Realtime publishers

The Definitive Guide™ To

Windows Application and Server Backup 2.0

sponsored by

appAssure
APPLICATION ASSURANCE

Don Jones

Copyright Statement

Chapter 2: 12 Horror Stories—We Thought We Had a Backup!

Horror stories. Tales from the trenches. Case studies. Call them what you will, I love reading them. They're a look into our colleagues' real-world lives and troubles, and an opportunity for us to learn something from mistakes—without having to make the actual mistakes ourselves. In this chapter, I'm going to share stories about backups to highlight problems that you yourself may have encountered. For each, I'll look at some of the root causes for those problems, and suggest ways that a modernized "Backup 2.0" approach might help solve the problem. Some of these stories are culled from online blog postings (and I've provided the original URL when that is the case), while others are from my own personal correspondence with hundreds of administrators over the years. One or two are even from my own experiences in data centers. Names, of course, have been changed to protect the innocent—and those guilty of relying on the decades-old Backup 1.0 mentality.

The important takeaway is that each of these stories offers a valuable lesson. Can you see yourself and your own experiences in these short tales? See if you can take away some valuable advice for avoiding these scenarios in the future.

Corruption: Not Just for Politics

I'll start with this story, as it's one I imagine every administrator has been able to tell at least once in their careers.

> This must be the oldest backup story in the world, and you have probably heard it a million times: We dutifully take our backups every night, even though it takes forever on the big machines. When we finally need to use one, the backup is corrupted. Either it's the tape, which is actually pretty rare, or something went wrong and the backed up data itself is no good. And so the boss wants to know why we even have jobs since we are all useless. Aaaah!

As I said, we've all been there. Two problems, both related to the old-school Backup 1.0 mentality:

- Backups shouldn't take forever; they should take constantly. That is, a point-in-time backup is just a snapshot. Even if the backup and restore work perfectly, you're still losing data. With continuous backups, you're not losing data—it's really that simple.
- Why don't backup programs tell you when they see corrupted data? This should be the simplest thing in the world—our industry has had checksums and other means of validating data pretty much forever. It drives me crazy that we have to discover problem backups by poring through log files.

This is why we have to rethink backups: *They just don't do what we need them to do.* Even when we do everything perfectly, there's a huge risk that the backups just won't work. We need to rethink what we're doing, how we're doing it, and how we're monitoring it to make sure it works.

Of course, I won't mention that *testing* those backups might have revealed our author's problem before he actually *needed* those backups. Another problem with the Backup 1.0 mentality is that nobody (well, hardly anybody—some of the upcoming stories do give me hope) seems to test their backups.

Crackberry Withdrawal

If ever one device has managed to somehow make email even more mission critical than it already was, Research in Motion's Blackberry family must be it. When something goes wrong with the Blackberry infrastructure, it's a race to see what you spend more time on: fixing the problem or answering the phone calls from users who are sure they're the only ones who've noticed the problem:

> I work in an environment that—like everyone, I guess—uses Blackberries and cannot live without them for one moment. The infrastructure is actually very complex: In addition to our Exchange Server, there's also the Blackberry Enterprise Server (BES) and a supporting SQL Server. The last time we had a failure, it took us 5 hours to get back online, which I thought was pretty good—but there was an inquisition afterwards to find out why we were offline for so long. Even with all the right tape backups, do you have any idea how long it takes to restore a SQL Server and the BES? Apparently too long for my boss.

It's the Backup 1.0 mentality: Rely on those backup tapes, even though just streaming the data off of tape can take hours—assuming it's all in good shape, not corrupted, and that the data on the tape is actually a good backup in the first place. This is exactly where Backup 2.0 methodologies can make a difference: With a disk block-based backup, streamed into the backup store in almost real time, you can restore an entire server to almost any point in the recent past by pushing a button. Does 5 minutes sound better than 5 hours? By restoring into a standby virtual machine, 5 minutes is entirely realistic.

Exchange Failure

Nobody likes it when the Exchange Server goes down. Honestly, I think some companies could go without a file server for longer than they could live without email—especially companies whose employees have Blackberries. So here's a story to move your emotions, drawn from http://appassure.ning.com/profiles/blogs/what-i-learned-on-my-worst-day?mkt_tok=3RkMMJWWfF9wsRow5%2FmYJoDpwmWGd5mht7VzDtPj1OY6hBssJJKtUg%3D%3D:

> When my company's Microsoft Exchange Server failed at the end of the quarter, it could not have happened at a worse time. It began with the VP of Sales yelling "Email is down, and customers can't send us their orders!" Then my Blackberry started going off, calls, emails, IMs—it was relentless. When I logged on to the Exchange Server, I found that some of my most critical mail stores were no longer mounted. When I tried to remount them, I received the ambiguous yet ominous **JET-1601 JET_errRecordNotFound** error message. I immediately connected to the replication server that runs at one of the company's remote sites, only to find that I couldn't mount those mail stores either.
>
> When I called Microsoft, technicians prescribed the standard procedure of running Eseutil. They warned me, however, that the error message probably indicated a corruption problem deep within the database and that running Eseutil might result in cleaning the stores of all user data. I took the leap, on the chance that it would be quicker than getting the restore process underway. Running Eseutil took hours, then failed with the even more ambiguous **JET -1003 JET_errInvalidParameter**. At that point, I knew I HAD to go to the backup.
>
> My company runs full backups every Saturday night and incremental backups the rest of the week. I started by recovering the most recent full backup, then applying the incrementals until I had the backup from the night before the failure. As you can imagine, the calls, emails, etc. kept coming all the while I was copying the mail stores from my disk to disk backup—although they did taper off a bit after 11:00pm, when our West Coast office closed.
>
> Once our data was back on the primary server, it was time to roll the logs and mount the database. However, when the logs were about 80 percent applied, they failed with the **JET -501 JET_errLogFileCorrupt** message. At that point, Microsoft support could only suggest running Eseutil through my entire log chain, noting the corrupted log, deleting anything except log files from the log directory, and deleting the corrupted log and all the logs created thereafter. Then I could finally restart the log roll operation from scratch. This procedure took more than 6 hours. In the end, my company lost 2 days of email messages, and recovery took more than 30 hours. The cause turned out to be a problem with the RAID controller driver that had taken months to manifest itself after a previous server upgrade.

> Executive management figured it cost the company about $50K, so they definitely wanted to know what had happened and how it could have been prevented—and how it would be prevented from happening again. Let's just say "wanted to know" means that if I didn't have a good answer, my name was going on the top of the next layoff list. I was seriously committed to finding a better recovery solution.
>
> So here's what I learned on my worst day as a network admin: You can have multiple copies of your data—on replicated servers, on disk, and on tape—but if you can't mount the copies, they aren't any good.

The Backup 1.0 mentality cost this company $50,000. Why? *Because the Backup 1.0 mentality focuses on making backups, not restoring data.* With Backup 1.0, we tend to focus on backup windows, tape drives, and so on—we don't tend to focus on what will happen in the event of a disaster. Even with their backups and 30 hours of effort, the company still lost 2 days of emails—this is a backup plan?

Exchange Server is certainly a complex and difficult product when it comes to backups. The back-and-forth between the Exchange Server product team, the Windows product team, and Microsoft's own backup products (in the System Center family) means that Exchange and Windows alone don't offer an effective backup plan. Third-party vendors, however, tend to focus on Exchange-specific agents that just make copies of the data *as handed over by Exchange.* As this horror story points out, Exchange might not always be handing you good data, meaning your backups are useless.

So how would Backup 2.0 change things? We'll cover Exchange Server backups in detail in Chapter 4 of this book, but for now, let's just compare this situation to my wish list form the previous chapter:

- This fellow's situation could have been improved immeasurably by having *continuous* backups rather than point-in-time backups of log files and databases.
- *Block-level imaging*, which grabs the changes as they hit the server's hard drives, would have provided an uninterrupted stream of backup data all the way to the point where Exchange's data files were corrupted. The server could quickly have been "rolled back" to that point in time.
- Exchange Server's native backup technologies could still have been used, if desired.
- Our poor admin could have *tested* his restore capabilities more frequently, been more familiar with the process, and likely spent far less than 30 hours getting his server back online.

Remember: 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. The Backup 1.0 mentality originally employed in this horror story certainly didn't meet any of those criteria.

Migrating the Cluster—Or Not

Sometimes, "disaster" doesn't always mean a failed server. Sometimes a solid lack of planning can provide all the disaster you need!

> We received the new servers for the new cluster. The job: Swap out an old Windows for a brand new one, configure it, and move all the files and data over to the new cluster. This has to be done within a late-night maintenance window, which means my wife will not be happy again, but I'll buy flowers. Start time is around 11:00pm, and it must be done by 7:00am—8 hours.
>
> Both clusters would eventually be running the same version of Windows, once I got everything installed. The cluster servers came with no OS installed at all, though, so installing Windows would be my first step. Fortunately, the server hardware was basically the same in both clusters.
>
> After driving 300km—a horror driving on Polish roads—I set up the server hardware. The old cluster is humming along next to me, and I'm ready to get Windows installed. "Where are the drivers?"
>
> And the problem arose: Someone had lost the server manufacturer's drivers disc. I know what you're thinking—just download them, right? Well, suffice to say that this is a very secure organization—perhaps governmental—and nobody in the building at that hour could actually get to the Internet. So I had to pull out my mobile phone and, despite the high cost of data transfers, download the drivers for the server over the 3G cellular network. Then my phone's battery died—and me without my charger.
>
> I asked them about server backups, figuring we could perhaps just use those to restore to the new machine, but all they back up are the files. They said they had never been able to do a bare-metal restore using their backups, so they just stopped backing up the operating system (OS).
>
> Internet access was available again at 8:00am, and someone had to take me into the secure area where Internet access was available. I was tired, the entire night was wasted, and I had to do it all again the next night when I finally had the drivers in hand.

Ouch! We've all had a late night like that at some point, and it's never fun. And the first thing I have to ask myself is why they couldn't simply take a backup of the old cluster machines and restore them on the new hardware? Because in the Backup 1.0 world, restoring to dissimilar hardware is often impossible or at least "not recommended." As a result, many organizations just back up their files—but a backup is useless without someplace to restore it. A more modern Backup 2.0 mentality, however, would say that this cluster migration was really no different than a bare-metal restore after a complete cluster failure—why not restore the backup to the *new* hardware, and call it a night? In the Backup 2.0 world, we'd be taking block-level backups of the *entire server*, so we could just apply that backup to the new hardware (which we're told was substantially similar to the original hardware) and call it a night. Less time on the job, a lower cell phone bill, and a less frustrated wife.

Virtually Exchange

It's becoming more and more common to run server software inside virtual machines, often on a virtualization host running VMware or Microsoft's Hyper-V. But you'd be surprised how this scenario might impact your disaster recovery situation:

> After just 2 weeks on my new job, our Exchange Server decided to operate in "comatose" mode and stop sending and receiving email. I found that it was running on a virtual server and was being backed up daily by a third-party software package. I figured I was fine.
>
> The backup administrator, it turns out, really had no idea about how Exchange worked or what, if anything, was needed to complete the backups properly. The backups had never been tested, so I was starting to think fondly of my old job.
>
> I was able to get the server partially operational but wound up having to call the software vendor—only to get a refund for my support call and a terse statement that they didn't (at the time) support Exchange running in a virtual environment. Oops.
>
> I wound up having to 'port my half-functioning system to a physical server and was eventually able to recover everything except for 2 days' worth of email, which does not endear you to your new bosses, let me assure you.

"Not supported in a virtual environment" is kind of a cheap trick for a support organization to pull, but it happens. That's why it's critical to work with software and backup solutions that *explicitly* support virtual environments. In a Backup 2.0 world, virtualization is *expected;* anyone who thinks that virtualization is still novel or unusual is a dinosaur.

Of course, it's also critical to *test* those backups, and frankly the backup solution—once installed and configured—should just work. If there's something "special" needed to make a valid Exchange backup, the *solution* should know about it and take those steps automatically—that's why it's called a *solution*, not an *additional problem.* The right vendor will have all the knowledge they need on their own staff, and should make a backup tool that does the job you paid for. It should make testing those backups easy—especially in a virtualized environment where you don't even have to come up with another physical machine to do a test restore on!

For Want of a Check Box

Windows' native backup capabilities are, and always have been, pretty primitive. They're also firmly entrenched in the Backup 1.0 mentality of "just make a copy of the necessary data." Sometimes, that mentality can really bite you where it hurts:

> It was late 2006, and I was working for a branch of the US government. One afternoon, we began receiving large numbers of calls at the Help desk that people were having trouble logging into a particular Active Directory (AD) domain. Now, this particular domain was used for certain special projects, and only had two domain controllers, but they did support about 1000 users—all of whom were pretty upset that they couldn't log on and access the resources they needed.
>
> Both domain controllers, it turned out, had somehow been corrupted. I never figured out what happened, but I was even having problems logging on to the console directly on either computer. So along with a couple other admins, I decided to just do a full restore of the entire domain. After all, that's what backups are for, right?
>
> To my horror, I discovered that while the server was indeed being backed up every single night, nobody had ever checked the "System State" check box to ensure that AD was backed up too. So the backups were essentially useless.
>
> It took four administrators 96 hours, working around the clock in shifts, to rebuild that domain by hand. We slept an hour or two at a time at our desks, then got back up and started working again. We started on a Tuesday and didn't leave until early Saturday morning—all wearing the same clothes we'd work in on Tuesday. Somebody at some point had the presence of mind to go pick up some toiletries for us, but it was the longest Tuesday I can remember—all because one stupid check box wasn't checked.

This drives right back to the Backup 1.0 mentality: Only back up what we absolutely need. That attitude comes from a few facts of life in a Backup 1.0 world:

- "Backups take a long time." We seek to minimize the time it takes to pull a backup, so we cut out "unnecessary" files, folders, and systems. In this case, the "unnecessary files" were unfortunately the very thing that was actually needed.
- "Backup windows are limited." The very concept that we can only back up our data during certain dead or slow times is crazy, when you think of it. Why wouldn't we be backing data up while it's changing so that we can capture every change? What about users who might have just been created on the day of the failure—they won't be in the previous night's backup, so we lose all that work?
- "The backups are on a schedule." You hope. Too often you find out—when it's too late—that the backup wasn't working like you thought. The problem is that the schedule is usually late at night when nobody's around to notice a problem—and nobody thinks to check *every morning.*
- "We don't need to test our backups." This translates as, "it's too much of a pain in the neck to test our backups, and we don't have the time, so we're not going to do it." Testing, of course, is how you figure out that your backups *aren't working*, and find out *before* you actually need to rely on them.

A solid Backup 2.0 mentality changes things around:

- "Backups are continuous." They don't "take a long time" because they're always running, grabbing every little change that comes down the line.
- "Backup windows are unlimited." They're 24×7×365, in fact, because the "backup window" is always open, using a solution that grabs *every* change *as it happens.* This point, combined with the previous bullet point, means you just go ahead and *back up everything.* When you're not picking and choosing data to back up, you wind up always having whatever you need to do a restore.
- "The backups aren't scheduled." They're always running, and you can always see that happening in real time.
- "It's easy to test backups." Whether restoring to a virtual server or a physical one, testing backups should be easy and fast so that those tests can become routine. If you're not going to test your backups, why bother making them in the first place?

Weight Loss Plan

This one is just funny—I had to share it. Taken from http://appassure.ning.com/profiles/blogs/backups-what-backups:

> I was lead tech for a big server upgrade for a company that had to keep records for 7 years. We were imaging the old server data up on the network, then re-imaging back down [to the new server].
>
> Well, one day I thought I would multitask four machines at the same time and go to lunch. When I came back from lunch, I did a wipedisk of the old hard drives and noticed that one of the techs kept leaning over the work bench; his belly was so big that it would hit the space bar, which would cancel the data transfer.
>
> Well, to say the least, I found that he had done this on the day I went to lunch. Well, no one had any backups, so 7 years of data was gone.

I know it's not funny, but…it kind of is. The lesson to learn here, though, is that point-in-time images are really no better than an old-style tape backup. *Anything* that is just making a copy of the data at a particular point in time is Backup 1.0 mentality, where we're more concerned with getting a copy of the data—and, as we've seen in this story, a lot of strange things can go wrong to interrupt that copy and make it useless.

So how would Backup 2.0 solve this problem? By having a *continuous backup* of the old server in the first place. There'd be no need to take an image during a server upgrade, and in fact, the upgrade could be done faster and more smoothly by just relying on that block-by-block, Backup 2.0-style backup. An upgrade is really no different than a bare-metal disaster recovery—just perhaps less urgent. So a good backup solution should be able to assist with a server upgrade or migration—all the more reason to have a good backup solution in place.

So, 640KB Isn't Really Enough for Everyone?

The amount of storage used by modern organizations is truly staggering. In fact, getting enough time—and network bandwidth—to back up all that data can often be impossible. Impossible, that is, when you're living in a Backup 1.0 world that's focused on point-in-time copies of data:

> I am the only Storage Administrator for a medium-sized enterprise. We have more than 280 terabytes of storage used on various vendors' equipment. When I first started, we were running all of our backups through a silo with LTO1 drives in it.
>
> We couldn't ever back up all we needed to in the backup window we were given. We finally upgraded to LTO4 drives after several failed attempts to recover critical data and servers, along with complaints about network slowdowns, which, of course, came right to me.

A bit of digging revealed that the company was actually pulling weekly full backups of 30 to 45 terabytes, which each required about 14 hours to complete. 14 hours! And obviously someone is having to pick and choose the data that gets backed up because they're only backing up about 16% of their data.

Once again, it's the most insidious points about the Backup 1.0 mentality: You have to grab backups during some fixed point in time, you only have a certain amount of time to work with, and you have to grab what you can during that window. Backup 2.0, by *continually* grabbing changes *as they occur* can back up much larger data stores, keep the backups entirely up to date at all times, consume less network bandwidth in doing so, and grab *all* your changes—not just the data you can grab in a 14-hour window.

This story brings out some other weaknesses of Backup 1.0. Notice that the author started with LT01 drives but eventually had to upgrade to LT04 drives for the improved speed and reliability. Backup 1.0—by forcing us to live within backup windows—often forces us to spend a *lot* more on our backup infrastructure than is necessary. You could easily drop $15,000 on an LT04 tape library equipped with a fast Fibre Channel 4Gbps network interface—and every penny of that expense is simply to allow you to cram more data into your backup window. But if you expanded that window to 7×24×365 through continuous backup, you'd be able to capture everything—on significantly less-expensive hardware.

SQL Server Syndrome

I've worked with SQL Server for many years (since v6.5, I think), and organizing backups has always been a challenge. My story is a lot like this one:

> I have backups in three versions of SQL Server, in development, production, and testing environments, on multiple servers. Lots of backups, in other words—probably 45 to 50 instances total, with as many as 45 databases per instance.
>
> We grab full backups every night, and transaction log backups every 15 minutes. So our maximum data loss, if everything goes right, is 15 minutes. We test our backups by restoring the full backups to our test environment. That helps keep the test environment closely matched to the production environment, which is valuable for the software developers, and helps us make sure those backups are working. We roll last week's backups into a different folder and grab those to tape.
>
> We get failure notifications through email and System Center Operations Manager, and we rely on manual inspection of backup jobs and folders to make sure everything works.
>
> Right now we're trying to play with different backup schedules and look at using differential backups, all to lessen the network traffic and the drive space occupied by the many full backups. However, we need to keep our recovery times short, so we're testing to see how much overhead the differentials add to recovery times.

Have you ever gone on vacation for a couple of weeks and forgotten to put a hold on your mail? You come back to an enormous pile of mail and spend hours going through it all. But dealing with that same mail spread out over the course of a week is much easier, right?

Backups are the same way. The Backup 1.0 mentality tells us to wait until the evening or the weekend to grab all of our data. That means we have to hammer the network hard to pull all that data over to the backup server quickly (after all, the time window is only so big). Wouldn't it be easier if we could just stream the changes over constantly, as they happen? Reading one postcard a day isn't a big deal; coming back to a stack of them at the end of the vacation is what's painful. "Streaming the changes" is what Backup 2.0 is all about.

Remember, too, the goal of backups: 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.

Losing 15 minutes' worth of work, just because that's when you made your last transaction log backup, is nuts. Why should *any* data be at risk? *Point-in-time backups always place data at risk;* continuous backups don't. I want to revisit my Backup 2.0 wish list, from the previous chapter, one more time—and see how it applies to this case study:

- Continuous backups eliminate backup windows—no more at-risk data, no more pounding the network during a backup window.
- Backed-up data can be moved to tape on a daily basis for safekeeping without all the cumbersome rolling of files. Who became a DBA so that they could spend time managing files on disk?
- Being able to restore either a single database or an entire server would provide the test environment with a lot more flexibility, in addition to being an excellent test of restore capabilities.
- Block-level imaging would allow any database to be rolled back to a point in time—without having to figure out what combination of differentials, full backups, and log backups is needed. What's more, block-level imaging permits restoration to *any* point in time, whereas our author's transaction logs only provide "rollback" in 15-minute increments.
- Still need those old-school backup files for other purposes? Fine—a true Backup 2.0 solution won't interfere with them.

It just seems as if Backup 2.0, as a set of practices and capabilities, is much more well suited to SQL Server than the old-style Backup 1.0 mentality our author is relying on. We'll look at SQL Server in more detail in Chapter 5.

Patch Problem

This story really illustrates why I dislike point-in-time backups so intensely. Sure, you can achieve a lot of business goals using Backup 1.0 techniques and tools, but you have to be so *very* careful in order to get exactly what you want. Who needs that extra mental overhead?

I work in one of my company's larger data centers, and support about 100 servers. Most of these are file servers, but there are a couple of domain controllers, a few SQL Server machines, and three Exchange Server boxes.

We are very good about patching our computers. We typically will not apply a set of patches until we have a full backup of the computer's OS and application files, and we usually make a backup right after applying the patch, too. We tend to apply patches during maintenance windows when the servers aren't otherwise available. On some of our larger servers (the Exchange machines come to mind), it gets difficult to grab one full backup and apply patches during our 6-hour maintenance windows (you try taking email away from people for longer), so sometimes we take a backup one night and apply patches the next, then take a second backup the following night.

The system works—but not always well.

I can recall a couple instances where Exchange patches have caused problems with some of our third-party software, and we needed to roll back to the pre-patch backup. Unfortunately, a whole day of work had passed since that backup was made, so we lost all that work. People become incredibly unhappy when email goes missing.

In at least once instance, we didn't realize a general Windows hotfix was causing problems for about a week. At that point, the pre-hotfix backup was pretty aged. This was on a domain controller, so we decided to apply the old backup anyway, knowing that the domain would bring itself up to date through replication. Unfortunately, the backup also—we found out—had some deleted objects in the domain, which were near the end of their tombstone life. The practical effect was that about a dozen formerly-deleted objects suddenly reappeared in the domain. Our security auditors freaked out, people were yelled out, and it actually took us a while to work out what had happened, since that's not a scenario you see every day.

We've since decided to rely less on backups for undoing patches. We've started spending more time testing patches, which is of course a good idea but it's very boring and it takes a lot of time we didn't really have to spare. It also means our patches only get rolled out about every other month, rather than every other week, and I worry about what happens when one of those patches fixes some major security hole—and we have to leave the hole open for 2 months just because of our processes.

Again, the Backup 1.0 mentality has deeper-reaching effects than just disaster recovery problems. In this instance, the company has actually decided to run out-of-date software for longer *simply because of the way their backup processes work.* Unbelievable. If ever there was a case of the "technology driving the business" rather than the other way around like it should be—this must be that case.

There are easy-to-recognize problems here, which should be familiar to you at this point:

- Backup 1.0's point-in-time snapshots don't provide much granularity when it comes time to roll back something.
- Backup 1.0's reliance on backup or maintenance windows took away some of the author's flexibility with regard to his Exchange infrastructure.
- Except in a few special situations, Backup 1.0 tends to be an all-or-nothing proposition: either you roll back the entire server or you live with what you've got. There aren't many good ways to restore a single *application*; Backup 2.0, by contrast, can more easily pull out just the bits related to a specific application and restore it—with a single click.

Virtual Hot Spares

Virtualization is being used in more and more creative ways—and it's sad when Backup 1.0 can't keep up. Here's an excellent story about using virtualization to provide hot spares for critical computers—I can see this technique eliminating the old-school rental facilities where you'd have a bunch of physical servers ready to act as standbys in the event of a disaster. But see if you can spot where Backup 1.0 methodologies wreck the elegance of the solution:

> My organization used to rely on an outsourced "hot site" for disaster recovery. The theory is that, in the event our data center was hit by a meteor or something, we would relocate to this offsite facility. They have lots of servers handy, and we'd just restore our latest backups to those servers. Sure, we'd lose some data—but we wouldn't lose it *all.* We could then operate out of that site until our own data center was brought back online.
>
> The cost for these facilities can be staggering, so we've recently constructed our own recovery center in one of our larger remote offices. We can't afford to buy all the servers we'd need, so we are relying on virtualization. We've identified our two dozen or so most important servers, and we have enough servers in the spare facility to *virtualize* all the critical servers. Performance won't be tops, but in the event of a disaster that serious, we're okay with the tradeoff.

> To keep these hot spares working, we regularly take offline our critical servers for maintenance and do a physical-to-virtual (P2V) conversion, converting each physical server into a virtual machine in the spare site. We actually do the P2V conversion locally during the maintenance window, then copy the new virtual machine images later because the files are of course huge and the WAN can't support giant copies like that very quickly.
>
> In theory, that means our critical servers are always ready to go and are no more than a week or so out of date. Our plan would be to restore our more recent backups to each virtual machine to bring it even more up to date.

Great plan—almost. Having to pull servers offline to do a P2V migration is nuts. Why take servers offline at all? The bones of this method are a good idea, but the whole Backup 1.0 "snapshot" mentality—which most P2V migrations play into—is messing things up.

Consider this instead: Use a Backup 2.0-style solution, which makes continuous, block-level backups of your source servers without taking them offline. Restore those backups to an empty virtual machine—one without even an OS installed. This is essentially the same as a "bare-metal" restore, just that the metal in question is virtual. Your backups will always be up to date to the latest changes, and you can do a weekly restore to your "hot spare" virtual servers so that they're ready to go at a moment's notice. Or, if your backups are being stored safely—so that a complete disaster in your data center won't also take out your backups—you could just do yet another bare-metal restore when disaster strikes, and your hot spares will be virtually indistinguishable from the real servers.

By the way, where to keep your backups so that they're safe is obviously a key part of the strategy—and it's something we'll examine in more detail in Chapter 9.

Is it a Server or a Photocopier?

This scenario outlines a problem that a lot of us have, but that we often don't think about. It's not uncommon for companies to exercise storage resource management on their file servers—prohibiting certain file types, for example, or restricting users to a certain maximum amount of disk usage. One oft-quoted reason for this kind of storage resource management is that storage is expensive to manage and maintain—especially backups. But it's strange that even companies with the strictest disk quotas rarely exercise any kind of storage resource management on the backups themselves:

> I recently started working for a new company and was pleased to see that they had a very well-implemented Distributed File System (DFS) infrastructure. I don't like mapping network drives, and the users in this company had learned to access UNC paths like \\Company\Sales\Files\November rather than relying on the good old S: drive.

The company was also using DFS replicas to help users in remote offices—which often have slower WAN links—get to critical files. If you're not familiar with it, DFS uses Windows' File Replication Service to copy a given DFS leaf to one or more other file servers. So accessing \\Company\General\Policies might actually get you to any one of a dozen servers that all host that same content. In this organization's case, each local office file server has a copy of this and other commonly-accessed file shares, like \\Company\General\Sales\Forms. The files in these shares aren't updated really frequently (maybe a couple files a day change), so there isn't really much replication traffic, and having local copies lets everyone get to the files without relying on the WAN.

Like any good company, we back up all our servers. Smaller remote offices actually have direct-attached tape drives for this purpose. They've been using this model for years, but only recently have we started realizing that the backups weren't completing because the tape drives didn't have enough capacity.

I started looking into it, and realized that the company has been doing a *lot* of DFS replicas—about 50 to 60GB worth right now, and they're adding more all the time. This is the same data being backed up over and over again in different locations. All that duplicated data is what's causing the problem. I started trying to figure out exactly how much duplicated data we have so that I could make a business case to management. In doing so, I found that many of our file servers have multiple copies of the same files, all on the same server. A lot of the time it comes from users' home directories: the two servers that host the \\Company\Users DFS node have *tens of thousands* of duplicated files because users drag commonly-used files into their own home folders.

We were backing up a total of 13.2TB of data, and close to 30% of it was duplicated data. One-third! We're currently trying to figure out how to exclude the duplicates, but it's actually very tricky—we can't just not back up users' home folders!

Data duplication—the bane of storage resource managers everywhere. In fact, data *de*-duplication is an incredibly hot new technology; so much so that industry giants like EMC, Dell, and HP battle it out over acquisitions they believe will put them at the forefront of this important new technology. So important, in fact, that I'm going to add a bullet point to my Backup 2.0 wish list:

- Backups should not contain duplicate data. Back up something once, not more than once.

This is a tall order, and it might only be possible to implement it to a degree. For example, a solution might de-duplicate data across an entire server but allow duplicates of that same data to be backed up from other servers. Other solutions might take a broader view and de-duplicate data across the entire organization—that would certainly be powerful, but it seems like a technologically-difficult task.

Horror Stories and the Lessons They Teach Us

These twelve stories have obviously had common themes. Before I wrap up this chapter, I think it's worth spending just a few moments reviewing those themes, focusing on the takeaway (what we should learn from them), and reiterating what we *should* be doing *instead* to provide better backup, recovery, and other capabilities for our organizations:

- **Point-in-time backups are horrible.** You're always going to lose some data if you have to rely on point-in-time backups, and who wants to lose data? Continuous backups, or continuous data protection, if you prefer, offer much more flexibility, less risk of loss, and generally less manual effort and overhead. Point-in-time backups also lack granularity—even those quarter-hour SQL Server transaction log backups leave too much data at risk and don't let you roll back the server to a precise point in the past, if needed.
- **Backup windows are horrible.** Taking servers offline is never going to add much value to the organization. Sometimes, for true maintenance, you have no choice; you do have a choice when it comes to backups. Backup windows force you to choose what data to back up—you can only grab what will fit within the window—place unnatural strains on the network, and drive a number of other constraining decisions that simply add no value to the business. Continuous backups don't require a window, so you are free to make better decisions without all the time constraints.
- **Duplicated data is horrible.** Nobody likes that we have to do backups, so why back up anything more than once? Duplicated data bloats your backups, requires more time, and more importantly requires more space for backup storage—space that might be expensive (especially for offsite storage), slow (like tapes), and so on. Look for solutions that can automatically de-duplicate data when making backups.
- **Anything less than the entire server is horrible.** Sure, your worst nightmare might be losing a single email from the CEO, but that's far from the only nightmare you need to prepare for. Don't make decisions about what's important—back it all up. With the right solution, that one all-inclusive backup will enable everything from single-file restores to bare-metal server recovery—and everything in between, including restoring an entire application to an earlier point in time.
- **Backups get corrupted—which is horrible.** Backup solutions should tell you when something goes wrong—not wait for you to find out for yourself.
- **Not testing your backups is horrible.** True, old-school backup solutions make it incredibly difficult to actually test backups, but that's no excuse. Or rather, it's an excuse for finding a solution that makes testing backups easier, like one that supports restoring to a virtual machine or dissimilar hardware, so that you can practically test your restores.

There's plenty to learn from these stories, and plenty to look for in a Backup 2.0 solution. The key to the whole thing seems to be this idea of *continuous data protection,* grabbing individual blocks off disk as soon as they change and storing those blocks in a way that makes it possible to restore an entire server or just a single file. Yes, solutions like SharePoint, SQL Server, and Exchange Server add some complexity to the picture, but by continually streaming changed disk blocks into the backup system, you can grab *any* type of backup you need—continuously, without the point-in-time troubles of Backup 1.0.

Coming Up Next

It's time to dig into this Backup 2.0 philosophy a bit deeper. In the next chapter, I'll start looking at backups beginning with the most common type of backup—or what *should be* the most common type of backup: whole-server backups. Whether you use them to protect data or to prepare for a complete, whole-server disaster, these backups should be the staple of your disaster recovery plan.

I'll dive into the technical details and hurdles that whole-server backups present, and cover some of the native solutions that Windows provides. I'll outline specific problems and challenges that both native and third-party solutions have to deal with, and look at some of the Backup 1.0 methodologies you're doubtless familiar with. That will all help set the context for a discussion on rethinking server backups: I'll make a wish list of capabilities and features, outline better techniques that more closely align to business requirements, and point out ways that Backup 2.0 can make backup management easier, too. I'll finish by examining specific scenarios—like domain controllers, public key infrastructure (PKI), and Web servers—where these techniques offer an advantage.