Geo-Replication Performance Gains with Microsoft SQL Server 2008 Running on Windows Server 2008

The MSDN Case Study

Writers and Technical Reviewers: Scott Greene, Sunjeev Pandey, Gopal Ashok, Katarzyna Puchala, Peter Gvozdjak, Vaughn Washington, Matt Lee, Tim Cashman, Mark McAlpin

Published: November 2008

Applies to: SQL Server 2008; Windows Server 2008

Summary: Microsoft.com Engineering Operations (MSCOM Ops), the team responsible for architecting and managing many of the most heavily trafficked Microsoft Web sites, recently documented substantial replication performance improvements for both push and pull subscriptions in Microsoft® SQL Server® 2008 running on Windows Server® 2008—up to 100 times faster than Microsoft SQL Server 2005 running on Windows Server 2003. This paper provides IT pros with an in-depth look at the tests conducted by the MSCOM Ops team, insight into the results, and technical analysis of the enhancements to Windows Server 2008 and SQL Server 2008 that enable wide area network (WAN)–based geo-replication.

Introduction

The Microsoft.com Engineering Operations (MSCOM Ops) team is continuously searching for ways to improve the performance of its Web sites, which include Microsoft.com, the Web site for the MSDN® developer program, and TechNet. These Web sites are complex, distributed applications with Microsoft® SQL Server® database platforms that attract an enormous amount of traffic from users around the world. Site performance and user experience are two of the principal metrics that the team uses to evaluate its own success.

In late summer of 2008, MSCOM Ops initiated extensive replication performance testing, comparing SQL Server 2005 running on Windows Server® 2003 with SQL Server 2008 running on Windows Server 2008. By using both operating environments to replicate data between a server located in Tukwila, Washington, and another located in Blueridge, Virginia—a distance of approximately 3,000 miles—the team discovered that SQL Server 2008 running on Windows Server 2008 yielded up to 100 times faster performance without requiring any expensive wide area network (WAN) acceleration hardware.

Encouraged by these impressive results, the team decided to test replication of "live" content from its MSDN database in Redmond, Washington, to a remote data center in Dublin, Ireland. The performance enhancements in SQL Server 2008 running on Windows Server 2008 enabled the team to successfully replicate this content and demonstrate that WAN-based geo-replication is feasible in a real-world scenario. Further, the MSCOM Ops team used this capability to cut page-to-load times on its MSDN Web site by approximately 33 percent, resulting in a dramatic user experience improvement in Europe.

Now, in addition to providing an improved Web experience to people around the world, MSCOM Ops benefits from enhanced business continuity. And, with solid evidence of the feasibility of WAN-based geo-replication, MSCOM Ops plans to expand its implementation of this solution.

This white paper details the business challenges that prompted the MSCOM Ops team to explore WAN-based geo-replication, describes the testing the team performed, documents the test results, and conveys insights and conclusions based on these results.

Key Operational Challenges and Opportunities for MSCOM Ops

The Microsoft.com Engineering Operations (MSCOM Ops) team (http://technet.microsoft.com/en-us/mscomops/default.aspx [http://technet.microsoft.com/en-us/mscomops/default.aspx]) architects, deploys, manages, and sustains highly available, highly scalable, and highly secure Web and SQL Server infrastructures for many key corporate Microsoft Web sites. Microsoft.com (http://www.microsoft.com [http://www.microsoft.com/default.aspx]), Microsoft Update (http://update.microsoft.com [http://update.microsoft.com/default.aspx]), the site for the

MSDN developer program (http://msdn.microsoft.com [http://msdn.microsoft.com/default.aspx]), and the Download Center (http://www.microsoft.com/downloads [http://www.microsoft.com/downloads/default.aspx]) are just a few examples of the sites and databases that the team runs and supports. MSCOM Ops aggressively adopts new technologies and works closely with the teams that develop Windows Server, SQL Server, and Internet Information Services (IIS).

The MSCOM Ops team faces the persistent challenge of efficiently distributing massive amounts of data across its network to a large number of users around the world while minimizing latency. Beginning recently, MSCOM Ops has addressed this challenge by providing copies of data and content in multiple data centers worldwide.

As a core part of its mission, the team is actively involved in early technology adoption (http://technet.microsoft.com/en-us/library/cc627315.aspx [http://technet.microsoft.com/en-us/library/cc627315.aspx]) programs at Microsoft. It maintains a high awareness of and responsiveness to emerging industry trends. One current trend that MSCOM Ops closely tracks is the global proliferation of data centers, which is driven in part by investments from Microsoft to improve the performance of its Web sites and online services. Leaders from MSCOM Ops work closely with developers to ensure that the applications they build will function optimally in architectures that include geographically dispersed data centers.

Pilot-Testing Geo-Replication

Seeking to improve the user experience across all of the sites it manages, MSCOM Ops continuously tests various methods of replication. Replication is the process of copying data from one server location to another to synchronize the data and improve data availability.

MSCOM Ops was eager to test the enhanced TCP/IP stack in Windows Server 2008 (http://www.microsoft.com/windowsserver2008/en/us/default.aspx [http://www.microsoft.com/windowsserver2008/en/us/default.aspx]) in its own production environment to evaluate potential gains in replication performance. Running SQL Server 2005 on an early build of Windows Server 2008, the team discovered that the Replication Engine of SQL Server 2005 was not taking full advantage of the TCP/IP enhancements in Windows Server 2008. The SQL Server team advised that new code introduced in SQL Server 2008 addressed this problem.

The MSCOM Ops team was confident that by using SQL Server 2008 running on Windows Server 2008, they could efficiently replicate data to remote data centers, even as the volume of application data continued to grow. The resultant geo-redundancy would provide global consistency to applications and potentially reduce constraints on application architectures. The team tested SQL Server 2005 running on Windows Server 2003 and performed similar tests with SQL Server 2008 running on Windows Server 2008.

The Publisher and Distributor databases for the MSDN applications are located in a data center in Tukwila, Washington. The Subscriber databases are hosted in separate facilities in Washington and California. As shown in Table 1, the team calculated baseline latency of four milliseconds between the MSCOM Ops data centers in Washington; a maximum of 23 milliseconds between facilities in Washington and California; and 150 milliseconds from Tukwila, Washington to Dublin, Ireland.

Data Center	Tukwila	Quincy	Santa Clara	Blueridge	Dublin
Tukwila, Washington	1 ms	4 ms	19 ms	77 ms	150 ms
Quincy, Washington	4 ms	1 ms	23 ms	68 ms	146 ms
Santa Clara, California	19 ms	23 ms	1 ms	79 ms	156 ms
Blueridge, Virginia	77 ms	68 ms	79 ms	1 ms	84 ms
Dublin, Ireland	150 ms	146 ms	156 ms	84 ms	1 ms

Table 1: Baseline latency in milliseconds (ms) between Microsoft data centers used in MSCOM Ops geo-replication pilot testing

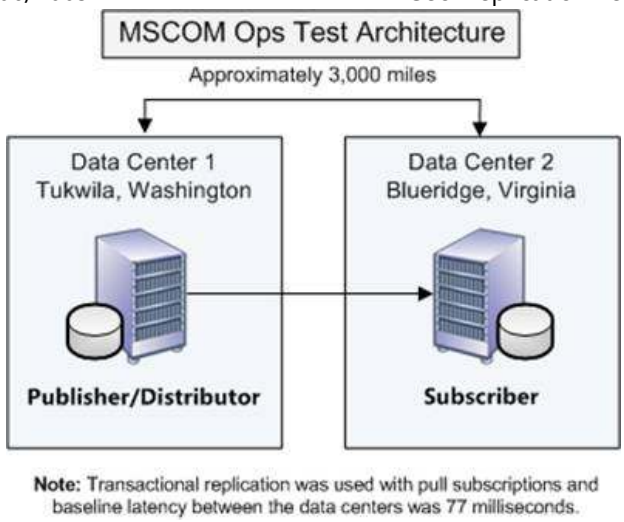

Figure 1: The replication topology used to compare performance of SQL Server 2008 running on Windows Server 2008 with SQL Server 2005 running on Windows Server 2003

SQL Server 2005 Running on Windows Server 2003

The team's previous attempts to replicate data between Microsoft data centers located in Redmond, Washington, and Virginia by using SQL Server 2005 running on Windows Server 2003 helped reduce latency to a certain extent. But, to account for the real-world demands of the data-centric applications targeted for geographic redundancy, the team decided to conduct further tests on this platform by using the same data center locations.

The following describes the methodology that the MSCOM Ops team used to evaluate replication performance of SQL Server 2005 running on Windows Server 2003.

- Tests were conducted between data centers located in Redmond and Virginia—a distance of approximately 3,000 miles.
- Baseline testing was conducted for transactional replication of both push and pull subscriptions.
- Push subscription model was used for initial test and pull subscription model was used for all subsequent testing.
- Rows in database tables were 1K in length.
- Rows were inserted in increments of 1,000, 10,000, and 100,000.
- Each test pass was run three times, and run times were averaged.
- The distribution database was housed on the Publisher in the replication topology for all tests.
- Tests included "live" data from the MSDN database, including two tables that had schemas identical to those used in production.

Based on a scenario in which the database system was separated into reads and writes, latency was determined during testing to be as high as four seconds.

Windows Server 2008 and SQL Server 2008

In late summer of 2008, the MSCOM Ops team initiated testing of the updates to the Database Engine in SQL Server 2008 alongside the improved TCP/IP stack in Windows Server 2008.

The following describes the methodology that the MSCOM Ops team used to evaluate replication performance of SQL Server 2008 on Windows Server 2008.

- Tests were conducted between data centers located in Redmond and Virginia—a distance of approximately 3,000 miles.
- Baseline testing was conducted for both push and pull subscriptions with transactional replication.
- Push subscription model was used for initial test and pull subscription model was used for all subsequent testing.
- Rows in database tables were 1K in length.
- Rows were inserted in increments of 1,000, 10,000, 100,000 and 1,000,000.
- Each test pass was run three times, and run times were averaged together.
- The distribution database was housed on the Publisher in the replication topology for all tests.
- Live data from the MSDN database, including two tables with identical schemas used in production, were included in tests.

- The agent profile changes for testing on SQL Server 2008 and Windows Server 2008 were made by using 1,000,000 records, equal to 15.5 gigabytes (GB) of data transfer.

Test Results Comparison

Testing showed that using transactional replication with SQL Server 2008 running on Windows Server 2008 dramatically outperformed SQL Server 2005 running on Windows Server 2003. As illustrated in Table 2, the most substantial performance gains occurred when the Publisher and Subscriber were both running SQL Server 2008 on Windows Server 2008.

Testing also showed that the scope of the performance gains correlated with the type of replication and the type of data. Push subscription replication of character data with SQL Server 2008 running on Windows Server 2008 yielded a 104 percent increase over SQL Server 2005 running on Windows Server 2003, and pull subscription replication of the same data yielded a 1,298 percent gain. The team noted that changing the **PacketSize** or **ReadBatchSize** parameters in the Replication Distribution Agent (http://msdn.microsoft.com/en-us/library/ms147328.aspx [http://msdn.microsoft.com/en-us/library/ms147328.aspx]) profile during testing did not significantly affect performance; these changes resulted in a savings of less than 60 seconds for replicating 1,000,000 rows of **varbinary (max)** data equal to 15.5 GB of data moved.

It should be noted that not all of the disk performance counters were relevant during testing, as the partitioning of the disks to support dual boot and two instances of SQL Server on each partition rendered the disk performance counters questionable. The key take-away from the disk counters is that the distribution database files are "hot spots" during replication, and that the process is input and output intensive. Disk-queue length averaged 150 for the Publisher with the test hardware.

MSCOM Ops is continuing to drill down into various parameters and variables in subsequent testing to further develop guidance and best practices. However, based on the substantial performance gains witnessed in the initial round of testing, the team believes it is possible to build a geographically redundant, fault-tolerant database system, including a high read and write version.

Performance Indicators	**Test Scenarios**	**SQL Server 2005 on Windows Server 2003 (A)**	**SQL Server 2008 on Windows Server 2008 (B)**	**Performance Gains or Losses [(A-B)/B]*100**
CPU Utilization (%)	All	15%	15%	0%
Memory	All	99%	99%	0%
Push Replication	1-GB 1,000,000 1k character records	226.12 *(minutes)*	110.42 *(minutes)*	104.78%
Pull Replication	1-GB 1,000,000 1k character records	174.87 *(minutes)*	12.5 *(minutes)*	1298.96%
Linked Server	10-MB 10,000 1k character records	107.6 *(minutes)*	113.6 *(minutes)*	–5.28%
Push Replication	112-MB 100,000 **varbinary (max)** records	247.07 *(minutes)*	59.13 *(minutes)*	317.84%
Pull Replication Records	112-MB 100,000 **varbinary**	223.18 *(minutes)*	1.95 *(minutes)*	11345.13%

	(max) records			
Snapshot Replication	11.3-GB 10,100,000 1k records	Not tested	22.75 *(minutes)*	Comparison not available

Table 2: Results from comparison testing of SQL Server 2008 running on Windows Server 2008 and SQL Server 2005 running on Windows Server 2003

Note: See Appendix B for details on hardware and software used.

Replication in the Real World: The MSDN Case Study

The MSCOM Ops team was encouraged by initial testing that showed impressive performance gains enabled by SQL Server 2008 running on Windows Server 2008. The team hoped to achieve similar replication improvements by using "live" data from the MSDN production environment. The primary goal of further testing was to take advantage of WAN–based geo-replication to reduce page-to-load times and boost the overall performance of its MSDN Web sites in Europe and Asia. Page-to-load time is the amount of response time required for each page of a Web site to display.

MSDN Architecture and Performance Challenge

The MSDN Web site is built on a large, distributed application that the team architected and designed for resilience and performance. As illustrated in Figure 2, the MSDN site is also a data-centric application, with almost all of its content stored within SQL Server. The content for the MSDN Web site is dynamically rendered from the servers running SQL Server by using XSL transforms with the results cached.

The MSCOM Ops team has documented that because the MSDN site is heavily database driven, it tends to exhibit poorer performance in page-to-load times in Europe and Asia than in the United States. This is caused by the frequent calls back to databases located in data centers in the United States.

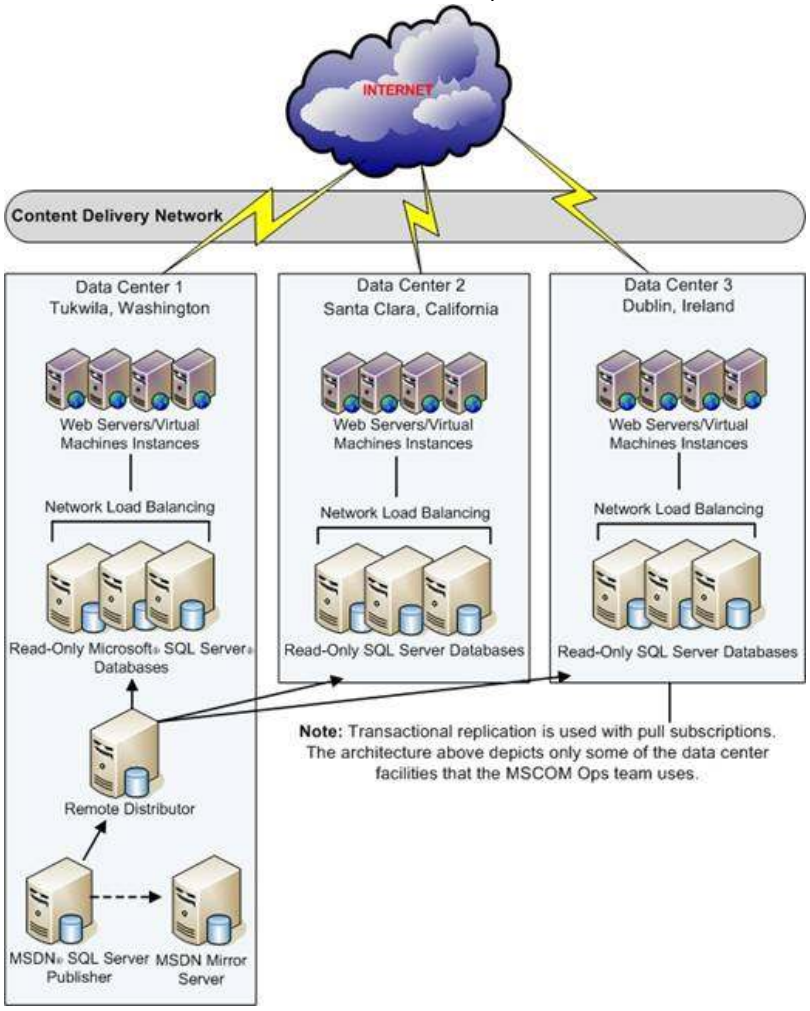

Figure 2: MSDN application architecture

Replicating MSDN Data to Dublin, Ireland: Comparison Testing of Initialization Methods

To determine whether it was feasible to replicate MSDN database content to data centers across Europe and Asia, the MSCOM Ops team first needed to complete initialization of the database content to a data center in Dublin, Ireland.

The team tested several approaches to initializing replication, including the following methods:

- Compressing the snapshot
- Completing the snapshot locally and manually copying the snapshot data to Dublin
- Completing the snapshot locally, compressing the files through a zipping process, and copying the files manually to Dublin
- Completing the snapshot directly to Dublin
- Initializing snapshot replication from backup by copying a database backup from Tukwila, Washington, to Dublin*

***Note:** All of the copies were initiated from Dublin so that the replication could take advantage of the TCP/IP stack improvements.

The team tried the snapshot replication process by using various distribution agent (http://msdn.microsoft.com/en-us/library/ms147328.aspx [http://msdn.microsoft.com/en-us/library/ms147328.aspx]) settings, such as adjusting the **CommitBatchSize**, **CommitBatchThreshold**, **OledbStreamThreshold**, **PacketSize**, and **MaxBcpThreads**. The team discovered that the fastest way to establish transactional replication to Dublin was to initialize replication from a backup copy of the database. After the team successfully completed replication, the data transfer proceeded as expected based on test results and the known latency.

Key Replication Benefits

Based on the positive pilot test results for SQL Server 2008 running on Windows Server 2008, MSCOM Ops determined that it is feasible to replicate database content for its MSDN Web site to data centers in Europe and Asia. In addition to performance improvements, increased resiliency, and system failover benefits, the MSCOM Ops team now has the confidence to adopt similar solutions and strategies for other sites and applications.

Performance Improvements for Local Users

As MSCOM Ops introduced additional infrastructure in data centers located in Europe and Asia (with particular emphasis on Europe in this scenario), the page-to-load time dropped significantly, resulting in an improved experience for users in those regions. This involved replicating data from the write server in Tukwila to a load-balanced set of servers running SQL Server 2008 in Dublin. Figure 3 illustrates a decrease in page-to-load time of more than a second for noncached visits to the MSDN home page from users in Europe directed to these servers. This is compared to noncached visits to the MSDN home page from European users that required connecting to SQL Server databases in the United States. The differential shown in Figure 3 represents a 33 percent page-to-load time improvement.

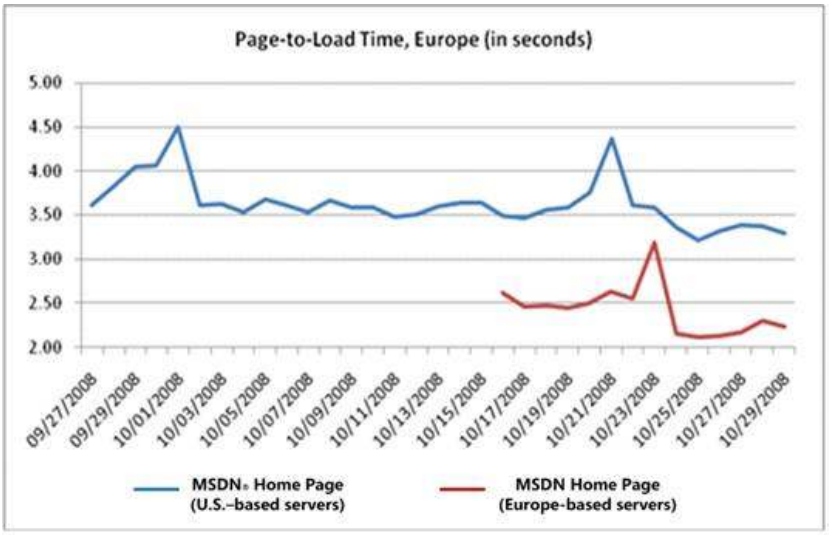

Figure 3: Page-to-load times for MSDN Web site in Europe

Resiliency and System Failover

By replicating MSDN application data to a remote data center in Dublin, MSCOM Ops has bolstered the continuity of its global operations. Although the team already uses database mirroring in its publishing system in Tukwila, MSCOM Ops now has the added assurance that in the event of a catastrophic data loss event, the team could move its database publishing to Dublin.

Conclusion

Both lab and real-life testing by the MSCOM Ops team indicate that highly trafficked Web sites can gain the benefits of geo-replication most effectively when the site is built on SQL Server 2008 running on Windows Server 2008. Based on solid evidence of the feasibility of WAN–based geo-replication, MSCOM Ops plans to expand its implementation of this solution.

In addition, the MSCOM Ops team learned several valuable lessons because of its extensive performance testing of SQL Server 2008 running on Windows Server 2008, including:

- Windows Server 2008 and SQL Server 2008 with the TCP/IP stack improvements and partnering with application development teams can bolster global user experiences, produce higher availability, higher scalability, and better resiliency for sites, services, and applications through WAN-based geo-replication.
- Replication performance is significantly better for pull subscription scenarios than push subscriptions.
- The solution identified in this paper will not work for all applications, particularly applications that cannot handle the inherit latency involved with replicating data between geographically dispersed data centers.

Appendix A: A Closer Look - Examining the Enhancements

in Windows Server 2008 and SQL Server 2008 That Enable Faster Performance

The purpose of this appendix is to provide greater technical detail regarding the specific improvements in Windows Server 2008 and SQL Server 2008 that enable faster throughput performance for database replication. Section 1 highlights three key networking improvements in Windows Server 2008. Section 2 describes how SQL Server 2008 uses Send Buffer Scaling and Ideal Send Backlog in Windows Server 2008 to improve end-to-end performance of large data transfers over a WAN. This section also discusses ODBC improvements in SQL Server 2008. Section 3 focuses on how SQL Server 2008 takes advantage of the enhanced TCP/IP stack in Windows Server 2008 to improve replication performance in both push and pull subscription scenarios. In particular, this section details how SQL Server 2008 uses the Receive Window Autotuning feature in Windows Server 2008 to boost replication performance.

Section 1: Three Key Networking Improvements in Windows Server 2008

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

1.1. Receive Window Autotuning

The TCP Receive window controls the amount of data that can be sent at any one time, and it provides receiver-side flow control for TCP peers. The window is the amount of data that the receiver permits the sender to send on any given connection. The sender can send only the number of bytes of the byte stream allowed by the window. The maximum TCP Receive window is a fixed value (65,535 bytes), which limits the throughput of any given connection. For example, a 65,535-byte Receive window can only achieve an approximate throughput of 5.24 Mbps on a transmission path that has a 100 millisecond (ms) roundtrip delay, regardless of the path's actual bandwidth. To accommodate larger window sizes for high-speed transmission paths, the window can be scaled by the TCP Window Scale factor that, when combined with the 16-bit Window field in the TCP header, can increase the Receive window size to a maximum of approximately 16 MB.

To optimize TCP throughput, the networking stack in Windows Server 2008 automatically tunes the Receive window on a per-connection basis. The optimal Receive window size for any given connection is determined by measuring the bandwidth delay product (BDP) of the path (how much data can be in transit at any given time) and the application retrieve rate (how fast the data can be consumed by the receiver), and adapting the window size for ongoing transmission path and application conditions. Receive Window Autotuning has a number of benefits. Specific applications no longer need to specify TCP window sizes through Windows Sockets options. IT administrators no longer need to manually configure a TCP Receive Window size for specific computers. With Receive Window Autotuning, a Windows Server 2008 peer will typically allow for much larger Receive window sizes than a Windows Server 2003 TCP peer, thus allowing the former to receive more TCP data segments without having to wait for the pipe to become available. This results in an overall increase in network performance. The higher the BDP and application retrieve rate for the connection, the better the performance.

1.2. Send Buffer Scaling

When a Windows Sockets application sends data, the data is copied to an intermediate Winsock buffer, which is then submitted to the TCP/IP stack for sending. Currently, the default Winsock buffer size is 8 kilobytes (KB), which effectively throttles how efficiently an application is able to saturate the connection's pipe—the BDP. This has the effect of limiting the number of bytes in transit on a TCP connection.

With Send Buffer Scaling, TCP/IP keeps track of the number of bytes that a connection can sustain "in flight," which can be much greater than the default Winsock limit. This Ideal Send Backlog (ISB) value can be queried by the application to update its send buffer size. Send Buffer Scaling enables the application to adjust its buffer sizes based on the current networking conditions, such as congestion, delay, and Receive window size. The changes in ISB are primarily driven by the receiving TCP's Receive Window Autotuning. By taking advantage of the new ISB application programming interfaces (APIs), Winsock applications such as SQL Server 2008 can achieve better TCP throughput on connections that have a high BDP (larger than 8 KB). The tradeoff for these throughput improvements is higher—yet, still optimal—memory consumption.

1.3. Compound TCP

TCP has algorithms in place that are meant to prevent the sending TCP peer from overwhelming the network—namely, slow start and congestion avoidance. These algorithms increase the Send window (the amount of data that the TCP sender can send) when starting up the connection and when recovering from TCP packet loss. These algorithms work well for smaller bandwidths, latencies, and Receive window sizes. However, for a TCP connection that has a large Receive window size and large BDP, for example geographically replicating data across a high-speed

WAN link, these algorithms do not increase the Send window size fast enough to take advantage of the connection's capabilities.

To make better use of a connection's capabilities in these situations, the networking stack in Windows Server 2008 includes a new TCP algorithm called Compound TCP (CTCP), in addition to regular New Reno TCP. CTCP more aggressively increases the Send window for connections that have large Receive window sizes and BDPs. To maximize throughput, CTCP monitors variations in packet delay in addition to packet losses. CTCP also ensures that its behavior does not negatively impact other TCP connections.

Section 2: Microsoft SQL Server 2008 Data Programmability

The main factors contributing to end-to-end performance improvements for large data transfers over a WAN by using SQL Server 2008 running on Windows Server 2008 are the Send Buffer Scaling and Ideal Send Backlog (ISB) features of Windows Server 2008.

When SQL Server 2008 running on Windows Server 2008 sends data to a client over the TCP protocol, it asks the operating system for notifications of the ISB size value changes and adjusts the per-socket Send buffer size accordingly. This lets SQL Server 2008 take advantage of the Send buffer scaling available in Windows Server 2008 and, in turn, increases the throughput over connections that have large BDP, such as replication of data across geographically distributed locations over a WAN. Instead of using an application-determined value, SQL Server 2008 uses a value based on an ISB notification, as dynamically determined by the Windows networking stack based on measurements of system parameters, like the delay characteristics of the connection. Similarly, the SQL Server 2008 Native Client ODBC driver and OLE DB provider use Send Buffer Scaling and ISB when running on Windows Server 2008 and sending data to SQL Server 2008.

2.1. SQL Server ODBC Improvements

SQL Server 2008 Native Client introduces several performance and scalability improvements from which applications will benefit after upgrading and without making any code changes. The most significant improvements fall into two high-level categories:

- Invoking stored procedures with ODBC call syntax and OLE DB remote procedure call (RPC) syntax
- Executing prepared statements in ODBC

Invoking Stored Procedures

Using stored procedures in applications has long been a performance best practice. In SQL Server 2008, the ODBC driver and OLE DB provider were further optimized by reducing the number of instructions executed in the driver or provider during this scenario. This translates into lower CPU usage in the application tier that uses SQL Server Native Client. Specifically, this was accomplished by improving the algorithms used during the process of translating application data to the serialization format of SQL Server and reducing the number of times data was copied. In performance testing, MSCOM Ops observed reductions in CPU usage from 5 to 10 percent.

Executing Prepared Statements in ODBC

Prepared execution generally provides optimal performance in cases where a statement, optionally with parameters, is executed multiple times. In ODBC, this is accomplished by using the SQLPrepare and SQLExecute APIs. In the SQL Server 2008 Native Client ODBC driver, the performance of this model is further improved for several specific usage patterns. These usage patterns include executing prepared or nonprepared statements with parameter arrays of size greater than 1. This is accomplished by setting SQL_ATTR_PARAMSET_SIZE to statements with greater than 1024 parameters; executing statements with data-at-execute parameters can be accomplished by using the SQL_DATA_AT_EXEC or SQL_LEN_DATA_AT_EXEC macros in the *StrLen_or_IndPtr* argument of **SQLBindParameter**. Specifically, this performance improvement was accomplished by using a more efficient serialization format for the statement and associated parameter data. In performance testing, MSCOM Ops observed performance improvements of up to 80 percent for end-to-end performance. Actual performance results will vary based on number of parameters, parameter array row sizes, and parameter data types and data values.

Section 3: Replication Using Microsoft SQL Server 2008

As illustrated in this white paper, throughput performance of SQL Server 2008 transactional replication is improved significantly when SQL Server 2008 runs on Windows Server 2008. This section offers a detailed, architecture-level explanation of the replication process. Further, it describes how specific improvements in the TCP/IP stack in Windows Server 2008 and corresponding improvements in the SQL Server client layer combine to provide this performance boost.

3.1. Defining the Difference Between Push and Pull Subscription

Transactional replication consists of two subscription types or models: push subscription and pull subscription. As shown in Figure 4, the primary difference between push and pull subscriptions is where subscription metadata is stored. For either subscription type, the distribution agent can

technically be run on any computer. But, in standard practice—and by default—the distribution agent for push subscriptions typically runs on the Distributor. For pull subscriptions, the distribution agent typically runs on the Subscriber.

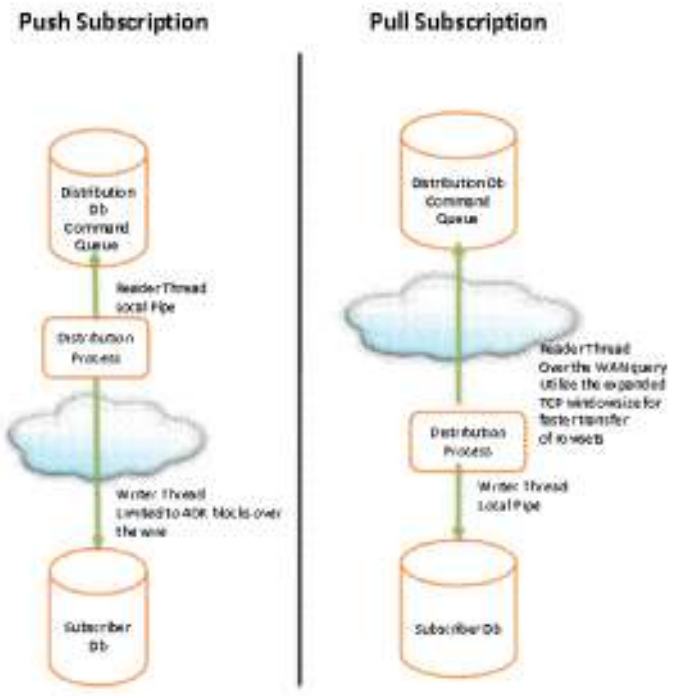

Figure 4: Differentiation of push and pull replication

In a local data center scenario, push subscriptions are typically preferred due to their manageability. Pull subscriptions are preferred over push subscriptions when it is important to offload work from the Distributor; for example, there might be a large number of subscribers, or simply lower technical specifications on the distribution server itself.

3.2. Distribution Agent Architecture

The following provides a detailed description of how the distribution agent works. This concept is further illustrated in Figure 5. At a high level, the distribution agent consists of the following:

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.

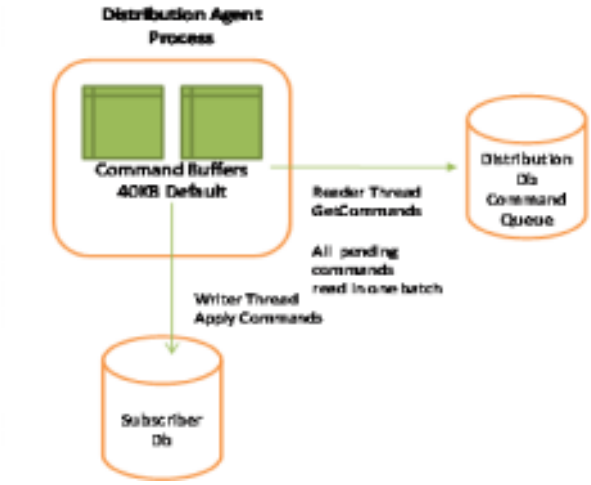

Figure 5: Distribution agent architecture

Figure 1, which illustrates the replication topology used for MSCOM Ops testing, shows that the Publisher and Distributor reside in the same geographical location and the Subscriber is set up across the WAN. In such a topology, using push subscription, the writer thread traffic will be sent over the WAN. In the case of a pull subscription, the reader thread traffic is sent across the WAN.

3.3 How Expanded Window Size Contributes to Faster Replication Performance

Table 2 illustrates that both push and pull subscriptions benefit from significant performance gains on Windows Server 2008 compared to Windows Server 2003. Further, the test results demonstrate that pull subscriptions perform much better than push subscriptions in a WAN scenario. The following section describes the key enhancement to the TCP/IP stack that enables this performance gain and explains why throughput performance varies with subscription types.

Autotuning of the Receive Window Size

One of the most significant improvements to Windows Server 2008 is the autotuning of the receive window size, designed for high-latency environments. In previous versions of the operating system, the maximum window size was limited by the 16-bit Window field in the TCP header amounting to a maximum window size of 64 KB. In Windows Server 2008, by combining the Window field together with a Scale Factor field of the TCP header, the window can be scaled or tuned, up to 16 megabytes (MB) in size. The impact of this improvement to the Windows Server networking stack is explored further in Scenario 1.

Scenario 1: *If a user relying on a previous version of the networking stack were to send 128 KB data over a network that has 100 ms roundtrip latency (50 ms each way), with the previous window of 64 KB, at any given time there could be no more than 64 KB of data outstanding in flight. This means that regardless of the actual network bandwidth available, the connection throughput was 64 KB over 100 ms latency, equal to 5.24 Mbps; and the data transfer would take about 200 ms. In comparison, by taking advantage of the expanded maximum window size in Windows Server 2008 (default maximum value 16 MB), a user can send 128 KB data without incurring an additional roundtrip and with much greater overall throughput.*

3.4 Examining Replication Performance by Subscription Type

The following describes how data is transferred for push and pull subscriptions, and the impact on network usage.

Push Subscription

In a push subscription scenario, the distribution agent runs on the Distributor, sending data to a Subscriber where the link has 100 ms latency. In this model, the Distributor relies on two buffers to pass data back and forth. If both of these buffers become full at any time, the system is unable to read additional data and must wait for the commands to be executed on the Subscriber before a buffer becomes available again.

As soon as the writer thread is full, the contents of the buffer are sent through the network to be executed on the Subscriber. This is executed through RPC calls over a tabular data stream (TDS). The problem is that after one batch of replication commands is sent over the network, the agent will wait for a TDS response packet to come back before anymore data is sent. This tends to produce "bursty" network usage because the batch size is smaller than the potential sliding window size. An example of this is shown in Figure 6, which illustrates a simulated workload and 300 ms roundtrip latency.

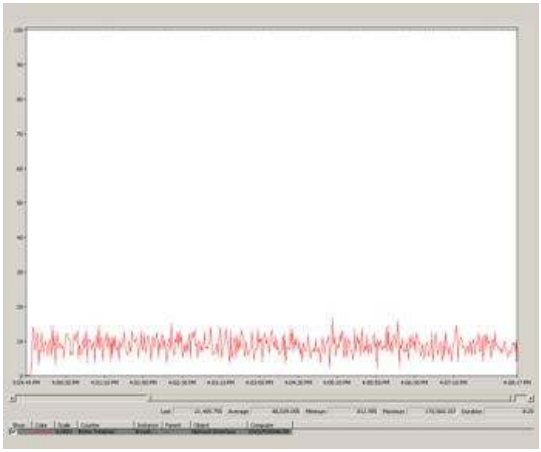

Figure 6: Example of network usage in a push subscription scenario

Pull Subscription

In a pull subscription scenario, the distribution agent is running on the Subscriber. This means that the data sent across the network consists of the reads from the distribution database rather than the writes to the Subscriber. Because applying the commands to the subscription database is quick relative to the network latency, the buffer from the writer thread can quickly be emptied and passed back for more data to be read. Because of the expanded window size in Windows Server 2008, users can send data across a WAN much faster than previously possible, as shown in Figure 7.

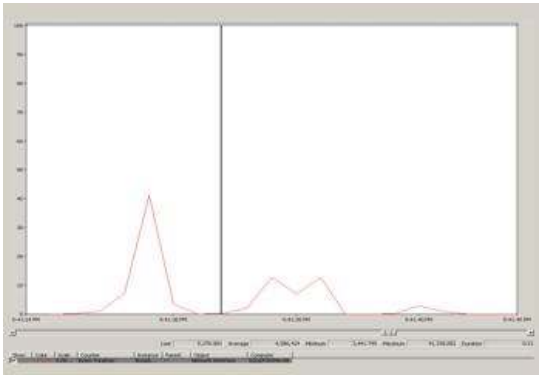

Figure 7: Example of network usage in a pull subscription scenario

Note: The spikes in the graph relate to the slowdown in the write performance that causes the buffer to be held by the writer thread, which prevents more reads from the network.

Appendix B: System Configuration

Below are the hardware and software used for the results shown in Table 2

Hardware:

- Two 2-socket servers with two Intel Quad-Core Xeon E5410 2.33GHz/12MB L2 Cache processors; 8 GB of RAM. There was one server in each data center.
- 546 GB of total hard disk capacity, 180 GB database storage partitioned between two operating systems (Windows Server 2008 and Windows Server 2003) and two versions of database software (SQL Server 2008 and SQL Server 2003) per operating system.

Software:

Operating systems:

- Windows Server 2003 (Build 3790: Service Pack 2)
- Windows Server 2008 RTM (Build 6.0.6001.18000) Enterprise

SQL Server"

- SQL Server 2005 (9.00.3042.00 [x64]) Enterprise
- SQL Server 2008 CTP6 (10.0.1300.13 [x64]) Developer

For more information:

Microsoft Teams and Groups

http://www.microsoft.com/sqlserver/ [http://www.microsoft.com/sqlserver/default.aspx] : SQL Server Web site

http://technet.microsoft.com/en-us/sqlserver/ [http://technet.microsoft.com/en-us/sqlserver/default.aspx] : SQL Server TechCenter

http://msdn.microsoft.com/en-us/sqlserver/ [http://msdn.microsoft.com/en-us/sqlserver/default.aspx] : SQL Server DevCenter

http://www.microsoft.com/windowsserver/ [http://www.microsoft.com/windowsserver/default.aspx] : Windows Server

http://technet.microsoft.com/en-us/windowsserver/ [http://technet.microsoft.com/en-us/windowsserver/default.aspx] : Windows Server TechCenter

http://msdn.microsoft.com/en-us/windowsserver/ [http://msdn.microsoft.com/en-us/windowsserver/default.aspx] : Windows Server DevCenter

http://www.iis.net/ [http://www.iis.net/default.aspx] : Internet Information Services

http://blogs.msdn.com/sqlperf/ [http://blogs.msdn.com/sqlperf/] : SQL Server Performance Team Blog

http://blogs.technet.com/windowsserver/ [http://blogs.technet.com/windowsserver/default.aspx] : Windows Server Team Blog

http://blogs.technet.com/mscom/ [http://blogs.technet.com/mscom/] : Microsoft.com Operations Engineering Team Blog

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

- Are you rating it high due to having good examples, excellent screen shots, clear writing, or another reason?
- Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

Send feedback [mailto://microsoft.com:25/default.aspx?subject=White%20Paper%20Feedback:%20Geo-Replication%20Performance%20Gains%20with%20Microsoft%20SQL%20Server%202008%20Running%20on%20Windows%20Server%202008] .