

Breaking the Availability Barrier I
Survivable Systems for Enterprise Computing

Volume 1 of 3 Volume Series
Dr. Bruce Holenstein, Dr. Bill Highleyman, and Paul J.
Holenstein

© 2007 Gravic, Inc. All rights reserved.

No part of this book may be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the authors.

ISBN: 978-1-4107-9231-0 (e-book)

ISBN: 978-1-4107-9232-7 (Paperback)

ISBN: 978-1-4107-9233-4 (Dust Jacket)

Library of Congress Control Number: 2003099708

All products mentioned in this book are trademarks of their respective owners. The information in this book is provided on an as-is informational basis. The authors, owners, and publisher disclaim liability for any errors or omissions. The reader accepts all risks associated with the use of the contents of this book.

About the Authors:

The authors of the book, Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein, have a combined experience of over 90 years in the implementation of fault-tolerant, highly available computing systems. This experience ranges from the early days of custom redundant systems to today's fault-tolerant offerings from HP (NonStop) and Stratus.

Click for Book Order Information: [Breaking the Availability Barrier](#)
or visit Amazon.com or Authorhouse.com to purchase.

Chapter 6 - RPO and RTO

In the previous chapters, we have discussed how system availability can be significantly enhanced by replicating a system or by breaking it up into several smaller independent cooperating nodes. We have discussed to some extent the various failure modes of distributed systems, the immediate recovery from these failures, and the ultimate restoration of the full system to service following the repair and recovery of a failed node.

This chapter explores two important questions relative to failures in a distributed system that must be considered when deciding how to replicate or split a system:

- How much data can be lost due to a node or network failure? This is called the Recovery Point Objective, or RPO.
- How much time can be lost due to a node or a network failure? This is called the Recovery Time Objective, or RTO.

Replicating the System

Fundamental to the achievement of high availability in distributed systems is the provision of one or more similar nodes, either locally or at remote sites, that can provide the application services to all users in the event that a node in the distributed system becomes unavailable for any reason. If nodes are geographically dispersed, then the system is protected also from natural or man-made disasters.

Of course, replicating a system involves much more than just the provision of computing capacity at other sites. It requires adequate

Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein

staffing at all sites, the network facilities to switch remote users from one site to another, the transfer of the current state of the application databases, and solid procedures for the recovery process.

In this chapter, we focus on methods for making up-to-date copies of the application database available at multiple sites. We will explore different methods for doing so and will discuss the recovery process for bringing up a consistent database at a surviving node following the failure of another node so that application services can be restored to all users who had been serviced by the failed node. Specifically, we will evaluate the time it takes to recover from a node failure and the risk of data loss caused by such a failure.

RTO and RPO

When considering the true and total costs of a node failure, two important factors are the time that the users of the system are denied service and the amount of data, if any, that may have been lost due to the failure. Of course, associated with a recovery are many other costs, such as the movement of personnel from the failed site to the backup site. However, we will focus here on the issues of denied service and lost data.

When deciding how nodes in a distributed system will back each other up, the amount of tolerance that a business has to recovery time and lost data should be established as a pair of goals, or objectives, that the system must achieve in the event of a primary system loss. This tolerance can be established as Recovery Point and Recovery Time Objectives²⁸:

- The Recovery Point Objective, or *RPO*, is a measure of how much data loss due to a node failure is acceptable to the business. A large RPO means that the business can tolerate a great deal of lost data.

²⁸ LaPedis, R.; “*Will Enterprise Storage Replace NonStop RDF*,” The Connection, Volume 23, Issue 6; November/December, 2002.

- The Recovery Time Objective, or *RTO*, is a measure of the users' tolerance to down time. A large *RTO* means that users can tolerate extensive down time.

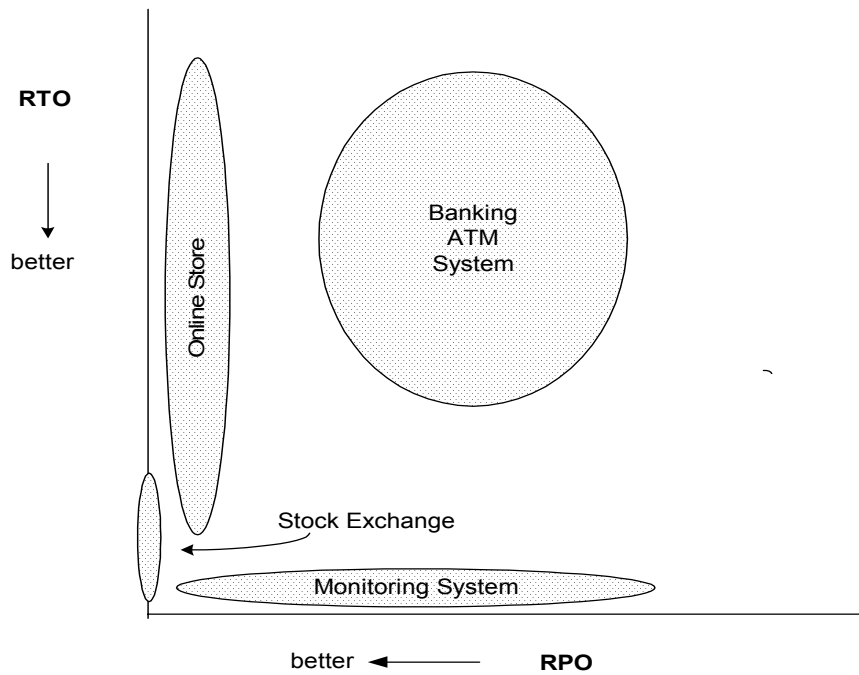
These two objectives are not closely related – they may both be almost zero, they both may be large, or one may be small but the other large.²⁹ Various examples of the needs of different applications are shown in Figure 6-1. For instance, a stock exchange trading system must be brought back to life very quickly and can lose no data. Since the price of the next trade depends upon the previous trade, the loss of a trade will make all subsequent transactions wrong. In this case, the *RTO* may be measured as a few minutes or less, but the *RPO* must be zero.

On the other hand, a critical monitoring system such as those used by power grids, nuclear facilities, or hospitals for monitoring patients must have a very small *RTO*, but the *RPO* may be large. In these systems, monitoring must be as continuous as possible; but the data collected becomes stale very quickly. Thus, if data is lost during an outage (large *RPO*), this perhaps impacts historical trends; but no critical functions are lost. However, an outage must end as quickly as possible so that critical monitoring can continue. Therefore, a very small *RTO* is required.

A Web-based store must have an *RPO* close to zero (the company does not wish to lose any sales or, even worse, acknowledge a sale to a customer and then not deliver the product). However, if shipping and billing are delayed by even a day, there is often no serious consequence, thus relaxing the *RTO* for this part of the application.

A bank's ATM system is even less critical. If an ATM is down, the customer, although aggravated, will find another one. If an ATM transaction is lost, a customer's account may be inaccurate until the next day, when the ATM logs are used to verify and adjust customer accounts. Thus, neither *RPO* nor *RTO* need to be small.

²⁹ LaPedis, R.; "RTO and RPO Not Tightly Coupled," Disaster Recovery Journal; Summer, 2002.



Example Objectives
Figure 6-1

Once a company decides what RPO and RTO are applicable to an application, the method for backup and recovery of that application becomes much more evident. The remainder of this chapter relates various methods for backup and recovery to their RPO and RTO characteristics.

Replicating the Application Data

Immediately recovering services to users on a failed node requires that a current copy of the application database be available on at least one other node accessible to all nodes in the system, and that users be switched to a surviving node. Providing a replicate of the application

data at another node can be done in many ways. Classically, magnetic tape has been used to periodically take a snapshot of the database so that it can be loaded onto the backup system. This is still the most common means for database recovery as of the writing of this book. Although this technique results in a large RPO (many hours of data may be lost) and a large RTO (it may take hours to days to restore service), tape backup is relatively inexpensive.

However, several data replication techniques allow a remote database to be kept in near synchronism with the primary database, thus reducing the RPO to a very small value (a few seconds or less of lost data). In some cases, exact synchronism can be achieved, thus reducing RPO to zero. These techniques also lend themselves to very fast recovery, or small RTO, measured in minutes or even in seconds.

Various data replication techniques are shown in Figure 6-2 and are described next. To simplify the discussion, we will first consider replication between two systems. In some cases, one might be a primary system and the other a passive backup system. In other cases, both systems might be active with each serving as a backup system to the other.

These techniques are extendable to application networks comprising several independent but cooperating nodes. The implications of multiple-node application networks follow the description of the basic techniques.

No Replication

The use of tape to back up a system requires no data replication facility. Such backups tend to have long recovery times and may have a probability of significant data loss. The backup system is not contributing to application processing.

No Replication, Periodic Backup Only

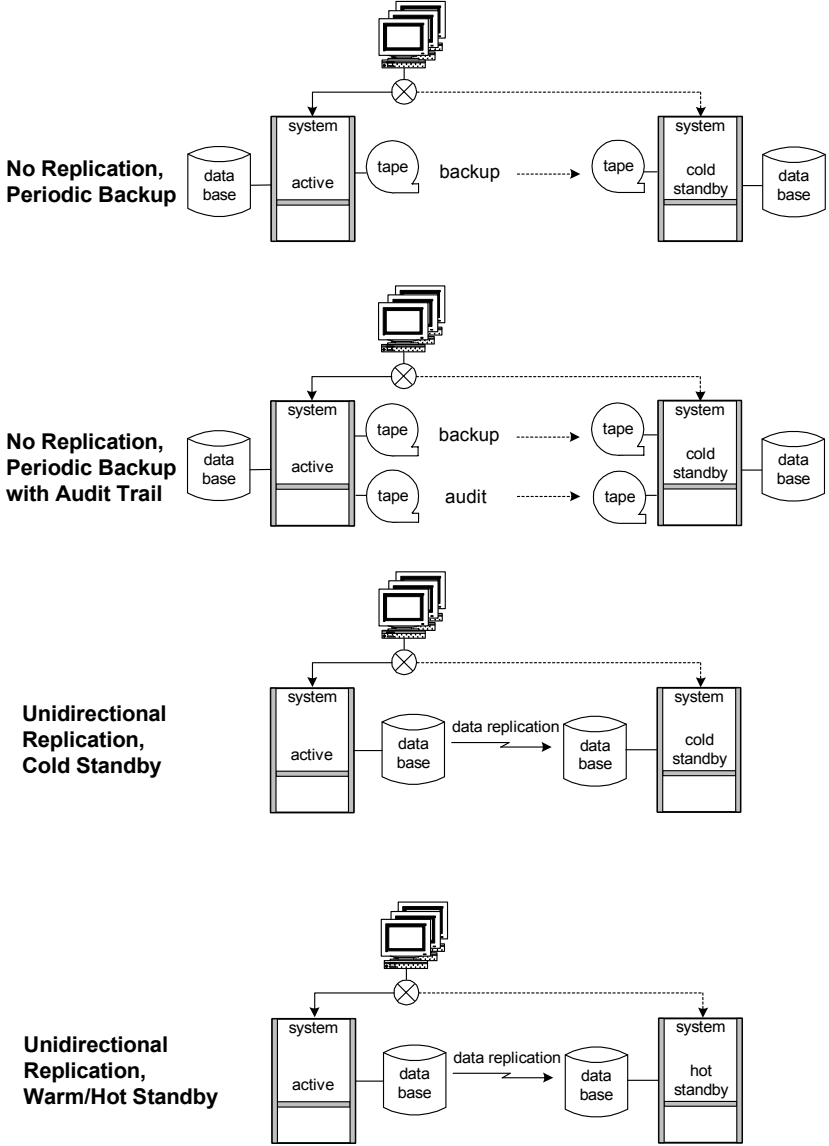
The classic method for restoring a database at a backup site is via magnetic tape. Periodically, a snapshot of the database, or of the changes made to the database since the last snapshot, are written to tape. If the snapshot is to represent a consistent view of the database, the application must be paused during the snapshot. If it is not paused, the snapshot represents an inconsistent view of the database since it will show updates partially made by some transactions which may or may not be committed.

In the event of a failure, if the latest backup tapes have not been applied to the backup system, they must be recovered from the tape storage site and moved to the backup site. They must then be loaded before the backup site can take over operations. There is some probability that a tape will be unreadable. This will require the retrieval of an alternate backup tape if one exists. If multiple backup tapes have not been created, then a significant amount of data will be lost if a tape is unreadable.

If the backup tapes have already been applied to the backup system, then the database reload is complete as far as it can go. All database updates between the last backup and the time of the failure are lost. Thus, not only does this backup method have a potentially long recovery time (large RTO), but it is also susceptible to a large amount of data loss (large RPO).

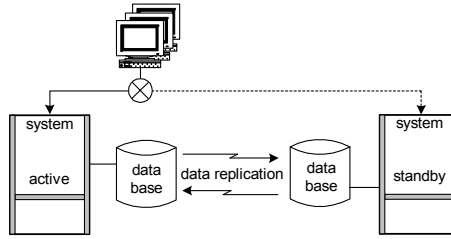
No Replication, Periodic Backup with Audit Trail

In some systems, a continuing audit trail of database changes made since the last snapshot is written to tape. Playing back the audit trail against an inconsistent snapshot can restore a recovered database to a consistent state. Under proper procedures, the snapshot copies are applied to the backup system as soon as practicable. Then, should an outage occur, only the audit trail tapes since the last snapshot (at least, those that can be made available) need be applied to provide a reasonably up-to-date database for the backup system.

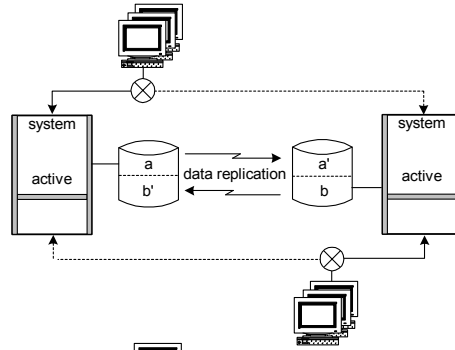


Replication Methods
Figure 6-2

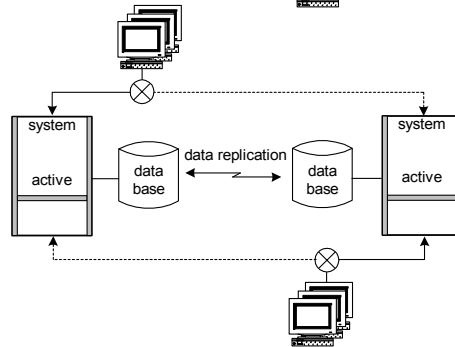
**Bi-directional
Replication,
Hot Standby**



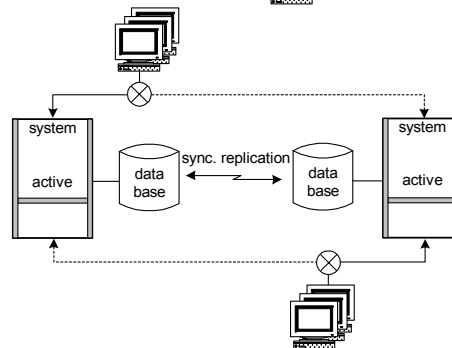
**Partitioned
Active/Active
Replication**



**Asynchronous
Active/Active
Replication**



**Synchronous
Active/Active
Replication**



**Replication Methods
Figure 6-2 (cont)**

Playing back the audit trail tapes may extend the recovery time (RTO) if snapshot tapes have to be loaded first, but doing so can dramatically reduce the amount of lost data (RPO).

Unidirectional Replication

With unidirectional replication, the application is only running on the active system. The backup system may use the replicated data for read-only purposes but cannot be actively updating that data.

Unidirectional Replication – Cold Standby

The simplest form of data replication feeds a backup database in near real time with updates that are being made to the active database. Except for data replication activity, the backup system has no participation in the application. There are no application processes running on the backup system, though the system may be used for other work. Thus, it is called a cold standby.

Unidirectional Replication – Warm or Hot Standby

The same data replication procedure can be used with a warm or hot standby. Both warm and hot standbys have all associated applications up and running. However, a warm standby has opened files only for read access. If it is to take over, it must reopen these files for full access. A hot standby has all files open for full access and is available to immediately take over in case of a primary failure. In such cases, updates are still replicated from the active system to the backup database. However, until the backup applications are requested to take over, they are not making any changes to the database. If they are doing any work, they are only executing query and reporting functions against the database.

Bi-Directional Replication

A primary system with a hot backup system may also be configured with bi-directional replication. In one configuration, it will behave exactly as if it were a unidirectionally replicated hot standby

Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein

system as described above, except that now it is rather straightforward to switch active/standby roles. This can be particularly advantageous for testing system recovery and keeping operations personnel current in switchover procedures via periodic training.

To switch systems for testing purposes, all that is necessary is to switch users to the standby system. Since all applications are up and running, the only impact on users will be the short switchover time. There will be no data loss as there might be if the primary system failed (the time required for the replication engine to drain its primary system workload will usually be sub-second). However, as the users now continue with their normal activities, the old primary system acting as the new standby system will continue to have its database kept in synchronism with the new active system's database. A return to the original configuration can be accomplished at any time by simply switching users.

Active/Active Replication

A major advantage of bi-directional replication is that the full capacity of both systems in the network is available for application processing. Thus, the reliability and disaster tolerance afforded by replicating systems can be achieved with a much smaller complement of equipment. With no replication or with unidirectional replication, only half of the system capacity is available for application processing (though it may be available for query and reporting or for running other applications that do not affect the replicated application's database).

If applications are running in the backup system, there is no reason why they can't be actively engaged in providing the same services as the primary system. Each system can be supporting its own community of users, be making its own updates to its local database, and be replicating these updates to the remote database. Such distributed applications are called *active/active* applications.

However, there are some serious problems with active/active replication. One is ping-ponging, or the replication by a system of an

update back to that system from which it was received. Provisions must be made to avoid ping-ponging.³⁰

Another problem is data collisions. A data collision occurs when two users update the same data item on different copies of a database. Since the same data item now has different values in different instances of the database, typically only data content or application-knowledgeable reconciliation procedures are capable of returning the database copies to a consistent state. That is, the correct state of the affected data item depends upon how it is used by the application. In some cases, the data collision may be safely ignored, or it may be resolved automatically by the replication engine. In others, the collision needs to be resolved manually.

Partitioned Active/Active Replication

Data collisions can be avoided by logically *partitioning* the database and the processing activity so that each system will still have a full copy of the database, but the users at each system will update only the database partition that their system owns. These updates can then be replicated to the other system with no fear of data collisions.

For instance, it may be that only the sales office that services a customer can make updates to that customer's records. A cell phone service provider might send call records to a specific system based on the first digit of the calling or called number.

Partitioning might even be done on a time basis. The ownership of the database might be rotated to give different systems an opportunity to process requests which they have been queuing, or a brokerage's database might be transferred during the 24-hour day to offices which are currently open around the world.

³⁰ Strickler, G.; et al.; "*Bi-directional Database Replication Scheme for Controlling Ping-Ponging*," United States Patent 6,122,630; Sept. 19, 2000.

Asynchronous Active/Active Replication

If data collisions are not deemed to be a serious problem, then both systems can be actively processing all transactions; and no partitioning is required. This may be the case, for instance, if users are geographically segregated and if the risk of data collisions is small. Alternatively, if the cost of resolving data collisions is small, or if data collisions can be resolved automatically, asynchronous active/active replication may also be appropriate.

Synchronous Active/Active Replication

In many applications, partitioning will not work or may not be feasible. Any transaction may possibly affect any data item in a database, and resolving data collisions may simply be too difficult or require too much manual intervention to use asynchronous active/active replication. Both systems must cooperate to ensure that a transaction they own will not cause a data collision with a transaction owned by the other system.

This can be accomplished by ensuring that a transaction can lock throughout the network all of the instances of a data item which are affected by that transaction. It can then update all data item instances before releasing the locks. We call this *synchronous replication*. Synchronous replication is discussed in Chapter 4, “*Synchronous Replication*.”

One way to implement synchronous replication is to use distributed transactions to start a single transaction that spans both systems. Then updates to all data item instances are guaranteed, or else none will be made (the transaction aborts).

Distributed transactions may work well in campus environments but are inappropriate for geographically dispersed networks. This is because of the many channel propagation delays caused by the communication channel round trips that are required for a transaction – two per update (a read followed by a write) plus two for the prepare and commit messages. Recognizing that round-trip signal propagation

time between the U.S. coasts is about 50 msec. and that it can be 250 msec. between London and Sydney, it is clear that network transactions can add seconds to an application's transaction response time. This is called *application latency*.

Geographically dispersed systems are a requirement for disaster recovery. For these systems, data replication with coordinated commits offers a satisfactory alternative³¹ as described in Chapter 4. With this technique, the data replicator will start independent transactions on each system. Updates are replicated asynchronously, but the commits of these independent transactions are coordinated by the data replicator. Neither transaction is committed unless both are guaranteed to commit.

By using synchronous replication, both systems may participate as equals in the transaction. If one system goes down, the users on the other system are unaffected and can continue to process transactions while the downed users are switched over to the surviving system.

Recovery Time

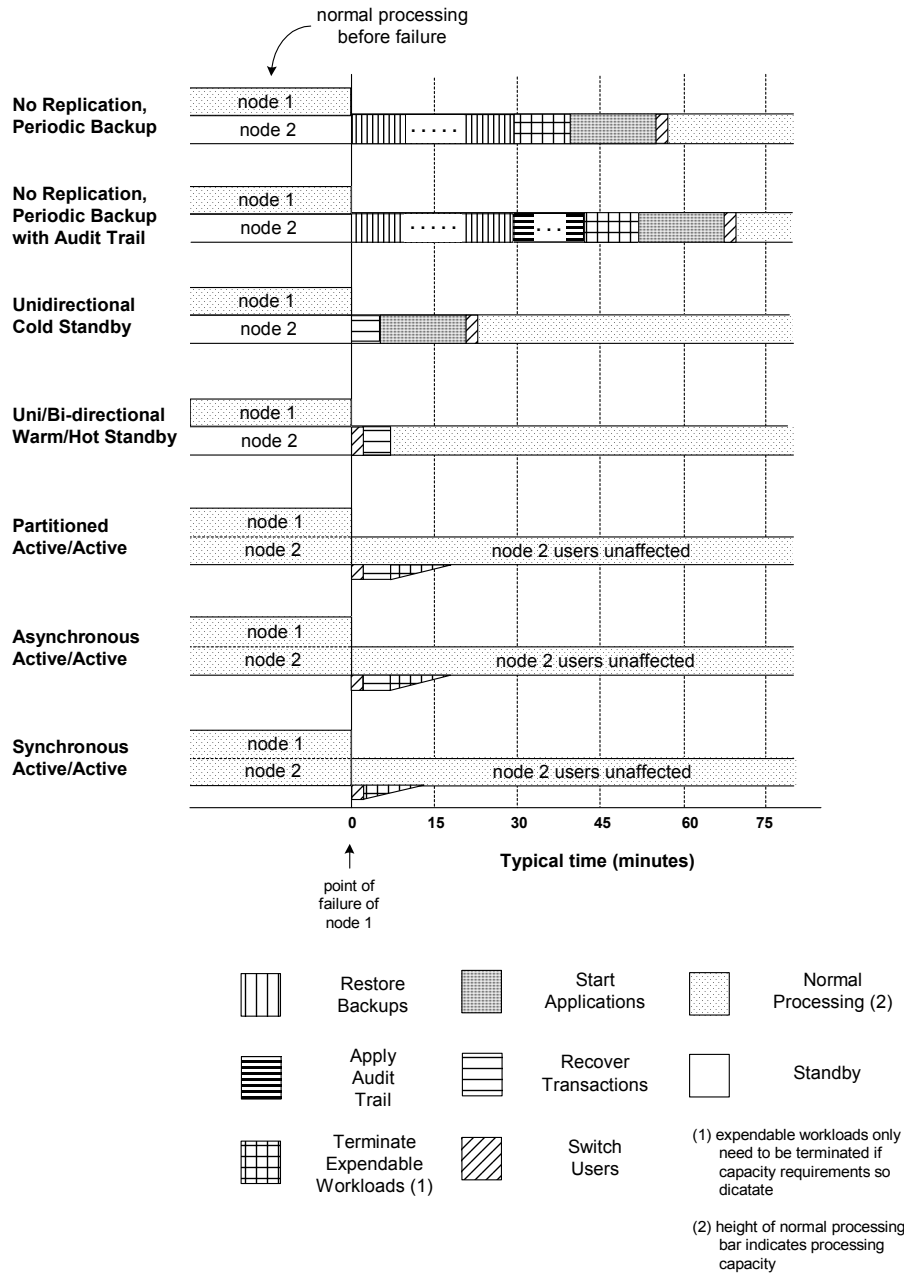
Recovery time, and hence RTO, can be vastly different for these recovery strategies, as shown in Figure 6-3.

No Replication

No Replication, Periodic Backup Only

If data replication is not used, the current database must be restored at the backup site. This involves first retrieving the appropriate backup tapes and then getting them to the backup site. The latest snapshot (or even an entire backup) must be loaded.

³¹ Holenstein, B. D.; et al.; "Collision Avoidance in Database Replication Systems," United States Patent Application 20020133507; Sept. 19, 2002.



Typical Recovery Times
Figure 6-3

Breaking the Availability Barrier

At this point, non-critical workload can be shed; and the applications to be recovered are started. Finally, users can be switched from the off-line system to the backup system; and normal activity can continue.

The reconstruction of the database can take several hours to several days, depending upon its size. The time required for the other functions of expendable workload shedding, application startup, and user transfer usually pales in comparison to database recovery time. In fact, these functions can be done during database recovery.

No Replication, Periodic Backup with Audit Trail

A major problem with the simple tape backup procedure discussed above is that there can be hours or days of data updates that are lost from the time of the last backup or snapshot to the time of the system failure. This results in a very large RPO. The RPO can be significantly reduced if provisions have been made for audit trail tapes which record all updates made to the database since the last backup or snapshot.

The use of audit trails not only results in a lower RPO, but also can bring the database to a consistent state by completing incomplete transactions or by purging aborted or incomplete transactions.

However, the use of audit trail tapes has the effect of increasing the recovery time, or RTO. Not only must the database be restored, but now the audit trail tapes must be re-played to bring the database more up-to-date.

Unidirectional Replication

Unidirectional Replication – Cold Standby

If data is being replicated to a cold standby, and if the standby must take over processing, then the applications must be started and the users switched over before service can be restored. Application startup generally requires many minutes before users can reinitiate

Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein

their sessions and continue their activities. In addition, incomplete transactions must be purged. This may take a few more minutes.

Unidirectional Replication – Warm or Hot Standby

If data is being replicated to a warm or hot standby, then all that is required is for incomplete transactions to be recovered or aborted and for the users to be switched over. Also, if the backup is a warm standby, files must be reopened for full access. User switching can be accomplished in seconds, especially if switchover is automatic upon detection of a non-operating primary. File reopening is usually a matter of seconds for smaller applications but may be minutes for larger applications. Moreover, it may take several minutes to restore all incomplete transactions.

Bi-Directional Replication

Hot Standby

So far as recovery time is concerned, a hot standby bi-directional replication system recovers in the same recovery time as the unidirectional hot standby system described above.

Partitioned Active/Active Replication

The recovery from a system failure when using bi-directional replication is similar to that of unidirectional replication to a hot standby. The downed users simply need to be switched over to the surviving system, and transactions need to be recovered. Transaction recovery is usually accomplished by each application (and perhaps ultimately by each user) checking that the last transaction was accepted. If the transaction was not accepted, it is resubmitted. The other users continue their activities on their application partitions. However, the partitioned activities of the downed users are terminated until recovery is complete.

The surviving system becomes the owner of both database partitions. In this case, it may be desirable to shed some load by

stopping non-critical applications in order to provide enough capacity for all users.

Asynchronous Active/Active Replication

If asynchronous active/active processing is being used, there is no impact on the users connected to the surviving system. They continue to be provided with all services. The loss of a system is only the loss of capacity.

The downed users must be switched over to the surviving system. Following recovery of transactions that had been in process on the failed system, and perhaps the shedding of some non-critical applications, full service can be restored.

Synchronous Active/Active Replication

Likewise, if synchronous active/active replication is being used, there is no impact on the users connected to the surviving system. They continue to be provided with all services. The loss of a system is only the loss of capacity.

After the downed users have been switched over to the surviving system and perhaps after shedding some non-critical applications, full service can be restored. No transactions need to be recovered since any transactions being processed by the failed system at the time of failure will be aborted.

Data Loss

RPO is the goal for the maximum amount of data that may be lost due to a failure. The amount of data that may be lost varies dramatically with these techniques.

No Replication

If no audit trail tapes are used, all data since the last backup is lost. This can be hours or even days worth of data.

If audit trail tapes are used, data loss depends on whether some of these tapes were destroyed in a disaster. Data loss can range from seconds to hours.

Asynchronous Replication

Asynchronous data replication may be used with either unidirectional or bi-directional replication. That is, the source application does not wait for the data to be safely stored and/or applied at the target system. The interval between the time that an update is applied to the source database and the time that it is applied to the target database is known as replication latency. Replication latency is the time that replicated data is in the replication pipeline.

In the event of an outage, data in the replication pipeline may be lost. This typically represents one to thirty seconds of database updates, providing that updates are sent to the target system as soon as they have been applied at the source system. If transactions are held at the source system until they have been committed before sending them to the target system, then replication latency and therefore the amount of lost data can be significantly higher. This is especially true if long-lasting transactions (such as batch transactions) hold up a completed transaction and prevent it from being sent over the replication channel.

Synchronous Replication

Data collisions can be avoided by the use of synchronous replication. The source application does not commit a transaction until it is assured that the transaction will complete on the other system. If it is unable to commit, then the transaction is aborted at the

other system. Therefore, there is no data loss (zero RPO) under normal operations if synchronous replication is used.

All synchronous replication methods have a very small but very critical uncertainty window. For instance, with networked transactions that use a two-phase commit protocol, if the network should fail between the receipt of the phase 1 response (ready to commit) and the receipt of the phase 2 response (commit), the source system does not know whether the target system received the commit command. Conversely, if the target system did not receive the commit command, then it does not know what to do with the transaction – commit it or abort it. This is often referred to as a *hung* transaction.

A similar window of uncertainty exists for coordinated commits. However, in either case, data changes are not lost. They are simply held in a locked state at the target system. This gives an opportunity to query the source system to determine whether the transaction actually completed. Assumed here, of course, is that the source system is available and accessible. Otherwise, the application or the user will have to determine whether or not to commit or abort the transaction at the target system. This situation will self-correct via the inherent recovery facilities of the replication engine should the source system once again become available.

Recovery Strategies

If either the primary or backup system should fail, or if a network fault should isolate a system, then the failed system's database will become rapidly out-of-date as processing activity continues at the surviving system. When the failed system is returned to service, its database must be brought to the current state.

If magnetic tape is being used to back up a system, then the database recovery of the failed system is straightforward. The latest tape copy of the database and any change tapes are simply loaded onto the recovered system, and operation proceeds as usual.

Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein

The recovery of a downed or isolated system when data replication is being used is somewhat different and is, in some cases, more difficult.

Unidirectional Replication

Asynchronous Replication

If data is being replicated to a cold, warm, or hot standby, and if data replication is asynchronous, then no special action need be taken. During the period of failure, the queue of transactions to be replicated simply grows at the source system.

When the failed system (or the interconnecting network, whichever caused the failure) is returned to service, asynchronous replication continues; and the queue to the downed system is drained. When the queue size falls below a level considered to be normal, the failed system can be considered to be recovered.

Synchronous Replication

If synchronous replication is being used, then the active system (which now may be the original backup system if the primary system failed) must switch to asynchronous replication and queue new transactions for later replication. When the failed system is returned to service, the queue must be drained. The saved transactions are replicated to the system being recovered.

The active system continues to function during recovery, using asynchronous replication to the recovering system. It will do so until the replication latency falls to the point that it will not seriously affect the response times of subsequent synchronously replicated transactions (i.e., the application latency is within acceptable bounds). At this point, synchronous replication is resumed; and synchronous replication messages are queued behind the remaining asynchronous replication messages.

Active/Active Replication

System Failure

If both systems are processing transactions in an active/active configuration, and one system fails, then the affected users may be switched to the surviving system and full functionality continued (within the capacity, of course, of the surviving system). Operation and recovery are as described above with respect to unidirectional replication, with the surviving system queuing transactions that must be posted to the failed system.

When the failed system is recovered, the transaction queue is drained while normal processing continues at the active system. Users may then be reconnected to the recovered system.

If asynchronous replication is being used, users may be switched when the queue size is small enough to reduce the chance of data collisions to an acceptable level.

If synchronous replication is being used, then the queue is drained to an acceptable level. At this point, normal processing can proceed with synchronous replication, as described above with respect to unidirectional replication.

Network Failure

If the network connecting the two systems should fail during bi-directional replication, then both systems are capable of continuing independent processing. However, in the general case, this can lead to extensive data collisions if the network outage is lengthy. There are several options for continuing operation in this case:

- a) If the application can be partitioned so that any data item can be modified only by users at one system, then do so (partitioning may already be in effect). Each system can then continue to function independently and to queue its transactions to the other

Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein

system for later replication when the network becomes operational. Recovery procedures for asynchronous and synchronous replication are described above.

b) If the application cannot be partitioned, and if data collisions are not acceptable, then all users can be switched to one of the systems. The isolated system is, in effect, down and is recovered as described above when the network become available.

c) If data collisions can be tolerated, then each system can continue to function and to queue its transactions for later delivery, with recovery proceeding as described previously. If replication is asynchronous, no change in operation occurs. Rather, the replication latency time is now the network down time. If synchronous replication is being used, the systems must switch to asynchronous replication until recovery, as described above. In either case, data collisions occurring during the outage must be detected and resolved.

Comparison Summary

Recovery times (RTO), data loss (RPO), and the possibility of data collisions for these various techniques are summarized in Figure 6-4. As can be seen, if replication is not used, both RTO and RPO can be very high – it may take hours to days to recover; and hours to days of data may potentially be lost.

If replication of any kind is used, RTO is reduced dramatically to minutes, seconds, or even to near-zero if active/active replication is used (depending upon the time that it takes to switch users on the failed system to the active system). Likewise, the amount of data that may be lost, as measured by RPO, may be as little as those changes which occurred in the few seconds before failure or, in the case of synchronous replication, may even be reduced to zero.

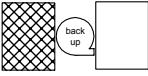

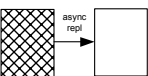
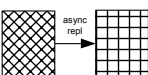
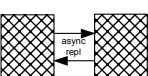
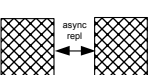
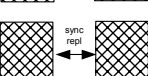
Even when replication is used, the RPO and RTO that can be achieved is very dependent upon the replication technology that is used. This is illustrated in a general way in Figure 6-5.




As shown in Figure 6-5, synchronous replication achieves a zero RPO. That is, no data is lost when a system fails. Any transaction in process at the time of the failure is aborted (as described earlier, there is a very small uncertainty window which leads to locked data but not to lost data). Furthermore, if the application is configured as an active/active application, RTO is also zero for all users but those on the failed system. Even the users on the failed system can be quickly restored to service by switching them to a surviving system.

The amount of data which may be lost with asynchronous replication is a direct function of the replication latency of the replication channel. If replication is substantially process-to-process (i.e., changes are extracted directly from the source database and applied directly to the target database with no intermediate queuing), replication latency can be quite small (sub-second) and RPO can be driven fairly close to zero. To the extent that there are queuing points in the replicator or that there are intermediate disk storage points, replication latency will be longer and RPO higher. Typical queuing points occur at the communication channel as well as at processes that are reading source data or are updating target data. Process queuing can be reduced by multithreading but perhaps at the expense of referential integrity as described in Chapter 10, “*Referential Integrity*.” Communication channel queuing can be reduced by using smaller block sizes at the expense of communication channel utilization efficiency.

Recovery time, which directly affects RTO, is also a function of several factors. If the application processes in the backup system are inactive, they have to be activated; this can take several minutes. However, if the application is configured as an active/active application, then the only requirement is for the users on the failed system to be switched to the surviving system. The RTO is zero for all other users.

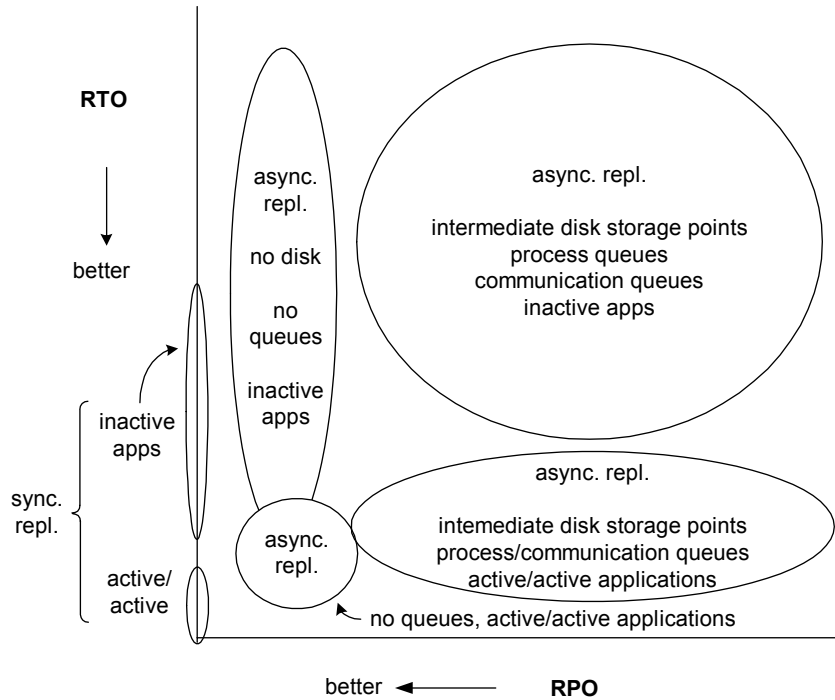
Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein

Type	Recovery Time (RTO)	Data Loss (RPO)	Data Collisions
 No Replication, Periodic Backup	hours to days	hours to days	no
 No Replication, with Audit Trail	hours to days	seconds to hours	no
 Unidirectional Cold Standby	minutes	< 1 second	no
 Uni/Bi-directional Warm/Hot Standby	seconds	< 1 second	no
 Partitioned Active/Active	seconds	< 1 second	no
 Asynchronous Active/Active	none	< 1 second	yes
 Synchronous Active/Active	none	none	no

 cold standby	 hot standby	 active system
--	---	---

**Replication Method Comparison
Figure 6-4**

Furthermore, if the replication scope is an update rather than a transaction, or if the replication engine does not preserve the source transaction boundaries, then the database may not be in a consistent state. It will have to be cleaned up by determining incomplete transactions and backing them out, which could take a significant amount of time.



The Impact of Replication Technology on RPO and RTO
Figure 6-5

The above recovery time comments apply to asynchronous replication and to a lesser extent to synchronous replication. With synchronous replication, the database will not have to be cleaned up because all changes are within the scope of a transaction. However, synchronous replication can be used with a cold standby, in which case recovery time necessarily includes the activation of the application processes in the standby system.

Thus, as summarized in Figure 6-5, the actual RPO and RTO that can be achieved are not just a matter of synchronous versus asynchronous replication but are strongly dependent upon the specific technology used to implement the replication facility.

Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein

Rule 28: *RPO and RTO are both a function of the data replication technology used to maintain databases in synchronism.*

Multi-Node Applications

Any of these data replication techniques can be extended to multi-node applications to provide multiple recovery sites or to provide backup at one site for multiple primary sites.

Moreover, bi-directional replication can be used to spread an application load over several nodes, as described in Chapter 2, “*System Splitting*.” One advantage of this is a significant increase in full capacity availability. Another advantage is that if a node fails, only a portion of system capacity is lost. Except for users at the failed node, substantially full service continues to be provided. Moreover, the downed users can be returned to service within seconds by switching them to surviving nodes. Their only penalty is a session loss which they may not even recognize as a system failure since session loss due to communication problems is often a frequent occurrence anyway.

Furthermore, the loss of two nodes-worth of capacity is so unlikely that it typically can be ignored.

One problem with multi-node bi-directional replication when using partitioned applications is that database partitioning must be finer grained. Moreover, the transfer of ownership of a failed partition must be resolved. This is not a problem with active/active bi-directional replication because database partitioning is not required.

Recovery Decision Time

Should a disaster destroy a site, the site’s inability to function is pretty obvious. There is not much need for a decision process to decide to activate the backup site.

However, replicated architectures also protect against normal system outages. Now when a failure occurs, a decision must be made whether to recover the failed system or to switch over to the backup system.

This can be a high-stress process. Remember Bartlett's Law (Rule 19) - *When things go wrong, people get stupider*. First, operational personnel must acknowledge that there is a problem. Then the problem must be diagnosed and consensus reached on the optimum recovery action. Often, management must be consulted to approve any drastic actions such as a switchover and the termination of other applications. This process can take an hour or two, especially if it happens during off-hours.

Thus, for non-disastrous system outages, the time lines in Figure 6-3 should be preceded by an extended decision-making time. This directly impacts the non-replication and unidirectional replication methods. It only affects the downed partition for partitioned bi-directional replication.

Active/active bi-directional replication does not suffer a decision time since these applications keep on running in the presence of a node outage. The only decision necessary is whether to switch the users at the failed site to the surviving site.

Summary

In the early days of computing, only tape was available for backups. In those days, the batch nature of computing made long RPOs and RTOs acceptable. However, as systems came online and as real-time became a reality, RPO and RTO became more important. This importance only increased as systems became more-mission critical to enterprises.

Replication technology came about to satisfy the needs of these real-time, mission-critical systems. Early replication facilities reduced RPOs and RTOs from days or hours to minutes. As pressure increased

Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein

to further improve these objectives, replication technology improved to reduce replication latency and shorten recovery times. The use of higher speed communication channels and multithreaded replicators and the elimination of intermediate disk storage points have all contributed to faster replication and less loss of data. The tolerance of replication facilities to provide active applications at the backup have greatly speeded recovery time.

Recent solutions to critical bi-directional replication problems such as the ping-ponging of updates and the detection and resolution of data collisions have allowed active/active systems to become a reality. Active/active applications allow all of the data processing power in a network to be utilized. Now with the inherent increase in computing system power which compensates for the additional overhead inherent with synchronous replication, efficient active/active systems can be built using synchronous replication. These systems will not only reduce RPO to zero, but they will also virtually eliminate RTO as a consideration. Recent advances in more efficient transaction coordination over wide area networks (coordinated commits) have reduced the performance impact of synchronous replication even further.

As a result, the technology to build extremely highly available and disaster-tolerant systems at little additional cost, with no data loss due to a network or node failure, and with no service loss to users except briefly at a failed node is here today.

What's Next

We have now concluded our discussion of ways to split systems into independent cooperating nodes to achieve significantly higher availabilities. We have explored the various methods that can be used to keep the database copies which are distributed across the network in synchronization. We have considered the impact of hardware faults, software faults, failover faults, and recovery times on system availability, and have described some important measures of distributed system effectiveness.

Breaking the Availability Barrier

In the next Chapter, “*The Ultimate Architecture*,” we put all of this together to suggest architectures that can provide significantly increased availability at little or no additional cost using what we have learned so far.