GRAVIC®
Shadowbase

# Choosing a Business Continuity Solution to Match Your Business Availability Requirements

**A Gravic, Inc. White Paper**

## Executive Summary

Today, businesses with access to real-time online transactional data have a competitive advantage. To gain the most potential from this data it must be current and available at any given time. The counter to this advantage is that the inability to access or update current data carries a significant business cost, possibly measured in many thousands of dollars per second. These requirements necessitate an IT infrastructure that is continuously available.

Business continuity encompasses those activities that an enterprise performs to maintain consistency and recoverability of its data, operations, and services. Application availability depends upon the ability of IT services to survive any fault, whether it is a server failure, a network fault, or a datacenter disaster. Data availability depends on the existence of up-to-date backup data copies. An enabling technology for achieving high or continuous availability for application services and the timely backup of important data is *data replication*. Selecting the right data replication technology to achieve your business continuity goals is the focus of this white paper.

Availability doesn't come for free. Every organization has a variety of IT applications, ranging in importance from noncritical to mission-critical. Some applications can be down for hours or even for days with little impact on operations. Downtime for other applications may be quite costly but survivable. A handful of applications cannot be down at all without loss of critical services, loss of revenue, loss of life, or otherwise wreaking major havoc on operations. Likewise, some data is of lower value and can afford to be lost, but other data is highly valuable and must be recoverable. The net of this is that IT system downtime incurs business costs of one form or another.

Fortunately, a range of data-processing architectures exists for meeting every application-availability and data recoverability need. Typically, a common characteristic of such architectures is that the more availability they provide, the more costly they become and the more complex they are to implement and to manage. However, the overall return on investment of these more costly highly available architectures is positive, by preventing prolonged periods of downtime or data loss which can cost the business thousands or even millions of dollars.

Choosing among availability options can be a daunting task. Availability solutions range from *normal* availability (hours or days of recovery time) to *high* availability (minutes of recovery time) to *continuous* availability (seconds or less of recovery time). These levels of availability are often measured in 9s. Normal availability is typically three 9s or less, meaning that IT services are available 99.9% of the time. This availability corresponds to annual downtimes of nine hours or more. Highly available systems have availabilities of four or five 9s (several minutes to an hour of annual downtime). Continuously available systems have availabilities of six 9s and beyond, meaning that they average less than a few seconds of downtime per year. Likewise the amount of data loss varies across the range of availability solutions; some may lose hours or even days' worth of data, whereas others promise zero data loss in the event of a failure.

The technology now exists to achieve arbitrarily fast recovery times following a system failure with little or no loss of data. The key to such fast recovery and little if any data loss is *data replication*. As previously indicated, we discuss the data replication technologies that today's highly available and continuously available architectures use to achieve various levels of business continuity. We also review the process employed to decide what availability solution is best suited for each of your applications.

The key to the decision-making process is a comprehensive business risk assessment to determine for each application the cost of downtime and the cost of lost data. Cost can be measured in terms of dollars, lost opportunities, regulatory violations, and even loss of life or property. Given the risk assessment, management can then determine the recovery time needed for each application if it fails. This assessment is called the Recovery Time Objective, or RTO. The tolerance to data loss is also an important factor. The amount of data loss that is acceptable for an application is called the Recovery Point Objective, or RPO.

Data replication comes in several forms – hardware and software, asynchronous and synchronous, uni-directional and bi-directional. Each combination supports different ranges of recovery times (RTOs) and data loss (RPOs). By understanding the costs of downtime and data loss for each application and the costs of achieving various levels of high availability and continuous availability, IT management can make informed decisions concerning the availability approaches that are right for each application.

The focus of this paper is to provide management with an understanding of data replication technologies for informed decision-making. We also summarize the HPE Shadowbase suite of replication products, built by Gravic, Inc., sold by Hewlett Packard Enterprise. Shadowbase software provides the full range of replication technologies, from high to continuous availability, and is able to satisfy the most demanding IT availability requirements.

# Choosing a Business Continuity Solution to Match Your Business Availability Requirements

## Table of Contents

## Table of Figures

# Choosing a Business Continuity Solution to Match Your Business Availability Requirements

## 1 Redundancy

In order to achieve high levels of IT service availability, single points of failure should be eliminated, meaning that every component that is necessary to support an application should be backed up by an equivalent (not necessarily identical) component. Redundancy applies to processors, data storage, networks, sites, power, cooling, and others. In some cases, a company may elect not to back up certain components that are considered highly unlikely to fail.[1] In these cases, the company is willing to take the consequences of the (presumably) infrequent failure of those components.

### *1.1 Isolation*

Redundant components should be isolated so that the failure of one does not affect its backup. For instance, depending on two power feeds into a datacenter from the same power grid violates isolation since if one fails due to a grid fault, the other fails. Likewise, dual networks from the same communication carrier violate isolation.

Another violation of isolation is inherent in those architectures that require a backup system to be identical to the primary system, even down to the hardware, the software versions, and the database schema. One problem associated with the requirement for identical facilities is that a bug in one system may simply reoccur in the other system following a failover. Another problem is that it is often difficult for system administrators to know if changes made in one system do, in fact, make it to the other system. These problems often occur in active/passive system configurations and clusters.[2]

Truly isolated redundancy implies that the backup facility does not have to be identical to the primary facility. What is necessary to satisfy the isolation criterion is that a problem or a configuration change affecting one system does not impact the other system.

### *1.2 Dispersion*

Redundant components must not be collocated; they must be geographically dispersed so that some single event cannot disable both of the redundant components. An important single point of failure is the datacenter. Reports of entire datacenters going down for hours or days occur regularly. Disasters such as fires, floods, tornados, and earthquakes as well as more subtle events such as employee malfeasance, hackers, and even law-enforcement confiscation of datacenter equipment can cause datacenter outages.

The only solution to this problem is to have two or more datacenters located sufficiently far apart from each other so that a disaster at one site does not affect other sites. Fault-tolerant architectures such as a single HPE NonStop system, clusters, and virtual machines are highly redundant but are inherently not dispersible. Though they form a solid basis for a highly available or continuously available datacenter, true high availability means that they must be replicated in a distant datacenter.

The distance by which datacenters must be separated depends upon the threats. In Europe, for instance, where there is no expectation of significant earthquakes or hurricanes, datacenter separations of tens or hundreds of kilometers are often considered adequate to protect against local disasters such as fires, floods, and power outages. In other, more disaster-prone areas, hundreds or thousands of miles are frequently required.

Additionally, local regulations may play a role in dictating distances. For instance, European countries generally dictate their own distance requirements within the country for critical applications. In the United States, federal regulations call for a minimum distance measured in the hundreds of miles for critical financial applications.

[1] "Highly unlikely to fail" does not mean failure-free. Beware of letting a high number of 9s lull you into a false sense of security. The question is not, "Can it fail?" The question is "When will it fail?" More importantly, what are you willing to do to prepare for such an event? After all, that "highly unlikely" event may just happen tomorrow.

[2] See section 11.1 for a description of active/passive system configurations.

### *1.3 Failover*

A redundant component is useless if it cannot readily assume the duties of its failed counterpart, requiring multiple capabilities to be part of the architecture:

- *Fault detection* to rapidly identify faults
- *Failover decision* to determine whether it is better to try to restore the failed component (for instance, to reboot a processor) or to fail over to its backup
- *Failover action* to perform the failover if that is the desired action
- *Verification* to validate that the backup component is providing the required service

To the extent that these capabilities are automated, failover is faster. It also typically is more reliable and less error prone. However, in many cases, one or more capabilities are deemed too complex for automation and are left to the system operators and their management. For these cases, established procedures, thorough documentation, and periodic operator practice are important.

### *1.4 Testing*

Perhaps the availability requirement that is most often ignored is periodic testing of the failover plan. In many architectures, failover testing requires stopping processing and failing over to the backup, a course of action that might take down user access to the application for hours. There also is the possibility that the failover will fail, further increasing the length of the outage. For this reason, many companies don't test their failover plans or only do so rarely or in part. When a real failure occurs and they must fail over, they can only hope for the best (a hope that is frequently unsatisfied). Such issues arising which prevent the timely failover to a backup system are known as *failover faults*. Some architectures lend themselves to continuous backup system testing without taking the primary system offline so that the operations staff always knows that the backup system is working properly and will failover reliably (the backup is a "known-working" system).

## 2 The Components of Availability

Many factors influence the availability of data-processing systems. Factors range from planned outages to unplanned outages. Fortunately, there are data replication techniques capable of reducing or eliminating both planned and unplanned outages.

### *2.1 Planned Outages*

Planned outages are often necessary for hardware upgrades. Also, many software upgrades require that the system be taken down, including upgrades to the operating system, applications, and database management systems. Certain database operations such as rebalancing indices, modifying the database schema, or making consistent tape backups may also have to be performed on a quiesced system. Typically, a datacenter site move necessitates that all systems be taken down.

When a non-redundant processing system or a node is taken down for planned maintenance, the business processes are unavailable to users during this planned maintenance interval. However, if the system is redundant, another node in the system can continue to provide processing services.[3]

For non-redundant systems, sometimes there are maintenance windows when the system is idle and upgrades can be performed. These idle times are usually at night or over a weekend, but the risk remains that the upgrades may not be completed in time. With the increasing globalization of trade and the "always on" expectation of users, such windows are becoming increasingly short, if they exist at all.

### *2.2 Unplanned Outages*

If business services are to be maintained, an unplanned outage requires either an immediate recovery of the failed component or failover to a redundant component, if there is one. Any number of faults can cause an unplanned outage, including hardware failures, software bugs, operator errors, environmental faults such as power or air conditioning, employee malfeasance, hackers, denial-of-service attacks, and even datacenter

[3]For more information, see the Gravic white paper, *Using HPE Shadowbase Software to Eliminate Planned Downtime via Zero Downtime Migrations*.

destruction due to a disaster. Nearly 95% of companies that experience data loss or downtime for 10 days or more will file for bankruptcy within 12 months, and 43% of businesses without a disaster recovery plan will go out of business in the wake of a major data loss, according to the National Archives and Records Administration.

There are many types of disasters that can render your business ineffective and unable to operate; four common ones are:

- **Natural Disasters –** 40% of business do not reopen following a disaster, according to the Federal Emergency Management Agency (FEMA). On top of that, another 25% fail within one year.
- **Hardware failures –** 65% of businesses reported unplanned downtime due to a loss of power or utilities.
- **Human Errors –** 80% of unplanned outages are due to ill-planned changes made by administrators, according to the IT Process Institute. Additionally, misconfigurations account for a high number of extended outages.
- **Cyber Crimes –** Data breaches are on the rise in the U.S. In fact, more than 60% of businesses experienced phishing and social engineering attacks in 2018, and more than four billion records were exposed through data breaches during the first half of 2019 alone.[4]

There are two major availability concerns when an unplanned outage occurs – the availability of the processing infrastructure and the availability of the application data. Having all of the servers up and running does no good unless they have valid data upon which to operate. Processing availability and data availability is covered next.

# 3 Application-Services Availability and the Recovery Time Objective (RTO)

The availability of the processing infrastructure – servers, networks, power, etc. – varies widely depending upon the redundant architecture used, and we explore these architectures later. First, we review how to characterize the availability of processing services.

## *3.1 Uptime*

The typical way to measure availability is to predict (or measure) the percentage of time that the system is up – its uptime. Uptime is often reported in *"nines" (9s)*. For instance, if a system is up 99.9% of the time, its uptime (thus, its availability) is quoted as "three 9s." The relationship between 9s and downtime is shown in Table 1.

| Uptime | Downtime/year |
|---|---|
| one 9 | 876 hours |
| two 9s | 88 hours |
| three 9s | 9 hours |
| four 9s | 50 minutes |
| five 9s | 5 minutes |
| six 9s | 30 seconds |

**Table 1 – Downtime and 9s**

Keep in mind that these times are averages. Four 9s, for instance, does not mean that you will experience a one-hour outage each year. You will achieve four 9s if you have a four-hour outage every four years or a one-day outage every twenty-four years. Can your business survive that? It is up to you to decide the criticality of your various applications and the downtime that you are willing to tolerate for each.[5] We are concerned with techniques for achieving five 9s of availability and beyond, which is typically the minimum requirement for the most mission critical business systems where even a few minutes of downtime can cost thousands or even millions of dollars.

[4] *7 Steps to Create a Reliable Disaster Recovery Plan*; Odyssey Information Services; Blog by Donna Stanford; January 31, 2020.
[5] It is obviously a business decision, as the amount of availability that the business requires will drive the architectures to consider and the cost and complexity of the system to be implemented.

### 3.2 Disaster Recovery versus Disaster Tolerance

*Disaster recovery* is the ability to bring up backup facilities to carry on the business following the loss of data processing. This ability may require recovering the database, loading applications, testing the backup site before IT operations resumes, and finally switching the network and the users over to the recovered system. Disaster recovery can be as fast as a few minutes, but typically takes hours or days.

*Disaster tolerance*, on the other hand, is the ability to continue operations in the event of a disaster without users noticing that there has been an outage or at least without denying application services to the users for very long. How long is “very long”? It depends upon your application requirements, as discussed below, but would typically be measured in seconds. In this paper, we focus on achieving disaster tolerance for your critical applications.

### 3.3 High Availability

Typically, in the industry today, *high availability* is seen as an availability of five 9s or greater. Again, keep in mind that this amount is an average. It is achieved if a system fails five minutes once per year or one hour every twelve years.

### 3.4 Continuous Availability

If an application requires an availability of six 9s or beyond, the measure of 9s is no longer very helpful. Measuring seconds of downtime per year is simply not practical. Rather, what is important is failover time and failover reliability. If the system fails over from a hardware or software fault or even a datacenter disaster in sub-seconds or seconds, and if it is known that the failover will succeed, then true disaster tolerance as defined above is achieved; this is known as *continuous availability*.

### 3.5 Recovery Time Objective (RTO)

The tolerable failover time for an application is defined as the *Recovery Time Objective* (RTO). It is the target time by which a failover recovery and resumption of business services must be completed following an outage. It is determined by careful evaluation of the all business costs involved with system downtime. We use this measure extensively in the following discussions. Data replication technology is a key enabler to achieve either high availability or continuous availability by minimizing RTO (an RTO approaching zero is possible in some cases).

## 4 Data Availability and the Recovery Point Objective (RPO)

The previous section dealt with system availability. But an available system with full processing capacity is of no use if it does not have correct and current data to process. In some cases, access to a complete history of activity is also required.

There are many reasons why important data may be lost, from system failures to datacenter failures. Data is typically protected by backing it up. If there was a failure of some sort, the data could be restored from its last backup, but any data updates that had occurred since the last backup would be lost.

Each application has a different importance to the company, and the importance of the application-generated data varies. In some cases, the data has little value; but in other cases, the loss of a few minutes of data, while painful, may be acceptable. For mission-critical applications, especially those with high value transactions, zero data loss may be required.

### 4.1 High Data Loss

Perhaps the data associated with an application has little value. Hours of data can be lost with no serious impact to the company. Data used only for statistical analysis, such as counting web clicks, is an example of this type of data.

### *Little Data Loss*

For many applications, data is very important. Though its loss would not imperil the company, the cost of losing this data might be very high. In some cases, it might be possible to recreate the lost data via manual data entry once the system is restored so long as not more than a few minutes of data are lost. Examples of this kind of data include ATM transactions and cell phone call data records.

### *4.2 No Data Loss*

Some applications represent the core of the business, and no data loss is tolerable. Any lost data may be unrecoverable, and such loss could seriously impact the fundamental operations of the business. An example is the trading data for a stock exchange. If trade data is lost and cannot be reconstructed, there is no basis for establishing the price of the various instruments traded on the exchange. In these cases, data loss cannot be tolerated. Other examples are electronic funds transfers (EFT), where individual transactions may be worth millions of dollars, and health care records, including patient medication records, where a loss could lead to incorrect dosages being given with potentially serious consequences.

### *4.3 Recovery Point Objective (RPO)*

The amount of lost data that an application can tolerate is its *recovery point objective* (RPO). Like *recovery time objective* (RTO), RPO must be determined by careful evaluation of the business costs involved with loss of data. We address data replication techniques that reduce RPO to seconds and even to zero if required.

## 5 Specifying Availability with RTO and RPO

RTO is expressed as time and is the amount of time that the company can afford to operate without the data-processing services of an application. RPO is also expressed as time and is the amount of application data loss that the company can tolerate due to an unplanned outage.[6]

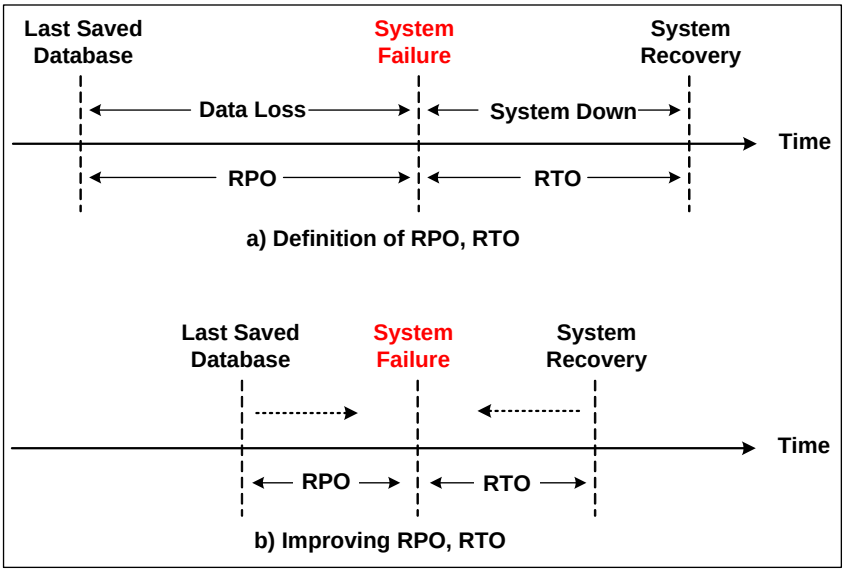


**Figure 1 – The RPO/RTO Relationship**

In Figure 1 a, we show the relationship between RPO and RTO. Data backups might be made by quiescing the system and dumping the database to magnetic tape, by executing online dumps to disk, by using audit-trail rollups, or by employing a number of more immediate techniques such as data replication. If the primary system fails, the last database backup can be loaded onto a backup system and then processing continues. The amount of application data lost is defined as the new data generated from the time of the last database

[6]Expressing data loss in terms of time may seem counterintuitive. In real systems, however, it is a practical method to use. It is the time from the generation of data to the time that it is safe-stored on the target system. This time value can be converted directly to the amount of lost data by using the transaction rate. If the RPO is one minute, and if transactions are being generated at a rate of 1,000 transactions per minute, it is equivalent to specifying that no more than 1,000 transactions are lost (specifically, no more than the data updates associated with those 1,000 transactions).

backup or replication point to the time of failure. This amount is the maximum amount of data that can be lost and yet still achieve the desired RPO goal. In Figure 1a, this amount is represented as the period between the left vertical bar ("Last Saved Database"), and the center vertical bar ("System Failure"), which is the point of complete failure (outage) of the primary business processing system. Techniques for moving the left vertical bar to the right, closer and closer to the point of system failure, thereby reducing or eliminating data loss, as shown in Figure 1b, is one focus of this paper.

Following a failure, steps are taken to recover processing, which may entail repairing the downed system, or may require switching over to a backup system. The time from the point of failure to the time of recovery must be less than the RTO specified for the application. In Figure 1a, time is represented as the period between the center vertical bar, and the right vertical bar ("System Recovery"). Techniques for moving the right vertical bar closer and closer to the point of system failure, thereby reducing system downtime, as shown in Figure 1b, is also a focus of this paper.

RPO specifies the maximum amount of time between a backup of the database (full or incremental) and the point of failure, assuming that the database is active. For instance, if RPO is four hours, database backups (full or incremental) must be taken more frequently than every four hours. If RPO is five seconds, data replication is required, and the latency of the data replication engine cannot exceed five seconds.[7]

RTO specifies the maximum amount of time from the point of failure to system recovery. Having to repair a system in order to restore processing services can result in RTOs measured in hours or even days. Failing over to a backup system reduces RTO to times ranging from hours down to minutes. Failing over to an operating backup system lessens this time to seconds or even sub-seconds. Data replication technology allows systems to be configured to meet any required RPO and RTO. In the extreme, zero RPOs and RTOs measured in sub-seconds can be achieved.

## 6 An Introduction to the Business Continuity Continuum

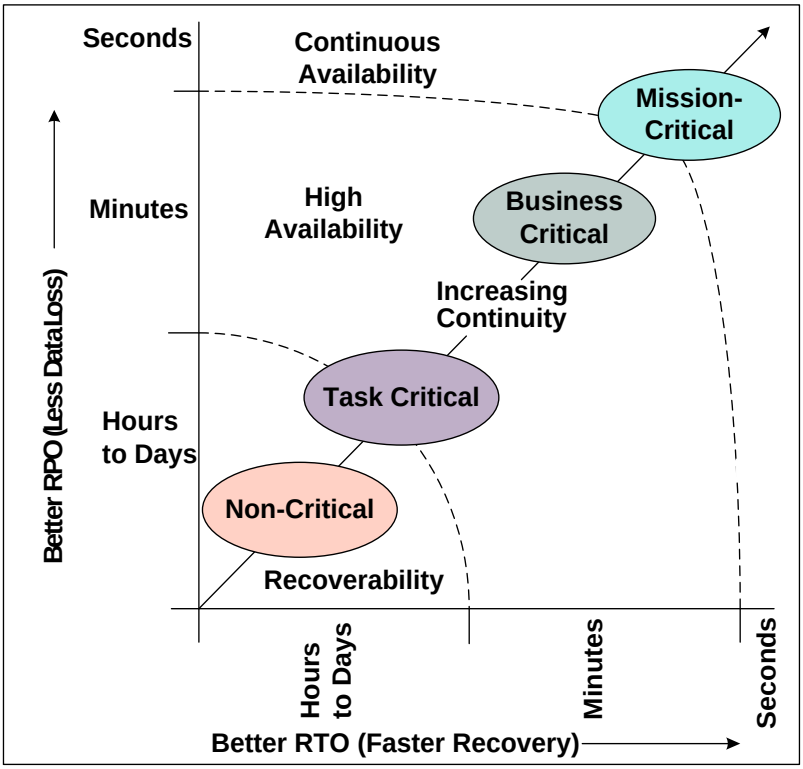


**Figure 2 – The Business Continuity Continuum**

The dependency of various business applications to availability is represented as a business continuity continuum in Figure 2. The continuum plots RTO versus RPO. The horizontal access shows improving RTO,

[7]Data replication latency is the time between when the data is changed in the database, and when it is replicated and safe-stored on a backup system. This is the data in the "replication pipeline" that is at risk when a failure occurs.

trending toward seconds of recovery time, while the vertical axis shows improving RPO, trending towards zero lost data. Various types of business application are shown related to their RTO/RPO requirements. Applications of low business value that tolerate longer outage times and more data loss and are towards the lower left on the continuum. Applications with the highest business value that do not tolerate much downtime or data loss are towards the upper right on the continuum.

The continuum represents various levels of availability. In the worst case, recovery following a failure may be measured in hours or days, and hours or days of data may be lost. This case represents applications in which there are little or no redundancies. Servers may have to be re-provisioned and databases rebuilt. This level of availability is appropriate for noncritical applications and task-critical applications that may have only a small impact on employees or users.

The next level of availability is high availability. Highly available applications are those that use redundancy to reduce recovery time and lost data to minutes. Such applications include most task-critical applications, in which employees are unable to complete their tasks while the application is down. Also included are business applications whose downtime could have a major impact on the operations of a company.

The highest level of availability is continuous availability. At this level, recovery time is measured in seconds. Data loss is also measured in seconds or, in some cases, zero. Continuous availability is required for many mission-critical applications whose failure could cause major harm to a company, to people, or to property.

We explore the technologies that support these various levels of availability and note examples of applications that fit in each level. As the level of availability increases, the cost of the solution also necessarily increases; however, the business impact of an outage and the overall TCO of the solution decreases. We guide you in choosing the right level of availability for your application given its RTO and RPO requirements, with a particular focus on availability solutions for applications requiring high or continuous availability.

## 7 Data Replication – The Fundamental Force Behind High and Continuous Availability

Improving your availability via data replication depends upon having at least two nodes, each being capable of hosting a database. Typically, each node must also host the application(s) to be protected.[8] As shown in Figure 3, the purpose of data replication is to keep a target database synchronized in real-time with a source database that is being updated by a source application.

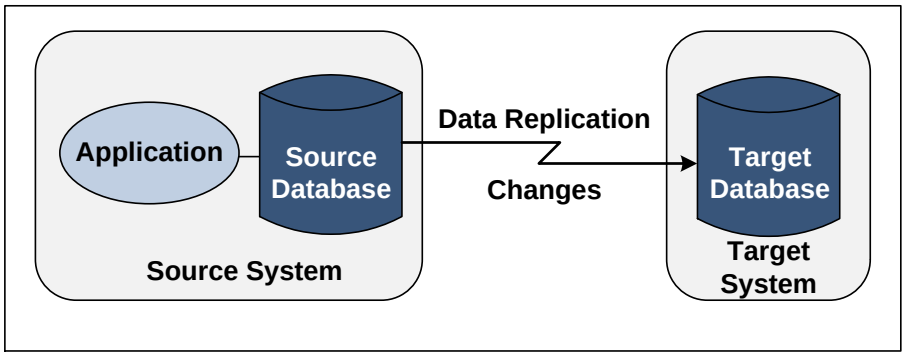


**Figure 3 – Data Replication**

We talk about the *source database* hosted by the *source node* and the *target database* hosted by the *target node*. The two nodes comprise the distributed data-processing system. As an application makes changes (inserts, updates, and deletes) to its local database (the source database), those changes are sent immediately over a communication channel to the target system, where they are applied to the target database (Figure 4). The target database typically resides on another independent node that may be hundreds or thousands of

[8]There are replication architectures in which the target node is simply a "data bunker" with the purpose to safe-store data; it does not run applications. In the event of a primary system failure, the data bunker is used to bring the database of the backup system up-to-date. Depending upon the architecture used, these approaches can offer little to no data loss. However, they typically have very long recovery times (many hours to days), and may be distance or dispersion-limited depending on the technologies selected. In this paper, we will focus on replication architectures that provide low to no data loss but have very fast recovery times and are not dispersion-limited.

miles away. We call the facility that gathers changes made to the source database and applies them to the remote target database a *replication engine*.

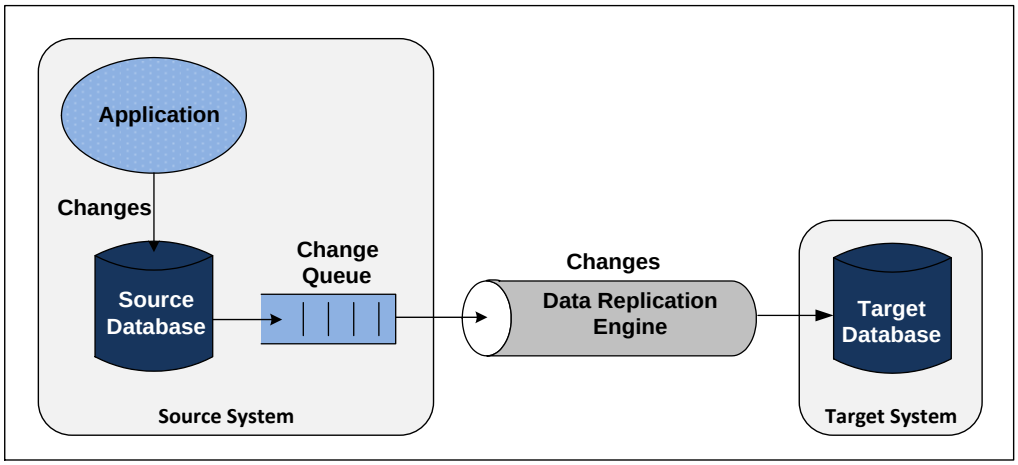


**Figure 4 – Data Replication Engine**

## 8 An Overview of Data Replication Techniques

Data replication engines can be categorized in several ways:

- Hardware versus Software Replication – With *hardware replication*, the replication engine is implemented via low-level device drivers, typically in the storage subsystem. A *software-based replication engine* can reside either in the storage subsystem or in the processing nodes themselves.

- Asynchronous versus Synchronous Replication – *Asynchronous replication* sends source database changes to the target database without impacting the source application. Changes are sent after-the-fact from a queue of changes maintained on the source node. The application and the data replication engine are decoupled from each other via the change queue. *Synchronous replication* allows no changes to be made unless they can be made to all database copies simultaneously (source and target). The application and the data replication facilities are coupled to each other. Depending upon the approach used, they may be either tightly coupled or loosely coupled.

- Uni-directional versus Bi-directional Replication – With *uni-directional replication*, changes are sent in just one direction from a source database to a target database. With *bi-directional replication*, both databases can be active, and changes made to either are replicated to the other. In this case, each database is both a source and a target. In either case, replication can be either asynchronous or synchronous.

## 9 Hardware versus Software Replication

### *9.1 Hardware Replication*

#### 9.1.1 Replicate on Cache Flush

Hardware replication is usually implemented in the storage system controller. It replicates disk blocks as they are written to the source disk, thus guaranteeing that the contents of the target disk are always identical to the source disk. However, disk blocks are typically only written to disk when they are flushed from the disk's cache. There is no logical order to the disk-write sequence since other factors control cache flushing – disk blocks that are the least recently used are flushed to disk when cache space is needed for new blocks that must be read from disk. As a consequence, the target disk is not guaranteed to be consistent; target disk blocks may be partially split; indices may exist without the rows or records to which they refer; and children may exist without parents. The data is consistent in cache, but the target disk image is generally useless. As a result, applications cannot use the target database for any application processing. If the source node fails, a lengthy recovery process is required to bring the target database into a useful, consistent state. Additionally, because

of the cache-flushing issue, large amounts of data may be lost due to a source-system failure even if synchronous replication is used, as any data still in cache will not have been flushed at the time of failure.

#### 9.1.2 Replicate on Cache Update

Some storage controllers replicate changes as they are made to a disk's cache regardless of whether they have been physically written to the source disk or not. The replication of cache updates ensures the logical consistency of the target database since changes are replicated to the target system as soon as they are made at the source system. If synchronous replication is used, no data will be lost following a source system failure. However, due to other limitations as described next, the target database may still not be useable by applications.

#### 9.1.3 Other Attributes of Hardware Replication

Hardware replication, whether based on disk flushing or cache updates, typically sends blocks of changes to the target. In some cases, the controller compresses data to only those bytes that have changed. Hardware replication is limited to specific hardware and may not be possible in your configuration. Both hardware replication techniques typically do not replicate current data locking protocols nor transaction end-state information (commits and aborts). Hence, the target database typically contains many "dirty records" and is unusable by applications.

These replication techniques generally require the identical storage technology down to the version to be used at both the source and the target. They also do not typically allow the target database to be opened by applications at the same time that replication is taking place, thus preventing their use in active/active systems. Consequently, hardware replication is not an option if recovery times measured in seconds or minutes is to be achieved.

Another issue with hardware replication is that the maximum distance between the source and target disks is limited. Having the target disk insufficiently removed from the source disk violates the redundancy requirements of both isolation and dispersion, as discussed above. For these reasons, and its consequent unsuitability as a method for achieving high levels of availability and minimal data loss, hardware replication is not discussed further in this paper[9].

### *9.1 Software Replication*

In the high availability systems that we consider, higher level software carries out the replication task. A data replication engine running on the source and target systems performs the replication. In only this way, can highly available active/passive and continuously available active/active systems be implemented.

Software replication engines typically read changes from a change queue of some sort and send them to the target system to update its database. As long as updates are made to the target system in the same order as they were made at the source system, the target database is consistent and usable by other applications. Some high-performance replication engines are multithreaded to improve replication throughput. In these engines, resynchronizing facilities reorder updates that may be received out-of-order from the various threads before the updates are applied to the target database, maintaining target database consistency.

Software replication may be by event, by transaction, or by request. *Event replication* replicates data-manipulation language (DML) events as they occur. DML events include insert, update, and delete operations. Event replication in some cases may also replicate data-definition language (DDL) operations that affect the database's data structure and schema. *Transaction replication* replicates entire transactions, either one operation at a time as they occur or as a group of operations once the transaction has committed on the source. When replayed at the target, the transaction is either committed or, if the entire transaction is not received, is aborted. *Request replication* replicates the entire application request, which is reprocessed in its entirety by the application running on the target system.

[9]For more information, see the Gravic companion white paper, *Hardware vs Software Data Replication for Business Continuity (Disk/Cache Flushing vs. Transactional Replication Engines)*.

# 10 Asynchronous versus Synchronous Replication

### *10.1 Asynchronous Replication*

An *asynchronous data replication engine* is completely transparent to the applications running in the source node. As shown in Figure 5, it extracts changes made to the source database from a change queue and sends them after-the-fact to the target database. The replication engine makes changes to the target database copy somewhat later than they were made to the source database. The result is that the databases are synchronized, but the target database copy lags the source database by a short interval. This interval is known as the *replication latency* of the data replication engine.

The replication latency associated with an asynchronous replication engine creates an issue that must be considered when using such technology. This issue is data loss following a failure of the source node. Any changes that were in the replication pipeline and that did not make it to the target node may be lost. The RPO in this case is the replication latency of the replication engine, which is typically measured in the tens or hundreds of milliseconds. Whether this amount of data loss is acceptable depends entirely upon the nature of the application and the value of the data. For many applications such data loss is tolerable, and for others it is unacceptable. These factors go into the decision regarding which replication solution is required for a particular application.

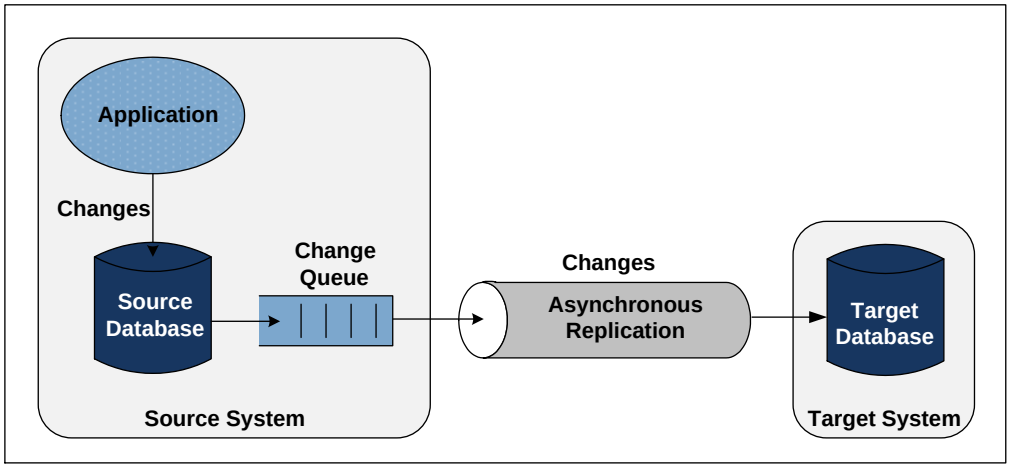


**Figure 5 – Asynchronous Replication Engine**

### *10.2 Synchronous Replication*

A *synchronous data replication engine* solves the asynchronous replication problem of data loss following a node failure. Synchronous replication makes no permanent changes to any database copy unless those changes can be applied to all database copies (for example, to both the source *and* target databases); so if a node or the network fails, no data is lost. Synchronous replication can satisfy RPOs of zero (that is, no data loss), but it has its own issue, which is *application latency.* Since the application must wait for the transaction's data to be safe-stored and optionally applied to all database copies in the application network, the source application's transaction completion is delayed.

There are two primary methods for synchronous replication – *dual writes* and *coordinated commits*. When using *dual writes*, all copies of the database are included within the scope of the transaction (Figure 6). The application must wait for each source database update to also complete across the network in the target database before proceeding to the next update. It must then wait for the transaction to be committed both locally and across the network before being informed that the transaction is complete, whereupon processing may continue. This delay is a function primarily of the communication latency between the nodes (which is related to the distance separating the nodes) and of the size of the transaction. Thus, the nodes typically must be near each other – such as in the same campus or metropolitan area – and connected by very fast media such as fiber, which may not allow the degree of separation required for proper disaster tolerance.

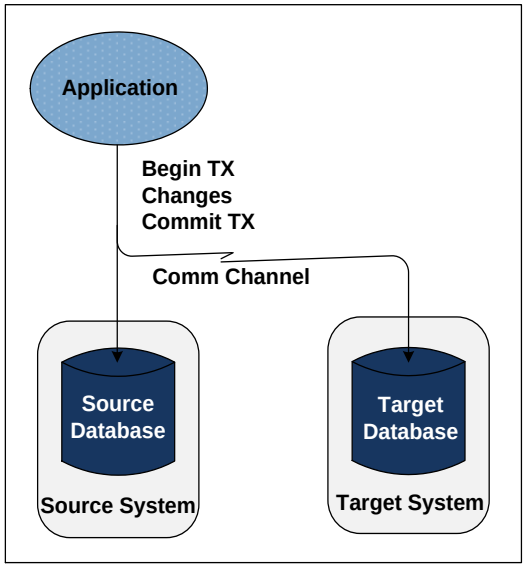


**Figure 6 – Dual Writes**

The *coordinated commit* method for synchronous replication minimizes application latency. A coordinated commit replication engine is a combination of synchronous replication and asynchronous replication techniques. The coordinated commit replication engine registers as a voting member of the source system's transaction. As shown in Figure 7, changes made by the application are sent to the target database asynchronously so that they do not impact the application, as they do with the dual write method. It is only at commit time that the coordinated commit replication engine must wait for one replication latency period to check with the target database to ensure that it has received all of the data for the transaction and can thus vote "yes" to commit the transaction.

Hence, the coordinated commit technique imposes an application latency of one replication latency plus one channel latency as it coordinates with its target database, but only at commit time. Even if the nodes are separated by thousands of miles, application latency with this method can be as small as tens of milliseconds. In summary, coordinated commits enables the benefits of synchronous replication while mitigating the drawbacks associated with dual writes, by minimizing application latency and imposing no distance restriction between the nodes.

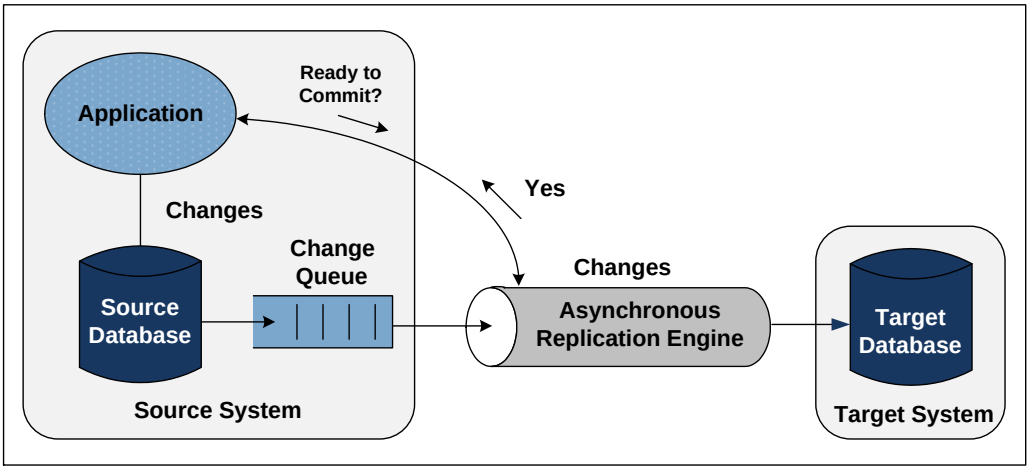


**Figure 7 – Coordinated Commits**

## 11 Uni-directional versus Bi-directional Replication

### *11.1 Uni-directional Replication and Active/Passive Systems*

A *uni-directional data replication engine* replicates data in one direction – from a source database to a target database. Figure 5, Figure 6, and Figure 7 are examples of uni-directional asynchronous and uni-directional synchronous data replication. Uni-directional replication is often used to keep a passive backup system[10] synchronized with an active production system. These systems are known as *active/passive systems*. Uni-directional active/passive systems always have much longer recovery times than bi-directional active/active systems, described next.

### *11.2 Bi-directional Replication and Active/Active Systems*

A *bi-directional data replication engine* replicates data in *both* directions between two databases. Each database is acting both as a source database and as a target database. Since any change made to one

[10]The system is passive in the sense that it is not processing transactions and updating the database.

database is reflected in the other database via data replication, every node in the application network has a current copy of the application database and can participate in the application. Each node is actively processing transactions, and such systems are known as *active/active systems*.

### 11.2.3 Advantages of Active/Active Systems

Active/active systems provide a wide range of advantages compared to active/passive systems. These advantages include the following:

- There are fewer users affected by a failure. In an active/passive architecture when the active node fails, all users are down. In an active/active system, only the users connected to the failed node are affected.
- Failover of the subset of users affected are very rapid, supporting RTOs measured in sub-seconds to seconds, because transactions only need to be rerouted and/or users reconnected to a surviving (known-working) node following a node failure.
- Failover may be routinely and safely tested since all nodes are known to be operational because they are actively processing transactions. When a failure does occur, it is to a known-working system, providing peace of mind for management. Because such testing results in a service outage, it is very difficult to accomplish for active/passive systems, resulting in failover faults which delay the restoration of service when a real outage occurs.
- Planned downtime may be eliminated by taking down one node at a time (while the other active nodes continue to provide service), performing upgrade or maintenance activities on it, and then returning it to service.
- All processing capacity of each node is available for applications. There is no idle standby system as there is with an active/passive architecture.
- Capacity may be added simply by adding nodes to the application network. There is no need to replace existing systems with larger systems. Alternatively, an existing node can be replaced with a larger or smaller node to increase or decrease the capacity of the system. The new node's database is synchronized with the application database, the application is started, and users are then rerouted to it. At this point, the old node is taken out of service. None of these changes result in a service outage.
- Load may be easily balanced by routing some transaction activity to underutilized nodes.
- Processing nodes can be located near clusters of users, providing data locality and a reduction in response time.
- A node may be located in a "lights-out" facility, since its failure does not deny users access to the applications services.
- Even if an application is not amenable to run in a fully active/active architecture for some reason, it is still beneficial to run it in a quasi-active/active environment. Such configurations are in all respects the same as a regular active/active architecture, however, all user activity is routed to only one node, which provides all transaction processing. This configuration is called a "*sizzling-hot-takeover*."[11] It resolves the application distributed-processing issues but retains all of the continuous availability features of an active/active system.

Figure 8 shows an example of an asynchronous bi-directional data replication engine. In effect, it is two uni-directional asynchronous replication engines (see Figure 5), each replicating in opposite directions. However, these two replication engines are not independent, and they are more complex than a uni-directional replication engine. Each side must cooperate to ensure that a change received via replication is not replicated back to the source of the change, a condition known as *data oscillation* or *ping-ponging*.[12]

In addition, with asynchronous bi-directional replication, it is possible that the same data item might be changed in each copy of the application database within the replication-latency interval. If this event happens, neither database knows of the conflict. Each replicates its change to the other database, and the replicated changes overwrite the original change in each database. Now both databases are different, and both are wrong; this is called a *data collision*.

[11] See the section, *Sizzling-Hot-Takeover (SZT)*.

[12] Replicated changes are updates to the database and are reflected in the target system's change queue. Since changes are replicated from the change queue, unless some protection is provided, replicated changes will be replicated back to the source system, and the process will be repeated.

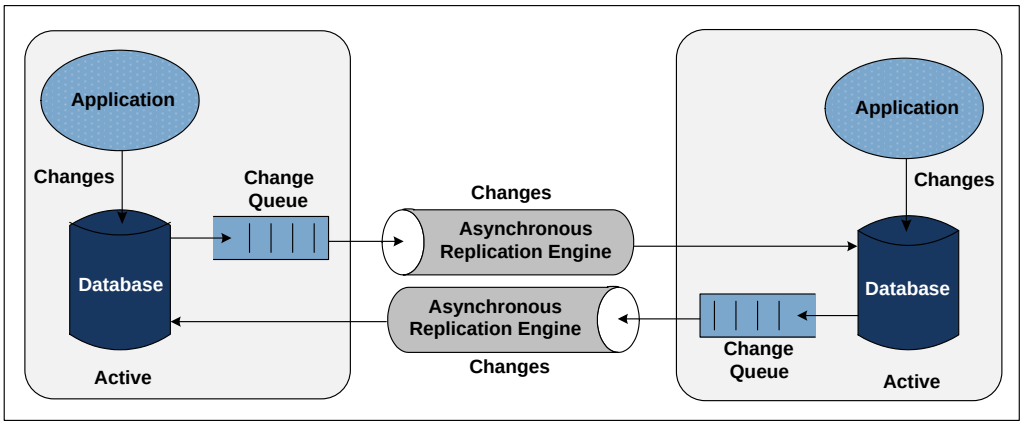


**Figure 8 – Asynchronous Bi-directional Replication Engine**

11.2.4 Data Collisions

There are some applications in which data collisions can be ignored, for instance, perhaps a temporary database divergence is not important, and the databases will be resynchronized when that data item is once again updated. Some applications avoid data collisions altogether, for instance, an application that is insert-only will not suffer data collisions.

Some active/active architectures can avoid data collisions. One frequently used method is to partition the database between the nodes. For instance, if the database can be partitioned by customer range so that only one node updates any given partition of customer data, data collisions cannot occur. In this case, the application must route all updates to the proper node that owns the customer data partition being updated.

The preferable method is for data collisions to be *avoided*; however, if data collisions are possible, they must be detected and resolved. There are several techniques for the replication engine to automatically detect and resolve data collisions. They include:

- Detection – Detection is generally accomplished by sending with the source change a row version of some sort that identifies the version of the row that the source system is changing. The row version can take many forms, such as a timestamp, a version number, or a before image of the row. If the target system finds that the row version being updated by the source system is not the same as the current row version in the target database, a data collision has occurred.

- Resolution – Once the replication engine has detected a data collision, it must make a decision as to which change to accept and which to reject. The decision rule must provide consistency so that all nodes make the same decision. Otherwise, the database copies may diverge. Many collision resolution rules exist. The selection of a resolution algorithm depends upon the application and how it processes data.

  Examples of collision-resolution rules include:

  - Choosing the update that carries the latest (or earliest) timestamp.
  - Choosing the update that was made by the node with the highest precedence.
  - Using relative replication, in which operations such as add and subtract are replicated rather than row contents. Since these operations are commutative (that is, they may be executed in any order and still arrive at the same result), data collisions do not result in database divergence.
  - Choosing the update according to customized business rules bound into the data replication engine.
  - In those cases where automatic resolution is not possible, collisions have to be resolved manually.

Synchronous bi-directional replication avoids data collisions completely since the replication engine must acquire locks on all copies of the data item across the network before or as it changes any of them; so only one application at a time can change a data item, thus avoiding collisions. A synchronous bi-directional replication engine can be implemented using two asynchronous uni-directional coordinated commit replication engines (see Figure 7), one for each direction, as shown in Figure 9.

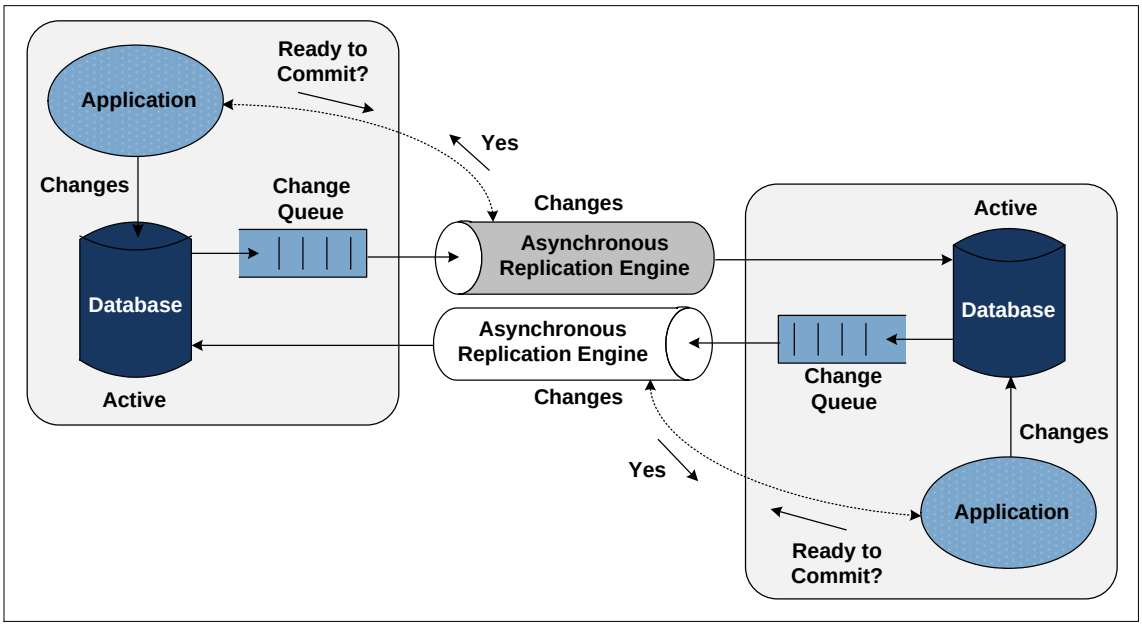


**Figure 9 – Bi-directional Synchronous Replication Engine**

## 12 Data Replication Provides Both High and Continuous Availability

Distinguishing between high availability and continuous availability, the required recovery time is a function of the criticality of an application. Recovery time measured in minutes is typically sufficient to provide *high availability* to critical applications. A system provides *continuous availability* if it recovers so quickly that no one is even aware that there has been an outage, or if users are immediately reconnected to a fully operational system. Recovery time in seconds generally qualifies for continuous availability. Data replication not only satisfies the five fundamental requirements of availability, described earlier, but it also can satisfy very small RTOs and very small or zero RPOs (i.e., it supports the full range, from high to continuous availability).

### *12.1 Redundancy*

Since the intent of data replication is to copy data from a source database to a target database, it inherently supports redundancy. There are at least two synchronized copies of the database in the application network. Also, since the replication engine can be configured to maintain the target database as a complete, current, and consistent copy of the source database, the target database can also be used for application processing. Thus, replication not only supports data redundancy, but it also supports processing redundancy.

### *12.2 Isolation*

A redundant system based on data replication ensures isolation of the redundant components. Since the two (or more) data-processing sites interact only via data replication, a problem at one site does not necessarily affect the other sites. Also, since the sites are loosely coupled and stand on their own, they do not have to be identical. They can use different hardware, different application and operating system versions, and even different database schemas.[13] It is only necessary that each site be fully operational in its own right.

[13] If this capability is important in your application, be sure that your data replication vendor supports heterogeneous replication between your systems and databases. Shadowbase software supports this capability.

### *12.3 Dispersion*

Data replication inherently supports geographic dispersion; the sites can be hundreds or thousands of miles apart. However, the farther apart the systems are, the greater the replication latency (and, in the case of synchronous replication, the application latency). Consequently, it is important to select replication architectures that minimize any latency that they add to replication or application processing.

### *12.4 Failover*

One site can automatically detect a fault in another site by monitoring the data flow in the replication channel. For instance, during idle times, the source system can send heartbeat, status, or "I'm alive" messages to the target system. If the target system detects a lack of replication or heartbeat traffic, it can initiate the failover process. If the target system is also actively processing transactions (active/active), or if it is kept in a ready state to do so (sizzling-hot-standby), failover is simply a matter of switching users from the failed site to the operational site. It is known that the surviving site is operational. There is no decision time involved – it is just done.

### *12.5 Testing*

Testing an active/passive architecture is difficult and risky and requires a service outage as failover is tested. Bringing up the backup system requires loading (if magnetic tape or virtual tape backup is used) or activating the backup database, starting the applications, switching the networks, and testing the system. It is expensive since critical personnel must be on site or on call in case something goes wrong.

Active/passive failover testing can take hours, during which the application is down. It is also risky because the failover may not work, the maintenance window may be exceeded, or the primary system may not come back up. As a result, full failover testing is often not done. Faith is often the driving factor when a failure does occur and the backup must be brought online. Management is often reluctant to make the call to fail over, thereby also lengthening the outage period.

Compared to active/passive systems, testing failover is more simplified and practically risk-free with sizzling-hot-standby (SZT) and active/active systems. In an SZT or active/active system, the backup is always operational. Testing it only requires periodically sending test or verification messages to the backup system. Alternatively, a subset of users can be periodically switched over to the known-working backup system, a process that typically can be done in seconds or sub-seconds. Thus, failover testing can be performed often and with little risk. Of course, in a fully active/active environment with all nodes processing transactions, it is always known that all nodes are performing properly. Every transaction is, in effect, a test message that proves that application processing is fully operational at that node.

### *12.6 RTO*

As discussed in the failover description above, active/active data replication allows recovery from a system fault to be accomplished in seconds or less. Since recovery is simply a matter of rerouting user or transaction requests to a surviving node in the application network[14], RTOs measured in seconds or sub-seconds can be achieved.

[14]Users can be switched using user redirection, network redirection, or server redirection. See the Gravic white paper, *Achieving Century Uptimes with HPE Shadowbase Active/Active Technology*.

### *12.7 RPO*

If asynchronous replication is used, RPOs in the order of tens or hundreds of milliseconds can be achieved. If synchronous replication is used, no data is lost following a source node failure, and an RPO of zero is achieved. However, it should be noted that whereas synchronous replication loses no data, there is a window of uncertainty following a source node failure. Since a synchronous replication engine typically safe-stores or tentatively applies the transaction changes on the target system before voting to commit, there is a brief window during which a source-node failure leaves the target transaction in doubt as to whether it was committed or aborted on the source system. Upon takeover, the target system does not know what to do with the transaction; this is a hung transaction and must be resolved manually or by business rules.[15]

## The Business Continuity Continuum

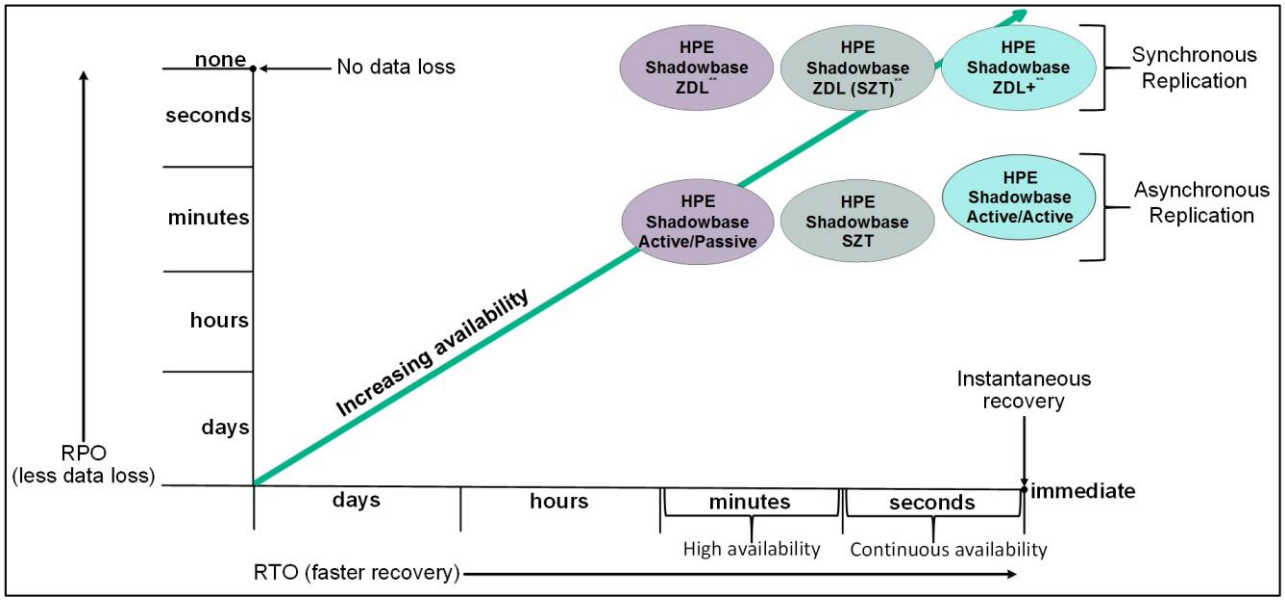


**Figure 10 – The HPE Shadowbase Business Continuity Continuum**

### *12.8 Business Continuity Summary*

The various business continuity approaches that we have described earlier are summarized in Figure 10. This diagram plots RPO versus RTO and shows where each methodology fits. The horizontal axis shows improving RTO as it trends toward zero (or unnoticeable) recovery time, and the vertical axis shows improving RPO as it trends toward zero data loss following a node failure. System continuity increases as you move up and to the right. Hence, most high availability implementations are in the upper right quadrant of the figure using one of the data replication architectures. Best practices are pushing implementations from active/passive to active/active and (assuming the appropriate technology exists) from asynchronous replication to synchronous replication.[16]

Though the figure reflects the full range of business continuity strategies, we are only really concerned in this paper with the data replication architectures that can provide RTOs measured in minutes to sub-seconds and RPOs measured in seconds to zero. They reflect the trends that are occurring in the marketplace as companies realize that even those services once deemed ancillary, such as email and internet access, need to always be available.

A broad range of techniques exist for minimizing data loss and unplanned downtime, from tape backups to active/active systems, and we briefly review them next for perspective.

[15]As a practical matter, if the target system knows that all I/O events were successfully applied to the source database, it assumes that the source system issued the commit and does the same. Coordinated commit replication engines know this information and proceed without manual intervention.

[16]Verify the range of support with your replication engine vendor. Shadowbase software supports the whole range, including synchronous replication.

### *12.9 Disaster Recovery*

Disaster recovery is the ability to *recover* from a disaster, even if it takes hours or days.

#### 12.9.3 Tape Backup

From the earliest days of commercial computing, systems have used magnetic tape as a backup medium. Operations staff take full backups followed by intermediate incremental backups taken periodically, such as every day. At the time of failure, the recovery point is the last backup; so the time between backups leads to very long RPOs – hours to days of lost data. Also, the time involved in retrieving the tapes, recovering the database, bringing up applications, and testing the resulting backup system also results in RTOs of hours to days before service is restored, assuming that you have a system to which to recover.

#### 12.9.4 Virtual Tape Backup

Virtual tape replaces magnetic tape with disk images of the database communicated to an offsite disk storage system. Because the need to handle and store large numbers of tapes is eliminated, the frequency of backups can be increased, significantly reducing RPOs to hours. When a recovery needs to be performed, the loading of the backup database is much faster, though applications must still be started and the system tested, leading to RTOs typically measured in hours or more.

Because of the lengthy requirement to procure an alternate system, load it, synchronize its database, and bring the application environment online following the loss of the primary system, magnetic tape and virtual tape approaches do not meet high or continuous availability requirements. We do not consider these technologies further in this paper. (See the sidebar on this subject, “Why Tape Backup Is Inadequate for Disaster Recovery”, for further discussion of the reasons why.)

#### 12.9.5 Clusters and Virtualization

Clusters and virtualization can lead to very good availabilities within the datacenter or campus environment (five 9s is often quoted), but these techniques apply only to collocated systems. The backup system and the storage shared between the systems typically must be in the same datacenter or otherwise located in close proximity with extensive high-speed interconnects. These technologies do not satisfy the dispersion attribute of highly available systems; they may not survive a local disaster. We do not consider these technologies further in this paper.

#### 12.9.6 Uni-directional Replication

Uni-directional data replication is the simplest form of data replication (see Figure 3 and Figure 4). Since an active node processes all transactions and replicates the database changes that it makes to a remote standby database, the two databases are in (or are nearly in) synchronization. With asynchronous uni-directional replication, only the data in the replication pipeline at the time of a source-node failure is lost, thus supporting sub-second RPOs. With synchronous uni-directional replication, no data is lost following an active node failure (RPO of zero).

Applications may be up and running in read-only mode in the standby node so that the standby database may be actively used for query and reporting purposes (we call this a *hot-standby*).[17] If the active node fails, the applications at the backup node can remount the database for read/write access and take over the role of the original active node. This process typically takes only a few minutes, leading to RTOs measured in minutes. Thus, uni-directional architectures provide high availability – RTOs measured in minutes and RPOs measured in sub-seconds (or zero if synchronous replication is used).[18]

[17] This process is possible, provided the data replication engine maintains transactional consistency between the source and target databases during normal operations, which is not the case for all data replication products. (It is true for Shadowbase software.)

[18] In some cases, the application on the passive target node already has the target database open for read/write access, thereby improving recovery time. We call this a *sizzling-hot-takeover (SZT)* system.

## *Note: Why Tape Backup is Inadequate for Disaster Recovery*

Disaster recovery using tape backup technologies involves creating production data backups on a storage medium of tape or virtual tape, restoring the data to disk on a backup system, and then rolling forward the audit trail (updates performed since the backup was taken), to bring the data up-to-date.

This process might be perfectly reasonable for some less important applications, but is a totally inadequate approach to business continuity for mission-critical applications (and with the "always on" requirements of today, more and more of a company's IT systems fall into this category). Yet, many organizations continue to use this outmoded technology.

So why is tape backup/restore such a problematic technique to use for business continuity? Here are a few of the more significant reasons:

- Locating a backup/standby system or datacenter may be difficult, assuming you don't already have one and need to go and declare an "event" at your recovery provider's datacenter.
    - Don't be last in line to declare your "event" either, as many DR service providers take customers on a first-come, first-served basis. Delay and you might be stuck behind someone else's use of the datacenter/DR equipment, thereby prolonging the outage.
- Downtime might be lengthy (higher Recovery Time Objective, or RTO) due to the time taken to find the right backup data, restore that data, roll-forward through any interim changes, load and then restart the application. This sequence often takes many hours or even days.
- Significant amounts of data are likely to be lost (higher Recovery Point Objective, or RPO). Data loss is governed by the frequency of taking and sending the backups and interim changes off site, hence a frequent safe-store period is required else significant data loss will occur.
- The recovery process has a high probability of failure, often referred to as a "failover fault." Many things can go wrong or not occur within specified Service Level Agreement (SLA) times, therefore making this type of business continuity solution far more risky than the other forms based on online data replication technology (as discussed in this paper).

Add the cost and inconvenience of periodically testing the failover process – assuming this is even done, since budget cuts have decimated DR testing budgets – and this approach simply cannot be trusted for many (most) applications.

All of these issues can lead to lengthy outages, often measured in ***days*** of downtime. In today's always-on world, a mission-critical application being down for days is simply unacceptable. Companies have gone out of business for less. Given the high value of some data, losing any of it is to be avoided.

However, there are solutions available ***today*** for disaster recovery and higher levels of business continuity protection which are comparatively easy to implement, relatively inexpensive when compared to the ***total cost*** of downtime, and which do not suffer from these egregious shortcomings. Figure 10 shows these various technologies and their relationship to the duration of the outage (RTO), and the amount of data loss (RPO) that each approach may cause to occur.

What is clear is that online data replication is a far superior solution for ensuring business continuity of mission-critical applications than tape backup/restore methods, as it reduces outage times and amounts of data lost from days to seconds or sub-seconds, or even to zero in the case of active/active synchronous replication.

The takeaway is that tape/virtual tape backup/restore techniques for disaster recovery are insufficient for mission-critical applications, and come with significant and unnecessary risks which will likely result in significant recovery times and amounts of lost data. Organizations that still use such methods should immediately put these risks behind them, and consider instead one of the online data replication solutions discussed in this paper, and begin the move to continuous availability as soon as possible.

### *12.10 Disaster Tolerance*

The above systems meet the needs of disaster recovery but not necessarily disaster tolerance, because their recovery times are measured in minutes or more. If users are down for several minutes, or even worse, hours or days, the outage has certainly affected them. To be *disaster tolerant*, recovery must be so fast that users are unaware of the outage or are immediately reconnected to a fully operational system. Recovery times measured in seconds or less with little or no data loss qualify a system to be called disaster tolerant. Disaster tolerance requires that there be a backup node that can take over in sub-seconds or seconds in the event of an active node failure. Below are two data replication configurations that satisfy this requirement.

12.10.3 Sizzling-Hot-Takeover (SZT)

A *sizzling-hot-takeover (SZT)* architecture is similar to an active/passive configuration using uni-directional replication, as described earlier, except that the passive system is immediately ready to start processing transactions (Figure 11). In this configuration, the passive system has its applications running and the local copy of the application database already open for read/write access. It is in all respects an active system, with the exception that all user transactions are directed to the primary active node (thereby avoiding data collision issues). Using data replication, the SZT system can very quickly take over processing because its local database is synchronized with the active database and is completely consistent and accurate, and the applications are already up and running with the database open for read/write access.

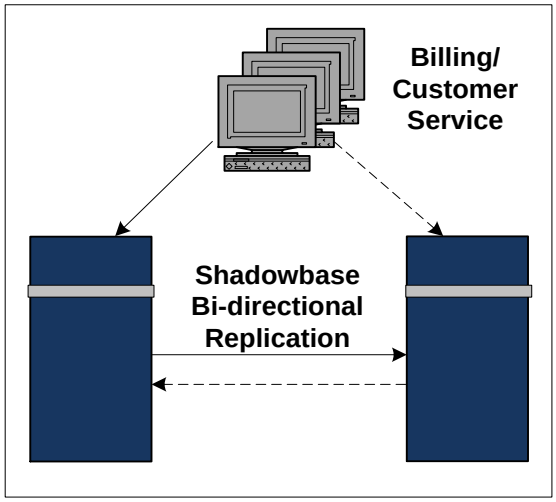


**Figure 11 – Sizzling-Hot-Takeover (SZT)**

If the active node fails, all that is required for failover is to switch the users or their transactions to the standby node. The switch can be done in seconds to sub-seconds, leading to very small RTOs. To operate in this mode, it is essential that the replication engine that is used allows the application processes to also open the target database for read/write access.

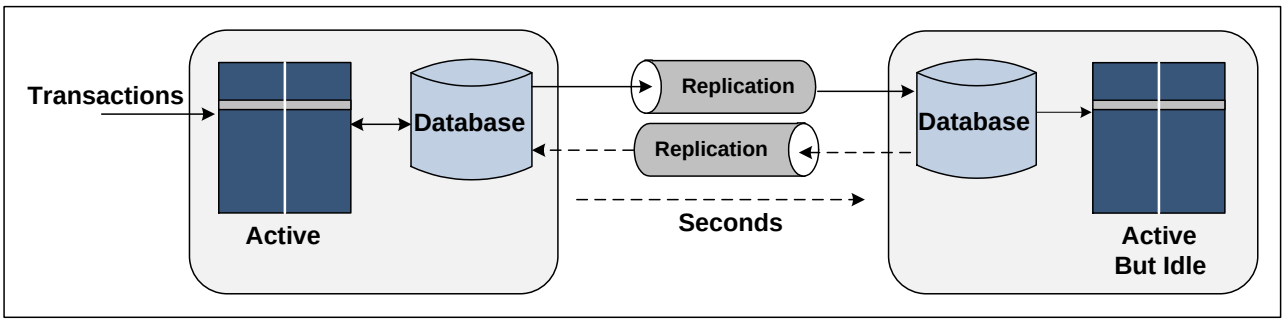


**Figure 12 – Sizzling-Hot-Takeover with Optional Reverse Replication**

The reason that an SZT configuration has a much better RTO than the active/passive disaster recovery systems described earlier, is the absence of *failover faults*. In active/passive systems, the standby system is not actively involved in the application. Consequently, you do not know whether it is really operational.

Furthermore, for the reasons discussed earlier, the standby system and failover procedures are unlikely to have been recently fully tested. Consequently, active/passive systems suffer from a high probability of failover faults, and when a failover attempt is made to a nonfunctioning backup system, the application is down. However, in an SZT configuration you know that the backup node is working correctly *without requiring an outage of the primary system* since it can be easily exercised by periodically submitting test or verification transactions to the application to ensure proper operation. Consequently, failover faults will not arise and failover can be automated, which is a requirement if very short RTOs are to be satisfied.

Finally, the SZT system can be optionally configured with reverse replication up and running so that it has a backup as soon as the former primary node is recovered (Figure 12). With reverse replication enabled, the standby queues the changes that it is making to its copy of the database so that the failed node can be resynchronized upon recovery. This configuration can achieve a zero RPO if synchronous replication is used or RPOs measured in tens or hundreds of milliseconds if asynchronous replication is used.

### 12.10.4 Active/Active

An active/active configuration takes the SZT arrangement one step further. Rather than routing all transactions to one node as in the SZT case, all nodes in an active/active network may be simultaneously processing transactions for the same application using their local copies of the application database (Figure 13). Bi-directional data replication is configured between each node pair so that any change that an application makes

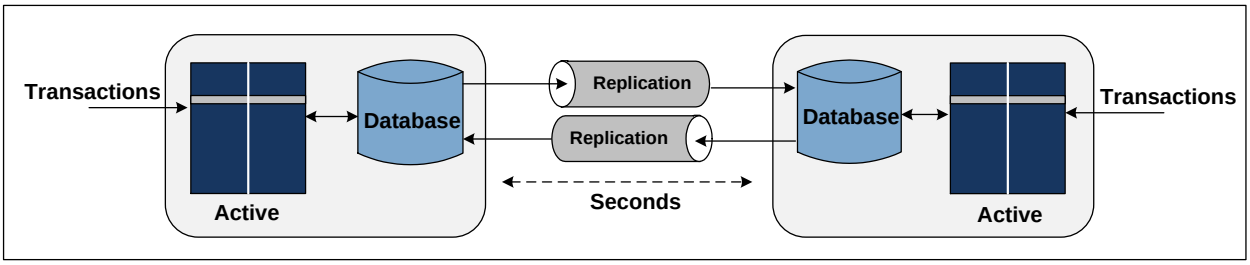


**Figure 13 – Active/Active Replication**

to its local copy of the database is immediately replicated to the other nodes in the application network. Thus, all nodes (and thus, applications) have the same view of the application database. One issue with active/active systems using asynchronous replication is that of data collisions, as described previously. However, data collisions do not occur if synchronous replication is used because the locking of data items ensures that only one change is made to all data copies at any one time. There are also other methods of avoiding data collisions as discussed in Section 12.2.2.

## 13 Eliminating Planned Downtime

The above discussions have focused on eliminating *unplanned downtime*. But what about *planned downtime*? After all, planned downtime occurs more frequently as systems are upgraded, applications are enhanced, database structures are changed, and so on.

The answer is simple. Using either of the continuously available, disaster-tolerant architectures previously described – SZT or active/active – a node to be serviced can easily and safely be taken out of service by moving its users to other nodes, in seconds. The node is then upgraded, resynchronized, and returned to operation, continuing once again to provide application services to its users. Upgrades are rolled through the application network in this manner.[19]

The procedure for avoiding planned downtime is as follows. Traffic routed to the node to be upgraded is rerouted to other nodes in the application network. This rerouting is done with no interruption to user activity since the other nodes are already processing transactions (or known to be working in the case of an SZT architecture).

The idled node is then upgraded and tested. Testing takes as long as necessary since users are being serviced by other nodes. When testing is complete, the upgraded node is returned to service, which requires first loading

[19]For more information, see the Gravic white paper, *Using HPE Shadowbase Software to Eliminate Planned Downtime via Zero Downtime Migrations*.

the current copy of the application database onto the upgraded node. The stale copy of the database on the upgraded node is brought into a current state by sending to it from an active node the changes that accumulated while the upgraded node was down. Alternatively, if too many changes have accumulated to make this step feasible, the current copy of the application database is downloaded from a currently active system using an online copy utility that does not require stopping the active system.

Once the application database is current, the upgraded node is returned to service, and users are migrated to it incrementally to ensure that it is operational before assuming its full transaction-processing load. If a problem surfaces, user traffic is again rerouted to other nodes and the node is taken out of service and repaired following the above procedure. When it finally becomes fully operational, the upgrade procedure is rolled through the other nodes in the same fashion.

One concern is that of a two-node system. While one node is being upgraded, the system is running with a single point of failure – the one operational node. For this reason, it is a good practice to make a backup copy of the database in case the single operational node fails. In systems with three or more nodes, this concern is less important since multiple nodes are operational while a node is being upgraded.

A highly available system using uni-directional active/passive replication is upgraded in the same way, except that users may be down for minutes or longer instead of seconds as the backup system is brought into operation.

# 14 Choosing a Replication Method

### *14.1 The Replication Quadrant*

In our discussion of highly and continuously available systems, we reviewed asynchronous and synchronous replication in uni-directional and bi-directional configurations. These systems give us four distinct combinations to consider, their differing characteristics and applications are reflected in the Replication Quadrant, shown in Figure 14. In this figure, there are typical RPOs and RTOs (Figure 14a), typical configurations (Figure 14b), and typical applications (Figure 14c) for each of the four combinations.

Representative characteristics for systems implemented with each of these combinations are as follows:

- A uni-directional asynchronous active/passive system has an RPO measured in seconds (the replication latency of the data replication channel). Its RTO is measured in minutes or longer as applications are started following a failure of the active node, the databases are mounted, and the network is reconfigured. Additional recovery time is typically required for the management decision time to fail over to the backup system and for testing to ensure that the backup is performing properly.

  This replication method is used for classic disaster-recovery, active/passive configurations. It supports applications that must be highly available but for which some small data loss is tolerable.[20] CRM (customer relationship management) and HR (human resources) corporate applications are examples of this class of application, as are ATM transactions. ATM transactions have a low value, and if the ATM machine is down, the customer could go to a different ATM machine serviced by a different bank.

- A uni-directional synchronous system has the same RTO characteristics as the uni-directional asynchronous system. However, it suffers no data loss following a node failure (its RPO is zero). Consequently, it is often referred to as a *zero data loss system* (ZDL).

  It is suitable for applications that require high availability while processing high-value transactions that cannot be lost. Back-office applications for brokerage firms and banks as well as funds-transfer applications are examples. With increasing automation, the health care industry is heading in this direction also.

- A bi-directional asynchronous system has an RPO measured in seconds (the replication latency) and can recover in seconds or sub-seconds in an active/active or SZT configuration.

[20] Actually, there are many systems today that use asynchronous replication in production and that process high-value transactions. In these systems, there must be a way to recover lost transactions, such as manual reentry from printed reports.

It is suitable for applications that require continuous availability but for which some small data loss is acceptable. Typical applications include telco applications (many call-related transactions worth pennies). Point-of-sale (POS) transactions are another example. Like ATM transactions, they generally have low value. However, if a POS application goes down, retailers cannot service customers using credit or debit cards.

- A bi-directional synchronous system is the ultimate in system availability. It suffers no data loss following a node failure (RPO = 0), and it can recover in seconds or sub-seconds in an active/active or SZT configuration. In fact, if the failover is sufficiently fast so that users don't realize that there has been a failure, in effect, there has been no failure.[21] An RTO of zero has effectively been achieved.

  This configuration supports applications that must be continuously available and in which transaction value is high. Typical applications with these characteristics include online financial applications such as online banking and stock market trading as well as some 911 applications. Online health care is another application that is moving towards this configuration.

### *14.2 Risk Assessment*

In order to make effective use of the Replication Quadrant, a company must know the availability and data loss requirements for each application. In an organization, some applications are typically mission-critical or even safety-critical and may require continuous availability. Business-critical and task-critical applications may require high availability. Some applications may be non-critical and can be down for hours or days without causing any great problem. Even if an application is not required to be continuously available, it may still have a requirement for minimal data loss. Hence both RTO and RPO characteristics must be considered in order to arrive at the correct solution for each application or system.[22]

The critical ranking of applications may change over time. A good example of this ranking is email. A few years ago, you could live without email for a few days. Today, however, email often forms the communication backbone between a company and its employees and customers. Company operations may grind to a halt if the email system is lost. In many companies, email has become a mission-critical or business-critical application that should have continuous availability.

Risk assessment is one of the steps in generating a proper business continuity plan (BCP), and the procedures for risk assessment are well-documented in references describing how to create a proper BCP. The result of the risk assessment should include the costs of system downtime and of lost transactions. Because of the constantly evolving nature of business, the BCP should be frequently re-evaluated to ensure it still meets the needs of the business (this re-evaluation is as important as frequent test execution of the BCP).

### *14.3 Cost Factors*

Along with the risk assessment, there should be an analysis of cost factors for the various architectures. Additional costs may include:[23]

- duplicate systems
- redundant networks
- bi-directional replication engine
- additional maintenance costs
- additional software licenses
- application modifications
- multiple sites
- additional staffing
- distributed management tools
- system testing

[21]Note also that for those users connected to other systems in the application network in an active/active configuration, no action is needed, and they see no outage at all.

[22]Sometimes certain systems are purposed specifically for the most important applications, which makes the process simpler since each application does not have to be considered separately.

[23]For more information on TCO, see the Gravic article, *Why Your Business Continuity Plan May be Inadequate*.

### *14.4 Cost/Benefit Analysis*

Knowing and protecting against the costs of downtime and of data loss, you can calculate the required RTO and RPO. For example, a large brokerage firm finds that its average trade is about $25,000. If the firm earns an average of 2% commission on each trade, a trade is worth $500 in revenues to the firm. If it is doing 10 transactions per second, its cost of downtime is $5,000 per second or $300,000 per minute. Its order-processing system currently runs on a UNIX cluster, which the firm estimates will be down about five minutes per year. The firm's total cost of downtime totals $1,500,000 per year.

The firm has determined that it will cost $1,350,000 to add enough hardware, software, and networking to eliminate all single points of failure. It wants to get a return on its investment in one year. An RTO of 30 seconds for the new system will save 4.5 minutes of downtime per year, yielding the required $1,350,000 savings. Since the brokerage cannot lose any transactions without being in violation of security regulations, it needs an RPO of zero. From the Replication Quadrant of Figure 14, it is clear that the firm should seriously consider an active/active configuration using synchronous replication.

In other cases, it is not the monetary cost of downtime that influences the choice of an availability architecture but rather a required level of service. Consider the Home Location Registers (HLR) that mobile telecommunication providers use to track mobile users and to place calls. An HLR typically handles 10,000 customers. If an HLR fails, none of the mobile users that it is servicing can make or receive mobile calls.

During a busy period, an HLR may be placing 1,000 calls per minute. If each call has a value of $1.00, the cost associated with the downtime of an HLR is $1,000 per minute. If an HLR is down for five minutes each year, the mobile provider would lose $5,000 per year due to the downtime of an HLR. This situation most likely would not justify the cost of an active/active system. However, the market and regulatory requirements for uninterrupted mobile service even in the face of a disaster argue strongly for the continuous availability of active/active systems.[24]

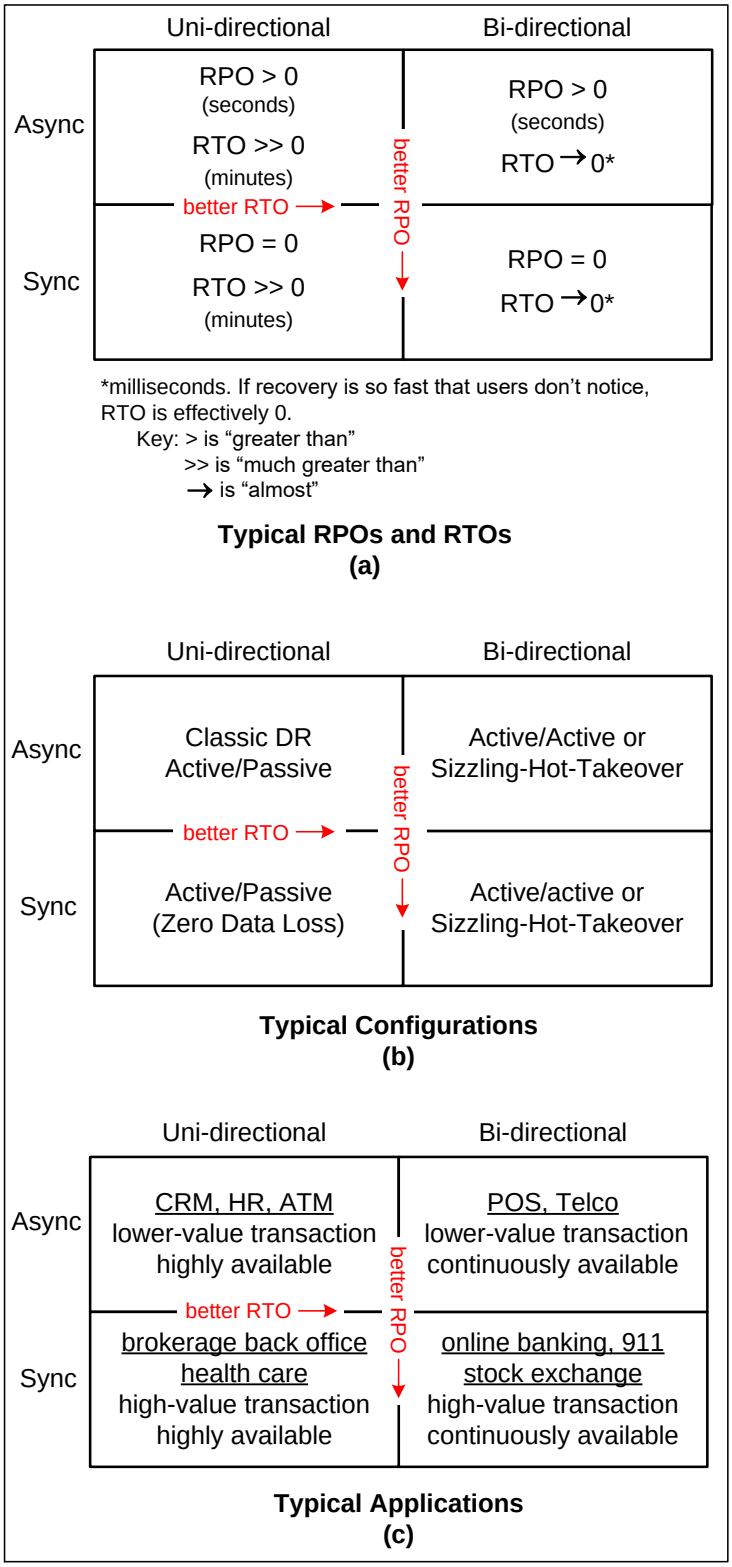


**Figure 14 – Choosing a Replication Method**

[24] See the Gravic case study, *Telecom Italia's Active/Active Mobile Services*.

Another factor, which is difficult to estimate, but which must also be considered, is the collateral damage arising from a prolonged outage or lost data. It is not just the immediate cost of lost business, but also the longer term impact on brand image and customer loyalty. Consider the mobile phone case discussed above; even though the immediate cost to the company is only about $5000/year, how many customers would it lose to competitors if such an incident occurred?

These two examples of costs and benefits illustrate an important point, it's not the average cost of a transaction that's most important when performing the analysis, it's the most *significant* potential cost. In the case of the brokerage firm the potential savings justified the expense, but more important is the fact that they cannot afford to lose any trades, both from a business and regulatory compliance point of view. Likewise for the telecommunications provider, even though their simple cost analysis would conclude a continuously available system is not justified, uninterrupted service is required from a business and regulatory point of view. It is very hard to calculate the side-effects of outages to a business, but in the highly competitive nature of today's "always on" global economy, it won't take many service interruptions before customers take their business elsewhere (hence the potential loss is huge, even if an average transaction is only worth $1). Consequently, even though high and continuous availability solutions may be more expensive to implement, when the entire gamut of potential costs of an outage are considered (business and regulatory), by significantly mitigating these costs, they actually have a lower TCO compared with lesser availability solutions (as illustrated by Figure 15).

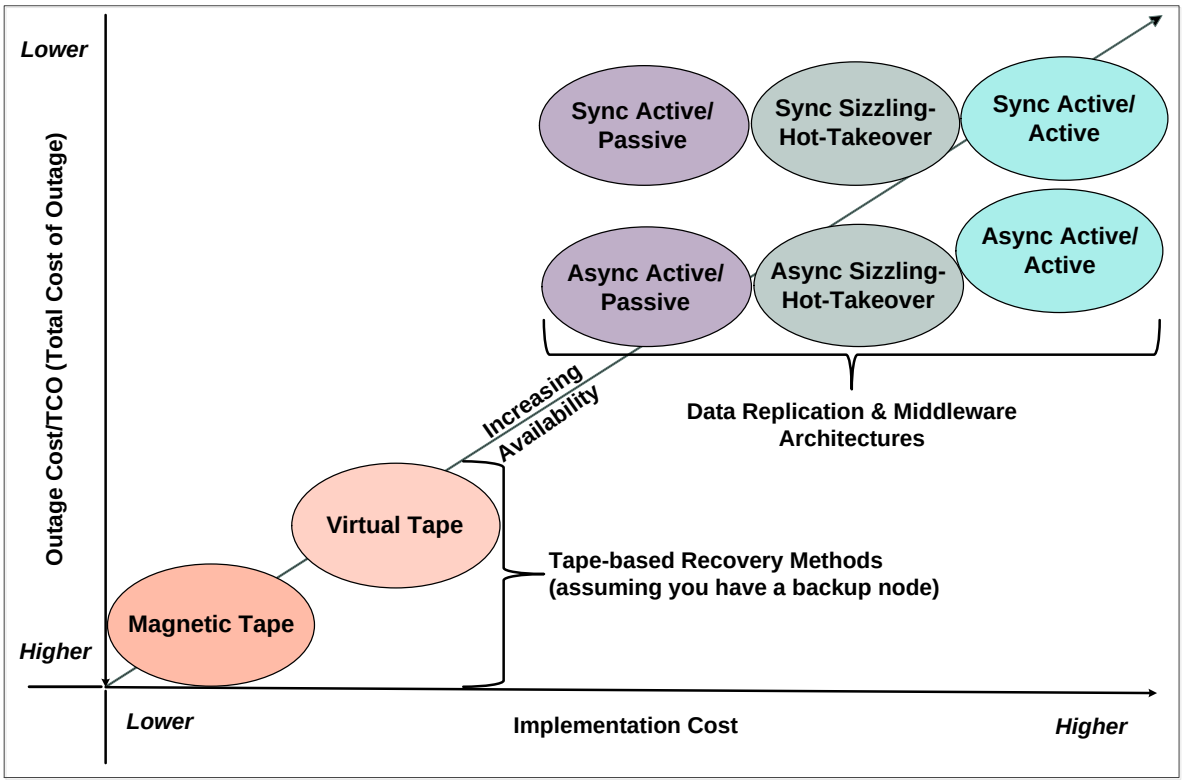


**Figure 15 – The Business Continuity Technology Cost vs TCO Continuum**

## 15 The HPE Shadowbase Data Replication Product Suite

This paper has discussed in theory the various software data replication architectures necessary to support high and continuously available applications, and to minimize data loss. The Shadowbase data replication product suite from Gravic, Inc. can actually deliver these requirements, enabling you to meet your business continuity goals. Shadowbase products support uni-directional and bi-directional data replication between homogeneous and heterogeneous systems. Shadowbase software currently supports both asynchronous and synchronous replication.[25] Shadowbase replication products have been successfully implemented for several decades in many high-demand, mission-critical installations.

[25] Contact Gravic for more details of its synchronous replication products.

HPE Shadowbase replication products include the following:

- ***HPE Shadowbase*** – a low replication latency asynchronous replication engine using efficient process-to-process data replication; it supports both uni-directional and bi-directional data replication.
- ***HPE Shadowbase Zero Data Loss (ZDL)*** – is a synchronous data replication engine using coordinated commits to eliminate data loss. ***Shadowbase ZDL*** minimizes application latency by safe-storing rather than applying replicated data during the source transaction. However, data collisions are still possible in active/active configurations for some applications.
- ***HPE Shadowbase ZDL+*** – a future technology, is a synchronous data replication engine using *coordinated commits* similar to Shadowbase ZDL. As well as eliminating data loss, Shadowbase ZDL+ avoids data collisions that can occur in asynchronous active/active replication environments.

The Shadowbase product suite supports high and continuous availability solutions, with low to zero data loss, covering the upper-right quadrant of the RTO/RPO Business Continuity Continuum depicted in Figure 16. Consequently, Shadowbase products are able to meet the business continuity requirements of everything up to and including your most mission-critical, high-value applications.

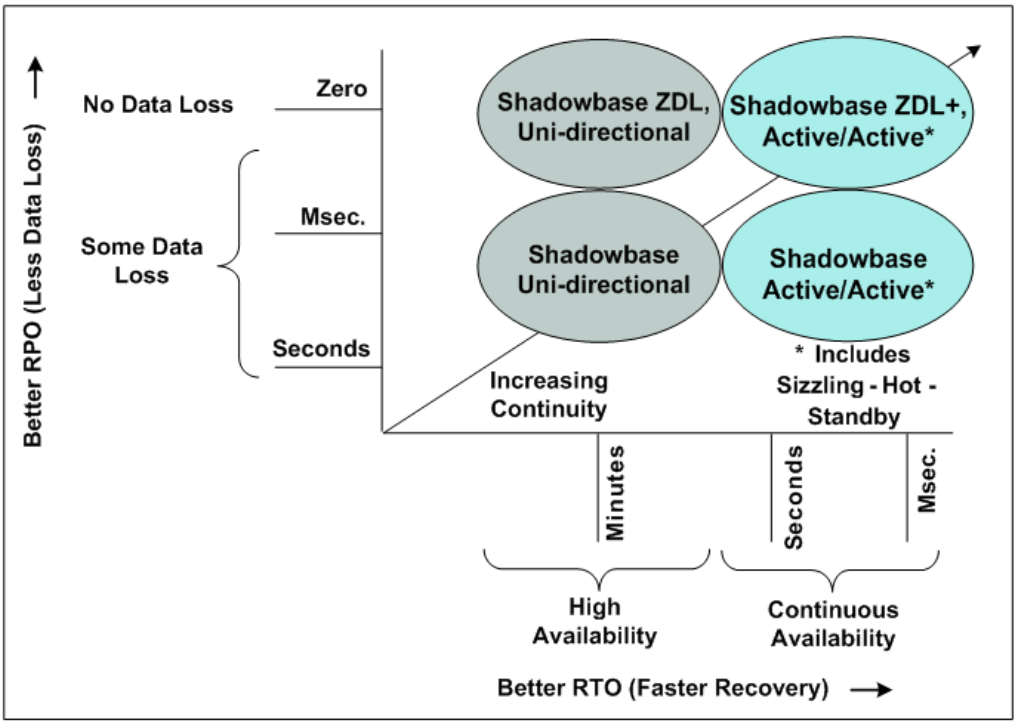


**Figure 16 – The HPE Shadowbase Business Continuity Continuum**

### *15.1 HPE Shadowbase Asynchronous Data Replication Engine*

The HPE Shadowbase asynchronous data replication engine uses process-to-process transactional replication to move data from a source database to a target database as shown in Figure 17. The Shadowbase Collector process reads new database changes from the source system's change queue and sends them over a communication channel to its Consumer process on the target system. The Consumer applies the changes to the target database.

The HPE Shadowbase replication engine can filter changes that do not need to be replicated and can transform data formats to be compatible with a heterogeneous target system. Since Shadowbase replication introduces no disk-queuing points, replication is highly efficient; and replication latency is minimized, leading to a very low RPO.

The uni-directional HPE Shadowbase asynchronous replication engine is useful to maintain synchronization of a backup database with its primary database in an active/passive architecture. In this case, the backup system is typically idle as far as update processing is concerned. However, it could be used for query

processing since Shadowbase replication provides for the target database to be a consistent copy of the source database, though delayed by the replication latency.

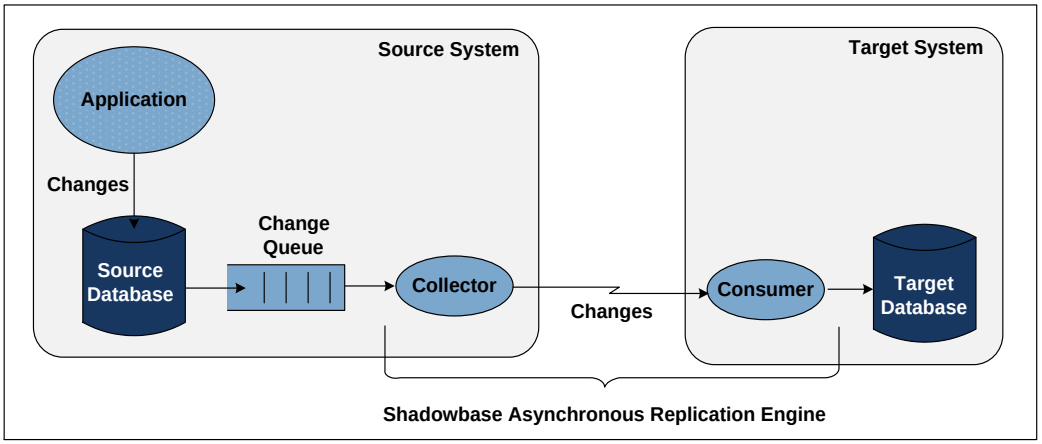


**Figure 17 – HPE Shadowbase Asynchronous Data Replication**

The backup database provides *disaster recovery* (or *high availability*). If the primary system fails, the backup system has a current copy of the database (within a fraction of a second) and is ready to take over the application functions that had been provided by the primary system. Failover to the backup system can be accomplished in minutes if it is a *hot-standby*. That is, the backup system has the applications loaded and ready so that application processing could continue as soon as the applications mount the database volumes and the system is tested.

By extending this architecture to bi-directional data replication, both nodes can be actively processing transactions in an active/active configuration since each has an up-to-date copy of the application. Shadowbase technology accomplishes bi-directional replication by using two replication engines, one in each direction, that share information about the data being replicated to avoid data ping-ponging.[26]

An active/active configuration provides *disaster tolerance* (or *continuous availability*), because if a system fails, further transactions are immediately rerouted to surviving active nodes in the application network. Recovery can be accomplished in seconds, often so quickly that no one notices that there has been an outage, achieving an RTO approaching 0.

As described earlier, data collisions are a serious problem encountered with bi-directional asynchronous replication. The HPE Shadowbase asynchronous data replication engine supports collision-avoidance methods. If collisions cannot be avoided, the Shadowbase engine provides the mechanisms necessary for detecting and resolving data collisions by embedding application logic into the replication engine.

Of course, if Shadowbase bi-directional capabilities may also be used to implement a sizzling-hot-takeover (SZT) system, where the high availability and disaster tolerance provided by active/active systems is achieved without the problem of data collisions. In this configuration, only one system is processing transactions, but the backup system is able to process transactions and is ready for immediate takeover. (It is a known-working system.)

### *15.2 HPE Shadowbase Zero Data Loss (ZDL)*

*HPE Shadowbase ZDL* is based on Gravic's patented coordinated-commit technology,[27] which uses asynchronous replication of database transactions coupled with synchronous commit. It extensively leverages the existing Shadowbase asynchronous architecture to replicate data. In addition, it joins the transaction at the source system and votes on the outcome of the transaction. If at commit time it has successfully applied all of the transaction updates to the target database, it votes to commit. Otherwise, it votes to abort. (Figure 18). Rather than directly applying the changes to the target database, Shadowbase ZDL safe-stores them in a target-side persistent queue file (or optionally, in-memory) and then applies them to the target database in the

[26] As described earlier, "ping-ponging" or data oscillation is the re-replication of replicated data back to the source database. See the section titled Bi-directional Replication and Active/Active Systems.

[27] B. D. Holenstein, P. J. Holenstein, W. H. Highleyman, "*Asynchronous coordinated commit replication and dual write with replication transmission and locking of target database on updates only*," *U.S. Patent 7,177,866*; February 13, 2007.

background. At commit time, Shadowbase ZDL votes to commit if it has successfully safe-stored all changes in its queue file, whether or not they have been applied to the target database.

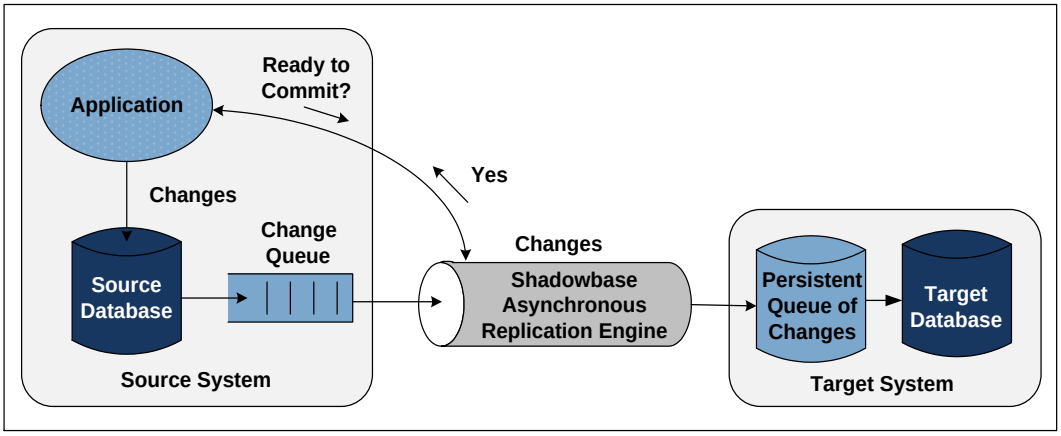


**Figure 18 – HPE Shadowbase ZDL Synchronous Replication Engine**

Consequently, application latency is reduced since it is much faster to queue changes than to apply them to the target database. However, since changes are applied asynchronously, applications in two different nodes in an active/active architecture can modify the same data item at the same time, thus leading to data collisions. Thus, Shadowbase ZDL provides zero data loss with minimal application latency, but data collisions are still possible in an active/active architecture.

If the target database is taken offline, Shadowbase ZDL can also be used with the HPE Shadowbase asynchronous replication engine to provide a synchronous safe-store on the target system to queue changes. Shadowbase ZDL supports both uni-directional active/passive architectures and bi-directional active/active and SZT architectures.

In summary, the major benefits of Shadowbase ZDL are:

- In uni-directional and bi-directional replication modes, zero data loss (RPO = 0)
- Much lower application latencies arising from use of the patented coordinated commit technology compared to other forms of synchronous replication (such as dual writes)
- Minimal application latency by safe-storing of changes in queue files or in-memory
- Relaxing of distance separation limitations arising from use of the patented coordinated commit technology compared to other forms of synchronous replication (such as dual writes)

### *15.3 HPE Shadowbase ZDL+ Synchronous Replication*

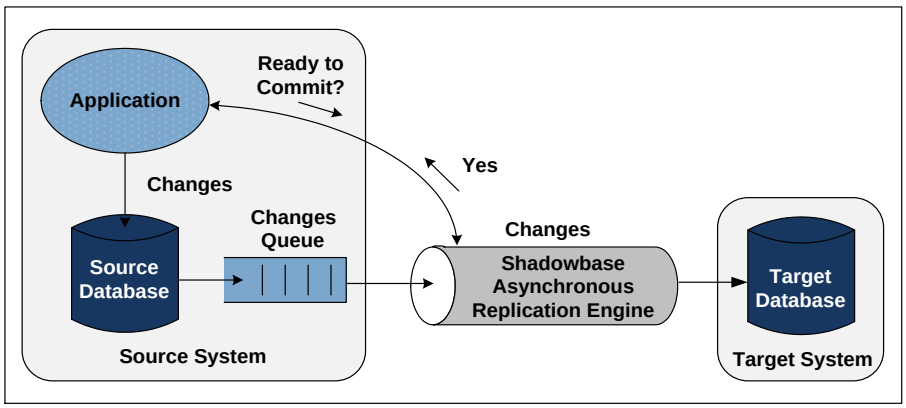


**Figure 19 – HPE Shadowbase ZDL+ Synchronous Replication Engine**

*HPE Shadowbase ZDL+* (a future technology) is an extension of Shadowbase ZDL. It also joins the transaction at the source system and uses the HPE Shadowbase asynchronous replication engine to replicate data (Figure 19). Unlike Shadowbase ZDL, Shadowbase ZDL+ applies changes to the target database in the context of the source application's transaction, eliminating the possibility of data collisions in an active/active environment.

Because it is typically slower to apply changes to the target database than to safe-store them in a queue file (or in-memory), application latency may be increased when using Shadowbase ZDL+. Thus, Shadowbase ZDL+ ensures that all data has been successfully stored on a backup system (so no data loss is possible, just as with Shadowbase ZDL), plus it eliminates the possibility of data collisions in an active/active architecture.

Shadowbase ZDL+ continues the best-in-class features of HPE Shadowbase technology, such as low replication latency, robustness, and communication efficiency. It supports both uni-directional active/passive architectures and bi-directional active/active and SZT architectures.

In summary, the major benefits of Shadowbase ZDL+ are:

- In uni-directional and bi-directional replication modes, zero data loss (RPO = 0)
- In bi-directional replication mode, no data collisions (consequently removes the requirement to avoid data collisions by partitioning applications and/or data, or resolve them if they do occur, neither of which may be possible for some applications, thereby enabling the use of an active/active architecture for such applications which would otherwise have been precluded)
- Lower application latencies arising from use of the patented coordinated commit technology compared to other forms of synchronous replication (such as dual writes)
- Relaxing of distance separation limitations arising from use of the patented coordinated commit technology compared to other forms of synchronous replication (such as dual writes)

## 16 Summary

The technology exists today to achieve inconsequential, fast recovery times following a system outage with zero loss of data. The key to this technology is data replication. Data replication comes in several forms – asynchronous or synchronous, uni-directional or bi-directional. Each combination supports different ranges of recovery times (RTOs) and data loss (RPOs). By understanding the costs of downtime and data loss for each application and the costs of achieving various levels of high availability and continuous availability, IT management can make informed decisions concerning the availability approach that is right for each application. The HPE Shadowbase suite of replication products provides the full range of replication technologies to satisfy the most demanding IT 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.