

OpenSMS

Stephen Cranage
StorageTek
Steve_Cranage@stortek.com

Abstract

Systems Managed Storage is a proven concept in traditional mainframe computing. Client-Server operating systems have traditionally lacked the tape I/O subsystem, file system intelligence, and data classification policies required to implement the storage management services that are necessary attributes of a scalable data processing environment. OpenSMS is an Open-source framework that addresses these deficiencies.

1. Introduction

The vision of ILM, or Information Lifecycle Management, has become the topic of great industry attention of late. Rising complexity and cost of storage management has become increasingly apparent. The previous generation of mainframe processing professionals viewed storage management as a systemic task, tied into the base operating system. In the process of migrating applications to new client/server platforms, we have lost some essential operating system services in this area.

While taking advantage of lower cost distributed computers, we have migrated systems that were traditionally departmental and desktop platforms into enterprise computing, and lost sight of how data growth would pose challenges when it occurred on operating systems that were not designed to manage enterprise storage. Situational awareness is now creeping up on the industry as the realization is setting in that scalability and ad-hoc management of storage are mutually exclusive concepts.

Backup processes are a good example. Michael Peterson, Program Director of SNIA's Data Management Forum reports that: [1]

A strong driver exists pushing the revolution to disk-based data protection and it is not cost. Cost is merely an enabler. The driver is IT's urgent need to solve the backup problem. IT has to reduce operating costs and cope with a smaller staff. IT must "stop backing up" to solve this complexity problem. It is the only real way.

The solution requires a fundamental shift in architecture, moving to a simple, transparent operation where redundancy is native to the write process, where data is always there, and even the concept of "restore" goes away.

Making data protection native to the write process, and making "restore go away" is a fundamental function of some traditional Hierarchical Storage Management (HSM) systems, which typically act to duplicate a file object in seconds to minutes after creation or modification, and if well designed, integrate this process into a comprehensive data protection model. Many other aspects of the ILM vision can also be addressed by HSM concepts; these include dealing with regulatory compliance and archiving issues that are not well served by the current crop of fixed content products on the market today

These products use high density disk arrays for storage in spite of the fact that storage comprised of massive arrays of spinning disk spindles is a poor choice for long term data retention due to the perishable nature of the underlying disk technology, and the high cost of maintenance and power for these technologies. The high operating costs combined with rapid technological obsolescence drives the need for routine retirement and replacement of the technology. Unfortunately, increasing array density makes this very problematic. The issue is movement of massive amounts of data off an obsolete platform to the new, and the affect on application availability.

Again, HSM concepts can address the problem. Classification of file objects as they are created can direct the duplication of data to an appropriate tier, including a tier with good archival properties, or good quality of service properties, or both, based on policy directives. In fact, HSM systems are very common in the mainframe environments, where they are an integral component of a larger solution that is referred to as Systems Managed Storage.

Many HSM solutions have come to market for client/server platforms over the years. However none has achieved broad commercial success to the extent of

becoming common in client/server environments, as they did in mainframe computing. There are a number of reasons for this, including poor product focus or reliability or poor hardware choices in implementation, but the principal reason is the technical challenge of operating system integration in a multi-platform, multi-operating system environment.

2. OpenSMS

One of the biggest challenges in creating a HSM product lies in being able to return migrated files back from multi-tier storage in a manner that is transparent to applications. Historically, creating data management applications that intervene within operating system services such as file I/O has been a difficult and expensive process given the proprietary nature of commercial operating systems. As evidence, note the lack of multi-platform products in the HSM space. Vendors who have entered this space have been compelled by intricate dependencies on proprietary kernel code to pick a single operating environment to support.

In the mid 90's, the industry developed an API for data management that was designed to address this set of problems. The Data Management Application Programming Interface (DMAPI), was designed as a standardized set of "hooks" into the file system that would allow data management companies to write HSM software to a standardized file system API [2].

DMAPI interfaces appeared in many file systems on virtually all computing platforms in the years since. Although compatibility between the interfaces is not perfect, dealing with the minor incompatibilities in DMAPI implementations is quite manageable (unlike maintaining compatibility with evolving proprietary operating system internals).

Silicon Graphics (SGI) recently released its high performance XFS file system for Linux under the GNU General Public License[3]. This gave the entire industry access to the source code for a DMAPI enabled file system. Shortly thereafter, IBM followed suit with its JFS file system[4], and based their DMAPI implementation on SGI's implementation.

We seized this opportunity to create a policy-based data mover framework. We called this framework OpenHSM, and published its source code under General Public License (GPL). In order to have something to move archival data to, we took an enterprise Tape Management System (TMS), which StorageTek previously sold as a commercial product called REELlibrarian, and published its source code under the GPL, we call that OpenTMS. Taken together we refer to the two projects as OpenSMS (Open Systems Managed Storage).

2.1.1. Architectural Approach

Our approach is differentiated from traditional HSM products in the way we view storage of file objects in the tape management system (TMS). We view the file system and TMS to be "parallel universes" of data storage. Each has properties that make it better for one storage requirement or another, but each is a storage namespace where file objects are directly accessible, regardless of where they reside.

While the concept of a TMS, or for that matter a tape I/O subsystem by itself, may not be intuitive to everyone, it is well tested in legacy mainframe environments, as well as in legacy UNIX based supercomputing environments.

In the last decade, when large monolithic high performance computers were more common than they are today, there were a few UNIX variants from Cray and others that did include tape I/O subsystems services. These services include device allocation, a mount request system and low level device control. The TMS services (often the very same REELlibrarian code licensed from StorageTek) provide an additional layer of intelligence that includes file cataloging, and media management.

While the TMS is a separate namespace that is in some ways comparable to the file system namespace, it also has some significant differences. The physical properties of sequential access media of course impose some limitations such as single user access and latency. But TMS services also have beneficial properties that make them an indispensable component in creating a scalable archive. Some of the distinguishing characteristics of the TMS namespace include:

- Enterprise wide scope. The TMS client services can satisfy access for a file object anywhere the TMS services are installed. This code is all user level, and has been historically ported very widely. Since TMS access methods don't include multi-user access, this is simpler than a shared file system. Shared SAN based tape transports can be used to distribute large amounts of data at high-sustained transfer rates through the TMS services using channel protocols for the transport without the complexity of a SAN file system.
- Logical vs. hierarchical organization. TMS file objects are stored in a logical container called a "volumeset". The volumeset has the form of `userid/volumeset_name:Vno:Gno`. Vno and Gno refer to version and generation numbers. Within each of these volumesets, is a flat namespace where files are stored and individually cataloged. We have individual

