware via `cr3`, and page tables visible to the hardware are maintained by the hypervisor. Since only the hypervisor can control the page tables the hardware uses to resolve translation lookaside buffer (TLB) misses, it can maintain the virtualization of the address space by copying and validating changes the guest domain makes to its copies into the real page tables.

Duplicate pages come at a price. A paravirtualization approach—one in which the guest domain is aware of the virtualization and complicit in operating within the hypervisor—can take a different tack. In the Sun xVM hypervisor, the guest domain is made aware of a two-level address system. The domain is presented with a linear set of pseudo-physical addresses comprising the physical memory allocated to the domain, as well as the machine addresses for each corresponding page. The machine address for a page is used in the page tables (the real hardware address).

Two tables are used to map between pseudo-physical and machine addresses. A shadow page table is used to notify the hypervisor when a page is written to, by keeping the PTE entry for the page read-only. An attempt to write to the page causes a page fault. The page fault is used to mark the domain page as dirty in a bitmap

maintained by the hypervisor, which then fixes up the domain's page fault and enables it to continue.

Meanwhile, domain management tools iteratively transfer unmodified pages to the remote machine. The dirty page bitmap is read, and any pages that were modified since the last transmission are re-sent. This process continues until the software can tell the domain to suspend, and switch over to running the domain on the remote machine. While page transmissions take some time, the time between service suspension and resumption typically is short.

More information on the live migration process and how it works can be found in *Live Migration of Virtual Machines* located at *http://www.cl.cam.ac.uk/research/srg/netos/papers/2005-migration-nsdi-pre.pdf*

Observing the Live Migration Process

Because the migration happens quickly, it is difficult to observe the process. Administrators and users can run the `xm list` command on both machines and see the domain appear on the remote host (and no longer appear on the original host) once the migration is complete. In addition, a system message is displayed about the migration on SMP guest domains.

Chapter 7
Troubleshooting

The chapter presents several topics that may prove helpful when diagnosing issues during the installation, configuration, or deployment of virtualized environments.

Differences from a Standard Environment

The Sun xVM hypervisor introduces two different platforms for Intel processor-based systems: i86pc and i86xpv. The i86pc platform refers to the Solaris OS running on bare metal hardware, while the i86xpv platform refers to the Solaris OS running paravirtualized on a hypervisor. The platforms are more similar than they are different.

- Code specific to the i86xpv platform can be found in the `usr/src/uts/i86xpv` directory. Note that the bulk of the code used to implement the i86xpv platform is taken directly from the i86pc sub-tree located at `usr/src/uts/i86pc`.
- Several publicly available header files are needed to build the i86xpv platform code. These headers can be found in the `common/xen/public` directory.
- Code that is meant to apply only to i86xpv platforms should be protected with `#ifdef__xpv` statements.
- Users and applications should be unaware of the platform on which they are running. Complete binary compatibility between the i86pv and i86xpv platforms exists at the user level. The platform in use can be obtained using the `uname –i` command.

Understanding Resource Virtualization

Developers may need to understand how processors and memory are virtualized and referenced by the Sun xVM hypervisor software.

- *Processor virtualization*
  The Sun xVM hypervisor assigns domains to one or more virtual CPUs (VCPUs). Each VCPU contains all the state information typically associated with a physical CPU, such as registers, flags, and timestamps. VCPUs can be scheduled, just like a thread in the Solaris OS. When it is a domain's turn to run on a CPU, the Sun xVM hypervisor loads the physical CPU with the state captured in the VCPU and lets it run. The Solaris OS treats each VCPU exactly as it would a physical CPU. Indeed, as far as the Solaris OS is concerned, a VCPU is a physical CPU. When the Sun xVM hypervisor selects a VCPU to run, it runs the thread that the Solaris OS loaded on the VCPU.

- *Memory virtualization*
  The Solaris OS manages memory in pages. When running directly on hardware, the Solaris OS uses the physical page frame numbers provided by the system hardware. When running under the Sun xVM hypervisor, the hypervisor provides the list of available physical memory pages. In order to isolate a guest operating system from the hardware, and to enable features such as live migration, the memory page numbers provided by the Sun xVM hypervisor do not reflect the physical pages as understood by the hardware. Instead, the pages reflect a virtualized view of system memory. The Sun xVM hypervisor maps the guest operating system's physical page numbers to the machine frame numbers recognized by the hardware. In general, the Solaris OS manages the page frame numbers provided by the Sun xVM hypervisor in the same way it does those used on real hardware.

Crash Dumps

On a running system, the hypervisor's memory is completely inaccessible to `dom0`. However, if the hypervisor crashes, the resulting panic dump generates a core file that provides a unified view of both the Sun xVM hypervisor and `dom0`. In such a core file, the Sun xVM hypervisor appears as a simple Solaris kernel module called `xpv`. For example:

```
> $c
xpv`panic+0xbf()
xpv`do_crashdump_trigger+0x19()
xpv`keypress_softirq+0x35()
xpv`do_softirq+0x54()
xpv`idle_loop+0x55()
```

If a `dom0` crashes with a standard Solaris OS panic, the dump includes just the `dom0`. It is only when the hypervisor panics that the resulting dump includes the virtual machine state as well.

For more information about handling panics, as well as an example of debugging a panic, see *http://blogs.sun.com/nilsn/resource/xen_solaris_panic.pdf*

Observability

Although the hypervisor and control domain work closely together to manage a running system, the dom0 operating system has little direct visibility into the hypervisor. The hypervisor's entire address space is inaccessible to the control domain. As a result, utilities that communicate with the hypervisor via hypercalls are needed to gain information. Table 7-1 identifies some useful commands.

Table 7-1. Some utility commands

Command	Description
virsh capabilities	Return the capabilities of the hypervisor.
virsh dominfo *domain-name*	Return basic information about a domain
virsh domstate *domain*	Obtain the state of a running domain
virsh dump *domain-name file*	Dump the core of the domain to a file for analysis
virsh list	Display information about one or more domains
virsh nodeinfo *domain-name*	Get basic information about a node
virsh schedinfo *domain-name*	Show or set the scheduling parameters for a domain
virsh vcpuinfo *domain-name*	Obtain basic information about the domain virtual CPUs
virsh version	Display version information about the virsh instance
xm info	Report static information about the machine, such as the number of processors, total memory, software version number, and more
xm list	List all domains and high-level information
xm top	Report domain info (similar to the Linux top command, but focused on domains rather than processes)
xm log	Display the contents of the xend log
xm help	List all available commands

Dynamic Tracing

Understanding how systems are performing and why is key to creating stable infrastructures. The Solaris Dynamic Tracing (DTrace) facility is a dynamic tracing framework for troubleshooting systemic problems in real time on production systems. Designed to quickly identify the root cause of system performance problems, DTrace gives developers insight how a system is performing — without having to configure separate test systems or create lengthy, customized testing scripts.

Due to the isolation of the hypervisor from `dom0`, currently there is no way to apply DTrace directly to the hypervisor. A new `xpv` DTrace provider enables developers to trace the interaction between `dom0` and the hypervisor. This provider is constructed of SDT probes introduced into the `privcmd` device driver. The available probes may be listed by using the `dtrace -l -i` command. While understanding the details of these probes requires a fair bit of knowledge about the Sun xVM hypervisor interface, simply enabling the probes using the `dtrace -n 'xpv::: {}` command provides a quick, high-level introduction to the steps involved with creating, destroying and migrating domains, and more.

For example, after the `dtrace -n 'xpv::: {}` command is run, the `xm destroy` operation triggers the following probes.

```
CPU     ID        FUNCTION:NAME
 1      6315      privcmd_HYPERVISOR_domctl:dom-destroy-start
 1      6312      privcmd_HYPERVISOR_domctl:dom-destroy-end
```

Similarly, the `xm create` *`domain`* operation results in the following probes.

```
CPU     ID        FUNCTION:NAME
 1      6314      privcmd_HYPERVISOR_domctl:dom-create-start
 1      6313      privcmd_HYPERVISOR_domctl:dom-create-end
 1      6298      privcmd_HYPERVISOR_event_channel_op:evtchn-op-start
 1      6297      privcmd_HYPERVISOR_event_channel_op:evtchn-op-end
 1      6298      privcmd_HYPERVISOR_event_channel_op:evtchn-op-start
 1      6297      privcmd_HYPERVISOR_event_channel_op:evtchn-op-end
 1      6308      privcmd_HYPERVISOR_memory_op:set-memory-map-start
 1      6299      privcmd_HYPERVISOR_memory_op:set-memory-map-end
 1      6304      privcmd_HYPERVISOR_memory_op:populate-physmap-start
 1      6301      privcmd_HYPERVISOR_memory_op:populate-physmap-end
 1      6325      do_privcmd_mmap:mmap-start
 1      6324      do_privcmd_mmap:mmap-entry
 1      6323      do_privcmd_mmap:mmap-end
[...]
 1      6325      do_privcmd_mmap:mmap-start
 1      6324      do_privcmd_mmap:mmap-entry
 1      6323      do_privcmd_mmap:mmap-end
 1      6318      privcmd_HYPERVISOR_domctl:setvcpucontext-start
 1      6309      privcmd_HYPERVISOR_domctl:setvcpucontext-end
 1      6317      privcmd_HYPERVISOR_domctl:dom-unpause-start
 1      6310      privcmd_HYPERVISOR_domctl:dom-unpause-end
```

Chapter 8

Hardware-Assisted Virtualization

This chapter provides an overview of the considerations and steps needed to install unmodified guest operating systems in a hardware virtual machine on systems running the Sun xVM hypervisor.

Installing Unmodified Operating Systems

Once the control domain is installed and running, a virtual disk and hardware virtual machine (HVM) configuration file must be created in order to get an HVM domU running. The following steps outline the procedure for creating a virtual disk and the HVM configuration file for a Microsoft Windows HVM.

1. Using a `lofi(7D)` backed file is an easy and high-performance method for obtaining a disk for use by a domain. It uses a qemu raw image file. Use the `qemu-img` utility to create the disk. The command shown below creates a 10 GB qemu disk image, which consumes only the space needed to store information written to it. Note that the `qemu-img` utility is available for download from Blastwave.

```
# /opt/csw/bin/qemu-img create -f raw windows.raw 10G
Formating 'windows-disk.raw', fmt=raw, size=10485760 kB
```

2. Create a configuration file for the HVM domain. The following example configuration file directions and contents can be used as the basis for creating an HVM configuration file. Some parameters may need modification based on a given environment. In general, most parameters are identical to those used for Solaris OS domains.

```
# Set the amount of memory allocated to your domain:
memory = 512

# Give your domain a name:
name = "Windows-on-Solaris"

# Give your domain a virtual network interface by uncommenting
# the vif line (optional):

# Optionally define mac and/or bridge for the network interfaces.
# Random MACs are assigned if not given.
vif = [ 'type=ioemu' ]

# If you need to specify a specific mac address, it would look like this:
vif = [ 'type=ioemu,mac=00:bb:cc:dd:ee:ff' ]

# Be careful not to get too random with the mac address - particularly,
if the least significant bit of the first hex pair is set, # it becomes
a multicast ethernet address, which is almost certainly not what you
want.)

# Assign virtual disks to your domain.

# For block devices, use the phy: prefix; for files, use the file:
# prefix, which will cause them to be automatically lofi-mounted. Each
# disk must have a unique hdx identifier, where 'x' is the drive letter
# (starting with a, b, c, etc.). The :cdrom tag after the drive
# intentifier means that the disk will be presented as a CD or DVD drive.

# For example, if you have created a qemu image at
# /xvm/isos/windows-disk.raw and the installer .iso for the Operating
# System you are trying to run is at /xvm/isos/winxp-sp2.iso, your disk
# line should look like this:

disk = [ 'file:/xvm/isos/windows-disk.raw,hdc,w', 'file:/xvm/isos/winxp-
sp2.iso,hda:cdrom,r' ]

# You can use the physical CD drive on your dom0 machine, for example:
disk = [ 'file:/xvm/isos/windows-disk.raw,hdc,w', 'phy:/dev/dsk/
c2t0p0,hda:cdrom,r' ]

# Set the boot device. This option instructs the virtual BIOS to boot off
# of the CDROM ('d'). To boot off the drive, use 'c'.
boot='d'

# Choose a graphical console. VNC and libSDL are your choices.
# libSDL uses X11 to display a window containing the virtual machine's
# graphics console on your local X desktop. You must make sure that the
# remote X11 display to your desktop is functioning (that DISPLAY is
# correctly set, that xhost permits the remote host, and that the X
# server is configured to accept remote connections. Test this by
# remotely displaying an xterm to your desktop.

# VNC has the usual advantage of allowing a client to disconnect and
# reconnect at will without destroying the session. If you close a libSDL
# console window, you cannot get it back and you'll have to restart your
# domain (and the domain might not continue to run).
```

```
# To use libSDL, set the option in the config file:
#---------------------------------------------------------------------
# enable SDL library for graphics, default = 0
sdl=0

#---------------------------------------------------------------------
# enable VNC library for graphics, default = 1
vnc=1

#---------------------------------------------------------------------
# address that should be listened on for the VNC server if vnc is set.
# default is to use 'vnc-listen' setting from /etc/xvm/xend-config.sxp
vnclisten="0.0.0.0"

#---------------------------------------------------------------------
# set VNC display number, default = domid
vncdisplay=1
```

Chapter 9
For More Information

About the Authors

Michael Haines is a Senior Engineer at Sun. Since joining Sun in 1989, Michael has worked in various engineering roles, with emphasis on the Solaris Operating System, naming services, and more recently virtualization technologies. Michael is also the co-author of two Sun BluePrints books: *LDAP in the Solaris Operating Environment,* and *Solaris and LDAP Naming Services*, and has written many other Sun publications.

David Edmondson is a Senior Staff Engineer in the Solaris Engineering organization at Sun. Since joining Sun eight years ago, David has worked mostly in the areas of networking capability and performance. David is the engineering lead for the networking components of Sun xVM Server, having spent the last three years focusing on virtualization support in the Solaris Operating System.

Acknowledgments

The authors wish to thank the following individuals for their contributions to this article:

- John Levon
- Mark Johnson
- Ryan Scott
- Ed Pilatowicz
- Ryan Scott
- Joe Bonasera
- Russ Blaine
- Chris Beal

References

General Hypervisor Information
`http://en.wikipedia.org/wiki/Hypervisor`

OpenSolaris Project
`http://www.opensolaris.org`

“Live Migration of Virtual Machines”
`http://www.cl.cam.ac.uk/research/srg/netos/papers/2005-migration-nsdi-pre.pdf`

“Solaris Operating System Hardware Virtualization Product Architecture”
`http://www.sun.com/blueprints/1107/820-3073.html`

"How to Install Sun xVM hypervisor and Use it to Configure Domains"
`http://www.sun.com/software/solaris/howto_guides.jsp`

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`

Appendix A
Glossary

Back-end driver
The portion of a virtual device driver that provides an interface between the virtual device and an underlying physical device.

Control domain
An instance of the Solaris OS modified to run under the hypervisor. Called dom0, the control domain controls all guest operating systems that run on the system. The control domain is the only domain given direct access to hardware by the hypervisor.

Domain
A virtual machine instance.

Front-end driver
The portion of a virtual device driver that communicates with the back-end driver hosted in a different guest domain in order to process requests.

Full virtualization
A virtual machine approach which emulates a machine architecture down to the register level, enabling the execution of unmodified guest operating systems.

Guest domain
A virtual machine instance running on a system. Because a guest domain is unprivileged, it is referred to as a domU.

GRUB
The Grand Unified Bootloader for the Solaris OS. Developed and supported by the Free Software Foundation, GRUB is a boot loader that gets the machine started and transfers control to the operating system kernel, which in turn initialized the remainder of the operating system.

Hard partitioning
A virtualization technique in which physical CPU, memory, and I/O resources are assigned to specific domains which run independent operating system instances.

Hypervisor
Software which makes it possible to partition compute resources on conventional hardware in a safe and effective manner without sacrificing performance or function.

Operating system virtualization
Technology which creates many private execution environments within a single instance of an operating system. By using flexible, software-defined boundaries, virtual operating system environments are independent of the hardware layer and are available for all platforms that support the operating system.

paravirtualization
A virtual machine architecture which increases efficiency by directly integrating the hypervisor with a paravirtualized operating system. Guest operating systems call support functions implemented in hypervisor code rather than trying to access a system register.

Resource management
A virtualization technique which leverage operating system controls to govern the utilization of processors, memory, and I/O resources.

VPCU
Virtual CPU.

Virtual machine monitor
Software which enables multiple, different operating systems to run concurrently as guests on a single physical machine.

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 05/08