The Evolution of a Distributed Storage System

Norman Margolus
Permabit, Inc.
One Kendall Square, Bldg. 200
Cambridge, MA 02139
nhm@permabit.com
tel +1-617-995-9331
fax +1-617-252-9977

Abstract

Permabit is a software company that makes a storage clustering product called Permeon. Permeon grew out of the need to reconcile a vision of the future of globally distributed, secure, private and robust storage with near-term marketplace realities. This paper discusses the evolution of the ideas embodied in Permeon.

1 Introduction

When Permabit was founded in June of 2000, it was directed towards making the "disk in the sky" Storage Service Provider (SSP) idea practical. Storage clients would send permanent data off into the network, with the guarantee that this data would be kept safe and private. Most of our initial concerns were interface issues between the storage client and the storage system, and were largely independent of the structure and implementation of the actual storage. It was only later that we focused on distributed storage clustering software.

2 Content Addressed Storage (CAS)

To make remote network storage immediately practical for ordinary users, we had to first address the issue of the availability and cost of bandwidth. We observed that if a cryptographic hash of a block of data is used as the name for the block, then users of a widely shared Internet storage system should be able to backup much of the data on their PC's to a "disk in the sky" without sending much data. Not only could they avoid transmitting data that the storage system has seen before, but we could also avoid storing it separately for each user. This could make it practical for a large number of users to store large amounts of data remotely, enabling data sharing and remote access to their data.

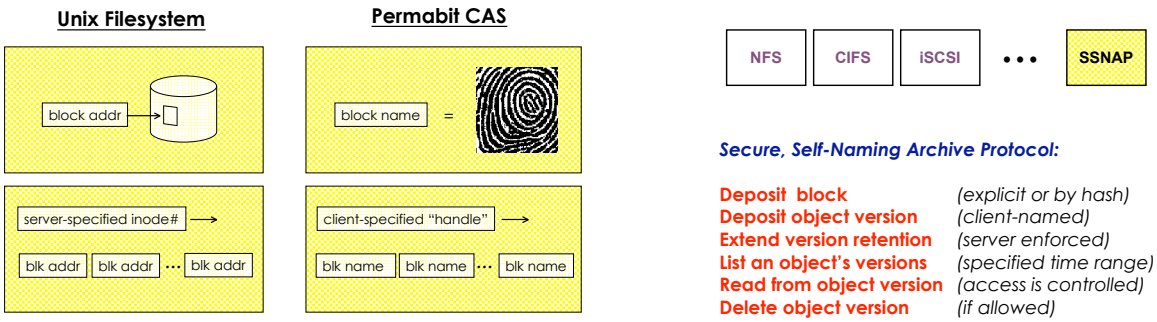

Figure 1: Content addressed storage (CAS) can save valuable bandwidth on the WAN, reduce storage requirements, and guard data integrity. In Permabit CAS, a cryptographic hash *fingerprint* of each data block is used as the address of that block, enabling storage sharing. An unshared metadata-level, analogous to the *inode* level in Unix, controls access and security. A secure network protocol (SSNAP) maintains privacy and thwarts IP-piracy.

2.1 Self-Encryption

CAS bandwidth and storage savings depend on sharing storage between unrelated users. This seems to require that the storage providers have complete access to all of the data. This is something that is unlikely to be palatable to the end user for privacy reasons, and is also a significant potential source of liability to the storage provider.

This issue can be dealt with using self-encryption: each block of data stored can first be encrypted using a key derived deterministically from the unencrypted block (again using a cryptographic hash). Unrelated users will produce the same encrypted block, but no one without the source block can determine the key. As long as keys are never stored "in the clear" in the storage system, no one who has not had access to an unencrypted copy of the block can determine what it contains. The Farsite project[1] uses a similar idea, but only to save storage, not network bandwidth of the depositor.

2.2 Access Control

If knowing the hash of a block is a sufficient credential for being granted access to the block, the storage system has no real access control. For example, in a widely shared "disk in the sky" storage system, the hash corresponding to the contents of a newly released DVD might become widely broadcast. The storage provider needs a way to obey a court-order to remove illegitimate access to this content without removing legitimate access.

It is natural, for this reason, to introduce a metadata level which is analogous to the *inode* level in a Unix file system, as is illustrated in Figure 1. A small amount of unshared per-client and per-object metadata allows conventional access control and provides a place to keep privately-encrypted key information. Only the block-level is globally shared and self-encrypted. Blocks can only be read by reference to unshared per-client metadata.

2.3 Network Protocol

The SSNAP network protocol for communicating with the "disk in the sky" is summarized in Figure 1. Hash-named blocks of data can be deposited by name to save bandwidth, but they can only be read as part of an object which belongs to a particular client (to permit read-access control). Depositing blocks by hash-name may involve a per-client challenge, to ensure that the client actually has the block and not just a hash that someone has broadcast to them.

Objects are analogous to inodes in a Unix file system, except that the *object handle* (inode number) is supplied by the client. Both block names and handles are 32 bytes long. Objects can have multiple historical versions, allowing retention policies for protecting history to be enforced by the storage system. This is used to protect against accidental or malicious corruption. Server enforced record retention is also useful for automating record retention requirements mandated by government regulations and business best practices.

2.4 Security and Privacy

No attempt is made to anonymize user access. Privacy maintenance is based on not storing records that allow a stored object to be linked to a user when the object is not actually being accessed. Depending upon accounting requirements, whatever information is desired can be logged or not logged at access time. By not recording who owns what data, storage service providers can avoid having these (non-existent) records subpoenaed.

The globally unique name of an object is obtained by combining a namespace identifier with the handle constructed by the client—which the client ensures is unique within the namespace and unguessable. The handle and namespace are combined by the storage system using a one-way hash function to locate the object. To prevent objects that are not being accessed from being linked to users, handles are not retained by the storage system.

2.5 Embedding File Systems

SSNAP provides an object level interface at the equivalent of the Unix inode level. Client libraries build upon this, to support file sharing protocols with directories. A generalized *Portable Hierarchical File System* (PHFS) data format is defined, with an associated API, which allows file system metadata for different file systems to be embedded. Multiple historical versions of files are supported, to allow convenient file system snapshotting for backup purposes. File system snapshots can be copied into Permabit storage and presented through familiar file sharing protocols. Permabit storage can also be used directly for live file storage, with copy-on-write snapshotting providing backup. By breaking files up into hash-named blocks at natural boundaries (e.g., email attachment boundaries), the likelihood of storage sharing (block coalescence) is enhanced.

2.6 Enterprise Storage

Although SSNAP and PHFS were originally conceived with end-users in mind, most of the considerations that went into their design are at least as important for enterprises. Divisions within an enterprise would like to be able to share storage without compromising their

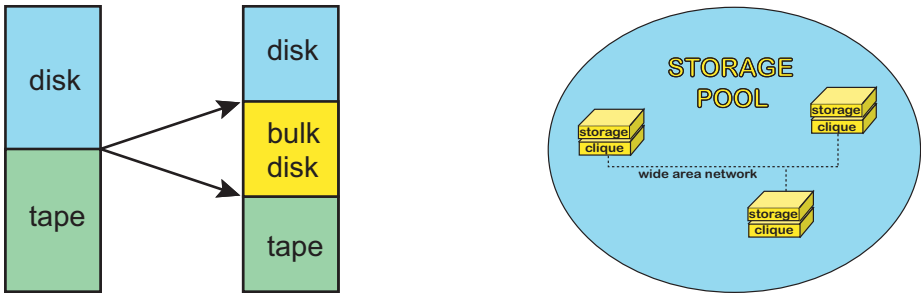

Figure 2: Distributed storage systems can displace tape as safe and economical long-term/large-scale storage. Geographically distributed storage cliques can protect redundant records from loss or corruption, and independently enforce record retention policies.

privacy or security. Storage-system enforced data retention is important in this context not only for preserving backup snapshots of a file system, but also for meeting government regulatory mandates for the long-term retention of email and other sensitive records.

The elimination of common storage is particularly interesting for enterprises. The Venti project at Lucent Bell Labs[2] showed that, with the combination of avoiding duplication of common blocks of data and compressing the blocks, all of the daily backup data for each of two large shared file systems *accumulated over a decade* could be stored on disk using only slighly more storage than the current active data on the file systems.

3 Distributed Storage

The considerations discussed above apply regardless of the nature of the "disk in the sky." Since hash-named data is a natural match for scalable clustered storage, and stimulated by the low cost of ATA disks and commodity PC's, we decided to develop our own storage clustering software.

3.1 Bulk Storage

Storage systems are naturally hierarchical. For example, RAM memory systems use slower but more economical memory for bulk storage, while caching the most active data to hide most of the latency. With the raw media cost per gigabyte of ATA disks below that of tape, it seems natural to expect that the bulk of storage will soon be low-cost disk storage, used within a storage hierarchy (Figure 2). But low cost does not mean just cheap disks. Most of the cost of disk storage today is in its management: adding and removing storage, keeping it working, backing it up and preparing for disaster recovery. Bulk disk storage needs to be self-managing, self-healing, self-backing and disaster tolerant in order to qualify as low cost storage.

To be disaster tolerant, the bulk disk storage must be distributed: information destroyed at one location must be redundantly represented elsewhere. In considering costs, however, it is important to also keep bandwidth in mind. The cost of bandwidth on the WAN is several orders of magnitude greater than the cost of bandwidth on a local network. This makes a two level system attractive, in which local clustering of economical standard hardware

constitutes the first level. In the Permeon system the local cluster provides fast recovery after failure of a server, fast access for nearby clients, high aggregate disk and network bandwidth, and cooperative caching behavior.

3.2 Scalability

Overlay network routing schemes developed for academic distributed storage systems depend on statistical allocation of storage capacity. This does not work well for small clusters. Permeon *cliques* are designed to scale up indefinitely starting from a very small number of servers, allowing a low-cost entry point. For this reason, a table-based routing scheme is used within cliques, in which address ranges for hash-named data (and replicas) are assigned and reassigned as servers are added or removed. Data replicas are created and moved automatically as address assignments change.

Because of high WAN bandwidth costs, if a clique is destroyed there is a large benefit in being able to recreate it using data resident at one or a small number of locations. This leads to a multi-clique scheme in which the data from each clique is redundantly represented at some small set of other cliques.

3.3 Portals

To provide convenient access to the distributed storage system, Permeon provides portal servers which present the clique storage using familiar file sharing protocols. These portals can live behind firewalls, providing secure access to shared storage cliques. A web-based storage management console runs on the portals.

4 Conclusions

The distributed "disk in the sky" storage systems of the future will be the descendents of decentralized bulk-disk storage of the post-magnetic-tape era which make storage inexpensive by solving the costly storage management, repair, data corruption and disaster tolerance problems. In particular, these storage systems will have to enforce data-retention policies, and do so independently in different locations, in order to provide useful backup and dependable archiving. Systems which save bandwidth and storage space by avoiding transmitting or storing duplicate data have a distinct advantage in the near term. Privacy and security provisions are already important today, and will be vital to making the systems suitable for widely shared access.

References

[1] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," ICDCS, July 2002.

[2] Sean Quinlan and Sean Dorward, "Venti: a new approach to archival storage," *First USENIX conference on File and Storage Technologies,* 2002.