

Disaster Recovery & Business Continuity for SQL Servers Through a Standby Approach

Introduction

Businesses of all sizes must take backup and disaster recovery seriously. As companies grow and data becomes more and more prevalent, users of all types, either internal or external, require access to all this information. This is essential at a time when even the smallest business may find operations impossible with any single point of failure. The ability to recover critical data quickly after a disaster is a fundamental requirement of economic viability and business continuity.

With the launch of Microsoft SQL Server 2005, Microsoft SQL Servers are becoming increasingly popular for use in mission-critical environments. With such important dependencies on software solutions, recovering from failures is crucial for business continuity. To initiate movement towards redundancy and failure protection, Microsoft offers SQL Server Log Shipping (SQL Server 2000 and 2005) and SQL Server Database Mirroring (SQL Server 2005) in its core products. In addition to Microsoft, third party vendors, including Microsoft Gold Certified Partners such as Sonasoftware, provide enhanced software solutions that simplify disaster recovery scenarios through easy to use Management Console and automating most of the manual steps. These components are focused on minimizing downtime while maximizing data retention in the event of catastrophic failure.

Standby Solutions invariably provide the same functionality—protection against all types of failures. The only way to truly prevent a gainst any failures is to isolate each location and replicate it to another physical geographically distinct site. This involves replicating SQL Server databases across Local Area Networks, or in most cases Wide Area Networks, to remote Standby servers. Ultimately, this provides failover options for organizations wary of hardware, software, storage and network connectivity failures.

High Availability Solutions for SQL Servers

Failover Clustering

Microsoft Clustering enables users to prevent against hardware failures by stringing redundant hardware, called nodes, together through a central cluster manager that coordinates load balancing and data activity. Typically, nodes share common storage space and have the capability of picking up load off of a node that goes down due to hardware or software malfunction. There are two types of cluster environments—active/active and active/passive. In the former, every node in the environment is live and capable of processing requests. When one active node goes down, the others simply process more requests as the load is evenly dispersed across the remaining nodes. In the latter, there is a single active node that processes all incoming requests. Upon hardware or software failure in the active node, the passive node is immediately and automatically brought up by the cluster manager to take over normal function of processing data requests. In this way, hardware exposure is mitigated through physical hardware redundancy.

Microsoft SQL Server supports both active-active and active-passive cluster environments. SQL Server Clustering provides high availability by protecting against a node failure. However, it does not prevent against storage failures. Given the size of typical cluster environments, multiple hard disks are used to build large storage arrays. In Network and System Administration, when large numbers of any one device is used, failure is expected. When a hard disk fails, application disruption is unavoidable as all the nodes in the cluster could be using that one particular disk as shared storage which contains all files, including SQL Server database files. As protection against this particular failure, RAID configurations are common. However, from a performance standpoint, this significantly slows down I/O in the subsystems due to writing the data to multiple disks at the same time. Administrators have to balance such performance degradation and understand that this particular implementation has limitations. Again RAID option is to protect against any hard disk failure but it cannot prevent site disasters.

In direct contrast to this storage dependency, using a Standby solution prevents against hardware, software and storage failures. Standby servers or databases are normally installed on unique, usually geographically independent, SQL Servers which serve as a barrier to failures of any type.

SQL Server Clustering environments are more cost-intensive compared to the Standby option. The primary reason for this is the high hardware and software requirements. Clustering requires Windows NT Enterprise Edition, Windows 2000 Advanced Server or Windows 2003 Enterprise Edition and SQL Server Enterprise Edition. Additionally, it only supports hardware listed on the Microsoft Hardware Compatibility list. On the other hand, a Standby server does not have any special hardware requirements and is simply a software solution to meet disaster recovery needs. An additional cost, LAN connectivity is required between SQL Server cluster nodes to send and receive what is called a heartbeat signal, among other communications. This signal is used by each node to determine if other nodes are still available. In case any node is not available then the remaining nodes take over. With Standby, LAN or WAN network connectivity will work to replicate SQL Server databases. The speed of this process is directly related to the size of the databases and network bandwidth. High speed compression of data will speed this process up significantly by limiting network transmissions.

SQL Server Native Backups/Restores

A simple and inexpensive solution to recovering from failure is to take backups of all your databases. Out-of-the-box Microsoft SQL Server operation allows for Native SQL Server Backups that can take full, differential or transaction log backups of a database. These backups are very primitive in nature as they simply copy data from the existing database files in uncompressed .bak formats. As a consequence, performing these backups degrades performance and response time of SQL Server when the complete backups are active.

With these native .bak database backups, users can recover from disasters by performing Native SQL Server Restores. SQL Server backup information is automatically stored in a system database called msdb on the same SQL Server Instance as where the backups are being performed. This is very handy when trying to perform recovery operations of existing databases as all the backups that have been taken appear in the Native Restore screen. Unfortunately, if entries in the msdb table are purged due to size constraints on the database or if the msdb table is somehow corrupted, native recovery operations become a lot more troublesome for Point in Time Recovery. Point in Time recovery would allow not only complete backup files to be used for restores, but also the more frequently taken Differentials and Transactionals. Users would have to either write scripts or manually select each file and restore them in the proper order, especially in the case of Transactionals, so as to preserve database integrity. Furthermore, restoring databases from one SQL Server Instance to another SQL Server Instance becomes problematic because the local msdb database will not have backup information available to restore the database from. Additionally, the logon account under which SQL Server Agent runs under may not have sufficient privileges to read from the backup files that reside on another folder or network share.

To protect against msdb corruption and failures, users might consider replicating the msdb database to another instance of SQL Server residing on another physical server. Unfortunately, this is not very useful, since the data stored in the msdb database is SQL Server Instance dependent. That is, the msdb database from one SQL Server Instance (A) does not contain the relevant information for another SQL Server Instance (B).

Still, by replicating, efficient and cost-saving users may find some amount of marginal benefit. Suppose A's msdb database is replicated to B with a different name than 'msdb' lest B's msdb database gets overwritten. In the event that A's msdb database gets corrupted, users have to replicate the database back from B to A. Users stop replicating the msdb database to B and back up the database from B and then restore it onto A. All the while, users need to be careful. This reverse replication needs to be done quickly enough so that the corruption in A does not percolate over to B. As a result, this process can be very time-consuming and error-prone.

Another disadvantage to SQL Server Native Restore procedures involves high downtime. In cases of disastrous failure, time is a crucial factor. Unfortunately, significant delays are introduced when trying to find the correct backup files and locations from which to restore databases. This may actually take more time than the time it takes to do the restore operation. Also one would not know of any corruptions in the backup files themselves, until attempts to restore from them are made and failures are received. Typically, when these restores need to be performed, the inability to restore is something most organizations cannot tolerate.

Additionally, restoring from network shared folders can introduce significant delays due to network bandwidth. Without compression, the backup files are roughly the same size as the database. When trying to restore, all this data needs to be pulled through the network pipe, which can be a severe bottleneck in cases of WAN connectivity.

Automated Standby Solution

A Standby Server is a server where the data from the primary databases is restored periodically using scripts, Microsoft Log Shipping or through a third party software. It acts as a hot standby which can be promoted to a primary role if the primary server goes down so that business operations can continue to function. An automated standby solution bridges the gap between a highly expensive Failover Clustering solution and time-consuming Native SQL Server Backups and Restores. An automated standby solution provides a high-availability solution, protecting data from hardware and software failures as well as from human errors. This procedure completely automates log shipping practices and ensures data integrity, enabling the standby system to take over instantly in the event of primary system failure. The standby system can be located on-site or at a remote site for protection against natural and man-made disasters.

Although this capability can be achieved by using SQL Server's built-in log shipping feature, it requires the services of an experienced DBA, as well as a high degree of monitoring. Log shipping is required to perform fairly complex pre-configured tasks. Also switching the secondary server to the primary role after a disaster strikes is not a trivial task. In the case of a complete system crash, the log shipping configuration information may also be lost. It is extremely difficult to keep track of log files already restored, log files yet to be restored, and, then, apply the tail of the transaction.

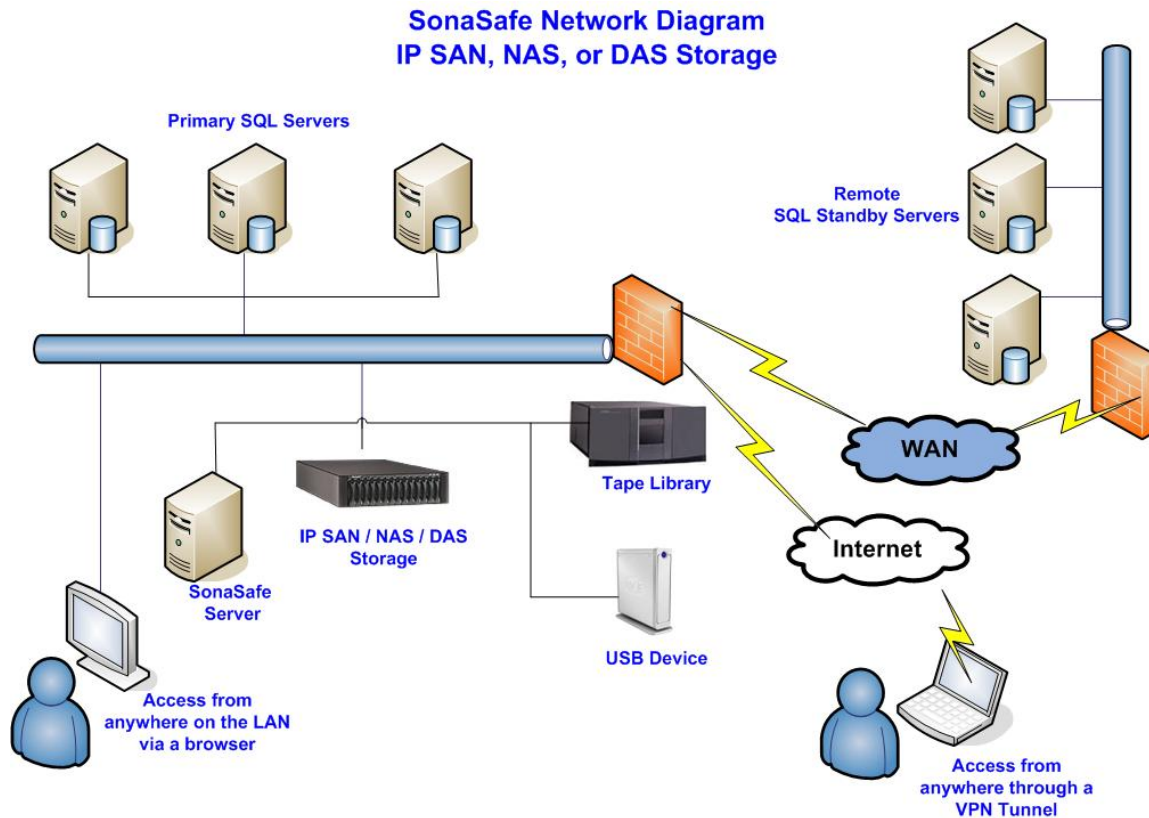
Sonasoftware Standby Solution

The Sonasoftware Standby Solution builds easy-to-manage standby plans that effectively replicate your data without interfering in normal business operations. This includes CPU utilization on the primary server as well in limiting network contention by reducing network traffic through on the fly, high speed compression. Because the Sonasoftware Standby Solution transfers only data that has changed since the last backup as well as uses high speed compression through the network to the standby server, data transmission can be handled by a low-bandwidth T1 or a low-cost DSL line.

The Sonasoftware Standby Solution automates all of the critical steps involved, making the process completely transparent to the user. The solution is driven by self-explanatory screen wizards, armed with highly powerful monitoring capabilities at different levels. When disaster strikes, role switching is just one click away, as a series of complex database operations are performed in the background to switch the destination server(s) to the primary role.

Sonasoftware's product, SonaSafe for SQL Server contains a unique architecture that not only creates easy to use backup tasks and schedules, but allows for efficient and simple recovery options all while minimizing chances of data loss. Considered two-tier architecture, SonaSafe for SQL Server consists of an application and agent environment. The application is hosted by an auxiliary server that runs Microsoft SQL Server and houses the SonaSafe Recovery Catalog. The application server also hosts

the network share that stores all the backup files. The files are stored on this network share and not on any particular target server so as to prevent loss of backup files. If the target server goes down, users would like to still access their backup files in order to rebuild the target server with as little downtime as possible.



Internally, Sonasoftware maintains a Recovery Catalog that manages task scheduling as well as backup and restore information. This design provides a centralized store for all the information related to Disaster Recovery for SQL Server. Resting on a SQL Server database, this Recovery Catalog can itself be replicated to protect from hardware failure on the Sonasoftware Application Server. Since the Recovery Catalog contains information for all SQL Server Backups and Restores, it is not SQL Server Instance dependent like msdb. Therefore, when replicating this database, it can be failed over at another SQL Server instance without minimal impact on the Sonasoftware Application.

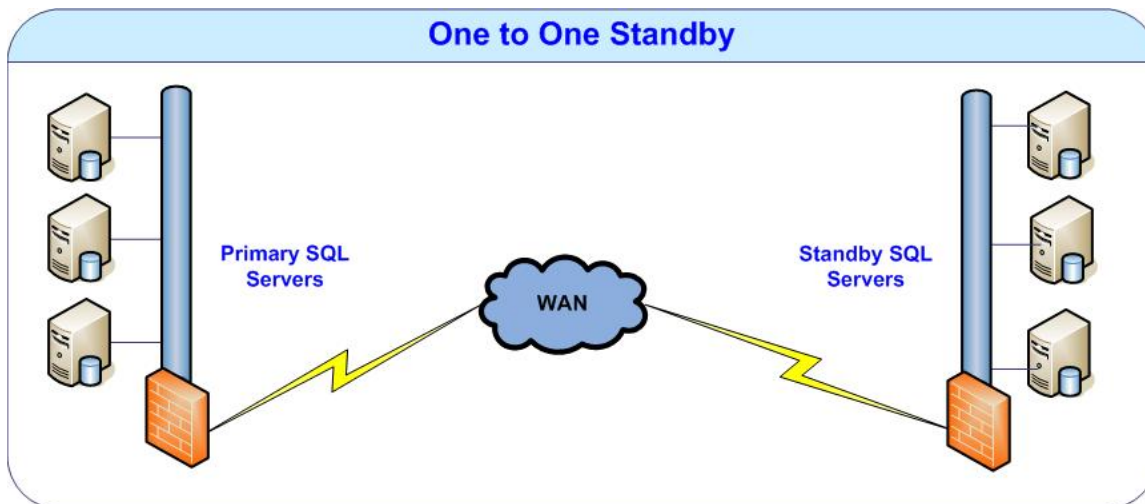
Agents are installed on each target server that runs Microsoft SQL Server. These agents are intelligent pull, not push, task managers that run as a Windows Service. They connect to the Recovery Catalog in order to pick up tasks, report task status and logging information and to maintain backup and restore file information. Agents are capable of performing multi-threaded backups with on the fly high speed and low CPU utilizing compression. This compressed data is then sent through the network and written to the shared folder, which decreases backup time by limiting network traffic.

Additionally, upon Restore operations, files need not be copied from the network share to a local drive as in Microsoft Log Shipping. Instead, compressed data is read across the network, uncompressed on the fly and simultaneously restored. This decreases storage costs as files need only be stored in one location. Additionally, with compression, multiple backup sets can be stored in lieu of a single uncompressed backup set. Decompression, like SonaSafe compression, is performed with minimal impact on CPU utilization.

Standby Scenarios

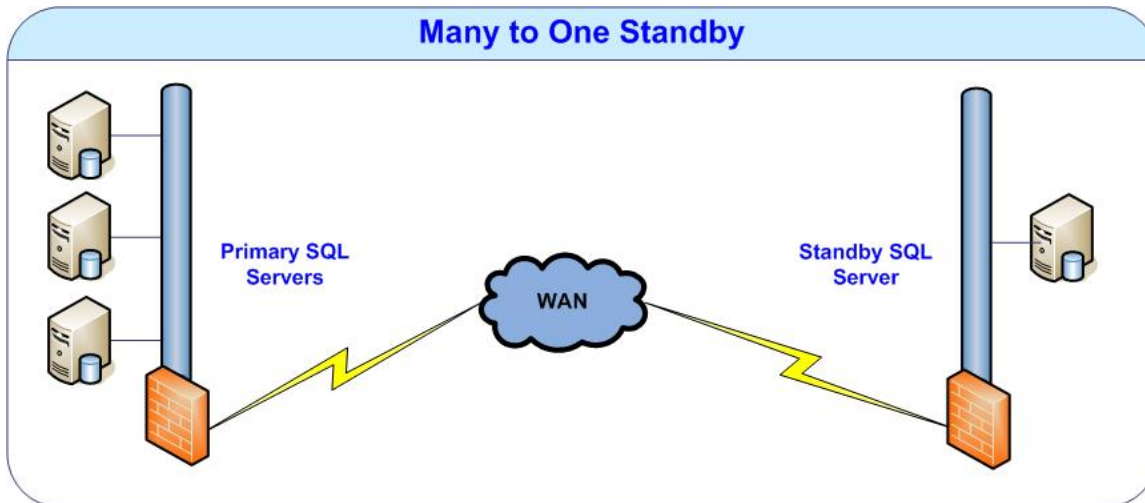
One-to-One Standby

In the one-to-one standby scenario, for each primary server there is a standby server at the remote site. For example, one of our customers has 25 servers in the San Francisco and 25 standby servers in Salt Lake City, Utah which is about 700 miles away. This customer has one server dedicated for each one of their end customer and needs to keep the data separate. Hence they decided to implement a one-to-one standby scenario. The standby updates are happening without any hitch every 15 minutes.



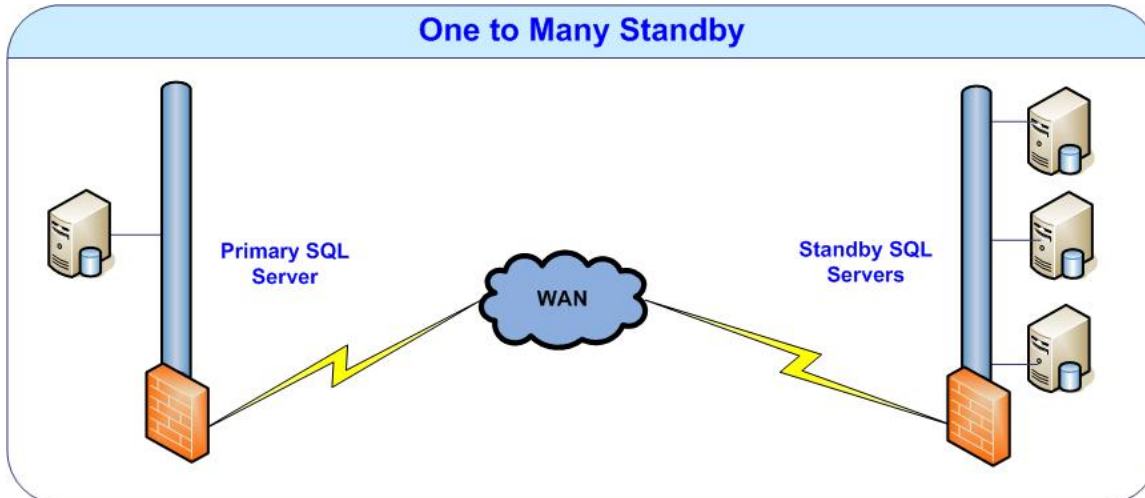
Many-to-One Standby

In the many to one standby scenario, multiple databases from different primary servers can be replicated to a single standby server at a remote site. By doing this one can reduce both hardware and software costs significantly. Instead of creating standby for all databases on all servers, one can select the key databases from different servers and create standby for them on a single server at the remote site.



One-to-Many Standby

One-to-many standby scenario allows one-to-many live replication of a single database, instance or entire SQL Server. For example one may want to create a local standby at the same location as primary server and a standby at a remote location. The local standby can be used for reporting purposes and the remote standby for disaster recovery purposes.



Conclusion

Standby servers are a cost-effective and viable way for businesses to maintain SLA and business continuity efficiently. While other solutions do exist, they are limited by either being error-prone or failing to protect against site failures. One can deploy standby servers using third-party software that utilize easy to use interfaces for a fraction of the cost associated with some alternatives as discussed above. Also, these standby servers can be functional for reporting or testing purposes.

About Sonasoftware[®]

Sonasoftware Corp. automates the disk-to-disk backup and recovery process for Microsoft Exchange, SQL and Windows Servers with its groundbreaking SonaSafe[®] Point-Click Recovery[®] solutions. Designed to simplify and eliminate human error in the backup and recovery process, SonaSafe solutions also centralize the management of multiple servers and provide a cost-effective turnkey disaster recovery strategy for companies of all sizes. *For more information, please visit www.sonasoftware.com.*

2150 Trade Zone Blvd, Suite 203 San Jose, California 95131
Phone: (408) 708-4000 Fax: (408) 946-5800
www.sonasoftware.com info@sonasoftware.com