Experiences Building an Object Based Storage System

David H.C.Du, Dingshan He, Jaehoon Jeong, Vishal Kher, Yongdae Kim,
Aravindan Raghuveer
What this talk is about

- DISC implementation of the OSD T-10 standard
- Describe high level details of the implementation
- How we are using it for some current projects and how we plan to use it in the future.
- How researchers can use it to demonstrate the capabilities of storage intelligence and the object interface
Outline

- Background
  - Object Based Storage
  - Existing Object Interfaces
  - Object Based Storage Ecosystem

- Motivation

- The DISC-OSD Implementation
  - Overview
  - Target
  - Client
  - Security Model
  - Test Suite

- Performance Evaluation

- Future work
Outline

- Background
  - Object Based Storage
  - Existing Object Interfaces
  - Object Based Storage Ecosystem
- Motivation
- The DISC-OSD Implementation
  - Overview
  - Target
  - Client
  - Security Model
  - Test Suite
- Performance Evaluation
- Future work
Object Based Storage

- **What?**
  - New paradigm of storage devices
  - The storage device can store, retrieve objects as against blocks
  - Operations: Read/Write Object, Read/Write Attributes
  - The storage device maps an object to the disk location where it is stored. (instead of filesystem)

- **Why?**
  - More functionality at storage
  - Narrow block interface assumes storage devices is dumb?
  - Better management at the storage is needed to cope with the complexity and scale of data
  - Fine-grained security
Object Interfaces

- NASD project at CMU
- Panasas and Lustre
  - Custom object interfaces
- Standardization efforts
  - Standardization of the block interface is essential to enable early adoption of OSDs
  - OSD-T10 was ratified by ANSI in January 2005.
- Reference Implementation Efforts
  - Intel
  - DISC
    - DISC-OSD project started circa summer 2005
  - IBM Haifa
    - Parallel to DISC-OSD
An Object Based Storage Ecosystem

- **Target:**
  - A permanent data store that exposes the object (ex: SNIA T-10) interface.

- **Client File system:**
  - A client file system that maps a hierarchical namespace to a flat object namespace.
  - Makes the control, data path separation transparent to end user.

- **Meta Data Server:**
  - Location tracking of objects.

- **Policy Manager and Security Manager**
  - Maintains access permissions on objects.
- A request must be accompanied by a valid credential
  - Credential is generated by Security Manager using key shared between SM and OST
    - Authenticating client
    - Operations that the client is allowed to perform (maintained @ policy manager).

- The MDS informs the location of the object

- Client contacts the required target with the credential
  - Target verifies credential
  - Performs command
Motivation for this project

- Two main factors:
  - Reference implementation
    - Can serve as a single point of interoperability testing for multiple vendors.
    - DISC projects (described later)
  - Object based storage ecosystem
    - Hands-on experience of a complete object based storage system
    - A platform for developing new ideas to exhibit advantages of object based storage and storage intelligence.
Outline

- Background
  - Object Based Storage
  - Existing Object Interfaces
  - Object Based Storage Ecosystem
- Motivation
- The DISC-OSD Implementation
  - Overview
  - Target
  - Initiator
  - Security Model
  - Test Suite
- Performance Evaluation
- Future work
DISC-OSD Implementation Overview

- **Target:**
  - iSCSI target
  - Exposes the T-10 interface

- **Client:**
  - File-system for OSD target, transparent interaction with the security modules

- **Security:**
  - Basic security model with policy manager, security manager implementations

- **Testing:**
  - Rigorous, extensible testing suite for functionality testing.
Overview of OSD target functionality:

- Uses a filesystem as permanent store for objects.
  - Objects $\rightarrow$ Files
  - Attributes $\rightarrow$ Files
  - Partitions $\rightarrow$ Directories

- OSD Command Interpreter to convert every OSD command into a corresponding filesystem command

- Commands
  - All commands go through 3 phases:
    - Retrieval and setting of attributes
    - Perform command
    - Update attributes affected by this command.
  - Commands not supported:
    - Perform Task Management Function
    - Set Master Key
    - Send diagnostic
OSD Target Contd...

- **Sense Data**
  - Reporting cause of failure of a command (quota violation etc).

- **Attributes**
  - Attribute pages: Related attributes are grouped into pages.
  - All the mandatory attribute pages
  - directory attribute page: per-object index that points to the various attribute pages of object.
  - Sense errors triggered by manipulation of attributes.
Outline

- Background
  - Object Based Storage
  - Existing Object Interfaces
  - Object Based Storage Ecosystem
- Motivation
- The DISC-OSD Implementation
  - Overview
  - Target
  - Client
  - Security Model
  - Test Suite
- Performance Evaluation
- Future work
Two types of clients provided

- User level program that can be used to send a-command-at-a-time to the target
  - Useful to test functionality etc.
  - Our test-suite internally uses this.

- OSD File-system
  - Provides a hierarchical namespace to use the object based target
  - Tested on Linux kernel 2.4
  - Currently can work with one target.
  - Transparent interaction with the security manager to fetch credentials etc.
Detailed Architecture: User Level Client

- **App.**
- **VFS**
- **so:** upper level driver
- **Middle level SCSI driver**
- **osdfs**
- **iSCSI:** low level driver

- **1cmd**

- **Security Manager**
- **Policy Manager**
- **ACL-D/B**

- **SendCDB()**
- **GetCredential()**

- **OSD Command Interpreter**
- **iSCSI target**

User space

Kernel space

User space - Kernel space interactions:
- GetCredential() → Security Manager
- Security Manager → Policy Manager
- Policy Manager → Security Manager
- Security Manager → GetCredential()

Kernel space - User space interactions:
- SendCDB() → Security Manager
- Security Manager → SendCDB()
Detailed Architecture: Filesystem as OSD-Client

VFS

osdfs

so: upper level driver

Middle level SCSI driver

iSCSI: low level driver

App.

1cmd

Security Manager

Policy Manager

ACL-D/B

GetCredential()

credential

response()

sendCDB()
Outline

- Background
  - Object Based Storage
  - Existing Object Interfaces
  - Object Based Storage Ecosystem

- Motivation

- The DISC-OSD Implementation
  - Overview
  - Target
  - Client
  - Security Model
  - Test Suite

- Performance Evaluation

- Future work
Scope of T10 Standard.

- Hierarchical key model
- Commands between Policy Manager (PM) and Target (OST)
  - SET KEY
  - SET MASTER KEY
- Credential generation and verification
- Security Methods (client and OST)
  - NO SEC: no security
  - CAPKEY: validates integrity of capability info
  - CMDRSP: validates integrity of CDB, sense
  - ALLDATA: validates integrity of all data between client and target
- Out of Scope
  - Security Manager and Policy Manager
  - Communication between client and Security Manager
  - Communication between Security and Policy Manager
DISC Security Implementation

- Supported security methods
  - NOSEC
  - CAPKEY
  - CMDRSP

- Supported security commands
  - SET KEY

- Beyond Standard
  - Initial prototype of Security and Policy Manager

- Future Work
  - Implement SET MASTER KEY command
  - Support ALL DATA security method
  - Use policy attributes
- Policy Manager (CGM + ACL DB)
- Security Manager (CG + KMM)
- Currently no communication between Security Manager and OST
  - SET KEY can be issued only by the initiator
Testing Suite

- Goal: Provide a generic, extensible framework to automate the testing of the OSD implementation.

- Used for:
  - Regression Testing. [making sure that new code does not break older/stable code]
  - White box Testing
    - Currently, we have designed test cases to ensure maximum coverage of target code.

- Components:
  - Batch Execute Tool: Lets the user execute a set of OSD commands sequentially and collect data on observed behavior of the target
  - Oracle: The desired behavior of the target. [currently we generate the oracle output manually]
  - Comparator: Check if the observed behavior as seen by the Batch execute tool matches the desired behavior as expected by the oracle.
- .osd files:
  - capture use-cases for various test scenarios.
  - code coverage based test generation for target.

- .out files:
  - Capture behavior of target into sense data, other command-specific data returned like objectID, partitionID, attributeValues etc.

- .ora files:
  - Represent the desired behavior of target in terms of sense data etc (same format as .out files)
Outline

- Background
  - Object Based Storage
  - Existing Object Interfaces
  - Object Based Storage Ecosystem
- Motivation
- The DISC-OSD Implementation
  - Overview
  - Target
  - Client
  - Security Model
  - Test Suite
- Performance Evaluation
- Future work
Performance Testing

Raw Throughput Comparison of OSD and iSCSI
## Latency of OSD Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Latency ($\mu$sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CAPKEY</td>
</tr>
<tr>
<td>CREATE PARTITION</td>
<td>15040</td>
</tr>
<tr>
<td>CREATE</td>
<td>3745</td>
</tr>
<tr>
<td>LIST</td>
<td>1928</td>
</tr>
<tr>
<td>LIST ROOT</td>
<td>1713</td>
</tr>
<tr>
<td>SET ATTRIBUTE</td>
<td>1689</td>
</tr>
<tr>
<td>WRITE</td>
<td>2141</td>
</tr>
<tr>
<td>APPEND</td>
<td>2085</td>
</tr>
<tr>
<td>READ</td>
<td>1654</td>
</tr>
<tr>
<td>GET ATTRIBUTE</td>
<td>1677</td>
</tr>
<tr>
<td>REMOVE</td>
<td>8387</td>
</tr>
<tr>
<td>REMOVE PARTITION</td>
<td>10046</td>
</tr>
<tr>
<td></td>
<td>CMDRSP</td>
</tr>
<tr>
<td>CREATE PARTITION</td>
<td>14797</td>
</tr>
<tr>
<td>CREATE</td>
<td>4024</td>
</tr>
<tr>
<td>LIST</td>
<td>1970</td>
</tr>
<tr>
<td>LIST ROOT</td>
<td>1896</td>
</tr>
<tr>
<td>SET ATTRIBUTE</td>
<td>1950</td>
</tr>
<tr>
<td>WRITE</td>
<td>2306</td>
</tr>
<tr>
<td>APPEND</td>
<td>2263</td>
</tr>
<tr>
<td>READ</td>
<td>1863</td>
</tr>
<tr>
<td>GET ATTRIBUTE</td>
<td>1902</td>
</tr>
<tr>
<td>REMOVE</td>
<td>8616</td>
</tr>
<tr>
<td>REMOVE PARTITION</td>
<td>10178</td>
</tr>
</tbody>
</table>
Future Work

- **Optimizations:**
  - **Client:** The client file system needs more sophisticated caching techniques for data and security credentials.
  - Should credentials always be retrieved on-demand?
  - **Target:** Using a file system as a backing store adds lot of overhead especially for small files used for attributes
    - Hierarchical namespace is not required
    - A flat namespace based file system?

- **The reference implementation as a:**
  - Framework for other researchers to build on.
  - Tool to demonstrate the advantages of object based storage:
    - Layout/Disk Geometry awareness
    - Processing close to data
Area: Databases on OSD

Contribution: Layout optimizations for performing hash-join operations

Motivation:
- Join operation is very expensive and common at the same time.
- Multiple on-disk read/write cycles
- Layout Optimizations may fetch good benefits

Concept:
- Spirit: if the storage device knows the access pattern of certain operations, it can optimize layout to suit it.
- Use knowledge of access pattern of a hash-join to enable optimized read and write
Area: Integration of structured and unstructured data

Contribution:

SQUAD: A unified framework for Storing and Querying Unstructured And Structured Data

Motivation:

- Emergence of new query / search primitives
- Mixed Queries: Search across structured and unstructured data
- Executing queries, building indexes using storage intelligence makes system scale to terabytes of data.

Concept:

- Query execution on OSD targets.
- Metadata Server to bridge the structured and unstructured gap.
The SQUAD Framework

- **ISNs**
  - Store and index the unstructured data
  - Self managing and layout aware storage
  - Can execute some queries to enable filtering

- **A simple database**
  - Stores the structured data
  - Uses object-IDs to identify related unstructured data

- **MDS**
  - Provides a uniform store and query interface to structured and unstructured data
  - Distributes queries across database and ISNs

- **Client**
  - Has both filesystem and database views to the data
Advantages of SQUAD

- Data is stored based on where it can be managed best.
  - Structured → Database
    - Store and query structured data
  - Unstructured → ISN
    - Store, query and manage unstructured data.

- MDS provides common entry point
  - Wrapper and indirection infrastructure
  - Filesystem not bothered about database feature
  - Database is not bothered about filesystem feature.

- Can work with existing silos of structured and unstructured data.
  - Not tied to any database and filesystem pair
  - Minimum disturbance → easy migration
  - Existing legacy applications can still work fine.
  - New applications can be written to exploit new functionalities exposed by SQUAD
Advantages of SQUAD

- Exploits rich semantics and relationships between structured and unstructured data to guide storage and retrieval of data
  - Not used by Database only or DB+ filesystem approach

- Scalability
  - Store: Simple databases to store structured data, ISNs have good scalability property due to the underlying OSD infrastructure
  - Query:
    - Structured data queried by D/B, unstructured data queries @ ISN, also uses structured to guide search
    - Early discard properties enhances querying scalability
Acknowledgements

- Mike Mesnier
- DISC consortium members
  - Engenio, SUN, Symantec, ETRI (Korea), ITRI (Taiwan),
- DISC Faculty members
- Cory Devor (Primary Contact for DISC-OSD project: devor@dtc.umn.edu)
- DISC-OSD Team
  - Yinping Lu
  - Changjin Hong
  - Girish Moodalbail
  - Pramod Mandagere
  - Biplob Debnath
  - Sojeong Hong
  - Sarah Sharafkandi
Thank You!

Questions / Comments?