

StorTrends Whitepaper

“Die Synchrone Replikation”

Ajit Narayanan, © American Megatrends Inc.
Winfried Proehl, © American Megatrends International GmbH
12.12.2007

INHALTSVERZEICHNIS

KAPITEL 1: EINLEITUNG	3
KAPITEL 2: REPLIKATION: KONZEPTE	4
KAPITEL 3: VERSCHIEDEN ARTEN DER REPLIKATION	6
KAPITEL 4: KONSISTENZ DER DATEN	10
KAPITEL 5: REPLIKATION MIT DUAL DIALECT SERVERN	14
KAPITEL 6: SYNCHRONE REPLIKATION	15
KAPITEL 7: ZUSAMMENFASSUNG	24
KAPITEL 8: SCHLUSSBEMERKUNG	26
KAPITEL 4:	27
TRADEMARKS AND COPYRIGHT ACKNOWLEDGEMENTS	27
FOR ADDITIONAL INFORMATION	27
LIMITATIONS OF LIABILITY	27
LIMITED WARRANTY	27
REVISION HISTORY	27

1. Einleitung

Der Wert der Daten übersteigt den Wert der Systeme auf denen Sie gespeichert, erzeugt und bearbeitet werden. Daher ist ihr Verlust schmerzvoller als der rein materielle Schaden im Falle einer Katastrophe. Viele Unternehmen sind so eng an ihre Daten gebunden, dass der Verlust das Überleben der Firma in der bestehenden Form unmöglich macht. Um sich von dieser Katastrophe zu schützen, werden Daten lokal getrennt mehrfach gespeichert. Hierbei wird die Kontinuität der Geschäftsabläufe weiterhin gegeben, selbst wenn ein Standort vollständig ausfällt. Diesen Prozess des Übernehmens des duplizierten Datenbestandes von einem Server zu einem anderen nennt man Replikation oder Remote Mirroring. Die eingebundenen Server werden hierbei als 'Primary' und 'Secondary' Server, oder 'Primary' und 'Replica' Server bezeichnet.

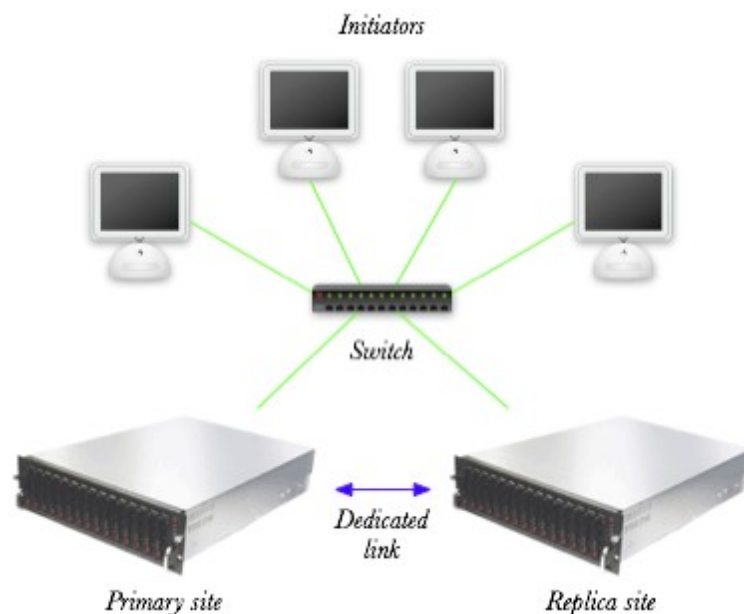


Abbildung 1: Typischer Aufbau der Replikation

Replikation unterscheidet sich von üblichem Backup (wie z.B. auf Bandlaufwerke) durch Gleichzeitigkeit, d.h. Online während der produktiven Zeit ohne Einwirkung auf die Produktivität. StorTrends Storage Appliance Server berücksichtigen die Bedeutung der Replikation und haben fortgeschrittene und neue Merkmale, welche ein Maximum an Datenschutz und höchste Flexibilität der Anwendung bieten. Dieses Dokument beschreibt die Konzepte, die für die Replikation angewandt werden, sowie die spezifischen Neuerungen in StorTrends, die diese Konzepte wirksam machen.

2. Replikation: Konzepte

Der zugemessene Wert der Daten ist äußerst unterschiedlich zwischen Unternehmen und auch zwischen verschiedenen Anwendungen innerhalb eines Unternehmens. Daher gibt es abgestimmte und stark differenzierte Konzepte für eine Replikation der Daten. Diese unterschiedlichen Lösungsansätze wägen zwei kritische Werte gegen die Kosten der Lösung: RPO = Wiederherstellungszeitpunkt (Recovery Point Objective) und RTO = Wiederherstellungszeit (Recovery Time Objective).

Um diese Werte besser zu verstehen, ist es notwendig den Prozess der Replikation im Sinne der Geschäftskontinuität zu betrachten. Eine übliche Einrichtung der Replikation besteht aus einem lokalen Server als erste Datenquelle ("Primary") für alle Lese- und Schreibzugriffe und einem Remote Server, der als Back Up Speicher für alle Daten des "Primary" zur Verfügung steht. Die Verbindung zwischen den beiden kann auf unterschiedliche Art hergestellt werden, abhängig von der Entfernung und den vertretbaren Kosten: entweder über ein eigenes Netzwerk oder über eine optische Verbindung oder über ein mitbenutztes bestehendes Netzwerk. In vielen Anwendungen ist die Verbindung zwischen dem Primary und Secondary Server ein bemerkenswerter Anteil der gesamten Kosten. Diese können für den Transport der I/Os zwischen Primary und Secondary durchaus höher sein als der, der I/Os von und zu den Anwendungen von und zu der Primary.

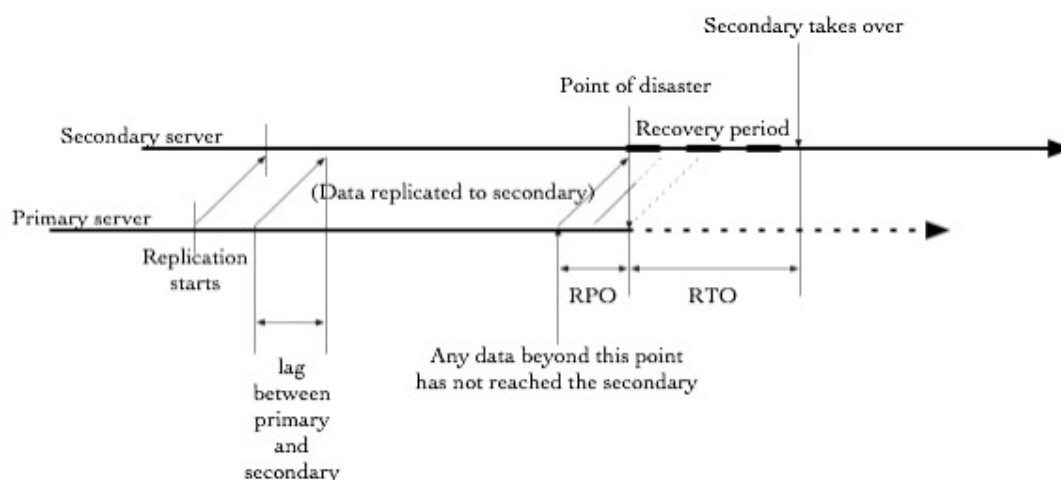


Abbildung 2: Replikation, Disaster Recovery, RPO und RTO

Sollte der Primary vollständig ausfallen, muss die Geschäftskontinuität durch den Secondary hergestellt werden. Dies wird manuell getätigt durch Zugriff auf die entsprechenden Volumen des Secondary und durch den Neustart der hiermit nun verbundenen Applikationen. Dieser Vorgang wird als Failover bezeichnet. In diesem Zustand agiert der Secondary als Empfänger der I/Os von den angeschlossenen Clients. Die Kosten für die Geschäftskontinuität während des Zustandes des Failover sind jetzt durch die erhöhten Kosten der Verbindung zum Secondary höher als zuvor. Jedoch sind diese Kosten bei weitem geringer als die, die mit einem Schaden der Geschäftsunterbrechung verbunden sind.

Wenn der Primary wiederhergestellt ist, entweder durch Reparatur und Wiederherstellung oder durch Neuinstallation eines Servers, wird man allein wegen der Kosten die Anwendungen vom Secondary auf den Primary zurück übertragen. Dieser Prozess ist gleichbedeutend einem weiteren Failover und wird Fail-back genannt. Fail-back wird ebenfalls mit einer Unterbrechung getätigt indem die Volumen des Primary wieder verbunden werden und die Applikationen wieder neu von diesen gestartet werden.

Zusammenhängend werden hier zwei wichtige Maßstäbe für die Wirksamkeit der Replikation definiert. Der erste ist die Dauer zwischen dem Ausfall des Primary und des Übernehmens des Secondary im Failover. Diese Zeitspanne wird als Wiederherstellungszeit = RTO = *Recovery Time Objective* bezeichnet. Der zweite ist definiert durch die Größe des vertretbaren Datenverlustes. In einigen Situationen (z.B. solchen wie Source Code Control) ist ein Datenverlust von einigen Minuten vertretbar, da die Wiederherstellung nur geringen Aufwand erfordert. In einigen anderen Situationen hingegen, wie bei Elektronischen Buchungen im Bankwesen und bei Reservierungen bei Fluglinien, kann sogar eine einzelne Sekunde Unterbrechung nicht wieder gut zu machende Schäden verursachen. Es gibt viele Situation zwischen diesen beiden Fällen, so viele wie es unterschiedliche Datenbanken und deren spezifische Einsatzgebiete hat. Die Höhe des tolerierbaren Datenverlustes wird gemessen als die Zeitspanne der Überbrückung der Unterbrechung als RPO = *Recovery Point Objective* = Wiederherstellungszeit.

3. Verschiedene Arten der Replikation

Es wurden unterschiedliche Lösungen verwirklicht, die RPO, RTO und die jeweiligen Kosten gegeneinander abwägen. Die teuerste Art der Replikation – mit einer verzugsfreien RPO und RTO von Null – wird *active-active Clustering/Mirroring* genannt. Bei dieser Form der Replikation sind beide, der Primary und Secondary aktiv und zur selben Zeit funktionsfähig. Die Anwender und Anwendungen sind gleichermaßen an beide Server angeschlossen und die Server gleichen sich jederzeit miteinander konsistent ab. Wenn ein Server ausfällt, ist dies als würde nur eine überflüssige Komponente ausfallen, denn der zweite Server hat nahtlos die gesamten Funktionen weiterhin aufrechterhalten – ohne die Notwendigkeit eines händischen Eingreifens für die Herstellung eines Failover.

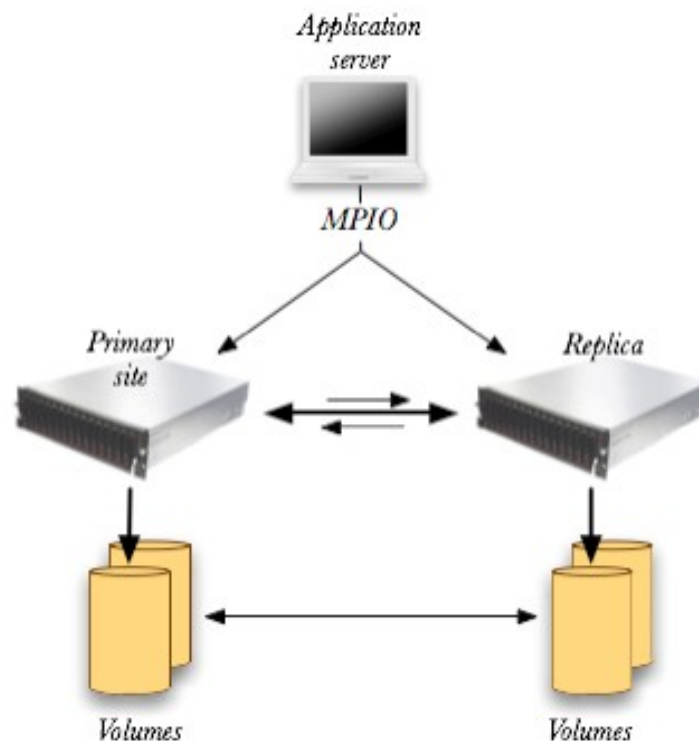


Abbildung 3: Active/Active Mirroring oder Clustering

Am nächsten ist die synchrone Replikation. Bei dieser Methode ist nur der Primary mit den Anwendern / Anwendungen verbunden. Jede Veränderung des Datenbestandes (Schreibbefehl) des Primary wird ebenfalls auf dem Secondary vorgenommen und der

Schreibzugriff wird erst dann als ausgeführt gemeldet, wenn er auf beiden Servern durchgeführt wurde. Auf diese Art wird gewährleistet, dass alle Schreibbefehle auf beiden Servern durchgeführt wurden und wann immer ein Server ausfällt wird der andere die gleiche Datenverfügbarkeit gewährleisten.

Der RPO (Recovery Point Objective) der synchronen Replikation wäre daher gleich Null. Denn, es mag dennoch nötig sein, dass man händisch einen Failover herbeiführen muss um die Anwendungen vom Primary zu trennen und mit dem Secondary zu verbinden. Dadurch kann der RTO (Recovery Time Objective) bis zu einigen Stunden in Anspruch nehmen

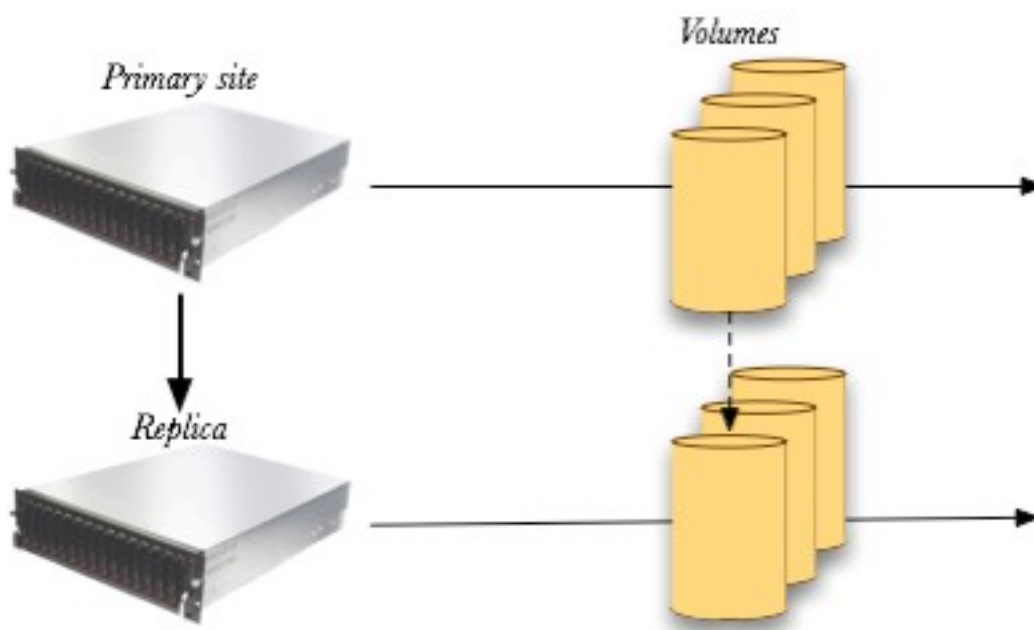


Abbildung 4: Synchrone Replikation

Die Einbindung von Synchroner Replikation ist kostspielig, da diese eine schnelle Verbindung mit hohem Datendurchsatz zwischen dem Primary und Secondary erfordert um Störungen bei der Übermittlung auszuschließen. Bei Betrieb von Standorten mit hoher Entfernung, z.B. kontinentübergreifend, können solche Kosten für eine separate hochleistungsfähige Verbindung bis über eine Million Euro betragen. Daher mag man stattdessen der Asynchronen Replikation den Vorzug geben. Bei dieser Methode werden die I/Os nicht gleichzeitig von den Anwendungen zum Primary und zum Secondary gesandt, sondern Sie werden vom Primary erst für einige Sekunden zwischengespeichert und doppelte Schreibzugriffe werden bis zu einem bestimmten Grad vermieden und dann gesammelt mit einer Verzögerung an den Secondary übertragen. Da das Zwischenspeichern von Daten eine bessere Nutzung

der Übertragungsbreite ermöglicht und da zwischengespeicherte Daten komprimiert und für die Übertragung optimiert werden können, darf die Übertragungsleistung bei der Asynchronen Replikation erheblich geringer sein als bei der Synchronen Replikation. Dadurch ergibt sich auch eine wesentliche Reduzierung der Kosten durch moderate Anforderungen an die Übertragungsgeschwindigkeit. Andererseits zahlt man für diese geringeren Kosten einen höheren Preis durch gestiegene RPO: im Falle eines vollständigen Ausfalls des Primary werden alle zwischengespeicherten Daten verloren sein und der Secondary wird sich nach Inbetriebnahme mit einem Datenverlust von einigen Sekunden gegenüber den Anwendern darstellen.

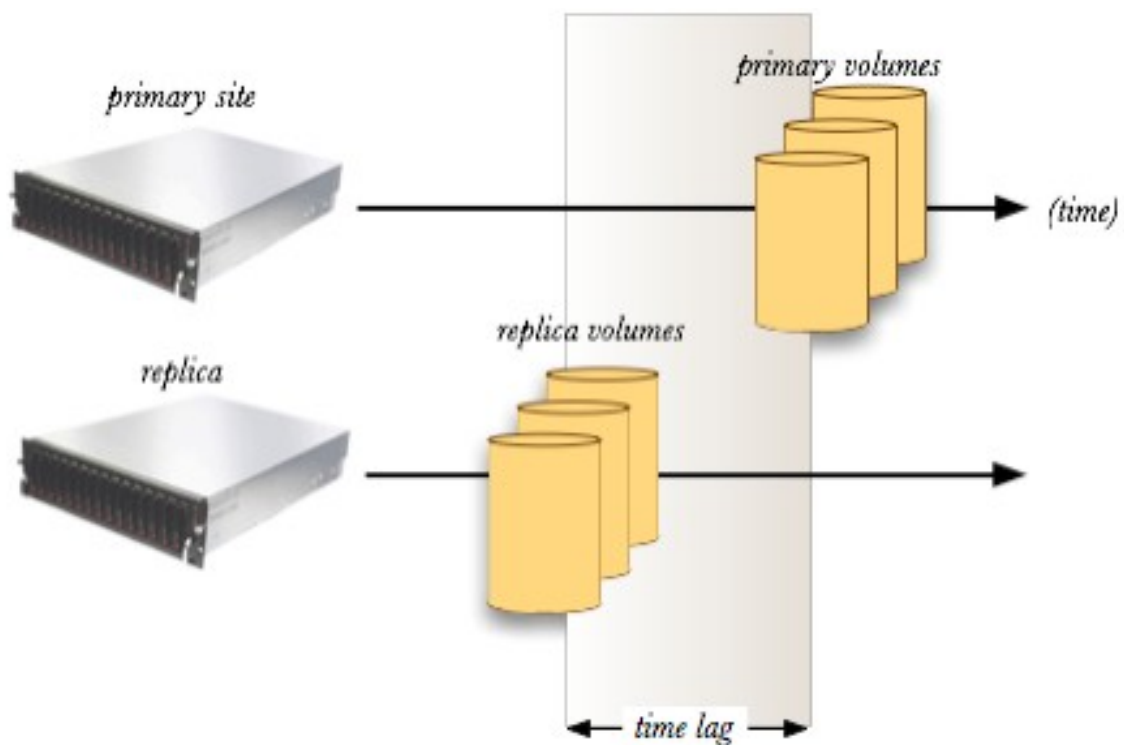


Abbildung 5: Asynchrone Replikation

Einen Schritt unter der asynchronen Replikation gibt es die *Snapshot-Assisted Replication*. Anstelle die Daten zwischen zu speichern um die Datenübertragung effizienter zu machen, nimmt diese Bezug auf Snapshots, Momentaufnahmen eines Datenbestandes um die Veränderungen zwischen den Snapshots als Abbild des Datenbestandes vom Primary zum Secondary zu ermöglichen. Ein Snapshot repräsentiert die Summe aller Veränderungen zum vorhergegangenen Snapshot und eliminiert somit alle doppelten Schreibzugriffe während dieser Zeit. Da beide, der alte und der neue Datenbestand verfügbar sind, ist es möglich wesentlich effizienter die

zu übertragenden Daten zu komprimieren. Der RPO steigt jedoch bei dieser Methode in eine Größe von mehreren Minuten wenn nicht sogar Stunden. Daher wird sie auch besser Periodische Replikation genannt. Der RTO jedoch mag häufig bemerkenswert geringer ausfallen als bei der Synchronen oder Asynchronen Replikation.

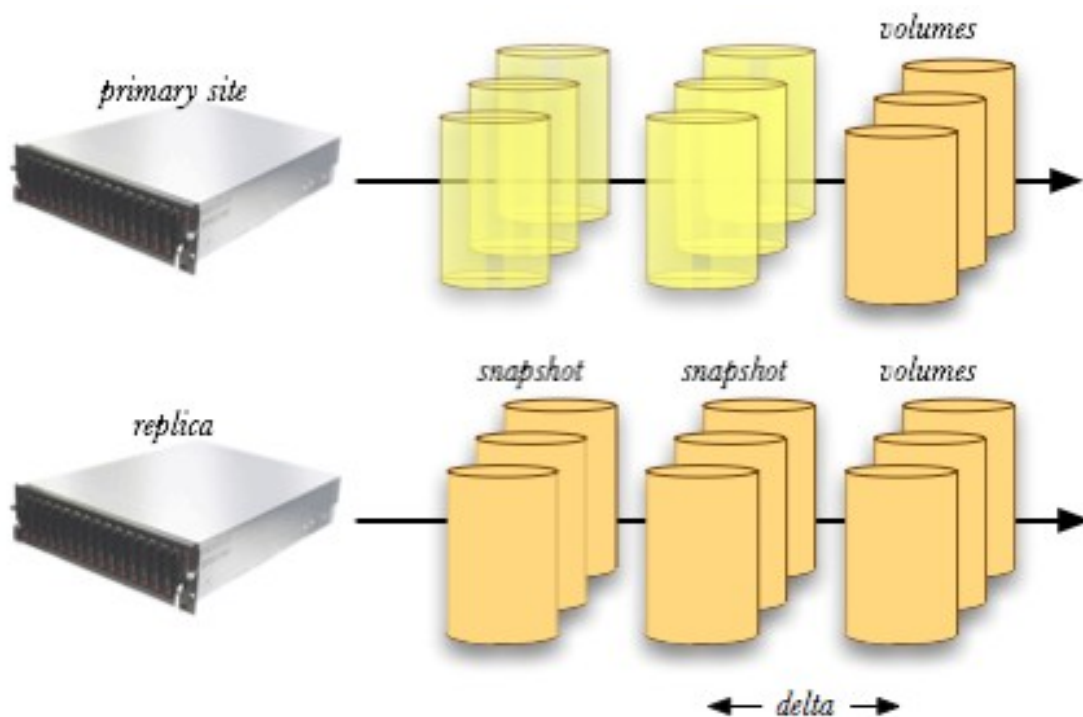


Abbildung 6: Snapshot-Assisted Replikation

Anwender, die nicht einmal in Snapshot-Assisted Replikation investieren, nehmen Zuflucht zu alternativen Möglichkeiten für den Schutz ihrer Daten, wie z.B. herkömmliches Sichern über Bandlaufwerke.

Alle StorTrends Systeme unterstützen synchrone, asynchrone und Snapshot-Assisted Replikation. Da keine Redundanz durch Aktive-Aktive Controller unterstützt wird, beschreibt dieses Dokument in einem gesonderten Kapitel ein Szenario mit synchroner Replikation, das denselben Effekt zeigt.

4. Konsistenz der Daten

Ein Szenario: der Primary Server ist ausgefallen, der Secondary Server ist erkannt und der Administrator hat den Failover eingeleitet und die Anwendungen neu gestartet – doch jetzt kommt der kritische Moment, in dem es sich zeigt ob die Replikation erfolgreich ist und der Geschäftsprozess fortgeführt wird oder ob dieses Investment nur ein kolossales Geldverschwenden war. Werden die Prozesse fortgeführt und kann eine Wiederherstellung vom Secondary Server erfolgen?

Auf den ersten Blick geht jeder davon aus, dass dem so sein müsste. Denn folgerichtig sind ja alle Daten parallel sowie auf den Primary Server auch auf dem Secondary Server enthalten. Jedoch ist das nicht ganz so einfach und wir bekommen es hier mit der Konsistenz der Daten zu tun, dem wichtigsten Aspekt bei der Replikation.

Warum sollte eine Anwendung nicht fortgeführt werden wenn sie von dem Secondary neu gestartet wird? Hierfür gibt es mehrere Gründe. Viele einfache Anwendungen können oft nicht weitergeführt werden wenn der Datentransfer plötzlich und gnadenlos unterbrochen wird, z.B. durch Stromausfall oder durch eine Katastrophe. Für solche Anwendungen ist die Unterbrechung des Datenflusses gleich einem Datenverlust. Solche Anwendungen sind nicht geeignet für eine Replikation. Glücklicherweise trifft man sie jedoch so gut wie nie in der Welt der Unternehmens – IT. Eine weitere Klasse der Anwendungen ist widerstandsfähig gegen Stromausfälle und diese Anwendungen sind die typischen, die innerhalb eines Unternehmens vorgefunden werden. Diese Anwendungen schreiben fortlaufende 'Check-Points' auf die Platten und markieren so konsistente Zustände, 'Consistent Points', im Datenfluss der Anwendung. Im Falle eines plötzlichen Stromausfalles oder Systemausfalles wird die Anwendung zu dem letzten Konsistenten Datenbestand gehen und von dort beginnen.

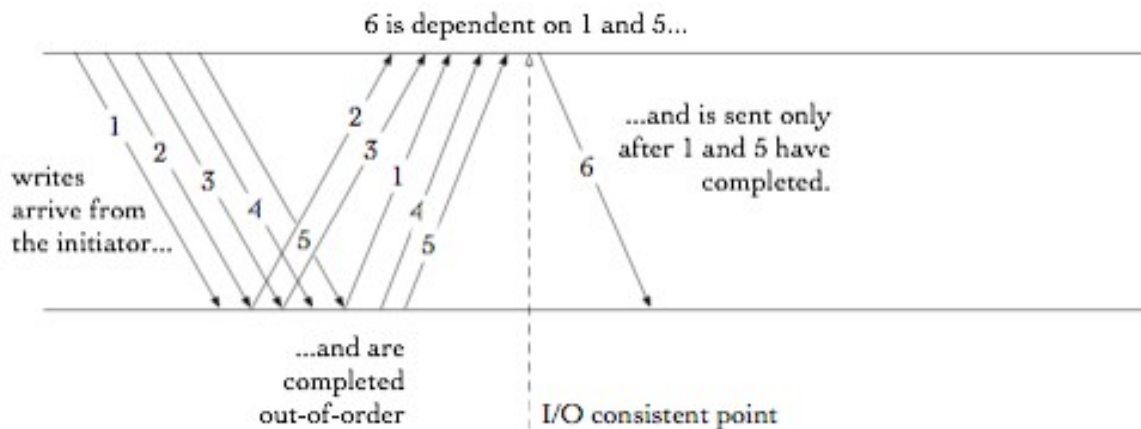


Abbildung 7: Abhängige Schreibzugriffe und I/O Konsistenz

Zwei schwerwiegende Probleme jedoch können die Fähigkeit der Anwendungen zu ihrem jeweiligen letzten konsistenten Datenbestand zurückzukehren stark beeinträchtigen. Das erste ist das Vorhandensein eines "Write-Back Cache" auf dem Storage Server. Der Write-Back Cache enthält die Schreibbefehle, die dem Storage System zugehen, und gibt diese in periodischen Intervallen oder bei nicht ausreichendem Hauptspeicher an die Platten weiter. Es ist hochwahrscheinlich, dass die Ordnung dieses Datenabflusses sich unterscheidet von der, wie sie ursprünglich geschrieben wurde. Daher kann es passieren, dass die „Check-Points“ der Anwendung, die die Vollständigkeit der Datensätze markieren vor den eigentlichen Daten abfließen und somit die Vollständigkeit markieren, die in Wirklichkeit nicht gegeben ist. Wenn die Anwendung in einem solchen Fall versucht das System wiederherzustellen, geht sie zu den letzten „Check-Points“ zurück, die jedoch keinen konsistenten Datenbestand markieren. Hierdurch misslingt entweder der Rollback oder es kommt zur Korruption der Daten.

Das andere Problem, das den Schutzmechanismus der Anwendung angreift, wird *Out-of-Order Writes* genannt. Dieser Fall tritt häufig auf, wenn Daten für die Übertragung vom Primary zum Secondary zwischengespeichert werden, wie dies bei der Asynchronen Replikation üblicherweise gemacht wird. Wenn ein Storage System über eine bestimmte Zeitspanne Daten zwischenspeichert, beseitigt es doppelte I/Os und überträgt dann zum Secondary Server in einer z.B. sequentiellen LBA Anordnung, wobei die Schreibbefehle beim Secondary in einer anderen Form ankommen als ursprünglich zum Primary. Auch hier kann es wieder zu „Checkpoints“ kommen, die einen nicht vorhandenen Datenbestand als in sich konsistent markieren.

Die Lösung für alle diese Probleme bei jeder Art der Replikation ist das Einfügen einer „Write-Order Fidelity“. Nehmen wir hierzu, dass eine Anwendung gleichzeitig drei Schreibbefehle absetzt und ein vierter sich auf diese drei bezieht. Wenn hierbei das Storage System über „Write-Order Fidelity“ verfügt, dann ist es gewährleistet, dass der vierte Schreibbefehl nicht vor den drei anderen erfolgen kann.

„Write-Order Fidelity“ gewährleistet, dass ein Storage System *Crash-konsistent* oder *I/O konsistent* ist und bleibt: es ist also so, wenn die Anwendung die Wiederherstellung nach einem Stromausfall oder Crash unterstützt, dann wird das Storage System dies nicht annullieren.

Eine andere Art der Konsistenz wird durch erweiterte Storage Architekturen wie bei StorTrends iTX gegeben. Diese wird als ein *Application-Consistent / Anwendungs-konsistentes* Verhalten bezeichnet. Wenn sich ein Storage System als Applikations-konsistent bezeichnet, heißt das, dass die Replikation dem Volumen des Secondary dann vertraut, wenn die Anwendung vorher von der Absicht der Replikation informiert wurde und die Anwendung stille hält, alle offenen File – Vorgänge geschlossen sind und das Gerät synchron ist. Selbst Anwendungen, die nicht zu einer Wiederherstellung nach einem Stromausfall fähig sind, können durch eine Applikations-konsistente Architektur der Replikation wieder hergestellt werden.

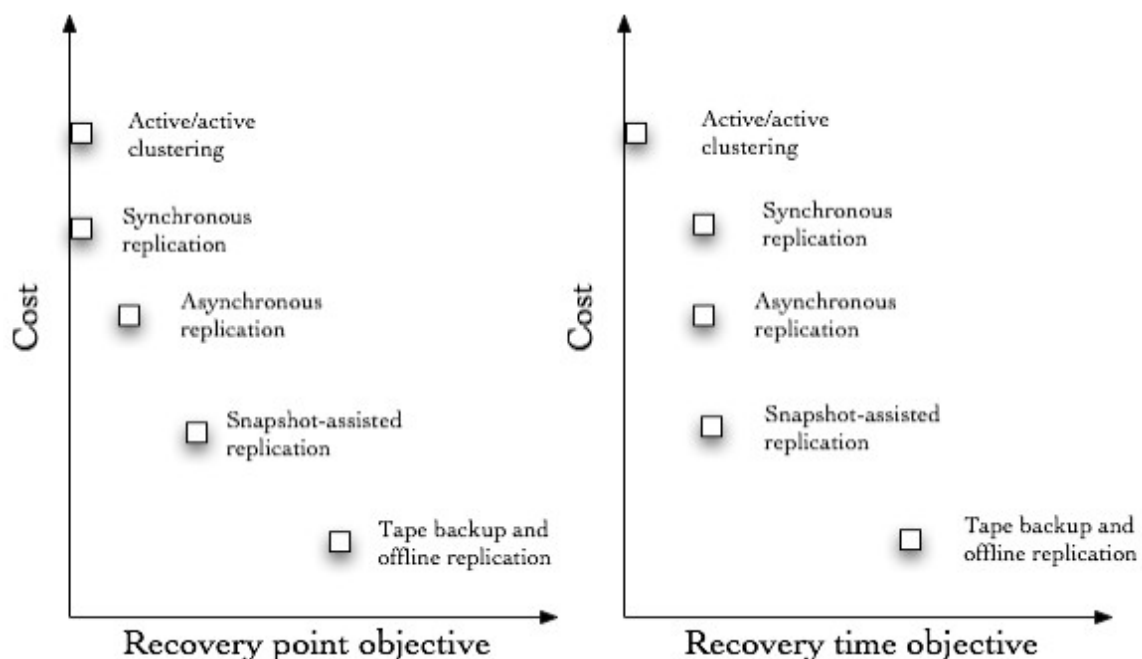


Abbildung 8: RPO und RTO der wichtigen Replikationsverfahren

Bei StorTrends sind alle drei Arten der Replikation sowohl Crash konsistent wie auch Applikations-konsistent (für die wichtigen Anwendungen). Dadurch haben Anwender einen größtmöglichen Spielraum für den Einsatz der StorTrends Storage Server für die meisten Applikationen und Programme.

5. Replikation mit Dual Dialect Servern

Dual Dialect Server sind Storage Server, die beides bieten, die Speicherung von Daten von File-basierten und Block-basierten Servern mit einer Verbindung sowohl als NAS wie auch als SAN. Die Replikation wird in beiden Modi unterstützt sowohl für die NAS wie auch SAN Daten. In beiden Fällen erfolgt die Replikation of Block-Level und nicht unbedingt notwendig über die Netzwerkschnittstellen.

Während die Replikation eines SAN-Volumens gänzlich ohne irgendeine Einbeziehung des Zwischenspeichers (Cache) der Server erfolgt, ist die Replikation der NAS Volumen nicht vollständig frei von Abhängigkeiten von zwischengespeicherten Daten. Dies beruht darauf, dass ein File System Cache häufig zwischen den Prozess des Schreibens vom Server auf das Storage System eingreift. Diese Eingriffe können verhindert werden indem der File-System Cache in einen Write-Through Modus konfiguriert wird. Zusätzlich muss berücksichtigt werden, dass Anwendungen, die auf einem File-System laufen, in der Regel gegen plötzlichen Stromausfall oder Crash durch einen Journaling Service geschützt sind. Die Gegenwart eines solchen Journaling Supports erlaubt eine synchrone Replikation unter üblichen Filesystemen wie XFS ohne eine Notwendigkeit eines erneuten Einrichtens des Filesystems.

6. Synchrone Replikation

Synchrone Replikation ist eine Methode um Daten zu replizieren so dass jeder Schreibzugriff auf den Primary gleichzeitig auch auf den Secondary gespiegelt wird selbst bevor dieser vom Initiator abgeschlossen wurde. Bei Anwendungen wie Elektronischer Handel oder Reservierungen bei Fluglinien, die ein hohes Maß an Schutz für ihre Daten erfordern mit einem RPO (Recovery Point Objective) von Null ergibt sich keinerlei Datenverlust zwischen den Beständen des Primary und Secondary, wann immer die Anwendung in den Failover Modus gesetzt wird. Die Realisierung der synchronen Replikation in StorTrends ist höchst innovativ und bietet einen soliden Schutz für alle Daten zu geringen Kosten.

Synchrone Replikation wird beeinträchtigt durch "Write Order Fidelity" Eingriffe wenn Daten in einer höheren Ebene zwischengespeichert werden. Daten, die in einem Write-Back Cache gehalten werden, können bei einem Disaster verloren gehen und hierdurch wäre die Replikation wertlos geworden. Um dieses Problem zu vermeiden, repliziert StorTrends die Daten der SAN Volumen bevor diese im Cache gespeichert werden und arbeitet unterhalb des Filesystems für Daten der NAS Volumen.

StorTrends ist für die synchrone Replikation vielseitig einsetzbar durch äußerst flexible Konfigurationsmöglichkeiten des Primary und Secondary zueinander. Sie können sich als Server im selben Subnet befinden und dies als I/O Netzwerk nutzen. Sie können über eine zugewiesene Verbindung mit einer zugesicherten Bandbreite und hoher Verfügbarkeit verbunden werden. Sie können über zusammengeschaltete redundante Netzwerkports verbunden werden um Netzwerkfehler zu überbrücken. Die Replikation kann auch durchgeführt werden von einem Volumen zu einem anderen Volumen im selben Server um zusätzliche Redundanz und Protektion zu gewinnen. Wenn sich diese beiden Volumen auf zwei separaten Containern innerhalb desselben Servers befinden, ist es einfach eine Wiederherstellung der Daten vom jeweiligen anderen Volumen durchzuführen wenn eine Festplatte oder Backplane ausgefallen ist. Die Replikation bietet hierbei wesentlich schnellere Wiederherstellung der Daten als dies üblicherweise ein RAID kann.

Die Leistung einer Maschine die sich im Status der synchronen Replikation befindet ist häufig dadurch erheblich gemindert. Das wird auch so erwartet, da die Schreibvorgänge auf beiden Servern, dem Primary und Secondary, durchgeführt werden müssen bevor sie zum Initiator abgeschlossen werden. Bei StorTrends

können jedoch die beiden Volumina, Primary und Secondary, mit Write-Back-Cache konfiguriert werden um die Leistung des Servers zu erhöhen.

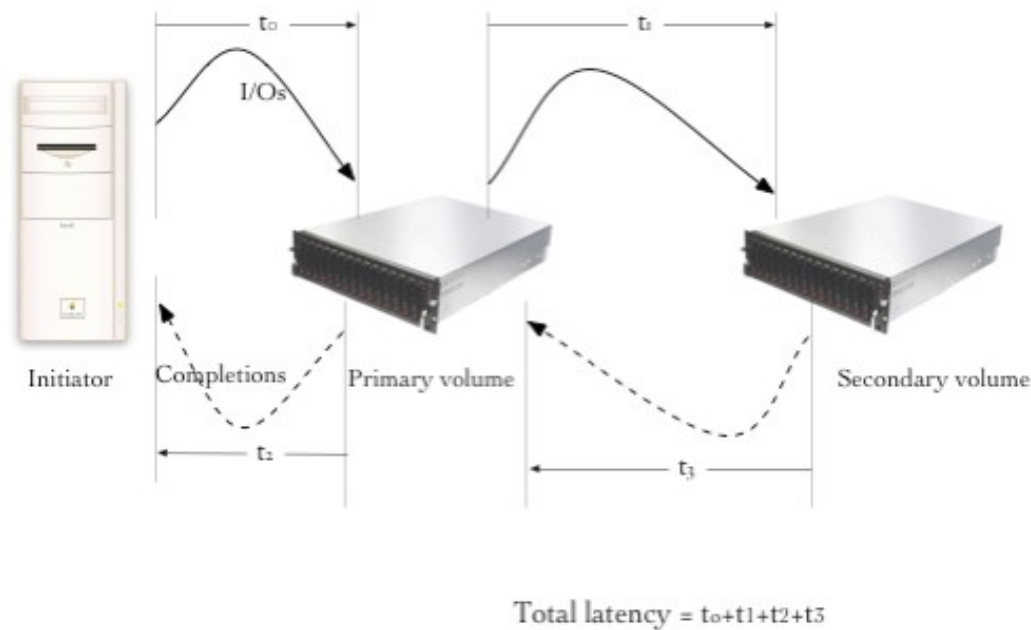


Abbildung 9: Latency und Performance der synchronen Replikation

Die synchrone Replikation von StorTrends unterstützt auch *Many-to-One* Replikation. Dies ist besonders attraktiv für viele Unternehmen die mehrere Volumina - auch auf mehreren Storage Servern - haben und die Replikation zentral auf einen Secondary Storage Server mit mehreren replizierten Volumina durchführen wollen. Die Replikation kann also so eingerichtet werden, dass der zentrale Secondary für alle Primary im Netzwerk derselbe ist.

Dies reduziert gewaltig die Kosten, denn der Secondary Server verlangt im Allgemeinen eine höhere Sicherheit und ein besser kontrolliertes Umfeld, daher ist es günstiger diese Funktionalität auf einem einzelnen Secondary abzubilden als jeden einzelnen Primary mit seinem Secondary zu versehen.

Ein weiterer Vorteil der "Many-to-One" Replikation ist, dass alle *Snapshots* und anderen Backup Prozesse jetzt über den Secondary Server zentralisiert werden können. Letzteres ist besonders attraktiv, da es eine Konsolidierung der Backup Aktivitäten auf einen Secondary Storage Server erlaubt, die ansonsten jeweils für jeden einzelnen Primary separat erfolgte. Tape Backups können nun von einem einzelnen Server gezogen werden und beinhalten dennoch die Backups aller separater

Volumen. Das Management des Backup Prozesses vereinfacht sich hierdurch gewaltig. Die Komplexität wird wesentlich reduziert.

Die Leistung ist weiter eingeschränkt durch neue Einwirkungen wie z.B. die Einrichtung von Snapshots auf dem Secondary Server. Die üblichen Mechanismen für Snapshots sind langsam – die Leistung geht bis um einen Faktor von 20 zurück wenn Snapshots aktiv sind. StorTrends bietet jedoch eine Snapshot Architektur mit einer geradezu gleich bleibender Leistung selbst wenn mehrere Snapshots gemacht werden. Ein Resultat hierbei ist die wesentliche Reduzierung der Latency bei Schreibzugriffen auf das Primary Volumen.

Da die synchrone Replikation von den Anwendungen bevorzugt wird, die die höchste Stufe der Datensicherheit verlangen, sind Möglichkeiten für eine Behandlung von verschiedenen Fehlern erwünscht. Stellen Sie sich z.B. *link failures*, Fehler in der Netzwerkverbindung, zwischen dem Primary und Secondary vor. Wenn dieses passiert, kann der Primary die I/Os zum Secondary nicht mehr vervollständigen, denn der Secondary kann jetzt nicht mehr erreicht werden. Jedoch um die Systemverfügbarkeit aufrecht zu erhalten ist eine Unterbrechung bzw. ein Abschalten der Operation und des Primary ausgeschlossen. Wie geht man mit dieser Situation heute um?

Eine Art der Fehlerhandhabung für unterbrochene Netzwerkverbindungen ist es das ganze Volumen als „failed“, nicht repliziert, zu markieren. Das gesamte Volumen wird somit als nicht synchronisiert betrachtet. Schreibzugriffe werden nun nur auf dem Primary zugelassen und wenn die Verbindung zum Secondary wiederhergestellt ist, wird die Replikation neu gestartet mit einer vollständigen Resynchronisation des gesamten Volumens.

Dieser Mechanismus ist verständlicherweise nicht in unserem Interesse. Die Verbindung bei einer synchronen Replikation ist häufig die höchste Ausgabe in der Liste der Gesamtkosten der TCO - Total Cost of Ownership; um eine volle Resynchronisation über die Netzwerkverbindung zu tätigen, wird eine enorme Bandbreite der Übertragung vorausgesetzt. Die Kosten der Übertragung sind entsprechend hoch. Ebenfalls steigt die Zeit der Resynchronisation je höher das Datenaufkommen ist – und während dieses Prozesses wird das System anfällig für Fehler und einen Ausfall. Dieser Mechanismus lässt sich verbessern indem die Resynchronisation nur die Differenz zwischen den Datenbeständen des Primary zum Secondary als Objekt handhabt. So ist es üblich die Daten des Primary Servers in einer bestimmten Auflösung zu lesen und darauf deren CRC Prüfsumme zu

berechnen, dieselbe CRC Prüfsummenberechnung vom Secondary anzufordern und nun beide Prüfsummen zu vergleichen und nur die Daten in die Resynchronisation einzubeziehen, bei denen die Prüfsummen nicht übereinstimmen. Dies ist ebenfalls eine sehr aufwendige Prozedur, die wenig Vergnügen bereitet. Die Verbindung wird hier zwar weitgehend entlastet jedoch ist der Zeitaufwand für die Rechenoperationen über jeweils das gesamte Volumen des Primary wie auch Secondary annähernd dem der bereits betrachteten Methode. Zusätzlich wird die Verbindung jetzt nicht nur für die Übertragung der Daten verwendet sondern auch für das gegenseitige Übertragen der Prüfsummen und schon wieder haben wir mehr Last auf unserer Netzverbindung. Noch dazu kommt, dass die Berechnung von CRC eine teure Angelegenheit ist und es mag trotzdem vorkommen, dass Datensequenzen dieselbe CRC haben, jedoch inhaltlich nicht übereinstimmen. All dies zusammen macht diese Methode der Resynchronisation etwas fragwürdig.

StorTrends nutzt daher eine innovative Methode für den Schutz vor Verbindungsfehlern und die Wiederherstellung durch Resynchronisation. Diese nennen wir *Tabbing* (von Tabelle). Immer wenn die Netzwerkverbindung unterbrochen ist, pflegt eine Datenstruktur eine Liste von exakt denjenigen Sektoren die I/Os empfangen haben. Dies wird in einer sehr feinen Auflösung getan mit bis zu maximal 64kB. Nach Abschluss der Resynchronisation werden tatsächlich nur diese Daten in der Verbindung übertragen. Die Auflösung kann durch einen neuartigen Algorithmus, *Collapsible Resync Tables* (CRT), noch weiter bis auf einen Sektor eingestellt werden. Hierdurch wird nur die absolut notwendige Datenmenge übertragen. Der positive Effekt ist eine wesentlich reduzierte Datenmenge (und damit Übertragungsbreite) sowie eine kurze Zeit für die synchrone Wiedererstellung der beiden Storage Server.

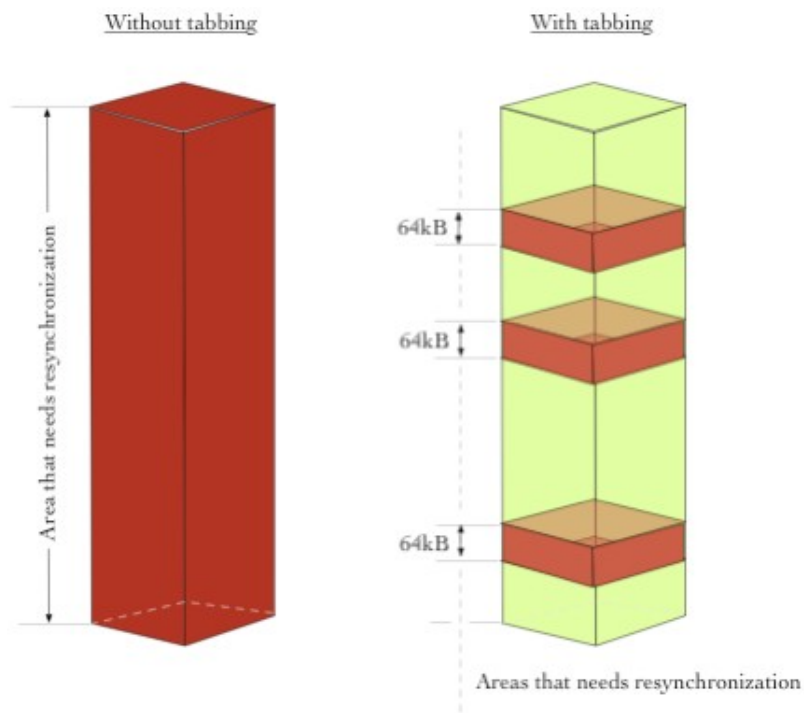


Abbildung 10: Vorteile des Tabbing bei der Resynchronisation

Die Resynchronisation läuft so, dass vom Primary gelesen wird und auf den Secondary geschrieben wird. Im Ergebnis erhält man während dieses Prozesses hierdurch eine höhere I/O Last auf die Verbindung und hierdurch eine Verminderung des I/O Durchsatzes vom Server zu den Initiatoren. Für einige Server ist es wichtig, dass der I/O Fluss weitgehend stabil bleibt, selbst im Falle einer Resynchronisation. Andere Anwendungen jedoch verlangen, dass der Prozess der resynchronisation so schnell wie möglich abgeschlossen wird. Um diesen unterschiedlichen Anforderungen gerecht zu werden, kann bei StorTrends der Anteil der Bandbreite und hierdurch auch die Zeit, die für die Resync zur Verfügung stehen soll je nach den Gegebenheiten angepasst werden.

Eine weitere Anforderung stellt sich für die synchrone Replikation für die Behandlung von plötzlichen Unterbrechungen der Versorgungsspannung. Durch Stromausfälle kann es tatsächlich dazu kommen, dass die Datenbestände zu einander inkonsistent werden. Wir erinnern uns daran, dass bei der synchronen Replikation ein Schreib – I/O auf beiden Servern, dem Primary und dem Secondary ausgeführt wird, bevor er dem Initiator als durchgeführt erscheint. Sollte die Versorgungsspannung dann ausfallen, wenn einer der beiden I/Os ausgeführt wurde, jedoch der andere noch nicht, dann tritt eine Situation auf, dass Daten auf einen Server geschrieben wurden, jedoch nicht auf den zweiten – und umgekehrt. Auch wenn dieser I/O dem Initiator

als nicht geschrieben mitgeteilt wurde, kann dies eine Korruption der Daten zur Folge haben, wenn der Zustand des Failover eingeleitet wird.

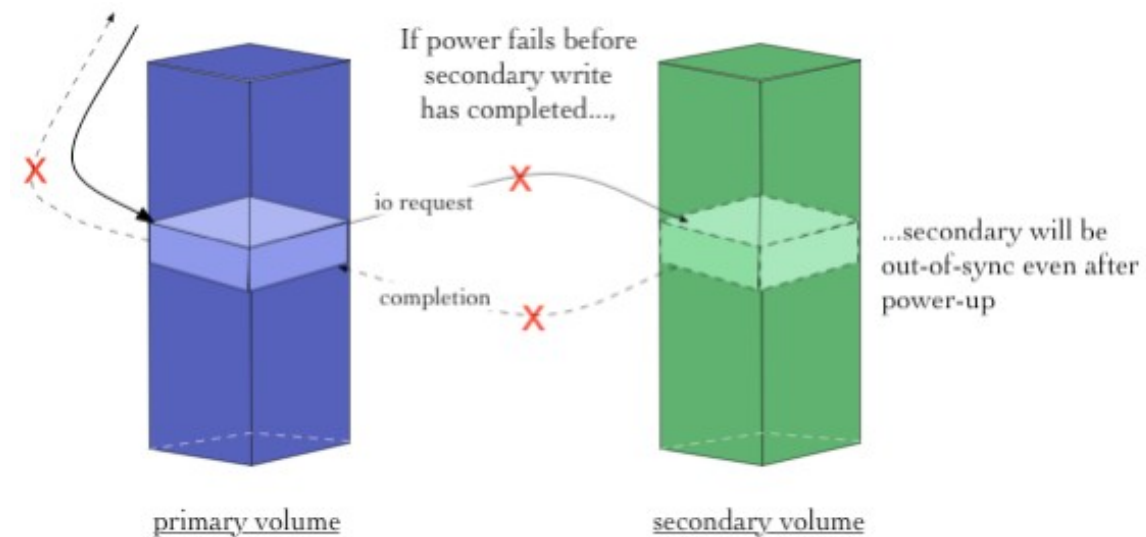


Abbildung 11: Verlust der Datenintegrität bei Stromausfall

Um diesen Prozess reibungslos durchzuführen, führt StorTrends ein *Write-Intent Logging (WIL)* vor jedwedem Schreibzugriff, gleich ob zum Primary oder Secondary, durch. Write-intent logging sind Einträge in ein Log von allen offenen ausstehenden Schreibzugriffen auf ein zu replizierendes Volumen. Beim Hochfahren nach einem plötzlichen Spannungsausfall werden diese Logeinträge berücksichtigt um offene, noch nicht abgeschlossene Schreibzugriffe festzustellen und eine Resynchronisation dieser LBA (Logical Block Address) auszuführen.

Der Vorgang des Write-Intent Logging wird sehr effizient in die Prozesse eingebunden, bei einer äußerst geringen Abnahme der Gesamtleistung von bis zu maximal 4 %. Dabei kann Write-Intent Logging in Systemen mit USV – Absicherung selektiv zugeschaltet werden, wenn die USV signalisiert, dass ein Übergang von der primären Spannungsversorgung auf die Notversorgung stattfindet und entsprechend vor einem Shutdown des Systems warnt. StorTrends bietet diese Unterstützung.

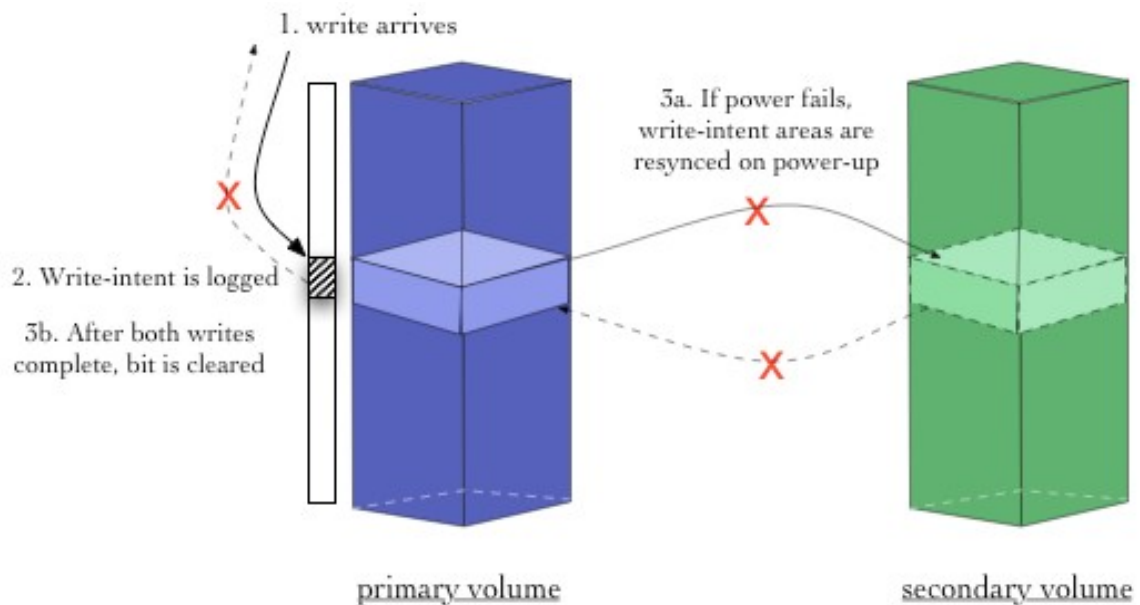


Abbildung 12: Write-Intent Logging für Spannungsausfälle

Zusätzlich zur Behebung von Verbindungs- und Spannungsversorgungsfehlern kann StorTrends gegen Mediafehler auf dem Primary Server wirken. Wenn ein Lesezugriff der Anwendung auf den Primary Server durch einen Mediafehler misslingt, wird üblicherweise dieser Sektor als schlecht markiert und beim nächsten Schreibzugriff korrigiert, jedoch bei Verlust dieses speziellen I/O. StorTrends jedoch hat den Vorteil, dass diese Daten auch auf dem Secondary Server vorhanden sind. Wenn also ein Mediafehler entdeckt wurde, werden die Daten vom Secondary Server geholt und zur Verfügung gestellt, während gleichzeitig der Primary Sektor mit dem Mediafehler als Bad Block markiert wird und die Daten wieder geschrieben werden in Übereinstimmung mit dem I/O. Hierdurch wird der Initiator nicht einmal einen Mediafehler auf den Festplatten bemerken.

Die synchrone Replikation ist zweifellos die erste Wahl für hoch kritische Vorgänge, denn sie bietet die kleinste RPO von Null. Jedoch wenn RTO gleichgewichtig sein soll, dann bevorzugen Anwender eine Konfiguration mit einer Spiegelung der Daten in einem Aktiv-Aktiv Modus, da hierbei eine sofortige Wiederherstellung möglich ist. Obwohl StorTrends dieses Aktiv – Aktiv Cluster derzeit noch nicht unterstützt, ist es trotzdem möglich etwas Gleichartiges durch die Synchrone Replikation zu erreichen. Wir nennen dies die *High-Availability Configuration* – Hochverfügbarkeitskonfiguration der StorTrends.

In einer Hochverfügbarkeitskonfiguration erhalten beide StorTrends Server, der Primary und der Secondary I/Os vom Initiator, jedoch bieten sie unterschiedliche LBA in jedem System. Die Unterscheidung zwischen 'Primary' und 'Secondary' wird in dieser Konfiguration etwas unklar – für unterschiedliche Bereiche können sich beide Server als der "Primary" darstellen, obwohl sie zueinander die Daten im synchronen Modus spiegeln. Ein Agent im Initiator sorgt dafür, dass immer die richtigen I/Os zum richtigen Server gelangen.

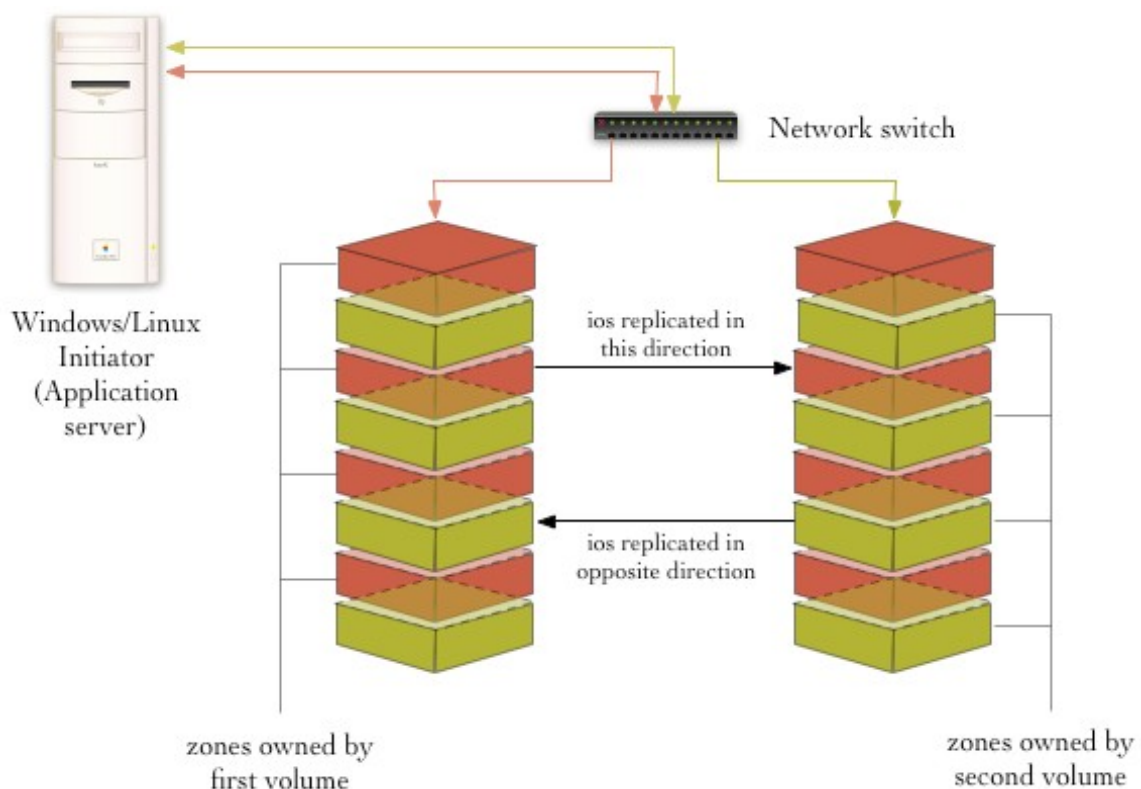


Abbildung 13: Hochverfügbarkeitsaufbau mit synchroner Spiegelung

Wenn demnach durch ein Desaster ein Server vollständig ausfällt, übernimmt die "lebendige" Maschine nahtlos das Vorrecht für die Bereiche, die der ausgefallenen Maschine zugeordnet waren. Bei Erkennung eines i/O Fehlers wird dieser Prozess durch den Agenten auf der Anwenderseite ausgelöst. Hier wird ein Microsoft Cluster Shared-Nothing Modell genutzt um eine korrekte Identifizierung und Reaktion im Disasterfall zu ermöglichen. Es ist möglich diese Konfiguration der synchronen Replikation von StorTrends für die Hochverfügbarkeit vieler verschiedener Anwendungen zu verwirklichen.

Diese Methode für das Erzielen der Hochverfügbarkeit hat sogar einige Vorteile gegenüber einer Redundanz von doppelten Controllern in einem Aktiv-Aktiv Szenario:

- Geringere Auswirkungen im Fehlerfall. Ein Ausfall innerhalb eines RAID-5 Verbundes bedeutet eine zusätzliche Verletzlichkeit des Systems bei Nutzung in einer Single-Box Lösung – jedoch bietet die Two-Box Lösung beides, die Redundanz durch RAID 5 und durch die Box-to-Box Redundanz.
- Schutz gegen den Ausfall von Komponenten, die nicht hot-swap fähig sind. Z.B. erlaubt die Box-to-Box Redundanz die Fortsetzung der Funktion des Gesamtsystems selbst bei Ausfall der Backplane.
- Schnellere Wiederherstellung. Da unser Resynchronisationsmechanismus alle Sektoren kennt, die nicht synchronisiert sind, erfolgt hier eine gezielte Wiederherstellung der Daten, die wesentlich schneller ist als der Wiederherstellungsprozess von einem RAID-5 Verbund.
- Multi-Site. Dies ermöglicht die gespiegelten Datenbestände an zwei unterschiedlichen Orten zu halten um auch vor dem Ausfall eines Ortes oder Gebäudes geschützt zu sein.
- Box Upgrades und Wartung. Es ist möglich jeweils eine Box für die Wartung und für Aufrüstungen aus dem Verbund zu entfernen und diese nach Abschluss wieder in den Verbund einzufügen ohne eine Beeinträchtigung der Funktionen des Gesamtsystems.

So bietet StorTrends mit seiner Architektur der Synchronen Replikation eine perfekte Lösung für eine große Anzahl unterschiedlicher Anforderungen und Szenarien.

7. Zusammenfassung

Die StorTrends Produkt Version 3.0 unterstützt alle drei Arten der Replikation: synchron, asynchron und Snapshot-Assisted. Sie unterstützt des weiteren Log-Assisted Replikation al seine Erweiterung der Asynchronen Replikation. Alle diese Arten der Replikation sind in StorTrends integriert und machen es einfach den jeweilig besten Replikationsmodus für eine bestimmte Anwendung zu wählen mit der Möglichkeit die gewählte Art der Replikation während des Betriebes zu ändern.

Zum Beispiel kann vom synchronen Betrieb in den asynchronen Betrieb umgestellt werden, wenn eine Verbindung als zu langsam festgestellt wurde oder wenn die Verzögerungen – Latency – für einen synchronen Betrieb unakzeptabel werden. Ebenso stellt sich die asynchrone Replikation auf den synchronen Betrieb ein, während der Resynchronisation um ein 'Constant Lag' Szenario zu vermeiden.

Die asynchrone Replikation kann in den Log-Assisted Replikationsbetrieb umschalten um die Resynchronisation zu vereinfachen wenn eine Verbindung für eine längere Zeitspanne unterbrochen ist. in order to simplify resync when a link has been lost for a long time. Die Log-Assisted Replikation und darauf die Resynchronisation bieten einen Grad hochauflösender Datensicherheit, da im Failover durch Roll Back entweder zu einem Applikations-konsistenten Zustand oder einen I/O-konsistenten Zustand zurückgegangen werden kann.

Sowohl die synchrone wie auch die asynchrone Replikation können in den Snapshot-Assisted Replikationsmodus übergehen, wenn ein schwerwiegender Fehler der Verbindung geschieht. Die Snapshot-Assisted Replikation kann ebenso in den synchronen oder asynchronen Betrieb überführt werden, wenn dies notwendig wird.

Verschiedene Operationen können gleichzeitig auf beiden Servern, dem Primary und dem Secondary, ausgeführt werden, unabhängig welcher Typ der Replikation ausgeführt wird – das Erstellen und Löschen von Snapshots, die Erweiterung des Volumens und sogar Rollback und Wiederherstellung.

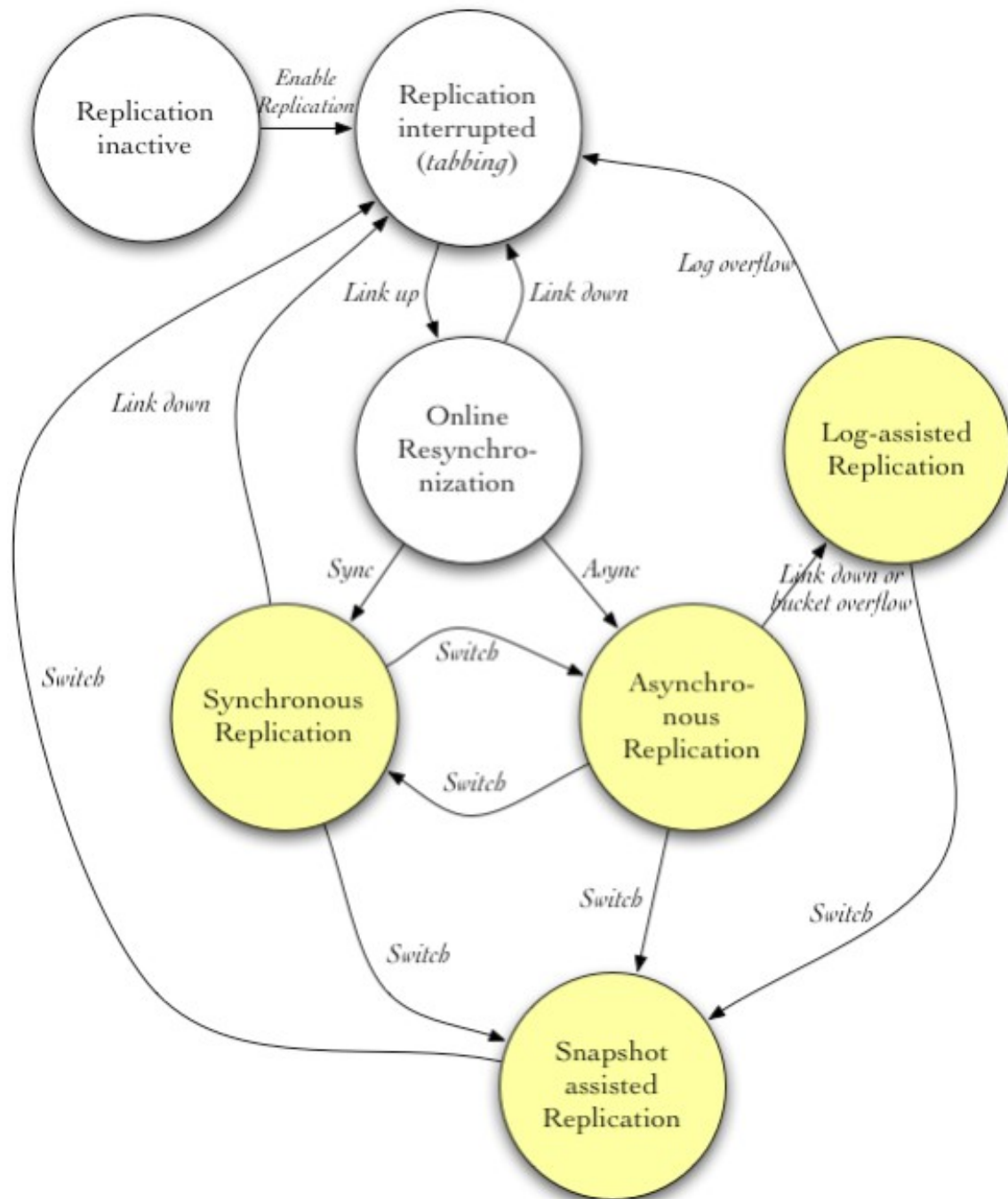


Abbildung 14: Zusammenfassung

Das Ergebnis der engen Kohärenz aller Leistungsmerkmale der Replikation von Stortrends ist erhöhte Leistung und Flexibilität für den Anwender und ein transparenter Gebrauch der Replikation mit größtmöglichem Schutz der Anwender im Katastrophenfall.

8. Schlussbemerkung

Dieses Dokument behandelt verschiedene mit der Replikation von Daten verbundene Konzepte und deren Anwendung in der heutigen Storage-Industrie. Es gibt eine Einführung in die Konzepte von RPO und RTO für die Messung der Vor- und Nachteile bei unterschiedlichen Anwendungen der synchronen, asynchronen und Snapshot-Assisted Replikation.

Im Besonderen wird die Replikation bei StorTrends behandelt mit den besonderen Eigenschaften der Fehlerbehandlung bei Unterbrechung der Verbindung und der Versorgungsspannung. Hierbei sind Möglichkeiten aufgezeigt um die Anforderungen an die Übertragungsbreite zu reduzieren und die Nutzung des Netzes wie auch die Nutzung der Rechenleistung bei ansteigender Gesamtleistung in Übereinstimmung zu bringen. Ebenso werden Beispiele gegeben, um vielfache zusätzliche Eigenschaften von anderen Anbietern in die StorTrends Anwendungen einzubinden.

Zusammenfassend kann man davon ausgehen, dass die Replikation von Daten einen Standard in Speichersystemen darstellen wird. Nicht nur in den heutigen High-End-Anwendungen der Großunternehmen, sondern zunehmend im Bereich der mittleren und kleineren Unternehmen, die dieses nun einfach und zu erschwinglichen Kosten für die Hochverfügbarkeit der Daten in Anspruch nehmen werden. Gerade durch eine Vielfalt unterschiedlicher jedoch einfach zu handhabender Methoden ist nahezu jede Anforderung aus diesen Marktsegmenten übererfüllt. StorTrends' Replikation ist schnell, sicher und reich an Eigenschaften, die einen zufrieden stellenden und transparenten Weg der Datensicherung in nahezu jeder Anwendung aufzeigen.

© Copyright 1998-2007 American Megatrends, Inc.
All rights reserved
American Megatrends, Inc.
6145-F Northbelt Parkway
Norcross, GA 30071

© Copyright 1998-2007 American Megatrends International GmbH
All rights reserved
American Megatrends International GmbH
D 81825 München , Wardeinstrasse 3 a

Trademark and Copyright Acknowledgments

This publication contains proprietary information that is protected by copyright. No part of this publication can be reproduced, transcribed, stored in a retrieval system, translated into any language or computer language, or transmitted in any form whatsoever without the prior written consent of the publisher, American Megatrends, Inc.

Trademarks and trade names may be used in this document to refer to either the entities claiming the marks and names or their products. American Megatrends, Inc. disclaims any proprietary interest in trademarks and trade names other than its own.

For Additional Information

Call American Megatrends at 1-800-246-8600 for additional information. You can also visit us online at ami.com.

Limitations of Liability

In no event shall American Megatrends be held liable for any loss, expenses, or damages of any kind whatsoever, whether direct, indirect, incidental, or consequential, arising from the design or use of this product or the support materials provided with the product.

Limited Warranty

No warranties are made, either express or implied, with regard to the contents of this work, its merchantability, or fitness for a particular use. American Megatrends assumes no responsibility for errors and omissions or for the uses made of the material contained herein or reader decisions based on such use.

Revision History

28.03.2007 Preliminary release
12.12.2007 Revised version

Bitte nehmen Sie für weitere Informationen Kontakt auf mit:
europe@ami.com - www.ami.de - www.ami.com