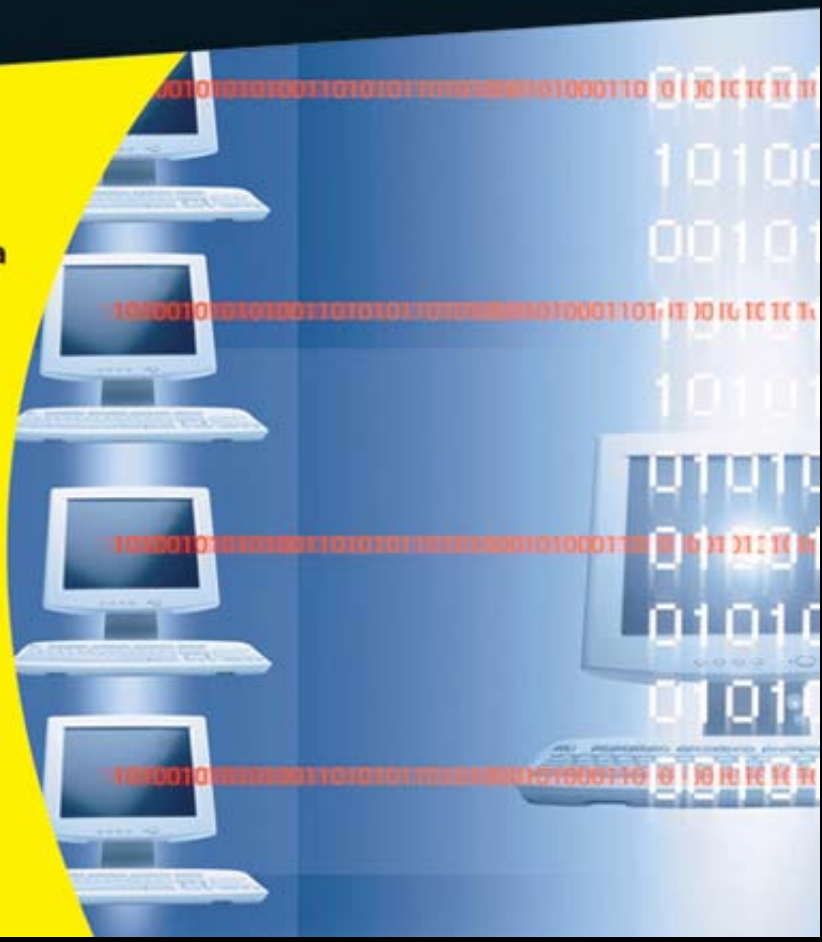
Making Everything Easier!™

2nd Edition

Storage Area Networks

FOR

DUMMIES®



Learn to:

- Implement new technologies such as data de-duplication, iSCSI, and Continuous Data Protection
- Design storage area networks that meet specific needs
- Maintain and troubleshoot SANs
- Develop SANs that will aid your company's disaster-recovery plan

Christopher Poelker
Alex Nikitin

Chapter 14

Continuous Data Protection

In This Chapter

- Defining CDP
- How CDP makes storage work like a database
- Best practices for storage when configuring CDP solutions
- The truth about *near* CDP and *true* CDP solutions
- Comparing CDP and snapshots
- Simplifying or eliminating backup with CDP

In this chapter, you find out how to use recent innovations in data protection to make protecting your data easier and more efficient. Continuous data protection enables you to shift your focus from backing up your data to recovering data faster and more efficiently and, perhaps, avoid the possibility of data loss all together.

Understanding What Continuous Data Protection Is

With CDP *(Continuous Data Protection),* your data is always protected, so you don't need to do backups. CDP can protect every write to a disk, so it's one of the only solutions that can protect your applications from any data loss and can rapidly recover your applications from disk. Because everything is continually protected to disk instead of tape, CDP greatly enhances your recovery point objective (RPO) and recovery time objective (RTO).

Continuous Data Protection (CDP) is one of the more interesting recent advances in SAN technologies for backing up data. With traditional backup solutions, the backup process is performed once or twice a day at the most, and therefore can only provide the ability to recover data to one or two specific points in time. A traditional backup process can take a long time to complete, because it has to move data from one place to another (either to tape or another disk). CDP is

different than backup, because there is no backup process, and there is no bulk data movement that must occur. CDP is just always on, like a service.

The amount of time it takes for the backup job to complete is called the *backup window.* During a normal backup job, data is read from production disks and then backed up to either a disk or tape target. Because it has to access production storage, a traditional backup can affect the performance of production applications while the process is running. This is why most backups are scheduled to run at night during the backup window when less business is being conducted, and the impact can be minimized. Because time is a critical factor in today's always-on data centers, the backup window and backups in general are becoming more of a problem for many organizations. Solving the backup problem by using CDP is what this chapter is about.

How CDP Makes Storage Work Like a Database

I know you didn't buy this book to become a database expert, but since CDP relies on some of the same principles that databases use to protect data efficiently and reliably, I need to cover some basic database concepts. If you have worked with a database application, you probably understand the concept of a *transaction log* and a database file. If not, don't worry, I try to explain how a database works in simple terms.

A database update can be made up of several read and write operations bundled into a *transaction.* A transaction is completed fully or not at all. For example, suppose you want to take some money out of an ATM, but you are on vacation in Europe and your bank is in the United States. The bank associated with the ATM you are using requires the money be transmitted to a European entity before dispensing in the ATM machine. This entire process is done as a single transaction; if any part of the transaction fails due to a network failure between where you are and your bank in the United States, the transaction is rolled back as though it never happened.

The ability of the databases involved in this complex transaction to recover from errors is a good thing for you and your money. The ability to roll data backward or forward to ensure that transactions are completed properly is called *atomicity*.

For database solutions to protect themselves from issues, database transactions must be *atomic,* that is, the data must be updated in such a way that if the update fails, the update to the data can be *rolled back*, or removed. If any part of a transaction fails, all other parts of the transaction must be rolled back. The CDP process also relies on this fundamental database principal to roll changes back to a point in time.

When writing data to disk, most relational databases use a discrete function called *transaction logging,* or *journaling writes.* Before data is written to the database, all writes are stored in a transaction log (journal) until the entire transaction is completed. The transaction log is like a temporary staging area where all information is held until everything checks out. Once everything is completed properly, the information in the staging area (transaction log) is applied to the database and stored. Because the log holds every update to the database, the transaction log file gets bigger over time.

Database administrators have control over the size of the transaction log files and when they are *committed* in the database. (A transaction is considered committed when it is written from the log file to the database file on disk.) The larger the size of the transaction log, the more data it can hold and the longer you can go back to recover any changes. After all the completed transactions in the transaction log are committed to the database, space is freed in the log for more transactions.

To protect against losing both the transaction logs and the database, best practices are to keep the transaction log areas on physical disks separate from the database itself. In addition, since writes to the transaction log are usually random in nature, they are usually stored on faster, higher-performing disks than the disks for the database itself.

A CDP and database best practice is to use RAID 1 or RAID 10 for the CDP journal or log, and RAID 5 for everything else. Databases can be recovered from the data in the logs if required, so this is why you keep the database and the log separate.

Now that you have a basic understanding about how databases work, you can apply that new knowledge to how CDP solutions work.

CDP data journaling

As mentioned, when data is written to a database, it always goes into the log, or journal, area first, until the entire transaction is complete. This way, if the transaction fails for any reason, the information can be rolled back without affecting the database. CDP solutions do the same thing for updates to disk drives. In a CDP solution, all updates to the disk are split off and written to both the primary LUN and the CDP journal area at the same time, as shown in Figure 14-1. This ability to multiplex writes to both the CDP journal and the production storage is the job of the write splitter agent. (A *multiplexed write operation* simply means writing to both places at the same time in one operation.)

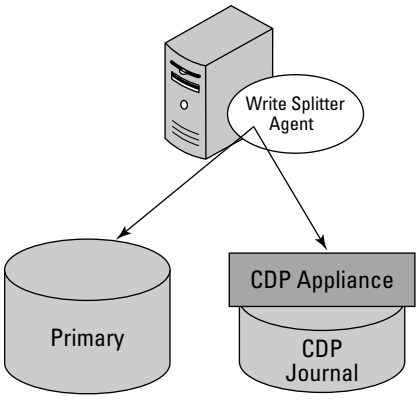


Figure 14-1: CDP write splitting to the journal area.

The CDP journal captures and holds all the updates in the sequence in which they were written. If the journal is large enough, it can store all updates that occur to the disk for days at a time. (You find out how to size the CDP journal later in this chapter.) Because all the updates are held in the journal in the same time sequence in which they occurred, you can "rewind the writes" from the journal back to the production disk to any point in time.

Splitting the writes

For CDP to do its thing, you need to capture and protect every write that goes into the SAN for a protected server, and store all the writes from that server in the CDP journal. The process of getting every write to the journal is called *write splitting*. The process of getting the writes to the journal can be performed in a number of different ways. You can use a host-based write splitter agent on the server itself, or use an appliance connected to the fabric switches to mirror all writes to the journal.

You can also buy more advanced switches that can perform the write split function in the fabric itself. (Cisco SANtap is an example.) Some storage arrays can copy data directly to a journal using firmware functions in the array (like Hitachi Shadow Image software in a Universal Storage Platform array) and by simply using a host-based logical volume manager to mirror the writes over the SAN or an iSCSI connection to the CDP appliance.

The write split process can occur in two ways:

- In-band, which means the write split happens in the primary data path. (The fabric appliance can be an in-band solution.)
- Out-of-band, which means the write split happens outside the primary data path.

Each approach has benefits and trade-offs.

CDP solutions that use host-based write splitters use a software-based host agent to split writes off to the journal, so they can recover only writes that were performed by that specific application or server. Host-based write splitting is considered an out-of-band approach. (Refer to Figure 14-1 for an example of a host-based write splitter.)

CDP solutions that use fabric-based write splitters at the SAN fabric level capture all writes by all applications attached to the SAN, and usually use functionality in the SAN switches themselves to split off and route all protected writes to the CDP journal (see Figure 14-2). Oddly enough, fabric-based splitting is also considered an out-of-band approach because the primary write is not intercepted; it is just copied by the SAN switches and routed to the CDP solution. Cisco's SANtap solution is an example of a fabric-based write splitter.

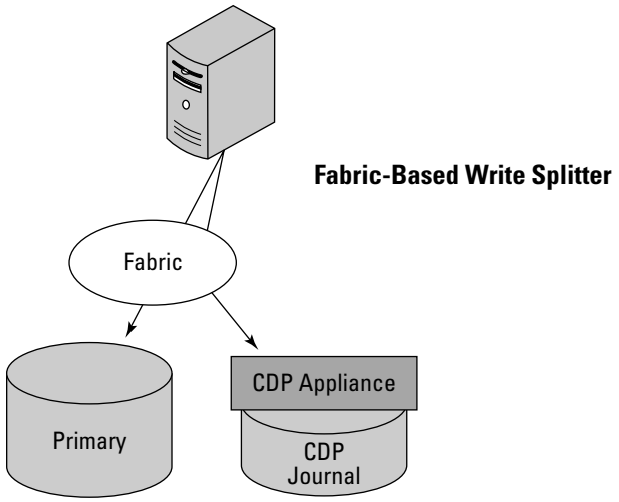


Figure 14-2: Fabric-based write splitting.

Appliance-based write splitting captures all writes going through the appliance and splits them off to the CDP appliance (see Figure 14-3). This solution removes the write-splitting function from the servers or the SAN switches. Using an in-band appliance means all the writes must go through the appliance before being stored in the primary array. The need to go through an appliance may cause concern for some because the appliance is in the primary data path. These appliances can usually be clustered to alleviate these concerns. Appliance-based splitting in the data path is considered an in-band solution. Using an in-band method may have some advantages though, such as the ability to simply mirror data to any storage, which would ease refreshing storage technologies and reduce downtime when migrating data!

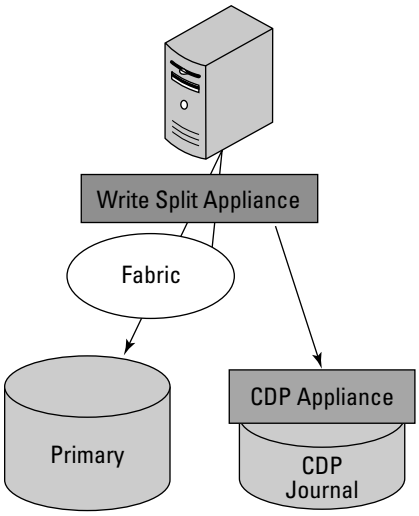


Figure 14-3: Appliance-based write splitting.

Using the storage array to split off writes allows the CDP solution to sit beside the array and out of the primary data path. Also, using array-based write splitting (see Figure 14-4) can be simpler and cheaper than having to buy proprietary write splitter blades for your SAN switches, and eliminates the load of splitting writes from the server. (A *blade* is a SAN switch component that is inserted into a slot in the switch chassis and performs a specific function.) Some storage arrays (from Hitachi, Sun, and HP) have the ability to connect to external storage, so you can use the internal array copy functions and hook up the CDP solution external to the array.

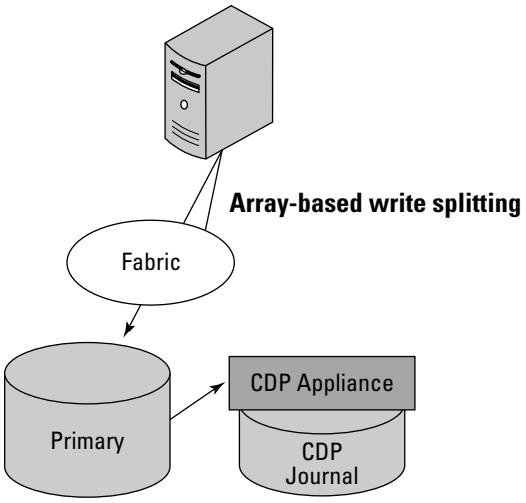


Figure 14-4: Storage array–based write splitting.

CDP makes everything different

One major advantage of CDP solutions versus traditional backup methods is the ability to create a consistency group across entire applications. (A *consistency group* is a group of LUNS that are recovered together at the same time.) Many applications have multiple tiers or components that need to be recovered to the same point in time, and the ability to create a consistency group across storage platforms and application servers can simplify the recovery of complex applications. CDP solutions usually are implemented as appliances that attach to both the SAN fabric and the network, so they can accept write splits of data from both SAN-attached servers and non-SAN-attached servers. Being able to capture writes from anywhere means you can create a CDP consistency group across all the servers that make up an entire multitiered application, as shown in Figure 14-5.

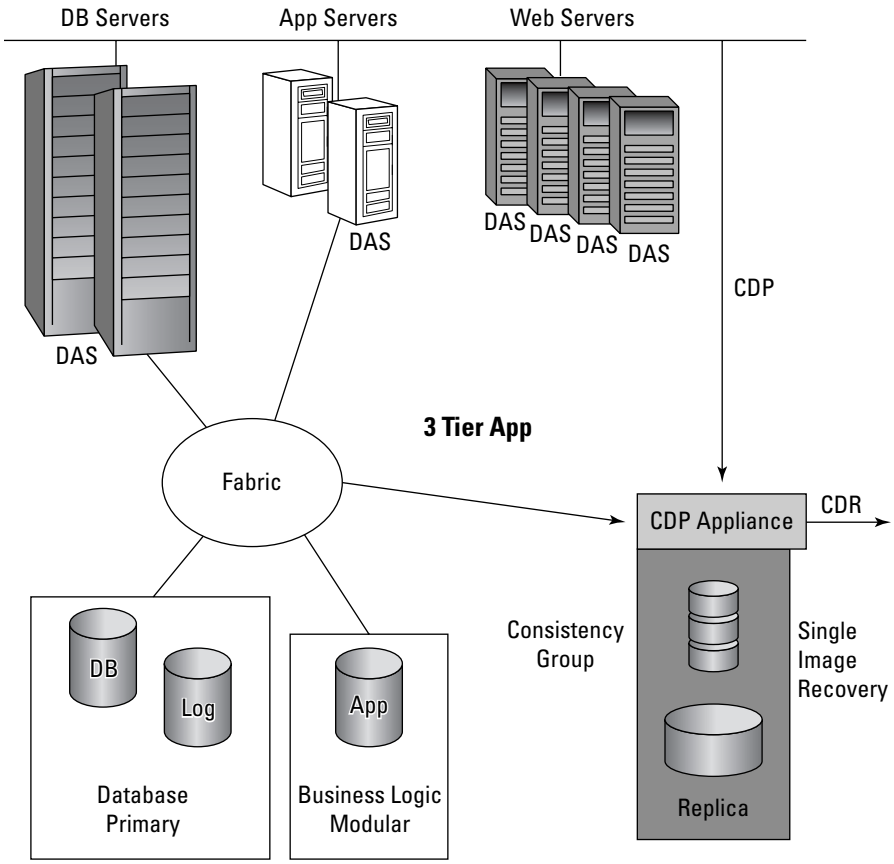


Figure 14-5: Cross-platform consistency grouping.

Figure 14-5 depicts a typical three-tier application where the application has multiple components, or tiers: a database tier, an application logic tier, and a Web server tier. Multi-tier applications typically can also use a tiered storage environment within the SAN, where the database gets its storage from fast and expensive enterprise storage arrays, and the application logic part of the application uses a less expensive modular array. The Web front end to the application is usually not even connected to the SAN, so it uses its own internal direct attached storage (DAS) in the server itself. The other servers use their own DAS storage to hold the servers' operating system.

If one of the components in the application fails, it's fairly easy to stop the entire application, recover the failed component using traditional methods (such as tape backup), and then start the application back up at a specific point in time that matches either the time of the backup or where the other components in the application were at the time of the failure.

For the application to recover from a disaster in which parts of the application may have been stopped at different points in time though (sometimes known as a *rolling disaster*), you need to bring up all the parts of the application at the same point in time. This can be a difficult undertaking and is one of the reasons why disaster recovery can be so hard for large complex applications that have many components or rely on data updates from other applications.

To recover these complex applications when using regular tape backup, each part must be recovered individually from a backup tape, and then someone must coordinate bringing up each component of the application in the proper order. Even if you use snapshot technology in the arrays or on the servers, you need to make sure that each component was snapshot at the exact same time for data consistency since the components of the application span multiple storage arrays and server platforms. This type of coordinated snapshot usually requires an external atomic clock mechanism.

Some companies have to go through complex and expensive *application interoperability* studies and *critical records analysis* of their data to create a coordinated recovery process for complex applications. Consultants can charge tons of money for this type of study. But CDP can make recovery of complex applications a simple process.

Using the CDP appliance as the external atomic clock that coordinates everything at the fabric and network levels, you can implement write splitters in the fabric and at the hosts to capture all the writes to a central journal (shown as the replica to the right in Figure 14-5) and create a consistency group across all the tiers of the application. When the CDP solution says go, it talks to the agents on each host, tells the agents to put everything into hot backup mode, flushes system memory and file system caches to disk, and then stops all writes.

Once all the data is synchronized to the CDP appliance, the CDP solution can take a snapshot of all the pieces of the application at the same time (shown as the snapshots labeled M, T, W, for Monday, Tuesday, and Wednesday, respectively). Because everything was snapped at the same time, everything can be recovered at the same time automatically. Simple. If the CDP solution also supports data replication, the snapshots can also be replicated to a disaster recovery site so that even in a major disaster, all the components of the application can be recovered to the same point in time at the disaster recovery site.

Sizing a CDP journal

A CDP journal is like a movie camera. When you press the record button on the movie camera, you are capturing everything that happens in front of the lens. After you press the stop button, you can rewind the tape and "go back in time" to review what you just recorded. The bigger the tape, the longer you can record, and the longer you can go back in time. So just as a database can recover itself by replaying the updates stored in the database journal over the last good backup, a good CDP solution can recover the production storage to a known good point in time by rolling back to just before the problem occurred.

The size of your CDP journal determines the length of time you can store any updates to the disk, and therefore how many hours or days you can roll back to should something go wrong. You need to know the following when sizing a CDP journal:

- **Time:** How far back you want to recover
- **Average write rate:** How much data is written in a given time
- **Number of servers to protect:** This is the division factor

Suppose you have 10 database servers that you would like to protect for 2 days to any point in time. You gathered the SAN statistics for these servers by collecting historical write information in the SAN switches, and you know the write rate across all 10 servers averages to about 100MB/second through the SAN.

You can create a formula from these three statistics to average the size of the journal on a per server basis based in gigabytes required per hour of protection. (Note: You need to multiply the average writes by 3.6 to convert megabytes per second to gigabytes per hour.)

The formula for average journal size per server is

```
(write rate average x 3.6) x (time in hours to journal) / (number of servers)
```

For example:

```
100MB/s x 3.6 = 360 GB/hour x 48 hours = 17280GB / 10 = 1728GB per server
```

To store two days' worth of writes across 10 servers that are writing data at 100MB/second for 2 days, you will need about 17.3TB worth of disk space for the journals. If you divide that number by the 10 servers, you get about 1.7TB per server for 2 days worth of protection to any point in time. That's a lot of disk space, which is why CDP journaling can be expensive for many applications and why CDP is best used for the most critical servers, where any data loss is a problem.

Using the preceding formula, you can see that storing only the last 2 hours in the journal requires 720GB for all 10 servers (100×3.6) $\times 2$) which comes to 72GB per server. Sizing the journal for the last 2 hours and using hourly snapshots allow you to recover to the last good write, and up to any hour after the first 2 hours from a snapshot.

Some CDP solutions (like from FalconStor) get around the storage requirement issue by using snapshots in conjunction with CDP journaling. These solutions simply journal the last few hours of data and then use snapshots on a periodic or hourly basis. If you do a good job monitoring your applications, you will probably notice within a few hours when something is wrong, missing, or broken, so you can recover from the journal or the snapshot.

If you want to use CDP journaling because you want a recovery point objective (RPO) of 0 (no data loss), you most likely want to get to the last good write, and not something from 2 days, 6 hours, and 23 minutes ago. If your CDP solution has the ability to take periodic snapshots in conjunction with journaling all writes, you can use the snapshots to recover from a consistent point really fast, or just use the journal to roll back to a recovery point right before the incident occurred.

Most snapshot solutions can handle about 255 snapshots per LUN, and since snapshots usually use storage space efficiently, you can use the snapshots with the CDP journal to go back in time to the last few hours, and also any hour within the last 255 hours (about 11 days). You can lengthen the time between snapshots to get about 30 days' worth of recovery points, which is an awesome solution to replace tape backup for both data recovery and longer term data archiving.

Best Practices for Storage When Configuring CDP Solutions

When using CDP technologies, where the data journal and any full copies of the data are stored can affect recovery time, and even whether the data can be recovered at all. CDP journals should be treated just like a database journal and stored separately from the primary data. If the solution you choose can leverage a different storage array than your primary array, that is even better.

In a CDP solution, the journal area is created on separate disks apart from the production data. All data writes or updates written to the primary disks are written simultaneously to the CDP journal. This is why CDP recovery is similar to database recovery. All data writes from your applications are written to the journal area away from your production data. A copy of the writes exists in the journal exactly as they were written to the production LUN. If a problem occurs or someone deletes data by mistake, the CDP journal can recover or roll back the production data to a known good point in time just before the delete occurred.

If the journal had been stored on the same disks as the production LUN and you lost the disks, you would also lose the journal. To make sure that the CDP journal can run at the same speed as the production disks, some CDP solutions use the same storage array as the production LUNs, as shown in Figure 14-6.

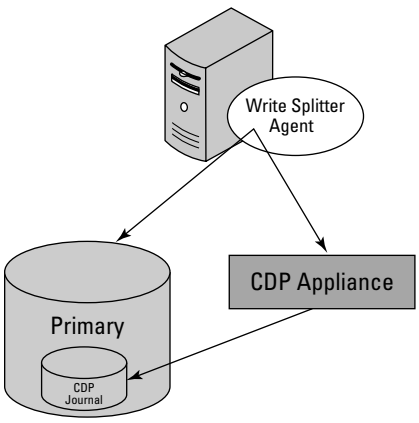


Figure 14-6: Placing the journal in the primary array.

With the journal on disks of the same speed as production disks, the CDP journal can keep up with production writes. The trade-off is that if the production array goes south, you can't recover anything. In these circumstances, having a tape backup of everything still makes a lot of sense.

If your CDP solution is fast enough to keep up with the primary writes, you should always place the journal external to the primary array, as shown in Figure 14-7.

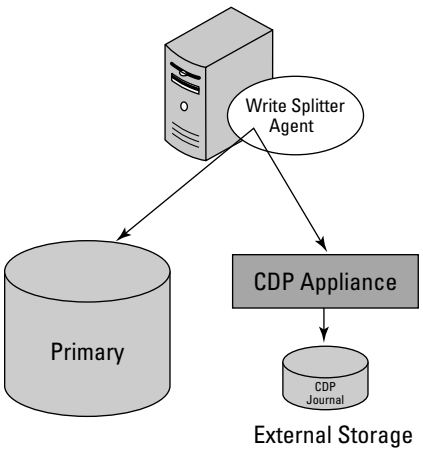


Figure 14-7: Placing the journal outside the primary array.

Some CDP solutions use the memory in the CDP appliance as a fast write cache so they can keep up with those expensive monolithic storage arrays from EMC, Hitachi, IBM, HP, Sun, and others. Being able to keep the journal on less expensive external storage also makes the CDP solution less expensive to buy and implement.

The Truth about Near CDP and True CDP Solutions

Continuous data protection is different than normal backups. In the following examples, you will see how different data protection solutions — traditional backups, snapshots, near CDP, and true CDP — recover data to a known good point in time, and how each method differs.

Example 1: Recovering a database with traditional backup

Most databases are self healing in most circumstances, in that they can use their transaction logs to either roll the database back to the last known good update or roll the log forward into the database to the last good stored, complete atomic transaction.

For example, consider a financial database that tracks all updates to the stock market. Millions of updates occur all day long to the database, and when the market fluctuates, updates can happen so fast that the database must perform over 100,000 I/O operations per second (100,000 IOPS). Consider what would happen if during one of those moments, a malicious hacker breaks in and in a split second causes the deletion of a massive amount of data, and the database crashes.

Assume that during that second, one of those 100,000 transactions was you sitting at your desk selling 10,000 shares of stock for $100 a share through your broker. (Nice chunk of change!) When the crash occurred, word immediately got out that a hacker broke into the stock exchange and deleted huge amounts of financial information. This news causes a panic, and the market plunges, causing the value of your stock to drop by 50 percent. (I know, this sounds more like a Tom Clancy novel, but bear with me; I'm getting to the point.) The way the database in this scenario was protected would have a profound effect on what happens to your critical transaction!

Simply using normal database capabilities and data protection methods, the database administrator could probably recover most of the data by doing a tape restore from the previous night's backup and applying any saved data in the logs to recover forward to the last good transaction that occurred right before the attack. The news would get out that all is well, and you would cross your fingers and hope that your transaction was one of those that was saved and recovered from the logs.

So that's a pretty good example for traditional backup with database recovery. But what if all the data in the database log files were also deleted by the hacker? In that case, the database administrator could recover only the data to whatever was contained in the most recent backup. Since the last backup was probably done last night, in this example your transaction of selling those 10,000 shares at $100 would be lost, and you would now be the proud owner of 10,000 shares of stock worth $50 instead of $100, and would be out $500,000! Not good. So what would happen if the database was protected by snapshot backups?

Example 2: Recovering a database from a snapshot

Since snapshots happen so fast and no copy operation really occurs (only the metadata [pointers] are captured in the snapshot; see Chapter 10 for more on snapshot backups), snapshots can be taken more frequently than traditional backups, which means you can protect your data more frequently. In the stock market scenario in the preceding section, if a snapshot was taken right after your transaction, your transaction data would not have been lost and you would be $500,000 richer. If the snapshot was taken just before the attack, however, you would be right back in the same boat as before. Snapshot solutions usually max out at about 255 snapshots per LUN. This is where CDP comes in. Continuous protection provides many more recovery points than just snapshots.

The benefits of CDP are most apparent when even the smallest amount of data loss could have a high financial impact on you or your company.

Example 3: Recovering a database using near CDP

Near CDP, unlike regular CDP, does not journal *every* write. The protection is not continuous but periodic at very short time intervals. In most cases, near CDP solutions create an up-to-the-second Point-in-Time copy (snapshot) of the data, and store those frequent snapshots in a separate pool area or journal. If something happens, near CDP solutions can recover the data to any *second* in time. For many applications, this is good enough. But for database applications that can perform hundreds or thousands of I/O operations or transactions per second, a near CDP solution cannot recover the transactional data that occurred within the second. If you were writing 100,000 transactions per second and using a near CDP solution, you would still lose 999,999 transactions. Let's hope the folks at the stock exchange used the right solution for the job.

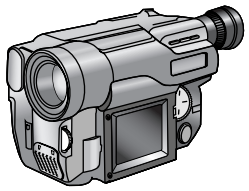
Example 4: Recovering a database using true CDP

True CDP solutions capture and record every write and split off those writes into a journal. Because *every* write is captured, *any* write can be recovered to *any* point in time. True CDP solutions can work at the sub-second level and provide a view for all the writes that occurred within the second. Recovery

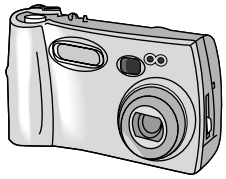
can be to any individual write operation within the second. If the stock market database was protected using a true CDP solution in the scenario, all the data would be completely protected, and there would be zero data loss. All transactions that occurred to the exact point in time right before the hacker whacked the data would be recoverable, even if the entire database and all the transaction logs were lost.

CDP versus Snapshots

Where CDP is like a movie camera, capturing everything that happens at all times, snapshots are like pictures, where information is captured at a point in time (see Figure 14-8). The concept of snapshots and near CDP versus true CDP has huge implications for data recovery for critical databases. If you delete a file on a disk by mistake, you can easily use a snapshot to recover back to a point in time before the deletion occurred. Because near CDP can take a snapshot every second, you can recover to the exact second before the deletion, which is pretty darn good. But if you need to recover to the level of a single transaction (such as your stock purchase) for an application writing at 100,000 transactions per second, you need true CDP.



CDP: Rewind to *any* point in time



Snapshot: Select *a* point in time

Figure 14-8: CDP versus near CDP and snapshots.

Snapshots can sometimes have an advantage over CDP solutions that do not have the ability to integrate with the applications they are protecting. Most snapshot solutions come with an intelligent application agent or script that runs on the server they are protecting to record the snapshot event. The agents can force the application into a *hot backup mode* (see the chapter on backup for more on this), which flushes any in-memory or cached transactions to disk and pauses any new writes to the disk while the snapshot process takes place.

Because the snapshot takes place with the full knowledge and cooperation of the application, all the data in the snapshot is in a consistent state. There is no need for file system checks or recovery processes when using the snap-

shot to recover. Some CDP solutions also have this ability to work with the application (check with your vendor), but others simply track writes only at the disk level, and because they are not "application integrated," recovery from the CDP journal may require an application or file system consistency check prior to being able to bring the application back online. So in some instances, snapshot recovery can be faster than CDP recovery, but the trade-off is data loss. Some CDP solutions get around the consistency issue by using a generic marker that adds an entry into the CDP solution through an agent or a script whenever the database does a checkpoint.

Using CDP to Eliminate Backups

Think about this: If every write from an application is being captured and protected in a CDP journal, isn't that much better than running a backup job to tape every night? The answer is absolutely yes! But to use CDP as a replacement for backup, you need to consider the following:

- Backup implies a full copy of your data, so you need to be sure your CDP solution can also create and access a complete replica, or R1, copy someplace else.
- Tape backups are also sometimes used to ship data off-site for disaster recovery. If you want to replace tape all together, you need to be sure that your CDP solution can also efficiently replicate your data to another site. If the solution can combine CDP for recovery and data de-duplication to reduce storage and WAN costs, then you have nirvana.

Most CDP solutions available today have the capability to replicate data to a disaster recovery location. And as the technology matures, vendors are realizing that although more storage may be required, having a full copy available locally can be a good thing from a recovery standpoint.

Even if you are protecting every write of a specific LUN in a journal, the journal would never usually contain a copy of every sector of a disk. For example, suppose that something happens to the first disk sector, known as sector 0. It would look like someone erased the disk, even though the disk would be missing only one sector. Sector 0 is usually touched only when the disk is formatted, so it would never have been updated and stored in a journal. Even though only a single sector would be missing, the journal would be useless, and you would need to recover the disk from a full backup, which would probably be on tape. When CDP needs to recover a full disk, it relies on either the primary disk or a replica (R1) copy for the data not stored in the journal.

If the data is not available in the primary disk due to a disk failure or other error, recovery is not possible. This is why having a full copy stored separate from the primary storage array makes a lot of sense; that copy should be local, if possible, so the WAN network would not be required for recovery.

Five things to know about CDP

Having a CDP journal is nice, but it's not a full backup. If you lost any data on the production LUNs that was not captured in the journal, you would not be able to recover. With CDP solutions, you need another complete copy of your data somewhere.

To keep up with all the writes, the disks used for the CDP journal usually need to be just as fast as your production LUNs. Some CDP solutions can bypass this requirement by using memory caches or other proprietary innovations. Make sure you know how your CDP vendor accomplishes this task.

Some CDP solutions can only journal writes and do not keep a full copy of data locally, in what is known as a replica, or R1, copy. Not having a full local *copy* available to recover data means the CDP solution cannot be used as a real backup.

Some CDP solutions integrate with your applications and some do not. If the solution does integrate with the application, the application can "see" the journal and use it for valid recovery points. If the solution is not integrated, the best you will get is crash recoverability from the journal.

True CDP solutions enables zero data loss, but the journals can be expensive and take up a lot of disk space, so CDP should be used on top-tier mission-critical applications that require a very stringent service level. Most other applications would be fine with backup to a disk-based or tape-based recovery solution using a VTL (Virtual Tape Library) or snapshots. If the CDP solution can journal and also do snapshots, then you get the best of both worlds.

In Figure 14-9, you see that a bad sector in the primary LUN is causing the application to fail. Since there is no copy of that bad sector in the journal, the CDP solution cannot be used to recover the data.

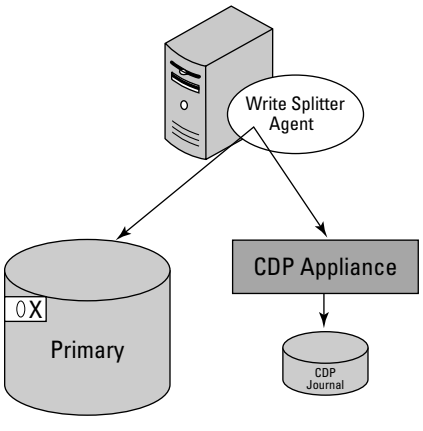


Figure 14-9: CDP recovery is not possible because there is no copy of sector 0.

In Figure 14-10, the same bad sector is causing an issue, but now you can recover the primary LUN from the copy of the bad sector located in the R1 copy. Because the CDP solution is not only journaling every write, but also mirroring all the data to a local R1 copy, the solution can be used as a full replacement for backup. If any data is deleted, or even if the primary disk goes completely bad, the data is still available for recovery.

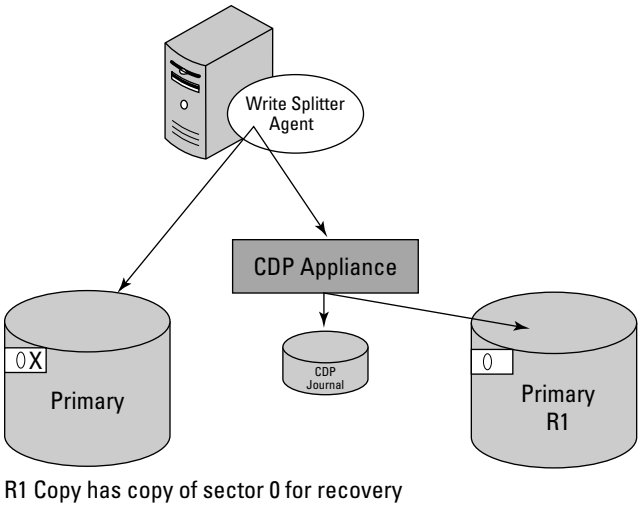


Figure 14-10: CDP recovery is possible because there is a full copy available.

This solution requires more disk space, but most companies can afford a single copy to protect mission-critical data. In fact, since the CDP solution also protects from corruption and deletion, the company can probably reduce copies that are being stored as business continuance volumes (BCVs) and even provide more recovery points at a cheaper price by using less expensive storage to hold the copy on a less expensive storage array. (A BCV is a storage-array-based full copy.)

Using CDP to Simplify Recovery and Reduce Costs

If the CDP solution can also replicate data, then it can even replace array-based data replication or shipping tape offsite, simplifying the datacenter even more.

CDP solutions that also replicate data can be a boon to many IT administrators because you can use the same solution not only for backup but also for

disaster recovery. Since every write is protected in the journal, every write can also be replicated to an off-site location for recovery. This capability can provide complete disaster tolerance for mission-critical applications and enable a more cost-effective solution for disaster recovery compared to traditional methods.

You can combine CDP (Continuous Data Protection) and CDR (Continuous Data Replication) to provide a complete solution for both backup and disaster recovery. What a paradigm shift for most companies! Most companies are still using tape-based backup, or perhaps have moved to a Virtual Tape Library (VTL), which uses disks instead of tapes for recovery. Tape backup is still pretty slow. A VTL makes disks look like tapes to the backup software, and makes backup and recovery faster, but a VTL still requires backup software to move the data from the production application over the LAN or over the SAN to the VTL.

Since backup usually runs only once a night, it provides only a single recovery point per day. Even VTL requires a fairly long recovery process before an application can be brought back online. Also, the data still needs to get off-site for disaster recovery, so someone will still need to create a tape to send off-site.

With the combination of CDP solutions that provide an R1 copy and CDR, no tapes are required for backup. All data is continually protected, so recovery can be to any point rather than just last night. With CDR, all the data is also continually replicated to the DR site, so no tapes are required to ship data off-site.

Continuous Data Replication (CDR) implies continuous, which is similar to array-based synchronous replication, but CDR, just like array-based asynchronous replication, can also be periodic. Data is simply stored in the journal and periodically updated to the DR site based on the policy you create. Some CDR solutions don't have the same stringent data fidelity capabilities as array-based synchronous replication, but they are also much cheaper to implement. Hey, there are always trade-offs!

One of the main benefits of using a CDP solution with CDR is the ability to use unlike storage at the remote site for recovery, as shown in Figure 14-11. Using array-based replication for disaster recovery, you usually have to buy the same expensive storage that you use in production. Because the intelligence for CDR replication can be appliance based and reside outside the production storage arrays on the appliance, you don't need to buy a similar array at the DR site. You can use the expensive and highly reliable storage at the production site, and less-expensive storage at the DR location. Some CDP solutions, such as the one from FalconStor, also include storage provisioning, data de-duplication, encryption, and virtualization so you can even use less components for DR, and the CDP solution can be used as the primary storage for the remote site.

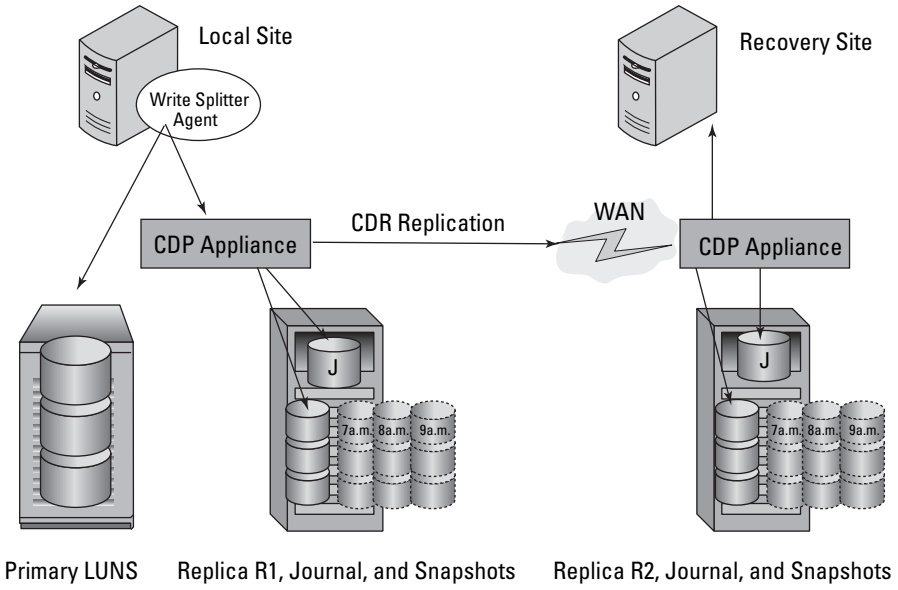


Figure 14-11: CDP can use low-cost storage at DR site for recovery.

In Figure 14-11, the local production site could use an expensive monolithic storage array, and the disaster recovery site could use less expensive SATA storage. By moving the data replication intelligence out of the storage array and into a fabric-level CDP appliance, you can eliminate all the licenses for replication solutions, such as SRDF, Universal Replicator, TrueCopy, PPRC, MirrorView, and any other array-based replication solutions. CDP can make storage a commodity. Although your storage vendor may not be pleased, you can save huge amounts of money by making backup and DR much more efficient.