ZFS And Digital Intermediate

Bob Friesenhahn
Simple Systems
bfriesen@GraphicsMagick.org
June 15, 2008

Overview

ZFS is an advanced pooled filesystem from Sun Microsystems which was first introduced in the Solaris 10 operating system in November, 2005 and is released as free open source as part of the OpenSolaris operating system. ZFS discarded 25 years of filesystem implementation "rules" and is a clean-slate implementation based on fresh thinking. This paper describes why the Digital Intermediate (DI) postproduction community should be keenly interested in ZFS and should start deploying it today.

The Nature of Digital Intermediate

The Digital Intermediate approach to postproduction typically produces hundreds of thousands (or even millions) of large files, with a single uncompressed image file per frame. Typical file sizes are on the order of 12MB each, but can be smaller or larger (8MB, 50MB) depending on colorspace (RGB, YCbCr), bit depth (10 or 16-bit), and resolution ("2K", "4K"). A typical finished work has an average of 144K frames (at 24 frames per second) and consumes about 2TB of storage. A work in progress can use vastly more storage. Some recent productions captured digitally have required over 200TB of storage since "dailies" were delivered in full resolution. Clearly storage requirements are quite high for DI.

The typical DI workflow involves a capture/ingest stage, one or more modifications stages, one or more playback stages, and final printing to film, authoring for DVD, Blu-Ray, and Digital Cinema, and archiving to magnetic tape.

Works shot on film need to be converted to digital form using a telecine or film scanner. Some film scanners are capable of capturing over 24 2K (2048 pixels per scanned row) frames per second, or over 7 frames per second for 4K (4096 pixels per scanned row). To put this in perspective, real-time 2K scanning requires at least 288MB/second of continuous write bandwidth. The capture rate is primarily limited by the currently available storage interface rates so if the available storage interface rate improves, the specified capture rates are sure to improve as well.

Work in digital form need to be played back for evaluation. Bandwidth requirements are similar to the real time capture scenario however real time requirements are quite rigid so that bandwidth specifications are typically on the order of 300MB/second even though the math says it takes a bit less.

Given that quite a bit of production is done using computers (e.g. rendering and compositing) the lines between production and postproduction have become much less well defined and original files may be transferred from production to postproduction. These could be in the form of finished DPX or TIFF files or they could be in format like EXR. Some shops have merged production and postproduction so that rendered output is used immediately in place by postproduction.

The Nature of ZFS

ZFS is a from-scratch pooled filesystem which discards all of the "hard and fast" rules set in stone over the past 25 years and re-thinks the storage model so that perhaps it will have value for the next 25 years. ZFS provides a standard POSIX filesystem but with support for NTFS/CIFS/NFSv4 ACLs, resource forks, and other features necessary to support Windows and Apple OS-X requirements.

A pooled filesystem borrows the "storage pool" concept pioneered in Storage Area Networks (SANs). Disks added to a ZFS pool expand the size of the pool so that all filesystems in the pool benefit unless they are artificially limited by quotas. One or more filesystems may be created in the pool and use the pool's shared resources. Filesystems take about as much thought to create as creating a directory. Hundreds of filesystems are quite feasable but thousands of filesystems are observed to substantially hinder initialization time and increase resource consumption so it is best not to go overboard.

Creation of ZFS storage pools is quite fast. Once all the requisite disk LUNs/devices have been figured out (the hard part), creation of the pool takes just a second or two. Likewise, creation of new filesystems takes just a fraction of a second so they can be created on demand for temporary purposes. ZFS storage pools may be enlarged at any time by simply adding more disks. ZFS supports simple load share (similar to RAID-0 but not the same), mirrors (similar to RAID-1 but not the same), raidz (similar to RAID-5 but not the same) and raidz2 (similar to RAID-6) but not the same. When disks are added to the pool they are added as a virtual device (VDEV). For example, two raidz2 VDEVs of six disks each could be added to the pool. The pool would then load-share writes across the two VDEVs. For the best performance VDEVs should be comprised of mirrors. ZFS supports any level of mirroring. For example data could be mirrored across three disks in order to improve data protection and tripple the read bandwith. ZFS also supports duplication of data blocks on the same disk for extra protection.

Data in ZFS is usually stored in fixed size blocks (unless compression is enabled). The default block size is 128K. Since the block size is large, fragmentation becomes much less of an issue. ZFS uses a copy-on-write approach so that if a disk block is to be modified then a new block is allocated and the old block is freed if it is no longer needed. This approach makes the filesystem much more secure against data loss and enables quite useful features (filesystem snapshots and clones) to be implemented. Data is always checksummed and all data is validated using the checksum prior to use. Data which fails the checksum test is corrected using ZFS (software) RAID mechanisms. Note that this is far more robust than any expansive hardware RAID system is able to provide.

Note the many "but not the same" notes associated with ZFS RAID equivalents. For example, while true RAID-0 will split blocks across mutiple disks, ZFS uses load sharing so that 128K blocks will be written on different disks based on a load share algorithm which considers the disks recently used, the current disk loading, and a randomization factor. This is quite beneficial for sequential access since a write will go to the most available disk and blocks are read from different disks according to how they were written. Since blocks are read from different disks, the "fragmentation" which is such a bothersome issue on legacy filesystems is nicely side-stepped since while one block is being read, the disk which will be used next is already seeking to position so that the seek time is no longer a factor. In fact, ZFS notices sequential access and performs read-ahead so that the data is available when needed. When mirrors are used there is the added benefit that either disk may be used to source a block of data so that using mirrors doubles the available read data rate.

ZFS snapshots are an extremely powerful and useful feature. A filesystem snapshot may be requested at any time and takes less than a second to complete. A snapshot is a read-only directory which is visible under a .zfs subdirectory in the filesystem. Any number of snapshots may be taken and given arbitrary names such as the date that the snapshot was taken. Snapshots may be used as a real time "backup" mechansim, and as a basis for the "clone" and "send" features. Snapshots are useful prior to undergoing any risky large-scale alteration of the filesystem since if an error is made, the filesystem may be almost instantaneously "rolled back" to the snapshot version. This is like an "undo key" for any error.

ZFS clones are also an extremely powerful and useful feature. A “clone” is a new filesystem created from a snapshot but with write access. For example, you could start off the day by taking a snapshot of yesterdays work and then creating a clone of it for today's work. Once today's work has been deemed a success the clone may be promoted to be the “master” and the original snapshot deleted.

The ZFS send feature allows the filesystem to be duplicated on another server (e.g. as a backup). Once a snapshot is taken, “zfs send” is used to send the changes over the network to a similar filesystem on the other server.

ZFS supports a “scrub” feature which runs in the background and makes sure that all data is coherent. It reads all data and metadata blocks and performs any repairs which are required.

Application of ZFS to Digital Intermediate

It is clear that ZFS satisifies many requirements for DI:

- Soon to be in an operating system near you (Solaris, OpenSolaris, FreeBSD, OS-X, Linux? ...)
- Excellent protection against data loss or corruption, and particularly the dreaded silent type.
- Easy to build large storage systems of several hundred terrabytes.
- Administration tools are easy to understand and astonishingly fast (most tasks take less than a second).
- Excellent sequential write performance.
- Excellent sequential read performance.
- Performance impact of fragmentation is limited due to use of large (128K) blocks and the ability to load-share block reads/writes across individual disks. Reads are scheduled in advance of consumption so that disk seeks are much less of a factor.
- I/O capacity can be scaled to any level by simply adding more disks.
- Total storage capacity may be scaled to any level by simply adding more disks.
- New filesystems can be instantaneously created to handle short term jobs and intantaneously removed when they are no longer needed.
- Filesystem quotas are inherited so that subordinate filesytems inherit the limits of their parent, but may have their own limits as well. This allows several filesystems to share a space but with a maximum limit per filesystem.
- Snapshots provide an instant “backup” and “recover” capability. Snapshots consume no space until the data changes.
- Clones provide a temporary working area to make changes without altering the original but may be promoted to be the “master” if the work turned out to be good.
- Software-based RAID has no noticeable impact on performance and is easier to manage than controller or SAN-based RAID.
- Cheap SATA disks with raidz (“RAID-5”) or raidz2 (“RAID-6”) may be used for second level or archive storage.
- Fast SAS or FC disks with load-shared mirrors (“RAID 1+0”) may be used for the highest

performance storage.

- ZFS in portable storage arrays makes sense due to its “healing” powers.
- No slow “fsck” required to recover from a system crash, and no “journalling” to slow down the writes and safely lose the most recently written data.

While ZFS can easily use SAN disk LUNs, ZFS is not a shared SAN filesystem. As such it is best for direct workstation use or as the basis for a network server.

Where to Learn More

There is quite a lot of ZFS information on the web since ZFS has attracted so much attention and has been taking off like a rocket. Good places to go for information include the OpenSolaris web site at http://www.opensolaris.org/os/community/zfs/ or WikiPedia at http://en.wikipedia.org/wiki/ZFS.