Intelligent Storage Systems in Digital Preservation

Tom Creighton; Family Search; Orem, UT, United States

Abstract

Great effort has been expended in making digital preservation repository systems reliable in terms of how they manage the objects within the repository. For the most part, these systems place no more demand on the storage system in which they store their digital objects than any software system places on a POSIX file system. All issues of integrity maintenance tend to be handled by the preservation system itself. For example, regular fixity checks are typically carried out by the preservation system running through its catalog inventory and reading each file stored in the storage system in order to compare with a previously calculated hash value stored in the catalog. This requires the attention of the preservation system itself in most cases. Large simplification and performance gains are to be made by delegating more of these integrity checking and data recovery issues to the storage system.

Introduction

Great effort has been expended in making digital preservation repository systems reliable in terms of how they manage the objects within the repository. For the most part, these systems place no more demand on the storage system in which they store their digital objects than any software system places on a POSIX file system. All issues of integrity maintenance tend to be handled by the preservation system itself. For example, regular fixity checks are typically carried out by the preservation system running through its catalog inventory and reading each file stored in the storage system in order to compare with a previously calculated hash value stored in the catalog. This requires the attention of the preservation system itself in most cases. Large simplification and performance gains are to be made by delegating more of these integrity checking and data recovery issues to the storage system.

Storage Systems Get Smarter

Modern storage systems go well beyond the relatively simple redundant array of inexpensive disks (RAID) models that have served us well for many years. Instead of "striping" data across a RAID set of four, five or a few more drives, vendors are now putting data from files that you write into chunks, spread across many, perhaps 20, drives. This allows much better performance in most cases just because the load can be spread across so many more disk drives. But in addition to spreading the load, vendors such as XIO automatically detect when a disk drive is having trouble, and will silently remap the data chunks stored on that drive so that it does not present a risk to data recovery. The disk can then be automatically re-tuned, or rejected, based on the types of failures.

Recent studies performed at Carnegie Mellon University demonstrate a failure rate for all types of disk drives to be at about 52,560 hours MTBF. This is much worse than the industry has traditionally claimed. But whether it is 52,560 or 600,000, or 1.5 million hours MTBF, if we are charged with maintaining data integrity for very long periods of time, we must manage those failure rates via multiple copies. And we need systems that can detect when errors occur, and correct them.

Newer storage arrays are moving away from traditional RAID configurations and are using erasure coding methods such as Reed-Solomon to ensure no data loss. This results in lower overhead for parity bit maintenance, but also works well with the concept of spreading data files in chunks over many disk drives. Some storage systems, such as Dell Compellent, also automatically map storage chunks to higher performance devices such as SSD, when those data chunks are requested more often than others. Dell, IBM, HP, and other vendors also provide systems that "rebuild" failed drives in the cage.

Implications for Digital Preservation

While it is a great boon to all industries to have intelligent subsystems perform more services for the application stack that depends on the data stored in those subsystems, in the case of digital preservation systems, it's important to have some level of awareness in the digital preservation management system about what is happening to the data in the underlying storage system.

For example, suppose an organization uses Fedora as its digital repository. And for purposes of managing the artifacts in its care over time, it chooses to make use of cloud storage such as Amazon's S3 or Glacier. In both of those systems there are automatic integrity checks made all the time, not just at read time. Unfortunately there is no way for either S3 or Glacier to inform Fedora that an integrity check was made on a particular digital object stored in the system. If the managed storage system detects an integrity failure, it automatically replaces the object from one of several replicas. But again, this action, which would typically be considered a preservation action, cannot be communicated to Fedora. And so while we might be pleased to have the service that S3 and Glacier provide, its value is somewhat diminished because we can't know when the last integrity check was made, or what its result was.

The Family Search Approach

At Family Search we have determined that our digital preservation system, which currently contains tens of petabytes of digital objects, can most cost effectively be maintained on tape. We currently use LTO-4 tape, but are moving to Oracle's T10000-C enterprise class storage cartridges. These offer much greater capacity, better performance, longer life, and very importantly, provide for end-to-end integrity verification. Such end-to-end integrity checks can now also be found on disk systems.

In the case of the T10000-C (as well as IBM's Jaguar) drives, data integrity can be ensured by means of a CRC code added to each block written to the tape. For example, this means that if a 512-byte block of data is being written, then it is increased by 8 bytes to 520 bytes. The extra 8 bytes contain the CRC code calculated on the tape drive at the time the block was written. This approach is based on the ANSI T10 data integrity field (DIF) standard developed by a storage consortium.

Prior to DIF technology being available, we were forced to plan to read all files from all cartridges within a period of time, in order to verify integrity via checksum verification. Our digital preservation stack (Tessella's SDB) has the ability to go through all the files of all the AIPs that it manages and request each one to be read out of preservation store and then calculate the checksum using one of several hash algorithms such as MD-5 or SHA-1 in order to verify integrity. Unfortunately, when working with a tape system, this presents a number of problems.

The first issue is that SDB has no understanding of the location of a particular file within a tape library, on a cartridge, and at a particular location of that cartridge. It only really understands POSIX file system paths. We have created a custom storage system to fit into SDB's pluggable storage interface that keeps track of file location based on unique file identifiers that SDB can track as metadata on the AIP. This storage system consists of a database catalog of file locations, along with the ability to read any file or group of files from tape and load them into a convenient disk-based storage system that can be accessed by either SDB or other software.

The second issue that we faced is that SDB could not be expected to request files for validation in an order that would optimize access from tape. Since tape media are inherently sequential access, with typically long latency for first byte access, we could have had a very sub-optimal situation with respect to on-going fixity checks. Our solution to this is to remove from SDB responsibility for thorough fixity checking. Instead, only random fixity checks occur as directed from an operator of SDB. For the thorough checking, we validate the CRC for each block for each cartridge in all our libraries. This is much more efficient than unloading all files from all cartridges and checking fixity. And it is tremendously better than an approach of requesting all files without respect to which cartridge they are on.

Another issue is that SDB is written to expect fairly low latency on any file request. Access to a particular cartridge can often take several seconds, sometimes even minutes. Because of this, the software requesting access to a tape-based file might time out before the operation can complete. To address this, we have written custom workflow operations using the Drools-based workflow engine included in SDB. By doing this, we created an asynchronous interface to the storage system, allowing for operations to be started, that might take a long time to finish.

Since the tape system itself can check the integrity of each data block written to a cartridge, without copying the files out to a disk system, we can perform a level of fixity checking within the storage system that is neither based on the original hash calculation made for each file at ingest time, nor dependent solely on orchestration from the SDB management stack. In other words, we can create policies that execute entirely within the storage system that periodically test the integrity of all the data on each cartridge. And these checks can execute independent from the management system.

The problem, of course, is that there is then no good way to report to SDB that an integrity check was performed, and what files it applied to, and what the results were. Since the storage system also maintains multiple copies of each artifact, it is capable of correcting any errors it detects. Again, however, there is no good way to inform SDB of such action.

Conclusion and Call to Arms

The issues of independent integrity checks, failure recovery, media integrity management and others, are not necessarily unique to tape based storage systems. In fact, Amazon's storage models share all of these characteristics. In the case of Amazon's Glacier, even the high latency for access is common with tape systems, whether or not it is based on tape. What is needed in our preservation industry is a set of clear standards for interfaces between the preservation management layer and storage systems. These interfaces need to allow for such things as asynchronous operation, policy-based actions, storage initiated operations. With such standard interfaces in place, a sophisticated system such as iRODS, not to mention the Family Search tape system, could operate on its policies and report to SDB or Fedora the results of those operations.

Family Search has barely scratched the surface of interaction between increasingly intelligent storage systems and the preservation management software that depends on them. One of the next areas to explore is how to make use of tape diagnostics gathered within the storage system that predict cartridge and drive failures, and communicate that information to the preservation system so that appropriate action can be initiated and recorded.

Author Biography

N. Thomas Creighton received his BS in Computer Science from Brigham Young University (1979) and his MS in Computer Science from University of Southern California (1983). He is currently employed as CTO of Family Search in Salt Lake City, UT. He is responsible for development of large web based applications and the preservation of digital still and moving images as well as large relational databases. He is a member of ACM, SAA, and IASA.