PRENTICE HALL

Xen

CiTRIX

Running Xen

A Hands-On Guide to the Art of Virtualization

Jeanna N. Matthews • Eli M. Dow
Todd Deshane • Wenjin Hu • Jeremy Bongio
Patrick F. Wilbur • Brendan Johnson

Foreword by Ian Pratt, Xen Project Leader
VP Advanced Technology, Citrix Systems

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales
international@pearson.com

Visit us on the Web: www.informit.com/ph

Library of Congress Cataloging-in-Publication Data:

Matthews, Jeanna N.

Running Xen : a hands-on guide to the art of virtualization / Jeanna N. Matthews, Eli M. Dow, Todd Deshane, Wenjin Hu, Jeremy Bongio, Patrick F. Wilbur, Brendan Johnson.

p. cm.

ISBN 0-13-234966-3 (pbk. : alk. paper) 1. Xen (Electronic resource) 2. Virtual computer systems. 3. Computer organization. 4. Parallel processing (Electronic computers) I. Title.

QA76.9.V5M38 2008

005.4'3--dc22

2007052439

Copyright © 2008 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax (617) 671 3447

ISBN-13: 978-0-132-34966-6
ISBN-10: 0-132-34966-3

Text printed in the United States on recycled paper at Courier Stoughton in Stoughton, Massachusetts.
First printing April 2008

This Book Is Safari Enabled

The Safari® Enabled icon on the cover of your favorite technology book means the book is available through Safari Bookshelf. When you buy this book, you get free access to the online edition for 45 days.

Safari Bookshelf is an electronic reference library that lets you easily search thousands of technical books, find code samples, download chapters, and access technical information whenever and wherever you need it.

To gain 45-day Safari Enabled access to this book:

- Go to http://www.informit.com/onlineedition
- Complete the brief registration form
- Enter the coupon code 9FJZ-X7UK-PMSU-TAKZ-N6SI

If you have difficulty registering on Safari Bookshelf or accessing the online edition, please e-mail customer-service@safaribooksonline.com.

Editor-in-Chief
Mark Taub

Acquisitions Editor
Debra Williams Cauley

Development Editor
Michael Thurston

Managing Editor
Gina Kanouse

Project Editor
Chelsey Marti

Copy Editor
Geneil Breeze

Indexer
Erika Millen

Proofreader
Meg Shaw

Technical Reviewers
Jessie Yu
Ken Hess
Jose Renato Santos
Andrew Warfield
Tom "Spot" Callaway
Simon Crosby
Tom Doeppner
Dan Kuebrich
Jim Owens
Zach Shepherd
Keegan M. Lowenstein
Ryan Kornheisl
Igor Hernandez
Alexander M. Polimeni
Erika Gorczyca
Justin Bennett
Joseph Skufca
Mathew S. McCarrell
Krista Gould
Ron Arenas

Publishing Coordinator
Kim Boedigheimer

Cover Designer
Alan Clements

Senior Compositor
Gloria Schurick

Foreword

The Xen open source hypervisor is changing the world of virtualization. It encourages the broad distribution of a common industry standard hypervisor that runs on a wide range of architectures from super computers to servers to clients to PDAs. By focusing on the hypervisor, the "engine" of virtualization, rather than a specific product embodiment, the Xen open source project enables multiple vendors and the community to combine the common cross platform virtualization features of Xen into exciting new products and service offerings.

To date, the community around the Xen hypervisor has been squarely in the camp of developers and expert users. While the Xen-users mailing list offers a friendly and useful source of advice for those wanting to deploy and manage Xen-based environments, the new user might find herself in need of advice about best practice and step-by-step instructions for the deployment of Xen. *Running Xen: A Hands-on Guide to the Art of Virtualization* speaks directly to this critical need. It provides users with everything they need to know to download, build, deploy, and manage Xen implementations.

To the authors, a set of Xen contributors, practitioners, and researchers, I would like to say thank you on behalf of the broader Xen community for an accessible and immediately useful book. Code might rule, but "know-how" builds the community itself. Clear information, advice, and documentation like this book will allow the Xen project to grow and solidify its user base, to renew its creativity and innovation, to focus itself on a larger set of new virtualization initiatives.

To the readers, I would like to say welcome to the community of Xen users. We look forward to your involvement and contributions! We believe this book will provide you with an excellent introduction to running Xen.

Ian Pratt, Xen Project Leader
VP Advanced Technology, Citrix Systems

Preface

We began using Xen in the fall of 2003 soon after reading the paper "Xen and the Art of Virtualization" published in the Symposium on Operating Systems Principles (SOSP). After attending SOSP and talking to some of the authors, Jeanna Matthews returned excited about Xen. She and her graduate operating systems course at Clarkson University decided to repeat and extend the results reported in that paper. That class included two of the coauthors for this book, Eli Dow (currently at IBM) and Todd Deshane (currently completing his Ph.D.), who were both studying for their Master's degrees at the time. In the process of repeating the results from the 2003 Xen paper, we learned a lot about running Xen—much of it the hard way! Our goal for this book was to write exactly the material we wished was available when we first started using Xen.

In July 2004, we published the paper "Xen and the Art of Repeated Research," describing our experience with Xen and presenting the results we obtained repeating and extending the results. All the authors, in addition to being a part of the Fall 2003 graduate operating systems course, were also members of the Applied Computing Laboratories at Clarkson University, specifically the Clarkson Open Source Institute (COSI) and the Clarkson Internet Teaching Laboratory (ITL). These labs were founded to provide students with hands-on experience with cutting-edge computing technologies and to form a community in which everyone both learns and teaches. Other students in the labs—both graduate and undergraduate—began to use Xen as the basis for both production systems and for research projects. Through the years, we have used Xen as the basis for a number of academic papers as well as the basis of award-winning team projects. In the process, we have learned a lot about running Xen. It is our goal in this book to share this knowledge with you and to make your experience running Xen as smooth and simple as possible.

The book is targeted at individuals and organizations that are deploying Xen systems. It walks the reader through the basics, from installing Xen to using prebuilt guest images. It even tells readers how to experiment with Xen using only a Xen LiveCD. It covers the basics of virtualizations and important elements of all Xen systems like the hypervisor and Domain0. It explains the details of the `xm` commands for managing guest domains. It helps users deploy custom guest images based on operating systems from Linux to Windows. It covers more advanced topics such as device virtualization, network configuration, security, and live migration. We hope you will find it a good mix of introductory and advanced topics that will prove useful from your first Xen deployment experiment to running production Xen systems.

Chapter 1, "Xen—Background and Virtualization Basics," is a quick introduction to virtualization in general and to Xen in particular. Chapter 2, "A Quick Tour with the Xen LiveCD," provides an overview of Xen's functionalities by exploring the Xen LiveCD. Chapter 3, "The Xen Hypervisor," focuses on the hypervisor that is the core of any Xen system and some other trusted components such as Domain0 and `xend`. We build on that common understanding of the Xen hypervisor by concretely showing you how to install and configure your own hard-disk-based Xen installation in Chapter 4, "Hardware Requirements and Installation of Xen Domain0." After you have your own hypervisor installation up and running, this book eases you into using guest images by first showing you how to download and use images available from the Internet in Chapter 5, "Using Prebuilt Guest Images." Chapter 6, "Managing Unprivileged Domains," covers the basics of administering the running DomUs or unprivileged guest domains. You are then guided through the various methods of creating your own custom guest images in Chapter 7, "Populating Guest Images." Now that you have all these guests, Chapter 8, "Storing Guest Images," covers a variety of choices for storing guest images for online use as well as backup and sharing.

The second half of this book delves into more advanced system management topics including device management (Chapter 9, "Device Virtualization and Management"), networking (Chapter 10, "Network Configuration"), security (Chapter 11, "Securing a Xen System"), resource distribution (Chapter 12, "Managing Guest Resources"), and migration (Chapter 13, "Guest Save, Restore and Live Migration"). We conclude with a survey of some of the popular administrative tools available for your Xen systems in Chapter 14, "An Overview of Xen Enterprise Management Tools."

Throughout the book, we include listings illustrating relevant commands and their output. We use the command prompt to indicate where the command should be run.

For example, the following would indicate a command to be run as root on the privileged domain, Domain0:

```
[root@dom0 ]#
```

The following would indicate a command to be run as any user in a regular guest domain:

```
[user@domU]$
```

Watching these command prompts will help you identify which of the many guests in your Xen system should be used for running any given command.

It is our intention to maintain a website with additional information and materials relevant to the book. We have registered the domain, runningxen.com, for this purpose and are working on assembling materials. We invite you to check on our progress and to send questions or suggestions.

Chapter 13

Guest Save, Restore, and Live Migration

In this chapter, we begin by exploring Xen's capability to easily checkpoint the state of a guest domain to disk for quick restoration at a later time. We continue by exploring how Xen makes the migration of guest domains a simple and powerful administrative task. We discuss cold static migration, warm static migration, and live migration of guests, along with the prerequisites and benefits of each.

Representing the State of a Virtual Machine

At the heart of any migration is the ability to fully represent the state of a guest. When a guest virtual machine is completely shut down, this is trivial. An inactive Xen guest is completely defined by its file system image(s), configuration file, and its operating system kernel. Clearly, a guest could be cloned or even moved to another physical machine by making copies of these files. Backup of a guest can be accomplished in this way.

A guest that is active in execution, on the other hand, is a more complicated matter. While a guest is running, saving its state additionally involves creating a snapshot of its memory, device I/O states, open network connections, and the contents of its virtual CPU registers. Xen can save this state information to disk or transfer it over the network, which allows for both backup and migration of VMs.

This idea of saving the state of a running guest is similar to the hibernation feature on many personal computers, which is especially popular among laptop users. In hibernation, a system's state is checkpointed and saved to disk so that the system can park the hard drive heads, power down, and resume its previous state next time it is powered up. Laptop users sometimes rely on this feature to temporarily suspend the state of the machine when moving from place to place, or for conserving battery power when the laptop is not being used. In the case of a virtual machine monitor like Xen, a similar facility can be used to checkpoint states to facilitate rollback in the event a guest fails, or to save the state of a guest past the shutdown of the physical machine on which it is running.

Xen provides a domain save and restore facility to handle the suspension of guest VMs to checkpoint files, operated by the `xm save` and `xm restore` commands. When a guest is saved to disk it is suspended, and its resources are deallocated. As in the case of hibernate, ongoing network connections are not preserved.

With the `xm migrate` command, Xen supports warm static migration (regular migration), where a running guest is temporarily suspended and then relocated to another physical host, as well as live migration, where a guest may be relocated from one host to another seamlessly, without dropping ongoing network connections and with little client perceptible delay. Live migration is particularly useful when bringing down a physical machine for maintenance. In this case, guests can be relocated to a new physical machine in preparation for the maintenance without a disruption in service. The ability to relocate a guest is also useful for load balancing guests and their resource consumption.

In this chapter, we discuss the uses of `xm save`, `xm restore`, and `xm migrate` in detail.

Basic Guest Domain Save and Restore

Xen makes it possible to suspend a guest domain, save its state to a file, and resume that state later through its domain save and restore facility. Figure 13.1 illustrates the process of saving a guest's state to disk. As with hibernation, when a domain's state is saved on disk, it is suspended, and network connections to and from that guest are interrupted (due to TCP timeouts).

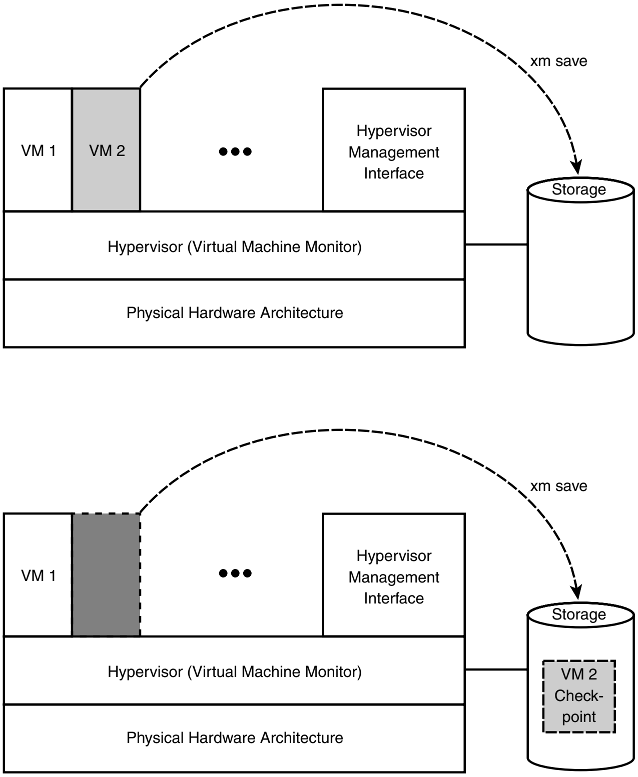

xm save
VM 1
VM 2
Hypervisor Management Interface
Storage
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture
xm save
VM 1
Hypervisor Management Interface
Storage
VM 2 Check-point
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture

FIGURE 13.1 Xen provides an easy-to-use facility for hibernating guests to checkpoint files, to be restored at a later point in time.

`xm save`

Saving a guest VM's state first suspends that guest VM and then dumps its state information to disk. The guest VM will not continue to execute until it is restored, much like in the case of hibernation.

Listing 13.1 shows the syntax of this command.

LISTING 13.1 Usage of `xm save`

```
xm save domain filename
```

When performing an `xm save`, an administrator supplies a filename argument to specify where the serialized state of the guest VM should be written. This file is colloquially called a *checkpoint file*. This does not erase any previously saved checkpoints. You are free to save as many distinct checkpoints as you want. Thus it is possible to archive a collection of different running states for a particular guest domain.

Either a guest domain ID number or domain name may be supplied as the `domain` argument. After running this command, Xen suspends the state of the guest-specified domain to the specified file, and the domain no longer executes on the host. Thus it is impossible to save the state of Domain0 in this fashion because the Xen Domain0 must remain operational as the control interface between the administrator and Xen.

In Listing 13.2, we show the output of `xm list` running in Domain0. The output shown illustrates we are running a guest domain named TestGuest that has 64MB of RAM allocated to it.

LISTING 13.2 Domains Running on Domain0 (Before Checkpointing)

```
[root@dom0]# xm list
Name                                ID Mem(MiB) VCPUs State   Time(s)
Domain-0                             0      511     2 r-----   1306.0
TestGuest                            1       63     1 -b----      4.5
[root@dom0]#
```

To save TestGuest's state, the corresponding ID number needs to be noted and passed as an argument to the `xm save` command. In Listing 13.2 we see that the ID that corresponds with TestGuest is 1, and an example suspension using the checkpoint file name `TestGuest.checkpt` is shown in Listing 13.3.

LISTING 13.3 Checkpointing a Guest's State Using `xm save`

```
[root@dom0]# xm save 1 TestGuest.checkpt
[root@dom0]#
```

Note that replacing 1 with TestGuest as an argument to `xm save` also works. We know that the state of TestGuest has successfully finished saving when it is no longer listed as residing on Domain0, which we check by invoking `xm list` as shown in Listing 13.4. The `xm save` command does not return immediately, but instead returns after checkpointing is complete.

LISTING 13.4 Domains Running on Domain0 (After Checkpointing)

```
[root@dom0]# xm list
Name                            ID Mem(MiB) VCPUs State   Time(s)
Domain-0                         0      511     2 r-----   1411.8
[root@dom0]#
```

We can observe that our checkpoint file now exists in the present working directory and determine its size by issuing the `ls -la` command as shown in Listing 13.5. The `ls -lah` command may instead be used to view file sizes that are more human readable.

LISTING 13.5 Checkpoint File

```
[root@dom0]# ls -la
-rwxr-xr-x 1 root root 67266796 Feb  6 03:54 TestGuest.checkpt
[root@dom0]#
```

Now that TestGuest has been suspended, is no longer executing, and has been saved to a state file, we have successfully used Xen's domain save facility. Note that if there is not enough disk space to save the checkpoint file, the guest remains running and the `xm save` command fails with a generic error message.

A checkpoint contains the contents of the entire memory state of the guest. Thus, the time required to save the guest's state is proportional to the amount of memory allocated for the guest. The size of the checkpoint file is also approximately the same as the amount of memory allocated to the guest VM, plus a small amount of extra disk space to store additional state information.

Listing 13.6 shows the output of `xm list` for a system with three sample guests—one with 64 megabytes, one with 128 megabytes, and one with 256 megabytes of RAM allocated. Table 13.1 shows the size of the checkpoint file and the total time taken to save each of these guests. In all three cases, the checkpoint file is slightly larger than the amount of memory allocated to the guest. It is also clear that the time taken to complete the save grows with the an increase in amount of allocated memory. The actual time taken to save the guest's state varies with the speed of the underlying file system and the hardware used to store the checkpoint file.

LISTING 13.6 Domains Running on Domain0 with Varying Memory Allocations

```
[root@dom0_Host1]# xm list
Name                            ID Mem(MiB) VCPUs State   Time(s)
Domain-0                         0      511     2 r-----   1306.0
Guest64MB                        7       63     1 -b----      3.1
Guest128MB                       8      127     1 -b----      2.8
Guest256MB                       9      255     1 -b----      2.1
[root@dom0_Host1]#
```

TABLE 13.1 Checkpoint File Size and Time Proportions

Actual Guest RAM Allocation (MB)	File Size On Disk (MB)	Time to Save Guest State (sec)
65.5	67.1	0.859
130.8	134.5	2.426
261.9	268.7	4.802

`xm restore`

Restoring a guest domain from a state file is initiated by the `xm restore` command. Listing 13.7 shows the syntax of this command.

LISTING 13.7 Usage of `xm restore`

```
xm restore filename
```

On the execution of this command, Xen restores the state of the guest located in the specified `filename`. The numerical domain ID of a guest domain is not preserved through save and restore, so no consideration needs to be taken to avoid an ID conflict—a unique ID is automatically assigned when restoring a guest from a checkpoint file.

On Domain0, we currently have a state file for a domain in the present working directory, observed by issuing the `ls` command, as shown in Listing 13.8.

LISTING 13.8 Checkpoint File

```
[root@dom0]# ls
TestGuest.checkpt
[root@dom0]#
```

Note that no guest domains are currently residing on Domain0, by invoking `xm list`, as we do in Listing 13.9.

LISTING 13.9 Domains Running on Domain0 (Before Restoration)

```
[root@dom0]# xm list
Name                            ID Mem(MiB) VCPUs State   Time(s)
Domain-0                         0      511     2 r-----   1702.2
[root@dom0]#
```

To restore the domain from our checkpoint file, we issue the `xm restore` command with its checkpoint's file name as an argument, as in Listing 13.10.

LISTING 13.10 Restoring a Guest's State Using `xm restore`

```
[root@dom0]# xm restore TestGuest.checkpt
[root@dom0]#
```

We know that restoration is complete when we can observe TestGuest residing on Domain0, by invoking `xm list`, shown in Listing 13.11. The `xm restore` command will not return until restoration is complete. Once TestGuest is restored from a state file, it continues executing where it left off at the time it was suspended, though network connections are likely to have timed out.

LISTING 13.11 Domains Running on Domain0 (After Restoration)

```
[root@dom0]# xm list
Name                            ID Mem(MiB) VCPUs State   Time(s)
Domain-0                         0      511     2 r-----   1795.9
TestGuest                        2       63     1 -b----      0.4
[root@dom0]#
```

Xen's domain save and restore facility has several potential uses. For example, the ability to restore from checkpoints may potentially be used in developing a rapid crash recovery procedure, where a guest is restored to a default state much faster than rebooting. Similarly, it is also useful while debugging or testing changes to a system, by allowing an administrator to save quickly restorable checkpoints that can be reverted to in the event of a failure. Quick installs cannot be performed simply by providing such an image because checkpoint files do not contain a guest's file system contents; instead, quick installs and zero setup installations would require an image of a guest's file system, and optionally a checkpoint file, if it is desired to ship the installation with the guest being in a particular execution state.

Possible Save and Restore Errors

Listing 13.12 shows the error that occurs when there is not enough disk space to store the checkpoint file of a guest at the location of the path specified by the user. You should free up space or specify a different path to store the checkpoint file to fix this problem. If you specify a checkpoint file name that is the same as an existing file name, `xm save` overwrites the existing file without warning.

LISTING 13.12 Error: xm_save failed

```
[root@dom0]# xm save TestGuest TestGuest.checkpt
Error: /usr/lib64/xen/bin/xc_save 46 82 0 0 0 failed
Usage: xc save <Domain> <CheckpointFile>

Save a domain state to restore later.
[root@dom0]#
```

The error shown in Listing 13.13 can occur under several circumstances. This message commonly occurs when the domain contained in the checkpoint file you specified is already running. The message may also occur if the checkpoint file you specified is corrupt or invalid, or in situations where there is not enough RAM on the host system to restore the guest contained in the checkpoint file you specified.

LISTING 13.13 Error: Restore failed

```
[root@dom0]# xm restore TestGuest.checkpt
Error: Restore failed
Usage: xm restore <CheckpointFile>

Restore a domain from a saved state.
[root@dom0]#
```

Listing 13.14 shows a different type of restore error. This occurs when the checkpoint file is inaccessible, which happens if the checkpoint file you specified does not exist, is inaccessible due to permissions, or cannot be read due to device error.

LISTING 13.14 Error Message When `xm restore` Is Unable to Read File

```
[root@dom0]# xm restore TestGuest.checkpt
Error: xm restore: Unable to read file /root/TestGuest.checkpt
[root@dom0]#
```

Types of Guest Relocation

The ability to easily move a guest operating system across the network from one physical host to another can be useful for a number of different administrative tasks such as load balancing or dealing with scheduled maintenance. Xen provides integrated relocation, or migration, support as illustrated in Figure 13.2. It helps manage the process of preparing, transporting, and resuming guests from one nearby host to another.

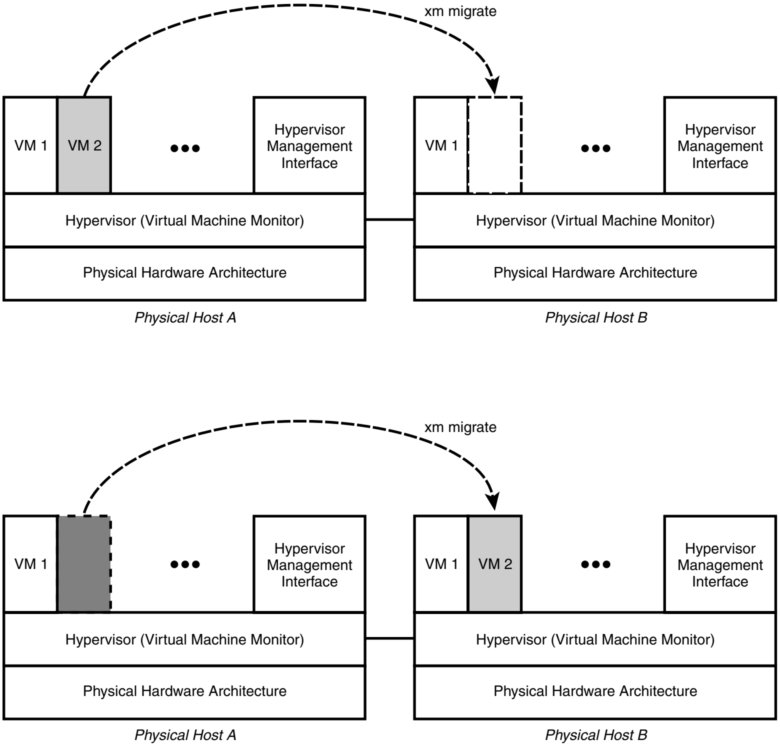

xm migrate
VM 1
VM 2
•••
Hypervisor Management Interface
VM 1
•••
Hypervisor Management Interface
Hypervisor (Virtual Machine Monitor)
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture
Physical Hardware Architecture
Physical Host A
Physical Host B
xm migrate
VM 1
•••
Hypervisor Management Interface
VM 1
VM 2
•••
Hypervisor Management Interface
Hypervisor (Virtual Machine Monitor)
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture
Physical Hardware Architecture
Physical Host A
Physical Host B

FIGURE 13.2 Guests may be relocated (migrated) between different Xen servers for various administrative purposes.

For comparison purposes, we begin our discussion about guest relocation by introducing the concept of cold static relocation, in which an administrator manually copies all the files that define a guest to a different machine, and executes `xm create` to start the guest at its new location. We then discuss Xen's integrated migration facility, which automates two major paradigms of guest relocation over a network—warm static migration and live migration. For warm static migration (also called regular migration), a guest is suspended on its source host, all relevant state information is transferred to its destination host, and the guest is resumed on its destination host after its state and memory have been safely relocated. This process is effectively the same as checkpointing a guest, manually copying the checkpoint file to another host, and restoring the guest on the new host. Warm static migration does not preserve ongoing network connections and does expose client visible downtime; for live migration, however, a guest is transferred without being suspended, and its services and connections are not only preserved but continue effectively uninterrupted.

Cold Static Relocation

Cold static relocation is accomplished manually without the help of Xen's integrated migration facility. Understanding the elements of this manual process aids in a better understanding of and an appreciation for the `xm migrate` command.

A halted guest may be relocated between two hosts by ensuring that its configuration file appears on and its file systems are available to both hosts. There are two ways of accomplishing this goal. The first occurs when both hosts share underlying storage (network attached storage). The second method involves manually copying the configuration file and file systems from the host to the target hardware. In the latter case, manual copying might take a long time because a guest's file systems might be very large. The transfer of a guest's file systems and configuration file by manual means might occur, for instance, through the use of optical media or FTP/SFTP. A much simpler option is to store the guest's file systems on network-attached storage, which makes copying unnecessary to make the guest's file systems available on both the source and destination hosts.

A running guest might also be relocated using this method, but must first be suspended to a checkpoint file. A guest domain may be checkpointed using the `xm save` command. Once suspended, a guest may be relocated by ensuring its file systems, checkpoint file, and configuration file are accessible on the destination host. When two physical hosts share the same underlying storage, this is equivalent to what `xm migrate` does for our next type of relocation, warm static migration.

Warm static migration could also be performed by manual copying methods if the two physical hosts did not share the storage systems on which all the needed files were stored. Once these three components are available on the desired destination host, the guest may be reactivated using the `xm restore` command.

Be advised that invoking `xm create` for the same guest domain on two different hosts at once has serious ramifications. Although the configuration file and operating system kernel would remain undamaged, multiple guests manipulating the same storage directly would likely lead to file system corruption.

Warm Static (Regular) Migration

Warm static migration, or regular migration, of a guest domain is the combined process of pausing the execution of that guest's processes on its original host, transferring its memory and processes from its origin host to a destination host, and resuming its execution on the destination host. Warm static migration enables a domain to be migrated from one physical host to another, only temporarily pausing its execution, without requiring it to be shut down and restarted. This is illustrated in Figure 13.3. Xen provides fast and simple integrated facility for performing regular migration of guest domains between hosts for load balancing, transferring from old to new hardware, or bringing physical hosts down for maintenance while adopting their guests on other hosts.

Migration requires a guest's memory contents to be transferred, I/O transactions temporarily quiesced, CPU register states transferred, and network connections rerouted and resumed on the destination host. Xen does not currently support the automatic mirroring of a guest's file systems between different hosts, nor does it automatically transfer a guest's file systems, because the act of copying an entire root file system is often too cumbersome to instantaneously transfer during migration. As a result, if a Xen user wants to use Xen's integrated migration support, it is currently necessary to configure guests to access their file systems over network shares that are equally available to both the source and destination host. In regular migration, a guest domain is suspended to easily handle the transfer of guest memory pages that would be continuously changing if the guest was otherwise allowed to continue to run. Suspending the guest also satisfies the need to quiesce I/O.

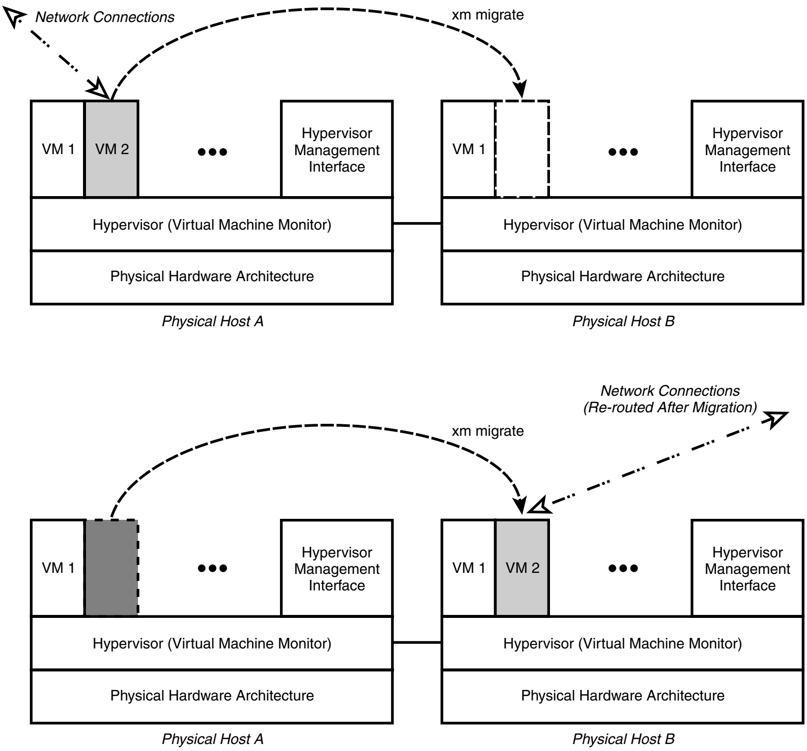

Network Connections
xm migrate
VM 1
VM 2
•••
Hypervisor Management Interface
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture
Physical Host A
VM 1
•••
Hypervisor Management Interface
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture
Physical Host B
Network Connections (Re-routed After Migration)
xm migrate
VM 1
•••
Hypervisor Management Interface
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture
Physical Host A
VM 1
VM 2
•••
Hypervisor Management Interface
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture
Physical Host B

FIGURE 13.3 In warm static migration, a guest domain is temporarily suspended on its source host prior to relocation and resumed on its destination host after its memory contents have been transferred.

Live Migration

The mere ability to migrate a guest domain from one physical host to another is beneficial, but performing migration by temporarily suspending and then restoring a guest's state is not suitable in all applications. This process of migration can incur outages that are perceptible to system users. These sorts of outages are typically on the order of seconds or minutes, depending on the network infrastructure and the amount of memory a guest is allocated. Warm static or regular migration is not applicable for use in situations where in-use guests must be relocated without their services experiencing downtime; instead, in such situations, the ability to migrate a guest while maintaining its current state, operation, and network connections is desired.

Xen's third form of relocation, live migration, enables a domain to be migrated while it is in operation and without the interruption of its services or connections, as illustrated in Figure 13.4. Live migration of a guest is the act of seamlessly moving its execution to the new physical host, including redirecting established and future network connections away from its original and to its new location.

Live migration is considerably more complicated than regular migration primarily because the state of the guest is changing while it is being copied. This requires an iterative process of copying the state, checking to see what has changed in the meantime, and then copying what has changed.

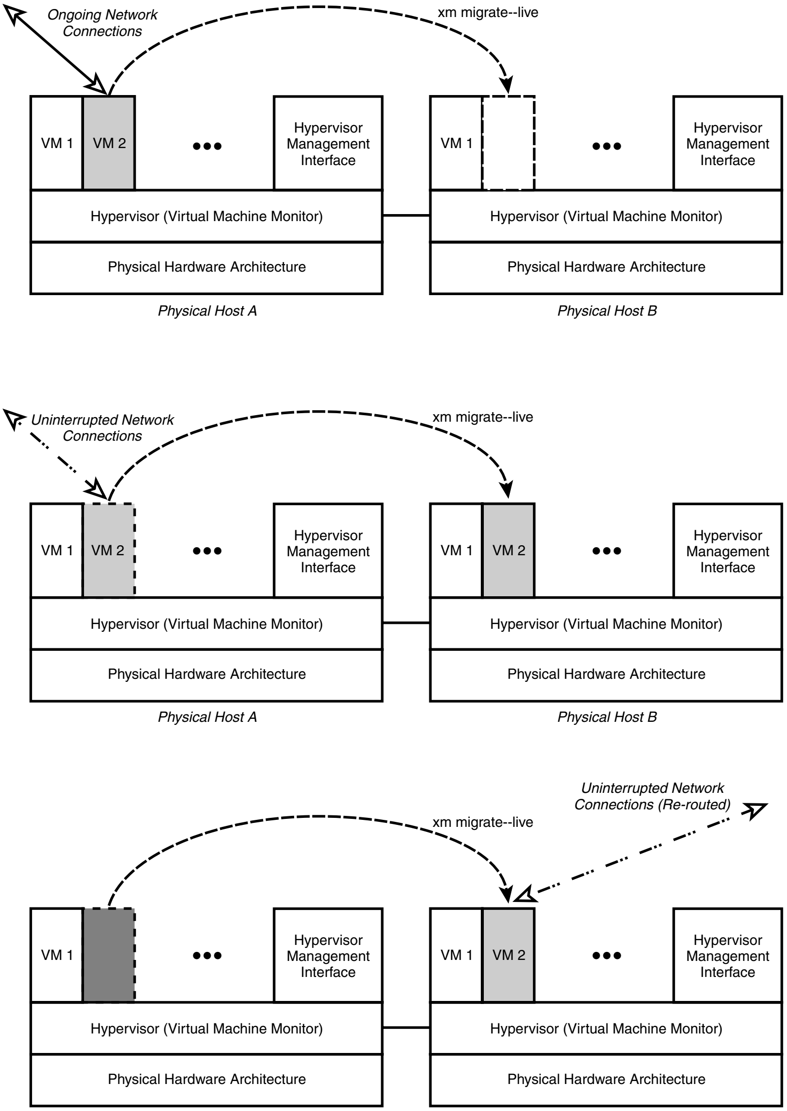

Ongoing Network Connections
xm migrate--live
VM 1
VM 2
Hypervisor Management Interface
Hypervisor (Virtual Machine Monitor)
Physical Hardware Architecture
Physical Host A
Physical Host B
Uninterrupted Network Connections
Uninterrupted Network Connections (Re-routed)

FIGURE 13.4 In live migration, a guest domain continues to execute and remains available throughout relocation, and client connections to its services go uninterrupted.

Making memory that belongs to an active guest available on a different host system is difficult because processes that are executing and memory pages that are swapped to and from disk consistently alter the contents of memory and consistently make attempted copies out-of-date. Like regular migration of a guest, Xen's live migration must also transfer and restore CPU register states, quiesce I/O transactions, and reroute and resume network connectivity on the destination host; however, in live migration, these events must occur in a way and over a small enough duration that there is no perceptible downtime of the guest's services. Additionally, like regular migration, a guest's file systems must be network-accessible to both the source and destination hosts.

The implementation of Xen's live migration involves the novel use of an iterative multipass algorithm that transfers the virtual machine guest memory in successive steps. After the source VM and the destination VM first negotiate to ensure resources are sufficient on the receiving machine, an initial pass over the guest's memory is performed with each page being transferred to the destination. On each successive iteration, only the guest memory that has been dirtied in the interim is sent. This process is executed until either the remaining number of dirty guest pages is sufficiently small enough that the remaining pages can be transmitted quickly or the number of dirty pages remaining to transfer in each pass is not decreasing. At that point, the system is actually quiesced and the final state sent to the new host and the transfer of control to the new physical machine completed. In most cases, this final step can be accomplished so quickly that there is no client perceptible delay and ongoing network connections are preserved.

Preparing for `xm migrate`

To support regular and live migration, a Xen configuration must include the following:

- Two or more physical hosts with `xend` configured to listen for relocation requests
- Hosts that are members of the same layer-2 network
- A storage server that is accessible by guests on both the source and destination hosts, which provides network access to guests' root and all other file systems
- A configuration file for each guest on both the source and destination hosts
- Sufficient resources on the destination host to support the arrival of guests
- The same version of Xen running on both the source and destination hosts

In the subsections that follow, we cover each of these steps in detail.

Configuring xend

To enable relocation (migration) support between hosts we must first edit their xend configuration files. In /etc/xen/xend-config.sxp on each host (or /etc/xen/xend.conf, depending on your system), remove the comment character ("#") from the beginning of each of the following lines if necessary (note that these lines might not immediately follow one another), as illustrated in Listing 13.15.

LISTING 13.15 An Example /etc/xen/xend-config.sxp Prior to Enabling Relocation Support

```
#(xend-relocation-server no)
#(xend-relocation-port 8002)
#(xend-relocation-address '')
#(xend-relocation-hosts-allow '')
```

On the xend-relocation-server line, ensure the value is set to yes. This enables xend to listen for relocation requests from other hosts so that it can receive the relocated guests. These lines should now read as shown in Listing 13.16.

LISTING 13.16 An Example /etc/xen/xend-config.sxp After Enabling Relocation Support

```
(xend-relocation-server yes)
(xend-relocation-port 8002)
(xend-relocation-address '')
(xend-relocation-hosts-allow '')
```

The second and third lines specify the port and address that xend should listen on for incoming migration requests. You may change the xend-relocation-port or leave it the default, port 8002. Leaving the xend-relocation-address field blank between the single quotes configures xend to listen on all addresses and interfaces. The fourth line may be used to restrict access to migration on a particular host to only certain hosts. Leaving this field blank allows all hosts with network access to the relocation port and address on which xend is listening to negotiate migration. Leaving these fields blank suffices for testing purposes but may not provide a desirable level of security in production systems. A better idea is to specify a list of hosts that should be allowed to negotiate migration with a particular host, and/or create a separate, secure network for guest migration. Of course for tighter security, a good idea is to configure both.

It is imperative for the relocation service to be accessible only by trusted systems and for live migration to be performed only across a trusted network. It is important to perform migration across a trusted network because when migrated, the contents of a guest's memory are transferred from the source to destination hosts in a raw, unencrypted form. The memory contents might contain sensitive data and can be intercepted on a shared network by other hosts capable of scanning the migration traffic. Additionally, allowing unrestricted access to the relocation service on a host could allow an attacker to send fraudulent guest domain data to that host, and possibly hijack or impair system and network resources. Whenever possible, the `xend-relocation-address` should be set on the source and destination hosts to the address of an interface that is connected to an isolated administrative network, where migration can occur in an environment that is not shared with untrusted hosts (or the Internet). Ideally, the transmission medium should be high speed for best results. Having an isolated network for migration and listening for relocation requests only on that network adds natural security to a Xen configuration by physically preventing untrusted hosts from accessing the relocation port or spying on a migrating guest's memory contents. Please see Chapter 11, "Securing a Xen System," for more tips on hardening your Xen configuration.

To have `xend`'s relocation service listen only on one address, specify the address in the `xend` configuration file. For example, if the address 10.10.3.21 is bound to a network interface available to Domain0, the setting illustrated in Listing 13.17 causes `xend`'s relocation service to only listen for relocation requests sent to 10.10.3.21. This configuration feature helps secure a host's relocation service if undesired hosts are blocked from making connections to the specified address, either by firewall rules or by being bound to a network interface connected to a separate administrative network.

LISTING 13.17 An Example of `xend` Bound to a Relocation Address

```
(xend-relocation-address '10.10.3.21')
```

The format for specifying a restricted set of allowed hosts to connect to the relocation service is a list of regular expressions, separated by spaces, that specify hosts that are to be accepted. For instance, the example in Listing 13.18 would cause a particular host to only listen to migration requests from itself and hosts within the our-network.mil domain. Host IP addresses may also be listed in this line. Restricting access to the relocation service by only certain hosts can also be achieved through good firewall practices, either in addition to or instead of setting `xend-relocation-hosts-allow`. Any host that matches any one of the regular expressions listed in this field will be allowed to negotiate migration with the local host.

LISTING 13.18 An Example of Restricting Allowed Source Hosts

```
(xend-relocation-hosts-allow '^localhost$                         ➥
     ^.*\.our-network\.mil$')
```

For our sample configuration, we want our hosts Domain0_Host1 (10.0.0.1) and Domain0_Host2 (10.0.0.2) to accept relocation requests only from themselves and each other, but, unfortunately, our hosts do not have separate real or virtual network interfaces on a separate network just to be used for relocation. Remember that this host setup can be less secure than having a separate network and interface for guest relocation, and can decrease network performance. Our sample `xend` configuration file for Domain0_Host1, contained in Listing 13.19, shows `xend` configured only to accept migration requests from its own host and Domain0_Host2.

LISTING 13.19 A Sample `xend` Configuration File for Domain0_Host1

```
# dom0_Host1 (10.0.0.1)
# Xend Configuration File
#

# = Basic Configuration =
(xend-unix-server yes)
(xend-unix-path /var/lib/xend/xend-socket)


# =*= Relocation Configuration =*=
(xend-relocation-server yes)  # Enable guest domain relocation
(xend-relocation-port 8002)                                       ➥
     # Port xend listens on for relocation requests
(xend-relocation-address '')
     # Interface to listen for reloc requests [ALL]                ➥
(xend-relocation-hosts-allow '^localhost$                         ➥
     ^localhost\\.localdomain$ 10.0.0.2')
     # Hosts that are allowed to                                  ➥
     send guests to this host [only 10.0.0.2!]


# = Network Configuration =
(network-script network-bridge)
(vif-script vif-bridge)

# = Resource Configuration =
(dom0-min-mem 256)
(dom0-cpus 0)
```

Likewise, our sample xend configuration file for Domain0_Host2, with xend configured only to accept migration requests from its host and Domain0_Host1, is contained in Listing 13.20.

LISTING 13.20 A Sample xend Configuration File for Domain0_Host2

```
# dom0_Host2 (10.0.0.2)
# Xend Configuration File
#

# = Basic Configuration =
(xend-unix-server yes)
(xend-unix-path /var/lib/xend/xend-socket)


# =*= Relocation Configuration =*=
(xend-relocation-server yes)  # Enable guest domain relocation
(xend-relocation-port 8002)                                    ➥
     # Port xend listens on for relocation requests
(xend-relocation-address '')                                   ➥
     # Interface to listen for reloc requests [ALL]
(xend-relocation-hosts-allow '^localhost$                      ➥
     ^localhost\\.localdomain$ 10.0.0.1')
     # Hosts that are allowed to                               ➥
     send guests to this host [only 10.0.0.1!]


# = Network Configuration =
(network-script network-bridge)
(vif-script vif-bridge)

# = Resource Configuration =
(dom0-min-mem 256)
(dom0-cpus 0)
```

Proximity of Sources and Destinations on the Network

For guest migration to be possible, the source and destination hosts need to be members of the same layer-2 network and the same IP subnet. This is because Xen needs to maintain the same environment for a guest's services before and after migration, including the same IP and MAC addresses. A guest's IP and MAC addresses are transferred with that guest so that its network services remain accessible to other hosts once its migration completes.

Packet redirection to a new host is generally accomplished through Address Resolution Protocol (ARP), a protocol already familiar with networking hardware. If the source and destination hosts are located on different subnets, connections have to be redirected to the distant destination host in a more complicated way—for example, through the use of tunneling on Domain0, "Mobile-IP," dynamic DNS, or reconnection at the application level. In addition to the administrative complexity these proposed methods would create, they would also cause undesirable effects such as increasing overhead, latency, and even bandwidth usage if tunneling is used. Due to issues such as these, at the time of this publication, Xen does not provide an integrated capability to migrate guests between hosts that are not on the same layer-2 network and IP subnet. Solutions such as IP tunneling in Domain0 must be manually configured as needed. In many scenarios, the need for geographically-isolated source and destination hosts coincides with the need to be able to shift all dependency away from the source location; as such, tunneling is not viable because it depends on the source location to be able to maintain resources at its end of the tunnel.

Network-Accessible Storage

Recall that the migration support for Xen guests requires guest root file systems located on some form of mutually shared storage. Xen does not yet provide a facility for the automatic mirroring of local storage volumes at the Domain0 level, though work is currently being done to explore such possibilities. It is therefore necessary for each guest that is to be migrated to have its file system(s) mapped to network shares because the local devices available to a guest's original host will not be available locally on its destination host following migration.

There are several approaches to making a guest's files network accessible, including NFS, iSCSI, AoE, GNBD, and many others. Services such as iSCSI, ATA-over-Ethernet, and GNBD share access to volume block devices over the network, whereas services such as NFS share access to portions of a file system. See Chapter 8, "Storing Guest Images," for details on configuring a suitable network storage service such as the one mentioned in this section.

Guest Domain Configuration

We configure our guest's virtual block device, storing its root file system to be mapped to an ATA-over-Ethernet (AoE) shared volume. You may choose whichever network storage service is most convenient for your own needs on your premises. First, we set up the AoE initiator (client) on both of our hosts. Then we create an LVM volume group and volumes on our AoE share to serve as our sample guest's root and swap logical partitions. We define a virtual disk in the guest configuration file that points to our shared block device and also use pygrub as a bootloader.

Both the source and destination Xen hosts need a configuration file for guests that are to be migrated, which will be the same for our hosts because all of our guest's virtual block devices will be accessible identically on both the source and destination hosts. We name this sample guest TestGuest, and Listing 13.21 shows the configuration in `/etc/xen/TestGuest` that is identical on both hosts. If a network block storage service is to be used but LVM is not, configure the `disk` line so that it points directly to the device on Domain0 that corresponds to the appropriate network block device (located in the `/dev/` tree).

LISTING 13.21 A Sample `/etc/xen/TestGuest` with LVM and ATA-over-Ethernet on Both Domain0 Hosts

```
name = "TestGuest"
memory = "64"
disk = [ 'phy:/dev/VolumeGroup0/TestGuest-volume,xvda,w' ]
vif = [ 'mac=00:16:3e:55:9d:b0, bridge=xenbr0' ]
nographic=1
uuid = "cc0029f5-10a1-e6d0-3c92-19b0ea021f21"
bootloader="/usr/bin/pygrub"
vcpus=1
on_reboot   = 'restart'
on_crash    = 'restart'
```

The configuration shown in Listing 13.21 works for network storage services that export entire block devices; however, because NFS exports a file system and not a form of raw access to block devices, configuring a guest to have an NFS-shared root file system is slightly different. The main difference is that the guest configuration file will not define a virtual block device pointing to a shared block device, but instead will have an NFS server configured that stores the guest's root file system. An NFS root on TestGuest may be set up by the configuration in `/etc/xen/TestGuest` on both hosts, shown in Listing 13.22.

LISTING 13.22 A Sample of /etc/xen/TestGuest with NFS on Both Domain0 Hosts

```
name = "TestGuest"
memory = "64"
vif = [ 'mac=00:16:3e:55:9d:b0, bridge=xenbr0' ]
nographic = 1
uuid = "cc0029f5-10a1-e6d0-3c92-19b0ea021f21"
bootloader = "/usr/bin/pygrub"
vcpus = 1
root = "/dev/nfs"
nfs_server = '10.0.0.40'              # Address of our NFS server
nfs_root = '/XenGuestRoots/TestGuest'                        ➥
     # Path on server of TestGuest's root
on_reboot   = 'restart'
on_crash    = 'restart'
```

Version and Physical Resource Requirements

In addition to both the source and destination hosts needing to run the same version of Xen to allow for migration, the destination host also must have sufficient resources available to it in order to support the arrival of a guest. The destination host must have access to memory that is unallocated to either Domain0 or other domains to handle the arriving guest, and the minimum amount of memory needed equals the amount of memory allotted to the guest on the source host, plus an additional 8MB of temporary storage.

Experience with `xm migrate`

Xen's internal relocation facility supports both warm static migration and live migration. It is available through the `xm migrate` command.

`xm migrate`

This command minimally takes the domain ID (or domain name) of the guest that is to be migrated and the destination host to which it is to be migrated to as its first and second arguments, respectively. If a domain with the same numerical domain ID exists on the destination host, migration still occurs, but the guest is assigned a different domain ID on the destination host. Listing 13.23 shows the syntax of the command.

LISTING 13.23 Usage of `xm migrate`

```
xm migrate domain_id destination_host [-l|--live] [-r|--resource rate]
```

`xm migrate` supports two optional arguments: `--live` and `--resource`. The`--live` argument specifies live migration as the form of migration to be performed. If `--live` is not used, regular migration will be the type of migration performed. To reduce network saturation, the optional `--resource` argument, followed by a `rate` in megabits per second, may be used to specify the rate of data transfer during migration. The `--resource` argument is generally unnecessary when using a private network dedicated for migrations to be performed, and it is best to avoid supplying it whenever possible to ensure optimal throughput; however, if a single network or even a single network interface on a host shares both migration and normal guest network traffic, it may be wise to supply this argument to reduce the network saturation so that connections to and from guests are not affected as dramatically.

Xen's relocation facility, as interfaced through the `xm migrate` command, makes guest relocation a simple task. We demonstrate the usage of the `xm migrate` command for performing both regular migration and live migration.

Using **`xm migrate`** for Warm Static Migration

In this section, we perform a warm static (regular) migration of a guest domain. The Domain0_Host1 system is currently running one guest domain, TestGuest, as seen by invoking `xm list` at a console on Domain0_Host1 and as shown in Listing 13.24.

LISTING 13.24 Domains Running on Domain0_Host1

```
[root@dom0_Host1# xm list
Name                             ID Mem(MiB) VCPUs State   Time(s)
Domain-0                          0      511     2 r-----   5010.6
TestGuest                         1       63     1 -b----     15.3
[root@dom0_Host1]#
```

The guest domain TestGuest is the guest to be relocated using warm static migration, from Domain0_Host1 to Domain0_Host2. TestGuest (10.0.0.5) is currently accessible from a separate workstation on our network, which we reveal using the `ping` command in Listing 13.25.

LISTING 13.25 Demonstrating Remote Accessibility Using ping

```
[root@Other_Workstation]# ping TestGuest
PING TestGuest (10.0.0.5) 56(84) bytes of data.
64 bytes from TestGuest 10.0.0.5: icmp_seq=1 ttl=64 time=2.48 ms
64 bytes from TestGuest 10.0.0.5: icmp_seq=1 ttl=64 time=2.31 ms
64 bytes from TestGuest 10.0.0.5: icmp_seq=1 ttl=64 time=2.98 ms
64 bytes from TestGuest 10.0.0.5: icmp_seq=1 ttl=64 time=2.77 ms
[root@Other_Workstation]#
```

Next, to perform a warm static migration of our sample guest from its current host Domain0_Host1 to its destination host Domain0_Host2 (10.0.0.2), we run the xm migrate command at a console on Domain0_Host1, illustrated in Listing 13.26.

LISTING 13.26 Performing a Warm Static Migration Using xm migrate

```
[root@dom0_Host1]# xm migrate 1 10.0.0.2
[root@dom0_Host1]#
```

After the migration is complete, TestGuest executes on Domain0_Host2. You can check that it has finished successfully and is currently residing on Domain0_Host2 by running xm list in a terminal on Domain0_Host2, as illustrated in Listing 13.27.

LISTING 13.27 Domains Running on Domain0_Host2

```
[root@dom0_Host2]# xm list
Name                            ID Mem(MiB) VCPUs State   Time(s)
Domain-0                         0      511     2 r-----    710.1
TestGuest                        4       63     1 -b----     16.2
[root@dom0_Host2]#
```

To demonstrate that the guest is still accessible with its same IP address and hostname after the migration, we repeat our ping from our generic workstation on the network, as shown in Listing 13.28.

LISTING 13.28 Demonstrating Remote Accessibility at Same IP Address after Warm Static Migration

```
[root@Other_Workstation]# ping TestGuest
PING TestGuest (10.0.0.5) 56(84) bytes of data.
64 bytes from TestGuest 10.0.0.5: icmp_seq=1 ttl=64 time=2.99 ms
64 bytes from TestGuest 10.0.0.5: icmp_seq=2 ttl=64 time=2.27 ms
64 bytes from TestGuest 10.0.0.5: icmp_seq=3 ttl=64 time=2.53 ms
64 bytes from TestGuest 10.0.0.5: icmp_seq=4 ttl=64 time=2.43 ms
[root@Other_Workstation]#
```

The domain is still accessible on the network the same way it was prior to migration.

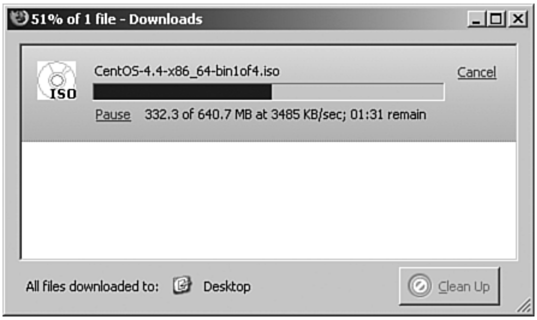
Using `xm migrate` for Live Migration

Now we demonstrate how to perform a live migration of a guest domain. First, let's examine our guest domain, TestGuest, residing on Domain0_Host1. To do so, we invoke `xm list`, as in Listing 13.29.

LISTING 13.29 Domains Running on Domain0_Host1

```
[root@dom0_Host1]# xm list
Name                                ID Mem(MiB) VCPUs State   Time(s)
Domain-0                             0      511     2 r-----   7170.6
TestGuest                            1       63     1 -b----     15.1
[root@dom0_Host1]#
```

Here, we see that TestGuest is residing on Domain0_Host1. The guest domain TestGuest is the guest we will once again migrate, only this time we will perform a live migration. We previously installed the Apache HTTP Web server on our sample guest in preparation for this example, and it is publishing a directory containing a large-sized file. To demonstrate the persistence of connections with our guest during and after live migration, we commence the downloading of a large file from a separate workstation on our network, as shown in Figure 13.5. We observe the status of our download after performing a live migration of our guest to confirm that connections to our guest's services remain uninterrupted throughout the process. We also present a test consisting of constant pings to illustrate the temporary increase in latency during migration when accessing the guest over the network.


51% of 1 file - Downloads

CentOS-4.4-x86_64-bin1of4.iso

332.3 of 640.7 MB at 3485 KB/sec; 01:31 remain

All files downloaded to: Desktop

FIGURE 13.5 A connection from another computer on our network is established with our guest domain, and the guest begins serving us a large file.

Next, we request live migration of our sample guest from its current host, Domain0_Host1, to its destination host, Domain0_Host2 (10.0.0.2), to occur through the xm interface. We do this by invoking xm migrate on its source host, Domain0_Host1, as shown in Listing 13.30.

LISTING 13.30 Performing a Live Migration Using xm migrate --live

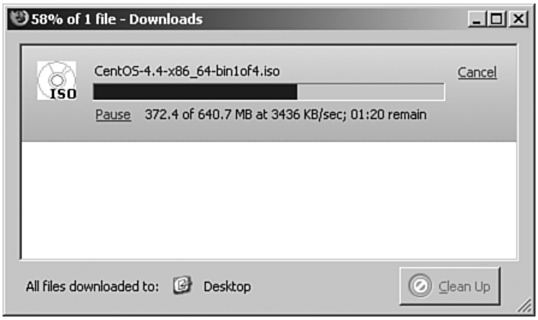
```
[root@dom0_Host1]# xm migrate --live 1 10.0.0.2
[root@dom0_Host1]#
```

After the migration is complete, TestGuest will reside completely on Domain0_Host2. You can check that it has finished successfully and is currently residing on Domain0_Host2 by running xm list in a terminal on Domain0_Host2, which is illustrated in Listing 13.31.

LISTING 13.31 Domains Running on Domain0_Host2

```
[root@dom0_Host2]# xm list
Name                                ID Mem(MiB) VCPUs State   Time(s)
Domain-0                             0      511     2 r-----   4314.2
TestGuest                            6       63     1 -b----     17.4
[root@dom0_Host2]#
```

To confirm that live migration did not disrupt any connections to our sample guest domain, we observe our download on the client workstation as shown in Figure 13.6.



58% of 1 file - Downloads

CentOS-4.4-x86_64-bin1of4.iso

372.4 of 640.7 MB at 3436 KB/sec; 01:20 remain

All files downloaded to: Desktop

FIGURE 13.6 The connection to our guest domain remains uninterrupted during and after migration.

Although the download rate of the file we were downloading decreased during migration, our connection was not dropped. From the client's perspective, the guest domain remained completely accessible during and after its transfer to Domain0_Host2.

During this example, we ran the ping utility on a separate workstation to illustrate the increase in latency experienced during live migration. We set up this test to perform constant pings before, during, and after TestGuest (10.0.0.5) was live migrated from Domain0_Host1 to Domain0_Host2. Listing 13.32 shows the results of the `ping` command.

LISTING 13.32 Guest Latency During Live Migration

```
[root@Other_Workstation ~]# ping TestGuest
PING TestGuest (10.0.0.5) 56(84) bytes of data.
64 bytes from TestGuest (10.0.0.5): icmp_seq=1 ttl=64 time=2.29 ms
64 bytes from TestGuest (10.0.0.5): icmp_seq=2 ttl=64 time=1.06 ms
64 bytes from TestGuest (10.0.0.5): icmp_seq=3 ttl=64 time=1.07 ms
64 bytes from TestGuest (10.0.0.5): icmp_seq=4 ttl=64 time=1.05 ms
64 bytes from TestGuest (10.0.0.5): icmp_seq=5 ttl=64 time=5.77 ms
64 bytes from TestGuest (10.0.0.5): icmp_seq=7 ttl=64 time=6.13 ms
64 bytes from TestGuest (10.0.0.5): icmp_seq=8 ttl=64 time=4.06 ms
64 bytes from TestGuest (10.0.0.5): icmp_seq=9 ttl=64 time=1.08 ms
64 bytes from TestGuest (10.0.0.5): icmp_seq=10 ttl=64 time=1.09 ms
64 bytes from TestGuest (10.0.0.5): icmp_seq=11 ttl=64 time=1.08 ms
64 bytes from TestGuest (10.0.0.5): icmp_seq=12 ttl=64 time=1.11 ms
[root@Other_Workstation ~]#
```

With Xen's approach for performing live migration, the guest domain is unreachable typically for only 50ms. Due to the brevity of its networking quiescence, the guest domain should have no apparent downtime from the perspective of client computers connected to that guest. Interruption of active connections should not occur because the time the guest domain is unavailable resembles the effects of an otherwise temporary surge in network latency. Even when migrating a computer gaming server, where high latencies often obviously affect application performance in an obvious way to users, this effective latency generally goes unnoticed (as seen in the famous demonstration of the live migration of a Quake game server while players are connected to it and participating in the game). Imperceptibility is what makes the illusion of live migration work.

Possible Migration Errors

Migration, though a fun demo, is more important when the capability is actually needed. In fact, for production systems, migration is something that should not be taken lightly. It is important to consider the level of computing power and other resources available when planning to migrate additional guests to a host. An easy mistake is to saturate all available hosts with guests when load balancing, and then at a later time attempt to migrate guests from one heavily loaded host to another that lacks the available resources to keep up with the demand. It is key to remember that if a host cannot normally support a particular guest due to resource limitations, it is not a viable destination host for the migration of that guest. Thus some planning of your migration infrastructure and contingency migration plans are in order.

In situations where systems are under extreme load, and where a noticeable and unacceptable decrease in performance might result from migrating a guest from one busy host to another, it is wise to have extra guests on the same hosts, and instead try to mix guests with different types of loads in a way that minimizes competition for the same resources of hosts.

The problem shown in Listing 13.33 can be encountered when:

- The destination host does not have enough memory to complete migration and now the guest domain is in a nonfunctioning zombie state.
- The `xm migrate` command is terminated prematurely on the source host.

LISTING 13.33 "Zombie-migrating" Guest Listed

```
[root@dom0_Host1]# xm list
Name                            ID Mem(MiB) VCPUs State   Time(s)
Domain-0                         0      191     2 r-----  49980.3
Zombie-migrating-TestGuest      14      256     1 ---s-d   1421.6
[root@dom0_Host1]#
```

Unfortunately, at the time of this writing, if this situation is encountered, the zombie guest cannot be destroyed, and the host needs to be restarted to alleviate this problem. A possible workaround for the zombie's memory consumption is to use `xm mem-set` to lower the memory allocated to the zombie to the minimum allowed amount (1MB).

Summary

In this chapter, we explored Xen's integrated facilities for guest domain checkpointing and migration that make these tasks trivial and useful for system administration. Xen makes saving and restoring guest domain states to disk possible through the easy-to-use `xm save` and `xm restore` commands. The ability to checkpoint guests makes it possible to pause and resume guest execution at a later date, roll back guest states for faster-than-rebooting crash recovery, and crudely transport guests between hosts.

Xen's integrated support for migration makes relocating guests between distinct physical hosts much more intuitive than manual means of relocating a guest image. Migration is performed with the `xm migrate` command, optionally with the `--live` argument for live migration. Live migration is particularly useful for crash avoidance, load balancing, and bringing down hosts for maintenance without interrupting the operation of important virtual machines.

References and Further Reading

Clark, Christopher et al. "Live Migration of Virtual Machines."
`http://www.cl.cam.ac.uk/research/srg/netos/papers/2005-migration-nsdi-pre.pdf`.

Virijevich, Paul. "Live Migration of Xen Domains." Linux.com.
`http://www.linux.com/articles/55773`.

Xen Users' Manual, Xen v3.0.
`http://www.cl.cam.ac.uk/research/srg/netos/xen/readmes/user/user.html`.

Index

A

-a option
- xm reboot command, 533
- xm shutdown command, 533

academic papers, 528-529

access
- mandatory access control
  - *overview, 422-423*
  - *sHype, 423-432*
  - *XSM (Xen Security Modules), 432-433*
- remote access, limiting, 408-412

acpi= boot option, 59

acpi_skip_timer_override boot option, 59

addlabel option (xm command), 535

address space identifiers (ASIDs), 77

addresses
- IP addresses, 343
- MAC (Media Access Control) addresses
  - *overview, 342, 389-390*
  - *specifying for guests, 390-391*

Allocate Memory and CPU window (virt-manager), 238

allocating memory
- balloon drivers, 451-453
- overview, 449-451
- shadow page tables, 451
- swap partitions, 454
- virtual devices, 303-307
- xm mem-max command, 457
- xm mem-set command, 454-457

AMD-V (AMD Virtualization), 15, 77-78

anticipatory scheduler, 467

AoE (ATA over Ethernet)
- client setup, 295-296
- disk parameter line, 127
- overview, 293
- Vblade, 293-295

aoe-stat command, 295

aoe module, 295

Apache JMeter, 549

apic=bigsmp boot option, 56

apic_verbosity=debug boot option, 58

application virtualization
- advantages/disadvantages, 13
- definition, 11

apt-get command, 99-100

as (anticipatory scheduler), 467

ASIDs (address space identifiers), 77

assigning
- IP addresses
  - *Broadcast IP, 392*
  - *Global IP, 392*
  - *manually assigning IPs to guests, 392-393*
  - *obtaining addresses via external DHCP servers, 392*
  - *obtaining addresses via internal DHCP servers, 393-394*
  - *overview, 391-392*
  - *Private IP, 392*
- MAC (Media Access Control) addresses, 389-391

Assigning Storage Space window (virt-manager), 237

associating virtual partitions, 310

ATA over Ethernet. *See* **AoE**

automated population with virt-install, 225-228

B

backend/frontend architecture
- backend information in XenStore, 323-325
- frontend information in XenStore, 325-326
- overview, 52, 322

split-driver model, 322-323
XenBus, 323
backend/vbd directory, 323-324
backend/vif directory, 324-325
backend vifs (virtual network interfaces), 347-349
backups, 434-435
badpage= boot option, 57
balloon drivers, 451-453
bare-metal virtualization systems, 18
BE item (XenStore), 399
BE-path item (XenStore), 399
beancounters, 24
binary packages, installing Xen from, 101-105
depmod command, 103
GRUB menu, 104
install script, 102
mkinitramfs command, 103
thread local storage, 104
update-rc.d command, 104
Xen-related utilities, 101
xm list command, 105
binding PCI devices at runtime, 328, 329
block-attach option (xm command), 532
block-detach option (xm command), 532
block devices
definition, 124
xm block device-related commands, 532
block-list option (xm command), 532
Blocked state (domains), 171
Bochs, 18
booting prebuilt system images, 131
boot methods, 132
disk images, 142-146
external Xen compatible kernels, 136-142
pygrub bootloader, 132-135
bootloader= parameter (guest configuration files), 541
bootloaders, pygrub, 132-135
boot process
boot options, 55-59
acpi=, 59
acpi_skip_timer_override, 59
apic=bigsmp, 56
apic=es7000, 57
apic_verbosity=debug, 58
badpage=, 57
com1=, 58
console=com1, 57
console=vga, 57
conswitch=Cx, 57
dom0_mem=X, 56
lapic, 56
nmi=fatal, 58
nmi=ignore, 58
noapic, 59
noirqbalance, 57
nolapic, 56
noreboot, 58
sched=credit, 57
sync_console, 58
tbuf_size=X, 58
watchdog, 58
xencons=off, 59
xencons=tty, 59
GRUB configuration file boot options, 59
GRUB configuration file excerpts, 54-55
guest boot parameters, 541
hiding PCI devices from Domain0 at boot, 327-328
overview, 54
brctl command, 386
bridge-utils command, 344
bridging, 344-347
default configurations, 355-356
Domain0 network interfaces, 359
driver domain bridging interfaces, 356-359
dummy bridges
configuring, 385-387
testing, 388-389
guest domain bridging interfaces, 360-361
overview, 343-355
testing results, 361-363
Broadcast IP, 392
buffers, virtual frame buffers, 210-213
bugs, reporting, 524
Bugzilla repository, 524
building from source, 116-117

C

-c option
xm dmesg command, 536
xm dump-core command, 532
--cap=integer value option (xm sched-credit command), 534
caps, 463, 464

cc_compile_by attribute (xm info command output), 442
cc_compile_date attribute (xm info command output), 442
cc_compile_domain attribute (xm info command output), 442
cc_compiler attribute (xm info command output), 442
CentOS, 84
  Domain0 installation, 91-98
    checking Xen kernel, 98
    GRUB menu, 97
    source code listing, 94-97
    virtualization options, 93
  virt-manager, 233-241
    Allocate Memory and CPU window, 238
    Assigning Storage Space window, 237
    Create a New Virtual System window, 234-235
    Creating Virtual Machine dialog, 239
    installation screen, 240-241
    main window, 234-235
    Naming Your Virtual System window, 236
    Open Connection dialog, 234
    Ready to Begin Installation window, 239
    Virtual Machine Console, 239-240
cfgbootpolicy option (xm command), 535
cfq (complete fair queuing scheduler), 467
changing schedulers, 468-469
checkpoint files, 474
checkpoints, 24
chroot command, 250
CIM (Common Information Model), 501
Citrix XenServer Product Group, 17
  XenServer Enterprise, 502-503
  XenServer Express, 85, 502-503
  XenServer Standard, 502-503
--clear option (xm dmesg command), 536
clearing xm dmesg log, 444
clients
  AoE (ATA over Ethernet), 295-296
  iSCSI initiators, 291-293
  NFS (Network File System), 298-299
  VNC clients, 208-209
cmdline entry (/vm/<uuid>/image directory), 69
cold static relocation, 480-481
com1= boot option, 58
commands. ***See*** **specific commands**
commodity virtualization technologies, 15
  Bochs, 18
  comparison of virtualization products, 24-25
  lguest, 22
  Linux-VServer, 23
  Linux KVM (Kernel Virtual Machine), 20-21
  Microsoft Hyper-V, 20
  Microsoft Virtual PC, 20
  OpenVZ, 23-24
  paravirt_ops, 22
  QEMU, 19
  UML (User-Mode Linux), 21
  VMware, 19
Common Information Model (CIM), 501
compiling
  custom Xen kernels, 117
  virt-manager (Virtual Machine Manager), 512
complete fair queuing scheduler, 467
compressed file system images, 128
  creating partitions with fdisk, 151-154
  creating partitions with GParted, 147-150
  formatting partitions with mkfs, 154-155
  formatting swap space with mkswap, 155-161
  overview, 146-147
  tar image files, 301-302
conferences, 528-529
configuration
  AoE (ATA over Ethernet)
    clients, 295-296
    Vblade, 293-295
  configuration files. *See also specific files*
    AoE (ATA over Ethernet), 296
    configuration options, 179-180
    definition, 123
    dhcp directive, 354
    disk parameter, 123-127
    gateway directive, 354
    guest configuration files, 352-354
    LVs (logical volumes), 281
    netmask directive, 354
    network scripts, 350-352
    NFS configuration example, 127-128
    nics directive, 353
    overview, 178, 352
    paths to, 181-182
    Python format, 178-179, 184
    SXP (S-Expression) format, 180-181
    vif directive, 352, 353
    virtual network interface (vif) scripts, 350-352

guests, 489-491, 541-544
iptables, 414-415
iSCSI
initiators, 291-293
targets, 289-291
networks. *See* network configuration
NFS (Network File System)
clients, 298-299
servers, 297-298
sHype policy, 425-432
virtual devices, 307
virtual partitions, 308-309
vnet (virtual network), 402
xend (Xen Daemon), 63-67, 485-488
configuration file for Domain0_Host1, 487
configuration file for Domain0_Host2, 488
configuration parameters, 537-539
console-limit option, 66
dom0-cpus option, 65
dom0-min-mem option, 65
enable-dump option, 64
external-migration-tool option, 65
logfile option, 63
loglevel option, 64
network-script option, 66
sample xend-config.sxp file, 66-67
vif-script option, 66
xend-address option, 64
xend-http-server option, 64
xend-port option, 64
xend-relocation-address option, 65
xend-relocation-hosts-allow option, 65
xend-relocation-port option, 65
xend-relocation-server option, 65
xend-tcp-xmlrpc-server option, 64
xend-unix-server option, 64
xend-unix-xmlrpc-server option, 64
console=com1 boot option, 57
console-limit configuration option (xend), 66, 539
console=vga boot option, 57
--console option (xm create command), 533
console option (xm command), 202-203, 533
conswitch=Cx boot option, 57
controllers, Domain0 requirements, 78-79
Controlling Virtual Systems. *See* **ConVirt Project**
converting system images, 161-162, 270-274
ConVirt Project, XenMan, 513-518
binary package installation, 514
running, 514-518
source-derived installation, 514
copy on write (CoW), 286-287
cores_per_socket attribute (xm info command output), 440
CoW (copy on write), 286, 287
cpu entry (/local/domain/<domId> directory), 70
cpu_mhz attribute (xm info command output), 440
cpu_time entry (/local/domain/<domId> directory), 70
cpu_Weight entry (/local/domain/<domId> directory), 70
CPU utilization
logical CPUs, 458
virtual CPUs (VCPUs)
HVM VCPU management, 459
manual administration, 462-463
overview, 458-459
xm vcpu-list command, 460-461
xm vcpu-pin command, 462
xm vcpu-set command, 461
cpus= parameter (guest configuration files), 543
--crash option (xm dump-core command), 532
Crashed state (domains), 171
Create a New Virtual System window (virt-manager), 234-235
create option (xm command), 533
--c option, 183-184
console output, 183-184
creating guests with, 175-178
--dryrun option, 182-183
overview, 174
prerequisites, 174-175
Creating Virtual Machine dialog (virt-manager), 239
CREDIT scheduler, 56
isolation between guests, 464
overview, 463
weight and cap, 463-464
xm sched-credit command, 465
customizing system images
file system table, 268-269
hostnames, 266-267
packages/services, 268
users, 267-268
custom Xen kernels, compiling, 117

D

daemons
- vbladed, 293
- XenConsole, killing, 61
- xend (Xen Daemon). *See* xend
- xenstore, killing, 61

dd command, 218-219, 303-307, 312-313

deadline scheduler, 466

Debian, 84
- debootstrap, 242-246
- guests, creating, 34-38

debootstrap command, 242-246

debugging
- advantages of virtualization, 3
- guest creation
  - *console output, 183-184*
  - *--dryrun option (xm create command), 182-183*
  - *duplicate guest names, 185-186*
  - *inability to run networking scripts, 189*
  - *incorrect syntax of Python configuration files, 184*
  - *insufficient loopback devices, 187-188*
  - *insufficient memory, 186-187*
  - *kernel image does not exist, 185*
  - *loading kernel without modules, 190*
  - *running wrong kernel in Domain0, 190*
  - *running xm command from non-root account, 191*

deleting
- guests, 38
- swap files, 159

depmod command, 103

designing virtual network topology, 341-343

desktops, Xfce, 32

destroying guests, 198-199

destroy option (xm command), 198-199, 533

detaching
- disk image files, 311
- virtual partitions, 311

Device-mapper, 252

device/vbd directory, 325-326

device/vif directory, 326

device management. *See also* **backend/frontend architecture**
- additional device types, 336
- device emulation with QEMU-DM, 334-335
- exclusive device access, 331-332
- PCI devices
  - *granting to another domain, 329-331*
  - *hiding from Domain0 at boot, 327-328*
  - *identifying, 326-327*
  - *manually binding/unbinding at runtime, 328-329*
- smart devices, 336
- trusted driver domains, 332-334
- virtualization. *See* virtualization

dhcp= parameter (guest configuration files), 543

dhcp directive, 354

DHCP servers
- external servers, 392
- internal servers, 393-394

directives
- dhcp, 354
- gateway, 354
- netmask, 354
- nics, 353
- vif, 352-353

Direct Memory Access (DMA) remapping, 77

Director (IBM), 506

directories. *See* **specific directories**

disabling thread local storage, 104

Disco project, 15

disk= parameter (guest configuration files), 542

disk image files, 129, 142-146
- allocating virtual devices, 303-307
- associating virtual partitions, 310
- configuring virtual devices, 307
- creating virtual partitions, 308-309
- detaching, 311
- detaching virtual partitions, 311
- formatting virtual partitions, 310-311
- mounting, 314
- overview, 302-303
- populating HVM guests from, 218-225
  - *configuration files, 219-225*
  - *dd command, 218*
  - *display configuration, 220-221*
  - *xm create command, 221*

disk parameter (configuration files), 180, 123-127
- AoE example, 127
- example, 123
- file: prefix, 125-126
- format, 123
- iSCSI example, 127
- phy: prefix, 124
- tap:aio prefix, 125

disk partitions. *See* **partitions**
Distributed Management Task Force (DMTF), 501
Distributed Management Task Forceís Common Information Model (DMTF CIM), 17
distributions
Linux
distribution-specific resources, 530
metadistributions, 105
Xen
enterprise class Linux support, 85-86
open source distributions, 83-84
Virtual Iron Software, Inc., 85
XenServer Express Edition, 85
DMA (Direct Memory Access) remapping, 77
dmesg option (xm command), 536
DMTF (Distributed Management Task Force), 501
DMTP CIM (Distributed Management Task Force's Common Information Model), 17
Dom0. *See* **Domain0**
dom0-cpus configuration option (xend), 65
dom0-cpus parameter (xend), 539
dom0_mem=X boot option, 56
dom0-min-mem configuration option (xend), 65
dom0-min-mem parameter (xend), 539
--domain option (xm sched-credit command), 534
Domain0
building from source, 116-117
choosing operating system for, 59-60
compared to driver domains and guest domains, 52
comparison of distribution choices, 115
definition, 29
GRUB (Grand Unified Boot Loader), 87
hardware device support
disk and controllers, 78-79
graphics devices, 80
networking devices, 80
unsupported hardware, 81
hiding PCI devices from at boot, 327-328
images:choosing from GRUB menu, 30-31
installation, 86-87
CentOS, 91-98
Gentoo, 105, 107-112
OpenSUSE, 88-90
Ubuntu, 98-101
Xen from binary packages, 101-105
memory requirements, 81-83
network interfaces, 359
non-Linux Domain0 installations, 114-115
overview, 51-52
power management, 81
AMD-V, 77-78
HVM, 78
Intel VT, 77
x86 processors, 76-77
running wrong kernel in Domain0, 190
security
limiting remote access, 408-412
limiting user access, 412
moving device drivers into DriverDomains, 412
removing software and services, 407-408
XenExpress, 112-114
domain identifier (DOMID), 68
domains
comparison of, 52
Domain0. *See* Domain0
DomainU. *See* DomainU
DOMID (domain identifier), 68
driver domains:multiple network interfaces, 394-396
granting PCI devices to, 329-331
guests. *See* guests
multiple network interfaces
handling in driver domains, 394-396
handling in guest domains, 396-399
overview, 394
pausing, 199-200
states, 171
unpausing, 200-201
xend domain-related parameters, 539
xm domain-related commands, 532
DomainU
definition, 29
guests. *See* guests
security
backing up VM images, 434-435
overview, 433-434
running VMs only when needed, 434
starting automatically, 191-192
xm DomU-related commands, 533
DOMID (domain identifier), 68
domId entry (/local/domain/<domId> directory), 70
domid option (xm command), 532
domi scripts, 252-256
booting guests, 253
building guests, 253-256
configuration files, 253-254
dependencies, 252

Device-mapper, 252
installing, 252
domname option (xm command), 532
DomU. *See* **DomainU**
downloading
prebuilt guest images
from jailtime.org, 130
from rPath, 131
from Virtual Appliances, 131
sHype, 424
Snort, 419-420
Xen LiveCD image, 29-30
driver domains, 412
compared to Domain0 and guest domains, 52
multiple network interfaces, 394-396
drivers, moving into DriverDomains, 412
--dryrun option
xm command, 532
xm create command, 182-183, 533
dummy0
configuring, 383-385
testing, 385
dummy bridges
configuring, 385-387
testing, 388-389
dump-core option (xm command), 532
dumping XenStore contents, 72-73
dumpkey() function, 73
dumppolicy option (xm command), 535
duplicate guest names, 185-186
Dying state (domains), 171

E

-e value option (xm sched-sedf command), 534
e2fsck command, 284
e2label command, 157-158
ebtables, 346
echo command, 108
emerge command, 107-110, 247
emulation
advantages/disadvantages, 12
Bochs, 18
definition, 6
QEMU, 19
QEMU-DM, 334-335
enable-dump configuration option (xend), 64, 539
Enomalism, 507-509
Enomalism Virtual Appliance (EVA) transport format, 507
enterprise class Linux support, 85-86
enterprise management
Citrix XenServer Enterprise, 502-503
Citrix XenServer Express, 502-503
Citrix XenServer Standard, 502-503
Enomalism, 507-509
IBM Virtualization Manager, 506-507
multiple system management, 518
overview, 499
programmatic interfaces to Xen hypervisor
legacy interfaces to xend, 502
libvirt, 500-501
overview, 500
Xen API, 501
Xen-CIM, 501
references and further reading, 519
virt-manager (Virtual Machine Manager), 509-513
capabilities, 509-512
command-line options, 513
compilation procedure, 512
Virtual Iron, 504-506
XenMan, 513-518
binary package installation, 514
running, 514-518
source-derived installation, 514
errors
kernel image does not exist, 185
migration errors, 497
xend (Xen daemon), 167
/etc/exportfs file, 297-298
/etc/fstab file. *See* **fstab file**
/etc/hosts file, 266-267
/etc/make.conf file, 107
/etc/snort directory, 420
/etc/xen/auto file, 191-192
/etc/xen/xend-config.sxp file, 485
EVA (Enomalism Virtual Appliance) transport format, 507
evaluating performance. *See* **performance evaluation**
evt-ch item (XenStore), 399
exclusive device access, 331-332
exportfs file, 297-298
extending
file systems, 283
LVs (logical volumes), 283-284

"Extending Xen with Intel Virtualization Technology" (article), 454, 460
external-migration-tool configuration option (xend), 65, 539
external DHCP servers, 392
external Xen compatible kernels, 136-142
extra= parameter (guest configuration files), 541
extra option (configuration files), 180
--extra=flag value option (xm sched-sedf command), 534

F

-f option (xm network-list command), 534
fdisk, 151-154, 308-309, 314
Fedora, 84
resources, 530
virt-manager, 233-241
Allocate Memory and CPU window, 238
Assigning Storage Space window, 237
Create a New Virtual System window, 234-235
Creating Virtual Machine dialog, 239
installation screen, 240-241
main window, 234-235
Naming Your Virtual System window, 236
Open Connection dialog, 234
Ready to Begin Installation window, 239
Virtual Machine Console, 239-240
files. *See also* **specific files**
checkpoint files, 474
configuration files. *See also specific files*
AoE (ATA over Ethernet), 296
configuration options, 179-180
definition, 123
disk parameter, 123-127
guest configuration files, 352-354
LVs (logical volumes), 281
network scripts, 350-352
NFS configuration example, 127-128
overview, 178
paths to, 181-182
Python format, 178-179, 184
SXP (S-Expression) format, 180-181
virtual network interface (vif) scripts, 350-352
growable files, 162
restoring guest state from, 476-478
saving guest state to, 474-478
Filesystem in Userspace (FUSE), 129
file systems
checking for errors, 284
extending, 283
file system table customization, 268-269
NFS (Network File System)
client configuration, 298-299
overview, 297
as root file system, 299-300
server configuration, 297-298
firefox command, 41
firewalls with iptables
configuration, 414-415
example, 415-419
overview, 413-414
rules:adding, 415-417
rules:IP spoofing, 418
rules:listing, 417
rules:removing, 417
rules:rule sets, 418
--force option
xm block-detach command, 532
xm network-list command, 534
formatting partitions
with mkfs, 154-155
with mkswap, 155-161
virtual partitions, 310-311
ForwardX11 parameter (SSH), 206
ForwardX11Trusted parameter (SSH), 206
FourInARow, 547
free_memory attribute (xm info command output), 441
FreeBSD, 84
Freenx, 212-213
frontend/backend architecture
backend information in XenStore, 323-325
frontend information in XenStore, 325-326
overview, 322
split-driver model, 322-323
XenBus, 323
frontend vifs (virtual network interfaces), 347-349
fstab file, 156-158, 268-269
Full System Emulation mode (QEMU), 19
full virtualiation, 321
advantages/disadvantages, 12
definition, 7-8
Linux KVM (Kernel Virtual Machine), 20-21
Microsoft Hyper-V, 20

Microsoft Virtual PC, 20
VMware, 19
functions, dumpkey(), 73
FUSE (Filesystem in Userspace), 129

G

gateway= parameter (guest configuration files), 543
gateway directive, 354
GDM (GNOME Display Manager), 31
Gentoo, 84
Domain0 installation, 105-112
echo command, 108
emerge command, 107-110
/etc/make.conf file, 107
GRUB menu, 110
kernel compilation, 109-110
kernel configuration, 109
unmasking xen-source packages, 108
xend service, 111
domi scripts, 252-256
booting guests, 253
building guests, 253-256
configuration files, 253-254
dependencies, 252
Device-mapper, 252
installing, 252
quickpkg, 246-251
resources, 530
getlabel option (xm command), 535
Global IP, 392
GNOME Display Manager (GDM), 31
Gnome Partition Editor (GParted), 147-150
GParted, 147-150
Grand Unified Boot Loader. *See* **GRUB**
granting PCI devices to other domains, 329-331
graphical interaction with guests
comparison of methods, 215
overview, 204
rdesktop, 213-214
virtual frame buffers and VNC/SDL libraries, 210-213
VNC (Virtual Network Computing), 207-209
X forwarding with SSH, 205-207
graphics, Domain0 requirements, 80
growable files, 162
GRUB (Grand Unified Boot Loader), 87
configuration file
boot options, 59
excerpts, 54-55
GRUB menu, 30-31
guest mode (KVM), 21
guests
compared to Domain0 and driver domains, 52
configuration, 489-491, 541-544
configuration files. *See also specific files*
AoE (ATA over Ethernet), 296
configuration options, 179-180
definition, 123
dhcp directive, 354
disk parameter, 123-127
gateway directive, 354
guest configuration files, 352-354
LVs (logical volumes), 281
netmask directive, 354
network scripts, 350-352
NFS configuration example, 127-128
nics directive, 353
overview, 178, 352
paths to, 181-182
Python format, 178-179, 184
SXP (S-Expression) format, 180-181
vif directive, 352, 353
virtual network interface (vif) scripts, 350-352
CPU utilization
logical CPUs, 458
virtual CPUs (VCPUs), 458-463
creating, 33-38, 174-178
debugging
console output, 183-184
--dryrun option (xm create command), 182-183
duplicate guest names, 185-186
inability to run networking scripts, 189
incorrect syntax of Python configuration files, 184
insufficient loopback devices, 187-188
insufficient memory, 186-187
kernel image does not exist, 185
loading kernel without modules, 190
running wrong kernel in Domain0, 190
running xm command from non-root account, 191
definition, 29
deleting, 38

destroying, 198-199
displaying running guests, 446-449
graphical interaction with
comparison of methods, 215
overview, 204
rdesktop, 213-214
virtual frame buffers and VNC/SDL libraries, 210-213
VNC (Virtual Network Computing), 207-209
X forwarding with SSH, 205-207
HVM (Hardware Virtual Machine) guest population
automated population with virt-install, 225-228
overview, 218
populating from discs or disc images, 218-225
image files, 312-313
interacting with, 38-41
listing information about
basic information, 169-170
information about specific guests, 171-172
nontabular output, 172-173
security privilege information, 173
MAC (Media Access Control) addresses, 390-391
memory allocation
balloon drivers, 451-453
overview, 449-451
shadow page tables, 451
swap partitions, 454
xm mem-max command, 457
xm mem-set command, 454-457
multiple network interfaces, 396-399
configuring guests with xm, 396-399
configuring multiple vifs to guests, 396
naming, 185-186
nongraphical interaction with
overview, 201
SSH (secure shell), 204
xm console command, 202-203
number of, 44
operating system kernels, 123
overview, 122
pausing, 199-200
PV (paravirtualized) guest population
debootstrap, 242-246
domi scripts, 252-256
overview, 228-229
quickpkg, 246-251
virt-manager, 233-241
Xen Express, 256-266
YaST Virtual Machine Management, 229-233
rebooting, 196-198
relationships to hypervisor, 48
relocation
cold static relocation, 480-481
live migration, 482-496
migration errors, 497
overview, 479-480
preparation, 484-491
references and further reading, 498
warm static migration, 481-494
xm migrate command, 491-496
returning information about
xm info command, 438-443
xm log command, 444-445
xm top command, 446-449
xm uptime command, 449
shutting down
overview, 193
xm destroy command, 198-199
xm reboot command, 196-198
xm shutdown command, 193-196
state, 171
possible save and restore errors, 478
restoring, 476-478
saving to file, 474-478
system images. *See* system images
unpausing, 200-201
virtual guests
relationship with hypervisors, 2-3
viewing, 32
zombie guests, 497

H

handle entry (/local/domain/<domId> directory), 70
handle item (XenStore), 399
hardware
additional device types, 336
balloon drivers, 451-453
commodity hardware, virtualization on. *See* commodity virtualization technologies
device emulation with QEMU-DM, 334-335
Domain0 requirements
disks and controllers, 78-79
graphics devices, 80
networking devices, 80
power management, 81
processors, 76-78
unsupported hardware, 81
drivers, moving into DriverDomains, 412

exclusive device access, 331-332
NUMA (nonuniform memory access computer) hardware, 15
PCI devices
granting to another domain, 329-331
hiding from Domain0 at boot, 327-328
identifying, 326-327
manually binding/unbinding at runtime, 328-329
relationship with hypervisors, 2-3, 48
smart devices, 336
trusted driver domains, 332-334
Hardware Virtual Machine guest population. *See* **HVM guest population**
help option (xm command), 532
hiding PCI devices from Domain0 at boot, 327-328
history of virtualization, 13
commodity hardware, 15
IBM mainframes, 14
time line, 14
x86 virtualization extensions, 15
Xen origins and time line, 15-17
host-only networks. *See* **purely virtual networks**
host attribute (xm info command output), 439
hosted virtualization systems, 18
hostname= parameter (guest configuration files), 180, 543
hostnames, 266-267
hosts, restricting, 486-487
hosts file, 266-267
HVM (Hardware Virtual Machine) guest population, 17, 78
automated population with virt-install, 225-228
overview, 218
populating from discs or disc images, 218-225
configuration files, 219-225
dd command, 218
display configuration, 220-221
xm create command, 221
VCPU management, 459
hw_caps attribute (xm info command output), 440
Hyper-V (Microsoft), 20
hypercalls, 51
hypervisor
anticipatory scheduler, 467
boot process. *See* boot process
capabilities, 49, 50
complete fair queuing scheduler, 467
credit scheduler
isolation between guests, 464
overview, 463
weight and cap, 463-464
xm sched-credit command, 465
deadline scheduler, 466
noop scheduler, 466
programmatic interfaces
legacy interfaces to xend, 502
libvirt, 500-501
overview, 500
Xen-CIM, 501
Xen API, 501
protection rings, 50-51
relationship with physical hardware and guest virtual machines, 2-3, 48
system messages:viewing, 443-444

I

I/O Memory Management Unit (IOMMU), 78
IANA (Internet Assigned Numbers Authority), 412
IBM
Director, 506
Virtualization Manager, 506-507
zServer, 551-553
IBM mainframes, 14
identifying PCI devices, 326-327
Idx item (XenStore), 399
ifcfg-dummy0 file, 384
ifconfigcommand, 330-331, 388
image entry (/local/domain/<domId> directory), 70
images
Domain0 images:choosing, 30-31
system images. *See* system images
virtual machine images, backing up, 434-435
info option (xm command), 536
initializing partitions, 279-280
initiators (iSCSI), 289-293
Input Output Memory Management Unit (IOMMU), 332
installation
Domain0, 86-87
building from source, 116-117
CentOS, 91-98
comparison of distribution choices, 115
Gentoo, 105-112
non-Linux Domain0 installations, 114-115

OpenSUSE, 88-90
Ubuntu, 98-101
XenExpress, 112-114
Xen from binary packages, 101-105
domi scripts, 252
vnet (virtual network), 400-401
insufficient loopback devices, 187-188
insufficient memory, 186-187
insufficient memory warning, 44
Intel Virtualization Technology (VT), 15, 77
intensive resource consumer applications, 554
interacting with guests, 38-41
graphical interaction
comparison of methods, 215
overview, 204
rdesktop, 213-214
virtual frame buffers and VNC/SDL libraries, 210-213
VNC (Virtual Network Computing), 207-209
X forwarding with SSH, 205-207
nongraphical interaction
overview, 201
SSH (secure shell), 204
xm console command, 202-203
internal DHCP servers, 393-394
Internet Assigned Numbers Authority (IANA), 412
Internet Small Computer Systems Interface. *See* **iSCSI**
intrusion detection mode (Snort), 420
IO schedulers
IOMMU (I/O Memory Management Unit), 78, 332
anticipatory scheduler, 467
complete fair queuing scheduler, 467
deadline scheduler, 466
noop scheduler, 466
IP addresses
Broadcast IP, 392
Global IP, 392
manually assigning to guests, 392-393
obtaining via external DHCP servers, 392
obtaining via internal DHCP servers, 393-394
overview, 343, 391-392
Private IP, 392
ip route command, 381
IP spoofing, 418
iptables
configuration, 414-415
example, 415-419
overview, 381-382, 413-414
rules
adding, 415-417
IP spoofing, 418
listing, 417
removing, 417
rule sets, 418
iSCSI (Internet Small Computer Systems Interface), 288
disk parameter line, 127
Enterprise Target project, 293
iscsi-initiator-utils package, 291
iscsiadm tool, 291-292
isolation (performance), 553-556
intensive resource consumer applications, 554
well-behaving versus misbehaving VMs, 554-555

J-K

jailtime.org, 130
Java Virtual Machine, 11
JMeter, 549

kernel= parameter (guest configuration files), 179, 541
kernel entry (/vm/<uuid>/image directory), 69
kernel image does not exist (error message), 185
kernels
custom Xen kernels, compiling, 117
external Xen compatible kernels, 136-142
loading kernel without modules, 190
pciback.hide kernel parameter, 327-328
running wrong kernel in Domain0, 190
KERNELS parameter (makefile), 117
Kernel Virtual Machine (KVM), 20-21
killing daemon instances, 61
kpartx command, 310-311
KVM (Kernel Virtual Machine), 20-21

L

-l option
xm list command, 533
xm network-list command, 534
-l value option (xm sched-sedf command), 534
--label option (xm list command), 173, 533
labeling root file system, 159
e2label command, 157-158
mkswap command, 159
labels option (xm command), 535

lapic boot option, 56
latency, 496
--latency=ms value option (xm sched-sedf command), 534
legacy applications, 5
legacy interfaces to xend, 502
lguest, 22
libraries
  pcap, 419-420
  virtualization
    advantages/disadvantages, 13
    definition, 11
  VNC/SDL libraries, 210-213
  Wine, 11
libvirt, 500-501
licenses, open source, 24
limit subdirectory (/local/domain/<domId> directory), 71
limiting
  remote access, 408-412
  user access, 412
Linux
  CentOS, 91-98
  distributions
    distribution-specific resources, 530
    metadistributions, 105
  KVM (Kernel Virtual Machine), 20-21
  Linux-VServer, 23
  OpenSUSE, 88-90
  Ubuntu
    Domain0 installation, 98-101
    LTS (Long Term Support), 99
    version numbers, 99
  UML (User-Mode Linux), 21
  Wine library, 11
  XenoLinux versus native Linux, 550-551
Linux-VServer, 23
Linux Partition HOWTO, 125
listing
  AoE block devices, 296
  guest information
    basic information, 169-170
    information about specific guests, 171-172
    nontabular output, 172-173
    security privilege information, 173
  iptables rules, 417
  partitions, 143
  running processes, 61
  system image contents, 139
  TCP (Transmission Control Protocol) ports, 409-411
list option (xm command), 533
  --label option, 173
  listing basic guest information, 169-170
  listing information about specific guests, 171-172
  listing security privilege information, 173
  --long option, 172-173
  nontabular output, 172-173
LiveCD
  Domain0 images, 30-31
  downloading LiveCD image, 29-30
  guests
    creating, 33-38
    deleting, 38
    interacting with, 38-41
    number of, 44
  login process, 31
  network testing, 41-43
  references and further reading, 45
  running, 28-29
  test system details, 28-29
  Xfce desktop, 32
live migration, 3, 482-496
load balancing, 463
loading
  kernels without modules, 190
  vnet (virtual network), 401
loadpolicy option (xm command), 535
/local/domain/<domId> directory, 70-71
log files, 444-445
logfile configuration option (xend), 63, 538
log option (xm command), 536
logical CPUs, 458
logical volume managers (LVMs), 278-279, 542
logical volumes. *See* **LVs**
login process, 31
loglevel configuration option (xend), 64, 538
logs (xend), 62-63, 444-445, 538
lomount command, 144-145
--long option
  xm block-list command, 532
  xm list command, 172-173, 533
  xm network-list command, 534
  xm vnet-list command, 536
  xm vtpm-list command, 532
Long Term Support (LTS), 99

loopback devices
  definition, 348
  insufficient loopback devices, 187-188
losetup command, 307, 311
ls command, 139, 295-296
lspci command, 327, 330
LTS (Long Term Support), 99
lvcreate command, 280, 287
lvextend command, 283
LVMs (logical volume managers), 278-279, 542
lvreduce command, 284, 285
LVs (logical volumes)
  configuration files, 281
  CoW (copy on write), 286-287
  creating, 280
  creating VGs for, 280
  extending, 283-284
  initializing partitions for, 279-280
  LVMs (logical volume managers), 278-279, 542
  populating, 281
  reducing, 284-285

M

MAC (Media Access Control) addresses
  overview, 342, 389-390
  specifying for guests, 390-391
MAC Addr. item (XenStore), 399
machine attribute (xm info command output), 439
mailing lists, 524
makepolicy option (xm command), 535
managed runtime, 11
management of enterprise environments. *See* enterprise management
mandatory access control
  overview, 422, 423
  sHype
    adding to Xen, 424-425
    configuring sHype policy, 425-432
    downloading, 424
    overview, 423-424
  XSM (Xen Security Modules), 432-433
measurements (performance)
  overview, 546
  repeatability of test results, 546-548
  virtual Web hosting, 548-550
  XenoLinux versus native Linux, 550-551
  Xen on x86 versus IBM zServer, 551-553
Media Access Control addresses. *See* MAC addresses
mem-max option (xm command), 534
mem-set option (xm command), 534
memory
  allocating guest memory
    balloon drivers, 451-453
    overview, 449-451
    shadow page tables, 451
    swap partitions, 454
    xm mem-max command, 457
    xm mem-set command, 454-457
  DMA (Direct Memory Access) remapping, 77
  Domain0 requirements, 81-83
  insufficient memory, 186-187
  insufficient memory warning, 44
  OOM (out of memory) error, 187
memory= parameter (guest configuration files), 179, 542
memory entry (/vm/<uuid> directory), 69
Mercurial repository, 527
metadistributions, 105
Microsoft Hyper-V, 20
Microsoft Virtual PC, 20
migration
  guests. *See* relocating guests
  live migration, 3, 494-496
  references and further reading, 498
  warm static migration, 492-494
  xend migration-related parameters, 539
mkfs, 154-155
mkfs command, 312-313
mkinitramfs command, 103
mkswap, 155-161
mkswap command, 159, 310-313
models, split-driver model, 322-323
modprobe command, 286, 295, 310
modules, loading kernel without, 190
mount command, 298
mountable file systems, 131
mounting
  disk images, 314
  partition images, 315
  partitions, 145
  root file system, 156
  system images. *See* booting
moving
  device drivers into DriverDomains, 412
  guest images, 141

multiple network interfaces
handling in driver domains, 394-396
handling in guest domains, 396-399
configuring guests with xm, 396-399
configuring multiple vifs to guests, 396
overview, 394
multiple system management, 518
mv command, 141

N

name= parameter (guest configuration files), 179, 542
name entry
/local/domain/<domId> directory, 70
/vm/<uuid> directory, 69
names, guest names, 185-186
Naming Your Virtual System window (virt-manager), 236
NAS (Network Attached Storage), 79
NAT (network address translation), 344
driver domain NAT interfaces, 376-379
guest domain NAT configuration, 375
guest domain NAT interface, 375-376
overview, 344, 373-375
testing results, 379-382
xend NAT mode configuration, 375
native virtualization, 7-8
NetBSD, 84, 530
Netfilter iptables
configuration, 414-415
example, 415-419
IP spoofing, 418
overview, 413-414
rules, 415-418
netmask= parameter (guest configuration files), 543
netmask directive, 354
netstat command
-ta option, 409-410
-t option, 409
-tua option, 410-411
network-accessible storage, 489
network-attach option (xm command), 397-398, 534
Network Attached Storage (NAS), 79
network address translation. *See* **NAT.**

network configuration
AoE (ATA over Ethernet)
client setup, 295-296
overview, 293
Vblade, 293-295
backend vifs (virtual network interfaces), 347-349
bridging, 344-347
default configurations, 355-356
Domain0 network interfaces, 359
driver domain bridging interfaces, 356-359
guest domain bridging interfaces, 360-361
overview, 343-355
testing results, 361-363
comparison of network storage options, 300
Domain0 requirements, 80
frontend vifs (virtual network interfaces), 347-349
guest configuration files
dhcp directive, 354
gateway directive, 354
netmask directive, 354
nics directive, 353
overview, 352
vif directive, 352-353
guest network parameters, 543
high-level steps, 349-350
initiators, 289-293
iSCSI Enterprise Target project, 293
Open-iSCSI project, 293
overview, 288-289
targets, 289-291
MAC (Media Access Control) addresses
overview, 342, 389-390
specifying for guests, 390-391
multiple network interfaces
handling in driver domains, 394-396
handling in guest domains, 396-399
overview, 394
NAS (Network Attached Storage), 79
NAT (network address translation)
driver domain NAT interfaces, 376-379
guest domain NAT configuration, 375
guest domain NAT interface, 375-376
overview, 344, 373-375
testing results, 379-382
xend NAT mode configuration, 375
network-accessible storage, 489
networking scripts, 189

overview, 339
purely virtual networks
dummy0, 383-385
dummy bridges, 385-389
overview, 382-383
references and further reading, 403-404
routing, 344-347
driver domain routing interfaces, 368-371
guest domain network interfaces, 367-368
guest domain routing mode configuration example, 367
overview, 364-365
testing results, 371-373
xend network-route script, 365-367
routing:xend routing mode configurations, 365
testing, 41-43
virtual network performance, 556-558
virtual network segments, 407
virtualization, 340-341
virtual network topology, 341-343
VNC (Virtual Network Computing), 207-209
vnet (virtual network)
configuring, 402
creating, 402-403
installing, 400-401
loading, 401
overview, 399-400
running, 401-402
xend configuration files
network scripts, 350-352
virtual network interface (vif) scripts, 350-352
xend network-related parameters, 539
xm network-related commands, 534
network-detach option (xm command), 397, 534
network-list option (xm command), 397-398, 534
network-route script, 365-367
network-script configuration option (xend), 66, 539
network scripts, 350-352
NFS (Network File System)
as root file system, 299-300
client configuration, 298-299
configuration example, 127
guest NFS parameters, 542
overview, 297
server configuration, 297-298
nfs_root= parameter (guest configuration files), 542
nfs_server= parameter (guest configuration files), 542
nics directive, 353
nmi=fatal boot option, 58
nmi=ignore boot option, 58
noapic boot option, 59
noirqbalance boot option, 57
nolapic boot option, 56
NoMachine NX, 80
non-Linux Domain0 installations, 114-115
nongraphical interaction with guests
overview, 201
SSH (secure shell), 204
xm console command, 202-203
nonuniform memory access computer (NUMA) hardware, 15
noop scheduler, 466
noreboot boot option, 58
nr_cpus attribute (xm info command output), 439
nr_nodes attribute (xm info command output), 440
NUMA (nonuniform memory access computer) hardware, 15

O

on_crash= parameter
guest configuration files, 544
/local/domain/<domId> directory, 70
/vm/<uuid> directory, 69
on_poweroff= parameter
guest configuration files, 544
/local/domain/<domId> directory, 70
/vm/<uuid> directory, 69
on_reboot= parameter
guest configuration files, 544
/local/domain/<domId> directory, 70
/vm/<uuid> directory, 69
OOM (out of memory) error, 187, 450
open-iscsi package, 291
Open-iSCSI project, 293
OpenBSD, 84
Open Connection dialog (virt-manager), 234
OpenSolaris, 84
open source distributions (Xen), 83-84
open source licenses, 24
OpenSUSE, 84
Domain0 installation, 88-90
YaST Virtual Machine Management, 229-233
OpenVZ, 23-24

operating system kernels, 123
operating system level virtualization
advantages/disadvantages, 13
definition, 9-11
Linux-VServer, 23
OpenVZ, 23-24
OSs (operating systems), choosing for Domain0, 59-60
ostype entry (/vm/<uuid>/image directory), 69
out of memory (OOM) error, 187, 450

P

-p value option (xm sched-sedf command), 534
Pacifica, 15
packages
customization, 268
iscsi-initiator-utils, 291
open-iscsi, 291
PAE (Physical Addressing Extensions), 82
paravirt_ops, 22
paravirtualiation, 320-321
advantages/disadvantages, 12
definition, 8
lguest, 22
paravirt_ops, 22
paravirtualized guest population. *See* PV guest population
QEMU, 334-335
UML (User-Mode Linux), 21
partitions
checking contents of, 145
creating with fdisk, 151-154
creating with GParted, 147-150
formatting with mkfs, 154-155
formatting with mkswap, 155-161
images, 129, 315
initializing for LVs, 279-280
listing, 143
mounting, 145
virtual partitions
associating, 310
creating, 308-309
detaching, 311
formatting, 310-311
--path=directory list option (xm create command), 533
paths to configuration files, 181-182
Paused state (domains), 171
--pause option
xm create command, 533
xm command, 200, 533
pausing domains, 199-200
pcap library, 419-420
pci= parameter (guest configuration files), 543
pciback.hide kernel parameter, 327-328
PCI devices
device emulation with QEMU-DM, 334-335
exclusive device access, 331-332
granting to another domain, 329-331
guest parameters, 543
hiding from Domain0 at boot, 327-328
identifying, 326-327
manually binding/unbinding at runtime, 328-329
trusted driver domains, 332-334
pci parameter (xm create command), 329-330
performance evaluation
CPU utilization
logical CPUs, 458
virtual CPUs (VCPUs), 458-463
measurements
overview, 546
repeatability of test results, 546-548
virtual Web hosting, 548-550
XenoLinux versus native Linux, 550-551
Xen on x86 versus IBM zServer, 551-553
performance isolation, 463-464, 553-556
intensive resource consumer applications, 554
well-behaving versus misbehaving VMs, 554-555
published papers, 545
Xen virtual network, 556-558
performance isolation, 463-464, 553-556
intensive resource consumer applications, 554
well-behaving versus misbehaving VMs, 554-555
--period=ms value option (xm sched-sedf command), 534
PEs (physical extents), 278
phy: prefix (disk parameter), 124
Physical Addressing Extensions (PAE), 82
physical extents (PEs), 278
physical volumes (PVs), 278
ping command, 361
pinning, 459
platform_params attribute (xm info command output), 441

populating guest images
HVM (Hardware Virtual Machine) guest population
automated population with virt-install, 225-228
overview, 218
populating from discs or disc images, 218-225
LVs (logical volumes), 281
PV (paravirtualized) guest population
debootstrap, 242-246
domi scripts, 252-256
overview, 228-229
quickpkg, 246-251
virt-manager, 233-241
Xen Express, 256-266
YaST Virtual Machine Management, 229-233
references and further reading, 274-276
root file system, 156
Portage, 106
ports, 409-411
port subdirectory (/local/domain/<domId> directory), 71
power management
advantages of virtualization, 5
Domain0, 81
preallocated image files, 129
prebuilt system images
comparison of image types, 161
compressed file system images, 128
creating partitions with fdisk, 151-154
creating partitions with GParted, 147-150
formatting partitions with mkfs, 154-155
formatting swap space with mkswap, 155-161
overview, 146-147
references and further reading, 163-164
converting from other virtualization platforms, 161-162
disk images, 129
downloading
from jailtime.org, 130
from rPath, 131
from Virtual Appliances, 131
mounting and booting, 131
boot methods, 132
disk images, 142-146
external Xen compatible kernels, 136-142
mounting and booting:pygrub bootloader, 132-135
overview, 128
partition images, 129
preallocated image files, 129
sparse image files, 129
preparing for xm migrate
guest domain configuration, 489-491
network-accessible storage, 489
overview, 484
proximity of network sources and destinations, 488-489
version and physical resource requirements, 491
xend configuration, 485-488
Private IP, 392
privileged domain. *See* **Domain0**
privileges, listing information about, 173
processes, listing, 61
processors, Domain0 requirements
AMD-V, 77-78
HVM, 78
Intel VT, 77
x86 processors, 76-77
programmatic interfaces
legacy interfaces to xend, 502
libvirt, 500-501
overview, 500
Xen-CIM, 501
Xen API, 501
protection rings, 50, 51
protocols. *See specific protocols*
proximity of network sources and destinations, 488-489
ps command, 61
purely virtual networks
dummy0
configuring, 383-385
testing, 385
dummy bridges
configuring, 385-387
testing, 388-389
overview, 382-383
PV (paravirtualized) guest population
debootstrap, 242-246
domi scripts, 252-256
booting guests, 253
building guests, 253-256
configuration files, 253-254
dependencies, 252
Device-mapper, 252
installing, 252
overview, 228-229
quickpkg, 246-251

virt-manager, 233-241
Allocate Memory and CPU window, 238
Assigning Storage Space window, 237
Create a New Virtual System window, 234-235
Creating Virtual Machine dialog, 239
installation screen, 240-241
main window, 234-235
Naming Your Virtual System window, 236
Open Connection dialog, 234
Ready to Begin Installation window, 239
Virtual Machine Console, 239-240
Xen Express, 256-266
YaST Virtual Machine Management, 229-233
pvcreate command, 279
PVs (physical volumes), 278
pygrub bootloader, 132-135
Python configuration scripts, 178-179, 184

Q

qcow (QEMU Copy-On-Write), 125
QEMU, 19
Copy-On-Write (qcow), 125
device emulation with QEMU-DM, 334-335
Quantifying the Performance Isolation Properties of Virtualization Systems (paper), 545-554
quickpkg command, 246-251

R

ramdisk= parameter
guest configuration files, 179, 542
/vm/<uuid>/image directory, 69
RBAC (Role-Based Access Control), 423
rdesktop, 213-214
Ready to Begin Installation window (virt-manager), 239
rebooting guests, 196-198
reboot option (xm command), 196-198, 533
recovery, 4
Red Hat Enterprise Linux 5 (RHEL 5), 86
Red Hat Linux resources, 530
reducing LVs (logical volumes), 284-285
release attribute (xm info command output), 439
relocating guests
cold static relocation, 480-481
live migration, 482-496
overview, 479-480
preparation
guest domain configuration, 489-491
network-accessible storage, 489
overview, 484
proximity of network sources and destinations, 488-489
version and physical resource requirements, 491
xend configuration, 485-488
references and further reading, 498
warm static migration, 481, 492-494
xm migrate command, 491-492
live migration, 494-496
migration errors, 497
warm static migration, 492-494
remote access, limiting, 408-412
Remote Desktop (rdesktop), 213-214
removing
iptables rules, 417
services from Domain0, 407-408
software from Domain0, 407-408
xenstored directory contents, 61
repeatability of Xen performance test results, 546-548
reporting bugs, 524
repositories
Bugzilla, 524
Mercurial, 527
resize2fs command, 283
resizing LVs (logical volumes), 282
extending LVs, 283-284
reducing LVs, 284-285
Resizing Memory with Balloons and Hotplug (paper), 452
resources
academic papers and conferences, 528-529
bug reporting, 524
distribution-specific resources, 530
resource management, 534
Xen Community site, 522
Xen mailing lists, 524
Xen Source code, 526-528
Xen Summit meetings, 525-526
XenWiki, 523
resources option (xm command), 535
restart argument (xend daemon), 61
restarting xend (Xen Daemon), 61
Restore failed error message, 478
restoring guest state, 476-478

restricting hosts, 486-487
RHEL 5 (Red Hat Enterprise Linux 5), 86
ring 0, 50
ring 1, 50
ring 2, 50
ring 3, 50
ring-ref subdirectory (/local/domain/<domId> directory), 71
rm command, 159
rmlabel option (xm command), 535
Role-Based Access Control (RBAC), 423
root= parameter (guest configuration files), 542
root file systems, NFS (Network File System), 299-300
root option (configuration files), 180
/root/centos-conf configuration file, 34
/root/deb-conf configuration file, 34
/root/opensuse-conf configuration file, 34
Rosemblum, Mendel, 15
routing, 344-347
- driver domain routing interfaces, 368-371
- guest domain network interfaces, 367-368
- guest domain routing mode configuration example, 367
- overview, 364 -365
- testing results, 371-373
- xend network-route script, 365-367
- xend routing mode configurations, 365

rPath, 131
rules (iptable)
- adding, 415-417
- IP spoofing, 418
- listing, 417
- removing, 417
- rule sets, 418

running
- Snort, 420-422
- virtual machines only when needed, 434
- vnet (virtual network), 401-402
- Xen LiveCD, 28-29
- XenMan, 514-518
- xm command from non-root account, 191

running entry (/local/domain/<domId> directory), 70
Running state (domains), 171
runtime
- managed runtime, 11
- manually binding/unbinding at, 328-329

Russel, Rusty, 22

S

-s option (xm uptime command), 532
-s value option (xm sched-sedf command), 534
S-Expression (SXP) configuration scripts, 180-181
saving guest state to file, 474-478
sched-credit option (xm command), 534
sched-sedf option (xm command), 534
sched=credit boot option, 57
Scheduled state (domains), 171
schedulers
- anticipatory scheduler, 467
- changing, 468-469
- complete fair queuing scheduler, 467
- CREDIT, 56
- credit scheduler
    - *isolation between guests, 464*
    - *overview, 463*
    - *weight and cap, 463-464*
    - *xm sched-credit command, 465*
- deadline scheduler, 466
- displaying information about, 467-468
- noop scheduler, 466
- SEDF, 56

scripts
- domi scripts, 252-256
    - *booting guests, 253*
    - *building guests, 253-256*
    - *configuration files, 253-254*
    - *dependencies, 252*
    - *Device-mapper, 252*
    - *installing, 252*
- network-route, 365-367
- networking scripts, 189, 350-352
- virtual network interface (vif) scripts, 350-352
- xendomains, 192

SDL (Simple DirectMedia Layer), 210-213
Secure Hypervisor Access Control Module. *See* **sHype**
secure shell (SSH)
- overview, 204
- X forwarding, 205-207

security
- Domain0
    - *limiting remote access, 408-412*
    - *limiting user access, 412*
    - *moving device drivers into DriverDomains, 412*
    - *removing software and services, 407-408*

DomU
backing up VM images, 434-435
overview, 433-434
running VMs only when needed, 434
iptables
configuration, 414-415
example, 415-419
IP spoofing, 418
overview, 413-414
rules, 415-418
overview, 405
privileges, listing information about, 173
references and further reading, 436
sHype
adding to Xen, 424-425
configuring sHype policy, 425-432
downloading, 424
overview, 423-424
Snort
building, 419-420
downloading, 419-420
intrusion detection mode, 420
overview, 419
running, 420-422
special purpose virtual machines, 406
virtual network segments, 407
xm security-related commands, 535
XSM (Xen Security Modules), 432-433
SEDF scheduler, 56
serve option (xm command), 536
servers
AoE Vblade, 293-295
DHCP servers
external servers, 392
internal servers, 393-394
Freenx, 212-213
iSCSI targets, 289-291
NFS (Network File System), 297-298
VNC servers:starting, 208
xend server-related parameters, 538
services
customization, 268
removing from Domain0, 407-408
shadow page tables, 451
shutdown option (xm command), 193-196, 533
Shutdown state (domains), 171
shutting down guests
overview, 193
xm destroy command, 198-199
xm reboot command, 196-198
xm shutdown command, 193-196
sHype
adding to Xen, 424-425
configuring sHype policy, 425-432
downloading, 424
overview, 423-424
Simple DirectMedia Layer (SDL), 210-213
SLES (SUSE Linux Enterprise Server), 86
--slice=ms value option (xm sched-sedf command), 534
smart devices, 336
Snort
building, 419-420
downloading, 419-420
intrusion detection mode, 420
overview, 419
running, 420-422
sockets_per_node attribute (xm info command output), 440
Solaris resources, 530
SOSP (Symposium on Operating Systems Principles), 16
source code (Xen), 526-528
Source Forge, 514
sparse image files, 129
special purpose virtual machines, 406
split-driver model, 322-323
spoofing (IP), 418
SSH (secure shell)
overview, 204
X forwarding, 205-207
ssidref entry (/vm/<uuid> directory), 69
stability, improving with swap partitions, 454
start argument (xend daemon), 60
starting
DomainU, 191-192
VNC servers, 208
xend (Xen daemon), 60
state
of domains, 171
guest state
possible save and restore errors, 478
restoring, 476-478
saving to file, 474-478
of virtual machines:representing, 472-473
state item (XenStore), 399
status argument (xend daemon), 62
stop argument (xend daemon), 61

- **stopping xend (Xen daemon), 61**
- **storage**
  - AoE (ATA over Ethernet)
    - *client setup, 295-296*
    - *overview, 293*
    - *Vblade, 293-295*
  - comparison of storage mechanisms, 300, 315-316
  - iSCSI
    - *initiators, 289-293*
    - *iSCSI Enterprise Target project, 293*
    - *Open-iSCSI project, 293*
    - *overview, 288-289*
    - *targets, 289-291*
  - LVs (logical volumes)
    - *configuration files, 281*
    - *CoW (copy on write), 286-287*
    - *creating, 280*
    - *creating VGs for, 280*
    - *extending, 283-284*
    - *initializing partitions for, 279-280*
    - *LVMs (logical volume managers), 278-279*
    - *populating, 281*
    - *reducing, 284-285*
  - NFS (Network File System)
    - *client configuration, 298-299*
    - *overview, 297*
    - *as root file system, 299-300*
    - *server configuration, 297-298*
  - overview, 277, 287-288
  - PVs (physical volumes), 278
  - references and further reading, 316-318
  - system image files
    - *compressed tar image files, 301-302*
    - *disk image files, 302-311*
    - *guest partition image files, 312-313*
    - *mounting disk and partition images, 314-315*
    - *overview, 301*
- **Sun Java Virtual Machine, 11**
- **SUSE Linux resources, 530**
- **SUSE Linux Enterprise Server (SLES), 86**
- **swap files, deleting, 159**
- **swap partitions, 155-161, 454**
- **SXP (S-Expression) configuration scripts, 180-181**
- **SXP format, 329**
- **Symposium on Operating Systems Principles (SOSP), 16**
- **sync_console boot option, 58**
- **sysrp option (xm command), 533**
- **System/370 (IBM), 14**
- **system images**
  - converting existing installations, 270-274
  - converting from other virtualization platforms, 161-162
  - customization
    - *file system table, 268-269*
    - *hostnames, 266-267*
    - *packages/services, 268*
    - *users, 267-268*
  - definition, 122-123
  - HVM (Hardware Virtual Machine) guest population
    - *automated population with virt-install, 225-228*
    - *overview, 218*
    - *populating from discs or disc images, 218-225*
  - prebuilt system images
    - *comparison of image types, 161*
    - *compressed file system images, 128, 146-161*
    - *disk images, 129*
    - *downloading, 130-131*
    - *mounting and booting, 131-146*
    - *overview, 128*
    - *partition images, 129*
    - *preallocated image files, 129*
    - *references and further reading, 163-164*
    - *sparse image files, 129*
  - PV (paravirtualized) guest population
    - *debootstrap, 242-246*
    - *domi scripts, 252-256*
    - *overview, 228-229*
    - *quickpkg, 246-251*
    - *virt-manager, 233-241*
    - *Xen Express, 256-266*
    - *YaST Virtual Machine Management, 229-233*
  - references and further reading, 274-276
  - storing. *See* storage
  - system image files
    - *compressed tar image files, 301-302*
    - *disk image files, 302-311*
    - *guest partition image files, 312-313*
    - *mounting disk and partition images, 314-315*
    - *overview, 301*

T

- **-ta option (netstat command), 409-410**
- **-t option (netstat command), 409**
- **tap:aio prefix (disk parameter), 125**
- **tar command, 301-302**

tar files, 301-302
.tar.gz file extension, 302
targets (iSCSI), 289-291
target subdirectory (/local/domain/<domId> directory), 71
tbuf_size=X boot option, 58
TCO (total cost of ownership), 4
TCP (Transmission Control Protocol), 409-411
testing
- bridging configuration, 361-363
- dummy0, 385
- dummy bridges, 388-389
- NAT (network address translation) configuration, 379-382
- networking, 41-43
- routing configuration, 371-373

thrashing, 454
threads local storage, 104
threads_per_core attribute (xm info command output), 440
TLB (translation lookaside buffer), 77
topology (networks), 341-343
top option (xm command), 532
total_memory attribute (xm info command output), 441
total cost of ownership (TCO), 4
TPM parameters, 543
translation lookaside buffer (TLB), 77
trusted driver domains, 332-334
tty subdirectory (/local/domain/<domId> directory), 71
-tua option (netstat command), 410-411
tx-/rx-ring-ref item (XenStore), 399

U

Ubuntu, 84
- debootstrap, 242-246
- Domain0 installation, 98-101
- LTS (Long Term Support), 99
- resources, 530
- version numbers, 99

udev device node, 292
UML (User-Mode Linux), 21
umount command, 315
Unable to read file (error message), 479
uname command, 97
unbinding PCI devices at runtime, 328, 329
unmasking xen-source packages, 108
unmounting
- guest images, 140
- root file system, 159

unpause option (xm command), 200-201, 533
unpausing domains, 200-201
update-rc.d command, 104
uptime option (xm command), 532
User-Mode Linux (UML), 21
user access, limiting, 412
user customization, 267, 268
User's Manual, 529
User Mode Emulation (QEMU), 19
uuid= parameter
- guest configuration files, 542
- /vm/<uuid> directory, 69

V

Vanderpool, 15
van Doorn, Leendert, 78
/var/lib/xenstored/tdb file, 68
VBDs (virtual block devices), 511
Vblade, 293-295
vbladed, 293
vcpu_avail entry (/vm/<uuid> directory), 69
vcpu-list option (xm command), 535
vcpu-pin option (xm command), 535
vcpu-set option (xm command), 535
VCPUs (virtual CPUs)
- HVM VCPU management, 459
- manual administration, 462-463
- overview, 458-459
- xm vcpu-list command, 460-461
- xm vcpu-pin command, 462
- xm vcpu-set command, 461

vcpus= parameter (guest configuration files), 543
vcpus entry (/vm/<uuid> directory), 69
version attribute (xm info command output), 439
vgcreate command, 280
VGs (groups), 278-280
viewing
- guest information
  - *xm info command, 438-443*
  - *xm log command, 444-445*
  - *xm top command, 446-449*
  - *xm uptime command, 449*
- hypervisor system messages, 443-444
- scheduler information, 467-468
- statistics on running guests, 446-449
- virtual guests, 32

vif= parameter (guest configuration files), 179, 543
vif-script configuration option (xend), 66, 539
vif directive, 352-353
vifs (virtual network interfaces)
  frontend/backend network drivers, 347-349
  scripts, 350-352
  visibility, 343
virt-install, 94, 225-228
virt-manager, 94, 233-241, 509-513
  Allocate Memory and CPU window, 238
  Assigning Storage Space window, 237
  capabilities, 509-512
  command-line options, 513
  compilation procedure, 512
  Create a New Virtual System window, 234-235
  Creating Virtual Machine dialog, 239
  installation screen, 240-241
  main window, 234-235
  Naming Your Virtual System window, 236
  Open Connection dialog, 234
  Ready to Begin Installation window, 239
  Virtual Machine Console, 239-240
Virtual Appliances, 131
virtual block devices (VBDs), 511
virtual CPUs. *See* **VCPUs**
virtual devices
  allocating, 303-307
  configuring, 307
Virtual Disk Migration Utility, 162
virtual frame buffers, 80, 210-213
virtual guests, viewing, 32
Virtual Iron Software, Inc., 504-506
  Virtualization Manager, 85, 505
  Virtualization Services, 85, 505
virtualization
  application virtualization
    advantages/disadvantages, 13
    definition, 11
  backend/frontend architecture
    backend information in XenStore, 323-325
    frontend information in XenStore, 325-326
    overview, 322
    split-driver model, 322-323
    XenBus, 323
  bare-metal virtualization systems, 18
  benefits, 3-5
  commodity virtualization technologies, 15
    Bochs, 18
    comparison of virtualization products, 24-25
    lguest, 22
    Linux-VServer, 23
    Linux KVM (Kernel Virtual Machine), 20-21
    Microsoft Hyper-V, 20
    Microsoft Virtual PC, 20
    OpenVZ, 23-24
    paravirt_ops, 22
    QEMU, 19
    UML (User-Mode Linux), 21
    VMware, 19
  emulation
    advantages/disadvantages, 12
    Bochs, 18
    definition, 6
    QEMU, 19
  full virtualiation, 321
    advantages/disadvantages, 12
    definition, 7-8
    Linux KVM (Kernel Virtual Machine), 20-21
    Microsoft Hyper-V, 20
    Microsoft Virtual PC, 20
    VMware, 19
  history of, 13
    commodity hardware, 15
    IBM mainframes, 14
    time line, 14
    x86 virtualization extensions, 15
    Xen origins and time line, 15-17
  hosted virtualization systems, 18
  library virtualization
    advantages/disadvantages, 13
    definition, 11
  networks, 340-341
  no virtualiation, 321
  operating system level virtualization
    advantages/disadvantages, 13
    definition, 9-11
    Linux-VServer, 23
    OpenVZ, 23-24
  overview, 2-6
  paravirtualiation, 320-321
    advantages/disadvantages, 12
    definition, 8
    lguest, 22
    paravirt_ops, 22
    QEMU, 334-335
    UML (User-Mode Linux), 21
  references and further reading, 26
Virtualization Manager, 85, 505

Virtualization Services, 85, 505
Virtualization Technology (VT), 15, 77
Virtual Machine Console, 239-240
Virtual Machine Extensions (VMX) root operation mode, 77
Virtual Machine Interface (VMI), 22
Virtual Machine Manager. *See* **virt-manager**
virtual machine monitors. *See* **hypervisors**
virtual machines
  images, backing up, 434-435
  representing state of, 472-473
  running only when needed, 434
  special purpose virtual machines, 406
virtual network. *See* **vnet**
Virtual Network Computing, 38, 80, 207-209
virtual networking interface connections (VNICs), 511
virtual network interfaces. *See* **vifs**
virtual networks
  performance, 556-558
  segments, 407
  topology, 341-343
virtual partitions
  associating, 310
  creating, 308-309
  detaching, 311
  formatting, 310-311
Virtual PC (Microsoft), 20
virtual Web hosting, 548-550
/vm/<uuid> directory, 68-69
/vm/<uuid>/image directory, 69
VMcasting, 507
vm entry (/local/domain/<domId> directory), 70
VMI (Virtual Machine Interface), 22
VMs. *See* **virtual machines**
VMware, 19
VMX (Virtual Machine Extensions) root operation mode, 77
VNC (Virtual Network Computing), 38, 80, 207-209
vnc-listen parameter (xend), 539
VNC/SDL libraries, 210-213
vncpasswd parameter (xend), 539
vnet (virtual network)
  configuring, 402
  creating, 402-403
  installing, 400-401
  loading, 401
  overview, 399-400
  running, 401-402
  xm Vnet-related commands, 536
vnet-create option
  xm command, 536
  xm vnet command, 403
vnet-delete option
  xm command, 536
  xm vnet command, 403
vnet-list option
  xm command, 536
  xm vnet command, 403
VNICs (virtual networking interface connections), 511
volume groups (VGs), 278-280
volumes
  LVs (logical volumes), 278-279
    configuration files, 281
    CoW (copy on write), 286-287
    creating, 280
    creating VGs for, 280
    extending, 283-284
    initializing partitions for, 279-280
    populating, 281
    reducing, 284-285
  PVs (physical volumes), 278
VT (Virtualization Technology), 15, 77
VT-d, 77
VT-i, 77
VT-x, 77
vtpm-list option (xm command), 532
vtpm= parameter (guest configuration files), 543

W

-w option
  xm reboot command, 533
  xm shutdown command, 533
-w value option (xm sched-sedf command), 534
warm static migration, 481, 492-494
watchdog boot option, 58
Web hosting, 548-550
web resources. *See* **resources**
--weight=float value option (xm sched-sedf command), 534
--weight=integer value option (xm sched-credit command), 534
weights, 463-464
Wiki, 78, 523

Windows XP HVM (Hardware Virtual Machine) guest population, 218-225
- configuration files, 219-225
- dd command, 218
- display configuration, 220-221
- xm create command, 221

Wine library, 11
writable persistent snapshots, 286

X

X11DisplayOffset parameter (SSH), 206
X11Forwarding parameter (SSH), 205
X11UseLocalhost parameter (SSH), 205
x86
- protection rings, 50-51
- virtualization extensions, 15
- Xen-compatible processors, 76-77
- Xen on x86 versus IBM zServer, 551-553

XauthLocation parameter (SSH), 206
.Xauthority file, 207
Xen
- definition, 1
- distributions
  - *enterprise class Linux support, 85-86*
  - *open source distributions, 83-84*
  - *Virtual Iron Software, Inc., 85*
  - *XenServer Express Edition, 85*
- origins and time line, 15-17

xen_caps attribute (xm info command output), 441
xen_changeset attribute (xm info command output), 441
Xen-CIM, 501
xen_extra attribute (xm info command output), 441
xen_major attribute (xm info command output), 441
xen_minor attribute (xm info command output), 441
Xen-oprofile, 17
xen_pagesize attribute (xm info command output), 441
Xen and the Art of Repeated Research (paper), 545-546
Xen and the Art of Virtualization (paper), 546
XenAPI, 17, 501
XenBus, 323
Xen Community site, 522
xencons=off boot option, 59
xencons=tty boot option, 59
XenConsole, 61
xend (Xen daemon)
- checking status of, 166
- configuration, 63-67, 485-488
  - *configuration file for Domain0_Host1, 487*
  - *configuration file for Domain0_Host2, 488*
  - *configuration parameters, 537-539*
  - *console-limit option, 66*
  - *dom0-cpus option, 65*
  - *dom0-min-mem option, 65*
  - *enable-dump option, 64*
  - *external-migration-tool option, 65*
  - *logfile option, 63*
  - *loglevel option, 64*
  - *network-script option, 66*
  - *sample xend-config.sxp file, 66-67*
  - *vif-script option, 66*
  - *xend-address option, 64*
  - *xend-http-server option, 64*
  - *xend-port option, 64*
  - *xend-relocation-address option, 65*
  - *xend-relocation-hosts-allow option, 65*
  - *xend-relocation-port option, 65*
  - *xend-relocation-server option, 65*
  - *xend-tcp-xmlrpc-server option, 64*
  - *xend-unix-server option, 64*
  - *xend-unix-xmlrpc-server option, 64*
- controlling, 60-62
- error messages, 167
- log files, 62-63, 444-445
- NAT mode configuration, 375
- network-route script, 365-367
- overview, 60
- querying current condition, 62
- restarting, 61
- routing mode configurations, 365
- starting, 60
- stopping, 61
- xm Xend-related commands, 536

xend-address configuration option (xend), 64, 538
xend_config_format attribute (xm info command output), 443
xend-config.sxp file, 66-67
Xen daemon. *See* **xend**
xend-debug.log file, 445
xend-hotplug.log file, 445

xend-http-server configuration option (xend), 64, 538
xend-port configuration option (xend), 64, 538
xend-relocation-address configuration option (xend), 65, 539
xend-relocation-hosts-allow configuration option (xend), 65, 539
xend-relocation-port configuration option (xend), 65, 539
xend-relocation-server configuration option (xend), 65, 539
xend-tcp-xmlrpc-server configuration option (xend), 64, 538
xend-unix-path parameter (xend), 538
xend-unix-server configuration option (xend), 64
xend-unix-xmlrpc-server configuration option (xend), 64, 538
xend.log file, 445
xendomains script, 192
xendsxp file, 401
XenExpress
- installation, 112-114
- PV (paravirtualized) guest population, 256-266

Xen hypervisor. *See* **hypervisor**
Xen LiveCD
- Domain0 images, 30-31
- downloading LiveCD image, 29-30
- guests
  - *creating, 33-38*
  - *deleting, 38*
  - *interacting with, 38-41*
  - *number of, 44*
- login process, 31
- network testing, 41-43
- references and further reading, 45
- running, 28-29
- test system details, 28-29
- Xfce desktop, 32

Xen mailing lists, 524
XenMan, 513-518
- binary package installation, 514
- running, 514-518
- source-derived installation, 514

XenoLinux, 550-551
XenoServer project, 15
Xen Project Web site, 529
xensec_ezpolicy, 425
Xen Security Modules (XSM), 432-433
XenServer Express Edition, 85
XenSource, 16, 526-528
XenStore
- backend information in, 323-325
- commands, 71-72
- dumping contents of, 72-73
- frontend information in, 325-326
- /local/domain/<domId> directory, 70-71
- overview, 67-68
- /var/lib/xenstored/tdb file, 68
- /vm/<uuid>/image directory, 69
- /vm/<uuid> directory, 68-69
- xenstore-list, 72

xenstore-chmod command, 72
xenstore-control command, 72
xenstore-exists command, 72
xenstore-list command, 72
xenstore-read command, 72
xenstore-rm command, 72
xenstore-write command, 72
xenstored, killing, 61
xenstored directory, removing contents of, 61
Xen Summit meetings, 525-526
xentop command, 32
Xen User's Manual, 403, 529
XenWiki, 78, 523
Xfce desktop, 32
X forwarding with SSH, 205-207
xm command
- overview, 166
- prerequisites, 166-167
- running from non-root account, 191
- syntax, 167-168, 531
- xm addlabel, 535
- xm block-attach, 532
- xm block-detach, 532
- xm block-list, 532
- xm cfgbootpolicy, 535
- xm console, 202-203, 533
- xm create, 33-37, 533
  - *booting Debian guest, 142*
  - *booting Fedora guest, 242*
  - *booting HVM guests, 221*
  - *booting openfiler guest, 134, 160*
  - *--c option, 183-184*
  - *console output, 183-184*

creating guests with, 175-178
--dryrun option, 182-183
overview, 174
pci parameter, 329, 330
prerequisites, 174-175
xm destroy, 38, 198-199, 533
xm dmesg, 443-444, 536
xm domid, 532
xm domname, 532
xm dump-core, 532
xm dumppolicy, 535
xm help, 532
xm info, 438-443, 536
xm labels, 535
xm list, 97, 105, 169, 533
--label option, 173
listing basic guest information, 169-170
listing information about specific guests, 171-172
--long option, 172-173
xm loadpolicy, 535
xm log, 444-445, 536
xm makepolicy, 535
xm mem-max, 457, 534
xm mem-set, 454-457, 534
xm migrate
live migration, 494-496
migration errors, 497
preparation, 484-491
syntax, 491-492
warm static migration, 492-494
xm network-attach, 397-398, 534
xm network-detach, 397, 534
xm network-list, 397-398, 534
xm pause, 196-200, 533
xm reboot, 533
xm resources, 535
xm restore, 476-478
xm rmlabel, 535
xm save, 474-476
xm sched-credit, 534
xm serve, 536
xm shutdown, 193-196, 533
xm sysrp, 533
xm top, 446-449, 532
xm unpause, 200-201
xm uptime, 449, 532
xm vcpu-list, 460-461, 535
xm vcpu-pin, 462, 535
xm vcpu-set, 461
xm vnet, 402-403
xm vnet-create, 536
xm vnet-delete, 536
xm vnet-list, 536
xm vtpm-list, 532
xsls command, 72
XSM (Xen Security Modules), 432-433
Xu, Herbert, 349

Y-Z

YaST Virtual Machine Management, 89, 229-233

zombie guests, 497
zServer, 551-553