Chapter 7

Scripting Hot Backups and Recovery for Virtual Machines

Topics in this chapter:

- Anatomy of a VM Backup
- Existing VM Backup Tools
- VMX Backups
- Backup and Restore Methodology

Introduction

You probably picked up this book because you need to automate some functions in your virtual infrastructure. Scripting is all about automating our menial tasks. And no menial task begs for automation more than regular backups. Fortunately, VMware provides a rich platform for effective backup and restore solutions that can be controlled through scripts. In this chapter, we will exploit the functionality provided by VMware ESX Server to perform hot (that is, live, while VM is running) backups of our virtual machines. We'll show how to back up the data files and config files. In addition, we'll veer a bit out of the command line and into some consultative topics. We'll discuss the whys and hows of recovery planning for a virtual infrastructure. This will help you decide how you should implement a solution using the scripts and technologies presented in this chapter.

Anatomy of a VM Backup

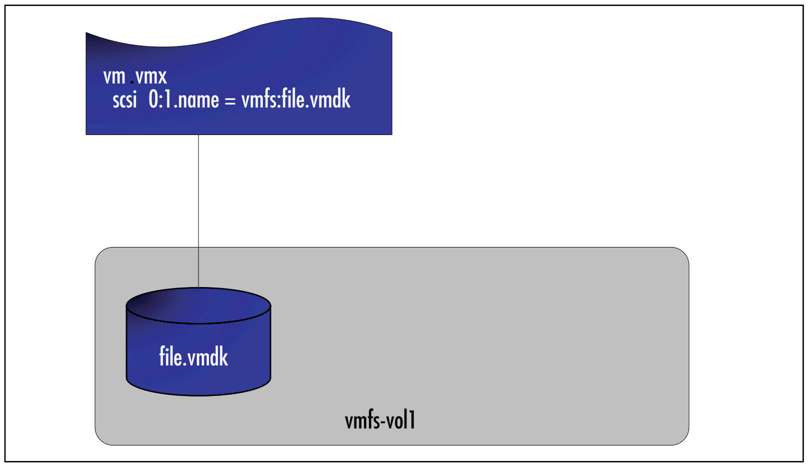
Before getting into details, it is important to briefly discuss the fundamentals of a VM backup. The feature of virtualization that enables disaster recovery backup is encapsulation. In the VMware ESX world, this is the virtual hard disk, or VMDK. A VMDK file contains the entire contents of a hard disk, the partitions, boot sector, files, everything. A VMDK takes the thousands of files involved in a typical OS and bundles them all together in one VMDK file. We have the ability to create a copy of a VMDK and use this as a complete backup, then treat the backup as we would any file, choosing where to store the file and for how long.

For the purposes of this chapter, we will assume that a VMDK stored on a VMFS volume is a type 2 file, and an exported VMDK is type 1. They are the only file types supported by ESX 2.x. For review, a type 2 file is a preallocated virtual disk, and type 1 is a growable virtual disk split into 2GB files.

Because the data inside a live VMDK file may be constantly changing, simply making a copy of a VMDK file will result in corruption without some additional technology. Now it is not practical for most organizations to power off a virtual machine prior to backup. Instead a REDO log may be placed on the VMDK file prior to making a copy. The VMDK is placed in append mode, and all changes are written to an alternate file. The REDO log file has the extension .REDO.

Let's walk through a visual representation of the high-level backup process referencing native tools shipped with ESX to perform the operation. Figure 7.1 is a virtual machine in its simplest form. A VMX file references a single VMDK file on a VMFS volume.

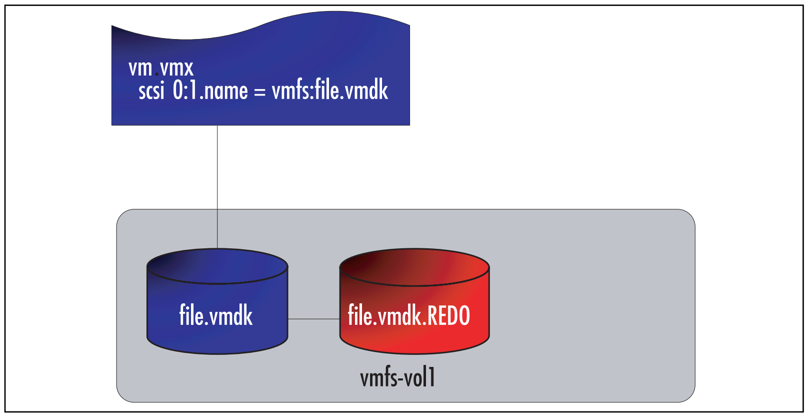
Figure 7.1 Normal State, Persistent Disk



The first step of the process is to create a REDO log on the VMDK. The command *vmware-cmd* provides a quick and easy way to create a REDO log (see Figure 7.2):

```
vmware-cmd /home/vmware/vm/vm.vmx addredo scsi0:1
```

Figure 7.2 REDO Is Applied to the VMDK



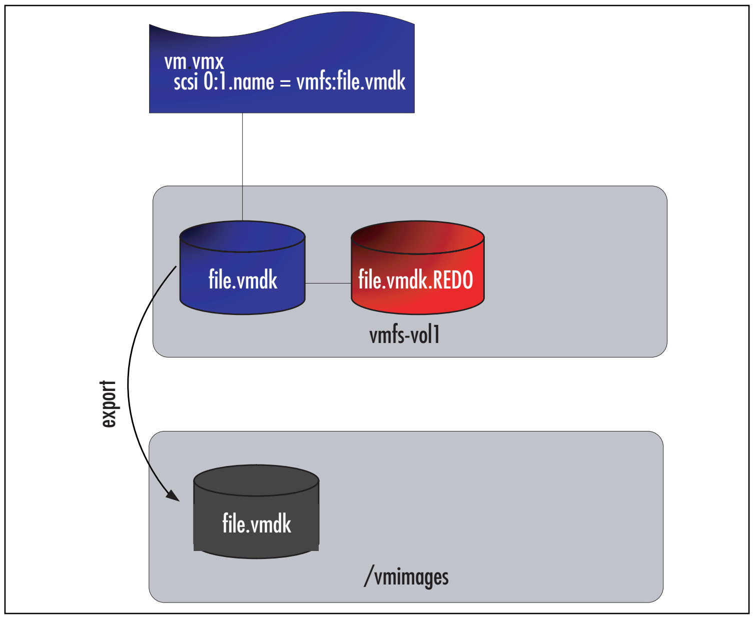
vmware-cmd is a command-line tool that ships with ESX and is for managing virtual machines. We are using one of many functions in this tool, addredo. The only argument to this function is the SCSI address of the VMDK in question. The command refers to the logical SCSI ID assigned to the disk file of the virtual machine, found in the VMX file. Don't confuse this SCSI ID with the physical SCSI ID of your hard disks or SAN LUNs. A number of ways exist to find the SCSI address, including Virtual Center, MUI, or the VMX. This command shows all SCSI lines in the VMX; only devices with the present flag set to TRUE are really there:

```
grep scsi /home/vmware/vm/vm.vmx
```

At this point, changes are being made to the REDO and the VMDK is static. You may now safely make a copy of this file. To keep things simple, we will export this VMDK to an ext3 filesystem (see Figure 7.3). Backup target options are discussed in more detail later in the chapter. The syntax of this command is a bit different than you might expect: vmkfstools –e <target> <source>. The result is a file on the ext3 /vmimages volume in a type 3 format.

```
vmkfstools -e /vmimages/file.vmdk /vmfs/vmfs-vol1/file.vmdk
```

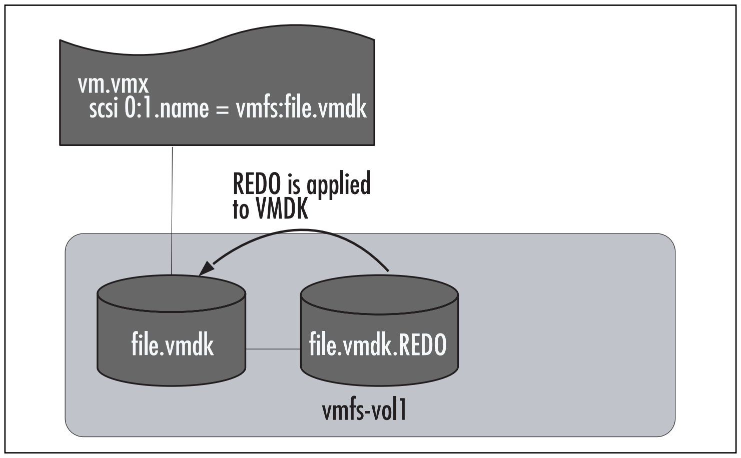
Figure 7.3 VMDK Is Exported



After the export is complete, your next step is to put things back into a normal operating state. This means applying all changes stored in the REDO file back to the VMDK file (see Figure 7.4). Again, *vmware-cmd* is the simplest tool to use.

```
vmware-cmd /home/vmware/vm/vm.vmx commit scsi0:1 0 1 1
```

Figure 7.4 REDO Is Committed



The syntax of this command is

```
vmware-cmd <cfg> commit <disk_device_name> <level> <freeze> <wait>
```

<level> only applies when you have more than one REDO. Actual usage of this option is covered in the "Layered REDO Logs" section of this chapter.

<freeze> is ignored and a freeze 0 is used unless <level> is 1.

<wait> 0 returns when commit begins; 1 returns after the commit is completed.

Limitations

It is important to discuss some of the limitations of this type of backup. The limitations include

- **Crash Consistent State** The most important thing to understand is that once a REDO log is placed on the disk file, the disk file is now in a crash consistent state. The guest operating system is not aware that this has happened. It is as reliable as pressing the power button on the machine, or crash consistent.

- **File-Level Recovery Challenges** Another limitation of this type of backup is the fact that doing a file-level restore can take a significant amount of time. The entire disk file must be restored and mounted somewhere before you can copy off the file in question. This is a function better left to an agent running inside the guest that is intended as a file-level recovery agent. File-level agents will also help with indexing, versioning and searching. If you must pursue a file-level restore without an agent, the VMware Diskmount utility is your friend. It will save you a significant amount of time mounting VMDK files and looking for the file in question.
- **Wall-Clock Time** We are talking about a significant amount of data here. Depending on the size of the environment, it may not be practical to copy entire VMDK files around on a regular basis. As your environment grows, you may be looking at a lack of wall clock time to accomplish your backups. Factors that will effect the time your backup takes are the amount of data inside a VMDK, the speed of the disk subsystem, available resources in the service console, and the type of transport used to move the backup data.
- **Performance Considerations** There are performance considerations when running with a REDO log. The REDO file grows 16MB at a time. Each file growth requires a SCSI reservation on the LUN. Also, the REDO log needs to be committed after you have a copy of the file. This will rewrite all changes back to the VMDK file. All of this activity requires CPU from the ESX service console and increases activity on the disk subsystem. Resources in the service console are generally limited to 1 CPU, < 1GB RAM, 1 NIC, and 1 SCSI/RAID device. Considering that this represents a fairly underpowered server, you will run into limitations when trying to do multiple concurrent backups. The available resources will likely limit you to 2–4 concurrent backups before the service console becomes too overloaded. Overloading the service console is very risky. If the service console crashes, so does the ESX server and all the VMs running

on it. Use caution, test, and fall on the side of conservatism when planning how many backups to do at once.

- **Frozen Disk Files** While the REDO log is being applied to the VMDK file, the disk is frozen, meaning I/O is halted. If the REDO is small, application of the REDO log is relatively quick. If you have been running with a REDO for some time, this frozen state may cause problems. The suggested way to approach this situation is to use a second REDO log on the VMDK, while the first is being applied. The method for applying this strategy is covered next.

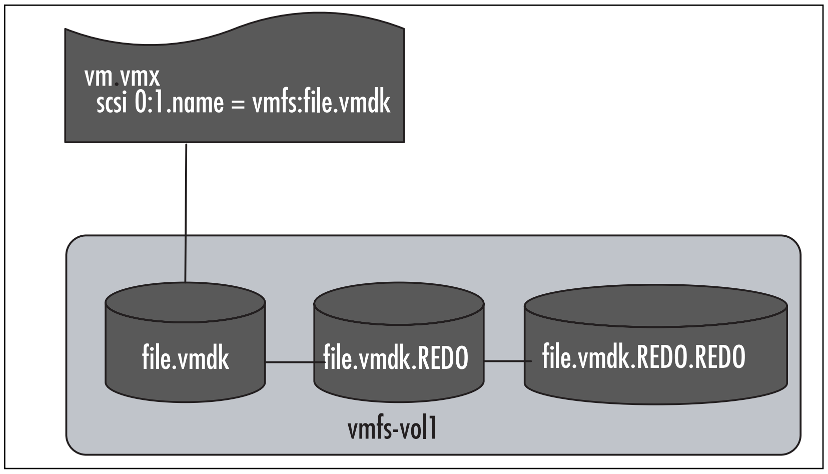
Layered REDO Logs

As mentioned, while the REDO log is being applied (committed), I/O to the VMDK is frozen. If your REDO file is large enough, users and applications will experience some problems due to the amount of time this takes. A common technique used to mitigate the risk of the commit taking too long is to use two REDO files. The freeze is only necessary while applying the last REDO log. As we pick up the previous walk-though of a backup, we will replace the final commit step with a slightly different process.

First, we add a second REDO log right after our export is completed. The syntax to add this second REDO is exactly like the first (see Figure 7.5):

```
vmware-cmd /home/vmware/vm/vm.vmx addredo scsi0:1
```

Figure 7.5 Second REDO Created



At this point, all transactions are written to the REDO.REDO file. We can commit the first REDO log to the VMDK using the following command.

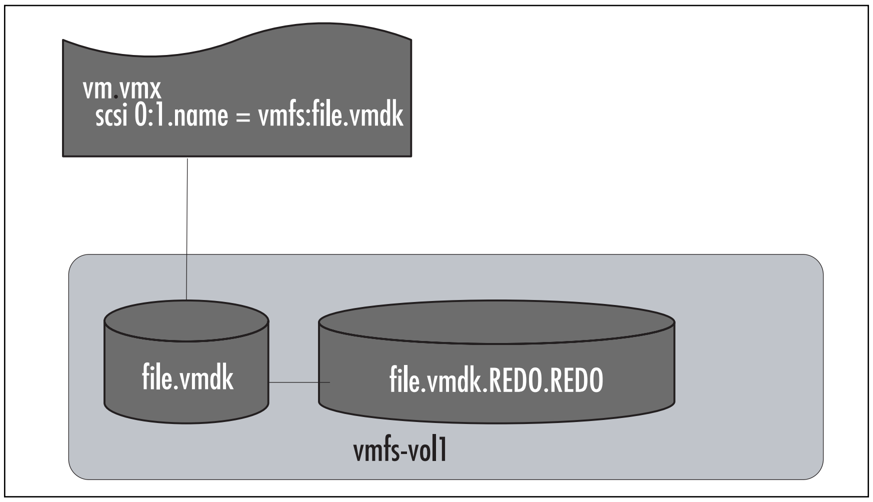
```
vmware-cmd /home/vmware/vm/vm.vmx commit scsi0:1 1 0 1
```

We give the commit command the following options:

- **<level> = 1** This tells ESX to only commit one of the two REDO logs.
- **<freeze> = 0** We will not freeze I/O to the VMDK while the commit is running.
- **<wait> = 1** Wait for the commit to complete before returning.

As seen in Figure 7.6, we are now in a familiar state with one REDO on the VMDK, except this one is hopefully smaller than the first.

Figure 7.6 First REDO Has Been Applied

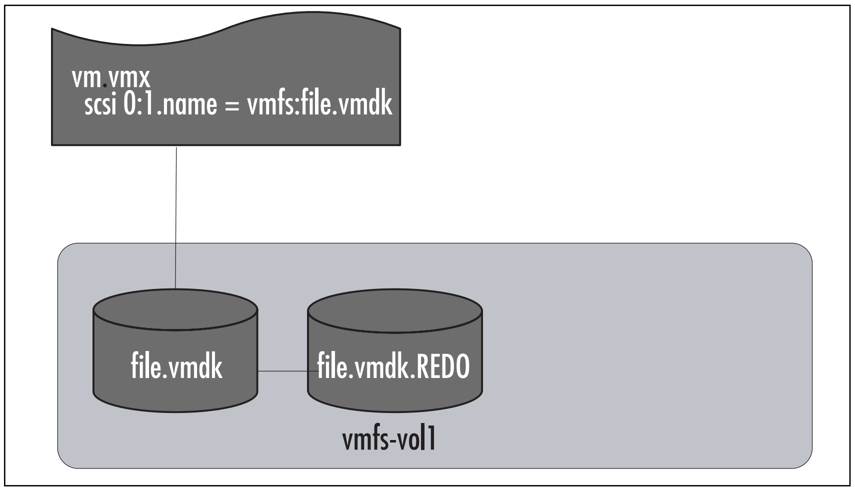


Finally, we will commit the remaining REDO file. Regardless of the freeze option chosen, we are now going to freeze the VMDK.

```
vmware-cmd /home/vmware/vm/vm.vmx commit scsi0:1 0 0 1
```

When complete, this command will leave you as you started. One VMDK and no REDO files (see Figure 7.7).

Figure 7.7 Backup Is Complete; Back to the Normal Operating State



Master Craftsman…

Detecting the Current Mode for a VM Disk

Before you start adding and committing REDO log files to running virtual machines, you need to know what state the current disk file is in. You want to be sure a disk file is in Persistent mode before going to work on starting a hot backup. We've included some code as part of this Master Craftsman tip that you can use to determine the current mode of your disk file.

```
#!/usr/bin/perl -w
#
# This script is an example only
# Usage: detectDiskMode.pl <vmxConfigFile> <scsiDisk>
#
# Example: detectDiskMode.pl /home/vmware/vm/vm.vmx scsi0:1

use VMware::VmPerl;
use VMware::VmPerl::ConnectParams;
use VMware::VmPerl::VM;
```

Continued

```
use strict;

# User variables
my ($cfg, $disk) = @ARGV;

# Connect to the virtual machine
my $params = VMware::VmPerl::ConnectParams::new();
my $vm = VMware::VmPerl::VM::new();
$vm->connect($params, $cfg);

# Retrieve the mode of the disk in question
my $mode = $vm->get_config("$disk.mode");

if ($mode ne "persistent") {
  print "Warning: $mode\n";
} else {
  print "$mode\n";
} # End if not persistent

$vm->disconnect();
```

Hot VM Backup Sample Script

Using the preceding information, you could put together a quick shell script to run a hot backup. Now, we can pull together all of the concepts shown earlier, except we'll use Perl as the scripting language this time. The following script does exactly what was discussed previously, but processes all disk files for the VM in order. This script has the following objectives:

- The only command-line option is to provide the path to the virtual machine VMX file (required).
- Script will find all VMDK files attached to the virtual machine.
- Process each VMDK, one at a time.

- Apply a REDO log to the VMDK.
- Vmkfstools export on the VMDK.
- Apply a second REDO log.
- Commit the first REDO log.
- Commit the final REDO log.

This script shown in Code Listing 7.1 is an example only and should not be used in a production environment. It lacks user feedback and error checking/reporting.

Code Listing 7.1 Perl Script for Running a Hot Backup of a VM

```
#!/usr/bin/perl -w
#
# This script is an example only
# Usage: simpleBackup.pl <vmxPath>

use VMware::VmPerl;
use VMware::VmPerl::Server;
use VMware::VmPerl::ConnectParams;
use VMware::VmPerl::VM;
use strict;

# User variables
my $target="/vmimages";
my $cfg=$ARGV[0];
print "$cfg\n";

# Set up a connection to a virtual machine
my $params = VMware::VmPerl::ConnectParams::new();
my $vm = VMware::VmPerl::VM::new();
$vm->connect($params, $cfg);

# No smooth way to return the number of scsi controllers
# We will cycle through all possibilities checking if it is present
for (my $scsiController=0; $scsiController<=3; $scsiController++) {
  my $presentScsiController = $vm->get_config("scsi$scsiController.present");

  # If it is there, we will continue processing
  if ($presentScsiController eq "true") {

    # Again, cycle through all possible scsi IDs
    for (my $scsiID=0; $scsiID<=15; $scsiID++) {
      my $presentScsiID = $vm-
>get_config("scsi$scsiController:$scsiID.present");
      if ($presentScsiID eq "true") {
        # Get the path to the vmdk
        my $vmdk = $vm->get_config("scsi$scsiController:$scsiID.name");

        # $vmdk format is now vmfsvol:vmdk
        # Let's break this up into 2 variables
        my ($vmfsvol,$vmdkname) = split (':',$vmdk);
        my $vmdkPath = "/vmfs/$vmfsvol/$vmdkname";

        # Add the first redo
        $vm->add_redo("scsi$scsiController:$scsiID");

        # Do a backup
        `/usr/sbin/vmkfstools -e /$target/$vmdkname $vmdkPath`;

        # Add a second redo
        $vm->add_redo("scsi$scsiController:$scsiID");

        # Wait a second for the redo to be created
        sleep(1);

        # First commit with same options as vmware-cmd
        $vm->commit("scsi$scsiController:$scsiID", 1, 0, 1);

        # Commit final redo
        $vm->commit("scsi$scsiController:$scsiID", 0, 0, 1);
      } # End If SCSI ID is present
    } # End for SCSI ID Cycle
  } # End If SCSI Controller is present
} # End for SCSI Controller Cycle

# Cleanup
$vm->disconnect();
```

Master Craftsman…

Answer VM Questions from a Script

After some events occur, VMware ESX Server won't continue until you answer a question. ESX requires your answer to the question before the process can resume. For example, if you accidentally try to add a third REDO log, a question is generated. This question has only one answer, OK. Once you answer the question, the process resumes.

The problem here is that your scripts need to be able to answer these questions as they come up. Otherwise, your script will pause indefinitely. The following code can be used in your scripts to answer single option questions. You could also easily modify the script to answer more difficult questions.

```
#!/usr/bin/perl -w
#
# This script is an example only
# Usage: detectQuestion.pl <vmxConfigFile>
#

use VMware::VmPerl;
use VMware::VmPerl::ConnectParams;
use VMware::VmPerl::VM;
use VMware::VmPerl::Question;
use strict;

# User variables
my ($cfg) = @ARGV;

# Connect to the virtual machine
my $params = VMware::VmPerl::ConnectParams::new();
my $vm = VMware::VmPerl::VM::new();
$vm->connect($params, $cfg);
```

Continued

```
# Check for a question. Will return undef if
# no questions.
my $question = $vm->get_pending_question();

# If $question is defined, there is an outstanding question
if (defined $question) {
  my $text = $question->get_text();
  my @choices = $question->get_choices();
  if ($#choices == 0) {
    # There is only one choice, easy to answer it
    $vm->answer_question($question,0);
    print "Question answered: $text\n";
  } else {
    print "More than one choice.\n";
    print "Choices: @choices\n";
  } # End if only one choice
} else {
  print "No Questions\n";
} # Endif

# Cleanup
$vm->disconnect();
```

Choosing the Target for VM Backups

At some point, when writing your backup script, you'll need to decide where your backups will go. You'll also need to decide how to get them there. In most cases, you'll choose some type of mass storage device, like a file server, a NAS device, or a SAN array as the target to store your backups. How you get those backups to the chosen target can vary greatly. Considering that the VMware ESX service console is running a modified version of Red Hat Linux, there are a plethora of options available as to where you may target your VM backups. Some protocols copy faster data than others. Some are simpler to use in scripting. Some integrate better with your chosen storage target. We'll cover some of the available options and provide some recommendations on when to use each.

In this section, we'll address the transports available for backups and discuss where the data will be stored. We won't address specific storage types, such as specific SAN arrays or NAS providers. We'll talk about these in more general terms. Since we're more concerned here about the transport protocols used to get your backups from ESX to your target storage.

Some of the more common and popular ways of moving backup data are NFS, CIFS, FTP, and copies to VMFS. We'll define each of them here, and then discuss the benefits of each in turn. Each of the following methods is listed in our order of preference. Consider these options when deciding what will be best for your scripting needs.

NFS

Network File System (NFS) is a common file sharing protocol used mainly in UNIX and Linux environments. It could be considered the standard file sharing protocol for *NIX systems. NFS works by exporting a file system from one machine and making it available to the network. Other systems use an NFS client to mount the exported file system at a mount point on their local file system. The exported file system is then accessible from the mount point as if it were part of the local file system.

NFS is a fairly simple way to share, or export, a file system from one machine and access it from another. Generally, we like NFS for facilitating all file sharing from the ESX service console, especially for VM backups. NFS is fast, native to the service console, and simple to use in scripts.

Attributes of NFS for VM Backups

In this section, we'll discuss the pros and cons of using NFS for VM backups.

Pros

The pros of using NFS for backups include:

- NFS doesn't require authentication, so you don't have to code in usernames and passwords.
- NFS is very fast over Gigabit Ethernet networks.

- NFS is usually an available option on a NAS device.
- NFS exports mounts directly into the file system on mount points. Very easy to copy data back and forth using native copy commands like *cp* and *vmkfstools*.

Cons

The cons of using NFS for backups include:

- NFS does not have any native support in Windows. Requires Services for UNIX. Not recommended.
- NFS is not as secure as other options, due to lack of authentication and data encryption.

CIFS

Common Internet File System (CIFS) is a standard implementation of the SMB (Server Message Block) protocol largely developed by Microsoft. It is essentially the base protocol that Windows uses to copy data between systems. Windows file servers and many NAS devices use CIFS as the protocol to authenticate and transfer data.

Linux uses an open-source implementation called SAMBA to interact with CIFS servers. In order to copy data to a Windows share, you'll need to install the SAMBA client on your ESX service console. CIFS is second on our list of transports because it is a more complicated implementation than NFS. It needs authentication and sometimes requires a two-step process to copy a VM.

Attributes of CIFS for VM Backups

Now we'll discuss the pros and cons of using CIFS for VM backups.

Pros

The pros of using CIFS for VM backups include:

- CIFS is easy to integrate into a Windows sharing environment.

- CIFS is commonly the preferred, or only, protocol supported on an NAS device.
- CIFS can be mounted, via SAMBA, to a local mount point.

Cons

The cons of using CIFS for VM backups include:

- CIFS is more difficult to configure in the service console.
- CIFS requires SAMBA installation and configuration.
- SAMBA has been less stable than NFS in our experience.
- CIFS is not as fast as NFS over GigE.
- CIFS is a very chatty protocol, which decreases performance over latent connections.

FTP

File Transport Protocol (FTP) is a very common protocol for copying data over a network. It is a standards-based protocol that is supported on nearly every modern computing platform. FTP is useful for copying backups to a file server. It is natively supported on the ESX service console. It is pretty easily scripted and has a substantial amount of reference resources available on the Internet.

Attributes of FTP for VM Backups

In this section we'll weigh the pros and cons of using FTP for VM backups.

Pros

The pros of using FTP for VM backups include:

- FTP servers are common and supported natively on most servers.
- FTP copies data quickly over a noncongested network.

Cons

The cons of using FTP for VM backups include:

- FTP often requires a server platform as a target since many NAS devices do not support it natively.
- FTP takes all available bandwidth it can for copying. It may step on other network traffic.
- FTP does not have the capability to mount on the local file system.
- FTP generally requires authentication, but without certificates it sends usernames and passwords in clear text.
- FTP passwords must be coded into your scripts. This is insecure and will break the script if accounts and passwords change.

VMFS

VMware File System (VMFS) is the file system used for virtual machine disk file storage in VMware ESX server. It is a distributed file system, which means it can be accessed by multiple ESX servers at the same time and not corrupt any data. VMFS locks individual files rather than entire volumes. This means many ESX servers can access files from the same VMFS volumes without any trouble.

The nature of VMFS makes it an attractive target for VM backups. A VMFS volume can be designated as a backup target and shared across all of your ESX servers. This way, backups can be directed straight from the source VMFS to the target backup VMFS volume. Since the .vmdk file format doesn't need to change when moving from VMFS to VMFS, you can copy the .vmdk files directly. This simplifies the scripting required to move data around.

Don't be too easily lulled into using VMFS as your backup target. Generally, we prefer to use non-VMFS targets for VM backups. VMFS isn't a good file system for sharing files (for example, there is no support for directories), it only supports a maximum of 192 files, and it has SCSI reservation issues when copying large amounts of data. You're better off using one of the methods discussed earlier for a permanent solution for backup targets.

Attributes of Copies to VMFS for VM Backups

Now we'll discuss the pros and cons of using copies to VMFS for VM backups.

Pros

The pros of using copies to VMFS for VM backups include

- Sharing VMFS volumes between ESX servers is easy.
- Scripting syntax is fairly simple and doesn't require additional mounts or connection syntax as FTP or CIFS might.
- VMFS is often stored on SAN LUNs, which can help facilitate a larger backup strategy. (For example, back up to VMFS, then take a snapshot and/or replicate the SAN LUN.)

Cons

The cons of using copies to VMFS for VM backups include:

- VMFS doesn't scale well in large environments. It's not practical to attach a VMFS to more than 16 ESX servers. You can run into contention issues and SCSI reservation problems when performing a large number of simultaneous backups to a single VMFS.
- VMFS was designed to host large VM disk files, not be a file server.
- VMFS has no support for a directory structure. Organizing backup files in a sensible way is difficult.
- There are limits to the number of files that can be stored in a VMFS volume. Each VMFS extent can hold 192 files. Most often you'll only have one extent, and are therefore limited to 192 total files in the VMFS. This is a big inhibitor for doing a large-scale backup solution with a VMFS target.

TIP

Never use the *cp* command when copying .vmdk files. Always use *vmkfstools*. An undocumented, but useful switch for *vmkfstools* copies a .vmdk in one command and is very fast. This method exports and imports the VMFS in one step. The syntax is as follows:

```
    vmkfstools -e /vmfs/vmfsname/target.vmdk -d vmfs /vmfs/vmfs-
name/source.vmdk
```

If you're going to use VMFS for backup storage, dedicate an LUN to it. Don't combine active VMs on the same VMFS that you're using for backups. You could run into major performance problems due to the large amount of SCSI reservations that can occur on the VMFS volume during copies. These locks, if frequent enough, will be noticed by your VMs and can cause undesirable results.

Existing VM Backup Tools

Now that you know the basics of a hot backup, we hope that you do not set out to write your own backup application without checking out some existing applications. There are many options, both free and commercial, that cover the full spectrum of price and support. Before you sit down and reinvent the wheel, check out some of the wheels that have been created before. We'll go into detail about some affordable (free) options and provide guidance on where to look for commercial solutions.

vmsnap.pl, *vmsnap_all*, and *vmres.pl*

VMware ESX 2.x ships with three scripts that work together to create a backup system. *vmsnap.pl* will back up a single virtual machine, while *vmsnap_all.pl* will call *vmsnap.pl* for all virtual machines on the host. *vmres.pl* is the restore portion. The three tools are fully supported by VMware with no additional charges other than the original ESX licensing.

vmsnap.pl has basically the same logic as the simple sample we went through in the beginning of the chapter. It will manage the REDO log process for you and copies VMDK files using a *vmkfstools* export. It will also back up your VMX, nvram, and virtual machine log files. The script also handles logging, local or remote. The output destination options include local filesystem and ssh. VMware refers to the ssh destination as an archive server in the documentation.

vmsnap_all.pl is essentially identical to *vmsnap.pl* in functionality, except that it will back up all VMs on an ESX server.

This application has some downsides, however. It does not natively support keeping multiple versions, and will even overwrite files by default. If you have a requirement to keep more than one version of a backup, you need to apply additional scripting and sweep up the output files using a different backup system on a regular basis. Also, *vmsnap.pl* is missing file compression capabilities.

TIP

The three native scripts, *vmsnap.pl*, *vmsnap_all*, and *vmres.pl*, are a good place to start for ideas to apply towards your own scripts. They expose many ESX functions that are useful for other purposes.

vmbk.pl

We have to make mention of *vmbk.pl* in this text. Considering that this Perl script is made freely available by Massimiliano Daneri, and it has a broad range of fantastic features and functionality, we feel obliged to promote his efforts and provide a link to his Web site. You can find the scripts and information at www.vmts.net/.

Basically, *vmbk.pl* employs many of the functions we've described in this chapter. It uses Perl as the scripting engine (our personal favorite). Its main function is to perform hot VM backups. It adds .REDO logs to running VMs and exports the .vmdk files using *vmkfstools*. It grabs the VM config files, .VMX and CMOS files, then facilitates the transfer of the backup files via NFS, CIFS, FTP, or through Veritas NetBackup to a backup target—for example, NAS, SAN, or tape. At that point, it commits the .REDO log files back to the running VM.

vmbk.pl is a good option to consider as a script, given that you can immediately start using it for backups. It also provides a great place to start if you're looking to incorporate some of these features we've discussed into scripts of your own.

Commercial Options

Many commercial options are available that perform VM backups in various ways. Thus, the following reasons should be considered when deciding whether to use a commercial product versus writing your own scripts:

- You don't have to write your own application. This can save a tremendous amount of time and/or money.
- They carry support contracts. If things break, you have a professional to call. It also helps you keep your job if you have a real disaster.
- The vendors are generally continuing to add features and functions that will make your life easier.
- Scripted solutions generally require significant knowledge of the Linux shell. If your staff is not comfortable here, a Windows GUI option, provided commercially, will make life easier for your admins.

If you're interested in looking at a few options, consider some that we have worked with and feel have good approaches and appropriate pricing models:

- Vizioncore esxRanger
  www.vizioncore.com/esxrangerPro.html
- esXpress
  www.esxpress.com/

Swiss Army Knife…

Using Backup Technologies for Other Purposes

In the new world of virtualization, users are continuing to come up with unique uses for the technology. One idea discovered in the field is using backup technologies as version control tools for the support and development of software products. This is a rather simple but useful technique for the software development community.

The idea is that as your software goes through its various versions, an archive backup is written to a file system and stored with the version number referenced in the description. This can be simply one VM, or a complicated multitier environment. When a customer calls looking for help with an old version of your software, you can restore the complete environment to an alternate virtual infrastructure. Use this duplicate version to facilitate re-creating and solving the problem. In the physical world, this would be a large and possibly expensive task due to the amount of hardware required. In the virtual world, you can do this entirely from your desk or couch with a minimal amount of hardware.

VMX File Backups

Thus far, our focus for backup has been on VMDK files. While VMDK files are critical because they contain your actual data, VMX files are also important. They tend to sit on the local disk of an ESX host, and a copy of the configuration is not located in the VirtualCenter database. Oftentimes, the local copy of the VMX file is your only record of the configuration of each virtual machine. It would be a disaster to lose the local disk and need to figure out each virtual machine's configuration when the heat is on.

TIP

Maintain an inventory of your virtual machines outside of ESX or VirtualCenter. We recommend creating a spreadsheet that has the configuration details for all of your virtual machines. Include every option listed in the VM configuration. With the VMX files stored on the local file system of ESX server, this document will prove invaluable in a disaster.

Things you should document:

- The virtual machine name
- Which ESX host it resides on
- The path to the config files
- The number of CPUs
- The amount of RAM
- Each virtual disk, its SCSI ID, and its path to the VMDK file
- The virtual disk mode settings—for example, Persistent versus Undoable, and so on
- Any other peripherals and their config information
- The startup order in relation to other VMs on the ESX host
- The performance policy settings—for instance, the CPU and RAM shares and Min/Max settings

Many of the products listed in the existing VM backup tools section of this chapter cover VMX backups, but you may be looking outside of the existing tools for your VMX backups. An option would be to install a local backup agent in the service console and configure it to back up the /home directory on a regular basis. If you don't want to shell out for the agent costs just to back up a couple MB of data, then you can easily put together a script to copy the VMX files once a day.

The script shown in Code Listing 7.2 is an example of how to copy VMX files using Perl. This is intended to be a starting point. By default, it will copy to a locally mounted directory on the ESX host. Also included is an example line to copy to another host via SSH.

The script does not do many things that you may wish to cover. You could add /etc/vmware/ to store your ESX configuration files. You could add /var/log to cover the log files in case of system crash or security incident.

Also, you may want some versioning on the files to store older VMX files to find out what has changed.

Code Listing 7.2 Perl Script for Copying VMX Files

```
#!/usr/bin/perl -w
#
# This script is an example only
# Usage: vmxBackup.pl

use VMware::VmPerl;
use VMware::VmPerl::Server;
use VMware::VmPerl::ConnectParams;
use VMware::VmPerl::VM;
use strict;

# User variables
my $target="/vmimages/vmxBackup";

# Setup a connection to the local ESX host
my $params = VMware::VmPerl::ConnectParams::new();
my $host = VMware::VmPerl::Server::new();
$host->connect($params);

# List of registered virtual machines
my @vmlist = $host->registered_vm_names();

foreach my $vm (@vmlist){
  # Get the displayName of the vm
  # We will use the displayName to title the backup output file
  my $vmo = VMware::VmPerl::VM::new();
  $vmo->connect($params, $vm);
  my $displayName = $vmo->get_config("displayName");

  # Finally, you may have some problems with special characters
  # I recommend removing them to prevent hassles.
  # This line will remove ( and ) and spaces.
  $displayName =~ s/[\() ]//g;
  # This will tell us what directory the vmx is in.
  my @path = split("/",$vm);
  my $dir;
  my $cnt=0;
  until ($cnt == $#path) { $dir = $dir . "$path[$cnt]/"; $cnt++; }

  # Here is the actual backup command
  my $cmd = `tar cvzpf \"$target/$displayName.tgz\" \"$dir\"`;

  # To go remote via ssh, use this command instead
  # Remember to set up ssh key auth first
  #my $cmd = `tar cvzpf - \"$dir\" | ssh user\@host \"dd
of=\"$target/$displayName.tgz\"\"`;

  # Cleanup
  $vmo->disconnect();
} # End foreach vm

# Cleanup
$host->disconnect();
```

This script will copy all registered VMX files to the location specified. It will cover all files in the directory with the VMX, such as nvram and log files. Be aware, in its current form, the files will be overwritten each time the script is run. The output is tar gzip format with the filename of the configured display name .tgz.

Swiss Army Knife…

Scripting the Synchronization of VMX Files to Another ESX Host

You may have a need to store VMX files on another ESX host, preregistered. This may be due to a couple of reasons. First, you are replicating the SAN-based VMFS volumes and have warm servers waiting to be used at the DR site. Second, you have a need to recover a failed ESX host very fast—fast enough to warrant the additional complication of managing a

Continued

sync process. The preceding sample VMX backup script could be slightly modified to cover this situation. Only a couple of simple changes need be made.

1. The tar statement must use SSH, and needs to explode the tarball on the remote side. An example is shown next. Note the capital P options on both sides. This will preserve file paths.
   ```
   my $cmd = `tar cvzPpf - \"$dir\" | ssh user\@host \"tar
   zxPf -"\"`;
   ```
2. Following the tar command, the VMX needs to be registered. We recommend using *vmware-cmd* to accomplish this.

Incorporating Hot VM Backups into Your Recovery Plan

Up to this point, we have discussed the essential knowledge needed to perform backups and restores with scripts. We also covered a few very useful scripts packaged into applications, some free and some commercial. Where do you go from here? Well, you've now got to assimilate all this technical information and merge it into your backup/restore/disaster recovery strategy. This section is where the rubber meets the road. We're going to dive into why and how you would use hot VM backups as part of your total recovery strategy.

Before we dig in, let's pause and face reality for a moment. Have you ever had an end user give you a high-five after a standard nightly backup job? I didn't think so. No one really cares about backups. No one was ever considered a hero after their backups successfully completed. What does matter, what people love, and what will get you much praise and many free lunches are successful restores. When you restore the sales forecast spreadsheet an end user lost after a week of work, you become the instant hero. Backups are important, restores are critical. The time it takes to restore data matters. The data integrity of restores matters. The amount of data your business can afford to lose and keep on running matters.

With the perspective that restores are what matters most, let's discuss how to incorporate hot VM backups (and restores) into your recovery strategy.

When talking about a backup strategy and disaster recovery, it's critically important to start with the end result in mind. You should know now what you need to have happen after a disaster occurs. Without getting into a full out discussion of DR planning topics, let's cover a few basic DR planning topics.

Some key information you need to know about every application or set of data in your environment is its RTO and RPO. Let's define these acronyms.

- **RTO (Recovery Time Objective)** This is the amount of time that may elapse after a disaster until the application or data needs to be operational. In other words, the RTO is your deadline for recovery.
- **RPO (Recovery Point Objective)** This is the largest amount of time that may exist between the present and the last recoverable point in time for the application or data. In other words, the RPO is how much data, measured in time, you can afford to lose.

Before you can determine your backup strategy, you should go through and inventory your systems, group them into applications and data sets, and then determine the RTO and RPO for each one. Done correctly, this process isn't really completed by the IT staff. It's a process that is highly dependent on the opinion of those that run your business. If the business says that the CRM database has an RTO of 12 hours and an RPO of five minutes, then your job is now defined. At this point, you can apply strategy and tools to accomplish those objectives. Without those guidelines, it's impossible to create a recovery strategy that is valid to your business.

Often, as you take the guidelines from your business and translate that into tools, human resources, and ultimately expenses, you may get a different answer regarding what the RTOs and RPOs are. Money talks, loud. A few rules of thumb when it comes to determining how redundant to make your systems based on recovery requirements:

- The lower the recovery requirements (RTO and RPO times), the more expensive and difficult the solution to achieve them will be. Zero downtime and zero data loss, for example, generally require completely redundant systems with expensive replication software and high availability clustering. Whereas a slightly less resilient system can be implemented that is good enough with much less investment and generally highly satisfactory results.

- The more complicated your redundancy systems are, the more prone you are to failures. We've often seen "highly available" systems end up with more downtime than less redundant systems. In most cases, it happens because the system became so complex in an effort to be redundant that the human factor mismanaged it.
- The K.I.S.S. factor most often works better than over engineering. K.I.S.S. = Keep It Simple Stupid. A simple system, compared to a complex system with many moving parts, generally has less chance of failing. Simply put, fewer components equates to fewer failures.

Once you and the business come to agreement on what needs to be protected, you'll get the opportunity to dig through the myriad tools and techniques to determine the best way to get it done. You'll then be armed with the data you need to determine what tools, scripts, agents, applications, libraries, arrays, replication, and so on to use for backup and recovery.

Now, the scope of this work is not to teach you how to do disaster recovery planning. However, we thought it very important to frame the concepts of hot VM backups within the discussion about disaster recovery planning. There seem to be misconceptions in the community about what hot VM backups can do. Often, they are given more credit than they deserve. Rarely have we found an enterprise implementation of VMware that can be fully protected by a standalone hot VM backup tool. Now that you've been through this chapter and understand what hot VM backups can do, you can start to figure out where it fits in your plan.

Let's simply state the functionality of hot VM backups by listing what they can and cannot do in the two lists shown in Table 7.1. In this table, the plus column stands for functions that hot backups can do; the minus column stands for functions they cannot do.

Table 7.1 Capabilities of Hot Virtual Machine Backups

Plus	Minus
Perform zero downtime backups of VMs without a performance hit	Perform file level backups and restore
Capture the entire state of the VM, including boot, sector, OS, and applications.	Create detailed catalogs of backed-up files.
Back up virtual machines without guest OS agents.	Close files and databases before taking a backup. State of backed-up VM is crash consistent.
Be written to disk, tape, or network shares.	

Crash Consistent State

Let's define crash consistent and explain why it matters to you. Have you ever pulled the power from a server while the OS is running? How about hold the power button down for 15 seconds or so? Or, have you ever pulled the fiber cable from a server that boots from SAN? The state that your server is in after it reboots is a crash consistent state. Crash consistent state usually follows an abrupt and immediate power off or freeze of the operating system. The OS and applications were not made aware of the shutdown, so consequently they didn't do any of the things they normally do before powering off. Some of these activities are quite important, such as committing transactions to a database and closing files, writing uncommitted data from memory to files, committing outstanding I/Os to disk, and other items of this sort. When the server comes back up, it has to deal with the sometimes unpleasant and often very messy situation of cleaning up after the crash.

The good news is that most operating systems and applications are aware that crashes occur once in a while. They have mechanisms built into them to recover from this type of disaster. Databases write uncommitted data to transaction logs before it is written to the database. File systems have journaling features that log any changes to a temporary journal before committing them to the main file system. These transaction logs and journals are used to replay data that wasn't committed before the crash back into the main data set.

Keep in mind, however, that crash consistent means that there is a chance that you may have corrupted data, broken file systems, uncommitted transactions, or untold other failures after a crash. We need to throw this warning out there even though in the vast majority of instances with standard applications there are no problems coming out of crash consistent states.

When you perform a hot backup of a virtual machine, you are essentially freezing the disk and taking a snapshot of it. At the exact moment you add a .REDO log to a .vmdk file, the state of the data is frozen, whether or not files are open or closed and databases are running or quiesced. The good news is that VMware takes care to commit any transactions that are in flight to the .vmdk file when the .REDO log is added. In 99+ percent of the cases, you'll have no problem recovering the data in the .vmdk file.

WARNING

We said 99+ percent of the time you'll be able to recover the data in the .vmdk file. That doesn't mean it will meet your usability expectations. If you have an application that doesn't like to be frozen in the middle of a transaction, then you may have a situation where your data is recovered, but useless. The disclaimer is this: test test test this functionality out before you rely on it as part of your disaster recovery plan. That should go without saying, but we've been consulting long enough to know that there isn't much that we leave unsaid and unchecked.

Replication

If you can't afford to lose any data, then hot VM backups are not for you. Neither are file-level backups. You just graduated to an advanced level of backups, called replication. Data replication can be performed at the file system level or at the storage array level. High-end solutions require a lot of bandwidth and can provide synchronous replications of all data. Synchronous replication ensures no data is lost. For situations with less bandwidth, asynchronous solutions queue up replications and trickle them over connections at set times. These can be five minutes behind or 24 hours behind. It's adjustable based on your configuration.

Real-time replication is currently the best solution for zero data loss environments. It's the only way to guarantee that you don't lose a single transaction during a disaster. Replication is reliable and it works, but it comes with a price. Replication solutions are generally many times more expensive than traditional backup methods. However, if you need it, you need it, and you'll be willing to pay for it. If not, then it's time to compromise.

Hot VM Backups as Part of the Recovery Plan

Now, you've taken the earlier advice and considered where this type of backup/restore procedure will fit into your disaster recovery plan. You've considered which of your applications recover well from crash consistent states and which absolutely do not. You've decided that you'll enable journaling on your ext3 and reiser file systems and you'll use transaction logs with your Exchange and SQL servers. Good. Now let's discuss a common approach for using hot VM backups in your plan.

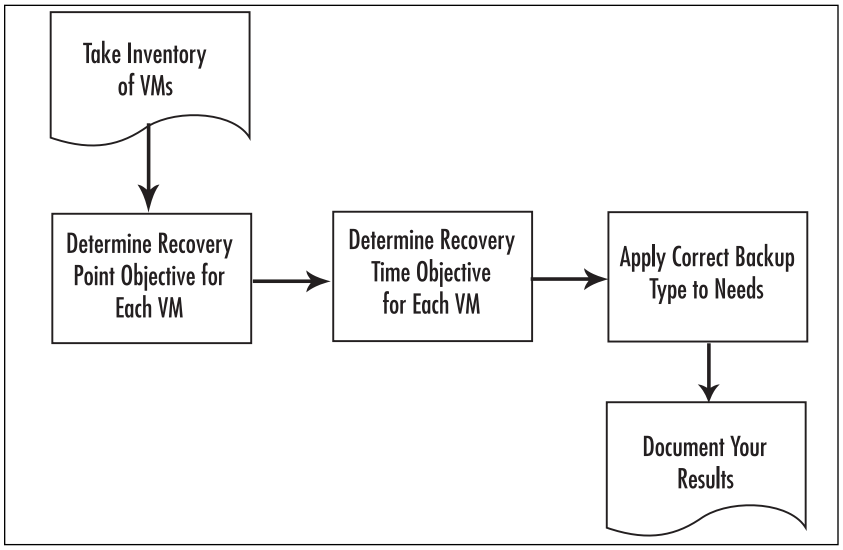
To begin with, it's important to understand that one of the major limitations of a hot VM backup is it has absolutely no knowledge of the files inside the .vmdk file. If you need to recover that sales forecast spreadsheet that is backed up inside a .vmdk file, you're going to have to find it yourself. There is no catalog of files contained inside the guest OS file system that you can refer to. To achieve file-level restores, you'll need to use a file-level backup tool in addition to your hot VM backup tool.

Let's walk through the steps to determine the correct recovery strategy for your applications and data sets. The five-step process shown in Figure 7.8 will help you establish the correct policy for each application.

Step One: Take an Inventory of Your Virtual Machines

You can't plan for recovery unless you know what you have. A wise Electronic Janitor once told me, the majority of IT is inventory. To begin, create a simple spreadsheet that contains a detailed inventory of your virtual machines and the applications running within them. You'll need to record at least:

Figure 7.8 Process to Determine Backup Strategy


Take Inventory of VMs
Determine Recovery Point Objective for Each VM
Determine Recovery Time Objective for Each VM
Apply Correct Backup Type to Needs
Document Your Results

- The operating system
- Which applications run on each OS
- The location where data is stored

Especially note if some data for your VM is not stored in VMFS volumes. This data will need to be addressed individually.

Now that you've begun this document, you'll be able to use it as a foundation for building out the rest of the recovery plan. Expand the spreadsheet during the next few steps to include RTO and RPO requirements for each virtual machine.

Step Two: Determine the Recovery Point Objective for Each VM

The recovery point will tell you how often you need to perform a backup of your VM. Answer the following question for each VM:

How much data can I afford to lose?

Once you know how much data you can afford to lose, you can decide the frequency of your backup jobs. If you can afford to lose seven days work, then only back up once a week. If you can afford to lose up to 24 hours of work, then a daily backup is perfect.

Step Three: Determine the Recovery Time Objective for Each VM

Earlier, we discussed planning for recovery first. At this step, think about the type of recovery that will be required for this application or data set.

Answer the following question for each VM:

How fast does it have to be recovered after a disaster? (RTO)

The time required to recover a VM is often overlooked when applying a blanket backup strategy to systems. If you only have a tape backup of an application, the recovery time will include the process of installing a new operating system, setting up a backup agent, and restoring the application data from tape. This process at a minimum will be several hours. If your RTO is less than those several hours, rethink your tool selection.

Hot VM backups take about as much time to restore as they do to back up. If you're using compression on the backups, then the recovery time will go faster. The compression calculations are not as intense on a recovery as they are on a backup.

Step Four: Apply the Right Backup Job to the Need

Once you have the business requirement for how fast you need to recover, and how much data you can afford to lose, you can use this information to decide on the right backup tool. The tool must back up frequently enough to meet the RPO and be able to provide recovery quick enough to meet the RTO.

At this point, you have gathered enough information to decide which type of backup tool will meet your recovery requirements.

Table 7.2 shows a general comparison between the different backup/recovery tools we've discussed in this chapter. You can use this as a starting point to help decide which tool fits your recovery requirements best, and, ultimately, to determine whether hot VM backups are for you.

Table 7.2 A Comparison of Backup Tools

Backup Type	Min RTO*	Min RPO**	Cost	Complexity
VM hot backup	< one hour	24 hours	Low	Low
Tape backup agent	1–24 hours	24 hours	Medium	Medium
Storage replication	< five minutes	Real time	High	High

* Minimum Recovery Time Objective is an estimate based on experience of the time required to recover typical data using the specified tool. Your situation may vary greatly depending largely on the amount of data to be backed up and recovered.

** Minimum RPO depends on the frequency of backups. For example, daily backups provide a < 24-hour RPO, while weekly backups provide a < seven-day RPO.

Decide here whether a crash consistent copy of the VM will meet your requirements, or whether you need file-level protection and restore capability as well. Your application may require a special agent to perform a proper backup and restore—examples are open file agents, exchange agents, and SQL database agents.

If crash consistent is good enough and the recovery time is acceptable, then a hot VM backup is perfect for you. If you need file-level recovery, then you need file-level backups as well. If your requirements say that you need zero downtime, and your budget supports the need, explore highly available solutions with storage replication.

At this point, you need to prioritize which VMs (applications) are more important than others. The importance of the VM determines its priority in a recovery. You probably can't do all your restores at the same time, and you're more likely to perform recoveries in a serial manner. So, decide which VMs are the most important and categorize them as your top-tier systems. These will be the systems that get restored first after a disaster. Other VMs will be categorized as a lower tier, and will therefore be recovered after the top-tier VMs. Make sure to set expectations with your end users that top-tier systems will be restored first. To change the priority will either cost more money or require a reprioritization of the order in which systems will be recovered.

Step Five: Document Your Results

It is critical to document your plan. Although there seems to be a general aversion to documentation in the IT community, it is nevertheless of utmost importance. If your plan is not documented, you will have a difficult time explaining it to others. If the plan is not documented, you may find yourself in trouble during a disaster. For that matter, make sure your documented plan is stored outside of the system it is protecting. If the plan is stored on the file server, and the file server goes down, you won't have much luck reviewing the plan. Keep a digital and printed copy of the plan at all times. Copies of the plan should be kept in multiple locations in the event a location is inaccessible as part of the disaster.

Hybrid Backup Strategy

For systems that have file-level restore needs, the hybrid approach generally is best. The hybrid backup strategy combines the best attributes of the hot VM backup with the best attributes of the file-level backup. The advantage of the hot VM backup is that the restore is fairly quick and requires little user intervention. Once the server is restored, the last file-level backup can be applied to bring the data back to as current as possible. This approach eliminates the need to reinstall an operating system and tape backup agent. Helping you avoid having to search for OS CD-ROMs, drivers, and agent install disks during a disaster. These are small issues that can waste precious minutes and hours during a disaster.

Let's review a common hybrid backup strategy.

Backup method:

- Take a hot VM backup regularly, such as once per week.

  Take a file-level backup using an agent in the guest OS every day.

Restore method:

To recover the entire server:

1. Perform a restore of the entire VM from the last hot VM backup.
2. Apply the latest file-level restore to that VM.

The advantage here is that you can recover the entire VM very quickly using the hot VM backup, and then bring its files up-to-date with the last file-level restore. This will bring your server up to a state where the OS is fully configured, the backup agent is already loaded and working, and data is current to the last hot backup. This entire effort is achieved with a minimal amount of human intervention. All that is left at this point is restoring data from the last incremental backup. You didn't have to build an OS from scratch, load the backup agent, and then perform a full system restore. You saved yourself hours of work, eliminated countless opportunities for human error, and in the end recovered your data much faster.

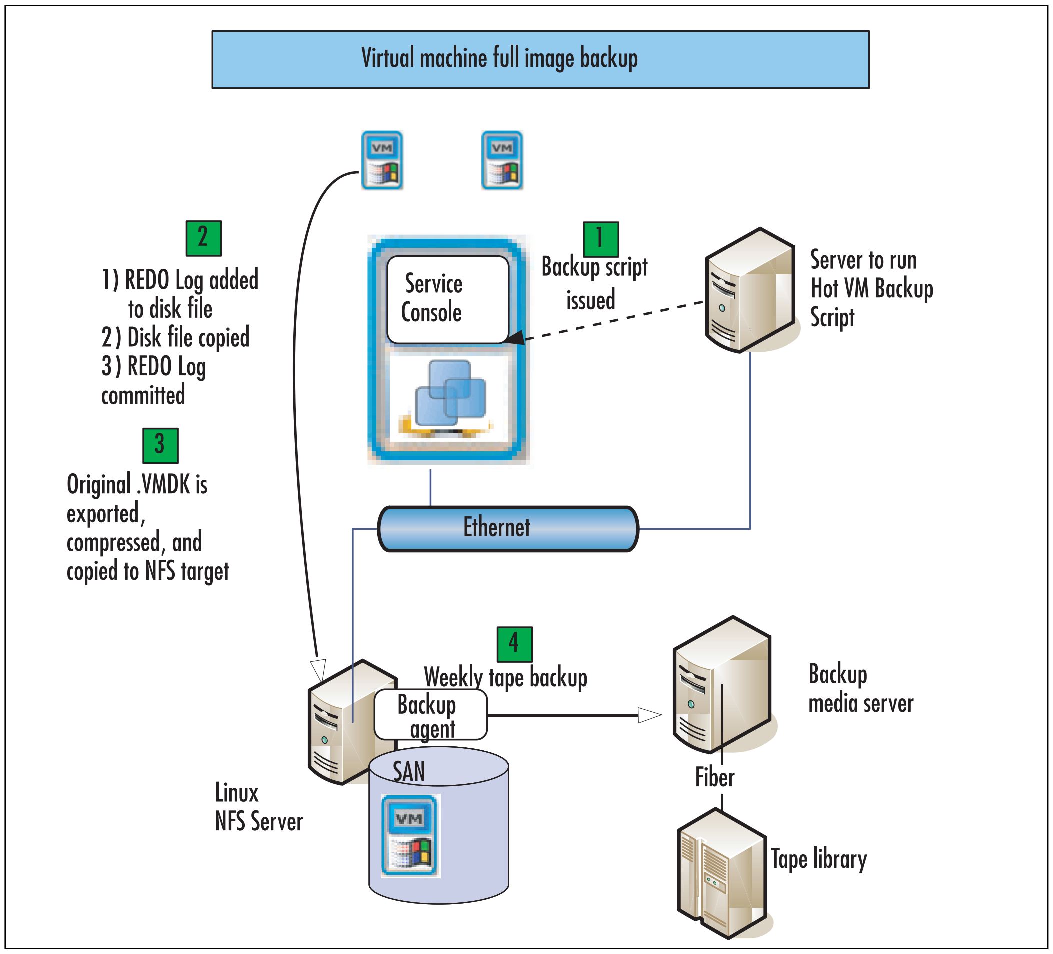
Table 7.3 shows an example of what a hybrid backup schedule may look like. It combines the file-level backup agent with hot VM backups, called Full VM Server images. To sum up the following schedule, a full image of the VM is captured once a week with the hot VM backup script—in this case, esxRanger. Then a daily file-level backup is taken using the CommVault agent. (CommVault is a backup software ISV.) Once a week, the repository of VM images is also copied to tape. The retention times listed here are subject to change based on your specific requirements. The times shown in Table 7.3 are merely examples to get you started on your plan.

Table 7.3 Example Backup Schedule

Backup Type	Tool	Media	Schedule	Retention
File-level backup	Example, CommVault	Tape (CommVault Media Server)	Daily	One month
Full VM server image	Example, Hot VM backup script	Disk (Linux NFS export on SAN LUN)	Weekly	One week
Tape backup of full server image files	Example, CommVault	Tape (CommVault Media Server)	Weekly	Two months

This backup schedule is also represented in Figure 7.9.

Figure 7.9 Virtual Machine Backup Process



Summary

If you've mastered the topics in this section, you are well on your way to a complete backup solution for your VMware virtual infrastructure. You should be able to confidently script hot backups of your virtual machines and their related config files. You are now armed with information about alternative commercial solutions, and have the knowledge to apply what you've learned to your overall recovery strategy.