Shadow copies - a peek under the hood

Adi Oltean 11 Dec 2004 8:54 PM | 11

Starting with XP and Windows Server 2003 we have a new shadow copy functionality feature built-in the operating system. In my previous posts, I already mentioned some of of the high-level scenarios that are enabled by this feature. One example is backup, and the ability to avoid the "open files" problem. Another example is the Shadow Copies for Shared Folders feature, also known as "Previous Versions". But I haven't mentioned how shadow copies are actually implemented.

First I would like to point out that the VSS (Volume Shadow copy Service) infrastructure allows third-party shadow copy implementations to be installed in the OS, and make them accessible through a common API. We won't describe here this general infrastructure. For now, we note that, in general, the built-in shadow copy implementation will be used if no other implementations are present in the system. And only this built-in implementation is the subject of the article below.

The main idea behind shadow copies is to have a static, immutable view of a certain file or directory. For example, no matter what you do with your word document (overwriting it or even deleting it), you still have a stable image of this file at, say, 12.00 PM. So if you seriously screw up the current document, you can quickly recover its contents by right-clicking on it in explorer, and selecting the "Previous Versions" tab. (Note, however that you must access this file through a share with SCSF enabled, otherwise the Previous Versions tab won't be available).

Under the cover, this versioning functionality is not implemented by copying the entire file each time the system creates a previous version. We use instead a copy-on-write approach, that deals only with fragments of changed data. The unchanged data is internally shared between the original file and the previous version.

Now here comes the interesting stuff. The copy-on-write functionality is not implemented at the file system level. It is actually implemented below the file system, at the volume level where you deal only with equal-size blocks (or sectors) of raw data. It would be much more complicated to to implement a versioning scheme above the file system level. The reason is that you need to keep track of all the file system metadata changes, for example. Being below the file system, you only need to deal with block/sector-level changes, and therefore you end up with a simpler implementation.

VOLSNAP.SYS just treats the volume contents as an uniform array of 16K blocks at sector level (not NTFS cluster level). From the NTFS point of view, this array includes of course the high-level NTFS data structures, the NTFS metadata like the $MFT, $Secure, etc. VOLSNAP doesn't need to know where $MFT is located, for example. Remember that the size of a NTFS cluster (4K usually) is usually different than the size of a VOLSNAP block (16K) which is also different than the disk sector size (512 bytes). For example one VOLSNAP block (16K) will contain exactly 32 sectors.

To understand better the copy-on-write algorithm, let's go to a practical scenario:

1. Let's assume that a file exists on C:\. Let's assume that the file metadata (contained in some file records in $MFT) belongs to VOLSNAP block A. Also, the file streams (file contents) belong to VOLSNAP blocks B, C, D, E. But again, VOLSNAP is not aware of the NTFS on-disk structures.
2. The Administrator enables Shadow Copies with default schedule at 11:55am.
3. A shadow copy is made at 12.00 AM. At this point, VOLSNAP creates an internal "shadow storage" file. This file is used to store any VOLSNAP blocks that were changed after the snapshot was created. The location of this shadow storage can be configured using the Shadow Copy UI or with the VSSADMIN ADD SHADOWSTORAGE command.
4. Application changes first 5 clusters of the file at 1:00pm. Let's assume that the VOLSNAP blocks B and C are now changed. Also, the file metadata might be changed, for example the file size and last modification time (so VOLSNAP block A is also changed). The copy-on-write works as follows: the new contents of these blocks (A, B, C) goes to the target volume as usual, but the original value of these sectors is kept by VOLSNAP on the shadow storage.
5. Now the user reads the file from the shadow copy. Note that the shadow copy is present in the system as a separate read-only volume device. The mounted NTFS file system on this new device is of course separate from the mounted file system from the original volume. (you can see this device when you run the command VSSADMIN LIST SHADOWS - it usually looks like this: \Device\HarddiskVolumeShadowCopy34).

The read on the shadow copy works in the following way: let's assume that the user reads the file above. The read I/O is intercepted by VOLSNAP as a sequence of reads for sectors contained in blocks A - E. For the first blocks (A, B, C), VOLSNAP sends back the saved versions of these sectors at 12.00 AM. For the next blocks D, E which weren't changed in the meantime, VOLSNAP sends back the current contents from the original volume.

In the end, the file system on top of the shadow copy device receives an exact copy of these sectors at the time of shadow copy creation (12.00 AM). So the user sees the exact copy of the file at 12.00 AM.

Comments

Dave 11 Dec 2004 11:20 PM #

Interesting post.

I type 'vssadmin list shadows' and get:

Error: 0x8000ffff

In the event log is:

7031

The COM+ System Application service terminated unexpectedly. It has done this 1 time(s). The following corrective action will be taken in 1000 milliseconds: Restart the service.

then 7034

The MS Software Shadow Copy Provider service terminated unexpectedly. It has done this 1 time(s).

Any ideas?

Dave 11 Dec 2004 11:23 PM #
Good 'ol windows eh?

Thanks to google the solution is:

regsvr32 ole32.dll

Another KB article?

Adi Oltean 12 Dec 2004 2:03 AM #
I noted that several customers had the same problem in the past. It turned out that ole32.dll was unregistered by mistake by the uninstall process of some applications. This causes further failure of the COM infrastructure. The workaround is to re-register ole32.dll (btw, ol32.dll comes registered by default in Windows XP).

I am not sure if this is the problem that happened in your case.

evil797 12 Dec 2004 6:00 PM #
Do not try to register ole32.dll by reinstalling windows xp. Try the following command in the interpreter : regsvr32 ole32.dll

余啊雷 12 Dec 2004 11:00 PM #
lazybones » Back to work

余啊雷 15 Dec 2004 2:36 AM #

余啊雷 15 Dec 2004 2:38 AM #

余啊雷 15 Dec 2004 4:02 AM #

余啊雷 24 Mar 2008 11:54 AM #
PingBack from http://drinkingrecipesblog.info/antimail-shadow-copies-a-peek-under-the-hood/

余啊雷 24 Mar 2008 11:38 PM #
PingBack from http://caferestaurantsblog.info/antimail-shadow-copies-a-peek-under-the-hood/

余啊雷 18 Jan 2009 11:51 AM #

PingBack from http://www.keyongtech.com/4218579-how-use-older-shadow-copies