MICROSOFT VSS WRITER FOR SQLITE DATABASE

[1]SIDDHARTH BHOLA, [2]SUMIT MURARI, [3]ANIRUDHATAMBOLKAR, [4]S R DHORE

1,[2,3,4]Department of Computer Engineering, Army Institute of Technology

Abstract—We implement VSS(Volume Shadow Copy Service) Writer supported by the Windows Operating system to create point in time consistent shadow copies (snapshots) for SQLite database without affecting the performance of the system on which database is resident.Microsoft VSS Writer for SQLite database will help other applications to use SQLite for time-critical and concurrent transactional applications providing an off-host backup mechanism.VSS backup technology for SQLite can be used for many concurrent transactional application,where simple back up takes a lot of time, and hence can further increase the use of open source SQLite and provide much better back up technique compared to conventional one available in market.

Keywords: COM Interface,Snapshot, SQLite, VSS(Volume Shadow Copy service) and VSS Writer

I. INTRODUCTION

The Volume Shadow Copy Service (VSS) is a set of COM interfaces that implements a framework to allow volume backups to be performed while applications on a system continue to write to the volumes. It captures and copies stable images for backup on running systems, particularly servers, without unduly degrading the performance and stability of the services they provide.

VSS serviceIt is consist of 4 major components that communicate with each other: VSS requester, VSS writerVSS provider and VSS Coordinator.

A. VSS requester: The VSS requester is the software that commands the actual creation of shadow copies or other high-level operations like importing, breaking or deleting them.

B.VSS provider: The component that creates and maintains the shadow copies. This can occur in the software or in the hardware. The Windows operating system includes a VSS provider that uses copy-on-write.

C.Writers: Writers are applications or services that store persistent information in files on disk and that provide the names and locations of these files to requesters by using the shadow copy interface. During backup operations, writers ensure that their data is quiescent and stable—suitable for shadow copy and backup. Writers collaborate with restores by unlocking files when possible and indicating alternate locations when necessary.

1)Writer Event Handling: A writer's VSS operations are initiated through the receipt of COM events.

When no events are present, a writer does not perform VSS operations (such as a VSS backup or restore). Instead, it performs its normal work, such as responding to database queries, managing user data, or providing other services.

2)Backup and Restore Events: Depending on whether it is participating in a backup or restore, a writer will receive between two and seven events, in addition to an initial Identify event. Handling these events constitutes (from the point of view of a writer) the life cycle of a backup or restore operation.

In a typical backup operation, a writer would handle the following events (in addition to an initial Identify event):

a. OnIdentify: For OnIdentify, VSS writer need to report/return back all instance names and file names stored in Windows registry.

b. OnPrepareSnapshot: Do any lengthy operation before snapshot, in this function.

c. OnFreeze: Roll forward all journals (master journal, roll journal, statement journal, journals associated with TEMP DB, memjournal). Hold all writes to any database file

d. OnThaw: Release all write that were hold.

e. On Post Snapshot: Do any lengthy post snapshot operation her e.g. copy fileIn case of SQLite VSS writer this function would be mostly empty.

f. OnAbort: Release all writes that were on hold.

g. On Backup Complete: The On Backup Complete method is called by a writer following a Backup Complete event. It is used to perform operations considered necessary following a backup. These operations cannot, however, modify the Backup Components Document.

h. On Pre Restore: The On Pre Restore method is called by a writer following a Pre Restore event. This method is used to put the writer in a state to support the restore for instance, taking database services offline and to make modifications in the Backup Components Document of the requester that is restoring files (such as setting the restore target to override the original restore method).The Pre Restore event occurs before backed-up data is actually being restored. This is an opportunity for the writer to determine what is being restored. This method enables the writer to determine what is being

restored, to retrieve stored private metadata in the stored Backup Components Document, and to update that data.

i. On Post Restore: The On Post Restore method is called by a writer following a Post Restore event. It is used to perform operations considered necessary after files are restored to disk by a requester. These operations cannot, however, modify the Backup Components Document.

If necessary, a writer should remove any temporary files and release any system resources that it needed for its participation in the restore.With the generation of a PostRestore event, a requester's Backup Components Document becomes a read-only document.

II. TECHNICAL SPECIFICATION

Any backup operation that attempts to copy a full and stable image of a system must deal with the following:

A. Inaccessible files during a backup. Running applications frequently need to keep files open in exclusive mode during a backup, preventing backup programs from copying them.

B. Inconsistent file state. Even if an application does not have its files open in exclusive mode, it is possible—because of the finite time needed to open, back up, and close a file—that files copied to storage media may not all reflect the same application state.

C. Need to minimize service interruptions. To ensure file accessibility and the integrity of the data being backed up can require the suspension and/or termination of all running programs during a volume backup. For large disk systems, this could be hours in duration.

D. Incompatible vendor implementations of volume capture. Many providers of RAID devices provide volume capture mechanisms. However, each vendor has its own interface and each must get support from the backup vendors for their volume capture interfaces. This means that backup application vendors must support multiple volume capture implementations, which is undesirable.

E. Lack of application coordination. Many devices that support a volume capture do not support the coordination of running applications with the freeze of data on disk. For those devices that do, as with the backup applications, each vendor has a different interface.

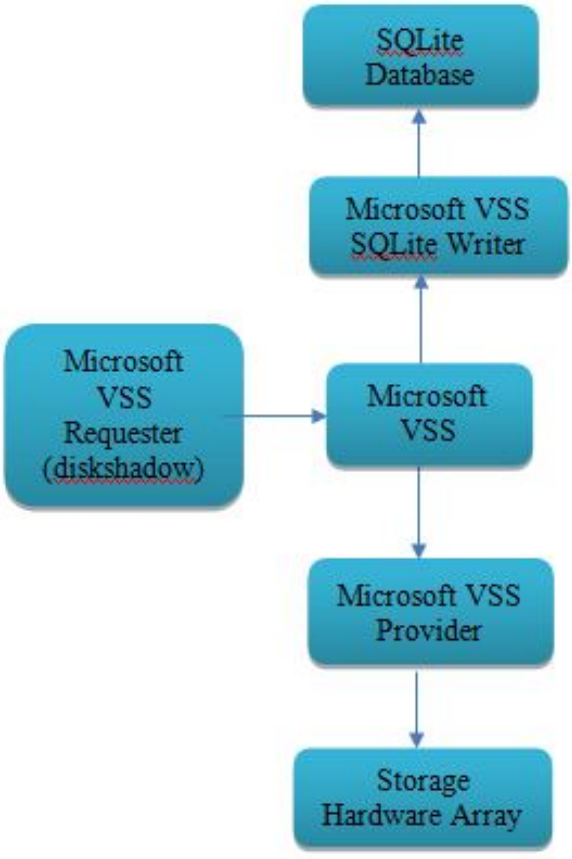
F. Limited support for non-RAID devices. Few if any conventional disk vendors provide support for any sort of volume capture in their device drivers. This means that capture mechanisms are limited to certain disk systems, and cannot typically support the backup of system areas.

G. Need to handle updates to disk during volume capture. Although storage-vendor-provided volume capture mechanisms can freeze the state of data on disk, they do not always interoperate with running applications. This frequently means that data sent to the volume while a storage device is undergoing volume capture may be lost.

H. Consistent multivolume backup. The storage device executes these volume captures, so there is generally no mechanism for coordinating the timing of the data freeze. This is particularly true if the devices come from separate vendors. Therefore, if several storage volumes are involved in a backup with a volume capture, the time image preserved for each volume may not be consistent.

III. SYSTEM DESIGN

A. Functional Diagram:



B.Design Considerations and diagram description:

1. SQLite VSS writer needs to interact with sqlite3. There could be many sqlite3 instances at given point in time
2. There is no single coordinator service amongst various sqlite3 instances
3. As a part of project we need to use windows registry as central coordination
4. Whenever sqlite3 CLI creates sqlite3 db file, it will register the file name in registry, along with named pipe

5. When sqlite3 DB is attached by sqlite3 CLI,CLI needs to add entry for named pipe which VSS writer can use to interact
6. VSS writer would be able to instruct sqlite3 to freeze or unfreeze given database
7. If a sqlite3 instance is not attached to any sqlite3 CLIs then, VSS writer needs to just open that DB file and lock, so that till snapshot of storage below is created, no slqite3 CLI can open it

CONCLUSION

A. SQLite backup- current solution vs proposed solution

1) Existing method 1

Historically, backups (copies) of SQLite databases have been created using the following method**:**

a) Establish a shared lock on the database file using the SQLite API (i.e. the shell tool).
b) Copy the database file using an external tool (for example the unix 'cp' utility or the DOS 'copy' command).
c) Relinquish the shared lock on the database file obtained in step 1.

This procedure works well in many scenarios and is usually very fast. However, this technique has the following shortcomings:

a) Any database clients wishing to write to the database file while a backup is being created must wait until the shared lock is relinquished.
b) It cannot be used to copy data to or from in-memory databases.
c) If a power failure or operating system failure occurs while copying the database file the backup database may be corrupted following system recovery.

2)Existing Method 2

Using online backup APIs SQLite database can be backed up. http://www.sqlite.org/c3ref/backup_finish.html#sqlite3backupinit

However shortcomings are

a) Backup agent required to be running on source machine. The agent integrates with online backup APIs and sends data to remote location
b) CPU cycles of host machine are consumed
c) Hardware array level snapshot of SQLite database is not possible
d) Off-host backup of SQLite database is not possible

B.Proposed Method:

Microsoft VSS framework enables backup applications, databases and storage arrays to co-ordinate for application consistent snapshot at hardware array level. We implement the interfaces provided by CVssWriter class to snapshot the database of Sqlite. We create sqlite3 object to copy data between the two databases (from a file into the in-memory database, or vice-versa). Then call a method to copy the entire source database to the destination and finish the backup by cleaning up resources allocated by SQLite methods. To ensure that error handling for multiple parallel backup and restore sessions is performed correctly, and to ensure that one backup or restore session does not corrupt another, following instructions are to be carried out :

1) If a writer's event handler **OnFreeze** calls the **GetSessionId, SetWriterFailure**, or **SetWriterFailureEx**method, the event handler must call the method in the same thread that called the event handler.

2) Writer's implementation of an event handler such as OnFreezecan offload work to worker threads if desired, as long as each worker thread marshals any needed error reporting back to the original event handler thread.

Advantages of the Method:

a) Off-host backup of SQLite is possible.

b) Snapshot takes few seconds to complete as against SQLite online backup APIs.

c) Snapshot doesn't take any space of server, but resides in storage array (If source SQLite database was residing on hardware array).

d) No separate application needs to be developed to backup, windows commands can create SQLite database snapshot.

REFERENCES

[1] Microsoft® Volume Shadow-Copy Service and Its Role in an Organization's Total Backup Strategy - A White PaperFor IT Managers, CTOs, and Security Administrators Concerned About Data Loss Due to Corrupt or Incomplete Backup Table of Contents , from open file manager : www.esafestor.com

[2] Microsoft Developers Network:http://msdn.microsoft.com/en-us/library/windows/desktop/ bb968832%28v=vs.85%29.aspx

[3] Microsoft TechNet : http://technet.microsoft.com/en-us/library/cc785914%28v=ws.10%29.aspx

★★★