

Microsoft SQL Server 2005 Database Mirroring

Applied Technology Guide

Abstract

This document reviews the features and usage of SQL Server 2005, Database Mirroring.

May 2007

Copyright © 2007 EMC Corporation. All rights reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” EMC CORPORATION MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com

All other trademarks used herein are the property of their respective owners.

Part Number H2649

Table Of Contents

Applied Technology.....	0
Database mirroring.....	4
Overview	4
Roles	4
Partner	4
Principal	4
Mirror.....	4
Witness	4
Operating modes	5
Overview	5
Other terms	5
Send Queue.....	5
Redo Queue.....	5
High Protection	5
High Availability.....	5
High Performance	6
Summary of operating modes	7
How to start a mirroring session	7
Backup / Restore	7
Creating end points	7
Synchronizing the Mirror	8
Adjustments.....	8
Considerations.....	8
Client considerations.....	8
Using the correct MDAC	8
Connection strings	9
Failover coding best practices	9
Special considerations	9
Mirroring and clustering	9
Mirror database usage	9
Monitoring database mirroring	10
Important points.....	10
References	12

Database mirroring

Database Mirroring is the latest addition to the SQL Server availability options. It provides many features that previous options such as Transactional Replication, Clustering, and Log shipping did not provide. However, it is not necessarily the best alternative in all situations and many times the best alternative is to mix more than one HA (High Availability) option, such as Clustering and Database Mirroring. For example, clustering could be used to handle local failover, while Database Mirroring is used to maintain a geographically separate and “hot” copy of a database.

Overview

Database Mirroring works by maintaining 2 copies of a single database. One copy (called the Principal) is usable just like any other database and the second copy (called the Mirror and residing on a different SQL Server Instance) is in a state where it is constantly receiving changes from the Principal and applying them locally.

Roles

There are 2 roles that can be defined for a SQL Server for Database Mirroring. They are Partner and Witness.

Partner

The partner role defines that a server will be participating as either the Principal or Mirror for the specified Mirroring Session. There is no role that can be defined that specifies that a given machine will be the Principal or the Mirror for the session. The determined role of the partner as Principal or Mirror is based on other factors that will be discussed later.

Principal

The Principal role may be thought of as the source in the mirrored pair. In a Database Mirroring session the Principal Server is the instance of SQL Server that is hosting the Principal Database.

Mirror

The Mirror role can be thought of as the destination in the mirrored pair. During a Database Mirroring session the Principal sends copies of transactions to the Mirror (Async or Sync) to be applied there. This results in the Mirror maintaining an exact copy of the Principal Database that is transactionally consistent. Additionally, since the Mirror is applying the same transactions as the Principal, the cache on the Mirror ends up being a close representation of the Principal’s cache, along the lines of a “warm” cache. Also, Database Mirroring sends read Metadata along with the transactions that tell the Mirror what data to cache that is related to reads, thereby making the “warm” cache on the Mirror “hot”. So, upon failover, there is not a sudden drop in performance while the cache is warmed.

Witness

The Witness Defined role is used by Database Mirroring to facilitate Automatic failover by helping to establish a quorum. Whichever partner can still see the Witness after a failure has occurred (usually a network issue in this case) will take on the role of Principal and the isolated server would cease to serve the database. This will be discussed in more detail later.

Operating modes

Overview

There are 3 primary operating modes for SQL Server Database Mirroring, High Availability, High Protection, and High Performance.

Other terms

Before we can begin reviewing the operating modes, there are a few terms that need to be understood.

Send Queue

The Send Queue shows how many kB of SQL Log data is pending to be sent from the Principal to the Mirror. The Send Queue is integrated with the database's transaction log therefore, no additional space is required. Effectively, there is a marker on each transaction in the transaction log that tells whether or not that transaction has been sent to the Mirror. Those transactions that are marked as not having been sent are what is considered to be the Send Queue. The size of the Send Queue is indicative of the "lag" from the Principal to the Mirror and is used to determine the Recovery Point.

Redo Queue

The Redo Queue is the set of transactions that have been sent to the Mirror by the Principal and hardened to the Mirror's transaction log, but have not actually modified the data pages yet. After a transaction is hardened to the Mirror's log, it is then considered to be in the Redo Queue. The Mirror is in a constant state of recovery (the Redo phase of recovery) and is constantly rolling forward any transactions that end up in the Redo Queue. Once the transaction is rolled forward, it is removed from the queue. The size Redo Queue is indicative of how long it will take for the database to be available once a failover has occurred and is used to determine the Recovery Time.

High Protection

A database mirroring session is in High Protection mode if its Transaction Safety is set to FULL. This places the session into Synchronous mode, which means that a transaction is not complete until it has been completed at both the Principal and the Mirror. In High Protection mode there is no Witness server participating in the Mirroring session, therefore there is only manual failover available. When a transaction is submitted to the Principal it is simultaneously sent to the Mirror and it is not considered committed until the Mirror acknowledges that it has been written to the transaction log on the Mirror. Then the Principal informs the client that the transaction was committed.

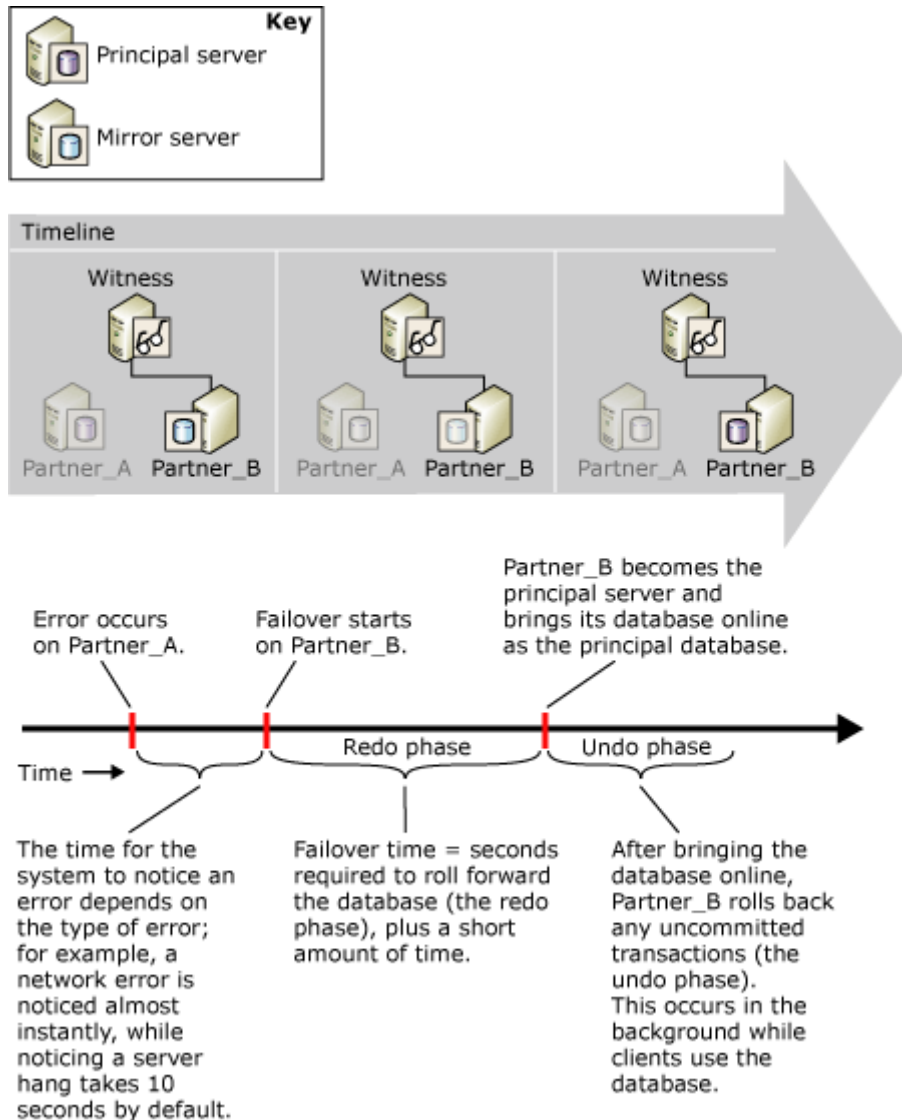
When in High Protection mode, the Mirror always has all transactions recorded in its transaction log, but it is not guaranteed that the data pages of the Mirror have been altered. When the Principal is under heavy load, it is possible and even likely that the Mirror will begin to build up a Redo Queue, which can increase the time for a failover to occur.

High Availability

High Availability is the same as High Protection, except for adding a Witness server. Adding the Witness server gives the Mirroring session the ability to perform automatic failover, because of the creation of a quorum. Without the Witness server, if a communications failure between the Principal and Mirror were to occur, it is possible that they could both become the Principal and both could end up receiving different transactions, thus causing a condition known as split-brain syndrome. This would cause the databases to diverge and remove the possibility reestablishing the mirror without substantial manual effort and possible data loss.

In High Availability mode, if the Principal fails, the Mirror and Witness establish a quorum and promote the Mirror to the Principal role. If there were a network issue near the Principal, then the Principal (no longer able to see the Witness and Mirror) would take itself off-line to prevent split-brain. Effectively, whichever Partner can still see the Witness will be the Principal. This does not mean the Witness is a single point of failure. Should the Witness fail, the remaining Partners are still a quorum and the Principal can continue to serve the database without concern. In the white paper titled, "Database Mirroring in SQL Server 2005" referenced at the end of this document, there is an exhaustive review of the HA scenarios and is recommended for further investigation into this topic.

The following diagram is from SQL Server 2005 Books Online and demonstrates the timeline for a Database Mirroring session failover.



High Performance

When in High Performance mode, the transactions are sent to the Mirror asynchronously, meaning that the Principal does not wait for the transaction to be received by the Mirror before considering it committed. As soon as the transaction is hardened to the Principal's log, it is considered committed. Once the transaction is hardened to the log, it is also in the Send Queue and will be sent to the Mirror as resources allow.

Summary of operating modes

To summarize, the following table shows the operating modes along with other parameters associated with each.

Mode	Transaction Safety	Witness	Failover	Performance		
				Low Latency	High Latency	When Bandwidth is overrun or high latency exists
High Availability	FULL (Sync)	Yes	Automatic	Good	Poor	Performance gets worse
High Protection	FULL (Sync)	No	Manual	Good	Poor	Performance gets worse
High Performance	OFF (Async)	Not Advised	Manual with Possible Data Loss	Better	Better	Performance mostly unaffected, but Send Queue deepens

How to start a mirroring session

To start a Mirroring session there are several steps that must be completed.

Backup / Restore

To prepare for the Database Mirroring session, the database that will end up being the Mirror needs to be brought to a point that is transactionally close to the Principal. It does not have to be to the exact point, but the closer it is, the quicker the mirror will get to the synchronized state. For a database that is relatively small and has a maintenance window during which a full backup can be gotten to the Mirror server and restored, this step is rather straightforward. All that would need to be done would be for the Principal database to have a full backup done, shipped to the mirror, and then it would be restored on the mirror with the NoRecovery option. This will leave the database that is to become the Mirror in the recovery state that it needs to be in.

If the database is larger and/or does not have a maintenance window then other techniques may need to be used. For example, a full database backup to an external hard drive(s) shipped to the mirror location, and log backups being transferred, until the Mirror is ready. The database could be restored to the Mirror instance, and then the transaction logs applied, in order, until the Mirror database was within one log backup of the Principal. Then the mirroring session could be started and the Mirror would begin synchronizing with the Principal. The requirement is that the transactions that are missing from the Mirror must still be in the Principal's transaction log, otherwise there will be a gap and the initialization of the Mirroring session will fail.

Either way, before, during, or after the restore, the communications paths for the Partners and Witness will need to be defined and they are called endpoints.

Creating end points

Endpoints are used to define how the servers will communicate with each other. Database Mirroring endpoints use TCP to send and receive messages between server instances in database mirroring sessions. If there are an existing set of Database Mirroring endpoints that will work for the new Database Mirroring session, then they may be used, otherwise new endpoints will need to be created. A single endpoint may be used for multiple mirroring sessions. The endpoint defines the TCP port that will be used, the type of authentication, the type of encryption (None, RC4, or AES) for communications, and the role of the server. An endpoint is created with the CREATE ENDPOINT command in TSQL or it can be created through the GUI as part of establishing a Database Mirroring session.

Additionally, the service account that the SQL Server instance is running under on each server (Principal, Mirror, and Witness) will need to be granted permission to connect to the endpoints on the other server. If all of the Instances are running under the same domain account, then no privilege granting is necessary. Also, if the SQL Servers are not in a single domain or not in trusted domains and running under domain accounts, the Certificate Authentication will need to be used. Setting up a Certificate and using Certificate Authentication for endpoint connections and Database Mirroring is covered in SQL Server Books Online.

Synchronizing the Mirror

To initialize the Database Mirroring session, the ALTER DATABASE command is used. First the command is run on the soon to be Mirror Server to alter the database in recovery mode and tell it who its Partner will be. When specifying the network name of the Partner, the fully qualified TCP address must be used in the form of “TCP://fully_qualified_domain_name:port”. This is because there is no assumed port for Database Mirroring; therefore the port must always be specified. Additionally, although the documentation says that a fully qualified domain address must be used; internal testing has shown that an IP address followed by a port will also work, in the form of “TCP://###.###.###.###:port”.

Once the ALTER DATABASE command has been issued on the Mirror, the ALTER DATABASE command is issued on the Principal. Once the ALTER DATABASE command is issued on the Principal, a High Protection Database Mirroring session is started with a status of Synchronizing. If the intent is to establish a High Protection mode session, then the necessary settings are complete. If the mode is intended to be either High Availability or High Performance, then other commands will need to be issued.

Adjustments

To adjust the Database Mirroring session from High Protection to High Availability a Witness must be added. Again, this is done by issuing the ALTER DATABASE command. Once a Witness is added the mode automatically changes from High Protection to High Availability.

To adjust the Database Mirroring session from High Protection to High Performance, the transaction Safety must be set to OFF. Again, this is done by issuing the ALTER DATABASE command. Once the Safety is set to OFF the mode automatically changes from High Protection to High Performance.

To adjust the Database Mirroring session from High Availability to High Performance, the transaction Safety must be set to OFF. However, it is recommended that the Witness be removed first, thereby returning the session to High Protection, before changing the transaction Safety. Again, this is done by issuing the ALTER DATABASE command. Once the Safety is set to OFF the mode automatically changes from High Availability or High Protection to High Performance.

Considerations

Client considerations

A database mirroring failover is not completely invisible to clients. A client must be using the correct version of the MDAC, should specify the proper connection string, and should be written so as to smoothly handle the failover.

Using the correct MDAC

An MDAC that supports the “Failover Partner” connection string parameter must be used. Any version from MDAC 2.5 or above should be capable of supporting this.

Connection strings

When a client connects to a database, it specifies information about which server instance to connect to, which database to use, credentials, etc, in a connection string. One of the parameters of a connection string is called the “Failover Partner” and can be used to specify which database server instance to connect to, if the primary choice is not responding or goes off-line.

Once the client connects to the Principal, the Principal will inform the client of its “Failover Partner”, in case it was not specified or has changed. This information is then cached for the connection’s duration. However, it is regarded as a best practice to provide the Failover Partner in the connections string, just in case the Principal is off-line when the initial connection attempt is made.

Failover coding best practices

Many developers do not consider failover possibilities when developing their client and end up creating an application that could lose data and cause user heartache, during a failover. Applications should be written to handle the eventuality that a database server may go off-line for a period of time and then come back on-line, even during in-flight transactions. To handle this, when a database goes off-line and an error is returned (which would occur for most type of data protection and availability solutions, including clustering and Database Mirroring), the application should trap the error and realize that any in-flight transactions (transactions that were started, but not yet committed) will need to be reattempted from their beginning. Therefore, the client should retain the data need for the transactions and begin a series of reconnect attempts. Each reconnect attempt should wait for a period of time, and then try to reconnect. When the failover completes and the database is again online, the reconnect attempt will succeed. Then the application will need to restart any transactions that were in-flight and continue its processing. Applications that use this method should experience no loss of any data, except for under the rarest of unusual circumstances.

Special considerations

Mirroring and clustering

When Mirroring a database in High Availability mode and using a cluster as the Principal there are potential issues that could arise during a cluster failover. During a cluster failover the Witness and Mirror will notice that the Principal has failed and will likely promote the Mirror to Principal, before the cluster failover can complete. There is a timeout setting that can increase the amount of time that the Witness and Mirror will wait before considering failover, but it has not been verified whether increasing the timeout beyond its default of 10 seconds will actually help to mitigate this issue. Therefore, it is recommended that when mirroring and using a cluster as the Principal, that High Availability mode not be used. The other 2 modes, High Protection and High Performance are recommended in this scenario.

Mirror database usage

Since the Mirror database is in a constant state of recovery, it cannot be accessed at all. Some users have expressed interest in using the Mirror Database for non-real-time operations, such as batch reporting. This can be done by using SQL Server’s Database Snapshot capabilities. When a snapshot of a database is taken, for most purposes it acts like an autonomous read-only database from its basis database. Therefore, the snapshot is brought out of the constant recovery mode that the Mirror database is constantly in and is thereafter able to be queried.

Some caveats exist related to the database snapshot, such as;

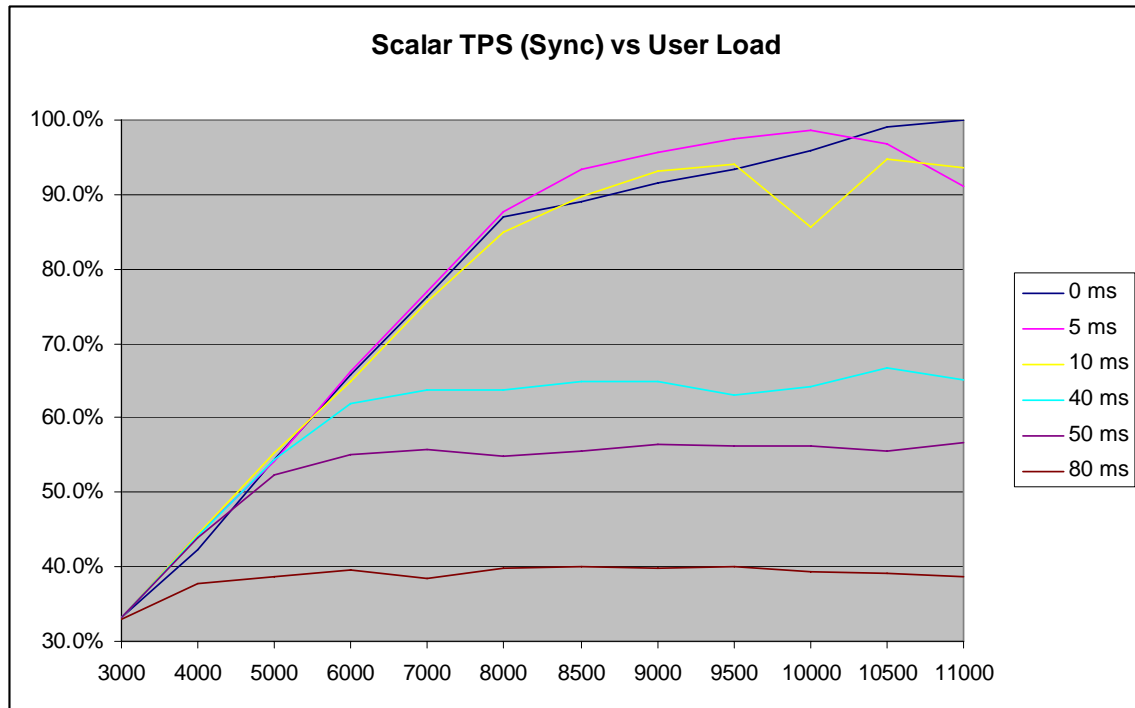
- The snapshot will exist as of the time it was created and will not be updated in any way by the Database Mirroring Session.
- If an additional or up to date snapshot is needed, then a new snapshot must be taken.
- Creating a database snapshot of the Mirror will add load to the Mirror server and could possibly slow the application of the Redo Queue, thereby increasing the recovery time.

Monitoring database mirroring

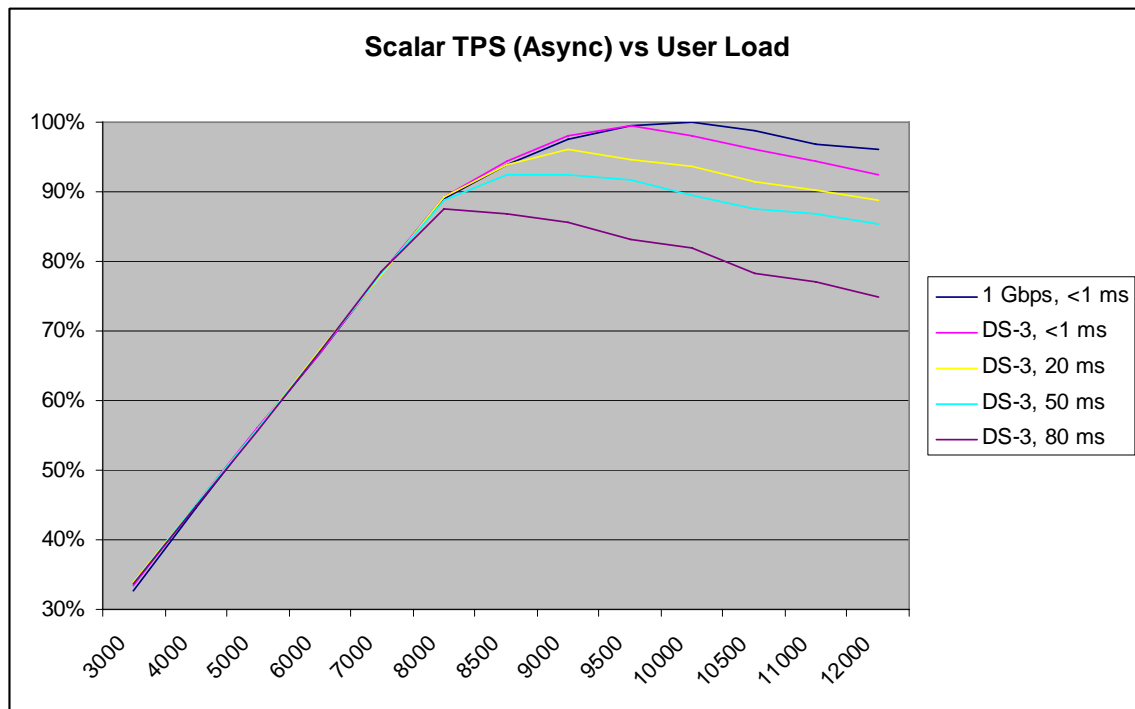
There are several methods of monitoring Database Mirroring including using System Monitor, Performance logs and Alerts, Dynamic Management Views, and SQL Profiler traces. However, these are all well documented in SQL Server 2005 Books Online, as well as the Whitepapers referenced at the end of this document.

Important points

- You cannot mirror the system databases, Master, MSDB, Tempdb, or model.
- A Mirroring session is for a single database and will only make a single Mirror.
- The database that is to be mirrored must be using the Full Recovery Model.
- Although multiple databases can be mirrored from Instance A to Instance B, there is no guarantee of inter-database consistency. There is also no guarantee that Instance A will be the Principal for all of the databases. For example, if Databases 1, 2, 3, & 4 on Instance A are mirrored (High Availability Mode) to Instance B and an error occurred to only Database 3, then only Database 3 would failover to Instance B. Therefore, it would end up with Databases 1, 2, and 4 being served from Instance A and Database 3 being served from Instance B.
- There may be a small Send Queue in Synchronous mode as these are the pending transactions.
- The RPO is determined by the Send Queue (especially in Async mode) and the RTO is determined by the Redo Queue (in either mode).
- The mirror has a substantially more aggressive writer than is normally used by SQL Server, therefore in systems where pages could be altered several times before being flushed by the lazy writer, the write IO may be substantially heavier on the Mirror, than it is on the Principal. In TPCC tests (R:W ratio of about 3:2), it was found that the Mirror had a range of 1.8 to 7 times the amount of write activity as did the Principal, with the vast majority of samples falling into the 3 to 4 times multiplier. Therefore, in a write heavy system, it is possible that the Mirror may need greater disk resources than the Principal, to prevent the buildup of a very large Redo Queue.
- The method used for Synchronous mirroring can be thought of as a two-phase commit, however it is not a true two-phase commit as there are substantial variations.
- Although Async has much better performance than the Sync modes, there is still a penalty as network latency increases. The following graph shows a scalar representation of TPS vs User Load vs Latency with Database Mirroring in Synchronous mode.



It is expected that network latency would lower performance as the latency increased. However, the same comparison is shown below, but this time it is with the Database Mirroring session in Asynchronous mode.



Again there is decrease in scalability levels as the network latency increases. The scalability levels are much higher and the decrease is not as drastic as Synchronous mode, but it does have an effect.

- Microsoft has made a general recommendation (due to resource constraints) that no more than 10 databases be mirrored per server.
- To determine the probable network bandwidth requirement for a Database Mirroring session, before Mirroring is implemented, monitor the Log Bytes Flushed/sec performance counter. Other documents have said that the network throughput for the mirroring session should be anywhere from 150% of the 95th percentile, 100% of the maximum, 300% of the average, and others. The reality is going to depend upon how high the peaks are above average, what the network latency is like, how long the peaks last, and other factors.
- All non-database information must be copied and synchronized separately and possibly manually. Some objects that are not maintained by DB Mirroring that may be required on the Mirror in the event of a failover are;
 - Logins (SSIS has a transfer logins task that may be useful here)
 - SQL agent jobs and alerts, and the jobs on the Mirror will remain disabled or a logic is added to them to ensure that they do not run in a non-failover condition.
 - SSIS packages, which may have other interdependencies on installed components, etc.
 - Linked server definitions
 - Backup devices
 - Maintenance plans
 - DTC settings, if DTC is used
 - Dependencies on system database objects, such as reviewing a job log in msdb
- There are issues with using Full Text Catalogs with Database Mirroring, since updates to the FTC are not logged.

References

The following resources were used in writing this guide and should be read for additional information:

- SQL Server 2005 Books Online
- White papers
 - “Database Mirroring in SQL Server 2005” by Ron Talmage
 - “Database Mirroring Best Practices and Performance Considerations” by Sanjay Mishra