****

Using Replication for High Availability and Disaster Recovery: A SQL Server 2008 Technical Case Study and Best Practices

SQL Server Technical Article and Best Practices

**Writers:** Manaphan Huntrakoon, Nicholas Dritsas

**Technical** **Reviewers**: Burzin Patel, Sanjay Mishra

**Published:** August 2009

**Applies to:** SQL Server 2008

**Summary:** An essential part of IT mission is maintaining high availability and disaster recovery capability. This technical case study shows how a company can use SQL Server 2008 database replication to eliminate single points of failure in data centers, and to enable fast recovery from a possible disaster at its primary data center. These strategies and solutions will be of interest to database administrators, senior IT managers, project leads, and architects.

**Table of Contents**

[1 Introduction 5](#_Toc238964564)

[2 Envision 6](#_Toc238964565)

[2.1 Current Infrastructure 6](#_Toc238964566)

[2.2 High Level View of Their Original Database Topology 6](#_Toc238964567)

[2.3 Applications 7](#_Toc238964568)

[2.4 Database Objects 8](#_Toc238964569)

[2.5 Technical Requirements 8](#_Toc238964570)

[3 Plan 9](#_Toc238964571)

[3.1 Key Decision Points 9](#_Toc238964572)

[3.1.1 Failover Clustering 9](#_Toc238964573)

[3.1.2 Database Mirroring 9](#_Toc238964574)

[3.1.3 Log Shipping 10](#_Toc238964575)

[3.1.4 P2P Replication 10](#_Toc238964576)

[3.2 Database Version Selection 11](#_Toc238964577)

[4 Design 12](#_Toc238964578)

[4.1 Database Design 12](#_Toc238964579)

[4.1.1 SAN Storage 12](#_Toc238964580)

[4.1.2 SQL Server Options 14](#_Toc238964581)

[4.2 Database Object Level Design 14](#_Toc238964582)

[4.2.1 Tables 14](#_Toc238964583)

[4.2.2 Identities 14](#_Toc238964584)

[4.2.3 Triggers 15](#_Toc238964585)

[4.2.4 Constraints 16](#_Toc238964586)

[4.3 Replication Design 16](#_Toc238964587)

[5 Stabilize 18](#_Toc238964588)

[5.1 Unit Testing 18](#_Toc238964589)

[5.1.1 P2P Replication with Conflict Detection and Read-Only Subscribers 18](#_Toc238964590)

[5.1.2 Max Replication Text Size Option 20](#_Toc238964591)

[5.1.3 P2P Publications with Spaces in Name 21](#_Toc238964592)

[5.1.4 Distribution Agent with OLE DB Streaming 22](#_Toc238964593)

[5.1.5 Distribution Cleanup Failure 22](#_Toc238964594)

[5.1.6 Dummy Updates Issue with Conflict Detection 23](#_Toc238964595)

[5.1.7 P2P Publication removal issue 24](#_Toc238964596)

[5.1.8 Log Reader Agent Fails After Adding an Article to a Publication with binary large object (BLOB) Data 24](#_Toc238964597)

[5.2 System Testing 26](#_Toc238964598)

[5.3 Performance Testing 27](#_Toc238964599)

[5.3.1 Testing Scenarios 27](#_Toc238964600)

[5.3.2 Effect of Push Subscription vs. Pull Subscription 28](#_Toc238964601)

[5.3.3 Effect of Stored Procedure Delivery vs. SQL Delivery 31](#_Toc238964602)

[5.3.4 Effect of Improved Windows Server 2008 TCP/IP Stack 33](#_Toc238964603)

[5.3.5 Effect of WAN Accelerator 35](#_Toc238964604)

[5.3.6 Effect of Subscription Streams 36](#_Toc238964605)

[5.3.7 Effect of Removing a Node 38](#_Toc238964606)

[6 Deployment 42](#_Toc238964607)

[6.1 Go-Live Planning 42](#_Toc238964608)

[6.2 Go-Live Testing 43](#_Toc238964609)

[7 Conclusions 44](#_Toc238964610)

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2009 Microsoft Corporation. All rights reserved.

Microsoft, Access, Excel, MSDN, SQL Server, Windows, and Windows Server are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Template Guidance

Description: The Vision/Scope document represents the ideas and decisions developed during the Envisioning phase. The goal of the phase, represented by the content of the documentation, is to achieve team and customer agreement on the desired solution and overall project direction.

The Vision/Scope document is organized into four main sections:

* Business Opportunity: A description of the customer’s situation and needs
* Scope: The boundary of the solution defined though the range of features and functions; what is out of scope, a release strategy and the criteria by which the solution will be accepted by users and operations
* Solution Design Strategies: The architectural and technical designs used to create the customer’s solution
* Project Structure: How the team and project will be organized.

Note: In this template, you must enter specific project information in certain placeholder areas. To view the fields containing these placeholders, follow these steps:

1. Click **Tools > Options**.
2. On the **View** tab, in the **Field Shading** list, pick **Always**.
3. Click **OK**.

Throughout the template, look for shaded text. Where shaded text appears, click the field and follow instructions.

Depending on the complexity of the project, not all of the sections might be filled out, or some sections might be cut back significantly.

Justification: Vision/Scope documentation is usually written at the strategic level of detail and is used during the Planning phase as the context for developing more detailed technical specifications and project management plans. It provides clear direction for the project team; outlines explicit, up-front discussion of project goals, priorities and constraints; and sets customer expectations.

Team Role Primary: Product Management is the key driver of the Envisioning phase and is responsible for facilitating the team to the Vision/Scope approved milestone. Product Management defines the customer needs and business opportunity or problem addressed by the solution.

Team Role Secondary: Program Management is responsible for articulating the Solution Concept, Goals, Objectives, Assumptions, Constraints, Scope and Solution Design Strategies sections of this document.

1. Introduction

An international company has deployed a number of SQL Server instances spread out over different locations in Asia, the United States and Europe. The main problem they wanted to resolve was the high availability of data critical to their main application. In the event this data ever became unavailable, the financial repercussions would have been substantial.

While the company had many ways and methods to alleviate this problem, they were unsure of which SQL Server high availability/disaster recovery technology would address their problem plus provide a strategic platform for the future. They engaged with Microsoft to help in the design of an architecture best fit for their needs, budget, expectations and skill set.

In the remaining of this document, we will be going through the steps and processes the project team went through to evaluate and decide on the best choices and options.

1. Envision

In order to determine the best solution for the customer, Microsoft engaged initially with its Consulting Services in a five-day envisioning workshop to obtain a clear understanding of the technical and functional requirements. Current infrastructure, database topology, applications, database objects, and customer requirements had to be discussed in depth before a Vision/Scope document could be constructed.

* 1. Current Infrastructure

The company has two data centers in place – one in the United States and one in Asia, with plans to deploy another one in Europe in the near future.

The company was running multiple instances of SQL Server 2005 Standard Edition and Enterprise Edition on multiple servers to serve various mission-critical applications and functions. The Asian data center served all read requests from the Asia-Pacific area, and the U.S.-based data center served all read requests from United States and Europe.

Although read requests had an adequate amount of redundancy, key tables associated to write transactions were only located in the Asian data center. Because the key tables weren’t located in all data centers, the company used a series of staging tables and DTS jobs to transfer data from the U.S.-based data center to the Asian.

The single point of failure with this infrastructure was the Asian data center. If the database that held the key tables were to become unavailable or if the data center lost due to a disaster, the company would not be able to carry out their write requests, which would result in a substantial financial loss.

* 1. High Level View of Their Original Database Topology

The following figure is an illustration of the high-level database topology that was in production at the time of the engagement. Note there is no core database in the U.S.-based data center. The SQL Server Integration Services jobs that moved write transactions from the U.S.-based data center to the Asia-based data center frequently failed due to network related issues.

Refreshes of the Read database were done via BCP Out/In. In the case of the U.S.-based read servers, the BCP data had to be exported to a file server in the data center and then imported via BCP. This operation occurred once a day.

In the case that an immediate update had to be performed on the Read database that supports their web sites, the database administrators (DBAs) would have to log into each database individually and update the data via Transact-SQL.

It is also important to note that the company had one high-powered server in the Asia-based data center and two high-powered servers in the U.S.-based data center that were not being utilized and were idle.



Figure 1

* 1. Applications

All of the applications written by the company were .NET based. The applications could be grouped into two different categories: internal and external.

Internal applications updated directly in the Core database, but external applications primarily read data from the Read databases. Write transactions from external applications only occurred in the Staging database and were transferred every five minutes via an SSIS job that inserted data from the staging table into the destination table.

In addition to end-user applications, IT developers and DBAs would access the tables directly via SQL Server 2005 SQL Server Management Studio, the Microsoft Office Access® database software, and the Office Excel® spreadsheet software.

* 1. Database Objects

The company had approximately 500 tables and over 3,000 stored procedures in their main Core database. Most tables used Identity columns and many tables employed a combination of cascading update/delete constraints and after update/insert table triggers.

* 1. Technical Requirements

The following is a list of technical requirements that were utilized in designing the proposed high-availability/disaster recovery proposal:

* Make two servers available as part of the high availability/disaster recovery strategy. Two servers were purchased for disaster recovery purposes prior to the engagement but were never provisioned. It was the customer's wish to use these idle servers as part of the high availability/disaster recovery strategy.
* Minimize downtime and business impact to users.
* Leave the current Read database and architecture in all data centers intact, because these are already redundant.
* Provide an architecture that will enable all servers to be utilized in a multi-master fashion.
* Provide an architecture that will ensure redundancy of key tables.
* Provide an architecture that will enable a single site to bring one server down for maintenance and still be able to serve users.
* Provide an architecture that will enable Read databases that support the Web sites to be refreshed more frequently and eliminate the need to update their content manually.
* Reduce the complexity and number of data flows interacting between the different databases inter and intra data center.
* Eliminate the complexity and number of data flows interacting between the different databases inter and intra data center.

Utilize existing serversDescribe the customer’s current situation that creates the need for the project. You may include a statement of the customer’s opportunity and the impact of capitalizing on that opportunity (product innovation, revenue enhancement, cost avoidance, operational streamlining and leveraging knowledge). You may also include a statement of the customer’s problem and the impact of solving the problem (revenue protection, cost reduction, regulatory compliance, alignment of strategy and technology). You should include a statement that connects the customer’s opportunity/problem to the relevant business strategy and drivers. The Opportunity Statement is written concisely using a business executive’s voice.

Justification: The Opportunity Statement demonstrates that Microsoft understands the customer’s situation from the business point of view and provides the project team and other readers with the strategic context for the remaining sections.

1. Plan

The second phase in the governance model is the Plan Phase. The goals of this phase is to evolve solution concepts into tangible designs and plans so they can be built in the develop track.

* 1. Key Decision Points

SQL Server Always-On technologies were designed to ensure maximum uptime of mission-critical applications. Because each customer has different demands and may or may not want to employ a specific technology, there is no uniform Always-On technology that will fit every customer need. It is crucial that SQL Server be able to provide the customer a choice when it comes to high availability and disaster recovery. The following Always-On technologies were considered and run through the entire set of customer requirements:

* + 1. Failover Clustering

Failover clustering is a combination of one or more physical disks in a failover cluster (formerly known as server clusters or MSCS) cluster group, known as a resource group, that are participating nodes of the cluster. The resource group is configured as a failover clustered instance that hosts an instance of SQL Server. A SQL Server failover clustered instance appears on the network as if it were a single computer, but it has functionality that provides failover from one node to another if one node becomes unavailable.

While failover clustering could have been utilized to ensure high-availability within the data center, it was the customer’s requirement that failover clustering not be used in the final solution. This was due to the additional hardware costs associated to clustering (certified hardware plus SAN) and their requirement for multi master nodes.

Because the customer did not want to utilize failover clusters, it was eliminated as a possible solution.

* + 1. Database Mirroring

Unlike failover clusters, which are a hardware solution, database mirroring is primarily a software solution for increasing database availability. Database mirroring maintains two copies of a single database that must reside on different server instances of SQL Server Database Engine. One server instance serves the database to clients (the principal server).

The other instance acts as a hot or warm standby server (the mirror server), depending on the configuration and state of the mirroring session. When a database mirroring session is synchronized, database mirroring provides a hot standby server that supports rapid failover without a loss of data from committed transactions. When the session is not synchronized, the mirror server is typically available as a warm standby server (with possible data loss).

Although database mirroring could have been utilized to ensure high-availability within the data center, it was the customer requirement that all four servers be able to be utilized as master databases.

Because the mirror server is unable to serve write requests while the principal is still available, database mirroring was eliminated as a possible solution.

* + 1. Log Shipping

Log shipping enables you to automatically send transaction log backups from a primary database on a primary server instance to one or more secondary databases on separate secondary server instances. The transaction log backups are applied to each of the secondary databases individually.

Log shipping does not provide the ability for the secondary server to act as a primary server because it is in either a NORECOVERY or STANDBY state. This prevents the secondary server from being used for any write requests while the principal server is available.

Because the secondary server is unable to serve any write requests while the principal is still available, log shipping was eliminated as a possible solution.

* + 1. P2P Replication

P2P replication provides a scale-out and high-availability solution by maintaining copies of data across multiple server instances, also referred to as nodes. Built on the foundation of transactional replication, P2P replication propagates transactionally consistent changes in near real-time.

This enables applications that require scale-out capabilities to distribute the transactions from clients across multiple nodes. Because data is maintained across the nodes in near real-time, P2P replication provides data redundancy, which increases the availability of data.

P2P replication meets the following customer requirements:

* Utilize the existing and unutilized servers in the U.S.-based and Asia-based data centers.
* Provide an architecture that is not based on Microsoft failover clustering.
* Minimize downtime and business impact to users.
* Provide an architecture that will enable all servers to be utilized in a multi-master fashion.
* Provide an architecture that will ensure redundancy of key tables.
* Provide an architecture that will enable a single site to bring one server down for maintenance and still be able to serve users.

After these requirements were met, it was then possible to meet the last two customer requirements by using basic one way transactional replication from the Core database to the Read databases:

* Provide an architecture that will enable Read databases to be refreshed more frequently and eliminate the need to log into multiple Read databases to update content.
* Leave the current Read database and architecture intact, because these are already redundant.
	1. Database Version Selection

While P2P replication was available on SQL Server 2005, significant improvements were made to P2P replication in SQL Server 2008, such as:

* Conflict detection
* P2P Topology Configuration Wizard
* Replication compression
* Addition of nodes without quiescing the topology

These improvements address common customer pain points in the following ways:

* Conflict detection helps the company determine whether there are conflicts in their data, build reports off these conflicts, and resolve application issues.
* The P2P Topology Configuration Wizard easily sees and reconfigures topology without coding.
* Replication compression and the ability to pass in parameters instead of string literals helps reduce the size of the packages transmitted across the WAN.
* The ability to add nodes without quiescing the topology enables the company to easily add the Europe based data center to the P2P topology without having to stop activity on all their nodes.

 In addition to the P2P improvements, other features in SQL Server 2008 also address customer pain points. These include the following but will not be discussed in detail:

* Backup compression – helps customers reduce the size and time of database backups. With backup compression, the customer is able to dramatically reduce the size and time taken to perform database backups. In the case of this customer, database backup size was reduced from over 45 gigabytes (GB) to 15 GB.
* Policy Based Management (PBM) – helps customers reduce administrative overhead and be able to push policies to other similar servers in the enterprise.
* Data compression – helps customers reduce the amount of space consumed by tables using either page or row compression.
* Management data warehouse – helps customers understand trends in their databases. Users can obtain intelligent reports in which they can drill down to details if necessary.

All of these new features in SQL Server 2008 easily convinced the customer to upgrade their core database system from SQL Server 2005 to SQL Server 2008.

Note: This customer also decided to use the Windows Server® 2008 operating system, due to its improved TCP/IP stack compression (see below test results).

1. Design

The third phase in the governance model is the Design Phase. The goal of this phase is to build all aspects of the solution in accordance with the deliverables from the plan track. All aspects of the solution are tested thoroughly to assess the state of the quality of the solution.

* 1. Database Design

The key considerations in the database design were proper SAN storage configuration and related SQL Server options, such as backup compression and maximum text replication size.

* + 1. SAN Storage

Proper configuration of I/O subsystems is critical to the optimal performance and operation of SQL Server systems. Here are some recommendations for disk, volumes, data files, and filegroups:

**Disk**

* Disks formatted with 1 megabyte (MB) starting offset
* Disks formatted with 64 kilobyte (KB) default file allocation unit size
* Verify that the following algorithms result in an integer value:
	+ PARTITION\_OFFSET / STRIPE\_UNIT\_SIZE
	+ STRIPE\_UNIT\_SIZE / FILE\_ALLOCATION\_UNIT\_SIZE
* Place log files on RAID 1+0 (or RAID 1) disks for better protection from hardware failure and better write performance

**Volumes**

It was determined that the customer’s database was not optimized per Microsoft Best Practices. Migrating to a new database server allowed the team to optimize the volume structure and prepare it for replication. (\*)

|  |  |
| --- | --- |
| Drive | Purpose |
| C | Operating system |
| D | SQL Server binaries and system databases |
| E | tempdb data and log |
| F | Distribution database |
| G | SQL Server data |
| H | SQL Server log |
| I | SQL Server backup |

\* All LUNS resided on separate sets of physical spindles.

Table 1

**Data File**

* Isolate the log from the data at the physical disk level.
* Isolate tempdb data files from user database data files.
* Isolate the backup location from the user database files.
* Isolate the system database location from users, **tempdb**, log, and backup.
* Configure **tempdb** so that it is:
	+ Presized
	+ Placed on RAID 1+0 array (if possible) and separated from user log and data files
	+ Configured with one data file per CPU
* Presize data and log files
* Avoid reliance on AUTOGROW
* Ensure that data files are of equal size

**Filegroup Optimization**

Having all data into one big filegroup – PRIMARY is not recommended. Logical separation of data would help for better performance. The PRIMARY filegroup should contain only the system catalog and system objects. No user objects should reside in the PRIMARY filegroup. You should also group nonclustered indexes into separate logical filegroups. If data volume is higher and big portion of the data is history data, you should separate history data into a separate filegroup. This can improve read and write operations performance.

* + 1. SQL Server Options

Two SQL Server options were explicitly set for purposes of this customer’s solution. The first option was SQL Server's 2008 backup compression. The second option was the maximum text replication size (which is set by using the **Max Text Replication Size** option in SQL Server Management Studio or the **max text repl size** option of **sp\_configure**), which has a default value of 65536.

If the **Max Text Replication Size** value is not big enough, and the data row being replicated is larger than the value, the transaction will fail and an error will be returned to the calling application. The maximum value allowed was used in our case study for P2P replication because although it was known that the text size would not exceed the maximum, it was inconclusive what the maximum value was going to be.



* 1. Database Object Level Design
		1. Tables

Before the tables can be added to a publication, they must have a PRIMARY KEY identified. This is a requirement for replication.

* + 1. Identities

A majority of the tables in the customer database had columns that used IDENTITY values. When these tables participate in a replication topology, it is very important that the IDENTITY values be preserved when data is published to its participating subscriber. If IDENTITY values are not preserved, subsequent replicated updates, deletes, and inserts may fail, which will cause replication to fail. In order to prevent this failure and to ensure IDENTITY values were preserved when replicated from publisher to subscriber, all identities needed to be modified with the NOT FOR REPLICATION setting set to TRUE.

After the NOT FOR REPLICATION setting was set, it was important to determine the reseed ranges for each of the new nodes. These ranges would be used to reseed the identity values in each of the four databases participating in the P2P replication topology.

**For a Four-Node Range:**

Node 1 Range: 1 - 536,870,911

Node 2 Range: 536,870,912 to 1,073,741,823

Node 3 Range: 1,073,741,824 to 1,610,612,734

Node 4 Range: 1,610,612,735 to 2,147,483,647

**For a Six-Node Range:**

Node 1 Range: 1 – 357,913,941

Node 2 Range: 357,913,942 – 715,827,883

Node 3 Range: 715,827,884 – 1,073,741,824

Node 4 Range: 1,073,741,825 – 1,431,655,765

Node 5 Range: 1,431,655,766 – 1,789,569,706

Node 6 Range: 1,789,569,707 - 2,147,483,647

If the identity columns change to **bigint**, storage requirements will increase, but the range will be greatly increased to 2^63 (9,223,372,036,854,775,808) to 2^63 (9,223,372,036,854,775,808). Applications may need to be modified to accept change in data type.

* + 1. Triggers

A few of the tables in the customer database contained triggers that fired upon insert, update, and delete events. Like IDENTITIES, triggers can be configured to either replicate or not replicate the trigger event. If it is *not* intended for the subsequent trigger actions to be replicated at the subscriber, the NOT FOR REPLICATION setting needs to be set to TRUE. If the trigger writes to a published table, it is important that the NOT FOR REPLICATION value is set to TRUE.

Data inconsistencies can occur if NOT FOR REPLICATION is set to FALSE. Take for example the following scenario:

**Nodes:**

* Node 1
* Node 2

**Replicated tables:**

* Table A
* Table B

**Triggers:**

* After update on Table A that writes to Table B

If the NOT FOR REPLICATION setting is set to FALSE, and an update transaction occurs on Table A on Node 1, the following will occur:

1. Table A UPDATE occurs on Node 1.
2. Table A UPDATE gets written to the transaction log and placed in the distributor database to be distributed on Node 1.
3. After update trigger fires and updates Table B on Node 1.
4. Table B UPDATEs get written to transaction log and placed in the distributor database to be distributed on Node 1
5. Table A UPDATE from Node 1 gets distributed and applied to NODE B.
6. Update trigger fires and updates Table B on Node 2.
7. Table B UPDATEs from Node 1 gets distributed and applied to NODE B.
8. Conflict occurs on Node B (see Figure 2).
	* 1. Constraints

Some of the tables in the company database contained constraints that performed CASCADE operations on referenced tables. Like identities and triggers, constraints can be configured to either replicate or not replicate the CASCADE event. As with triggers, in order to prevent data conflicts, the NOT FOR REPLICATION setting should be set to TRUE.

* 1. Replication Design

Proper replication design is crucial for performance reasons. Most P2P replication projects consist of a single publication, with all articles to be published in that single publication. While this is not incorrect, it may not be the best design. It is extremely important to also understand how the applications interact with the system, the type of data that is being replicated, and the customer expectations in regards to when data will arrive.

The initial replication design consisted of a single P2P publication with separate one-way transactional replication publications based on need. The publications are described here:

* **P2P publication** – publication that consists of all articles (tables and stored procedures) in the core database. These tables will replicate with the P2P replication and conflict detection enabled.
* **Read-only publication** – publication that consists of all articles (tables and stored procedures) in the core database to be replicated to standby, read-only servers in a different geographical location local to the IT development team. The first subscriber will be the data warehouse and the second subscriber will be used for emergency standby purposes.
* **Web publication** – a publication that consists of a subset of the main articles in the core database. The articles in this publication are the basis of the content that is pulled from the customer-facing Web application.

The initial suggested topology is illustrated in Figure 2. 

Figure 2

Designing replication in this way allowed the team to satisfy another customer requirement:

* Help eliminate the complexity and number of data flows interacting between the different databases inter and intra data center.

A combination of over 30 complex DTS, BCP, SQL Server Integration Services, SQL Server Agent, and Transact-SQL jobs were eliminated with the implementation of this replication topology.

1. Stabilize

The fourth phase in the governance model is the Stabilize Phase. The goal of this phase is to improve the solution quality to meet the release criteria for deployment to production. The solution is validated by stakeholders to ensure that it meets expectations and it is usable from a user’s perspective. The end goal is to maximize success and minimize the risks associated with solution deployment and operations in its target environment.

* 1. Unit Testing

After the database level, object level, and replication designs were developed and implemented, it was necessary to make sure that each individual application that interfaced with the new system worked as designed.

The customer decided that the best way to do this was for the IT team to point their applications to the test servers, use the application, and then verify that the data was replicated as designed.

After each application was tested against the database, a series of validations would take place:

1. Application able to work properly with no errors.
2. Row counts on all nodes and all Subscribers were the same.
3. If row counts were not the same, the **tablediff** utility was run between the Publisher and the Subscriber to determine discrepancies.
4. Conflict Detection Report from the Replication Conflict Viewer was reviewed.
5. Replication Monitor was used to determine whether errors had occurred.

The following section details out the issues that were encountered as part of the unit testing effort, the cause, and the solution to the problem.

* + 1. P2P Replication with Conflict Detection and Read-Only Subscribers

It is possible to have a replication topology that incorporates a combination of P2P replication and standard one-way (read-only) transactional replication. P2P conflict detection requires the creation of a hidden column to all tables that are part of the P2P publication. P2P conflict detection in itself works well. However, unit testing detected a bug that prevented from transactions being successfully replicated to the read-only subscribers.

The bug manifests itself when a read-only, one-way transaction replication publication is created off of the articles that are already part of the P2P publication with conflict detection on. When a transaction is replicated, the following error appears in Replication Monitor:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2009-03-18 23:20:07.768 Agent message code 8144. Procedure or function sp\_MSins\_dbot1 has too many arguments specified.

2009-03-18 23:20:07.956 Category:COMMAND

Source: Failed Command

Number:

Message: if @@trancount > 0 rollback tran

2009-03-18 23:20:08.002 Category:NULL

Source: Microsoft SQL Server Native Client 10.0

Number: 8144

Message: Procedure or function sp\_MSins\_dbot1 has too many arguments specified.

This bug has been confirmed by the product team and is expected to be resolved in an upcoming service pack of SQL Server 2008. The workaround to this bug is to utilize the SQL delivery format rather than the stored procedure for the read-only publication. This can be done by right-clicking the publication and then clicking **Properties**.



Figure 3

In the **Properties** dialog box for the publication, under **Select a page**, click **Articles**.



Figure 4

After the page is refreshed, click **Article Properties**, and then click **INSERT delivery format.**



Figure 5

After this value was set, replication was able to replicate data as designed. However, it is important to note that changing the delivery format from stored procedure to SQL will result in degraded performance due to the packet size that is going across the network. Stored procedure calls are optimized, whereas SQL delivery format results in the full DML statement being sent to the distribution database and then ultimately delivered to the Subscribers. For more information about the performance impacts associated to changing the delivery format to SQL, see “Effect of Stored Procedure Delivery vs. SQL Delivery” below.

* + 1. Max Replication Text Size Option

When SQL Server is installed, the Max Replication Size value is set by default to 65536. If a record is inserted into a published table that exceeds that value, an error will occur at the application.

The following error will be returned to the calling application:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 Server: Msg 7139, Level 16, State 1, Line 1

Length of text, ntext, or image data (200) to be replicated exceeds configured maximum 10.

The statement has been terminated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In order to prevent this from occurring, **Max Text Replication Size** needs to be overridden to the maximum allowed, which is 2147483647, using the UI as described in section 4.1.2.

* + 1. P2P Publications with Spaces in Name

It is possible to use spaces in publication names. However, there is an extreme case in which P2P is not able to synchronize due to a confirmed bug. The bug will manifest itself during Subscriber synchronization.

The following error will be shown in Replication Monitor for the subscription that is currently being synchronized:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Cannot find a valid Queue Reader Agent ID for the subscription to Publisher <publisher>, database <database>, publication <publication>. The specified subscription to an updating Subscriber publication is not initialized. Run the Snapshot Agent, Distribution Agent, and Queue Reader Agent to initialize the subscription. (Source: MSSQLServer, Error number: 21758)

If the error is analysed, it appears as if P2P replication is attempting to find a valid Queue Reader Agent. P2P replication does not rely on a Queue Reader Agent so the error message is misleading. In order to resolve the issue, Microsoft Services Premier Support was engaged. Within one hour, Microsoft Services Premier Support was able to identify that this issue was a confirmed bug and provide the team with the appropriate workaround.

It is important to note that the bug described in this section does not always occur for P2P publications with spaces in their names. It is an extremely rare bug.

The details of the bug follow:

* During synchronization, the script is broken into multiple partial commands and written into the **MSrepl\_commands** table. Rarely, during this process, an extra space is inserted, causing this bug.
* It occurs only if the script is divided so that the last character of a specific command in the **MSrepl\_commands** table contains a space. For example:

First partial command:

----------------------

"if (@@microsoftversion >= 0x09000000) begin exec sys.sp\_MSaddsubscriptionarticles @publisher=N'SRVHOUSCS002',@publisher\_db=N'SCSS',@publication=N'SGTS "

 Second partial command:

-----------------------

" PTP7',@artid=62,@article=N'ContractCreditTypes',@dest\_table=N'ContractCreditTypes',@dest\_owner=N'dbo' end"

In this example, a space has been inserted after SGTS and before PTP7. When the partial commands are concatenated, there are two spaces, which causes the problem.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@publication=N'SGTS PTP7'

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

In order to resolve this issue, the P2P publication had to be re-created with no spaces in its name. After the necessary changes were made, P2P replication was able to synchronize successfully with all Subscribers. Also, to avoid similar issues, it was decided to not use any spaces in future publication names.

* + 1. Distribution Agent with OLE DB Streaming

When trying to replicate commands, the account that the Distribution Agent is running under needs to have access to the COM folder on the server in which the Distribution Agent is running. If it is a pull subscription, the Distribution Agent needs access on the Subscriber, and if it is a push subscription, the Distribution Agent needs access on the Distributor. If the appropriate permissions are not provided, the following error will occur when commands are distributed to the Subscriber.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The Distribution Agent failed to create temporary files in 'C:\Program Files\Microsoft SQL Server\100\COM' directory. System returned errorcode 5. (Source: MSSQL\_REPL, Error number: MSSQL\_REPL21100)

After providing the appropriate permissions to the account in which the Distribution Agent is running, you should be able to successfully deliver transactions to the Subscriber.

* + 1. Distribution Cleanup Failure

When replication is configured, a series of maintenance jobs are automatically created to help maintain and administer various replication components. One of these jobs is the Distribution Cleanup job. The purpose of this job is to keep the size of the distribution database manageable. By default, the job is scheduled to run every 10 minutes, and it executes the following command:

EXEC dbo.sp\_MSdistribution\_cleanup @min\_distretention = 0, @max\_distretention = 72

This command removes already distributed transactions that are older than 72 hours old. The stored procedure then calls a series of stored procedures, one of which is **xp\_cmdshell**. If the appropriate permissions are not provided, the following error occurs:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Replication-@rowcount\_only parameter must be the value 0,1, or 2. 0=7.0 compatible checksum. 1=only check rowcount: agent distribution@rowcount\_only parameter must be the value 0,1, or 2. 0=7.0 compatible checksum. 1=only scheduled for retry. Could not clean up the distribution transaction tables.

Unfortunately, the error does not provide enough information to be able to determine the underlying cause of the problem. In order to determine the cause of the error, the stored procedure that the agent calls as part of its steps was manually invoked via SQL Server Management Studio. After the procedure was executed, a more descriptive error was returned:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Msg 20015, Level 16, State 1, Procedure sp\_MSreplremoveuncdir, Line 83

Could not remove directory 'd:\ReplData\unc\<publication>\20090604180809\'. Check the security context of xp\_cmdshell and close other processes that may be accessing the directory.

Replication-@rowcount\_only parameter must be the value 0,1, or 2. 0=7.0 compatible checksum. 1=only check rowcount: agent distribution@rowcount\_only parameter must be the value 0,1, or 2. 0=7.0 compatible checksum. 1=only scheduled for retry. Could not clean up the distribution transaction tables.

Based on this error, it can be seen that the issue is associated to the execution of **xp\_cmdshell**. **xp\_cmdshell** is a procedure that calls a Windows® command shell and passes in a string for execution. **xp\_cmdshell**, by default, is turned off when installing SQL Server. In order to enable **xp\_cmdshell**, the following needs to be executed on the server instance:

-- To enable the feature.

EXEC sp\_configure 'xp\_cmdshell', 1

GO

-- To update the currently configured value for this feature.

RECONFIGURE

GO

In addition to the enabling of **xp\_cmdshell**, it is also mandatory to provide execution rights to the account that is executing **xp\_cmdshell** which is in this case, the SQL Server Agent account. After the appropriate rights are provided, the stored procedure will able to be called.

While it was expected that this would resolve the issue, another change had to be made in order for the job to complete successfully. **xp\_cmdshell** executes on the operating system under the account of the SQL Server service. In order to resolve the issue, the account in which SQL Server is running under needs to have full permissions to the directory referenced in the error.

After making all of the changes described here, the Distribution Agent job was able to complete successfully.

* + 1. Dummy Updates Issue with Conflict Detection

A dummy update occurs when you update a field with its existing value. When this happens, SQL Server changes the row version but the log reader will not replicate the changes (which is by design for dummy updates). The next time a valid update happens, the row versions will differ and will raise a conflict and changes may not replicate.

 From the error log:

A conflict of type 'Update-Update' was detected at peer 9 between peer 1 (incoming), transaction id 0x0000000013dd9ccd and peer 1 (on disk), transaction id 0x0000000013dd9cc1

This conflict occurs because different transaction IDs come from the same node. The different IDs indicate that the incoming transaction ID (9cc1) is different from the previously replicated on-disk ID (9ccd), which can happen only if the new update version was not replicated.

There is already an open bug, and it is expected to be fixed in Cumulative Update 4 for SQL Server 2008 Service Pack 1 (SP1), which is due in September of 2009.

* + 1. P2P Publication removal issue

If you created a P2P publication with conflict detection on along with a read-only subscription and you remove the P2P publication without turning off the detection first, you will get errors when you create a new P2P publication. The errors will be such as:

 Procedure or function sp\_MSupd\_dbot1 has too many arguments specified. (Source: MSSQLServer, Error number: 8144)

We found out **sp\_droparticle** was not clearing the urReplPeelId attribute when a P2P publication with conflict detection was dropped while the table was actively being subscribed to by a read-only subscription.

There are two workarounds to consider:

* Workaround #1: Before you drop P2P replication, use SQL Server Management Studio to set the current P2P subscription option **Allow P2P conflict detection** to False, apply the changes, and then drop the p2p replication. Next time you create the P2P replication, the issue (MSSQLServer, Error number: 8144) does not occur.
* Workaround #2:
1. Enable conflict detection again (use a new originator\_id):

exec sp\_configure\_peerconflictdetection @publication = 'tranp2p', @action = 'enable', @originator\_id = 10, @continue\_onconflict='true'

1. Disable the conflict:

exec sp\_configure\_peerconflictdetection @publication = 'tranp2p', @action = 'disable'

1. Drop and add your P2P publication again.

This is a bug that is expected to be fixed in Cumulative Update 4 for SQL Server 2008 SP1, which is due in September of 2009.

* + 1. Log Reader Agent Fails After Adding an Article to a Publication with binary large object (BLOB) Data

The Log Reader may fail in the following situation:

1. One or more tables with BLOB data exist in a concurrent snapshot publication with an active subscription.

2. You add a new article into this same publication and run the snapshot.

3. During the snapshot of the new article created in step 2, there is update activity to table with the BLOB data in the existing publication.

4. When the Log Reader processes the newly generated concurrent snapshot for the new article, it will try to generate reconcile commands for the active articles with BLOB data as well.

The Log Reader will report the following error:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The process could not execute 'sp\_replcmds' on 'HK-AGMDB-201'. (Source: MSSQL\_REPL, Error number: MSSQL\_REPL20011)

Get help: http://help/MSSQL\_REPL20011

Could not locate text information records for the column "Comment", ID 21 during command construction. (Source: MSSQLServer, Error number: 18773)

Get help: http://help/18773

The Log-Scan Process failed to construct a replicated command from log sequence number (LSN) {00023895:00009d11:0003}. Back up the publication database and contact Customer Support Services. (Source: MSSQLServer, Error number: 18805)

Get help: http://help/18805

The process could not execute 'sp\_replcmds' on 'HK-AGMDB-201'. (Source: MSSQL\_REPL, Error number: MSSQL\_REPL22037)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The reason that this error occurs is that when the Log Reader is reconciling the commands, some of the off-row data may not be available. In order to solve the issue and get replication up and running, you can try removing the article that creates the issue or modify replication configuration.

To remove the article:

1. Determine the article that is causing the problem. This can be done by taking the column referenced in the error message and searching the INFORMATION\_SCHEMA views for a column with the same name that also has a BLOB data type.
2. Remove this article from the publication.
3. 3) Stop all activity, and then add the article back into publication. If activity is not stopped, it is possible that you will encounter the same error on another article with BLOB data.

To modify replication configuration, use one of the following:

* Create new publications for any new articles. Do not add new articles to existing publications.
* Use the database snapshot **sync\_method**. This is available only in Enterprise Edition. Database snapshot is preferred over the native sync method because table locks are avoided, and it is preferred over the concurrent method due to the issue described above. The database snapshot sync method is not available as an option through the UI. This option is only available through the replication stored procedures delivery method.

SQL Server Books Online provides further information about the @sync method argument of **sp\_addpublication**.



Figure 6

* 1. System Testing

After it was verified that all issues resulting from Unit Testing were resolved, applications were working as designed, and replication encountered no issues, the next step was to test the applications as they would behave in a production system.

It was agreed by the customer that they would allocate resources to help test the applications and to migrate production jobs to the new server with the same schedules as the production server.

After each iteration of system testing, the following validations were performed:

1. Application able to work properly with no errors.
2. Row counts on all nodes and all subscribers were the same.
3. If row counts were not the same, the **tablediff** utility was run between the Publisher and Subscriber to determine discrepancies.
4. Conflict Detection Report from Replication Conflict Viewer was reviewed.
5. Replication Monitor was used to determine whether errors had occurred.
6. Replication Latency was measured.

System testing allowed the team to discover a very interesting trend. The team noticed that there was substantial delay in the replication of key monetary transactions. From a unit testing perspective, there was no latency. Transactions were replicated to the various nodes within 10 seconds. However, during system testing, latency approached a couple of hours.

Through investigation, it was determined that a specific application that performed daily bulk uploads of textual data, which resulted in millions of replication transactions and commands, was causing backlog.

These test results were reviewed with the customer and a conclusion was made that this delay was unacceptable. In order to resolve this issue, the project team had to analyse the published tables and determine whether there was a way to prevent this backlog. After investigation, the project team decided on creating two separate P2P publications. One publication comprised the tables that were associated to reference data that was textual in nature, and the other publication comprised transactional data that was financial in nature. By creating two publications, a new set of distribution agents was created, which distributed transactions for a specific publication. If there is backlog in one set, the other set can still continue. This approach was tested and the latency associated to monetary transactions was eliminated.

The following figure shows the modified replication topology.



Figure 7

* 1. Performance Testing

After it has been verified that everything is working as designed and is meeting customer functional requirements, the next step is to performance-test the topology to make sure that all bottlenecks have been identified and the topology is configured optimally.

Performance testing allowed the team to understand the effect of different replication configurations based on a controlled dataset. Each test was run multiple times and the results averaged together. Results were logged and performance improvements/degradation calculated.

* + 1. Testing Scenarios

Replication is highly configurable. Although data may be replicated successfully, different configurations may yield better performance. The following test scenarios configurations were executed as part of the performance testing.

|  |  |  |
| --- | --- | --- |
| Configuration | Primary Purpose | Test |
| **Push vs. Pull** | To determine the performance advantage/disadvantages of push vs. pull subscriptions | Configure distribution agent to be push for one set of tests and pull for another set of tests on the same dataset. |
| **Windows Server 2003 with SQL Server 2005 vs. Windows Server 2008 with SQL Server 2008** | To determine whether the improved TCP/IP stack on Windows Server 2008 results in performance gains | Set up a SQL Server 2005 instance on Windows Server 2003 and a separate SQL Server 2008 instance on Windows Server 2008. These instances would then subscribe to the same publication. |
| **Subscription Streams** | To determine whether parallel subscription streams result in performance gains | Modify the Distribution Agent startup parameters to include the **-SubscriptionStreams** command-line parameter. |
| **WAN Acceleration** | To determine how much WAN Acceleration improves replication performance | Turn off WAN Acceleration and run test and then turn on WAN Acceleration for the exact same data set. |
| **Multi-Master Peer Distribution** | To determine the effect on replication performance if one of the master peers are down and one peer needs to handle all of the distribution | Stop all distribution agents on Asia Core 2 so that Asia Core 1 needs to distribute data to all Subscribers. |

Table 2

Each one of the scenarios was tested using the same set of transactions. The first transaction consisted of a set DELETE statement that removed 347,737 transactions from the database. After the DELETE was performed, the same 347,737 rows were then re-inserted using a customer application. It is also important to note that the data that was being deleted/inserted included BLOBs that were stored in a **nvarchar(max)** column in SQL Server.

* + 1. Effect of Push Subscription vs. Pull Subscription

A subscription is a request for a copy of the data and database objects in a publication. A subscription defines which publication will be received, and where and when it will be received. When planning for subscriptions, consider where you want agent processing to occur. The type of subscription you choose controls where the agent runs. With a push subscription, the Distribution Agent runs at the Distributor, whereas with a pull subscription, agents run at the Subscriber. After a subscription is created, it cannot be changed from push to pull without dropping the existing push subscription and creating a new pull subscription.

***Distribution Agent Architecture***

*Reader Thread*

The reader thread reads the commands from the **MSrepl\_commands** table in the distribution database by using the OLE DB interface. The entire set of rows in the table is read in one batch. For example, if there are 500,000 pending changes in the **MSrepl\_commands** table, the reader thread will query all 500,000 rows and pass on the result set.

*Command Buffer*

The result set from the reader thread is placed into a 40-KB command buffer. There are two such command buffers. When the first one is full, the writer thread is signaled and the changes in the command buffer are applied to the destination. Concurrently, the second command buffer is filled with the remaining results and waits for the writer thread to consume the changes. This enables the writer thread to be fully active in the case of a high-transaction-volume system.

*Writer Thread*

The writer thread is responsible for reading the changes from a command buffer and applying it to the Subscriber. As soon as all the changes from the command buffer are consumed, the writer releases the buffer and moves on to read changes from the second command buffer, if needed. Because the buffer size is 40 KB, the size of the data pushed each time over the network is constrained to 40 KB.



Figure 8

With a push subscription, the Publisher propagates changes to a Subscriber without a request from the Subscriber. Changes can be pushed to the Subscriber on demand, continuously, or on a scheduled basis. The Distribution Agent runs at the Distributor in a push subscription.

With a pull subscription, the Subscriber requests changes made at the Publisher. Pull subscriptions allow the user at the Subscriber to determine when the data changes are synchronized. The Distribution Agent runs at the Subscriber in a pull subscription.

**Test Results**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Push (A) | Pull (B) | Performance Improvement ((A-B)/B)\*100 |
| **Server** | Asia Secondary Site Server 1 | Asia Secondary Site Server 1 |  |
| **Operating System** | Windows Server 2008 | Windows Server 2008 |  |
| **SQL Server Version** | SQL Server 2008 SP1/CU1 | SQL Server 2008 SP1/CU1 |  |
| **Subscription Type** | Push | Pull |  |
| **Delivery Type** | Stored procedure | Stored procedure |  |
| **Total Commands** | 695,477 | 695,477 |  |
| **Delivery Total Work Time (minutes)** | 142.23 | 106.19 | 33.93% |
| **Delivery Tran/Sec** | 40.75 | 78.4 |  |
| **Delivery Cmds/Sec** | 40.75 | 78.4 |  |
|  |  |  |  |
| **Server** | Asia Secondary Site Server 1 | Asia Secondary Site Server 1 |  |
| **Operating System** | Windows Server 2008 | Windows Server 2008 |  |
| **SQL Server Version** | SQL Server 2008 SP1/CU1 | SQL Server 2008 SP1/CU1 |  |
| **Subscription Type** | Push | Pull |  |
| **Delivery Type** | SQL | SQL |  |
| **Total Commands** | 695,477 | 695,477 |  |
| **Delivery Total Work Time (minutes)** | 330.5 | 211.8 | 56.04% |
| **Delivery Tran/Sec** | 17.53 | 30.21 |  |
| **Delivery Cmds/Sec** | 17.53 | 30.21 |  |

Table 3

These tests expose two critical facts:

* Pull subscriptions are significantly faster than push subscriptions. This is primarily due to the architecture of the distributor agent.
* Stored Procedure delivery format is significantly faster than SQL delivery format. This will be explained in the next section.
	+ 1. Effect of Stored Procedure Delivery vs. SQL Delivery

Replication enables you to configure the format in which commands are delivered. The default format is stored procedure delivery format. This is the most efficient delivery format. In addition to stored procedure delivery format, replication can be configured to call a customized stored procedure, SQL statements without column lists, and SQL statements with column lists.

Due to a P2P conflict detection bug with read-only subscriptions, SQL delivery format had to be utilized for the read-only subscribers, because it is not possible to subscribe to partial publications. A new publication had to be created with the complete set of articles to be replicated to the read-only subscribers using the SQL delivery format. When SQL delivery format is used, the actual DML statement is constructed and sent to the distribution database. It was not recommended to use customized stored procedures due to supportability as well as any unforeseen upgrade issues in the future.

With SQL Server 2008, it is possible to compress the size of the packet that goes over the network by passing in parameters instead of string literals. Parameters are the default for stored procedure delivery format, but string literals are the default for SQL delivery format. In order to pass in parameters instead of string literals, the following commands must be run for every article in the publication that uses SQL delivery format:

[http://msdn.microsoft.com/en-us/library/ms175980(SQL.90).aspx](http://msdn.microsoft.com/en-us/library/ms175980%28SQL.90%29.aspx)

An example of the command that needs to be executed for each article in the publication is:

EXEC sp\_changearticle @publication = '<publication>', @article = '<article>', @property = 'status', @value='parameters'

**Test Results**

|  |  |  |  |
| --- | --- | --- | --- |
|  | SQL Delivery Format (A) | Stored Procedure Delivery Format (B) | Performance Improvement ((A-B)/B)\*100 |
| **Server** | Asia Secondary Site Server 1 | Asia Secondary Site Server 1 |  |
| **Operating System**  | Windows Server 2008 | Windows Server 2008 |  |
| **SQL Version** | SQL Server 2008 SP1/CU1 | SQL Server 2008 SP1/CU1 |  |
| **Subscription Type** | Push | Push |  |
| **Delivery Type** | SQL | Stored Procedure |  |
| **Total Commands** | 695,477 | 695,477 |  |
| **Delivery Total Work Time (minutes)** | 330.5 | 142.23 | 132.37% |
| **Total Tran/Sec** | 17.53 | 40.75 |  |
| **Total Cmds/Sec** | 17.53 | 40.75 |  |

Table 4

Based on these tests, you can see the performance degradation associated with the SQL delivery format across a slow WAN link. This is primarily due to the size of the packets that are going across the network.

In order to get around this problem, a couple of workarounds were considered.

1. Create a separate publication specifically for the Web-only publication.

Because the Web publication is a subset of articles of both P2P publications, it was necessary to create a separate read-only publication. The P2P bug manifests itself when a read-only publication is created on top of a table article already published as part of a P2P publication. Because it is not possible to subscribe to partial P2P publications, one of the options was to create a separate P2P publication for the Web publication. The problem with this approach is that the transactional consistency may be compromised, because related tables may be part of separate publications. Each publication and each subsequent subscription to the publication will create a Distribution Agent job. These Distribution Agent jobs are independent of each other and may or may not run on different schedules. Because the agents are independent, there is no guarantee that all the transactions will be distributed in the right order if the publications are separated into three separate publications. For this reason, option 1 was eliminated.

1. Create a read-only subscription to the P2P publication.

This option builds off of the currently existing P2P publications. Instead of using the Configure P2P Topology Wizard, one of the options is to create a read-only subscription by right clicking on the publication and then clicking **Create Subscription**. This enables you to create a one way subscription to the P2P publication. Changes from the Subscriber will not flow back to the Publisher.

In order to use this topology, the full set of tables must exist in the Subscriber. This is because it is not possible to subscribe to partial publications. This prerequisite was discussed with the customer and it was determined that the data warehouse and read-only Subscriber would be able to have all tables in the subscriber databases, but this was not possible in the Web databases due to security issues and concerns.

The performance degradation was only associated to Subscribers that were connected to the WAN. These Subscribers were the data warehouse and the read-only Subscriber. The Web databases were not affected because they were on the same subnet. Because the data warehouse and the read-only Subscribers were able to subscribe to the full set of articles in the P2P publications, it was decided to create a read-only subscription to the P2P publications for these two Subscribers. The Web databases, because they were not allowed to receive the full set of articles, would continue to use their own separate Web publication with the SQL delivery format.

The following figure illustrates the new topology.

Figure 9

* + 1. Effect of Improved Windows Server 2008 TCP/IP Stack

Windows Server 2008 provides an improved TCP/IP stack. Internal tests have proven that SQL Server 2008 running on Windows Server 2008 result in substantial performance improvements over SQL Server 2005 running on Windows Server 2003. For more information, see “Geo-Replication Performance Gains with Microsoft SQL Server 2008 Running on Windows Server 2008” (<http://msdn.microsoft.com/en-us/library/dd263442.aspx>).

Per the MSDN® Case Study, the main features contributing to improvement in the end-to-end performance of SQL Server 2008 running on Windows Server 2008 over Windows Server 2003 are:

* Receive Window autotuning
* Send Buffer scaling
* Compound TCP

Due to the results from internal testing, it was worthwhile to perform similar tests on the customer replication topology with actual data to see if the same performance improvements would be able to be achieved. The test results are listed in the following table.

**Test Results**

|  |  |  |  |
| --- | --- | --- | --- |
|  | SQL Server 2005 (A) | SQL Server 2008 (B) | Performance Improvement ((A-B)/B)\*100 |
| **Server** | Asia Secondary Site Server 1 | Asia Secondary Site Server 2 |  |
| **Operating System** | Windows Server 2003 | Windows Server 2008 |  |
| **Subscription Type** | SQL | SQL |  |
| **Transaction Type** | Single Transaction Deleting Many Records | Single Transaction Deleting Many Records |  |
| **Total Commands** | 347,737 | 347,737 |  |
| **Delivery Total Work Time (minutes)** | 73.48 | 70.83 | 3.74% |
| **Total Tran/Sec** | 0 | 0 |  |
| **Total Cmds/Sec** | 78.87 | 81.81 |  |
|  |  |  |  |
| **Server** | Asia Secondary Site Server 1 | Asia Secondary Site Server 2 |  |
| **Operating System** | Windows Server 2003 | Windows Server 2008 |  |
| **Subscription Type** | SQL | SQL |  |
| **Transaction Type** | 1 Transaction / 1 Command | 1 Transaction / 1 Command |  |
| **Total Commands** | 347,737 | 347,737 |  |
| **Delivery Total Work Time (minutes)** | 222.98 | 215.68 | 6.63% |
| **Total Tran/Sec** | 25.99 | 26.87 |  |
| **Total Cmds/Sec** | 25.99 | 26.87 |  |

Table 5

Although performance gains were achieved, the performance gains were not significant. Greater improvement was seen with the utilization of a WAN Accelerator between the Asia Secondary Site and the Asia Primary Site. For more information, see WAN Acceleration section (5.3.5).

* + 1. Effect of WAN Accelerator

All three data centers are connected via a WAN. In between each data center is a third-party WAN Accelerator appliance that is used to optimize throughput as much as possible. WAN Accelerators employ data reduction, compression, and quality of service techniques such as priority to improve WAN efficiency.

It was the request of the customer to test the effectiveness of the WAN Accelerator in between their three data centers. The results of the tests are detailed in the following table.

**Test Results**

|  |  |  |  |
| --- | --- | --- | --- |
|  | WAN Accelerator Off (A) | WAN Accelerator On (B) | Performance Improvement ((A-B)/B)\*100 |
| **Server** | Asia Secondary Site Server 1 | Asia Secondary Site Server 1 |  |
| **Operating System** | Windows Server 2008 | Windows Server 2008 |  |
| **SQL Server Version** | SQL Server 2008 SP1/CU1 | SQL Server 2008 SP1/CU1 |  |
| **Subscription Type** | Pull | Pull |  |
| **Delivery Type** | SQL | SQL |  |
| **Transaction Type** | Single Transaction Deleting Many Records | Single Transaction Deleting Many Records |  |
| **Total Commands** | 347,737 | 347,737 |  |
| **Delivery Total Work Time (minutes)** | 70.83 | 20 | 254.15% |
| **Total Tran/Sec** | 0 | 0 |  |
| **Total Cmds/Sec** | 81.81 | 289.78 |  |
|  |  |  |  |
| **Server** | Asia Secondary Site Server 1 | Asia Secondary Site Server 1 |  |
| **Operating System** | Windows Server 2008 | Windows Server 2008 |  |
| **SQL Server Version** | SQL Server 2008 SP1/CU1 | SQL Server 2008 SP1/CU1 |  |
| **Subscription Type** | Pull | Pull |  |
| **Delivery Type** | SQL | SQL |  |
| **Transaction Type** | 1 Transaction / 1 Command | 1 Transaction / 1 Command |  |
| **Total Commands** | 347,737 | 347,737 |  |
| **Delivery Total Work Time (minutes)** | 215.68 | 123.28 | 74.95% |
| **Total Tran/Sec** | 0 | 0 |  |
| **Total Cmds/Sec** | 26.87 | 47.01 |  |

Table 6

As can be seen from these tests, the use of a WAN Accelerator in a replication topology that communicates via slow WAN speeds can greatly improve performance.

From a WAN perspective, SQL replication is very similar to storage replication. As a result, WAN acceleration provides similar benefits. For example, by implementing data reduction, compression, latency mitigation and loss correction, WAN acceleration results in faster replication times, better WAN bandwidth utilization, and faster host server performance - all across wider geographic distances.

* + 1. Effect of Subscription Streams

The **-SubscriptionStreams** parameter can greatly improve aggregate replication throughput. It allows multiple connections to a Subscriber to apply batches of changes in parallel, while maintaining many of the transactional characteristics present when using a single thread. If one of the connections fails to execute or commit, all connections will abort the current batch, and the agent will use a single stream to retry the failed batches. Before this retry phase completes, there can be temporary transactional inconsistencies at the Subscriber. After the failed batches are successfully committed, the Subscriber is brought back to a state of transactional consistency.



Figure 10

**Test Results**

|  |  |  |  |
| --- | --- | --- | --- |
|  | No Streams (A) | Streams (B) | Performance Improvement ((A-B)/B)\*100 |
| **Server** | Asia Secondary Site Server 1 | Asia Secondary Site Server 1 |  |
| **Operating System** | Windows Server 2008 | Windows Server 2008 |  |
| **SQL Server Version** | SQL Server 2008 SP1/CU1 | SQL Server 2008 SP1/CU1 |  |
| **Subscription Type** | Pull | Pull |  |
| **Delivery Type** | Stored Procedure | Stored Procedure |  |
| **Subscription Streams** | 0 | 4 |  |
| **Total Commands** | 695,477 | 695,477 |  |
| **Delivery Total Work Time (minutes)** | 106.19 | 98.8 | 7.47% |
| **Delivery Tran/Sec** | 78.4 | 58.66 |  |
| **Delivery Cmds/Sec** | 78.4 | 117.32 |  |
|  |  |  |  |
| **Server** | Asia Secondary Site Server 1 | Asia Secondary Site Server 1 |  |
| **Operating System** | Windows Server 2008 | Windows Server 2008 |  |
| **SQL Server Version** | SQL Server 2008 SP1/CU1 | SQL Server 2008 SP1/CU1 |  |
| **Subscription Type** | Pull | Pull |  |
| **Delivery Type** | Stored Procedure | Stored Procedure |  |
| **Subscription Streams** | 0 | 2 |  |
| **Total Commands** | 695,477 | 695,477 |  |
| **Delivery Total Work Time (minutes)** | 106.19 | 90.8 | 16.95% |
| **Delivery Tran/Sec** | 78.4 | 63.82 |  |
| **Delivery Cmd/Sec** | 78.4 | 127.64 |  |

Table 7

These tests show that adding too many streams doesn’t necessarily result in linear performance improvements. Testing must be done to determine the optimal number of subscription streams. In addition, it was discovered that the SQL delivery format cannot utilize the subscription streams feature of SQL Server 2008. This has been confirmed by the SQL Server Product Team and is due to SQL delivery’s inability to perform rowcount checks. Stored procedure delivery format is able to perform this rowcount check.

* + 1. Effect of Removing a Node

The purpose of this test was to determine what the impact of removing a node from the P2P publication topology would have on replication performance. P2P topology will attempt to distribute transactions in the most efficient matter.

For example, in the below topology, with transactions originating from Asia Core 1, P2P replication will determine to find the most efficient way to distribute transactions.



Figure 11

The first set of transactions will attempt to be replicated via Path 1. While Path 1 transactions are being replicated to United States Core 1 and United States Core 2, the next set of transactions will be distributed to Asia Core 2, because the transaction delivery is more efficient due to the Asia databases being on the same subnet. After Asia Core 2 database receives the next set of transactions, it will attempt to deliver the next batch to United States Core 1 and United States Core 2, but still maintaining transactional consistency. It is able to do this because each node keeps track of the original publisher LSN and performs a check on this peer LSN before applying a set of transactions.

In the scenario where Asia Core 2 were down for maintenance or disaster, it was important to understand whether there would be any performance degradation if a node was moved from the Asia subnet. This was done by simply removing the subscriptions between Asia Server 2 and United States Server 1 and Asia Server 2 and United States Server 2. By removing the subscriptions on Asia Server 1, it was possible to simulate server downtime.

The following figure illustrates the tested topology.

**Tested Topology**



Figure 12

**Test Results**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Four-Node Topology (A) | Three-Node Topology (B) | Performance Improvement((A-B)/B)\*100 |
| **Server** | Asia Server 2 | Asia Server 2 |  |
| **Operating System** | Windows Server 2008 | Windows Server 2008 |  |
| **SQL Server Version** | SQL Server 2008 SP1/CU1 | SQL Server 2008 |  |
| **Subscription Type** | Push | Push |  |
| **Delivery Type** | Stored Procedure | Stored Procedure |  |
| **Total Commands** | 347,737 | 347,737 |  |
| **Delivery Total Work Time (minutes)** | 2.21 | 36.5 | -93.94% |
| **Delivery Tran/Sec** | 2610.7 | 158.84 |  |
| **Delivery Cmds/Sec** | 2610.7 | 158.84 |  |
|  |  |  |  |
| **Server** | United States Server 1 | United States Server 1 |  |
| **Operating System** | Windows Server 2008 | Windows Server 2008 |  |
| **SQL Version** | SQL Server 2008 SP1/CU1 | SQL Server 2008 SP1/CU1 |  |
| **Subscription Type** | Push | Push |  |
| **Delivery Type** | Stored Procedure | Stored Procedure |  |
| **Total Commands** | 347,737 | 347,737 |  |
| **Delivery Work Time (minutes)** | 49.5 | 174.76 | -71.67% |
| **Delivery Tran/Sec** | 1.19 | 33.16 |  |
| **Delivery Cmds/Sec** | 116.87 | 33.16 |  |
|  |  |  |  |
| **Server** | United States Server 2 | United States Server 2 |  |
| **Operating System** | Windows Server 2008 | Windows Server 2008 |  |
| **SQL Version** | SQL Server 2008 SP1/CU1 | SQL Server 2008 SP1/CU1 |  |
| **Subscription Type** | Push | Push |  |
| **Delivery Type** | Stored Procedure | Stored Procedure |  |
| **Total Commands** | 347,737 | 347,737 | -71.68% |
| **Delivery Work Time (minutes)** | 49.5 | 174.80 |  |
| **Delivery Tran/Sec** | 1.19 | 33.15 |  |
| **Delivery Cmds/Sec** | 116.71 | 32.91 |  |

Table 8

These tests conclude that having a separate peer on the same subnet will significantly improve the performance of P2P replication. This is due to the offloading of P2P commands to another master while the original publisher continues to distribute transactions to other subscribers. P2P replication is able to keep transactional consistency because replication keeps track of the original publisher database as well as its associated LSN.

Due to the results, it was decided to further distribute the replication to prevent one single node from being responsible for the distribution of transactions. This resulted in the final replication topology:

**Final Topology**



Figure 13

1. Deployment

The final phase of the governance model is the Deploy Phase. The goals of this phase are to place the solution into production at the designated environments and to facilitate a smooth transfer of solution from project team to the operations team as soon as possible.

* 1. Go-Live Planning

Setting up replication in a controlled environment is not difficult. Connected users, back-office applications, general database availability, and most importantly, customer-facing applications are not critical when setting up a test/development environment. Down time is not an issue in a test/development issue. However, application/database downtime is critical when it is time to go live with P2P.

In order to limit downtime, the go-live process needs to be thoroughly analysed. It is critical to work with the main stakeholders of the database to understand acceptable downtime. It is important to understand how long it takes for specific go-live tasks to complete. Full/Differential/Transaction Log database backups and restores, database upgrades, publication/subscription creation, snapshot delivery, and the FTP of database backups to remote servers are just a few of the many tasks that need to be analysed to understand the amount of time necessary to complete all go-live tasks. All of these items will dictate how, what, when, where, why, and in what order these tasks will be performed when migrating to production.

Extensive downtime was not an option. The steps necessary to perform the upgrade, the transfer of database backups, and replication setup took well over a day due to various infrastructure challenges associated to the transfer of database backups from the Asia-based data center to the U.S.-based data center. In an ideal situation, a downtime of a day would have been the suggested configuration, but the nature of the business did not allow for that.

In order to minimize downtime, we analysed ways to limit downtime yet still provide the stability and time necessary to complete the migration steps. After discussing all of the options with the customer, the option to reduce the amount of downtime consisted of the creation of a read-only transactional replication publication//subscription from their existing SQL Server 2005 database to the SQL Server 2008 database.

The following table lists the final migration time and application availability.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Process | Application Down | Downtime (min) |
| 1 | Upgrade SQL Server 2005 to SQL Server 2008 via Database Restore script | No | N/A |
| 2 | Turn off write applications | Yes | 5 |
| 3 | Take differential backup | Yes | 5 |
| 4 | Create publication | Yes | 1 |
| 5 | Create Subscription | Yes | 1 |
| 6 | Turn on write apps | No | N/A |
| 7 | Restore differential backup in SQL Server 2008 | No | N/A |
| 8 | Perform full database backup on SQL Server 2008 | No | N/A |
| 9 | Transfer database to remote servers | No | N/A |
| 10 | Setup Replication Topology | No | N/A |
| 11 | Synchronize Replication | No | N/A |
| 12 | Start Replication Agent on SQL Server 2005 | No | N/A |

Table 9

As illustrated in the table, total application/database downtime consisted of 12 minutes. Furthermore, these steps were performed during nonstandard work hours, which further reduced the potential downtime impact. The migration methodology also provided the customer more liberty to migrate their applications to the new server as data was constantly being synchronized with the transactions originating from the SQL Server 2005 database. After the customer was ready to cut over to the new system, the only steps necessary were to stop the replication agent on SQL Server 2005 and then repoint applications to the SQL Server 2008 instance.

* 1. Go-Live Testing

Testing of the go-live process is just as important as the go-live planning. After the go-live plan is created, it is important to script out those tasks and run them through as many times as possible to obtain an understanding of what issues may occur and how long the migration will take.

With proper testing, the actual go-live process should be less risky, prescriptive, and more predictive in nature.

1. Conclusions

P2P replication is a mature technology, and it is enterprise ready. It is currently being utilized inside Microsoft and many other global enterprises throughout the world.

The main purpose of this document was to prove that the replication topology would be able to be replicated to the peer servers within the expected timeframes and to help understand the issues that may surface as part of the implementation. In addition, the performance tests allowed the team to understand the effect of different replication configurations based on a controlled dataset.

Replication in general is not difficult to implement. However, properly designing, testing, and optimizing replication is iterative, and adequate time must be allocated to ensure that all components are extensively tested.