Object Storage: The Future Building Block for Storage Systems

A Position Paper

Michael Factor Kalman Meth Dalit Naor Ohad Rodeh
Julian Satran

IBM Haifa Research Laboratories
$\{$*factor, meth, dalit, orodeh, satran*$\}$*@il.ibm.com*

Abstract

The concept of object storage was introduced in the early 1990's by the research community. Since then it has greatly matured and is now in its early stages of adoption by the industry. Yet, object storage is still not widely accepted. Viewing object store technology as the future building block, particularly for large storage systems, our team in IBM Haifa Research Lab has invested substantial efforts in this area.

In this position paper we survey the latest developments in the area of object store technology, focusing on standardization, research prototypes, and technology adoption and deployment. A major step has been the approval of the T10 OSD protocol (version 1) as an OSD standard in late 2004. We also report on prototyping efforts that are carried out in IBM Haifa Research Lab in building an object store. Our latest prototype is compliant with a large subset of the T10 standard. To facilitate deployment of the new technology and protocol in the community at large, our team also implemented a T10-compliant OSD (iSCSI) initiator for Linux. The initiator is interoperable with object disks of other vendors. The initiator will be available as an open source driver for Linux.

1. Object Store in a Nutshell

An object store (ObS) or object storage device (OSD) enables the creation of self-managed, shared and secure storage for storage networks. This moves lower-level functionalities such as space management into the storage device itself, accessing the device through a standard object interface [12].

An object store (ObS) raises the level of abstraction presented by today's block devices. Instead of presenting the abstraction of a logical array of unrelated blocks, addressed by their index in the array (i.e., their Logical Block Address (LBA)), an object store appears as a collection of objects. An individual object is a container of storage (object-data and object-meta-data) exposing an interface similar to a file, presenting the abstraction of a sparsely allocated array of bytes indexed from zero to infinity.

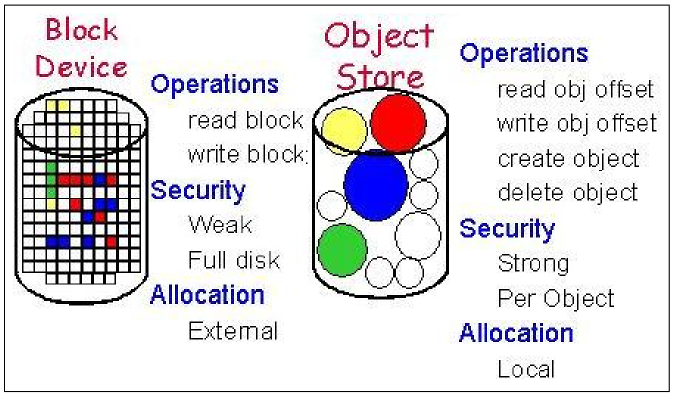

Figure 1. A block device vs. an object storage device (OSD)

In an object store environment space is allocated by the object store and not by the higher level software such as a file system. Users of an object store, e.g., the file system, operate on data by performing operations such as creating an object, reading/writing at a logical location in the object, and deleting the object. In addition, all operations carry a credential, and it is the responsibility of the object store to validate that the user's request carries a valid credential. This per-object credential allows the storage to enforce different access rights for different portions of a volume on an object granularity. A schematic picture of an object store is depicted in Figure 1.

1.1. Main benefits of Object Storage

1. An object store allows secure non-mediated and shared access to centrally-managed storage. This feature is particularly important in a SAN architecture which allows multiple data servers to access shared storage over the SAN. However, today's technology is limited in the amount of data sharing it provides and relies upon an underlying file server to mediate access to the underlying storage. Space allocation and access control are still performed by the host. In contrast, the delegation of space allocation along with access control and security enforcement to the storage device removes the centralized file server from the critical path, thus achieving the desired direct access to the data by a client over the SAN.

2. An object store allows a more rapid adoption of IP-based SAN (e.g., based on iSCSI [9]) instead of fibre channel based ones. IP networks are cheaper, open (non glass house), widely used and easier to manage. As such, they suffer from numerous hacking attempts and are considered more vulnerable than fibre channel networks. For this reason IP-based SANs require a tighter level of security both at the access control level and at the network level. The security enforcement provided by object stores, where every command is accompanied by a cryptographically secure capability, is an essential building block to secure a SAN environment.

3. An object is a storage container both for customer-data and attributes (including user-defined attributes). In an object store, meta-data is an integral part of the object, managed, stored persistently and recoverable with the object's customer-data. This built-in mechanism at the storage level considerably reduces the efforts required to manage and store meta-data in a distributed application along with the customer data.

4. We expect future versions of OSDs to provide advanced functionalities. Such functionalities include support for collections of objects, multi-object operations, snapshots, cloning, and space-management. When implemented close to the object device such functionalities can distribute the work that today takes place in the centralized file-server, resulting in improved performance and simplified management.

1.2. Difficulties with Paradigm Shift

Object Storage requires a paradigm shift. It changes a very basic and low level API in the system stack: hosts no longer access blocks of data on disk, rather, they refer to object id's and offsets within objects. Moreover, space management is no longer handled by the host and thus requires a change in the client accessing the storage device. This shift is probably the main barrier for wider acceptance of the technology.

Another difficulty stems from the fact that its benefits are most notable in large or highly distributed and scalable storage systems. This makes technology adoption even slower, since at the moment it pays only for high-end applications. However, we believe this is likely to change in the future as networked storage solutions become more prevalent with the adoption of iSCSI. Security and advanced copy functions are likely to benefit even smaller scale systems.

2. Object storage in the Industry

Object storage was first proposed in CMU as an academic research project [5, 6] and is still an active area of research in academia. As for the industry, while there is significant industry involvement in the standardization effort for object storage, the actual adoption is low and most current object stores use proprietary interfaces.

Panasas has a file-system based on object stores; the file-system is called *PanFS*[10] . PanFS is comprised of a network, meta-data servers, object stores, and clients. The clients talk directly to the OSDs when they need file-data, and the meta-data servers manage the out-of-band operations. While this object store is not currently standard compliant, Panasas is actively involved in the standardization effort and has expressed the intent of moving toward the standard object store.

Lustre [8, 2] is an open source file system developed by Cluster File Systems Inc. and marketed by HP. It aims to be a highly scalable SAN file system. It is built out of clients, cluster control system, and storage-targets (OSTs). The market space which uses Lustre is high performance computing. Lustre uses a proprietary object store, OST (Object Store Target), which aims to provide very high performance. The cluster control system manages the namespace, file system meta-data coherence, security, cluster recovery, and storage management. It does not handle file data. The OSTs store all persistent data. While acknowledging the importance of security, initial implementations of OST to not provide security for the operations against the object store, a key difference from the standard protocol.

EMC's Centera [3] may be considered a restricted form of an object store, supporting only a limited set of the operations provided in the T10 standard and doing so in a proprietary way. Centera, however, is not aimed at solving the general set of network storage problems which the standard object store aims to solves; rather, Centera is marketed as content addressable storage aimed at storing reference data. A Centera box contains a set of persistent objects, where objects are much like files in a file-system. An object name is, essentially, a hash of its contents. This type of addressing is geared toward immutable objects. Centera is intended for use as part of a compliance product; one that implements a new set of strict requirements for storage issued by governments in Europe and America.

3. OSD Standardization

Standardization is vital for the success of this technology, due to the fundamental changes it requires in host systems. For this reason, major efforts have been invested by our team and others in developing the standard and bringing it to acceptance. Our team also implemented this standard along with a testing suite, exhibited an object store that is interoperable with OSDs of other vendors, and is about to release an open-source iSCSI OSD Initiator for Linux that is compliant with the standard.

3.1. OSD T10 Standard

The first standardization effort [13, 12] of an Object Storage Device specification is embodied over the SCSI protocol and is being realized as a new set of SCSI commands. Version 1 of the T10 standard was publicly reviewed and approved in late 2004; OSD standard has been published as ANSI INCITS 400-2004 [13]. Many companies were involved and contributed to the standard. Among them are: EMC, HP, IBM, Intel, Panasas, Seagate, Veritas. Along with active engagement with the standard development, one of this paper's co-authors (Julian Satran) has co-chaired the SNIA OSD working group.

An important aspect of the OSDv1 (OSD-version 1) standard is its security model. The standard defines a set of OSD commands and specifies the security protocol that accompanies every object store command. The security protocol is based on a cryptographically secured capability (called a *credential*). Credentials are issued by the Security/Policy Admin, a trusted entity, which is assumed to enforce a given policy. Presently, the standard is being extended to version 2. Extensions include advanced functions such as snapshots, multi-object operations, collections, and error handling.

A large subset of the OSDv1 functionality was implemented by our group in IBM Research, both at the OSD target and the OSD initiator. Our group's OSD initiator for Linux is compliant with the T10 standard and will be made available as open source.

3.2. pNFS extension for NFSv4

Another standardization effort that is relevant to object storage and its ease of deployment is the pNFS extension to NFSv4 [14].

Large file repositories with scalable capacity and access bandwidth are now available in the industry. This was enabled by (1) the advent of SAN and of SAN based file systems, (2) the aggregation of file-server data in ”enterprise-wide” file-server solutions, and (3) prevalence and speed of the networking infrastructure. Access bandwidth scalability is achieved by segregating metadata access from data access. Clients of such repositories access a central server in order to obtain the file data location(s) and access the underlying storage directly for data. Every major storage vendor presently has at least one product operating on similar principles. Unfortunately, all those products are proprietary. Users have to buy the software and/or hardware for a repository control entity (the metadata server) and have to deploy proprietary client software on all the machines that use the data (clients in file-system terms). Those solutions are expensive to buy and expensive to maintain; client software has to be available for all the major user platforms - Windows, Linux, AIX, Solaris etc. Lately, a group of technologists representing all major storage vendors as well as Academia have been working towards standardizing an extension to NFSv4 that governs the way clients get their location information from metadata servers. This work is being done within IETF - the NFSv4 Working Group, and is popularly known as pNFS . pNFS covers the interactions between clients and metadata NFSv4 servers for the three major types of data storage: block, object and file, based on extension to the delegation mechanism of NFSv4. With this standard in place it will be possible to build large file repositories with standard clients.

4. IBM prototypes for Object Stores

IBM Research is active in the research and development of object storage, and its potential uses. Activities at IBM Haifa research lab are surveyed below.

4.1. Related projects at IBM HRL

An earlier work developed at IBM Haifa Labs that predates object storage is *DSF* (Data Sharing Facility). DSF [4] was an experimental project to build a serverless file system that distributes all aspects of file and storage management over cooperating machines interconnected by a fast switched network. In the same spirit of object storage, the DSF architecture delegates space allocation to the storage subsystem.

This work was followed by *Antara* [1], IBM's first proof-of-concept for a stand-alone implementation of an object store. Antara is a pure software implementation which implemented a proprietary protocol (as it predated the T10 standard). Antara's implementation and protocol was used in *zFS*, another research project carried out in IBM. *zFS* [11] – a scalable distributed file system using object stores – extends the research done in the DSF project by using an object store as storage media and by using leases and distributed transactions.

4.2. ObjectStone

ObjectStone is our prototype of an object based controller. It was originally designed to provide Storage Device capabilities with a block-based storage controller, in particular IBM's SAN Volume Controller (*SVC* [7]). Later versions of ObjectStone run on a standard Linux server, maintaining its controller characteristics.

As an object-based controller, ObjectStone aims to combine the world of object storage with that of controllers – allowing high-performance, highly-available object-related data to hosts. In ObjectStone, the object store abstraction is implemented entirely on top of the block-based storage controller. It uses the underlying controller as a reliable, high performance fast-write cache both for the customer data and for the meta data of the object abstraction.

A major feature of ObjectStone is that its front end interface implements the standard OSD SCSI T10 protocol. It uses iSCSI for its SCSI transport. It implements an iSCSI target in software that supports the extensions to iSCSI that are needed for OSD commands. Along with ObjectStone, our team developed an OSD simulator that runs on a Linux machine and is implemented over the local file system. This OSD simulator shares the same front-end as ObjectStone, namely the T10 protocol over iSCSI. It serves as a useful tool, both for testing purposes and to simulate an OSD standard interface for a higher-level application built on top of standard OSDs.

Figure 2 demonstrates the various components of the object store project. At the target side, there are two possible OSD targets: OC (an object store controller) and the OSD simulator. Both targets share the same two front-end software modules: an iSCSI target and a T10 front-end that encapsulates and interprets T10 OSD SCSI commands. At the host side, our OSD initiator runs on top of an extended iSCSI initiator. OSD commands between the host and the target are transported as SCSI commands over an IP network.

Preliminary performance results show that our ObjectStone target achieves a throughput of 109 MB/sec for all-cache-hits read operations on a 1Gbit ethernet switch. The equivalent raw tcp/ip performance on this network was 112 MB/sec. This read performance is achieved when the I/O size of a single read operation is 16KB or more. As for performance of write operations, non-allocating writes (i.e. rewrites) achieve the maximum performance of 105 MB/sec when the I/O size of a single write operation is 32KB.

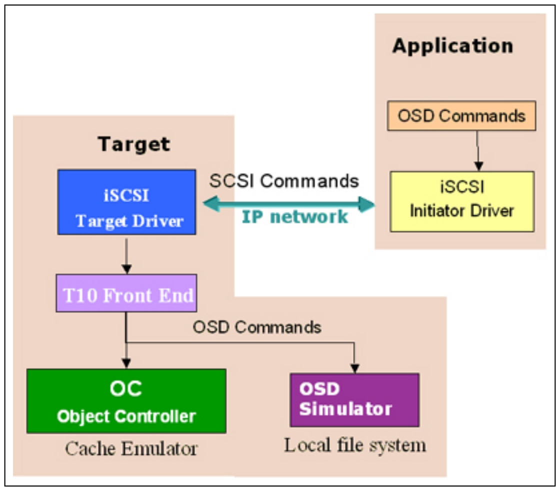

Figure 2. ObjectStone components

4.3. OSD iSCSI Initiator for Linux

As mentioned earlier, a major barrier to deployment of this new technology is that it requires a different driver to drive I/O commands between the host and the object store. To cope with this difficulty, our team developed an iSCSI OSD Initiator for Linux. This OSD driver will be released as open source to the Linux community. We strongly believe that it will serve as a vehicle to drive the technology to a wider section of the community and ease its deployment.

Our OSD initiator is built on top of the Linux SCSI stack. It is composed of two components (1) a generic T10 OSD component and (2) an iSCSI component. The generic T10 OSD component is independent of the SCSI transport. Its objective is to construct and parse OSD SCSI commands according to the T10 standard protocol. This includes implementing the security aspect of the OSD protocol (such as building the credential, signing the command, handling its security-related errors). It also runs on top of Fibre Channel.

Developing an iSCSI component for the OSD initiator was needed since OSD SCSI commands are different from typical SCSI block commands in that they may be bi-directional and they require extended CDBs (Command Descriptor Blocks). This places a requirement on the SCSI transport to support extended and bi-directional commands. Since there is little need in most existing storage environments for bi-directional commands and extended

CDBs most current SCSI drivers and adapters do not support them. Such support is needed on client hosts (initiators) as well as on the target ObjectStone. The simplest way to obtain SCSI driver support for extended CDBs and bi-directional commands was to implement these features ourselves in software over iSCSI.

4.4. Interoperability

In April of 2005, at Storage Networking World conference (SNW) Spring 2005, IBM, Seagate and Emulex exhibited an object-storage technology demonstration. This demo showed IBM application servers and metadata servers running an IBM Research shared file system with Object-Based Storage Device (OSD) support. The integration of the shared file system with OSD devices was developed by a team at the IBM Almaden Labs. This file system demonstrates interoperability in two ways:

- Multiple vendors (IBM, Seagate and Emulex) have designed this system based on the SNIA/T10 OSD standard.

- The shared file system is simultaneously accessing object storage devices over Fibre Channel and Ethernet storage area networks, showing ability to support this new protocol over multiple SCSI transports.

In this demonstration, our ObjectStone target implementation was shown to be compatible with another standard T10 device on a major subset of the OSDv1 protocol. Interoperability was achieved by using software components developed by our team: The OSD initiator, as well as a testing suite and a T10 compliant Tester user-mode application.

5. Summary and Acknowledgements

Despite its great potential, object storage is being adopted at a slow rate. Our team at IBM Haifa Research Labs has dedicated significant effort in advancing this technology, both implementation efforts (internal for IBM and for the community at large) and standardization efforts.

Special thanks to the ObjectStone team at IBM Haifa Research Lab. This team is responsible for all object storage related activities: Guy Laden, Petra Reshef, Liran Schour, Itai Segall, Allon Shafrir, Paula Ta-Shma, Adam Wolman and Eitan Yaffe. We would like to thank our colleagues from the IBM Almaden Research Lab: Ralph Becker-Szendy, Jody Glider, Richard Golding, Deepak Kenchammana-Hosekote and Miri Sivan-zimet, for collaborating on object storage issues and object-based SAN file system.

References

[1] A. Azagury, V. Dreizin, M. Factor, E. Henis, D. Naor, N. Rinetzky, O. Rodeh, J. Satran, A. Tavory, and L. Yerushalmi. Towards an object store. In *The 20th IEEE Symposium on Mass Storage Systems, 2003*, pages 165–, 2003.

[2] P. J. Braam. The Lustre storage architecture. Technical report, Cluster File Systems, Inc., 2002. http://www.lustre.org/docs/lustre.pdf.

[3] EMC Centera, content addressed storage, product description. http://www.emc.com/pdf/products/centera/-centera_guide.pdf, 2002.

[4] Z. Dubitzky, I. Gold, E. Henis, J. Satran, and D. Sheinwald. DSF: Data sharing facility. Technical report, IBM Haifa Research Labs, 2002.

[5] G. Gibson, D. Nagle, K. Amiri, F. Chang, E. Feinberg, H. Gobioff, C. Lee, B. Ozceri, E. Riedel, D. Rochberg, and J. Zelenka. File server scaling with network-attached secure disks. *Proceedings of the ACM International Conference on Measurement and Modelling of Computer System, Seattle, WA*, June 1996.

[6] G. Gibson, D. Nagle, K. Amiri, F. Chang, H. Gobioff, E. Riedel, D. Rochberg, and J. Zelenka. Filesystems for network-attached secure disks. Technical Report CMU–CS–97–112, CMU, 1997.

[7] J. S. Glider, C. F. Fuente, and W. J. Scales. The software architecture of a san storage control system. *IBM Systems Journal*, 42(2):232–249, 2003.

[8] Lustre: A scalable, high performance file system. http://www.lustre.org/docs/whitepaper.pdf.

[9] K. Z. Meth and J. Satran. Features of the iscsi protocol. *IEEE Communications Magazine*, 41(8):232–249, August 2003.

[10] Panfs: Object-based architecture. http://www.panasas.com/panfs.html.

[11] O. Rodeh and A. Teperman. zfs - a scalable distributed file system using object disks. In *IEEE Symposium on Mass Storage Systems*, pages 207–218, 2003.

[12] SNIA - Storage Networking Industry Association. *OSD: Object Based Storage Devices Technical Work Group*. http://www.snia.org/tech_activities/workgroups/osd/.

[13] R. O. Weber. *SCSI Object-Based Storage Device Commands (OSD), Document Number: ANSI/INCITS 400-2004*. InterNational Committee for Information Technology Standards (formerly NCITS), December 2004. http://www.t10.org/drafts.htm.

[14] B. Welch, B. Halevy, D. Black, A. Adamson, and D. Noveck. *pNFS Operations Summary, Internet Draft draft-welch-pnfs-ops-00.txt*. IETF, October 2004.