SNIA OSD TWG document cross posted to T10

ObS - version.09

Julian Satran
Michael Factor
Alain Azagury
Ealan Henis
IBM

Erik Ridel
Sami Iren
Seagate

Mikail Zelnikov
George Ericson
EMC

Zahir Rumi
Michael Mesnier
Intel

Garth Gibson
David Nagle
Panasas

ObS - Identifying Objects - 0.9

Abstract

Object Storage (ObS) is a new storage paradigm. While conventional storage devices (block storage) store information in blocks that are accessed based on the block address within a device with Object Storage different pieces of data belonging to one logical entity are accessed based on an object identifier (OID) and the relative position of the data within the object. With ObS access to the data can be controlled through credentials and capabilities.

As objects represent larger data pieces and have associated semantic beyond storage we have to address carefully the way objects are identified.

This memo is an attempt to outline the issues related to object identifications and several possible solutions

Acknowledgements

Log of Changes

The following changes where made from version 0.1 to 0.2

a) Committed/Uncommitted made an attribute (or group?) instead of a basic function
b) Local/global names - removed - suggest making it another attribute (not final)

The following changes where made from version 0.2 to 0.3

a) Added clarification on requirement to support simple migration of objects
b) Added a requirement for an "external name" (textual name) as an attribute

The following changes where made from version 0.3 to 0.4

a) Completely restructured text
b) Added examples
c) A different (to be discussed) attempt at versions

The following changes where made from version 0.4 to 0.5

a) Added ObS generating names
b) Restored version 3 versioning

The following changes where made from version 0.5 to version 0.6

a) Added full branching capability to version
b) Added version creation option to write
c) Fixed text
d) Added names for Create operations and made Create-and-Assign multiple explicit
e) Added an acronyms and definitions section
f) more comprehensive version semantic including COW and branching

The following changes where made from version 0.6 to version 0.7

a) Name structure - names universal names formed as a concatenation of ObS Identifier, Partition Identifier and Object ID - this is the Fully Qualified Object ID or FQOID
b) Name structure and usage based on the Gibson, Nagle and others paper - "File systems for NASD - CMU-CS-1997-118" with a more detailed discussion around name usage
c) Another attempt at versioning

The following changes where made from version 0.7 to version 0.8

a) partition IDs made 64 bit not extensible
b) clarified open issues in 0.7

The following changes where made from version 0.8 to version 0.9

a) editorial

1. Acronyms and definitions

ObSID - a variable format name of an Object Store (approximately equivalent to the world-wide-unique-name of a "classical" disk volume). This name is world-wide-unique

PtID - Partition ID - A 32-bit bitstring assigned as an identifier to a grouping of objects that will share space an security characteristics (key, hash function). PtID is unique within an ObS.

OID - Object ID - a 64 bit bitstring that uniquely identifies an object within a partition.

FQOID - Fully Qualified Object ID - Word Wide unique Object ID - formed by concatenating ObSID, PtID and OID

WKOID - Well Known Object ID - an OID for an object with a predefined and standardized usage

WKPtID - Well Known Partition ID - a PtID for a partition with a predefined and standardized usage

VID - Version ID - a 64 bit bitstring that concatenated to PtID and OID names uniquely a version of an object within and ObSID.

2. Requirements

It should be possible to logically divide the disk space between large sets of objects (partitions). Migrating objects between Object Stores (ObS) for entire partitions should be performed with ease without host computer involvement (peer-to-peer operation). The way objects are identified must not make migration difficult nor should it result in a mandatory host involvement in the migration process. Whether this migration is controlled on the primary (data access) interface or through some management interface is subject to further study.

Object within partitions share a security mode (for now a key but perhaps different security protocols in the future) and a preset share of the total storage space of an Object Store.

Data migration is always associated with some catalogue operation. When migrating large collections it should not be necessary to change every catalogue record for the individual objects.

Object name assignment and removal should require as little "distributed transaction" mechanism as possible from a cataloguing entity.

The ObS name space should enable easy object access and a fixed-length binary name is most probably adequate as an access parameter. An optional attribute including some form of external name is probably beneficial for recovery. However ObS will not have to access efficiently objects based on textual names.

The tuple (ObSID, PtID, OID) identifies at most one object stored in an Object Store world-wide. However no such requirement is imposed on objects stored on some "external media" (e.g, print, tape, archives on WORM etc.).

A subset of the object naming space should be reserved for well-known-objects.

The naming scheme may require the existence of specific attributes such as the life-span of an object (temporary, non-committed, permanent).

The Object Identifiers should not require the presence of a new long lived naming authority. Object creation should be possible without invoking a naming authority.

Including some form of version might be appropriate for many applications. However version requirements are not completely clear and somewhat conflicting. For most applications some form of ordering of versions is essential. However total ordering might be impractical even for moderately sized distributed storage and temporal ordering might be good enough for most applications. At least a simple form of versioning supporting copy-on-write should be provided in a future version.

3. Architectural assumptions

We assume a three party structure:

i) Administrator (Admin) - dispenses credentials and records/retrieves object related information. There can be more than one Admin administering an ObS and requiring some coordination between them if admissible while we will strive to avoid excessive coordination.

ii) Client (Client) creates and deletes objects and accesses object data (reads, writes, appends etc.) with credentials obtained from the administrator

iii) Object Storage (ObS)- performs the operation requested by the client after checking credentials

We will also assume that Admin and Client can communicate and so do Client and ObS but Administrator and ObS do not necessarily have to have a direct communications channel during regular operations.

4. Proposed structure for ObSID

For ObSID we use an type-prefixed bit string with the following structure:

i) 16 bit code-type prefix
ii) 128 bit ID

The type prefix defines the structure and composition of the 128 bit ID. Whenever the coded value is less than 128 bit it is right extended with 0's.

The coded prefix values defined are:

0x0000-0x0001	Reserved
0x0002	EUI-64 right extended with 0
0x0003	IEEE NAA
0x0004-0x007F	ObS-Reserved
0x0080-0x00FF	Vendor Specific Formats registered with T10
0x0100-0xFFFF	ObS-Reserved

5. Proposed structure for PtID

For PtID we use a 64 bit string.

PtID for user defined partitions be 0x0000000000010000 to oxFFFFFFFFFFFFFFFF.

The range 0x0000000000000000-0x000000000000FFFF is reserved for WKPtID. In particular PtID 0x0000000000000000 is used to denote the whole ObSID.

WKPtID will be registered with T10.

6. Proposed structure for OID

For OID we use a 64 bit string.

The range 0x0000000000000000-0x000000000000FFFF is reserved for WKOIDs. In particular OID 0x0000000000000000 is used for an object that describes the partition.

WKOID will be registered with T10

7. Use of identifiers in object creation, deletion and access

Creating an object involves Admin, Client and ObS. The Client has to obtain a credential to create (and perhaps access) an object including all the right capabilities for the operations involved. Credentials have limited life-span (a client may have to renew them). Credentials are attached to every operation.

An object creation call will be like:

Response = Execute(Target, ObS-Selector, Operation, PtID+OID, Number-of-objects, Credential, OutBuffer)

The target includes a set of ObS entities. ObS selects a specific device (volume or LU in SCSI lingo) with the ObS-Selector. It implicitly selects the ObSID.

In SCSI the Operation, OID, Number-of-objects and Credential may be included in a CDB like construct.

The Operation may be Create (in which case the OID is specified by the host) or Create-and-Assign (in which case the OID(s) are assigned by the ObS. The Create-and-Assign operation may specify the number of objects to be created. The Create operation specifies always a single object whose OID is specified during the operation.

The partition in which the objects are created must exist at the time the create or create-and-assign is executed.

7.1 Creating objects with client supplied IDs

To create an object the Client will interact with the Admin to obtain the credential required to create and write the object and with the ObS to actually create and write the object as follows:

a) Client->Admin ask for credential to create a new global object
b) Admin->Client credential (or reject)
c) Client->ObS create object OID
d) ObS->Client OK (or reject if the OID is out of the range or an object with the same OID was already created)
e) Client->ObS write data on object
f) ....

Please observe that the OID could be selected by the Admin or the Client but this selection must be done before the Admin can issue the credential.

7.2 Creating objects with ObS assigned IDs

The operation create and assign specifies the number OIDs to create The resulting OIDs can be returned in the OutBuffer.

Creating objects with ObS assigned OIDs involves Admin, Client and ObS.

The Admin must enable the principal to create objects within a specified partition and ObS must be able to ascertain that the credential is not used more than once.

To create objects with ObS assigned OID the Client will interact with the Admin to obtain the credential required create objects with ObS assigned OIDs and with the ObS to actually create the objects as follows:

a) Client->Admin ask for credential to create new objects within a specified partition (the partition must exist at creation time)
b) Admin->Client credential (or reject)
c) Client->ObS create objects and assign OIDs
d) ObS->Client OK (or reject if the credential is old or used already)
e) Client->Admin ask for credential to write/read OID
f) Admin->Client credential (or reject)

g) ....

Please note that in step d ObS must reject the create request if the credential was used already even if the created objects do not exist on the device any more (e.g., have been deleted). This type of credential has to be very short lived (as it has to be remembered by the ObS until it expires.

In step c the client specifies an existing PtID and OID 0.

7.3 Creating/Deleting Partitions

Partitions are created and deleted with special operations.

7.4 Object deletion

Deleting an object involves Admin, Client and ObS. The Client has to obtain a credential to delete the object from the Admin. It may then request object deletion.

7.5 Object Access

Accessing an object involves Admin, Client and ObS. The Client has to obtain a credential to access an object including all the right capabilities for the operations involved. Credentials have limited life-span (a client may have to renew them). Credentials are attached to every operation.

An object access call (remote procedure call) will be like:

Response = Access(Target, ObS-Selector, Operation, PtID+OID, Credential, Offset, Length, IN-Buffer, OUT-Buffer)

The target includes a set of ObS entities. ObS selects a specific device (volume or LU in SCSI lingo) with the ObS-Selector.

In SCSI the Operation, OID, Credential may be included in a CDB like structure.

8. Temporary vs. Permanent objects

Although object identifiers will not be immediately reused (except for WKOIDs) applications could benefit from having ObS perform

garbage collection on objects not properly recorded or not recorded at all.

To distinguish a temporary from a permanent object we require the attribute set to include a "committed"/"uncommitted" attribute.

ObS will be able to remove all "uncommitted" objects or a subset of them based on additional attributes.

The default value of this attribute is committed.

9. Additional security considerations

Different schemes of identifying objects have to be compared also based on their resilience to different attacks and the costs associated with achieving the required resilience. Of a major concern is assessing the possible harm done by a malicious client because clients are the only entities on which one imposes only weak security requirements (that it can identify a principal and the principal trusts it to act on its behalf).

The attacks we are concerned about are:

a) - client floods the ObS with object creation requests - a short term or long term DOS attack
b) - the client is not candid about the object creation results

To protect against a), any scheme that let to ObS create the OID at creation time must limit the validity of the create credential to one operation and that is costly. However any scheme in which the Admin is the one that hands out the OIDs is inherently safer against this type of attack.

The b) type of attack although more intricate may have an Admin record as created objects that do not exist indirectly limit it’s ability to function. Any scheme in which the OIDs are generated by the ObS handles better this type of attack since the malicious client can’t generate OIDs faster that a good client. To protect against this type of attack without adversely affecting legitimate clients and Admin may "measure" the rate at which object can be created by ObS and limit the number of credentials it hands out based on this rate. In addition we suggest having the OIDs protected with a MAC based on the partition key and include the creation credential.

10. Future work on Versioning (not for consideration for the current version of this document)

10.1 Version

The version support will be optional. However, if implemented it will include all the functions described here.

We suggest an 32 bit version-ID assigned by the ObS. Version is changed by either a write command with an option of create new version or specific operations on whole objects that include creation of a new version as part of their function.

For content it may be supplemented by a content cryptographic signature including some metadata.

Every versioned object will include the name of the previous version to enable version backtracking.

On an ObS only the "latest" version of an object is writable (changeable).

An object replica will include a reference to the FQOID and VID of he replicated object version. Replicas can be created only from non-changeable versions of an object. However replicas have different names than the replicated Object and start a new versioning branch. In summary the name & version information in an object is a structure containing: {FQOID,VID, Previous-VID, Replicated{OID, VID}}

10.2 Other operations that affect name or version

Some standardized or common administrative operations affect name and version.

We will discuss briefly some of those operations. Please note that although we are discussing those operations it is not our intent to standardize all or part of those operations in the first version of the standard.

10.2.1 Regular access operations

10.2.1.1 Write with the copy-on-write (COW) flag

A write operation on an object flagged as COW or a write operation with the COW flag raised in the operation will result in the creation of a new version of the object. The new version is the default version while the previous version is "frozen" (cannot be changed but can be replicated).

10.2.2 Operations on whole objects

10.2.2.1 Create a a new version of the object

This operation results in the creation of a new version of a object (default) while the original becomes unchangeable. This operation is equivalent to a 0-length write with the COW flag set.

10.2.2.2 Create a replica of the object

A replica operation on the default (changeable) version of an object will first create a new changeable version of the object and then replicate it the old (unchangeable) version (on the same or a different ObS).

A replica that explicitly specifies a version other than the changeable version of an object will create a replica without creating another unchangeable version of the object (on the same or a different ObS.

10.2.2.3 Delete version

Can delete any version except the current (default) version. ObS must maintain the backtrack chain integrity.

10.2.2.4 Reset-to-version

Resets current version to a specified previous version and deletes all the newer versions. If done beyond a replica creation point it may make a replica history non-trackable.

10.2.2.5 Restore-from-a-replica

Restores a frozen version from a replica and creates a new changeable version. If the object and/or version exists the conflict has to be solved by the user (may include reset to the restored version).

10.2.2.6 Delete Object

All version the OID on the specified ObS are deleted.

10.2.2.7 Move an object to another ObS

If versioning is supported the current version is "frozen" and a copy under the old name with a new VID is created in the destination ObS.

If versioning is not supported then a copy of the object (keeping it's original name) is created on the ObS and then removed from the source ObS.

10.2.2.8 Backup and Restore of Objects

The backup operation creates an image of an object on an "non-ObS" media. Restore creates an object on an ObS based a backup image.

Backup and Restore are somewhat equivalent to Replicate/Restore-from-replica only that the backup image is not usable and has no ObS - EQOID.