Switching Replication Engines with Zero Downtime

Paul J. Holenstein, Executive Vice President, Gravic, Inc.
Dr. Bruce Holenstein, President and CEO, Gravic, Inc.
Dr. Bill Highleyman, Managing Editor, Availability Digest

Part 3

In Part 1 of this article series, we described how a data replication engine could be switched with another replication engine that is a different version of the current engine or is in fact one from a different vendor without taking an outage. We call this zero downtime migration (ZDM). In Part 2, we showed how this migration could be improved by adding one or two additional nodes. We also discussed the unique considerations when using bidirectional replication. In this Part 3, we look at switching replication engines without missing or re-replicating any data.

Solving the *Jagged Edge* Problem When Switching Replication Engines

The so-called *jagged edge* problem occurs when switching from one replication engine to another. It may also occur when upgrading the replication engine version, if this aspect of the replication engine has been changed across versions. It is especially important when the original and new data replication engines are from different vendors, as they will each have their own algorithms for managing the data to be replicated. To properly resolve this issue, a thorough understanding of how transactionally-based engines replicate data is needed, along with reviewing various algorithms that resolve the issue, based on the type of engine that is being replaced.

Jagged Edge and Audit Trail Background

Figure 18 repeats an earlier figure (from Part 1 of this article series), but now we delve into the details of switching over, for example, from v6100 to v6400 of the data replication engine, where the new replication engine catches up to the proper point where the switch-over occurred, without missing or repeat-replicating any data, or causing the target database to roll back to its original version.

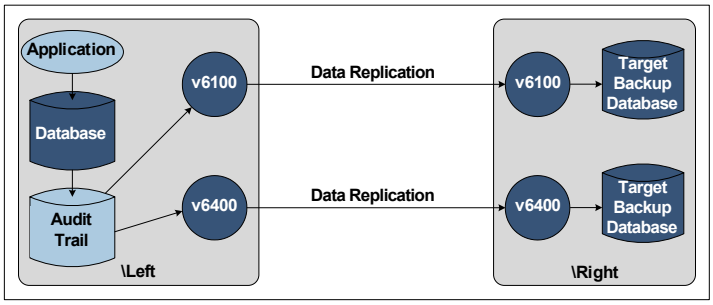

Figure 18 – v6100 to v6400 Switch-Over

The approach used to switch over from one replication engine to the other depends on the method each individual replication engine uses when it replicates, with the most important consideration being the method the *original* replication engine uses to track its restart point. The *new* data replication engine must adapt accordingly. Although other methods are also in use, we now discuss two common methods, and how properly to switch over when each is in use: *The Brute Force Replication Method* and *The Transactional Replication Method*.

For this discussion, assume that the "Target Backup Database" in Figure 18 represents the actual complete target database – the changes for a specific file/table (or individual partition in an HPE NonStop system) are replicated by either v6100 or v6400, but not both. In Figure 19, the audit trail represents the log of all changes made to the source database; these database changes are applied by all of the applications on the source environment. Although the audit trail may be comprised of multiple separate log files (e.g., a MAT sequence of log files along with one or more AUX sequences of log files on an HPE NonStop), it can be processed as a singular and sequential listing/queue of the change events across all files and tables in "roughly" *ascending time order.*[1] Note that the landed/received order prevails if it differs slightly from the assigned chronological order of each event in the aggregated trail.

Processing the audit trail in sequential/landed order yields a stream of transactional activity that has occurred against the source database. For any individual file/table (or partition on an HPE NonStop), the landed order reflects with absolute certainty the correct and consistent order that the events occurred in for each individual file or table (or partition thereof). On other platforms, the landed order may need to be re-sorted; however, the re-sorted order reflects the correct and consistent event order. In either case, replaying the events in the correct and consistent order results in a correct and consistent target database.

Often (and always on HPE NonStop), the landed order represents the simultaneous interthreading of many transactions occurring at the same time. On some platforms/databases, a re-sorted order returns the change events in transactional order, based on the commit or abort completion time, such that all events for a transaction are returned in a sequential group for each transaction. Regardless, replaying the events in the order received (whether landed or sorted), results in a correct and consistent target database.

[1] It is in "roughly" ascending time order due to the nature of the TMF flushing not following a pure chronological order across all of the disk packs.

When processing the audit trail, each replication engine typically maintains a persistent "restart point" that reflects the position from which it needs to restart should it stop/fail and subsequently restart. This position reflects the point (or can be used to derive the point) in the audit trail where the replication engine needs to start reading in order to guarantee that it does not miss/skip any data. At the restart in the best data replication engine designs/implementations, the data replication engine does not replay any data that it previously applied against the target database, because this replay can cause data consistency issues at the target during the restart/catchup sequence.

Figure 19 reflects a simplified view of the layout of the internal events inside the audit trail (or the sequence of events, because the audit trail is aggregated). In this figure:

- The audit trail is depicted as a sequenced queue of change data events, where each event has an associated transaction ID assigned to it. The front of the queue (removal/extraction point) is to the right; the end of the queue (insertion/arrival point) is to the left.
- Time moves from right to left. The oldest/earliest events are to the right in the queue; the newest/latest events are to the left in the queue.

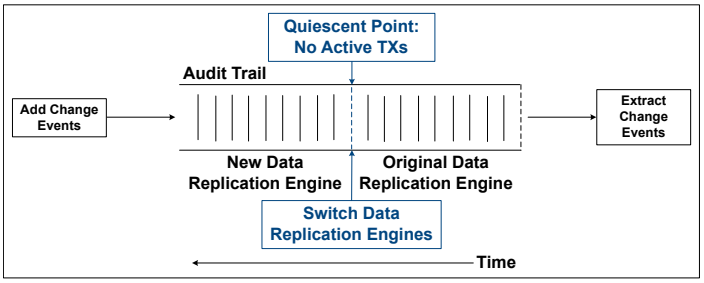

Figure 19 – Simplified View of the Audit Trail When the Application is Stopped at the Quiescent Point

If the application can be quiesced at a point where all transactions have ended (the "Quiescent Point: No Active TXs" in the figure), the original data replication engine takes responsibility to replicate the events to the right of the quiescent point, and the new data replication engine takes responsibility to replicate the events to the left of the quiescent point, with no transactions spanning it. It is fairly simple; however, since this paper is about avoiding an application outage while the migration or upgrade takes place, this "clean" switch-over point does not exist for applications/transactions that remain active during the switch-over process.

For this complex case, a more advanced algorithm is required, where the application is active when the switch-over takes place, as shown in Figure 20. In this figure:

- The audit trail again is depicted as a queue that grows with change data events from right to left (time increases from right to left).
- The switch-over point is selected to be at Timestamp(1). This is the point the original data replication engine is stopped/shutdown.
- The abort timer(2) represents a go-back interval from the switch-over point, that is discussed below.
- TX(3) represents a transaction that started before the abort timer(2) go-back interval, that commits before the switch-over point.
- TX(4) represents a transaction that started and aborts before the switch-over point.
- TX(5) represents a transaction that started before the switch-over, but does not end until much later after the switch-over.
- TX(6) represents a transaction that started before the switch-over, but does not commit (at the Commit(7) point) until after the switch-over point.
- TX(8) represents a transaction that starts after the switch-over point and ends at some later time.

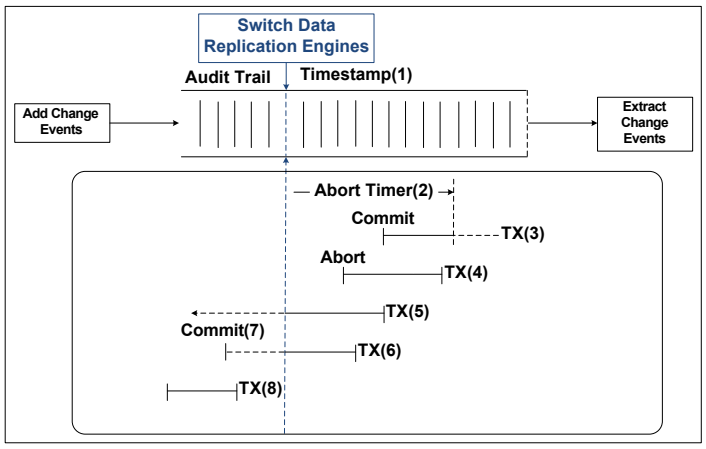

Figure 20 – View of the Audit Trail when the Application Remains Active Across the Switch-Over

The method to provide a clean, consistent, and complete switch-over from the original data replication engine to the new data replication engine depends on the method the original data replication engine uses to replicate the audit trail data.

The Brute Force Replication Method

If the original data replication engine uses the *Brute Force Method* to replicate:

- The original data replication engine takes responsibility to replicate and apply/commit all transactional events that have occurred *before* the switch-over point.
- In some cases, this replication may only include transactions that have committed before the switch-over point, excluding those that have aborted before the switch-over point.
- For transactions that are in progress at the switch-over point (e.g., TX(5) and TX(6) in the figure), the original data replication engine replicates and applies only those events that occurred before the switch-over point, performing a *commit* for each before shutting down. Since these transactions may ultimately abort, the new data replication engine is responsible for special processing of the transactions to apply all follow-on data and any backout events that might occur after the switch-over point.
- The new data replication engine takes responsibility to replicate all new transactions forward (i.e., the events for those transactions that start after the switch-over point).

The Brute Force Method does not maintain true data consistency as it commits and materializes events for transactions that may ultimately abort, as well as pre-commit transaction events for those that are in process at the switch-over point that ultimately commit. However, any such inconsistencies are short-lived, only until those transactions eventually complete (commit or abort) after the switch-over to the new data replication engine.

The Transactional Replication Method

If the original data replication engine uses the *Transactional Replication Method* to replicate:

- The original data replication engine takes responsibility to replicate and apply/commit all transactional events that have *completed* before the switch-over point.

- In some cases, this replication may only include transactions that have committed before the switch-over point, excluding those that have aborted before the switch-over point. Regardless, only those transactions that have committed are materialized in the target database at the switch-over point.
- For transactions that are in progress at the switch-over point (e.g., TX(5) and TX(6) in the above figure), the original data replication engine either does not replay them, or aborts them when it shuts down at the switch-over point. In some cases, the original data replication engine may replay the data for transactions that are in progress as of the switch-over point. However, before it shuts down, it retrieves the backout events from the target side audit trail and applies them for any transactions that were not completed before the switch-over point. This process effectively matches performing an abort for the transactions that are in progress at the time of the switch-over.
- The new data replication engine must take responsibility to replay:
  - All transactions that were in progress at the switch-over point.
  - All transactions that start after the switch-over point.

Although it is relatively straightforward to replicate all new transactions created after the switch-over point, identifying those that were in progress at the switch-over point can be complex. Although other methods exist, one simple algorithm to identify these transactions is to use the DBMS's "auto abort" time as an interval (Abort Timer(2) in Figure 20).

The abort timer is a DBMS-maintained timer that ensures no transaction lasts (exists/runs) for more than a set period of time (or audit trail duration). When an abort timer exists, it is known that no transaction can last more than that amount of wall clock time (or audit data) relative to the change's time in the audit trail. Hence, when the new data replication engine goes back from the switch-over point to at least the abort timer amount of time in the audit trail, it knows that no transactions started before this time are still active at the switch-over point. The new data replication engine computes the abort timer interval from using the switch-over point's timestamp (Timestamp(1) in the figure), deducts the abort timer setting from this timestamp (e.g., deducts 2 hours from the timestamp if the abort timer is 2 hours), and goes back that far in audit. It then reads forward to the switch-over point, tracking all transactions encountered. The new data replication engine disregards all transactions that end before the switch-over point since the original data replication engine took responsibility to replicate them. For transactions still in progress at the switch-over point, the new data replication engine replicates them as well as all additional transaction events encountered from the switch-over point forward.

The Transactional Replication Method maintains full data consistency at the target as each transaction is replayed only once at the target and in proper order. Of course, additional algorithms exist for managing the jagged edge at the switch-over point, and these algorithms are more complex for bi-directional replication environments, but the main tenets remain the same. The original data replication engine takes responsibility for replicating all completed transactions before the switch-over point, with the new data replication engine taking responsibility for replicating all in progress transactions at the switch-over point as well as any new transactions that started after this point.

Summary

Sometimes it is necessary to change or update a data replication engine. Properly undertaken, a data replication engine migration imposes no downtime on applications or users, and the databases all remain consistent, complete, and up-to-date during the process. This is called a zero downtime migration (ZDM).

Since there is no big-bang cutover, and the original and new data replication engines do not need to interoperate, the ZDM technique results in greatly reduced risks for error as well as staff stress levels during the migration process. The migration can take place over time at whatever pace the staff feels appropriate, for instance, at normal working times when the staff is at its best, rather than late at night or on weekends, thereby reducing migration costs. It can even occur over an extended time period, with the existing backup database continuously available and fully synchronized with the production database, ensuring full application and database availability during the migration process as discussed in the three and four node examples.

This technique is similar to and leverages the ***HPE Shadowbase Zero Downtime Migration*** technique that companies have used for decades to upgrade their applications, database schema formats, file and table locations/indices, operating systems, or to perform a hardware refresh. Application outages that either are planned to support an upgrade/migration or are caused by poorly executed upgrades/migrations are outdated and should no longer occur. Use the HPE Shadowbase ZDM technique to attain continuous application availability with no risk, even across the most disruptive migrations and upgrades.

There are numerous approaches and methods to thwart common problems faced during a migration: *Version Independence* avoids interoperating different software versions; utilizing three nodes for partial hardware migrations or utilizing four nodes for a full hardware refresh; and bi-directional environments that keep all nodes synchronized as the migration takes place. Also, the so-called jagged edge problem that occurs when performing any replication engine migration must be resolved.

Switching replication engines with zero downtime is obviously never easy. With proper preparation, planning, time, and project assessment, all migrations should go flawlessly. HPE Shadowbase software enables companies to leverage these migration methods with proven solutions and implementations. The HPE Shadowbase team is interested in discussing your specific project plans and help you realize them.[2]

2 **Contact Gravic** or your HPE account team to discuss your plans.

Paul J. Holenstein is Executive Vice President of Gravic, Inc. He is responsible for the HPE Shadowbase suite of products. The HPE Shadowbase replication engine is a high-speed, uni-directional and bi-directional, homogeneous and heterogeneous data replication engine that provides advanced business continuity solutions as well as 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. HPE Shadowbase software is built by Gravic, and globally sold and supported by HPE. Please contact your local HPE account team for more information, or visit **https://www.ShadowbaseSoftware.com**. To contact the authors, please email: **SBProductManagement@gravic.com**.

Dr. Bruce D. Holenstein leads all aspects of Gravic, Inc. as President and CEO. He started company operations with his brother, Paul, in 1980. His technical fields of expertise include algorithms, mathematical modeling, availability architectures, data replication, pattern recognition systems, process control and turnkey software development.

Dr. Bill Highleyman brings years of experience to the design and implementation of mission-critical computer systems. As Chairman of Sombers Associates, he has been responsible for implementing dozens of real-time, mission-critical systems - Amtrak, Dow Jones, Federal Express, and others. He also serves as the Managing Editor of The Availability Digest (availabilitydigest.com). Dr. Highleyman is the holder of numerous U.S. patents and has published extensively on a variety of technical topics. He also consults and teaches a variety of onsite and online seminars.
Find his books on Amazon. Contact him at **billh@sombers.com**.