The Best Backup Method for Mission-critical Applications — HPE Shadowbase Business Continuity

A Gravic, Inc. White Paper

Executive Summary

The need to back up data has existed for as long as data itself. Sometimes the backup is needed for historical purposes, for example, to preserve a snapshot of the information in time. In other cases, it is used to maintain an accurate and up-to-date copy of the information if the primary copy is lost or corrupted.

Magnetic tape is the oldest backup medium still in use. It was introduced in 1951, but tape sales began to fall with the introduction of high-speed and high-capacity hard disks, DVD's, CD's, and other innovations such as cloud storage. However, utilizing magnetic tape is on the rise again. Why? With so much big data created by mobile devices and IoT sensors, there is a growing need for an economical and efficient way to back up this data. Many companies are returning to tape to fill this need.[1]

Despite advances to these mediums, traditional backup and restore methodologies using tape, virtual tape, or other technologies still suffer from numerous inefficiencies that must be overcome to allow backups and restores to function in the new *big data* environments. This paper will discuss advances to address these issues and will primarily focus on backing up and restoring *transactional* mission-critical databases, since these databases support most companies and organizations' critical applications.

Note, we define a *mission-critical application* as one that requires constant (or near constant) uptime, as well as proper business continuity services to recover quickly from failure. Losing a mission-critical application will put the company (and users) at signification financial, regulatory, and/or physical risk, possibly resulting in failure of the business or loss of life or limb[2].

Unfortunately, this means that traditional backup and restore, magnetic tape, along with similar methodologies fail to provide acceptable RPO(s) and RTO(s)[3] along with an easily recoverable yet real-time backup. Clearly, *a better backup method is needed*. Preferably one that supports near-instant RPO(s) and RTO(s), an automated recovery method, low-latency data collection and replication from the active to the backup database, in near real-time, such as what HPE Shadowbase[4] software provides.

[1] For more information on the rise and fall and rise again of tape, please see *Magnetic Tape Makes a Comeback* published in the *Availability Digest*.
[2] For more information, see *Choosing a Business Continuity Solution to Match Your Business Availability Requirements*.
[3] Recovery Point Objective (RPO) is the maximum acceptable *amount of data loss* from an outage and
Recovery Time Objective (RTO) is the maximum acceptable *time to recovery* after an outage. For more information and an explanatory graph, please see *A Modern Look at Reliability, Availability, and Scalability*.
[4] For more information on HPE Shadowbase software, please see: *HPE Shadowbase Business Continuity*.

Table of Contents

Table of Figures

The Best Backup Method for Mission-critical Applications Using HPE Shadowbase Business Continuity

Physical Tape, Virtual Tape, and the Backup Problem

Magnetic tape, unfortunately, has its disadvantages. Physical tapes are bulky. Handling large numbers of them is a cumbersome and time-consuming process, for example, shipping tapes offsite or retrieving them from storage. Automated tape libraries are expensive and consume a large amount of space.

Tape is primarily a streaming medium, and it is relatively slow to access an arbitrary position to write or read the data stored on it. Inserting additional information often has to be appended to the end of the file (EOF) location instead of where other related data may be stored. Recovering information is usually a serial process, gathering the information in time-written order to recreate the latest committed (fully backed-up) value.

To solve the most problematic of these issues, tape was virtualized to allow disks and other storage media to archive the information, thereby allowing automatic processing for recording or retrieving the information with high-speed supporting networks to more easily transfer the information offsite or onsite. Despite these advances, traditional backup and restore methodologies using tape, virtual tape, or other technologies still suffer from numerous inefficiencies that must be overcome to allow backups and restores to function in the new *big data* environments. This paper will discuss advances to address these issues.

Current systems are generating such a large volume of data that backing up this data can easily overwhelm even the fastest virtual tape methods. The problem compounds itself if the body of data grows or quickly changes (big data volumes), and/or the database is constantly and actively being accessed to provide a critical service. We call these mission-critical databases and mission-critical services. In this paper, we primarily focus on backing up and restoring *transactional* mission-critical databases, since these databases support most companies and organizations' critical applications.

Most mission-critical databases cannot be taken offline, even briefly; therefore, enterprises must create backups of actively updated databases. Unfortunately, since transaction processing is active while the backup occurs, some of the data changes that are being backed up may abort and subsequently be undone (or backed out), which means the backup has "dirty" data in it. Additionally, as the database is being backed up, the data that was previously backed up is being changed, causing an inconsistent backup.

Fortunately, methods have evolved over time to not only back up the database, but to also capture the subsequent change data that has occurred since the backup started (or completed) so that the inconsistent and stale copy can be made consistent and brought current when retrieved and restored. HPE Shadowbase software solutions utilize such a method.

The key question is: *How is an online backup process accomplished, and how can it be improved and made more efficient?* Doing so would lead to faster backup and recovery methods, use less storage, and provide more consistent and current information when the backup copy is maintained and eventually restored.

Online Backup of an Active Database

A method to back up an active ("online") database is needed to ensure that the backup is *current*, *consistent*, and *complete*:

- **Current** means that the backup is up-to-date and not stale. A *snapshot* of the data means that all of the data that was backed up is only kept current up to a specific point in time.
- **Consistent** means that the backup is accurate and correct, for example, referential integrity is preserved and the so-called *dirty* data is removed (and returned to its last committed state).
- **Complete** means that the backup represents the entire database (or a specific/important subset of the data).

Additionally, the backup should not consume more resources (such as disk or other persistent storage) than is needed to reconstruct the database – either to a point-in-time or to the current state. Later in this paper, we will discuss such a method utilizing HPE Shadowbase solutions.

The Traditional Backup Method

The *Traditional Backup Method* is a common practice including periodic database backups onto a medium such as magnetic tape, virtual tape, cloud infrastructure, solid-state storage, or other persistent storage as shown below in Figure 1 (1). Throughout this paper, the use of the phrase *tape* for the backup copy medium is meant to include all of these storage medium locations and technologies and is not meant to limit the reference to just classic electronic tape technologies.[5] The use of the word *tape* or phrase *backup medium* implies a persistent storage device.

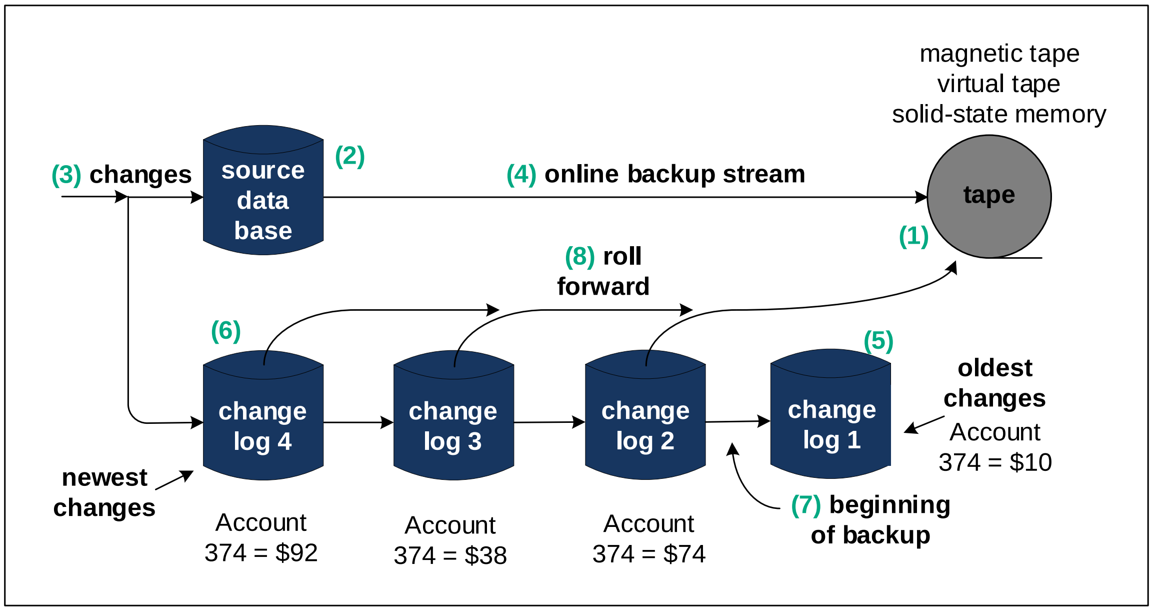

Figure 1 – The Traditional Backup Method

As shown in Figure 1, a backup is taken of a source database (2) while it is actively supporting transaction processing (3). Thus, the source database is changing as the backup takes place. This is known as an *online backup stream* (4).

The problem with an online backup is that it takes time to complete, and changes are occurring to the database during this time. Data written to the backup could be changing, and if the transaction aborts, the changes will be undone. Data written early in the backup phase is missing subsequent changes, but data written later in the backup contains more of the application's changes. This issue is not only true for the data records or rows themselves, but also for the database structure as the blocks used to hold the data and indices are also changing over time. Therefore, the data in the backup is inconsistent. The traditional method to resolve this issue is to capture *all* changes made to the database while the backup occurs, and eventually to replay them over a subsequently restored copy of the database to "roll" it forward to make it consistent and current.

More specifically, in order to restore a consistent (e.g., from a relational perspective, logically complete and usable to applications) database on a target system, the changes that are occurring during and following the backup must be written to a persistent change log such as an audit trail, a redo log, a journal, or equivalent data structure. In Figure 1, the oldest changes were written to change log 1 (5) and the newest changes to change log 4 (6).

The restore process then typically involves marking the persistent change log via various methods to note the time or relative position in the change log at which the backup began (7). The database is restored onto the target system by loading the backup copy onto it, and the pertinent change logs are rolled forward (8) to apply the changes that occurred after the backup started in order to make the target database *current*, *consistent*, and *complete*.

In Figure 1, the pertinent change logs are numbers 2, 3, and 4. (Change log 1 can be ignored for backup purposes since it was created before the backup began, and its changes are already reflected in the source

[5] The recent advances in tape density, writing and reading speeds, and the longevity of tape media over other storage technologies has reinvigorated the use of tried-and-true physical tape for saving copies of information for (very) long periods of time.

database and were captured by the backup operation at the time the backup began.) Therefore, in Figure 1, once the backup copy has been loaded onto the target database, the changes in change logs 2, 3, and 4 must be applied to the target database to bring it current and to a consistent state. It at least must be brought current to the time that the backup operation ended, since additional changes were likely made to the source database after the backup ended.

A problem with this technique is that several change logs may be required to hold the changes that occurred during the backup. For a very active source application with many changes occurring per second, there may be many such change logs required to hold all of the changes that occurred during the backup. These change logs all must be saved and made available (typically very quickly) if a restore sequence is needed.

For instance, as shown in Figure 1, Account 374 initially is backed up with an account value of $10. This change was made in log file 1, which occurred before the backup began. Account 374 subsequently is updated by the application to $74, then $38, and finally to $92; this sequence is reflected in the log files. These values are applied to Account 374 as the roll forward takes place. More specifically, the restore writes the initial value of account 374 from when the original backup occurred ($10). The log files then replay in succession, starting with log file 2, then log file 3, then log file 4 as shown in Figure 1. Unfortunately, the old values for this account replay temporarily during this restore sequence before ultimately ending at the correct account value of $92.

Besides being a lengthy process, which also requires a lot of storage for the log files, any access to the database during this time experiences old and inconsistent information while the replay of the data occurs. If the original database fails, denying users access to this information during this time will prolong an outage.

Furthermore, as shown below in Figure 2, many of the changes that occur during the backup operation already may have been captured by the backup if they occurred *after* the backup operation started, but *before* those particular data objects (or part of the database) were copied to the backup medium. Thus, these changes are a duplicate of data *that already was backed up*. Worse, there could be a series of changes to the same data that occurred after the backup began, but before that data was subsequently backed up, and rolling forward through those changes will actually cause the restored data to reflect older (and inconsistent) values while it is being rolled forward, as shown in Figure 2.

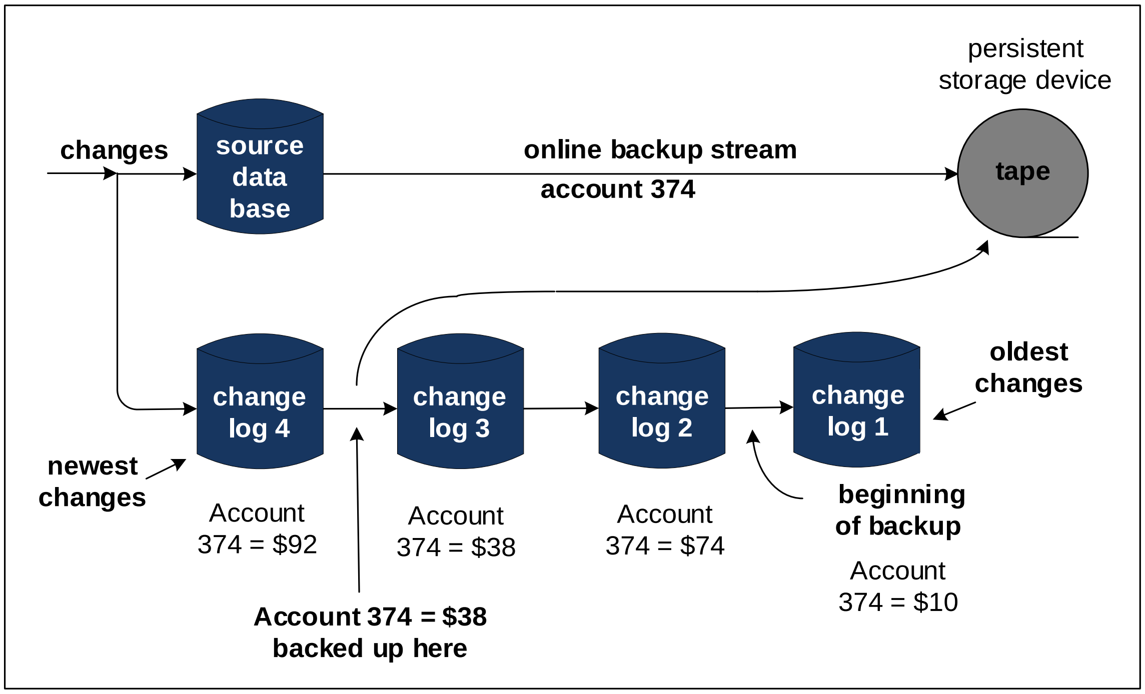

Figure 2 – Backing Up Duplicate Data

In Figure 2, account 374 starts off at $10 (after change log 1, when the backup starts), is updated to $74 in change log 2, then $38 in change log 3, and finally to $92 in change log 4, containing the newest change. However, the account is not backed up until change log 3, where the value is $38, as represented by the change captured in log file 3. Using this method of restore and roll forward, Account 374 is initially restored from the backup to $38, but then is updated to old account values while all of the log files are processed and the changes are rolled forward ($74 in log file 2, then $38 in log file 3, then $92 in log file 4).

Consequently, restoring a backup requires rolling forward through several change logs, which may take a great deal of time and consume a great deal of storage medium resources for all of the change log files. Furthermore, rolling forward through all of the changes that occurred during the backup makes the restored data out-of-date and inconsistent until the final set of changes are replayed from the log file(s).

Additionally, during this process, the source database is still being updated; these changes must be logged and rolled forward to update the restored backup to a current and consistent state to when the backup operation ended. All of this processing takes a considerable amount of time to accomplish.

The Better Backup Method

The *Better Backup Method* is shown in Figure 3. It is similar to the Traditional Backup Method shown in Figure 1 in that the contents of the source database (2) are written to a backup medium (1).

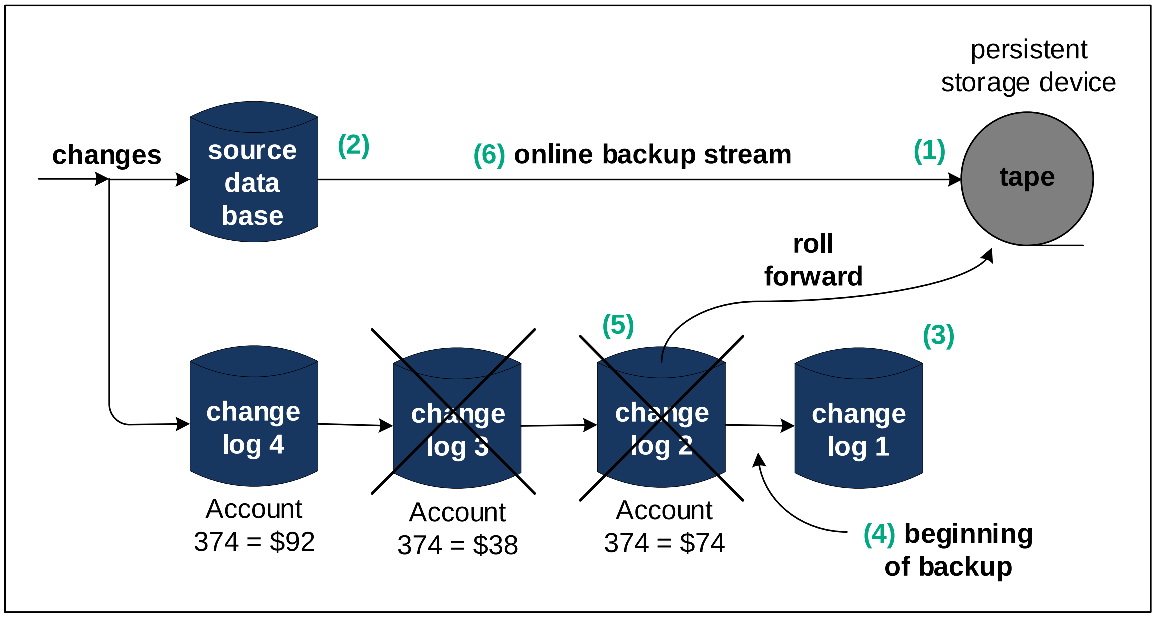

Figure 3 – The Better Backup Method

The Better Backup Method – Change Logs

Since the source database is actively being updated, restoring it from the backup medium does not provide a consistent database, because, for example, some of the data may be dirty, and changes made to that portion of the source database that were previously backed up are not included in the backup copy. These changes must be captured in a change log and applied to the restored version in order to make it *current, consistent,* and, *complete*.

The Better Backup Method recognizes that changes for data that are not backed up yet do not have to be written to a change log. These changes were made to the data in the source database and will be carried to the backup medium when they are written to that medium as part of the backup operation. Thus, the consistency of the backup database is preserved without having to roll these changes forward.

The Better Backup Method – Database Restore

During the restore process, the captured changes in the change logs must be rolled forward to the restored copy of the backed-up database. In Figure 3, change log 1 (3) contains changes that were made to the source database before the backup began (4). Therefore, its contents do not have to be rolled forward to the backup copy of the database when it is restored.

However, change log 2 (5) and subsequent logs contain changes that were made to the source database following the initiation of the backup; and these changes must be rolled forward to the restored backup copy to make the database consistent. Once the changes have caught up with the online backup, there is no further need to log changes and to roll them forward. All changes to the source database will be included in the online backup stream (6), guaranteeing the consistency of the backup database. Therefore, change logs 3 and 4 (and perhaps some changes in change log 2) do not have to be saved nor applied to the backup when it is restored.

Note that during the restore process, the database is not in a consistent state until all of the changes in the change log are rolled forward and applied to it. Thus, the restored database is *eventually current, consistent,* and *complete* (this process is also known as *eventual consistency*).

Also, note that the data being restored is not going to revert to previous values during the restore process. For instance, assume that the backup begins at time T1, and data D1 is changed after T1 to D2, then to D3, then to D4. This data object backs up at time T2 when its value is D2. The traditional approach backs up D2, then rolls forward changes and sets it back to D1 (as that is the first change restored), then D2, D3, and finally D4. Therefore, the database is very inconsistent during the restore process and in fact, is rolled back to a previous value when D1 is applied. If an application were to access this information during this process, it would read erroneous (old) data values for the data.

One alternative approach is to capture the database at D2 and not replay the D1 or D2 changes, and only replay the D3 and D4 changes. Over time, the database is consistent; it resets to older values than the final value, but not older than the initial value. Another alternative approach is to capture D2 and then overlay it with D3 and later D4 (either in the change log or the backup copy itself) before beginning the restore process.

To resolve backed up dirty data, either aborted information is removed from the logs during replay, or the dirty data is overwritten by the eventual “backout” data that is written when a transaction aborts. Removing the aborted information is a simple process if the logs are read in reverse, as discussed later, or if a list of aborted transactions is maintained along with the change logs so that when the change logs are applied (rolled forward), any aborted transactions can be skipped.

Only portions of the change logs that are required under the Traditional Backup Method are needed in the Better Backup Method. The fewer the change logs, the less processing is required to create them and the less storage is required to save them. Perhaps, even more importantly, the fewer the change logs, the less time is required to roll them forward, and the online backup/restore processing becomes much faster and more efficient. Additionally, the restored data goes through fewer data consistency issues (and in some implementations no issues) while it is being restored to a current and complete value.

Performance and Efficiency Improvements

An improvement in performance and efficiency can be achieved by saving only the last change to a specific data object that is being modified multiple times, as shown in Figure 4. In the figure, only the most recent change to a particular data item is shown ((5), change log 2); previous changes to that same data item are removed. More specifically, if a change is made to a data object that was previously changed, the first change can be located in the change log and replaced with the new change. If the first change previously was backed up, it can be located on the backup medium and replaced with the new change.

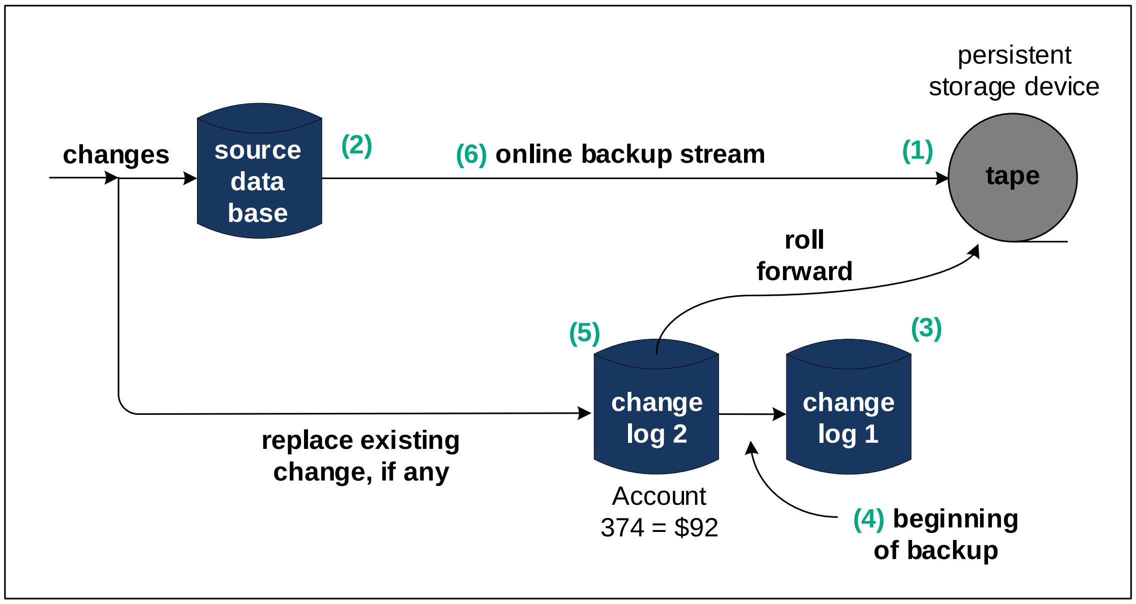

Figure 4 – Roll Forward an Existing Change

Alternatively, changes to previously backed up data can be directly made to the backup medium as shown in Figure 5. This method eliminates the need for change logs and roll-forward operations.

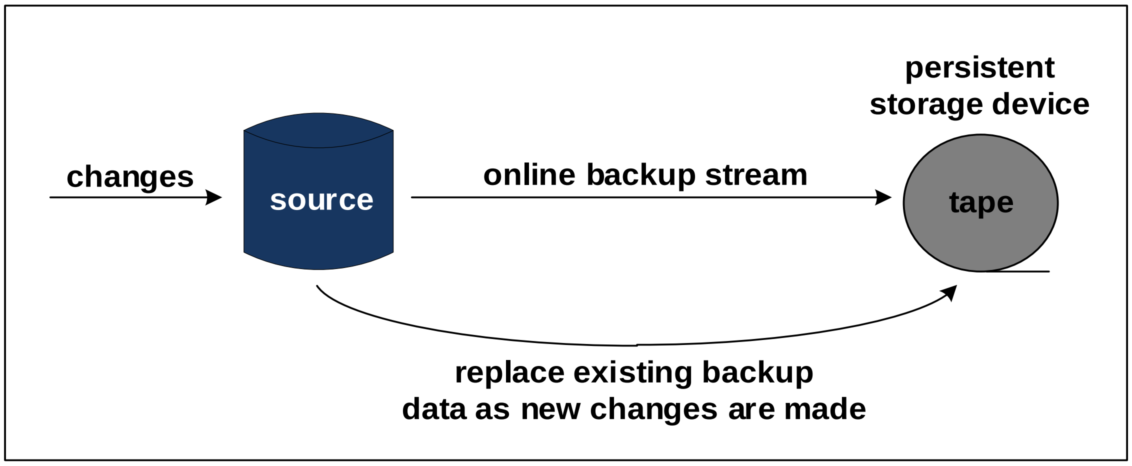

Figure 5 – Modify Existing Changes on Tape with New Changes

Another potential performance improvement can be achieved by reading the log files in reverse during the backup, and eliminating any data for transactions that abort as well as only saving the most recent (committed) change for each data item encountered.

In a similar manner, the backup operation can physically process the source database, block by block, rather than logically processing it by ascending (or descending) key path or some other logical or physical order (as mandated by the technology being used). This physical process can make the determination of whether to save a change that has occurred, since the backup is much faster. More specifically, using a physical path (such as the physical order the blocks appear in the file) to access the data is often much faster than using a logical path (such as an index tree) to access the data when the backup is initially taken.

The Best Backup Method — HPE Shadowbase Business Continuity

The Continuous Backup Method

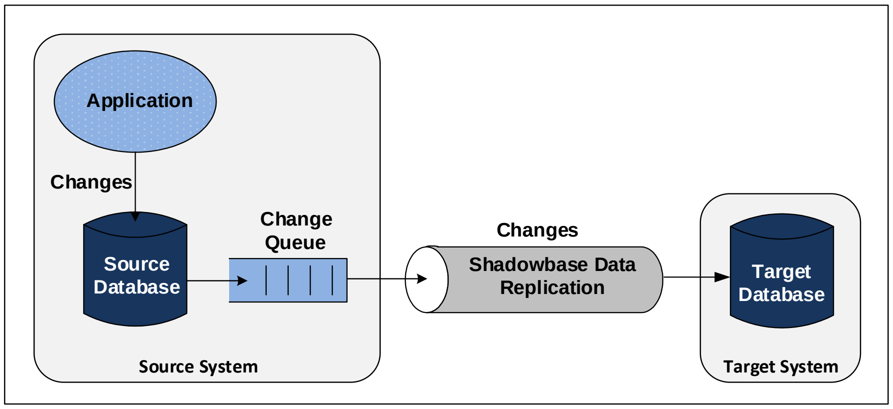

Figure 6 – The Continuous Backup Method with Data Replication

HPE Shadowbase Business Continuity solutions utilize the *Continuous Backup Method*, as shown in Figure 6 – The Continuous Backup Method with Data Replication which provides the capability to continuously save further changes made to the source database after the backup is taken in a persistent change log. As the backup copy is initially copied, any changes that were made to the previously copied portion are written to the *continuous backup* change log. Thereafter, all further changes to the source database also are written to the continuous backup change log. The backup copy becomes *current, consistent*, and *complete* at that (and every) point in time by continuously rolling forward the changes in the continuous backup change log onto the backup copy.[6] When it is time to restore the database, the backup copy simply is written to the target database to bring it *current, consistent*, and *complete*.

Of course, performing a continuous backup starts to approach the availability and consistency/completeness of using a data replication engine, such as the HPE Shadowbase software, to create and maintain the backup copy. While we advocate using data replication techniques to provide a viable backup copy of your production database, we understand that some customers will continue to require (or in addition to using a data replication engine) backup copies via the more traditional methods, especially for creating snapshot point-in-time copies of data. We hope that the new methods discussed in this white paper will help improve state-of-the-art solutions for such backups.

Summary

Periodically, it is necessary to create a backup copy of a database while it is actively being updated. The changes that occur during the backup must also be preserved so that a restore leaves the restored database in a *current, consistent*, and *complete* state. These changes are preserved in change logs resident on persistent storage.

Since there is a growing need for an economic and efficient way to back up big data, many companies will continue using Traditional Backup Method such as physical tape to fill this need.[7]

When creating a backup, it is not necessary to save changes for data that has not yet been backed up. Hence, the Better Backup Method creates a backup copy and logs only those changes that need to be applied to restore the database to a consistent state. (In other words, while the backup occurs, only the changes that have occurred for objects that previously were backed up are saved in the logs.) Similarly, it is not necessary to save all changes, or even the sequence of changes made since the data was backed up. Only the last (committed) value of the data needs saved while the backup is made. Hence, the Better Backup Method creates a backup and only one change log – to hold the last change made to any object that was updated while the backup occurred. Alternatively, the changes that occurred while the backup operation occurred are

[6] Visit Shadowbase Business Continuity for such a data replication engine implementation.

[7] Will tape continue its resurgence to become the backup media of choice? As of this writing, a Linear Tape File System (LTFS) cartridge can currently hold up to 2.5 terabytes of data. (Obviously, compression can improve the raw rates substantially further.) Tape's future certainly seems bright with the introduction of mega-density tape (about 29.5 gigabits per square inch in 2010 and approaching 145 gigabits per square inch in lab settings in 2018), which provides about three decades of shelf life.

directly overlaid on the backup data. Consequently, no change logs are necessary, and no roll forward operation needs to occur after the backup is restored. Thus, the Better Backup Method improves on the Traditional Backup Method since it minimizes the amount of changes that must be logged and played back to restore a database to a *current*, *consistent*, and *complete* copy.

The Best Backup Method — HPE Shadowbase Business Continuity

The HPE Shadowbase suite of replication products provides the full range of replication technologies to satisfy the most demanding IT availability requirements, meeting fast recovery times following a system outage with zero loss of data. The key to this technology is data replication.

The Best Backup Method using HPE Shadowbase Business Continuity software ensures the backup copy is always *current, consistent*, and *complete* from the point of the backup operation and afterwards, meaning subsequent source database changes are saved to the target backup database on persistent storage in real-time.

Since each backup method supports different ranges of recovery times (RTOs) and data loss (RPOs), it is important to understand the costs of downtime and data loss for each application in conjunction with the costs of achieving various levels of high and continuous availability[8]. Performing this analysis empowers IT management to make informed decisions concerning which backup method is right for each and every application.

Hewlett Packard Enterprise directly sells and supports Shadowbase for business continuity solutions, Shadowbase for data and application integration solutions, Shadowbase Compare solutions, and the Shadowbase Essentials Bundle under the name *HPE Shadowbase*. For more information, please visit ShadowbaseSoftware.com or contact your local HPE Sales representative.

[8] For more information, see *Choosing a Business Continuity Solution to Match Your Business Availability Requirements*.

International Partner Information

Global

Hewlett Packard Enterprise
6280 America Center Drive
San Jose, CA 95002
USA
Tel: +1.800.607.3567
www.hpe.com

Japan

High Availability Systems Co. Ltd
MS Shibaura Bldg.
4-13-23 Shibaura
Minato-ku, Tokyo 108-0023
Japan
Tel: +81 3 5730 8870
Fax: +81 3 5730 8629
www.ha-sys.co.jp

Gravic, Inc. Contact Information

17 General Warren Blvd.
Malvern, PA 19355-1245
USA
Tel: +1.610.647.6250
Fax: +1.610.647.7958
www.shadowbasesoftware.com
Email Sales: shadowbase@gravic.com
Email Support: sbsupport@gravic.com

Hewlett Packard Enterprise Business Partner Information

Hewlett Packard Enterprise directly sells and supports Shadowbase Solutions under the name ***HPE Shadowbase***. For more information, please contact your local HPE account team or visit our website.