



NETAPP WHITEPAPER

DATA REPLICATION AND RECOVERY OVER ANY DISTANCE ENSURING 100% DATA CONSISTENCY

Network Appliance, Inc.
December 2006 | WP-7007-0307

TABLE OF CONTENTS

1 Introduction	3
2 A Short History of Disaster Recovery Approaches	4
Tape-Based Backup.....	4
Synchronous Replication.....	5
Asynchronous Replication.....	6
3 The Challenges of Asynchronous Replication.....	6
No Mechanism.....	6
Serialization.....	6
Single Device Time Stamping	7
Group Stamping	7
Multidevice Time Stamping	7
4 ReplicatorX: The Optimal Replication Architecture	8
5 Summary	9

1 INTRODUCTION

All enterprise applications that handle data, such as file systems and databases, are written so that they can reliably recover from failures that cause the application to crash. These applications are called crash tolerant, and their recovery is based on the fact that any changes to their data are written in a very well-defined order that is controlled by the application. If replication is deployed in an application environment, any changes to the replica or copy at the remote site must be applied in exactly the same order as defined by the application at the primary site. If this order is preserved at the remote site, then crash tolerance is maintained, and the application can reliably recover. If this order is not preserved, then data consistency is compromised at the remote site and a reliable recovery cannot occur.

Synchronous replication is a well understood and widely deployed technology for disaster recovery in large enterprise environments. Synchronous replication preserves the primary site write ordering by performing all writes in lockstep at all sites. When a write is requested by the application at the primary site, it must be applied in all locations before an acknowledgement is returned, allowing the application to proceed. In this manner, primary site application write ordering is preserved and crash tolerance is maintained in the data replica at the remote site, ensuring a reliable recovery. However, there are limitations to synchronous replication in three areas: performance, distance, and use in IP-based networks.

Synchronous replication introduces latency while waiting for the completion of the writes. Even when replication is local (for example, when it is performed across servers that reside in the same room), the latency to complete all writes can be two to three times what it would be if replication were not in use. This latency affects the performance of the application environment being replicated, slowing it down in often noticeable ways. When distance is introduced between the replication source and target, latency is further increased by the amount of time required to send the write to the remote site and receive an acknowledgement of its completion. Eventually, the latency becomes so great that synchronous replication is not viable in a production environment. These latency issues have limited the use of synchronous replication to configurations where the sites are typically no more than 30 to 50 miles apart.

The use of IP-based networks is also problematic in synchronous replication environments. Delivery is nondeterministic for IP packets, and the latencies caused by IP routing and error recovery mechanisms generally introduce so much additional latency that synchronous replication cannot be used in these environments at all. This has led to the deployment of synchronous replication only across dedicated, proprietary, and very expensive networks.

The concept of asynchronous replication was introduced to meet the need for longer distance replication configurations that can effectively deal with latency and do not affect primary application performance. Many companies naturally assume that asynchronous replication offers the same data consistency and reliability over long distances that synchronous replication does over short distances; however, preserving primary site write ordering becomes a significant challenge in asynchronous environments. It is therefore important to understand, for any asynchronous replication solution being evaluated, how the issue of data consistency is addressed. If the replication solution provides for primary site write ordering to be maintained at the remote site, the operational impact of the particular method employed must also be understood. These tradeoffs are often overlooked during evaluation, and can cause significant impact on operations and recoverability as environments scale from pilot to production.

This white paper discusses the storage industry's attempts to address the distance issue for disaster recovery purposes, evaluates the approaches to preserving primary site write ordering in asynchronous replication solutions, and points out the

key issues to understand when considering longer distance disaster recovery configurations.

2 A SHORT HISTORY OF DISASTER RECOVERY APPROACHES

TAPE-BASED BACKUP

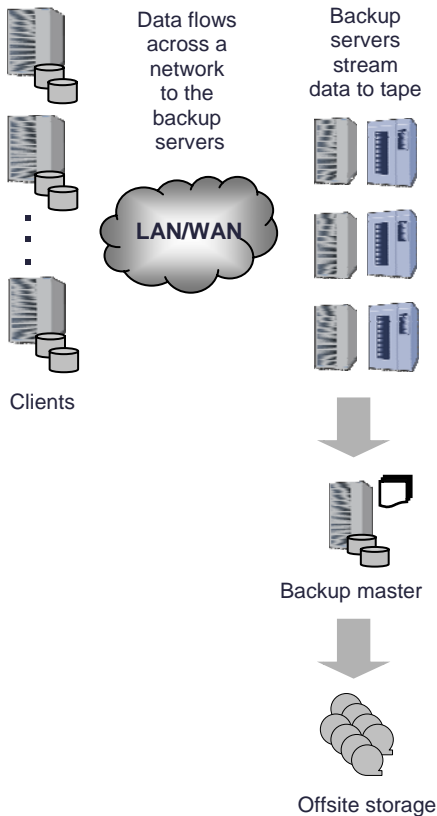


Figure 1) Tape-based backups (copies) are created and then shipped to an offsite location for safekeeping.

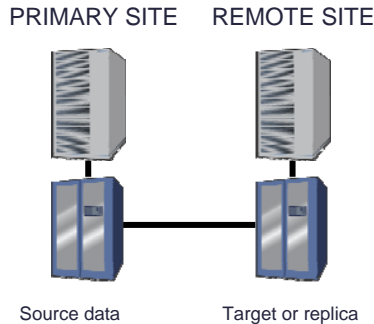
Disaster recovery has been a concern since the earliest days of business computing. Access to a geographically separate copy of data that would allow business operations to be recovered in the event of a catastrophic disaster was a requirement in larger environments. Initially, this need was met through the use of enterprise backup software that allowed copies of operational data to be periodically created on media that could easily be transported to off-site locations for storage. In the event of a catastrophic failure that destroyed servers, storage subsystems, or in some cases even entire sites, recent operational data could be retrieved from the off-site location and used to restart business operations. The medium of choice for backup operations has historically been magnetic tape.

When the response time of business operations was measured in hours or days, tape-based backup did a good job of meeting recovery requirements in a cost-effective manner. But as information technology took on a central role, more and more data needed to be written to create these copies, which took more time. Increasing globalization drove the industry toward 7x24 operations, cutting down the amount of time that computers could be offline to create these copies (or tape-based backups). Because of political and natural events that have led to widespread disruption of business operations, the ability to reliably recover has grown significantly in importance. The new demands placed on disaster recovery solutions pointed out the following weaknesses of the tape-based backup software approach.

- **Backup window:** Applications are shut down so that a transaction-consistent copy of the data can be backed up, creating a window of downtime. This backup window is the amount of time an application can be unavailable for business use because of backup operations. Shrinking backup windows have been a management issue for at least the last 10 years.
- **Data loss:** Backups created a copy of the data at a point in time (the point at which the application was shut down to take the backup). Because of shrinking backup windows and the increasing time it took to back an application up (due to larger capacity), backups were not taken very frequently. Even today, with higher speed backup technologies, backups are rarely taken more frequently than once a day. When a recovery from tape was required, the best that could be expected was recovery from the last backup. Any changes to the data that had happened since the last backup were usually lost.
- **Media integrity issues:** As data was copied to tape, it was converted from disk format to tape format. A restore would convert the data from tape format back to disk format. After backups were complete, tapes were removed and sent to off-site locations so that they would not be affected by catastrophic disasters, raising the issue of tape management. Large companies often stored thousands or tens of thousands of tapes at these sites. Tapes also wore out over time and with repeated usage. All of these factors combined to cause tape media integrity issues. In 2003, the Gartner Group stated that, on average, one in four backup tapes had unrecoverable files. In a survey of enterprise technology staff published by the Enterprise Storage Group in April 2004, 61% of respondents identified media failure as the most common cause of a recovery failure. The problem, of course, is that a company could not know that a file was unrecoverable until recovery was attempted, at which point it was too late.

- **Restore performance:** Most enterprise backup software is based on designs that are at least 15 years old. Backup performance was the key metric in the earlier era, when disasters were rare, and architectures and features were developed that traded off restore performance to get better backup performance. Now that restore performance is increasingly important, many backup vendors are limited by architectural decisions they made in the past that are very difficult to change.

SYNCHRONOUS REPLICATION



- Exact copies of the data are kept in sync at two sites
- Disk-based synchronous replication solves backup window, data loss, and data integrity issues
- Introduces new problems:
 - Vendor lock-in
 - Extremely high cost
 - Distance limitations

Figure 2) Synchronous replication addressed the issues with tape-based backup but introduced new problems.

It soon became apparent in the large, very computer-intensive business operations that backup could not adequately meet the need for business operation recovery. These companies began to look at an alternative technology called synchronous replication, which created a mirror copy of data in a separate physical location, keeping that replica continuously updated as changes occurred. These two mirror copies were referred to as the source (the original copy) and the target (the replica). The replication was synchronous in the sense that both copies were updated with every write so that they shared the same data state at all times and primary site write ordering was preserved. Synchronous replication is widely deployed in certain industries, and was almost a \$1B business in 2003.

Synchronous replication addressed the backup window, data loss, media integrity, and restore issues. Replication occurred in the background, transparently to the primary application, completely removing the backup window. Since replication was continuous and the source and target were kept in perfect lockstep at all times, recovery could always occur from the latest data state. Replicas were kept in disk format attached directly to a server at the secondary site, so there were no tape media integrity, format conversion, or lost media issues. Restores now occurred as a single-step process from disk because a dump from tape to disk was no longer required, minimizing operational issues associated with a restore. And because disk was a much faster medium than tape that also supported random access, restores were faster still.

Synchronous replication was not a panacea, however, and introduced three new problems. First, all the popular implementations are storage-based, locking customers into replicating data between arrays of a single vendor. This leads directly to the second problem, which is extremely high cost. Because customers are required to purchase a minimum of two enterprise storage arrays (one for each site), the entry price point for even a small replication configuration is on the order of \$500K to \$1M, regardless of the capacity required. And third, synchronous replication introduced distance limitations. The requirement to simultaneously write to an online target introduced latency, and the farther away the target, the more latency was introduced. This additional latency affected the primary application, slowing it down while it waited for writes to the target to complete. To keep this impact to a minimum, companies generally limited synchronous replication configurations to distances of no more than 50 miles.

ASYNCHRONOUS REPLICATION

To enable longer distance configurations that could ensure survivability even in the worst disasters, asynchronous replication was developed. Asynchronous replication decoupled the write to the secondary site from the write to the primary site, allowing the primary application to function at local disk write speeds. Changes were then picked up from a local queue and asynchronously written to a target at a secondary site. In these configurations, the primary application continued to perform as if replication were not in use, and there were no limitations in terms of distance. The downside to this approach was that the target lagged the state of the source by some number of writes, depending on the write volume, network bandwidth available, and so on. Still, asynchronous replication proved an extremely effective solution for all requirements except for real-time failover. However, decoupling the source write from the target write introduced a major data integrity issue: Primary site write ordering was no longer automatically preserved. This posed considerable challenges to asynchronous replication vendors.

3 THE CHALLENGES OF ASYNCHRONOUS REPLICATION

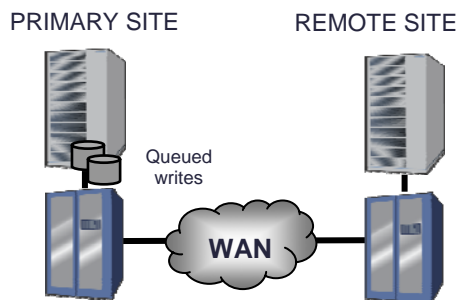
Decoupling the source and target writes raised the issue of write ordering, a problem that is at the very crux of data consistency. In synchronous replication, writes are in effect ordered by the requesting application, because each write is completed at both sites as it is requested. The application cannot proceed to the next write until it has received a write acknowledgement from both sites. Any write dependencies are managed by the application so that the write requests flow through both sites in the appropriate order. When the writes are decoupled, however, any sense of write ordering imposed by the primary site application may be lost due to the uncertainties of internal server processing times and network latencies. At this point, second site write ordering is no longer deterministic, and mechanisms must be put in place to consistency approaches in use today, each of which has unique implications for the overall solution.

NO MECHANISM

Low-end or departmental solutions designed for local applications may use this approach. Once the writes are decoupled, the write to the remote site may be delayed by primary server workload (for example, the CPU processing the remote write may be much slower than the CPU processing the local write) or by network latencies introduced by routing, failures, and so on. For whatever reason, if writes get out of order before they are applied at the remote site, then data is corrupted and will cause problems during recovery (corrupted or lost data). Additionally, these products have no way to notify the user that data corruption has occurred, requiring additional effort to validate the recovery.

SERIALIZATION

This approach logs all writes in the order they are requested, often on nonvolatile storage, at the primary site and then sends the writes across the network one at a time to ensure that they are applied remotely in the same order that they appear in the primary site log. This approach can maintain data integrity for applications running on a single device (typically a server with these types of products), but suffers from severe performance and scalability problems. Because writes are serialized, and remote writes have significantly more latency than local writes, local logs can get quite large and remote sites can lag quite far behind. This architecture does not take advantage of additional network bandwidth (if available) to scale performance. Transient network partitions, not an uncommon occurrence in wide area configurations, tend to exacerbate this problem even more. These types of products are not a good fit for environments that have performance and/or



- Disk-based asynchronous replication attempted to address the distance and performance problems
- Remote site may lag the data state of the primary site
- This approach imposed a variety of unexpected tradeoffs, which kept it from being widely deployed in the open systems market

Figure 3) Asynchronous replication solved the distance problem but imposed a variety of undesirable tradeoffs on the end-user

NO MECHANISM

Cannot guarantee write order fidelity at the remote site; used in low end nonenterprise products.

SERIALIZATION

Serializes all writes to maintain write order fidelity.

SINGLE DEVICE TIME STAMP

Uses a time stamp to maintain write ordering within a single device (server, array, or appliance).

GROUP STAMPING

Assigns all writes to a group and serially transfers groups to the remote site atomically.

MULTI DEVICE TIME STAMPING

Uses a time stamp to maintain write ordering across devices (servers, arrays, or appliances).

Figure 4) There are five general approaches to write ordering in use by asynchronous replication.

scalability requirements. Products that use serialization trade off performance and scalability to achieve data integrity.

SINGLE DEVICE TIME STAMPING

These implementations apply a time stamp for all writes that flow through a single device (server, array, or appliance) and then use that time stamp to reorder writes at the remote site appropriately. This approach does offer some scalability advantages over serialization because it can take advantage of more network bandwidth (if available) to achieve higher performance, and it can work well for environments that do not scale beyond a single device. However, applications like cluster file systems and some databases (IBM DB2 UDB Data Partitioning Feature) and n-tier applications like SAP and Peoplesoft exhibit multidevice dependencies and present problems for solutions that use single-device time stamping (or serialization) to maintain write ordering. In an appliance-based configuration, multiple servers and/or storage arrays can potentially be supported, but the appliance becomes a performance bottleneck. Adding more appliances reintroduces the cross-device problem, so performance cannot be scaled across appliances in a manner that maintains data integrity.

GROUP STAMPING

This approach in effect combines serialization and a form of single device time stamping, and to date requires purchase of proprietary array or appliance technology. Group time stamping suffers somewhat from the performance issues associated with serialization, and also manifests the same scalability problems associated with single device time stamping when deployed in enterprise environments. Group stamping assigns all writes that occur within a defined time period (30 seconds in its existing implementation) to a particular group. Writes are queued up at the primary site in a nonvolatile log, and at the end of the period are sent across the network atomically (as a group) and applied. Only the last change to a particular block during the defined time period is sent across, so some data reduction occurs as writes are collected at the primary. Subsequent groups cannot be sent across the network until all the updates associated with the previous group are completed. Hence there is a built-in data currency penalty equal to the group time period.

MULTIDEVICE TIME STAMPING

This approach offers a scalable, high-performance architecture that can maintain data consistency in multidevice (server, array, or appliance) configurations. It applies global time stamps to writes within each device, and then uses a network protocol to calculate global write ordering across devices. Writes from multiple sources can arrive at the remote site out of order, but they can be reordered according to the global time stamps to maintain data consistency. This architecture is very scalable and high performance because it can batch writes, sending them all across the network at once (to “catch up” after transient network partitions) as well as take full advantage of available network bandwidth to handle higher write volumes over time. It can be used in single server application environments as well as in multiserver environments like clustered file systems or parallel databases.

4 REPLICATORX™: THE OPTIMAL REPLICATION ARCHITECTURE

Although many vendors claim that their asynchronous solutions guarantee 100% data consistency, many of these solutions fall short of this goal when challenged by long-distance, low-network bandwidth or near real-time RPO requirements when replicating from a single source. For companies seeking to replicate from multiple sources, the field narrows dramatically. In fact, ReplicatorX is the only product available for use in open systems environments that can maintain data consistency in all of these scenarios (across one or more replication sources, short or long distance).

The ReplicatorX patent pending architecture maintains consistency between multiple replicating environments that support applications hosted on dependent servers or federated databases. In addition, ReplicatorX can maintain data consistency across any number of replicating nodes in a SAN fabric. With other solutions, data consistency is limited to individual servers, replication appliances, or appliance clusters, sacrificing scalability to maintain recoverability.

To meet the range of consistency requirements in today's IT environments, the ReplicatorX state-of-the-art architecture is designed to address the three critical components of a robust consistency engine:

- **Clock synchronization:** ReplicatorX maintains a global master clock within the replicating nodes of a consistency collection. This clock interacts with each individual system in the collection to reconcile any clock drift between the nodes and presents a single, synchronized time to manage operations.
- **Write-level time stamping and sequence tracking:** ReplicatorX applies global timestamps to individual I/Os within each device, and then uses a network protocol to calculate global write ordering across devices. Writes from multiple sources can then arrive at the remote site out of order, where before being committed to disk they are reordered according to the global time stamps and arrival of all writes up to a consistent point in time is verified.
- **Latency management:** Finally, ReplicatorX employs sophisticated algorithms to manage latency on two levels. First, latency in communication between the global clock and other system clocks in the consistency collection (sync latency) is tracked and applied in the synchronization calculation to ensure accurate global time. Second, the delta between I/O latency and sync latency is tracked to ensure that I/Os are not taking place during a clock synchronization operation. If this condition occurs, ReplicatorX suspends writes at the target until the condition clears and then updates the target from the last consistent point in time.

Without these three components in a replication architecture, consistency across all common replication scenarios in an enterprise IT environment cannot be guaranteed.

5 SUMMARY

In reviewing the architectures, it is important to note that only one does not impose a singular bottleneck of some kind at the primary site. Products that use no data integrity mechanism are not discussed here because they are not viable solutions for enterprise business. Serialization products can support many storage devices, but all traffic runs through a single server. Group time stamping products can support many servers, as can single device time stamping products, but they funnel all traffic into a single device at the primary site, whether that site is a server, an enterprise storage array, or an appliance. Only multidevice time stamping solutions can leverage a loosely coupled architecture at the primary site to avoid creating a performance bottleneck in a single physical device. This many-to-many design is another reason why multidevice time stamping architectures are much more scalable than competing architectures.

All asynchronous replication products can solve one of the key problems with synchronous replication—the inability to support long-distance configurations without significant impact on primary application performance. Architectural implementations to maintain data integrity in asynchronous replication environments may impose significant tradeoffs in the areas of data integrity, data loss, performance, scalability, vendor lock-in, and total cost of ownership.

