Taming the Storage Dragon: The Adventures of HoTMaN

Shahram Ghandeharizadeh, Andrew Goodney, Chetan Sharma
Computer Science Department
University of Southern California
Los Angeles, California 90089

Chris Bissell, Felipe Cariño, Naveen Nannapaneni, Alex Wergeles, Aber Whitcomb
MySpace Inc.
407 North Maple Dr.
Beverly Hills, California 90210

April 8, 2009

Abstract

HoTMaN (HoT-standby MaNager) is a joint research and development project between MySpace and USC Database Laboratory to design and develop a tool to ensure a 24x7 up-time and ease administration of Terabytes of storage that sits underneath hundreds of database servers. The HoTMaN tool's innovation and uniqueness is that it can, with a few clicks, perform operational tasks that require hundreds of keyboard strokes by "trusted trained" experts. With HoTMaN, MySpace can within minutes migrate the relational database(s) of a failed server to a hot-standby. A process that could take over 1 hour and had a high potential for human error is now performed reliably. A database internal to HoTMaN captures all virtual disks, volume and file configurations associated with each SQL Server and candidate hot-standby servers where SQL server processing could be migrated. HoTMaN is deployed in production and its current operational benefits include: (i) enhanced availability of data, and (ii) planned maintenance and patching. In the future, HoTMaN can be extended to migrate relational databases when a server reaches an autonomic threshold.

1 Introduction

While the cost of buying[1] storage is inexpensive, enterprises are alarmed by the cost of managing storage. For popular destinations on the Internet, this cost includes the crushing financial consequences of down time, often estimated at thousands of dollars per minute [8]. To address this and other costs, enterprises centralize management of data using Storage Area Networks, SANs.

A SAN is an architecture to attach remote computer storage devices such as disk arrays to servers in such a way that the devices appear as locally attached to the operating system. It virtualizes available storage to present a virtual hard drive, a Logical Unit Number (LUN), to each server. This virtualization increases storage capacity utilization because multiple servers share the storage space on disk arrays. A SAN may implement sophisticated load-balancing techniques [4] to harness the maximum available disk

[1] At the time of this writing, the cost per Gigabyte of magnetic disk storage is less than 10 cents.

Term	Definition
Hot Standby	A powered on server that is idle and attached to one or more SAN switches, waiting to substitute for a failed server.
LUN	A Logical Unit Number, representing a virtual disk drive to a server.
SAN Frame	A unit of physical storage, typically a rack, consisting of controllers, cache, battery backup, and space for physical disk drives.
ConfigDB	Production Configuration Database describes the physical connections, and configuration of the servers and devices that make up a SAN deployment.

Table 1: Terms and their definitions.

bandwidth. The common applications of a SAN are those that require high-speed block level access to the data such as email servers and transaction processing systems.

At MySpace, SANs are used to store data pertaining to a few hundred applications. MySpace employs a three-tier architecture that consists of web servers, cache servers, and relational database servers. A custom distributed file system (DFS) stores multimedia images, music, and video data. At the time of this writing, MySpace uses 10 data centers to house and manage all their user data. Microsoft SQL server is used at the database layer and Berkeley DB [7] is used to implement the cache servers. All the user data is stored in the SQL servers with references to objects stored in DFS.

A database might be partitioned across hundreds of SQL servers using a range declustering strategy [2, 3, 5]. For example, one of the tables contains user profile information and this table is range partitioned using its user-id attribute with 10 million key values assigned to each database server.

The system administrators are faced with tremendous challenges not adequately addressed by any tools. They must administer LUNs, file system mount points for each LUN and their logical interdependencies (e.g., the mount point for the LUN corresponding to "X:\data\" is contained in the LUN corresponding to "X:\" which must be mounted first), and databases contained in different LUNs. These administrators must manage a disaster recovery scenario and maintain replicas of LUNs scattered across different data centers. They may maintain alternative mechanisms for different data sets that they process on a daily basis and ensure the proper operation of the whole infrastructure. Developing effective techniques to manage data is a fundamental challenge faced by many enterprises today [1], including MySpace.

Application and storage administrators employ hot-standby servers to minimize down-time and enhance availability of data. The argument for such servers is simple: The cost of a hot-standby is significantly lower than the financial costs of down-time. To maximize the value of this argument, the time to replace a failed server with a hot-standby must be as short as possible. An administrator may author scripts to embody the processes and procedures used to manage hot-standby servers. However, these scripts might be error prone and time consuming to use. HoTMaN is our adventure to demonstrate the feasibility of unifying these processes and procedures into a GUI. A key advantage of HoTMaN is its possible deployment for use by operators with a range of skills broader than the highly specialized database and storage administrators, freeing these trusted experts to perform other tasks.

The rest of this white paper is organized as follows. In Section 2, we present HoTMaN and its key design decisions. We conclude with brief conclusions and research directions in Section 3.

2 HoTMaN

The HoTMaN (HoT-standby MaNager) project was initiated in April 2008 as a collaborative effort between the USC Database Laboratory, and both the research department and the storage group at MySpace. The storage group manages several SAN frames providing thousands of LUNs to hundreds of SQL Servers. We started with the challenge of how to migrate LUNs from a failed server to a hot-standby both quickly and accurately. Accurately because the following key questions must be answered when a server fails:

1. Which nodes are candidate hot-standby servers for the failed server? Several SAN switches provide server connectivity to different SAN frames containing magnetic disks with data. Given a failed server, a candidate hot-standby must have physical connectivity to the frames that provide virtual disks to the failed server.

2. Which LUNs must be presented to the hot-standby server? This information cannot be obtained from the failed server because it is unavailable.

3. What is the appropriate file system mount point for each LUN presented to the hot-standby server? This information is important because the database residing on a LUN is dependent on the mount point of the LUN when it was first created. If the LUN is not attached to the correct mount point then its database may not function properly once attached to a SQL server instance.

4. What is the order to mount the different file systems? File system mount points corresponding to different LUNs may depend on one another. For example, a LUN mounted at "X:\" might consist of a data volume "X:\data\" and a log volume "X:\log\". The LUN corresponding to "X:\" must be mounted first to facilitate successful mounting of the other two.

One may answer the first two questions using the SAN Command Line Interface (CLI)/Graphical User Interface (GUI). Consider each question in turn. Given a failed server S_f, an administrator may query the SAN CLI/GUI to obtain the set of LUNs assigned to S_f. For each LUN, the administrator may use the SAN CLI to identify the hot-standby servers with connectivity to that LUN.

The answers to the last two questions are not available from the SAN CLI. They are initialization parameters whose values are dictated by an administrator and stored in a Production Configuration Database (ConfigDB). Without such a database, the administrator would be forced to perform the following time consuming discovery operations: 1) present a LUN to a hot-standby, and 2) examine the database resident on that LUN to discover its mount point and dependencies on other volumes. Step 2 is error prone and may slow down the recovery process.

With thousands of LUNs to manage, the storage team at MySpace maintains a ConfigDB of: (1) the SAN frame that hosts different LUNs, the unique id of each LUN (LUN-id), servers connected to each SAN frame, and LUN-ids assigned to each server, (2) The file system mount points and the order in which they should be mounted, and (4) The databases assigned to different file system mount points.

To discover the mount point for a LUN quickly, each LUN stores a file containing its logical LUN-id. This file is named *VolumeKey.dat*. When a LUN is presented to a hot-standby server, HoTMaN obtains its LUN-id using its VolumeKey.dat file. Next, HoTMaN looks-up the LUN-id in the ConfigDB to retrieve its mount point.

To maintain the ConfigDB up-to-date, simple scripts execute on a nightly basis to retrieve the list of LUNs assigned to each server and the connectivity of different servers to different SAN frames using the SAN CLI. These scripts update the ConfigDB to reflect the current state of the production environment.

HoTMaN employs the ConfigDB to automate manual processes used to migrate LUNs from a target server to a hot-standby. When an operator selects a target server, see 1B in Figure 1, HoTMaN uses the meta-data found in ConfigDB to present valid hot-standby servers, see 1C in Figure 1. The target server might either be a failed server or a server scheduled for regular maintenance.

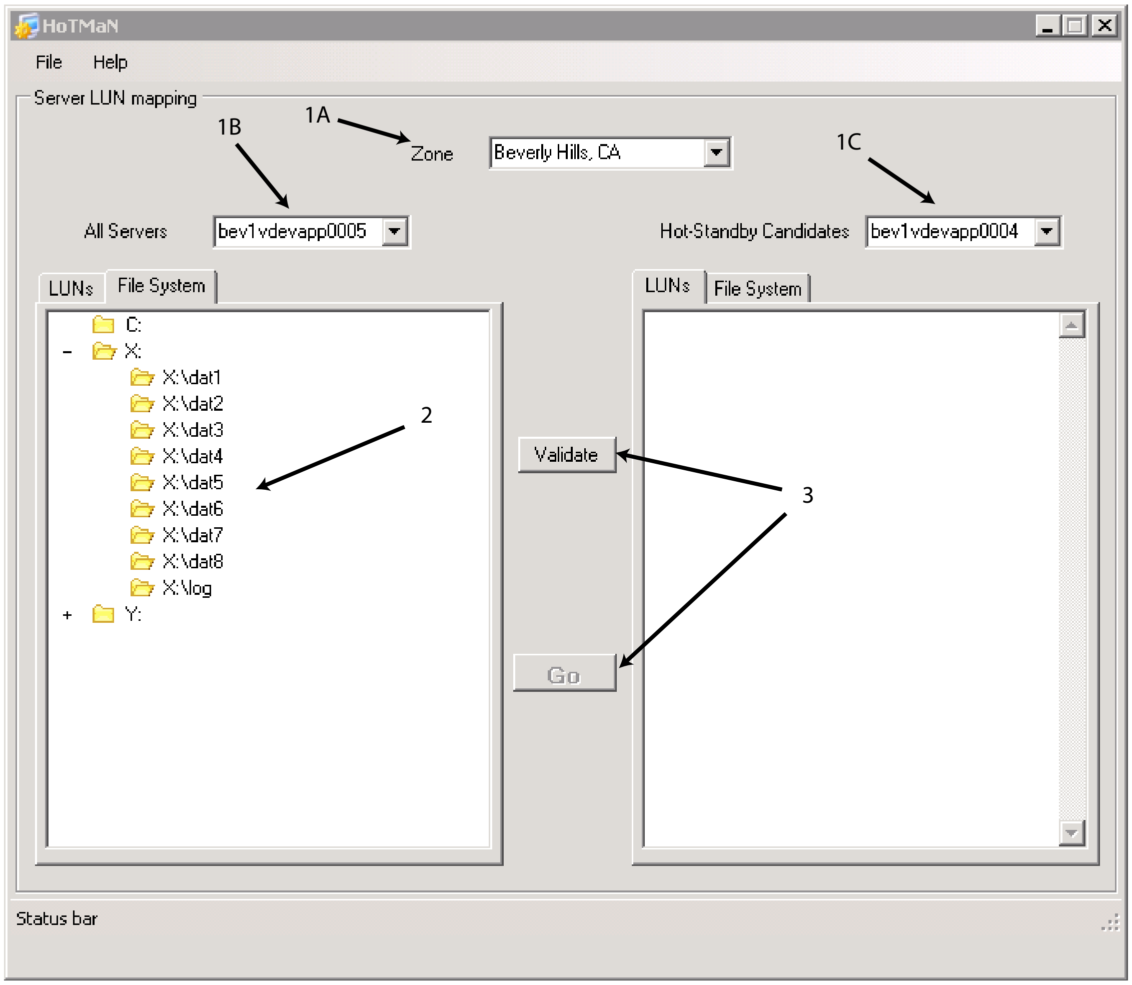
A key challenge is to verify ConfigDB contains up-to-date information such as the continued presence of the hot-standby on the SAN switch. The "Verify" button of HoTMaN, see Figure 1, enables the operator to validate the information pertaining to a chosen target server and its hot-standby. During the validation phase, HoTMaN employs the SAN CLI to verify the following hold true:

1. The number of LUNs assigned to the target server matches the number of LUNs reported by the ConfigDB.

2. The hot-standby has no LUNs assigned to it.

3. The hot-standby is connected to the SAN frame used by the target server.

Requirement 2 is specific to MySpace and one may modify this rule. For example, one may relax it by requiring a candidate server to have no existing mount point that conflicts with those of the failed server. See Section 3 for a discussion.



Zone Beverly Hills, CA

All Servers bev1vdevapp0005

Hot-Standby Candidates bev1vdevapp0004

Figure 1: Screen-shot of the HoTMaN GUI. 1A: Combo-box to choose the data-center hosting the failed server. 1B: Combo-box to choose the target, potentially a failed, server. 1C: Combo-box that shows the candidate hot-standby servers computed by HoTMaN to substitute for the target server. 2: File-system view showing the mount points for the LUNs on the target server. 3: "Validate" and "Go" buttons that enable the operator to perform sanity checks and initiate migration of LUNs from the target server to the chosen hot-standby. The "Go" button is enabled once validation is successful.

If the validation fails, in its current deployment, HoTMaN provides the operator with an error message and the option to proceed forward with the migration. This may cause HoTMaN to encounter additional errors and fail. An obvious alternative is for HoTMaN to prevent the operator from proceeding forward in the presence of errors.

Once the validation phase completes, the "Go" button is enabled. When the user clicks this button, the process to migrate LUNs from a failed server to a hot-standby is initiated. It begins by asking the operator if the target is a server scheduled for regular maintenance. If this is the case, with the operator's confirmation, HoTMaN shuts-down the instance of Microsoft SQL server on the target server and unmounts the file systems corresponding to its LUNs in preparation for migration. Obviously, these steps are irrelevant if the target is a server that has failed.

The migration process can be divided into three sequential tasks:

1. Migrate LUNs from a target server to a hot-standby using the SAN CLI.
2. Assign correct file system mount points to the LUNs presented to the hot-standby server.
3. Attach databases residing in the file systems to the instance of SQL Server running on the hot-standby.

These tasks are implemented by several scripts that are generated at run-time. HoTMaN uses Microsoft Windows WMI and psexec utilities to execute these scripts on the hot-standby remotely.

Below, we describe the details of each task in turn assuming three servers: a server hosting the HoTMaN application, a target server, and a hot-standby server. The term operating system refers to the Microsoft Windows operating system.

2.1 LUN migration

HoTMaN uses the SAN CLI to un-present LUNs from the target server and to present them to the hot-standby. The identity of these LUNs is obtained from ConfigDB. HoTMaN is designed to use the volume global unique identifier (GUID). These are strings of the following form: "\\?\Volume{4116b677-9dc2-11d3-8839-806d6172696f}". Thus, before the LUNs are presented to the hot-standby, HoTMaN generates a script to disable the capability of the hot-standby's operating system from automatically assigning either a label or a drive letter to the LUNs presented to it. Once the SAN CLI confirms that the LUNs are presented to the hot-standby, HoTMaN updates ConfigDB to show the hot-standby as the new owner of the migrated LUNs. By performing this update at this step, HoTMaN ensures ConfigDB is consistent with the SAN configuration in the event HoTMaN fails during one of the remaining steps.

Next, HoTMaN uses WMI to retrieve the volume GUIDs detected by the operating system of the hot-standby. Their total number should match the number of LUNs presented to the hot-standby. If there is a mis-match, HoTMaN invokes the rescan command of the Windows diskpart utility on the hot-standby to detect new disks and volumes. In some instances, this might be performed two times before the hot-standby detects all the migrated LUNs. If after three invocations the number of volumes does not match the number of presented LUNs, HoTMaN reports the limitation and requests for the operator intervention to manually invoke diskpart on the hot-standby and ensure the presence of the right number of volume GUIDs prior to proceeding forward.

If the operator decides the hot-standby is faulty and should be replaced, it is still correct for the ConfigDB to reflect this hot-standby as the owner of the LUNs because the SAN has migrated LUNs to the hot-standby. The operator may employ HoTMaN to choose this hot-standby as the target server and choose another server as the new hot-standby, repeating LUN migration from the beginning.

The output of this step is a set of volume GUIDs corresponding to the LUNs migrated to the hot-standby.

2.2 Discovery of file system mount points and their assignment

HoTMaN discovers the correct mount point of each volume GUID by creating a script that mounts each to a temporary mount point, reads its VolumeKey.dat file to obtain the assigned LUN-id, and queries ConfigDB for the mount point that should be assigned to this GUID. A naive approach is to perform this process for every volume GUID. With tens of LUNs migrated from a target server to a hot-standby, HoTMaN generates one script that mounts all volume GUIDs to different temporary mount points, reads the LUN-id from the different VolumeKey.dat files, and stores them in a file on the hot-standby. Next, HoTMaN remote copies this file from the hot-standby to its local disk drive and queries ConfigDB for the mount point of each LUN-id. There is a one-to-one correspondence between the order in which the volume GUIDs are mounted to a temporary mount point (in the script) and the LUN-ids retrieved from the hot-standby. This order enables HoTMaN to match the final mount point retrieved from ConfigDB for each volume GUID.

Next, HoTMaN creates a script to un-mount the volume GUIDs from their temporarily assigned mount points and to mount each to its final mount point (as dictated by ConfigDB, see the previous paragraph). HoTMaN uses the psexec utility to executes this script on the hot-standby.

Finally, HoTMaN uses WMI to retrieve the list of GUIDs and corresponding current mount points on the hot-standby. This list must be either the same set or a superset of the mount points assigned by HoTMaN. A superset because the list may include standard mount points for the local disk partitions such as “C:\” and “D:\”

2.3 Attaching databases

HoTMaN uses OLEDB to execute SQL server commands to attach databases that reside in the migrated LUNs to the SQL server instance executing on the hot-standby. This requires discovery of the database names. Once again, ConfigDB maintains the association between the file system mount points and the corresponding databases. HoTMaN queries ConfigDB to generate the correct run-time OLEDB commands to attach all databases to the instance of Microsoft SQL server on the hot-standby.

Once this step is complete, the hot-standby is ready to process queries.

3 Conclusions and Research Directions

HoTMaN demonstrates feasibility of a unified user interface to substitute a target server with a hot-standby assuming a SAN infrastructure. The target server might be either a failed server or a server scheduled for maintenance. If the later, HoTMaN is able to gracefully shutdown the Microsoft SQL Server of the target and un-mount its file system volumes prior to initiating the migration of LUNs to the hot-standby.

Currently, HoTMaN meets 90% of its production use cases at MySpace [9]. This is because HoTMaN provides a simple interface that does not try to do too much. Logically, HoTMaN implements a work-flow to identify a hot-standby for a failed server and resurrect databases of the failed server on the hot-standby.

The remaining 10% include both conceptual and logical improvements. A conceptual improvement may require a complete re-design of HoTMaN to approach management of storage top-down: instead of migrating LUNs, migrate databases from a failed server to one or more hot-standby servers. With this change, the operator is no longer distracted with the physical LUNs and their file systems. Instead, the operator is provided with the abstraction of databases assigned to the failed server that should be migrated to one or more hot-standby servers.

A logical improvement is to relax the requirement that the fail-over server be a hot-standby. The fail-over server may have assigned LUNs and service requests actively. This improvement would enable the operator to migrate the services of a failed server to one or more currently active servers that are not fully utilized. This is particularly useful when all hot-standbys on a SAN frame have been exhausted. HoTMaN may utilize the load imposed on the existing servers to present the operator with the least loaded server as the candidate fail-over server.

The logical improvements motivate the observation that no one management tool (such as HoTMaN) may support all use cases at an organization such as MySpace. Indeed, use of HoTMaN across different enterprises may not be practical due to their business specific processes and procedures. We hypothesize the feasibility of an expert system to automate the development of software tools such as HoTMaN. Such a tool abstracts enterprise storage management as a set of configuration 'states'. Each state represents a possible configuration of all of the infrastructure elements that make up the storage system. We propose developing a description language that can be used to describe the elements, their interconnections and dependencies as well as a set of desired states. An automated tool can analyze this meta-data and develop a set of operations that would perform the transition from one state to another. This same framework could also be used as a monitoring tool by analyzing the run-time state of the system (in effect performing the mapping from configuration to description language) and comparing this state to a known set of error states. It may employ tools such as Nagios [6] to detect failed/unresponsive servers to automate migration of LUNs, file systems and databases to other servers.

4 Acknowledgments

We wish to thank Hotham Altwaijry and Vinayak Borkar for participating in the design and implementation of HoTMaN, and Michael Carey for his valuable comments on an earlier draft of this paper.

References

[1] M. Creeger. CTO Storage Roundtable, Part Two. *Communications of the ACM*, 51(9):47–52, 2008.

[2] D. DeWitt, S. Ghandeharizadeh, D. Schneider, A. Bricker, H. Hsiao, and R. Rasmussen. The Gamma database machine project. *IEEE Transactions on Knowledge and Data Engineering*, 1(2), March 1990.

[3] S. Ghandeharizadeh and D. DeWitt. Hybrid-Range Partitioning Strategy: A New Declustering Strategy for Multiprocessor Database Machines. In *VLDB*, Brisbane, Queensland, Australia, August 1990.

[4] S. Ghandeharizadeh, S. Gao, C. Gahagan, and R. Krauss. An On-Line Re-Organization Framework for SAN File Systems. In *ADBIS*, volume 4152 of *Lecture Notes in Computer Science*, pages 399–414. Springer, 2006.

[5] M. Livny, S. Khoshafian, and H. Boral. Multi-Disk Management Algorithms. In *Proceedings of the 1987 ACM SIGMETRICS Int'l Conf. on Measurement and Modeling of Computer Systems*, May 1987.

[6] Nagios. http://www.nagios.org. 2008.

[7] M. A. Olson, K. Bostic, and M. I. Seltzer. Berkeley DB). In *USENIX Annual Technical Conference, FREENIX Track*, pages 183–191, 1999.

[8] M. Stonebraker, S. R. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The End of an Architectural Era (It's Time for a Complete Rewrite). In *VLDB*, Vienna, Austria, September 2007.

[9] A. Wergeles. HoTMaN Discussions on October 17, 2008. Private Communication, 2008.