Only on NonStop

The HP NonStop Synchronous Gateway

Paul J. Holenstein
Executive Vice President of Gravic, Inc.

Note: This article is excerpted from a white paper entitled Using the HP NonStop Synchronous Gateway for Implementing Synchronous Replication that more fully describes data replication architectures and how the HP NonStop Synchronous Gateway (SG) can be leveraged to implement synchronous replication. Please contact the author at SBProductManagement@Gravic.com to receive a copy of that white paper. The focus of this article is specifically on the features provided by the new and evolving HP NonStop Synchronous Gateway, and some recent enhancements that HP has made to it. Note that this article contains forward-looking statements and describes potential technologies that are under consideration/development, and either do not exist or are only starting to become available. There is no guarantee that the technologies described in this article will become available on the HP NonStop platform. Coordinated Commits is a future technology and requires HP's NonStop Synchronous Replication Gateway. Specifications are subject to change without notice and delivery dates/timeframes are not guaranteed.

RPO, or the Recovery Point Objective, defines the amount of data that an application can lose when a disaster occurs.[1] It is a decision driven by the needs of the business.

There are two methods for data replication that affect the RPO – asynchronous replication and synchronous replication. These methods are shown in Figure 1.

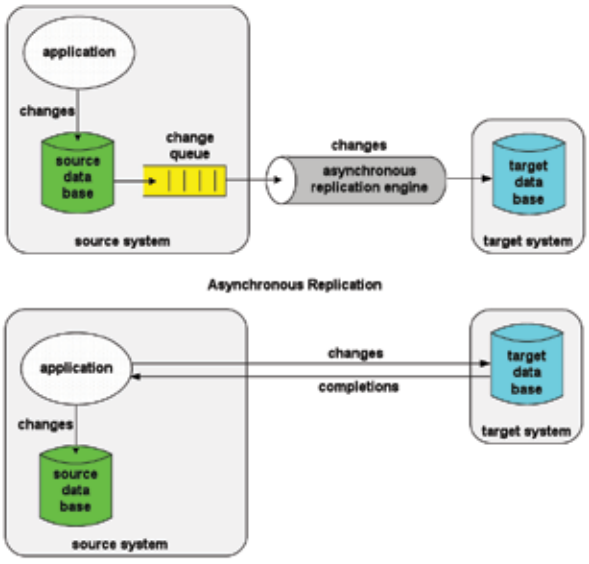

Figure 1: Data Replication Options

Asynchronous replication[2] replicates data under the covers and has no direct impact on the application. Typically, an asynchronous replication engine follows the database manager's transaction log and replicates changes to the target system as they are entered into the log. The target system's updates lag the source system, typically by a few seconds or less. This delay is known as replication latency. Any changes that are in the replication pipeline are lost in the event of the failure of the source system. Thus, the RPO for asynchronous replication is substantially equal to the replication latency – seconds or less.

Furthermore, in many cases, active/active systems (i.e., systems where the applications are actively running and processing requests on both systems) using asynchronous replication are subject to data collisions that can occur if the same data item is updated in both systems within the replication latency interval. Data collisions must be detected and resolved.

Synchronous replication[3] ensures that a database change is made to all databases in the application network before the change is completed. In other words, the change is applied everywhere or nowhere. In the typical form of synchronous replication, often referred to as Dual Writes, the application must complete an update to its local database and to its remote backup database before the individual update is considered complete by the application. In this way, every update is guaranteed to be made to both systems, and the RPO is reduced to zero. No data is lost when a failure occurs. Furthermore, data collisions cannot occur because the updated fields/records (columns/rows) are locked on both systems during the transaction. However, the greater the distance between the nodes, the greater the time that the application must wait for the remote update to complete. This delay is known as the application latency. For great distances, it can be measured in terms of tens or hundreds of milliseconds multiplied by the number of updates in the transaction. Though synchronous replication eliminates the data loss and data collision problems of asynchronous replication, the distance between the processing nodes is limited often to a distance that is relatively close, often not far enough to insure sufficient dispersion to provide geographic redundancy in the case of a regional disaster (e.g., earthquake or hurricane).

[1] Dr. Bill Highleyman, Paul J. Holenstein, Dr. Bruce Holenstein, Chapter 6 - RPO and RTO, Breaking the Availability Barrier: Survivable Systems for Enterprise Computing, AuthorHouse; 2004.
[2] For more information on asynchronous replication, see Dr. Bill Highleyman, Paul J. Holenstein, Dr. Bruce Holenstein, Chapter 3 – Asynchronous Replication, Breaking the Availability Barrier: Survivable Systems for Enterprise Computing, AuthorHouse; 2004.
[3] For more information on synchronous replication, see Dr. Bill Highleyman, Paul J. Holenstein, Dr. Bruce Holenstein, Chapter 4 – Synchronous Replication, Breaking the Availability Barrier: Survivable Systems for Enterprise Computing, AuthorHouse; 2004.

In this article, we will discuss another upcoming method of synchronous replication – Coordinated Commits. Coordinated Commits is a synchronous replication technique that substantially eliminates the distance problem. Thus, the distance between processing nodes is arbitrary and data loss and data collisions are prevented. In NonStop systems, Coordinated Commit technology uses HP NonStop's Synchronous Gateway. Therefore, let us first understand the HP NonStop Synchronous Gateway.[4]

The HP NonStop Synchronous Gateway

The X/Open Distributed Processing Model

The HP NonStop Synchronous Gateway follows the X/Open Distributed Transaction model that coordinates updates on multiple resources via a two-phase prepare/commit protocol.[5] As shown in Figure 2, an application uses resources such as databases, queues, etc. The

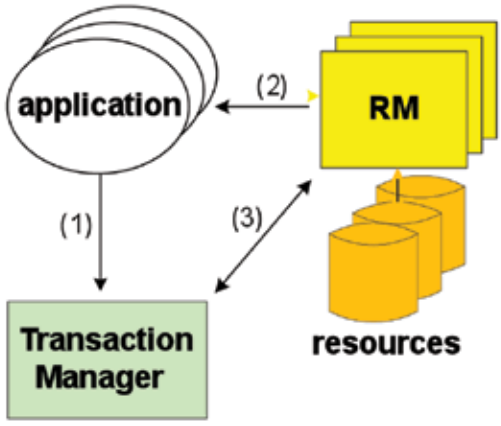

Figure 2: The X/Open Distributed Transaction Model

characteristics of each resource are hidden by a Resource Manager (RM) that provides a consistent interface to the application and to the Transaction Manager (TM).

When an application starts a transaction (1), the TM monitors its progress. Changes are made to various resources within the scope of the transaction by the application via the resources' RMs (2).

When the application directs the TM to commit the transaction (1), the TM first queries each RM to ensure that the RM is prepared to commit the transaction (3) (that is, the RM has safe-stored the transaction's changes and made them persistent). This is Phase 1, the prepare phase, of the two-phase commit protocol. If all RMs reply in the affirmative, the TM issues a commit directive to all RMs instructing them to commit the transaction (3). This is Phase 2, the commit phase, of the two-phase commit protocol. Should any RM reply negatively, the TM instructs all TMs to abort the transaction. Thus, the two-phase commit protocol guarantees that either all changes are made to all resources or that none are made.

The HP NonStop Distributed Processing Model

In NonStop systems, the Transaction Manager is the NonStop Transaction Management Facility (TMF). TMF has its own resource managers to manage the NonStop server disks via the disk processes (DP2s) associated with each disk. Thus, the RMs are not separate from the TM, as described by the X/Open model of Figure 2, but rather are included within TMF.

The HP NonStop Synchronous Gateway (SG) is an API and set of services that provides a special Resource Manager to include foreign resources within the scope of NonStop transactions (Figure 3). In the case of Coordinated Commits, the foreign resource of interest is the Coordinated Commit replication engine.

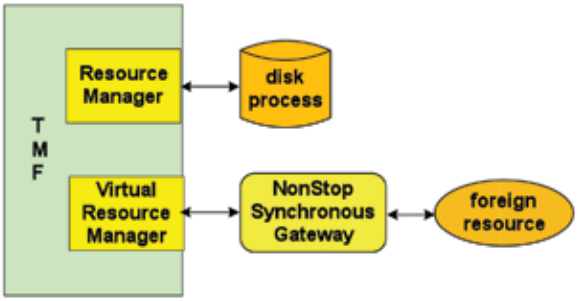

Figure 3: HP NonStop Synchronous Gateway

In the case of the SG, the foreign Resource Manager is a Virtual Resource Manager (VRM). The difference between a Resource Manager and a Virtual Resource Manager is that a VRM cannot participate in the recovery activities of TMF. If there is a system fault, it is the responsibility of the VRM to recover its own resource.

The VRM embedded in TMF talks directly to the SG. The SG provides an API that the foreign resource can use to join a TMF transaction and to participate in the two-phase commit protocol at the completion of the transaction.

Joining a Transaction

The steps required for a foreign resource to join a transaction are shown in Figure 4a and Figure 4b. A

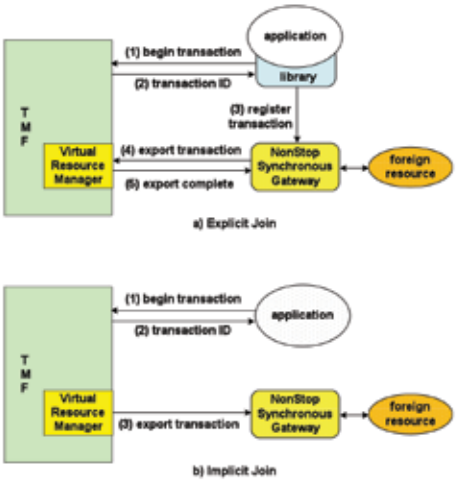

Figure 4: HP NonStop Synchronous Gateway Joining a Transaction

[4] See HP's NonStop Synchronous Gateway, Availability Digest; June 2006, for further detail describing the HP NonStop Synchronous Gateway.
[5] Dr. Bill Highleyman, Paul J. Holenstein, Dr. Bruce Holenstein, Chapter 4 – Synchronous Replication, pg. 83, Breaking the Availability Barrier: Survivable Systems for Enterprise Computing, AuthorHouse; 2004.

transaction may be joined either explicitly or, with a new enhancement, implicitly. For explicit joining (Figure 4a), the foreign resource must provide a library (either callable by the application or an intercept) that the application uses to join a transaction. When the application begins a transaction, it does so via a library call to the provided library or an intercept of the BEGINTRANSACTION call. The library issues a TMF begin-transaction request to TMF (1), which returns a transaction ID (2). The library uses this transaction ID to register the transaction with the SG by calling an API provided by the SG (3). The SG exports the transaction to the resource's VRM (4). At this point, the application has joined the transaction, and TMF is aware that the application's foreign resource is party to the transaction. It then notifies the SG that the export has completed (5).

Foreign resources may also specify implicit joins when they open the VRM (see Figure 4b). In this case, the VRM and SG automatically include the foreign resource in all transactions initiated by the application.

Explicit joins allow an application to decide which transactions it wishes to join. However, it is not able to join transactions that may be important to it that are started by other NonStop system processes, such as the disk process or SQL/MX. Implicit joining includes the foreign resource in all transactions. Once joined to a transaction, it is the responsibility of the foreign resource to decide whether to remain joined to the transaction, and if so to ultimately vote on the Prepare Phase.

Committing a Transaction

Transaction completion is shown in Figure 5. When the application issues a commit directive to TMF (1), TMF enters the prepare phase of the two-phase commit protocol and asks each RM if it is ready to commit. This includes the VRM for each foreign resource (2).

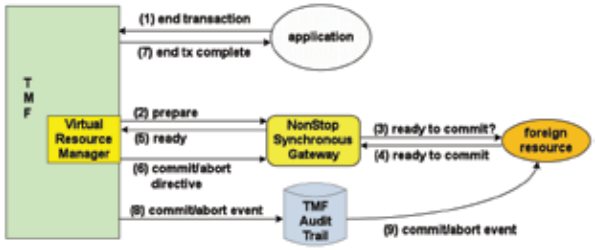

Figure 5: HP NonStop Synchronous Gateway Committing a Transaction

The SG asks its foreign resource if it is ready to commit (3), and if it receives a positive response (4), it responds to the VRM that its resource is ready to commit (5).

If TMF receives ready responses from all of the resource managers that were involved in the transaction, it issues a commit directive to each resource manager (6). If TMF receives a not-ready response from any resource manager, it issues an abort directive to all resource managers.

At this point, TMF informs the application that the transaction has completed and of its commit/abort status (7). It also writes a commit or abort token to its transaction log (8).

It is up to the foreign resource to provide a means to become informed of a commit or abort. As we shall see in our case, the foreign resource is a replication engine that is reading events from the TMF transaction log (9). As soon as it reads a commit or abort for a transaction that it has in progress, it completes the transaction.

Coordinated Commits

Coordinated Commits[6] is a future data replication technique that marries the advantages of asynchronous replication with the features of the SG to provide transactional synchronous replication benefits for the HP NonStop platform. Its integration with the SG is shown in Figure 6.

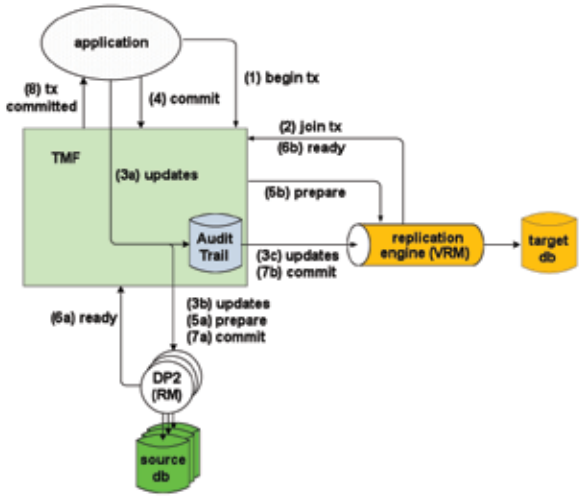

Figure 6: Coordinated Commit Data Replication

In effect, the coordinated-commit facility integrates an asynchronous replication engine with TMF as a foreign resource. When an application begins a transaction (1), the replication engine, via the Synchronous Gateway, can join a transaction either explicitly or implicitly (2). As the application makes changes within the scope of the transaction (3a), the changes are sent to the local disk processes (3b) and are written to the TMF Audit Trails (transaction log).

These changes are read from the TMF Audit Trails by the asynchronous replication engine (3c) and replicated to the target database, where the remote replication engine starts a remote transaction and uses it to apply the changes into the target database. This is the normal function of the asynchronous replication engine; there is no interaction with the application or SG during this phase. The transaction on the target is a separate transaction from the one on the source. The propagation of transaction events to the target system is transparent to the application and does not delay it.

[6] B. D. Holenstein, P. J. Holenstein, W. H. Highleyman, Asynchronous Coordinated Commit replication and dual write with replication transmission and locking of target database on updates only, U.S. Patent 7,177,866; February 13, 2007.
Dr. Bill Highleyman, Paul J. Holenstein, Dr. Bruce Holenstein, Chapter 4 – Synchronous Replication, Breaking the Availability Barrier: Survivable Systems for Enterprise Computing, AuthorHouse; 2004.

However, no permanent changes are made to the target database until the transaction is committed at the source. When the application issues a commit directive (4), the DP2's (5a) and the replication engine (5b) are asked if they are ready to commit (the prepare phase). The replication engine can respond positively if the target system has safe-stored all changes in a log file (to provide Zero Data Loss (ZDL) or has optionally tentatively applied them to the target database (e.g., end-to-end applying the events into the target database to avoid data collisions when running active/active). If all resources respond positively (6a, 6b), TMF commits the transaction. It informs the DP2s to commit (7a), and a commit token is written into the transaction log (7b). Upon replicating this token to the target system, the replication engine on the target system commits its target transaction. If any resource cannot respond positively, the source transaction is aborted. The target system aborts the target transaction when it receives the abort token via the audit trail.

Thus, replication is synchronous. The transaction's changes are either made to all resources (the local database and the remote target database) or to none of them. No data will be lost upon a source system failure (RPO = 0), and there cannot be any data collisions if a coordinated-commit replication engine is used in an active/active configuration when the replication engine is fully applying the events into the target database before it votes for the prepare phase.

In addition, the application does not have to wait for each update to propagate across the network before it issues the next update. It processes I/O's at its inherent speed. It is only at commit time that it must wait for the replication engine to vote on the transaction's outcome. Therefore, application latency is reduced to one round trip for the prepare phase or even to zero if the source side of the replication engine is notified in advance of target database update completions. The reduction in application latency is enhanced if the target system safe-stores updates in a log file rather than applies them to the target database as sequential writes to a log file are much faster than random updates made to a database.

Summary

Coordinated-Commit replication provides the benefits of both synchronous replication and asynchronous replication while eliminating (or at least minimizing) the issues associated with both of them. An RPO of zero is achieved. There are no data collisions, and nodes are separated by great distances with little if any impact on system performance.

Coordinated-Commit data replication is applicable to both active/passive and active/active systems.

Gravic, Inc.

The implementation of Coordinated Commits is a future technology currently being developed by Gravic, Inc. with support from the HP TMF team. It utilizes the HP NonStop Synchronous Gateway and the Shadowbase asynchronous data replication engine from Gravic, Inc. The Shadowbase replication engine is a high-speed, bi-directional, heterogeneous data replication engine that moves data updates between systems in fractions of a second. It supports active/passive through Sizzling-Hot-Takeover through active/active configurations to achieve high to continuous availability architectures.

In today's competitive environment, an enterprise must establish long-term strategies to optimize its operations while at the same time reacting with intelligence to events as they occur. Shadowbase Total Replication Solutions® provides products to leverage this technology with proven implementations.

Please contact the author for more information. Note that specifications are subject to change without notice and delivery dates/timeframes are not guaranteed.

Paul J. Holenstein is Executive Vice President of Gravic, Inc. He is responsible for the Shadowbase® suite of products. The Shadowbase replication engine is a high-speed, unidirectional and bidirectional, homogeneous and heterogeneous data-replication engine that moves data updates between enterprise systems in fractions of a second. It also provides capabilities to integrate disparate operational application information into real-time business intelligence systems. Shadowbase Total Replication Solutions® provides products to leverage this technology with proven implementations.

NonStop Nuggets

Thought Leadership – Empowering the Community

Jimmy Treybig, former CEO and founder of Tandem Computers and Kristi Elizondo, Chief Executive Officer of Connect Worldwide, took a few minutes out at the Connect NonStop Technical Bootcamp 2012 and discussed how the user group community we originally built was one of the first true forms of social media, how important the community was in guiding the direction of the products and how the current Connect organization isjust as relevant today as it was in the past when it was known as the International Tandem User Group (ITUG).

Scan me to learn how we did it!