Realtime publishers

The Definitive Guide™ To

Windows Application and Server Backup 2.0

sponsored by

AppAssure
HOME OF BACKUP 2.0

Don Jones

Copyright Statement

Chapter 7: Virtualization Server Backups

Virtualization is the hot new code word for today's businesses, and it's changing everything we thought we knew about backup and recovery. I actually consider that to be a Really Good Thing, because it means—at least with regard to virtualization—we don't have to un-learn as many Backup 1.0 habits in order to see how a Backup 2.0 technique might be more effective.

Native Solutions

Oddly, it's almost like virtualization vendors *know* that backups are complicated and that native solutions are usually deficient in key capabilities, because most virtualization vendors—we're talking Citrix, VMware, and Microsoft, here—don't really provide any native backup capabilities at all. Sure, most of them have backup solutions they'll *sell* you, but most of those are still good old Backup 1.0-style, "get it done during the evening backup window" solutions.

But let's take a step back and look at why virtualization backups have the potential to be more challenging than a physical server backup.

Problems and Challenges

As illustrated in Figure 7.1, a virtualized server runs on a virtualization host—like VMware vSphere or Windows Hyper-V. The host controls the hardware of the physical machine, while the virtualized server has its own, virtualized hardware environment. The virtual server's hard disks are generally just files sitting on the physical host. I realize this is probably nothing new to you; I'm just setting some context.

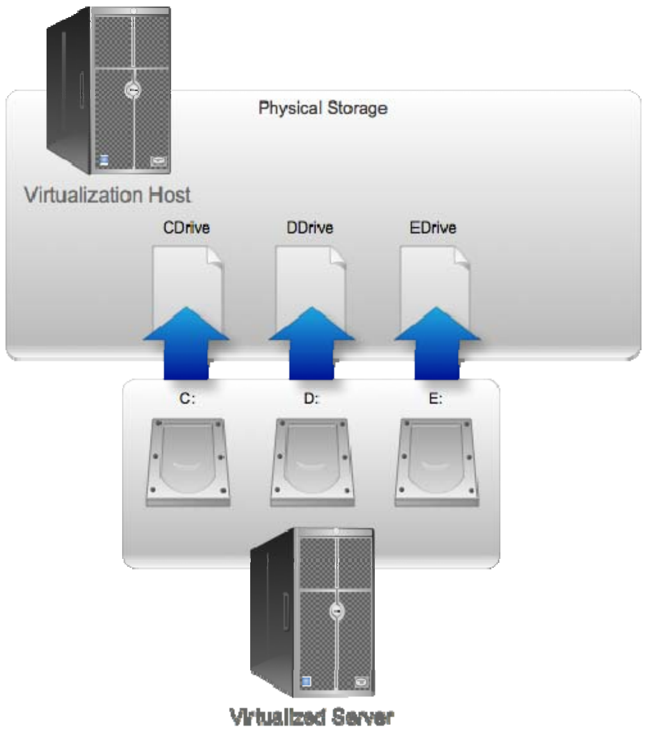

Figure 7.1: Virtual server running on a virtual host.

This whole architecture means that, unlike with a normal physical server, we've got two places we could run backups. The first place is on the virtualization host itself, where we'd simply back up the files that represent drives for the virtual servers. In other words, if we grab the "CDrive," "DDrive," and "EDrive" files, we're really backing up the complete contents of what the virtual server sees as its C, D, and E drives. That would seem to be a good idea because we get *everything* the virtual server has.

The problem is that, using purely Backup 1.0 technologies, doing this task is kind of tricky. Those hard disk files are always open and always in use while the virtual server is running. The simplest thing would be to temporarily shut down the virtual server, back up all of its files, then start it up again—and that's exactly what many companies do. But that takes us back to "backup windows," point-in-time backups, and other Backup 1.0 downsides.

The second possible scenario is to run backup software *within the virtual server.* The backup solution would see a C, D, and E drive, and would back them up exactly as if the backup solution were running on a physical server. That means we would back up Exchange exactly as we always have, back up SQL Server normally, and so on. Nothing changes, really; the backup solution is unaware that it's running inside a virtual machine (VM), so it behaves as if virtualization doesn't really exist.

There are pros and cons to both the "inside the VM" and the "outside the VM" scenario. When you're backing up the entire VM from outside, you're getting *everything.* Essentially, those virtual disk files are disk images, and by backing up those disk files, you're enabling pretty straightforward restore and disaster recovery scenarios. However, you lose any kind of granularity—you're backing up *the entire server,* and that's all you can restore, too—*the entire server.* The fact that it's a virtual server makes it very easy to restore to a different location—just pick a different virtualization host.

The "inside" approach gives you granularity, meaning you can back up individual files and so on. However, you pick up some of the downsides to any physical server backup: In order to restore the server from bare metal, you have to install the operating system (OS) and your backup solution, at a minimum.

There's a twist to VM storage that complicates all of this—the Storage Area Network (SAN). Figure 7.2 illustrates the differences.

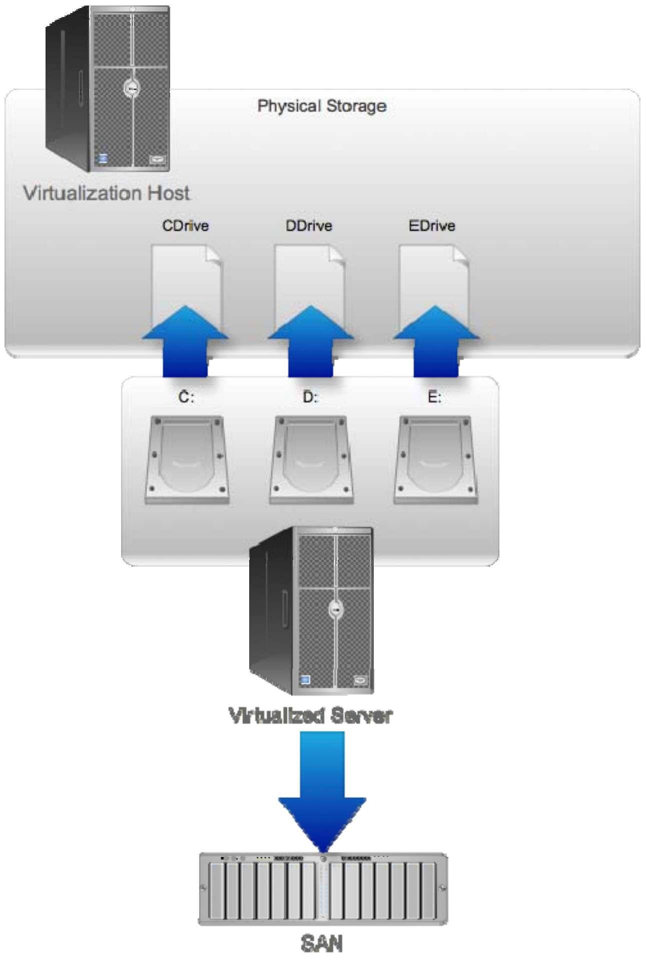

Figure 7.2: A virtual server using a SAN.

Virtual servers don't *have* to store data on virtual disks. They're also capable of connecting to a SAN, and storing their data directly on it—without creating the fiction of a virtual hard disk. But does that create new possibilities for backups? Not necessarily. SANs generally assign big chunks of disk space to a particular server for access, meaning whatever SAN disk space is being used by the virtual server can *only* be used by the virtual server. Many SANs do support the ability for backup solutions to *only read* the disk space assigned to a server, meaning the backup solution could grab the virtual server's files and folders while the virtual server was running—but that's adding a layer to your backup and recovery scheme.

Both the "inside" and "outside" techniques have pros and cons, but which one is best? Most importantly, which one comes closest to our Backup 2.0 manifesto?

> Backups should prevent us from losing any data or losing any work, and ensure that we always have access to our data with as little downtime as possible.

To answer those questions, let's look more closely at both techniques.

In the Old Days

Let's look at virtual server backup from a purely Backup 1.0 perspective, which is what most companies are using today. I've already outlined both the "inside" and "outside" techniques; now let's dive into a bit more detail. To keep things simple, I'm going to focus on Microsoft's Hyper-V virtual server hypervisor; obviously, the techniques I'm discussing are substantially similar on other virtualization platforms, but this book has had a nice Microsoft focus to this point, so I'll keep it there for the specifics.

Backup Techniques

Let's consider the "inside" approach first, which means you have a backup solution running inside each and every VM, grabbing the files and folders and data from inside that VM. This is a straightforward technique that a *lot* of companies use because it's exactly the same as the techniques they've always used for backing up servers. A backup solution can grab individual files and folders—subject to all the Backup 1.0 limitations that I've discussed in the preceding six chapters, of course.

A downside is that you're really devoting a bigger chunk of physical processing power to running backups. Imagine you have a smaller Hyper-V host running a mere five VMs. You run your backups during an evening backup window, and your backup solution consumes about 10% of each VM's processing power—which isn't much. At a physical level, however, those individual 10% bites might add up to around 50% of the *host machine's* computing power—just to perform backups. Figure 7.3 illustrates what I'm talking about.

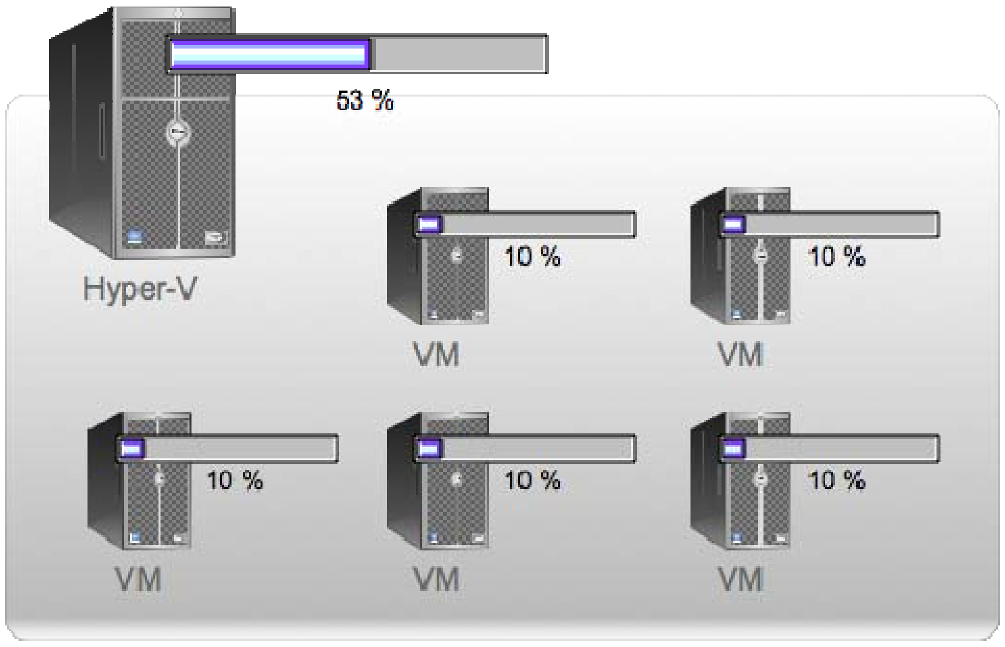

Figure 7.3: Processing power used for running backups.

I'm deliberately simplifying the numbers here just to make a point, which hopefully is clear enough: Running backups *inside* the VMs is, in aggregate, using a noticeable amount of physical processing power. It's kind of a waste, but it's getting you every single individual file and folder from within the VMs, meaning you can do very granular restores—to the single point in time when the backup was made, of course. It would probably not be efficient to run backups all the time from within the VMs; the aggregate use of physical processing power would be too high.

What about running a Backup 2.0-style backup solution *within* the VMs? What I've suggested is an agent, running inside the VM, that grabs changes to individual disk blocks as they are written to disk. Such an agent might add 2 to 3% overhead to the server, which is absolutely minimal. Again, however, that's per VM; that 2 to 3% would add up to a sizable chunk of physical processing power, which still seems like a waste. I don't mind dedicating 2to 3% of my computing power to backups, but dedicating 10% or more of a Hyper-V host's computing power—it just seems like there's something better I could be doing with that power.

Okay, so let's move to the "outside" technique, where we run some kind of backup solution right on the Hyper-V host. I'm only going to be losing 2 to 3% overhead *total* now, which makes me feel a lot less wasteful. But using purely Backup 1.0 technologies, all I can do is grab *entire VM hard disk files.* That's wonderful for disaster recovery, because the *entire* VM is exactly what I want in that scenario. But recovering an individual file or other piece of data gets really, really tedious. You also have to take each VM offline in order to fully back up each virtual disk, which is a huge interruption and ties you to a maintenance window—if your VMs can't, for business reasons, be offline every night, then you won't be grabbing a backup every night.

Hyper-V VSS Writer

Hyper-V does support Microsoft's Volume Shadow Copy Service, which slightly expands your options for "outside the VM" backups. Essentially, VSS allows Hyper-V to natively make "snapshots" of VM disk images while the VMs are running. This does exactly what you might suspect: It makes a *complete copy* of the virtual disk files, and you can then easily back up that copy while the VM continues running. You need to plan for the extra disk space on the Hyper-V server to take advantage of this, of course. Hyper-V even makes a snapshot of the virtual server's memory, so what you're backing up is literally a snapshot of the VM, including its current operating state.

After the first snapshot, subsequent ones are just "deltas," meaning they only include the bits that changed since the first snapshot. That helps conserve disk space, and VSS can expose the resulting snapshot by combining the base snapshot with the most recent differences.

This technology isn't *quite* as magical as it may sound. Virtualization products have had the ability to take snapshots like this almost forever; VMware Workstation was probably the first, and it was super helpful when you were using your VMs to do product demonstrations. You'd get everything set up the way you wanted, and take a snapshot. After each demo, you'd "roll back" to the snapshot, leaving yourself completely ready for the next demo. By building the capability into Hyper-V, Microsoft has simply leveraged this years-old technique for backup purposes.

But snapshots are still very specific points in time. You can't rely on the snapshots being kept entirely on the Hyper-V server; if the whole server goes kaput, you lose those snapshots, too. Snapshots are only *backups* when they're copied off-server, typically to tape. Many backup solutions are VSS-enabled, meaning they can "see" these snapshots and grab them during a backup operation. So you're still at-risk for any data that hasn't been moved off to tape; the big advantage is that you can *continuously* move recent snapshots off to tape because you're copying inactive files.

Of course, I've never met anyone who backs up their Hyper-V servers more than once per night, and most do it once a week. So that's a lot of data at-risk.

So it seems like these two techniques both have their upsides, but they both have downsides, too. Does either of them offer great restore scenarios that would make up for the downsides?

Restore Scenarios

Again, using purely Backup 1.0 techniques, both the "inside" and "outside" techniques actually offer pretty poor options for restoring data. Using the "inside" technique, you *still* have to hunt down the right backup tape, spin to the right file, and restore it. In the event of an Exchange, SQL Server, or SharePoint backup, you have to restore an entire database to an alternative location, open it using the native server application, find the bit you wanted to restore, export it to a standalone file, and so on. Tedious and time-consuming. Although Backup 1.0 is easy to pull off inside a VM, it still does a poor job of meeting actual business needs. You still can't restore to points in time other than your backup, meaning you stand to lose a great deal of data. When something does go wrong, you'll be spending a lot of time on recovery—pretty much the antithesis of our ideal situation:

> Backups should prevent us from losing any data or losing any work, and ensure that we always have access to our data with as little downtime as possible.

Using the "outside" technique *seems* like it would be a better deal. You're spending less overall processing power because the backup solution is running on the Hyper-V host. You're also grabbing the entire disk image in your backup, without the need for special Exchange-aware, SQL Server-aware, SharePoint-aware, or whatever-aware agents. All of your data is protected. But you're still dealing with backup windows—typically, even if you're using Hyper-V's VSS integration, you're still only grabbing one backup a night or per week, which means you have a lot of data at-risk. Also, restoring data means *restoring the entire VM.* Although it's easy to restore that to an alternative location—you do have a spare Hyper-V server sitting around for that purpose, right?—you have to wait for the restore to run, start up the restored VM, get whatever it is you needed, and so on. Doing all that to recover, say, a 5 kilobyte email that someone accidentally deleted is a horrible waste of time, akin to rebuilding an entire hotel just to figure out which bit of carpeting caught on fire and burned the place down.

Mountable Images

There's a little ray of sunshine when it comes to Hyper-V that I'll let you in on. Hyper-V VMs store their disk images as .VHD files. Microsoft has opened the .VHD file format specification, and they've created a number of utilities that let you mount and browse offline .VHD files.

So here's the scenario: You're grabbing point-in-time backups of the .VHD files by running a backup solution on the Hyper-V host. Perhaps you're using VSS to back up live snapshots—great. When restore time comes, you don't necessarily *have* to restore the entire VM, start it, and go into it to recover whatever you wanted. Instead, you can just get the latest .VHD file off of tape, then "mount" that .VHD as a read-only disk drive on another Windows computer—Windows 7 can actually do this natively.

This still isn't a perfect situation: You're restoring from a point in time, and if your backups are far apart, you still have a lot of data potentially at risk. You still have to spin the .VHD file off tape and find a place for it to live. But you can *mount* it without having to actually run the original VM, which can save a lot of time. For example, if the original VM had three .VHD files, you only need to restore the one you want something off of rather than restoring all three so that you can start the VM normally.

So virtualization backup techniques in the Backup 1.0 world aren't any better than the non-virtualization backups in that same world. Maybe Backup 1.0 can offer a better disaster recovery scenario?

Disaster Recovery

In fact, it can. Hooray, one point for old-school backup techniques! In the "outside" backup scenario, where you're backing up entire VM disk images, you've got great disaster recovery options. Because you've backed up *the entire server,* possibly even including its memory state at a specific point in time, you can recover *the entire server* pretty easily. What's even better is the fact that VMs actually have very little idea what physical hardware they're running on. Remember, all a VM sees, for the most part, is an abstract, virtualized hardware environment. So if your entire Hyper-V server dies, you can just restore all your VMs to another Hyper-V server—or even spread them out across multiple other Hyper-V servers, even at different locations. Once restored, the VMs will start with little complaint because as far as the OSs *inside* the VMs are concerned, nothing has changed.

This ability to relocate a VM to any available Hyper-V host is the exact thing that makes "live migration" possible. In a live migration, a running VM is moved to a different Hyper-V host, often so that the original host can be taken offline for maintenance. It works.

Well, it works to a point. You're *still* limited to widely separated points in time for your disaster recovery because your backups are points in time; they're not *continuous*. It's great that you can restore a VM, even including its memory state, but if what you're restoring is 5 days old? You've still got some bad news to deliver to the rest of the company because odds are there's a *lot* of lost data.

There's more bad news, too. When I write about *disaster recovery,* I'm really writing about a *complete* server failure—meaning you've lost the Hyper-V host, too. If you're relying on backups of VSS snapshots, then disaster recovery will still have to start with re-installing Windows and Hyper-V. Although newer versions of Windows Server install faster than ever, it's still not instantaneous. It'd be faster to restore those VM images to different, already-up-and-running Hyper-V servers, but that assumes you *have* some of those with available processing power, disk space, memory to allocate, and so on. If you don't, then your disaster recovery will still be a little slow.

Backup Management

Managing backups is the same old Backup 1.0 story. You must shuffle tapes, make sure you know which ones are the latest, and if you're doing full+incremental+differential backups, you still have to make sure everything stays in the right order for a successful restore or disaster recovery.

Rethinking Server Backups: A Wish List

Let's once again forget Backup 1.0. Let's even forget Backup 2.0. I want to write about what I *wish* virtualization backups could look like. It might not be possible, but the only way we'll eventually meet all our business needs is to push the technology a little. My wish list is simple:

> Backups should prevent us from losing any data or losing any work, and ensure that we always have access to our data with as little downtime as possible.

Now, more specifically, here's what I want to enable my wish list:

- Continuous backups of VMs with as little *aggregate* overhead as possible—That means I'd prefer not to have to run backup solutions *inside* the VMs because that is essentially duplicated effort and wasted computing power.
- Full, as-fast-as-possible bare-metal recovery of an entire virtualization server, including the base OS, hypervisor, and all the VMs
- The ability to grab single files and folders from a VM backup, preferably without having to restore the entire thing, start the VM, and so on
- The ability to roll back a single VM, or *the entire virtualization host server,* to a particular point in time—I'm having trouble thinking of specific scenarios when I might want to roll back the entire host, but I'm not letting my lack of imagination get in the way: I want the *capability.*
- I don't want—and this is a big point for me—to have to wait for tape drives to copy files back to disk—Honestly, I don't like tape. Sure, you *have* to use it because it's the easiest way to get data off-site, but I prefer tapes to be "Plan B." In other words, I want my "Plan A" backup to be on nice, fast disk space somewhere in my data center; if my whole data center floods or whatever, I'm happy to deal with tape.

Why the Tape Hate?

It's just a user perception thing. If a user wants a file, they don't tend to understand why it takes 6 hours for me to get the right tape, restore a file, and so on. If, however, the entire data center floods, everybody in the company tends to be a bit more understanding about why things are taking so long—it's clearly beyond anyone's control at that point. That's why I prefer a hierarchical storage system: disk-based backups for day-to-day use, and tapes as a backup to *that.*

So what does virtualization Backup 2.0 look like? Is it technically possible to make this wish list a reality?

New and Better Techniques

I think it is. Thinking about the Backup 2.0 techniques I've outlined in previous chapters, here's how I'd imagine it working:

- You'd start with the "outside" technique—running the backup solution right on the Hyper-V host, and running nothing special inside the guest VMs. This technique involves the lowest processing overhead—probably less than 3 to 4% of the host's total computing power.
- You'd need to rely on de-duplication and compression because they make disk-based backups more practical (disk space is still more expensive than tape space). By taking up less disk space on a backup server, you're enabling more backups to be kept for longer.
- You'd simply monitor for changes to the disk blocks on the host, capture those changes, and transmit them off to the backup server. This is going to capture every change to both the host OS *and* to the VM disk images, in near real time. Figure 7.4 illustrates this. Essentially, everything that happens on a Hyper-V host comes down to blocks stored on the host's physical disk storage. By intercepting those changes at the Windows NTFS file system level—by means of a file system filter—you can capture the changed disk blocks (which are only a few kilobytes each in size) and transmit them to a backup server for long-term storage.

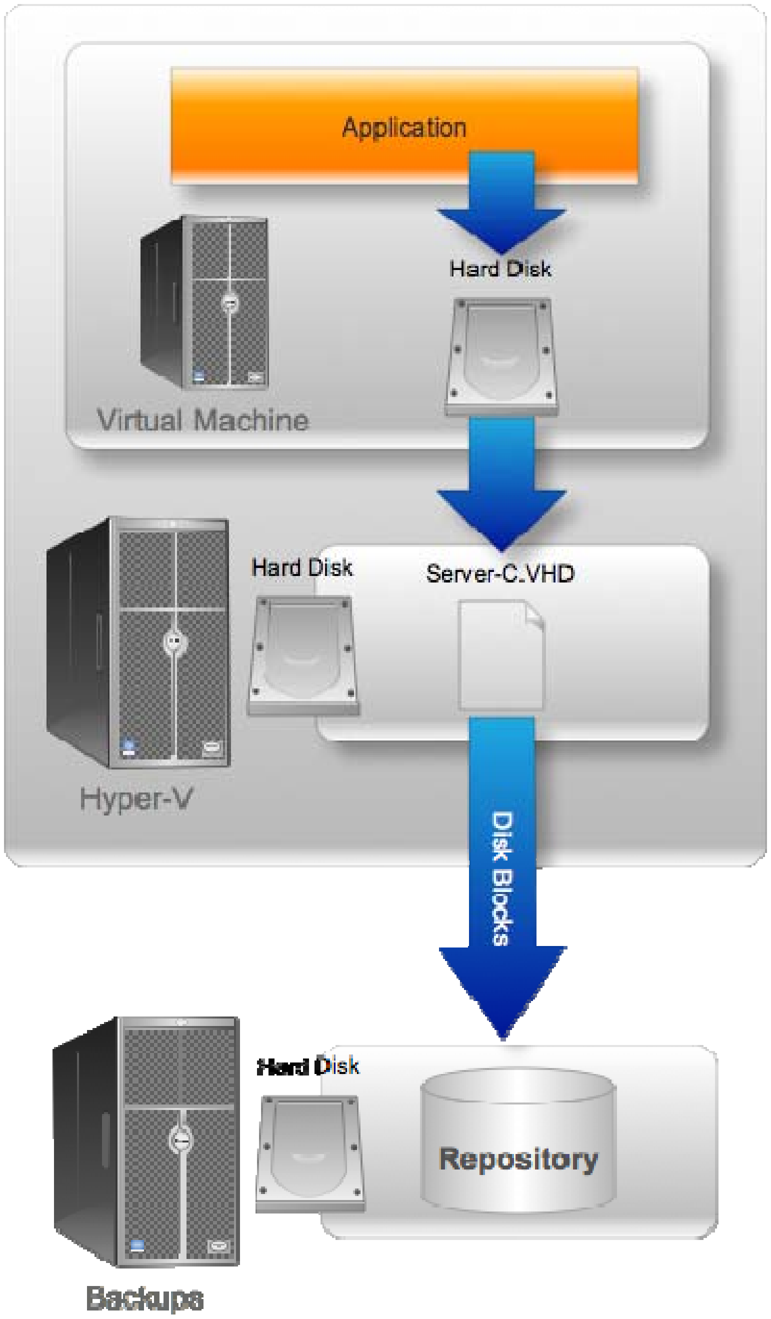

Figure 7.4: Disk block capture in a virtualization host.

The backup server "knows" which disk blocks go with which file. Therefore, it's not just keeping a giant repository of disk blocks; it knows which blocks comprise the host OS, which blocks make up a given VM's .VHD files, and so on. There are a few key facts that I need to highlight about this:

- We have a copy of every disk block that has ever changed on the file system.
- We know what time each change was made.
- We know what file each change went to.
- The previous three facts mean we can re-construct any file at any point in time, simply by grabbing the disk blocks associated with (a) that file and (b) that point in time.
- Windows already includes the technology needed to mount a .VHD file and browse it as an active disk.

These five facts lead to vastly improved restore scenarios.

Better Restore Scenarios

We *could* use a restore scenario like the one illustrated in Figure 7.5. Here, we've used the backup solution to assemble the disk blocks for a specific VHD file at a specific point in time. We've re-built that VHD file onto a file server on the network, then mounted that VHD file to a workstation. That provides access to the files inside the VHD, which can be copied to a "real" disk drive, then moved to wherever it is they need to go. The same technique could be used to recover a VHD and mount it as a drive on a SQL Server, and then SQL Server could attach a database from that VHD—a neat way to access an earlier version of a database.

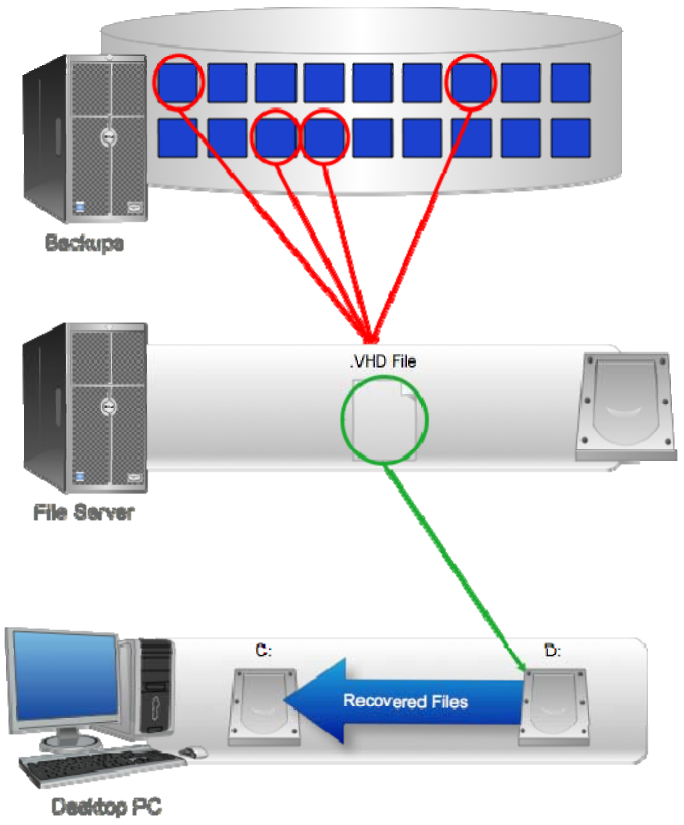

Figure 7.5: Restore scenario.

This is a great restore technique—but Backup 2.0 can do better. Remember, computers are supposed to *save us* work, not create intermediate steps for us. Why not simply design a backup server that could directly expose a specific set of disk blocks *as a file system*? In other words, we don't need to extract the disk blocks we want and re-construct a file; we should simply be able to dynamically access the desired disk blocks as they sit in the backup repository, exposing the resulting file through standard file system mechanisms. And, while we're at it, why not simply build in the technology needed to mount the VHD? In this fashion, the VHD would appear as a "shared drive" on the backup server itself. Figure 7.6 shows what I'm talking about here.

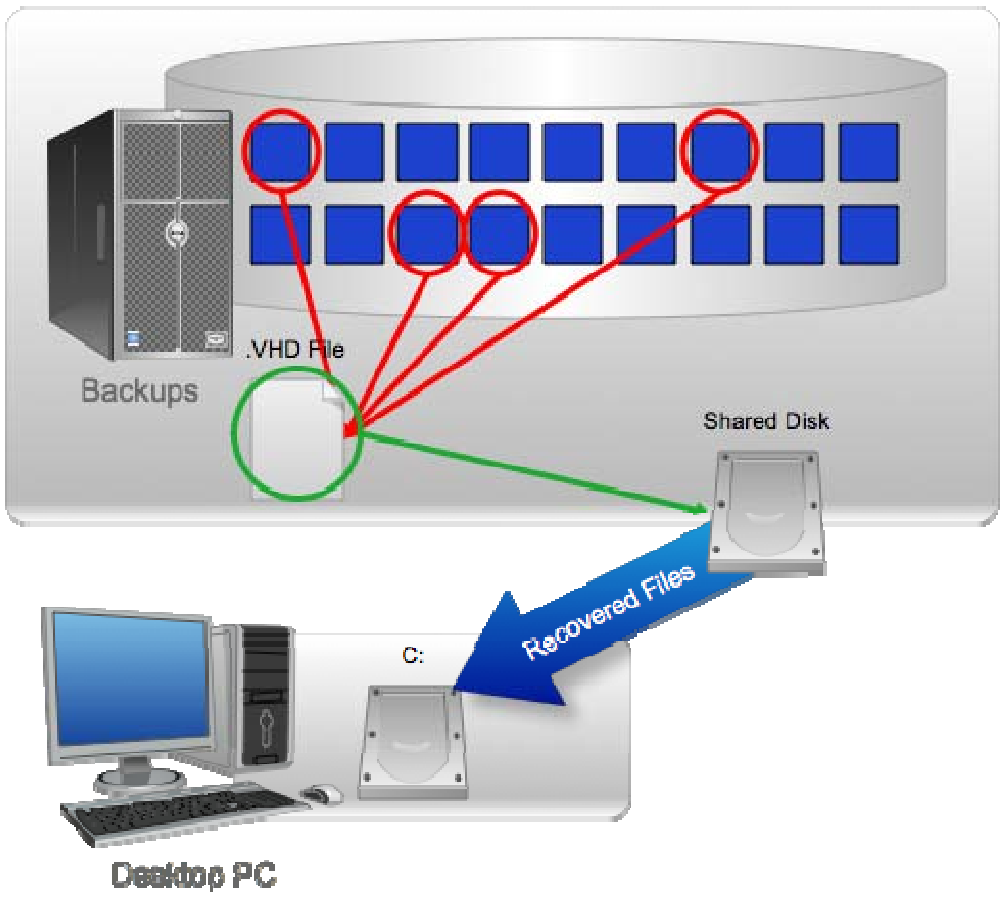

Figure 7.6: Faster, easier steps to restoring files.

The striking thing about this is that *it doesn't involve any radical new technologies.* Everything I've suggested here can already be done in independent ways; we're just combining them to create a better backup and restore capability for ourselves. By sitting down and thinking about what we actually *want* from backups, as opposed to what vendors have traditionally *given us,* we've been able to draw together existing technologies to create something infinitely more valuable.

It's the Block that Changes Everything

If you look at what I'm suggesting in Figures 7.5 and 7.6, it's really not that different from what Hyper-V and VSS can already do. VSS grabs snapshots of VMs, which you can then back up to disk or tape. You can restore those, mount a VHD, and recover files—exactly as I've suggested. Such a backup solution would run on the Hyper-V host, not within the guest VMs, and would use a minimum of computing power (albeit a lot of disk space because VSS doesn't support de-duplication or compression).

VSS' only downside is its scope: It focuses on snapshots of *the entire VM.* You can't just constantly take snapshots; there's too much overhead involved and way too much disk space.

My suggestion is just to get more granular and grab *disk block snapshots* instead of entire VM snapshots. That way, you're continually getting a low-overhead stream of data, and you're backing up *the latest changes* on a continual basis. That basically minor change in scope—individual disk blocks versus entire VMs—makes the near real-time backup possible, and that's what lets us operate with much less data at risk. *That's* Backup 2.0: continuous backups, not point-in-time backups.

Better Disaster Recovery

Disaster recovery—recovering an entire, bare-metal server in the event of a disaster—is straightforward. Simply grab the most recent disk blocks for the entire server and dump them back onto the server's disk. I imagine you'd have some kind of bootable CD or DVD with a minimal OS on it, capable of mounting the server's physical disks and receiving streamed-in disk blocks from the backup solution—as shown in Figure 7.7. You'd get the *entire* server—host OS, registry, all the VMs, and everything, to whatever point in time you needed.

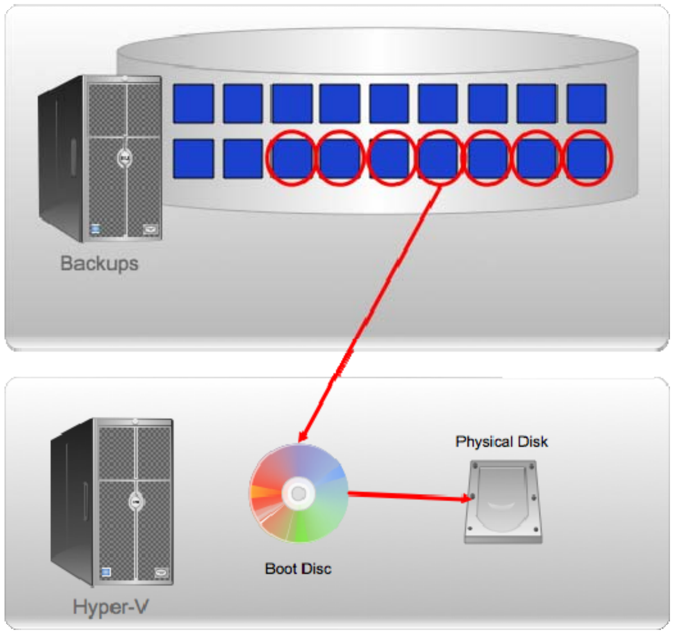

Figure 7.7: A bare metal recovery.

The neat bit is that you could recover to other hardware, too, provided the new hardware was of the same processor architecture, had the same basic configuration, and so on. That makes *real* disaster recovery—like offsite recovery—a real possibility.

This is the same basic disaster recovery model I've proposed in previous chapters. It occurs to me, however, that by adding a little replication action, you could *really* enable some interesting disaster recovery scenarios. For example, by replicating the contents of the backup repository to another physical server in a different physical location, you'd be assured of disaster recovery even in the event of a total facility loss. You could potentially replicate those disk blocks to an in-the-cloud service provider, who could potentially give you the ability to recover entire servers to VMs that run at the service provider's facility—creating a cloud-based disaster recovery facility. You'd need to do some math on the bandwidth required, but remember that data de-duplication and compression would help reduce the necessary bandwidth somewhat, and you could have a continuous replication stream at all hours of the day and night. It bears some additional thought to see if that's a practical addition to Backup 2.0's suite of capabilities.

Easier Management

Management overhead kind of goes away with Backup 2.0, or at least gets a lot simpler. I'm not suggesting that you'll never need tapes again, because you absolutely should be backing up the disk block repository to tape for off-site storage (unless you can get the replication thing figured out, which would be very cool). Figure 7.8 illustrates how tape backups would fit into the scheme; the thing to remember is that those are *redundant* backups. I imagine making a nightly backup of the day's newest disk blocks; there's no "backup window" because you're not backing up live data or taking production servers offline, so however long it takes to write the data to tape is fine.

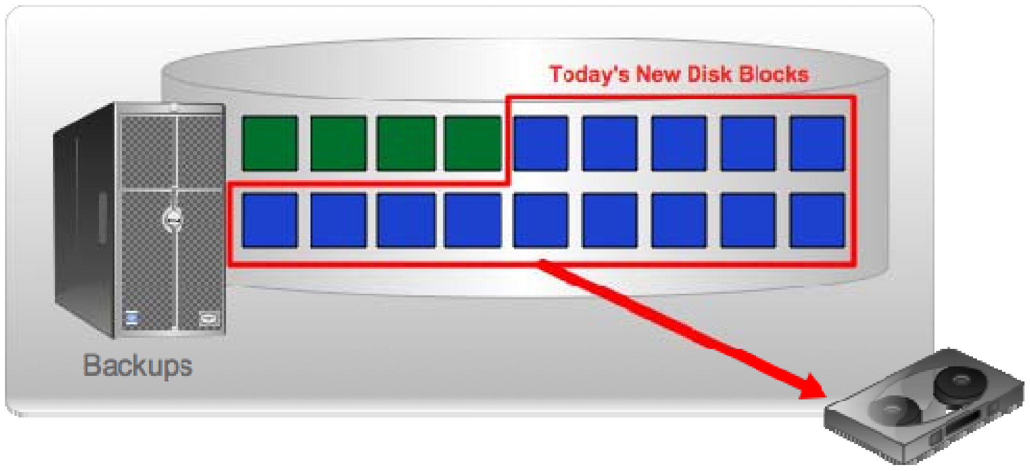

Figure 7.8: Making a redundant backup to tape.

Virtualization-Specific Concerns

Anytime you start talking about virtualization, there are specific concerns that come to mind. Virtualization is just a bit different than normal computing, and there are factors involved that we want to make sure we've fully considered.

Performance

Performance is a big one of those factors. As I pointed out earlier, simply running a backup solution *inside* each VM seems like a waste of computing power. All the things you need to back up are on one machine, but you're spending multiple times the needed computing power simply because each VM is responsible for the data under its control.

I think the "outside" scenario provides the answer to this. Provided you can back up the VM images from the host OS level—and you can—while still having granular access to the stuff *inside* the VM images—and I think you can do that, too—then this is the way to go. You're getting an aggregate computing overhead in the low single digits, which is completely acceptable.

Granular Recovery

Here it is again: granular recovery. I dislike a lot of traditional—Backup 1.0, that is—techniques simply because they're primarily geared toward full-failure disaster recovery models, whereas *most* of what we deal with on a daily basis is recovering a file or two.

I think I've shown how Backup 2.0 techniques—grabbing disk blocks in near-real time—can provide any level of granularity you need, from a full bare-metal disaster recovery to recovering a single file. Granular recovery should—and under Backup 2.0, *does*—require minimal effort; the ability of a Backup 2.0-style solution to dynamically expose a backed-up volume by dynamically assembling the right disk blocks on-demand—well, that covers "minimal effort."

Guests and Hosts

It's not enough to just back up the VM VHD files and related configuration files. I do realize that those VHD files are what contain the *data* you actually care about; without a virtualization host to run those VHD files, however, they're not nearly as useful to your business. Backup and recovery is all about *getting back online as fast as possible;* simply backing up VHD files does not accomplish that if it takes a couple hours to re-build a host for those VHD files to live on.

I've worked with any number of consulting clients who didn't quite "get" that point until they *did* have a complete server failure. Then, all the VHD backups in the world were useless until they'd built a new host.

I've shown how Backup 2.0 can easily scale to a full disaster recovery model by streaming disk blocks back to whatever physical server you want. This rebuilds not only the VM VHDs but also the entire host OS *and configuration*, meaning you can simply restart the server and pick up business where you left off—with very little manual effort.

Coming Up Next…

At this point, we've covered Windows Server, SQL Server, SharePoint Server, Exchange Server, and even virtualized servers. What else is left for Backup 2.0 to conquer? Plenty. Today's businesses have lots of other concerns and needs with regard to backup, and in the next chapter, we'll address some of them. For example, "restore" is a very different task than "disaster recovery," so I need to look at how Backup 2.0 might handle the latter. And what about high availability? You certainly can't afford for your *backup solution* to be down when you need it! There are other issues, too: security, compliance, data retention, mobile infrastructure, and more. And what if Backup 1.0 isn't completely dead? How can you integrate the 1.0 and 2.0 techniques into a consolidated backup plan—and why in the world would you want to do so? That's all in the next chapter.