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 12: Tales from the Trenches: My Life with Backup 2.0

In the second chapter of this book, I shared with you some of the horror stories of Backup 1.0. I did so primarily as a way of highlighting how poorly our traditional backup techniques really meet our business needs. In this chapter, I want to do the opposite: share with you some stories of Backup 2.0, both from my own experience and from stories you readers have shared over the year-long production of this book. Names have been changed to protect the innocent, of course, but I think you'll find these to be compelling examples of how Backup 2.0 has been applied. Where possible, I'll share information about the infrastructure that goes with these stories so that you can see some of the creative and innovative ways Backup 2.0 is being used in organizations like your own.

Backup 1.0 Fired Our Administrator…and Backup 2.0 Promoted Me

I'm going to start with what I think might be my favorite story. This was shared by a reader, although as I mentioned, I'm making up new names for everyone involved.

> I work for a pretty large retailer, at the corporate office. We have about 2000 users in the office, many of whom work in our distribution center. We have about 40 or so servers, most of which are file servers. We also have a couple of Exchange servers, and of course Active Directory (AD). We also have a SQL Server that handles our main database of products, sales, and everything else. That database is pretty much the center of our universe.
>
> Our lead network administrator's name is Kevin. Kevin set up our original backup system, which uses software agents on each server to send backup data to a backup server, which writes the data to temporary files on disk and then directly to tape. We have what I guess is a pretty common tape rotation plan: We make a full backup of everything on the weekends and send that off-site, then keep incremental backups on-site from each night of the week.

> I had been pushing for a continuous data protection backup system—what you would call “Backup 2.0,” I guess, for a couple of months. I had finally gotten permission to buy a system for a limited trial, which I decided would involve our SQL Server (which is so important) and one of our file servers. Kevin was not thrilled that I was stepping into his turf with the backup stuff, but I pushed ahead and did it anyway. He refused to turn off his old backup agents, which was fine because I was just trying to do a sort of proof-of-concept. He wouldn’t even give me space on his backup server, though, so I had to make do with a somewhat older machine. I was able to put lots of disks into it, though, and I was able to set up a dedicated backup network for the four servers involved. I’m attaching a diagram of what things looked like then [see Figure 12.1].

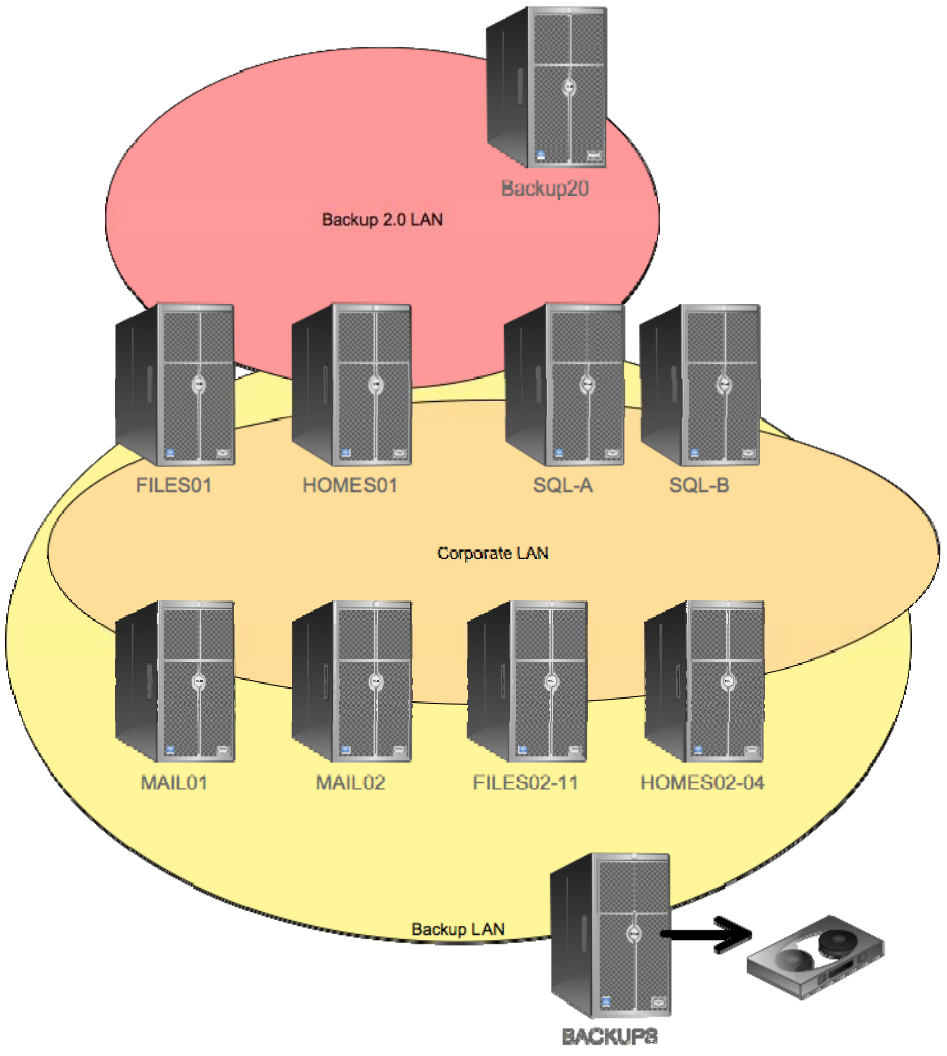


Figure 12.1: Adding a pilot Backup 2.0 network.

Things seemed to go pretty well at first. I was able to pull backups exactly like you have described in your book, and we did several tests of mounting those backups as live databases without having to do a restore. Kevin kept complaining that the system wasn't as stable or as reliable as backup tapes. I told him that if he would have let me put it on his backup server, we could have copied the backup data to tape every night or week or whatever, but he still didn't want to cooperate.

Then it happened. A sprinkler head broke in the data center one night (who installs water sprinklers in their data center?) and we lost both of the Exchange servers, the SQL Server, and one of the file servers—fortunately, one of the ones I was protecting with the new setup. We are probably lucky there wasn't an electrical fire! The SQL Server was actually clustered and we lost all the power supplies in those servers. So much for a cluster! We were able to cobble enough parts together to get one of the SQL Servers online, and one of the Exchange servers, but their disks were messed up. Kevin and I went to the boss to try to come up with a plan.

This happened on a Thursday night, so we were almost a week away from our last full backup. Kevin was planning to restore the Exchange databases to the one server he had running. My plan was to install VMware on the SQL Server box I had saved, as it was pretty hefty, and to bring SQL and the missing file server back online as virtual machines. The boss approved both plans, and we set out to do it.

So Kevin waits 3 hours for last weekend's full backup to come from off-site before he can get started. I was just installing ESX, which takes about a minute, and I had the virtual machines created in another 10 minutes or so. I used our Backup 2.0 solution to push the most recent disk images for the SQL Server and the file servers, and within about an hour total, they were both up and running with absolutely no data loss that we could detect. I was a hero!

Once Kevin finally got started, he realized that his Tuesday night incremental was bad—corrupted tape or something. That makes it tough to restore any of the other incremental because they all stack up. So we were basically back to Monday night—3 days of work, lost. Of course, it took him most of the day to realize this, so we wound up calling the boss at home and breaking the bad news. He Was Not Happy.

> Well, this wasn't the first time one of Kevin's plans had blown up in his face, and it wasn't the first time someone had had a better idea and been held back by Kevin. By the end of the week, Kevin was looking for a new job, and I had his job—and was busily setting up the rest of my backup network, moving my backup solution to the backup server, and putting Backup 2.0 into full use in our data center.

You have to love a story where the villain gets his just desserts (we'll assume they were just) and the hero wins the girl. Or rather, wins control of the data center.

Actually, this story points out a really important point about Backup 2.0: It doesn't have to immediately supplant your existing backup scheme. It can live quite happily beside it, coexisting for however long you need to complete the transition. You could choose to implement Backup 2.0 for your most critical servers, and continue backing them up via your old system, providing a "belt and suspenders" approach until you're comfortable and confident with Backup 2.0. Keep in mind that Backup 2.0 makes it a lot easier to practice full-server recovery, even enabling you to recover disk images to blank virtual machines in as little as a few minutes, in some cases. That means you can practice your recovery skills, tweak your Backup 2.0 installation, then decommission your old backup scheme.

The Cure for Crackberry Withdrawal

This one's from a reader who had responded to my pre-publication request for stories. He had a brief snippet included in Chapter 2 of this book, and he wrote back in to tell me how it's been going with their new Backup 2.0 solution.

> I wrote about the "Crackberry Withdrawal" that you included in Chapter 2 of your book on Backup 2.0. We've since implemented a Backup 2.0-style continuous data protection server for our Exchange Servers, and a couple of months ago, one of them failed again. Actually, the server got hit with some virus that corrupted some of the Exchange databases and was stopping the Blackberry Enterprise Server software from running. The executives were really upset—they can't handle being "off Blackberry" for more than a few minutes.
>
> Fortunately, that's all the time it took us. We figured out that the virus had hit during the evening prior, and we weren't worried about any emails that had been sent or received during that time because they might all contain infections. So we just rolled back the entire server to about 10 minutes before the virus had hit. It took maybe half an hour to do that, verify that the server was running, and get updated virus definitions on the server's antivirus software (it got hit about an hour before it was due to update those automatically). Back online, and everyone happy. Thank you, Backup 2.0!

This illustrates another creative and useful feature of Backup 2.0: the ability to recover a server *to any point in time.* You might not always *want* to recover to the most recent-possible disk images, as in this case where you would have just been recovering a virus. Because each change captured by the Backup 2.0 system is time-stamped, you can restore to literally any second in time that you like. If you're able to determine when something went wrong—as in this reader's case—you can recover to just *before* then, and you're good to go.

Exchange ~~Failure~~ Success

Also in Chapter 2 of this book, I republished a story from an Exchange administrator who had experienced a server failure—and hours of downtime that even Microsoft could barely figure out. He lost 2 days of mail, and his executives figured it cost the company about $50,000, so they wanted to put a better solution in place. I'll finish that story now:

> As you can imagine, 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.
>
> After evaluating several potential solutions of varying price (from $199 to $100K), I liked the Backup 2.0 solution we found. It's a block-based imaging recovery solution that captures the entire Exchange Server environment and supports recovery—anything from bare metal to an individual email message—in just a few clicks, simplifying the entire recovery process. What was totally cool was the Exchange "health checks," which made sure the data I was backing up was absolutely mountable. Before I made the recommendation to my boss, I wanted to be sure I was making the right choice and checked out a few of their customers, who said they were happy with the solution, too.
>
> In addition to capturing and validating your Exchange data, the tool employs a unique instant-replay capability that dramatically reduces volume recovery times from hours to minutes regardless of the data set size being recovered. After a rollback is initiated, the volume and storage groups are automatically and immediately mounted from the backup server, providing users with access to email during the recovery process. Apparently they call the feature Live Replay; all I know is it saved me from being Dead Admin.

You can read the whole story at http://appassure.ning.com/profiles/blogs/what-i-learned-on-my-worst-day?mkt_tok=3RkMMJWWfF9wsRow5%2FmYJoDpwmWGd5mht7VzDtPj1OY6hBssJJKtUg%3D%3D, if you're interested.

This highlights another advantage of the Backup 2.0 concept, which is that individual solution vendors are free to add really creative features. The "Live Replay" feature this reader wrote about is a good example: Rather than having to copy the server's entire disk image, this solution can mount the Exchange databases *directly from the backup server,* which eliminates a lot of file-copying time and gets you back online faster. Performance might be somewhat degraded, but you're *online,* and you can always handle the more time-consuming disk copy during a maintenance window when your users aren't so anxious to get into their email.

Virtually Online

This story illustrates a great use of not only Backup 2.0 but also the growing popularity of cloud storage and the use of the cloud as a recovery mechanism. This is definitely something that requires a clever vendor to be your business partner, but I think it's a great tale.

> Our company has only about 30 servers, including several SQL Servers and a couple of Exchange servers, but we use them to process almost $30 million in sales every year. So as you might imagine, these servers are very important to us—without them, we lose $80,000 a day or $8000 an hour. We're a small company, so it's difficult for us to justify the cost of a complete off-site recovery facility. Even with $80,000 a day at risk, those recovery facilities are expensive—and they can take a long time to bring online after a failure.
>
> We started looking at Backup 2.0 after I read the first few chapters of your book, and we worked with a couple of Value-Added Resellers (VARs) who have implemented their own services to supplement the Backup 2.0 solution that they resell. One of those services is a network appliance that sits on our network. Basically, we replicate our backup data to it, and it replicates that data off-site; it uses bandwidth throttling and does a lot of the replication at night, so it doesn't kill our Internet connection. I'll include a diagram of how we have this set up [see Figure 12.2].

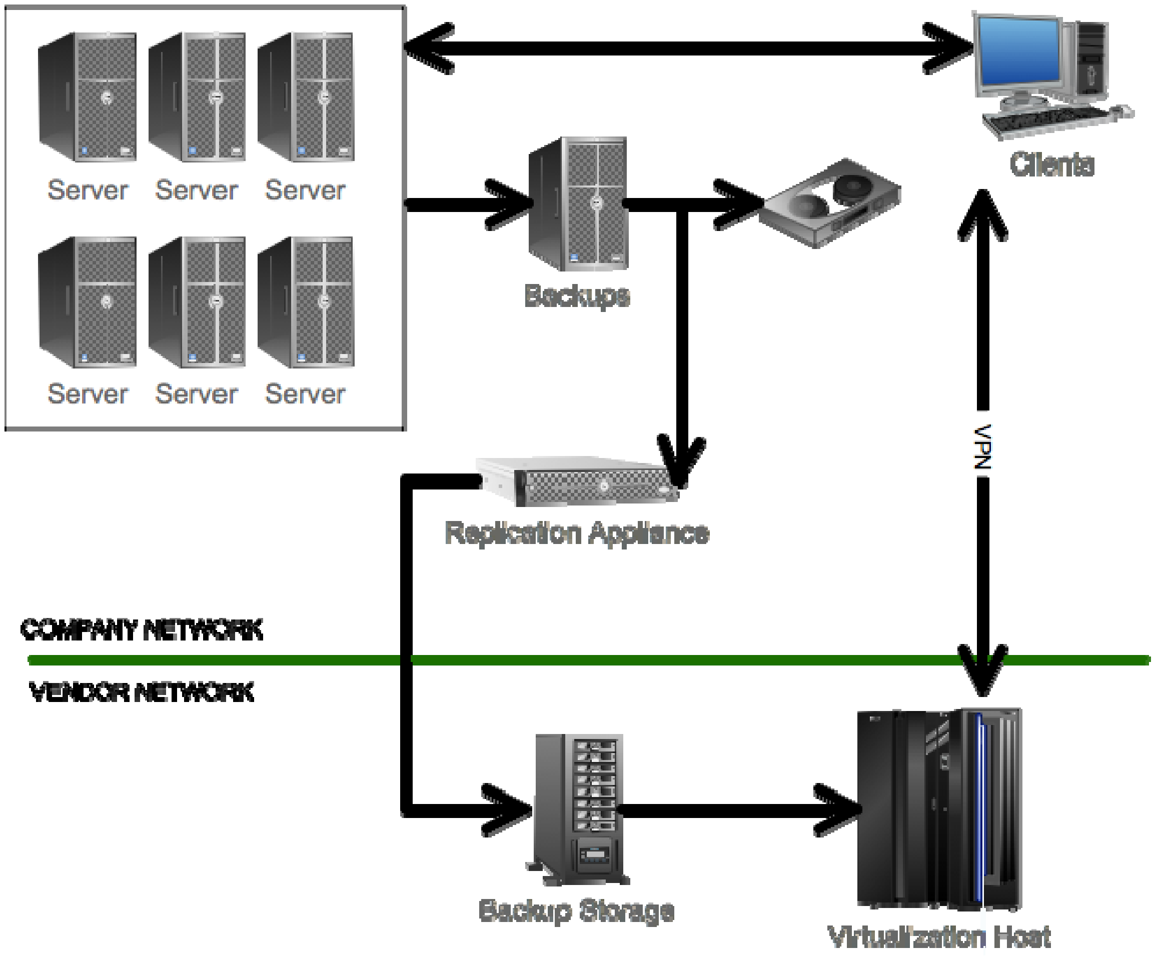


Figure 12.2: Off-site backup data replication.

Well, we finally put it all to the test last week. We deliberately shut down our entire data center—hit the big red "STOP" button that's just next to the door, something I've always wanted to do—and did a live test of our disaster recovery plan.

What happens is we call the vendor, and they initiate a "warm recovery" on their end. They spin up a bunch of virtual machines in their own virtualization infrastructure, then restore our most critical servers to those virtual machines. Then they begin spinning up other virtual machines for what we call our second-tier servers. They set up a Virtual Private Network (VPN) from their data center to ours, and our users access "our" servers from across that VPN. Because of the replication lag time, we were missing a few hours' worth of data, but it wasn't much. In a real situation, we might even be able to get more-recent files and so forth from the local backup copies, unless something really bad had happened to the data center.

It all took just a couple of hours before we were back online. Paying for "virtual space" is a lot cheaper than traditional off-site recovery facilities, and our plan worked perfectly for us. Management was very happy—we were glad the VAR was able to help us put together a plan like this.

It's also proven the model to us. Management has been considering opening a second office elsewhere in the country, and we now know that each of our offices could act as a "warm" recovery site for the other, simply by replicating backup traffic. Other than the WAN costs, we wouldn't have to invest much infrastructure in that, which is perfect for a smaller company like ours.

This is really a testament to the value of VARs, who can take a great piece of software—like a Backup 2.0 tool—and supplement it with services like this, to really create a complete *solution* that solves all of a company's business requirements effectively and economically. And, as this reader points out, you could do the same thing on your own if you already have more than one site. Consider Figure 12.3.

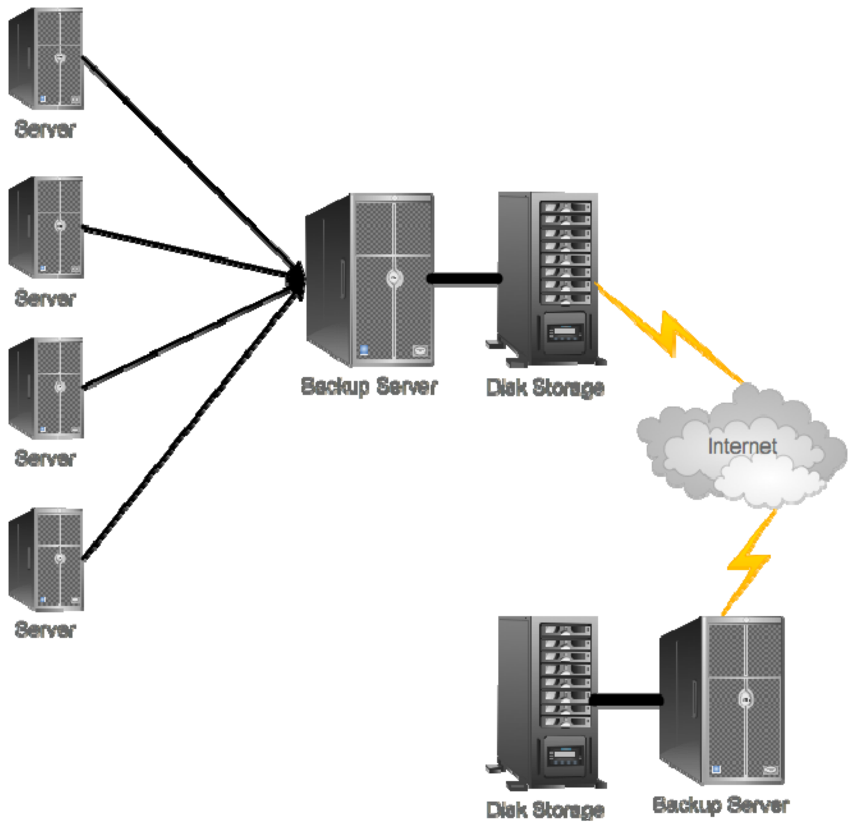


Figure 12.3: Creating your own "off-site" recovery.

By replicating backup traffic across your WAN (or the Internet), you can keep copies of your data safely off-site—without the hassle of shuffling tapes. If a time comes when you need to recover your complete data center, your data is safely in another physical location. Using Backup 2.0, you can restore those disk images to physical or virtual machines, bringing critical computers online quickly. Backup 2.0's continuous data protection techniques make it ideally-suited for this type of environment.

Wait, We Got That?

This is a fun story about that first, frantic week in a new environment—and what happens when the folks who preceded you were thinking clearly.

> So I start my new job, and I've been in the place maybe a week when both of our domain controllers fail. Of course, server recovery is the main part of my job description, so I'm supposed to be the superhero. This is a small environment, and the domain controllers were actually running in virtual machines and it was the virtual machine host that failed. Well, we get the thing back online—with 500 users yelling about how they couldn't do any work, of course—and we find that we still can't log on correctly. Whatever took the server offline had corrupted something in memory, probably because the domain was toast.
>
> So I go to the boss and tell him that we're going to have to rebuild the domain from scratch. "What about the backups?" he says. I explain that recovering an entire forest takes special tools, you have to call Microsoft Consulting Services, all that. That's when one of my new coworkers—fortunately, a bit more informed than I was—pulled me aside. "This isn't really a full-forest recovery," he said. "We just need to bring those two servers back from the point in time they were at. We don't use a normal backup solution, it can do this."
>
> Well, who knew. I was thinking backup tapes, but apparently the guy who had my job before me implemented a new server-based solution. Basically, I hit a dozen buttons or so and both domain controllers were back online. In like half an hour. That's when I started looking into Backup 2.0 and found your book. You're right—this is so much better than backup tapes I can't even tell you.

It's a nice feeling when you're able to make everything work, especially when your environment has the right tools for the job. In corresponding with this reader, I found that his biggest amazement came from the speed with which the recovery happened. The company had invested in pretty fast, dedicated Ethernet connections between their servers and the backup server, and that bandwidth made all the difference. He tells me now that he thinks he could actually pull off the same recovery a lot faster now that he knows what tools he has available and how to operate them. Interestingly, his first experience was done almost entirely without the product's documentation—he just got into the UI and started working. That's a sign of a great piece of software, isn't it?

Stop the Spindles! Or Whatever…

Working in technology for as many years as I have (and you probably feel the same way), it's tough to imagine how users envision it all working. Stories like this are a great reminder—and a good example of how Backup 2.0 can help with the little disasters, too.

> Here's a funny one you can use in your book if you want. I get a call last week from one of our users, who's yelling into the phone. "I just deleted a file by accident and it's gone—can you stop whatever is deleting it before it's too late?" He thought there was some kind of process or something that deleted files on a delay. Sadly, we all know this is not true. But I didn't tell him that.
>
> We have one of those Backup 2.0 solutions you talk about in your book, and it grabs changes as they happen. So I just popped it open, mounted the most recent point-in-time as a disk image, and copied the user's file to my workstation. I emailed it to him with a note that I was able to "pull it out before the file was permanently destroyed." He thinks I'm a hero.

This is probably representative of what *most* "disaster recovery" situations look like. To you and me, not a disaster; to the user who deleted the file, it's a tragedy in the making. We have to plan for the whole-server and entire-data center scenarios, but the fact is that most of us spend most of our "recovery" time getting back individual files here and there.

Enough Is Never Enough…Until It Is

I remember when I was an AS/400 operator, and we bought a new tape drive that used massively higher-capacity tapes. I was so excited that we could not only do the nightly incremental backups using a single rack of tapes, in most cases we could do them to just *one or two* tapes. Finally, I could kick back and let the backups run without having to remember to go change tape racks halfway through. Backups were a lot faster, too, which meant our night-shift users had to do without their AS/400 for a lot less time. Ah, the good old Backup 1.0 days. Sometimes, though, buying ever-better tape drives just isn't the right answer.

> When I first got to my company, we were using pretty decent tape drives. Full backups on the weekends required an operator on-site, though, because you'd have to change racks two or three times to get everything.
>
> About a year ago, we upgraded the drives, and you could do the whole run on a single rack. We'd load the rack on Friday, set the backups for "automatic," and enjoy the weekend. No more weekend unpaid overtime!
>
> Well, we just added a bunch of servers and that rack of tapes isn't enough. Of course, we've downsized a bit since a year ago, and now I'm the only one who's responsible for backups. So I'm expected to "float" half of my weekend—I can take off early on Friday, but I have to come in on Saturday afternoons to swap tape racks.
>
> To heck with that, I told my boss. I asked if we could upgrade to newer drives again, and he said no way—these ones are only a year old. Then I started looking at Backup 2.0 solutions, and digging into what some of them cost. I presented a plan to use a Backup 2.0 system instead, and to just stream the backup data to tape *during the week.* You know, when we're all here *anyway.* Well, it wasn't that expensive, so he got approval.
>
> Now we can recover right from disk-based storage. Tapes are just a backup of the backup, really—we have to send something offsite just in case, so that's what the tapes are for. And I get my whole weekend.

Again, a great illustration of re-thinking things. Of course, you could accomplish much the same thing with a disk-based Backup 1.0 system—but with Backup 2.0, of course, you're getting continuous data protection, point-in-time recovery, and so forth, so why not go that route while saving your weekend?

SQL Server Solution

Here's another follow-up from Chapter 2. I'm actually going to reprint the original story, because it's short and compelling:

> 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.

Crazy, right? It's a lot of work, but it's also an incredibly common scenario for SQL Server, especially in smaller and midsize environments. I heard back from this reader just as I was starting to put this chapter together. They've got Backup 2.0, and the news is good:

> I was able to sell the boss on a Backup 2.0 solution based on what you had written about SQL Server, but even I wasn't prepared for how much easier things are. We simply shut off all the native SQL Server backups, including the transaction logs. We still make a native full backup every week, just to keep the logs truncated and to have a static copy to send off-site, but we're backing up about 2000 instances of SQL Server—most of which are small, from development environments—continuously. We never have more than a minute or two of data at-risk. We don't have to monitor backup jobs, we don't have to copy backup files, and we don't even have to manually restore stuff. When it comes time to update the test environment, we just point the backup server at a destination and hit "go." We can provision the test environment to match any database, at any point in time.

Another creative use of Backup 2.0. Obviously, moving from 15 minutes at-risk to a minute or two is a benefit, not to mention a lot less manual monitoring and effort. But being able to quickly "restore" a database to a test environment is a big bonus, and being able to do so to a given point in time must open up a lot of really valuable test scenarios, not to mention lessen the overhead of managing all that backup data.

Hotfixes, Gone Cold

Patches and hotfixes can be just as much a cause for problems as user error, server failure, and other disasters. Here's a story that some of you may recognize, and a creative use of Backup 2.0 to solve the problem. I've changed the antivirus vendor name because the actual company isn't important—this could happen to anyone.

> We run about 100 servers, all of which run XYZ antivirus software. That company issued a patch or update recently, which we duly installed. Unfortunately, our servers started working not-so-well after that. A couple of them shut down and wouldn't restart, even. Not good.
>
> While one of my coworkers was on the phone with the company's tech support, and another one with Microsoft, I headed for the backup server. We have a backup solution that works like the Backup 2.0 kinds that you describe—it copies disk blocks and time stamps them. The problem is, I didn't want to restore the entire server; I just wanted to restore a few files. I looked at the patch we had applied and it only updated four files, so I went into the backup system and got the four old files out. They were slightly different from server to server—probably data files of some kind—so I just got them from each server's backups. The neat thing is that we can do this minute-by-minute, so I was able to get the files exactly as they were JUST before the patch was introduced.
>
> Put the files on their respective servers, rebooted, and all was well. Oh, and we're still with that same company for antivirus. Hopefully, it won't happen again, but at least now we don't have to explicitly make a backup before installing patches.

This is actually a scenario I don't recall writing about in this book. You know, the "Always Backup Your Server Before Installing Patches" notice that every hotfix or patch or service pack comes with. Who does this? Some companies do, I know, but easily just as many don't. And yes, we all should; it's just that backups *take so long.* Or they did, until Backup 2.0. Now, your servers are *already backed up*, so there's no need to perform a specific pre-patch backup. Just drop the patches in place, and if you don't like what happens, roll 'em back from your backup console.

De-Duplication Discovery

Another horror story from Chapter 2 involved a company that was backing up a *lot* of duplicated files, due in part to their use of DFS replicas. That story concluded:

> 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!

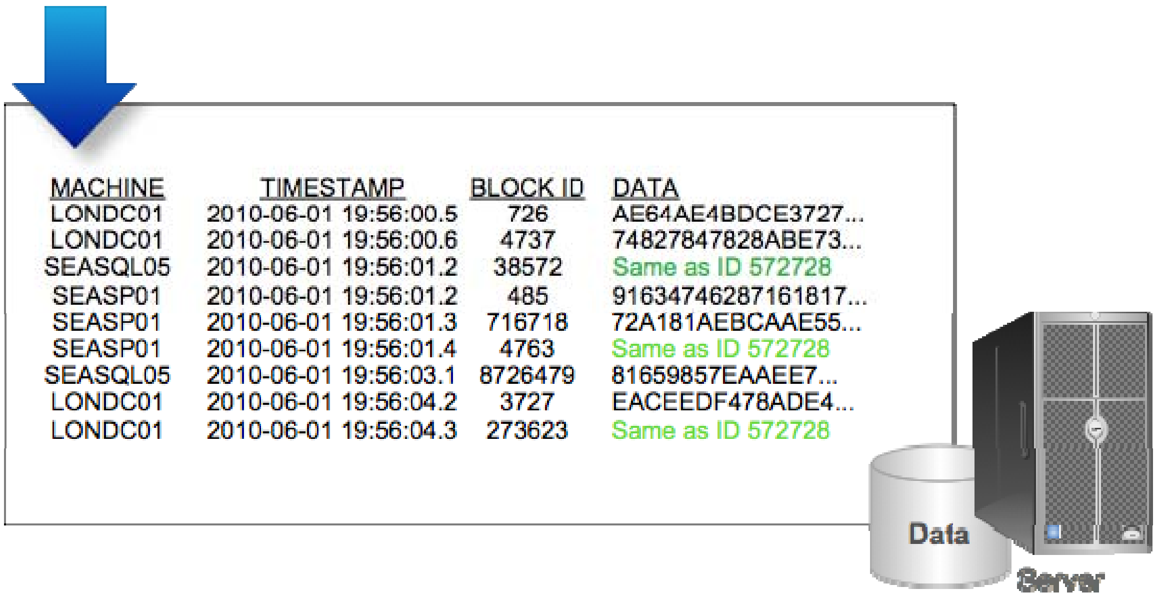
I found a colleague who had been in a similar situation. His company has begun a Backup 2.0 implementation, and they selected a solution specifically for its ability to compress and de-duplicate the backed-up data. I interviewed him to learn more, and here's what he said:

> We have a lot of duplicated data on our network, too, much of which is in user home folders. But you know, even if you don't have exactly-duplicated files, *everyone* probably has a lot of duplicate data at the disk block level. Continuous data protection solutions can operate at the disk block level, so we looked for one that de-duplicated at that level, too.
>
> I'll give you a specific example: Just across our dozen or so file servers, we have close to 24TB of data. With our old backup system, we could compress that, and we got about a 2:1 compression ratio, or about a 50% reduction in data size. That's still 12TB of data—an awful lot to back up. We have pretty new DLT tape drives, and each tape holds about 800GB, so I think we're using a couple of tapes to pull a full backup of all that.

> We had one of our developers write a little tool that analyzed duplication at the disk block level, and that's what really justified the Backup 2.0 expenditure: His tool (sorry, I can't share it) suggested that we could reduce our backup size by a total of 80% by using both compression and de-duplication. That's 24TB to 480GB. I think everyone's jaws hit the table. We could easily fit an entire full backup on one tape, and it also made it a lot more likely that we'd be able to afford the disk-based storage that Backup 2.0 solutions rely on.
>
> A couple of months later, and that's pretty much the level of reduction we're seeing. Right now, we've got five of the file servers on the new system, with a total of about 15TB of data, and we're seeing full backups occupy about 300GB of space. Of course, we really only do that once, then the solution just grabs changes from that point on.

De-duplication is available in a lot of ways, including in some Backup 1.0-style solutions. But having it built into a Backup 2.0 solution not only gives you the benefits of Backup 2.0 but also makes it a lot more practical to implement because you're making much more efficient use of the backup solution's storage.

I want to point out that you should specifically look for disk block-level de-duplication for maximum savings, rather than higher-level, file-based de-duplication. The latter can only save you space when entire files match up, whereas disk block-level de-duplication is a lot more likely to find a greater number of duplicates. Figure 12.4 illustrates what this kind of de-duplication will look like.



MACHINE	TIMESTAMP	BLOCK ID	DATA
LONDC01	2010-06-01 19:56:00.5	726	AE64AE4BDCE3727...
LONDC01	2010-06-01 19:56:00.6	4737	74827847828ABE73...
SEASQL05	2010-06-01 19:56:01.2	38572	Same as ID 572728
SEASP01	2010-06-01 19:56:01.2	485	91634746287161817...
SEASP01	2010-06-01 19:56:01.3	716718	72A181AEBCAAE55...
SEASP01	2010-06-01 19:56:01.4	4763	Same as ID 572728
SEASQL05	2010-06-01 19:56:03.1	8726479	81659857EAAEE7...
LONDC01	2010-06-01 19:56:04.2	3727	EACEEDF478ADE4...
LONDC01	2010-06-01 19:56:04.3	273623	Same as ID 572728

Figure 12.4: Replacing duplicate blocks with pointers to the first duplicate.

It's a Backup *and* an Archive!

In a previous chapter, I cautioned you against using your backup system as a permanent data archive. Backups should be used to store only as much data as you'd want to bring back into production use after a loss; they're not necessarily intended to store data for years and years. But I did have one reader who disagreed:

> I have to tell you that we've been very happy with our new continuous data protection solution, especially as it applies to our Exchange Server. In fact, despite your caution to not use the system as a permanent archive, we have found it to be very well-suited for just that.
>
> Our company policy requires us to keep about 3 years' worth of emails, and we used to leave that information in Exchange. Needless to say, our Exchange databases became pretty unmanageable. We've decided instead to just spend the money on disk space for our new backup solution, to let it keep a *lot* of historical data, and to trim down our production Exchange databases.
>
> Our backup solution's user interface actually has excellent search features, and it "understands" the format of the Exchange databases. So we're able to search archived messages right from within the backup solution, and we've already had a couple of times when we've had to access individual messages from a couple of years in the past. Because doing that doesn't touch the Exchange system at all, our production mail servers have become much more efficient. We're looking to create some kind of tiered storage hierarchy so that we can keep even longer periods of historical data, if need be, and we think our backup solution will accommodate us in that.
>
> Disk space is pretty cheap, even redundant server-based storage. A lot cheaper than it used to be, at least. So for us it's an easy investment to make.

Which just goes to show that you can't anticipate *everyone's* business priorities. I'll admit that a Backup 2.0-style solution *does* offer a compelling argument for secondary use as an archive, if you're willing to dedicate the disk space to it.

Virtual Nirvana

Finally, my last story is one that's probably all-too-familiar to those of you who have to answer calls from your users. I think it's a great example of how to really put Backup 2.0 to clever use, too.

> We use a Backup 2.0-style solution to back up all our virtual servers. Now, we do something a bit different than what some others might do. We put our backup agent inside the virtual machine, and we back up everything that way, but we also back up the virtual machine disk files themselves. I know, it's redundant, but there are cases where it comes in very handy.
>
> Perfect example: A few months back, we had a user call and tell us that the payroll system was offline. We figured out pretty quickly that it was due to some corrupted databases—this has happened before—but we were a little unsure of how far back we could roll the database, and we frankly weren't even sure which files we needed to restore. The system uses a proprietary database engine and the data is spread across what seems like a million files.
>
> Another admin was working this problem, and he was trying to figure out exactly which files to pull from the backup system. I found him digging through the files list, and asked what was up. He explained the situation, and I said, "You know what? Let's just blow away the whole virtual machine." He looked blank, then remembered that Backup 2.0 made that pretty easy.
>
> Now here's why we back up both the inside of the virtual machine as well as its disk images. Choice A is to restore the "inside" of the virtual machine, basically rolling it back as if it was a physical machine. That can take a while because you're getting the slower disk I/O of the virtual machine. In this case, time was a factor, so we rolled back the "outside" of the virtual machine—that is, its disk images. Much faster operation—less than 20 minutes, if I remember correctly.

A good example of how different forms of backups can meet different needs!

Conclusion

Well, there you have it. Hopefully these stories have given you some ideas about how Backup 2.0 can save you time, money, and anguish, and hopefully the 11 preceding chapters have helped you understand how Backup 2.0 can fit into specific niches of your organization.

In a few years' time, I'm betting we'll see more and more companies adopting a Backup 2.0-style continuous data protection solution. In fact, it won't be long before "Backup 2.0" is simply the standard that everyone uses. We're certainly seeing more and more vendors in this space, providing a variety of solutions, which indicates they're paying close attention to businesses' needs.

If nothing else, this book should have prepared you to be a smarter shopper so that when you start evaluating Backup 2.0-style solutions, you know what features to look for, what to prioritize, and what your business' full range of needs is likely to be.

Above all, I hope you've enjoyed reading this book as much as I've enjoyed writing it. This is the longest book I've written for Realtime Publishers, and I want to thank them for publishing it, and I want to thank the book's sponsor for not only making this project financially viable but also for giving me the freedom to really tell the Backup 2.0 story the way I wanted.

Good luck with your backups!