Experiences Building an Object Based Storage System

David Du, Dingshan He, Changjin Hong, Jaehoon Jeong, Vishal Kher, Yongdae Kim, Yingping Lu, Aravindan Raghuveer, Sarah Sharafkandi

Digital Technology Center
DISC
DTC Intelligent Storage Consortium

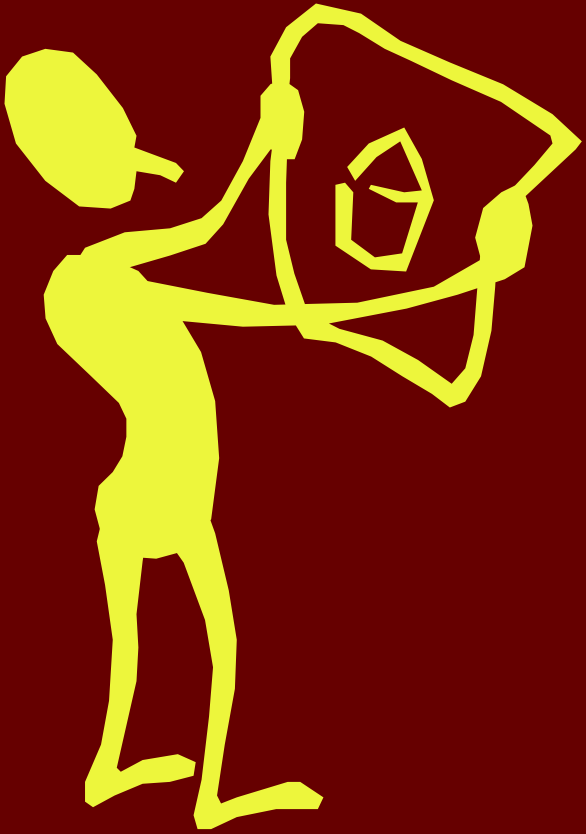
University of Minnesota

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

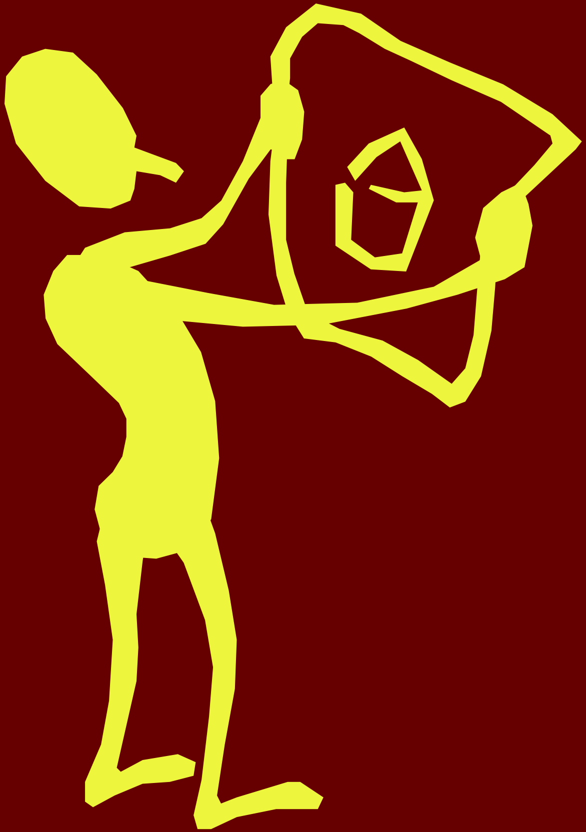
Outline

- 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



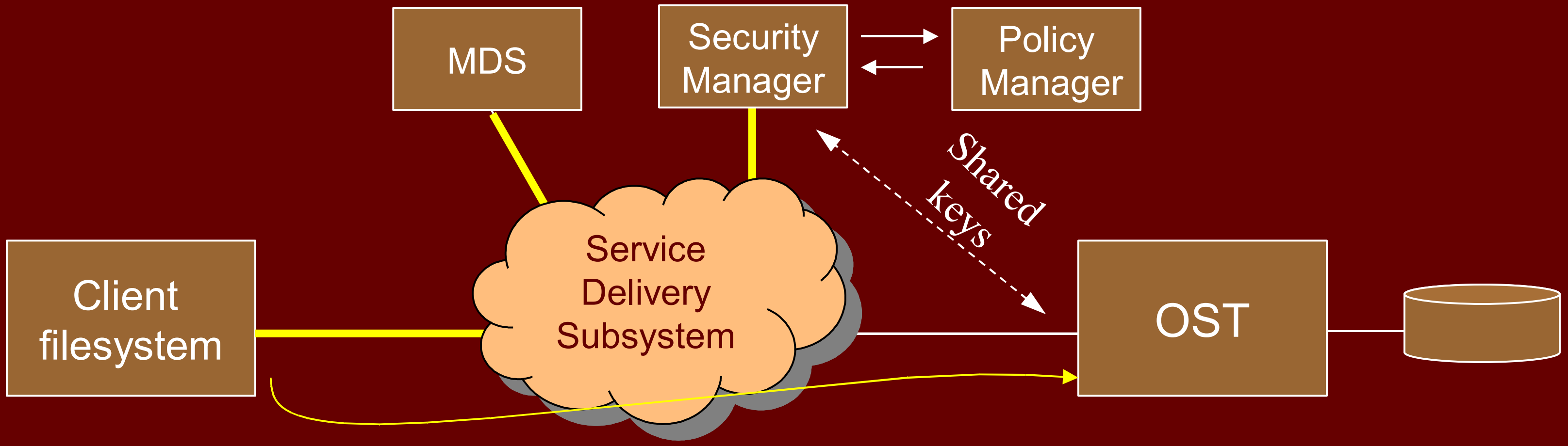
Object Based Storage

- 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

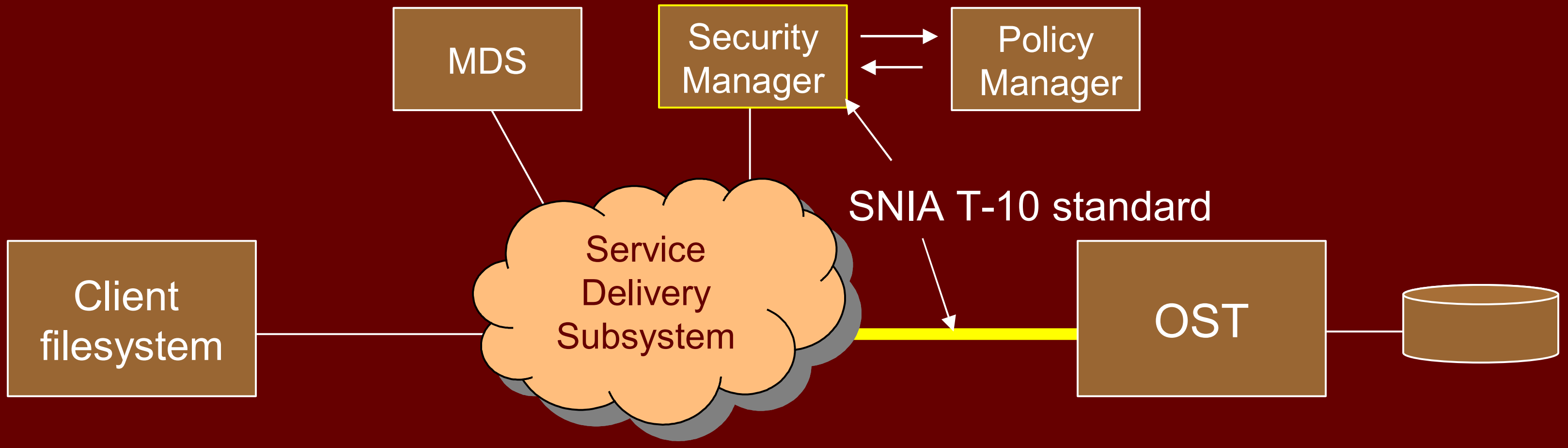
- NASD project at CMU
- Panasas and Lustre
  - Custom object interfaces
- Standardization efforts
  - Standardization of the block interface is essential to enable early adoption of OSDs
  - OSD-T10 was ratified by ANSI in January 2005.
- Reference Implementation Efforts
  - Intel
  - DISC
    - DISC-OSD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

An Object Based Storage Ecosystem



- A request must be accompanied by a valid credential
  - Credential is generated by Security Manager using key shared between SM and OST
    - Authorizing 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

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

Motivation for this project

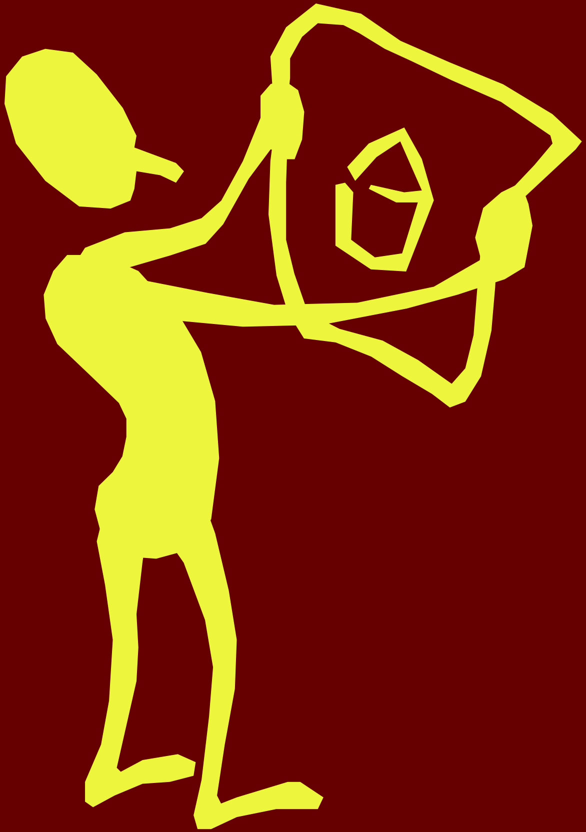
- Two main factors:

- Reference implementation
  - Can serve as a single point of interoperability testing for multiple vendors.
  - DISC projects (described later)

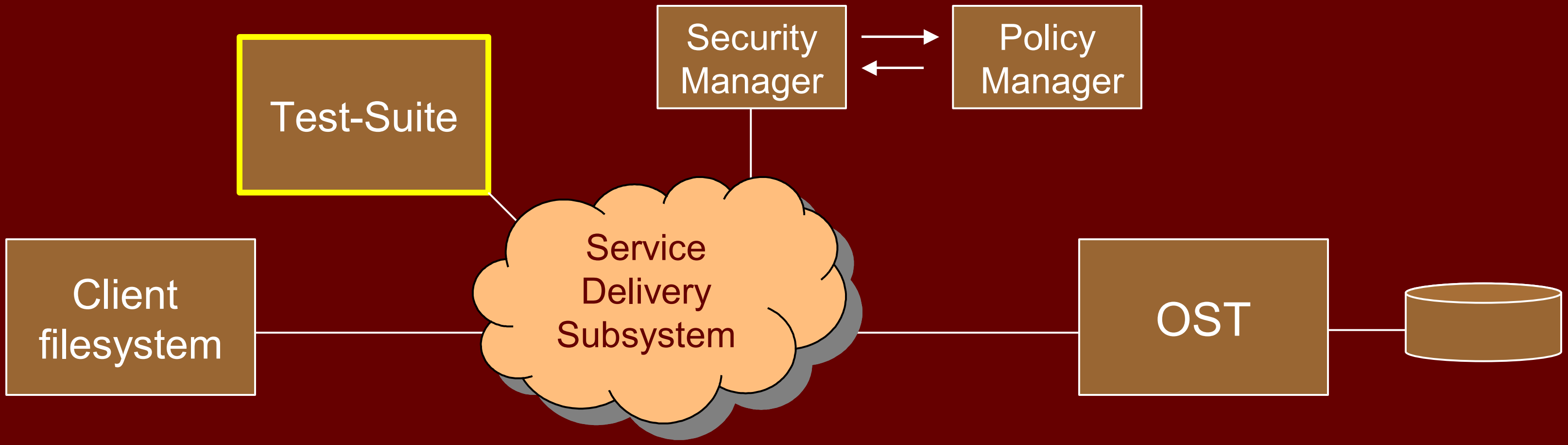
- Object based storage ecosystem
  - Hands-on experience of a complete object based storage system
  - A platform for developing new ideas to exhibit advantages of object based storage and storage intelligence.

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



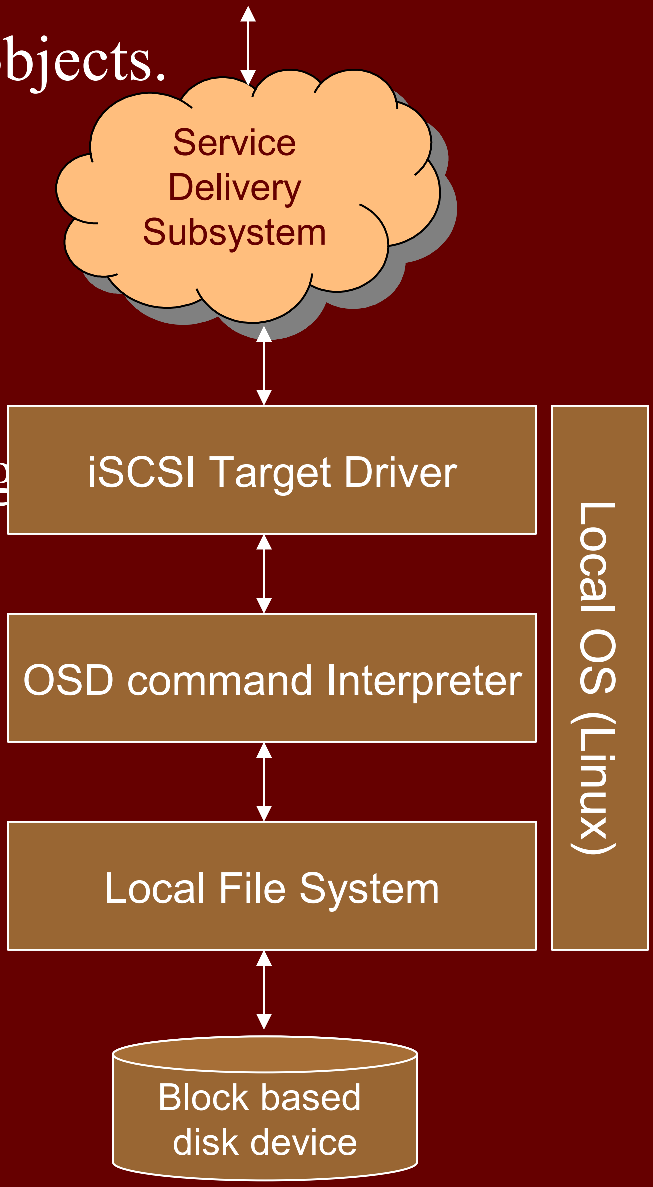
- Target:
  - iSCSI target
  - Exposes the T-10 interface
- Client:
  - File-system for OSD target, transparent interaction with the security modules
- Security:
  - Basic security model with policy manager, security manager implementations
- Testing:

Overview of OSD target functionality:

- Uses a filesystem as permanent store for objects.
  - Objects → Files
  - Attributes → Files
  - Partitions → 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

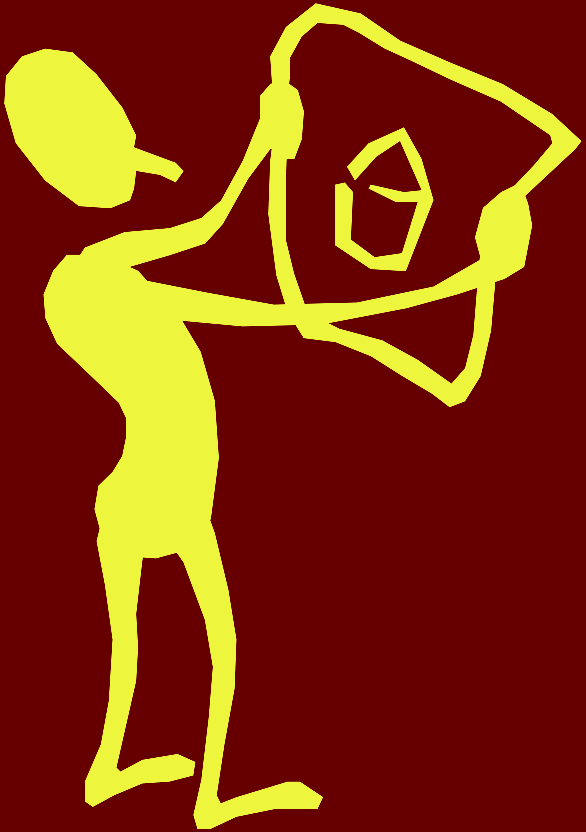


OSD Target Contd…

- 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

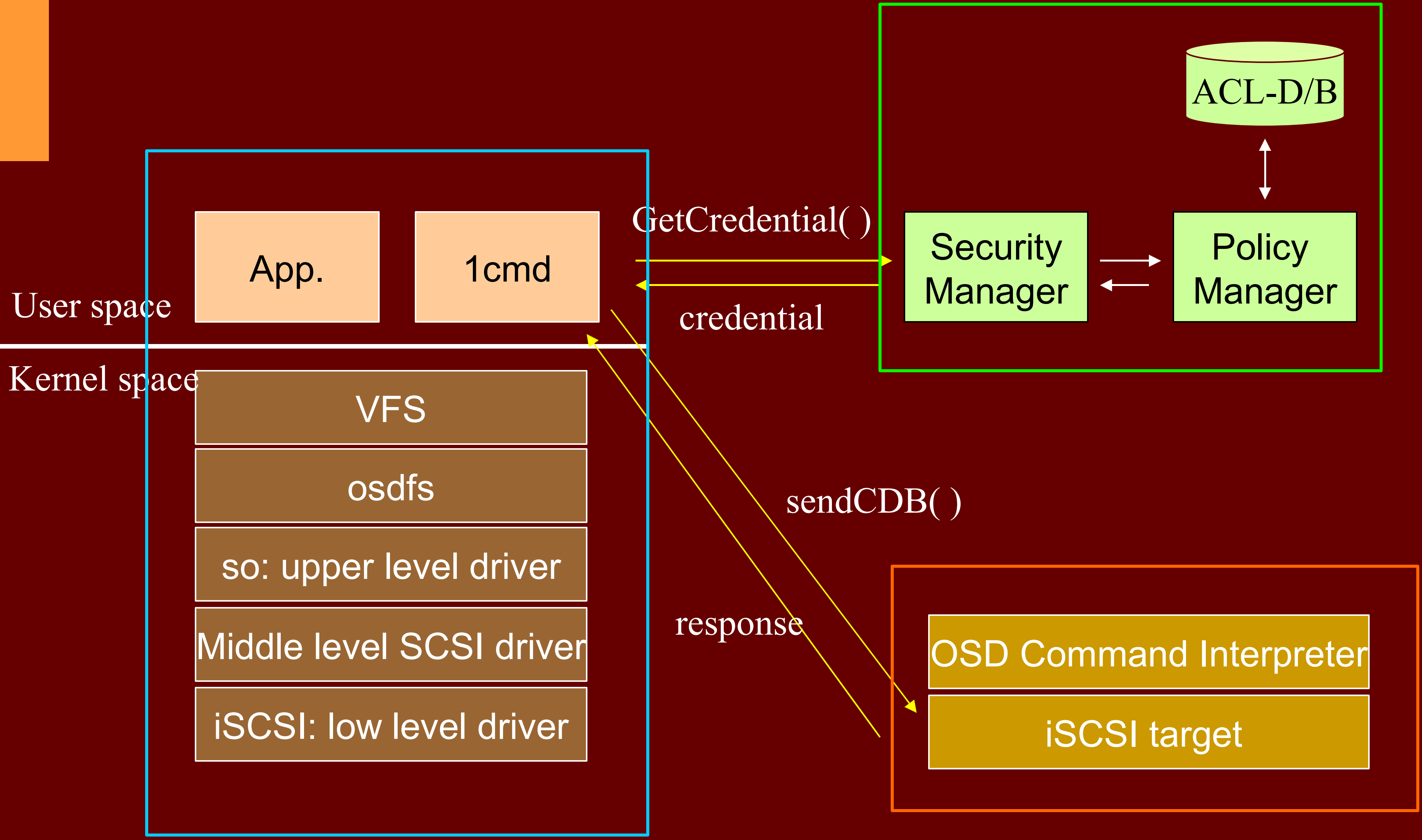
- Background
  - Object Based Storage
  - Existing Object Interfaces
  - Object Based Storage Ecosystem
- Motivation
- The DISC-OSD Implementation
  - Overview
  - Target
  - Client
  - Security Model
  - Test Suite
- Performance Evaluation
- Future work



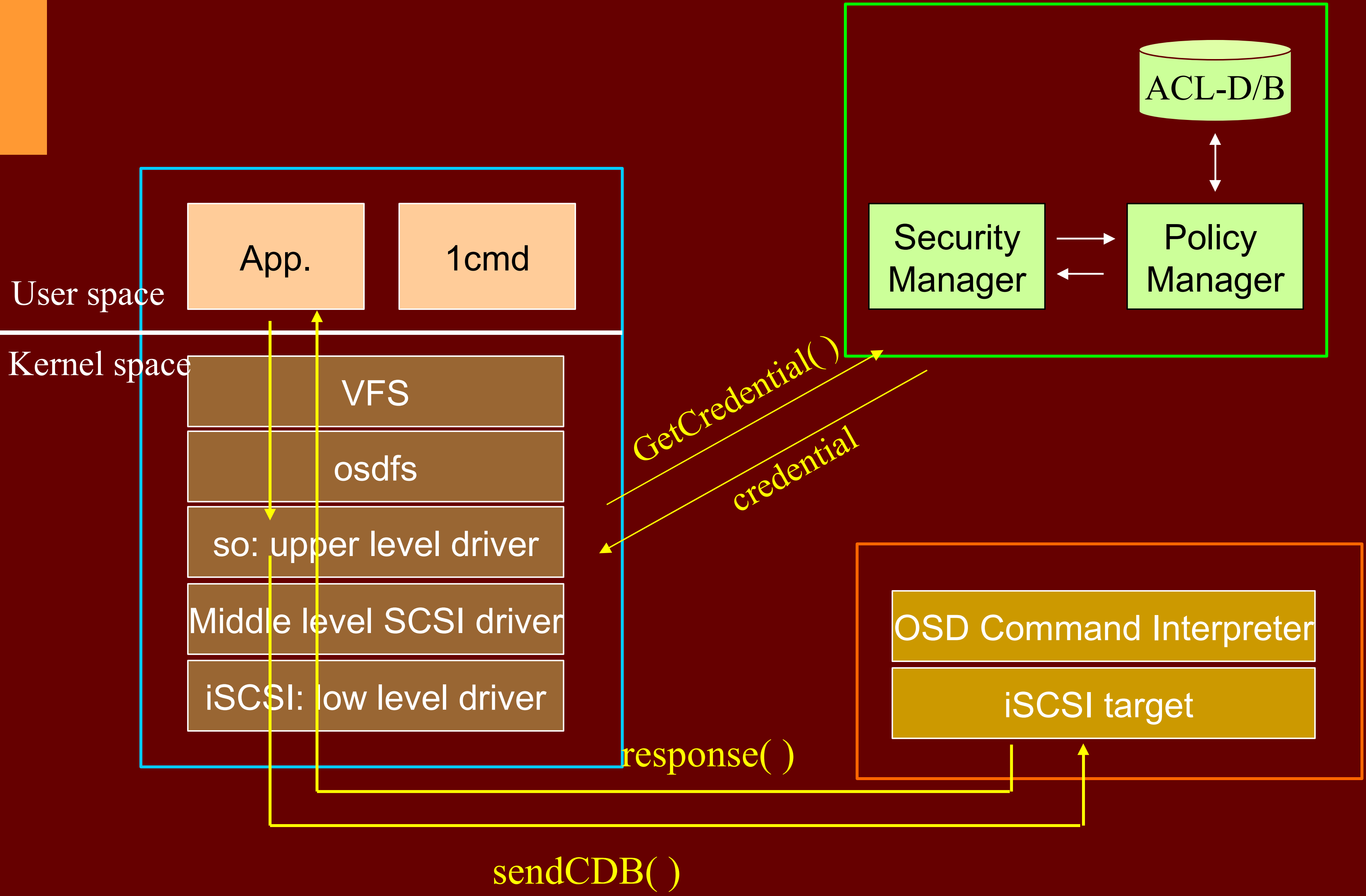
Client

- 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.

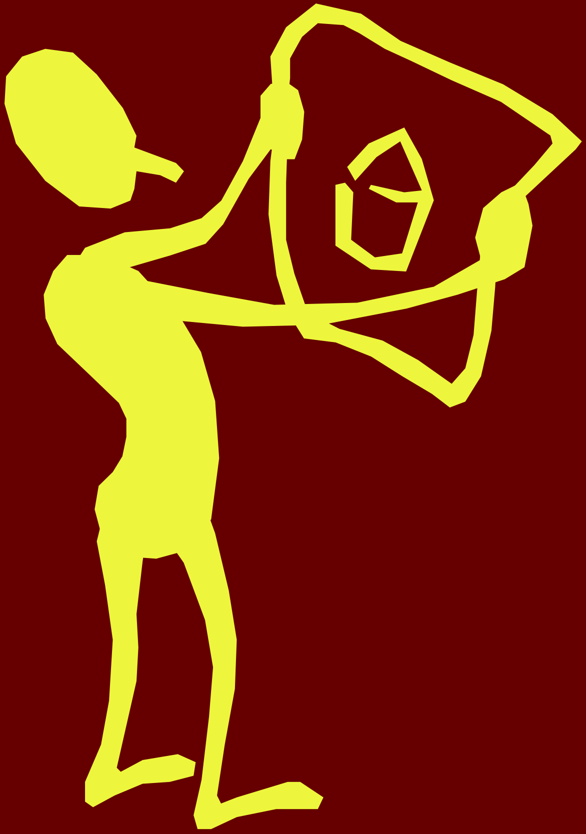
Detailed Architecture: User Level Client



Detailed Architecture: Filesystem as OSD-Client

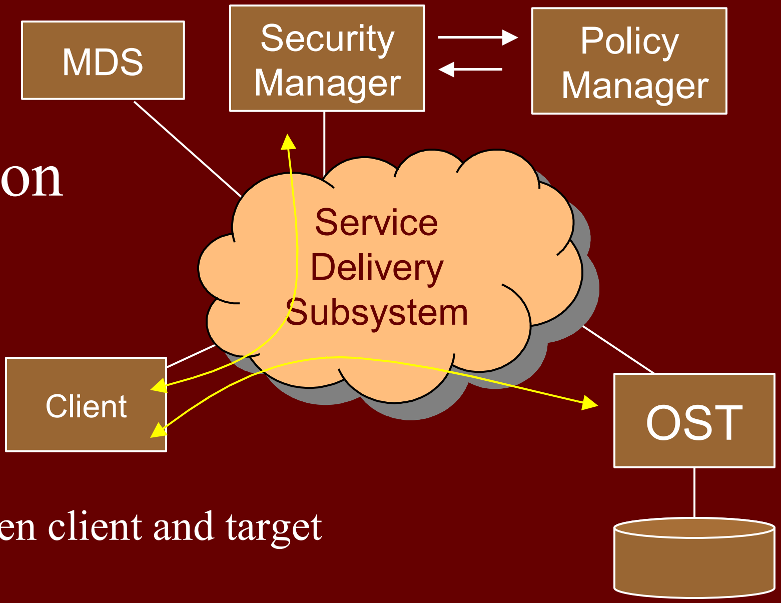


Outline



Scope of T10 Standard.

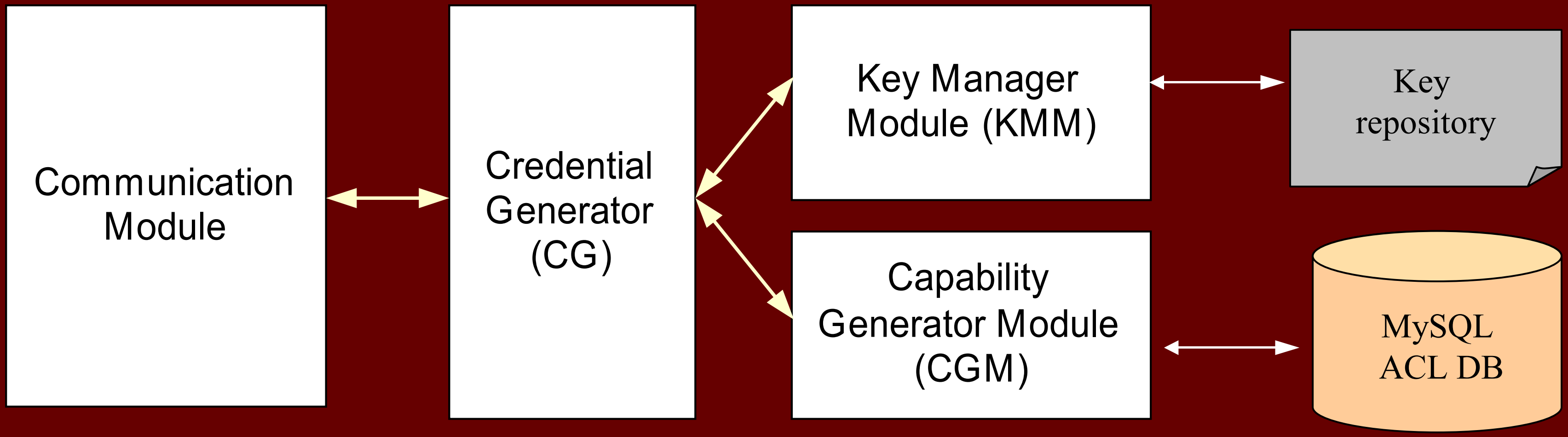
- Hierarchical key model
- Commands between Policy Manager (PM) and Target (OST)
  - SET KEY
  - SET MASTER KEY
- Credential generation and verification
- Security Methods (client and OST)
  - NO SEC: no security
  - CAPKEY: validates integrity of capability info
  - CMDRSP: validates integrity of CDB, 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

DISC Security Manager Implementation

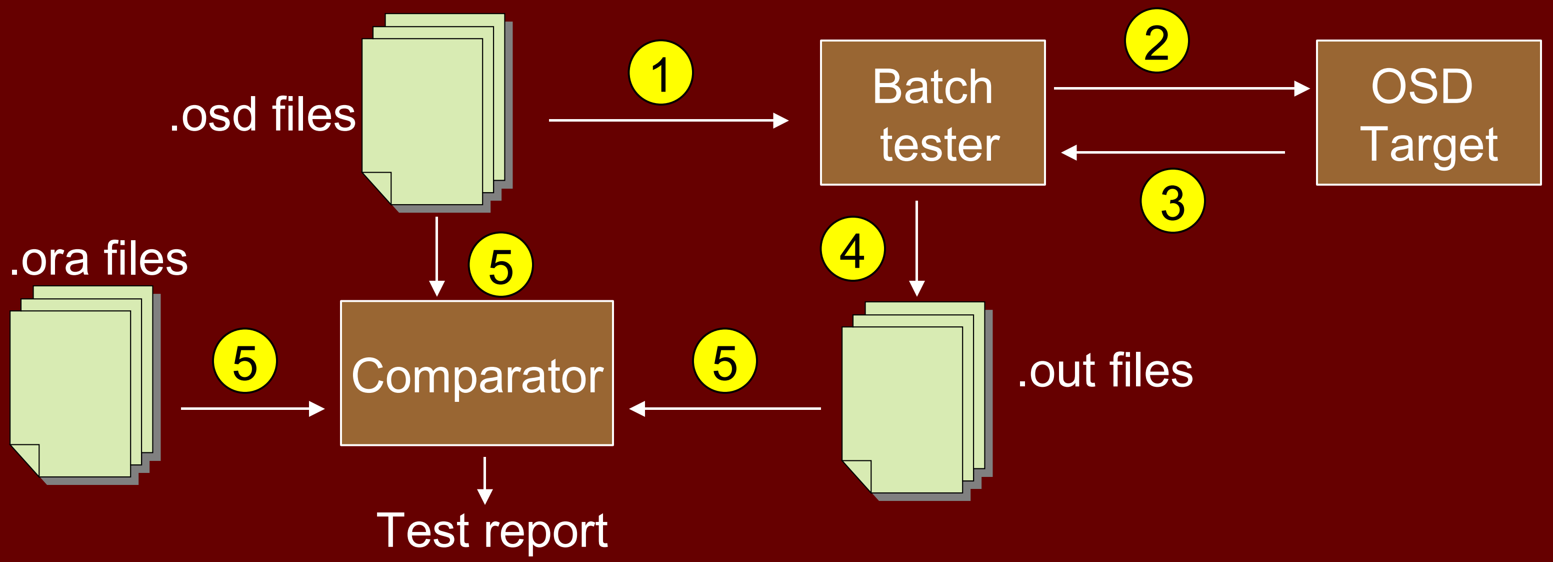


- Policy Manager (CGM + ACL DB)
- Security Manager (CG + KMM)
- Currently no communication between Security Manager and OST

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.

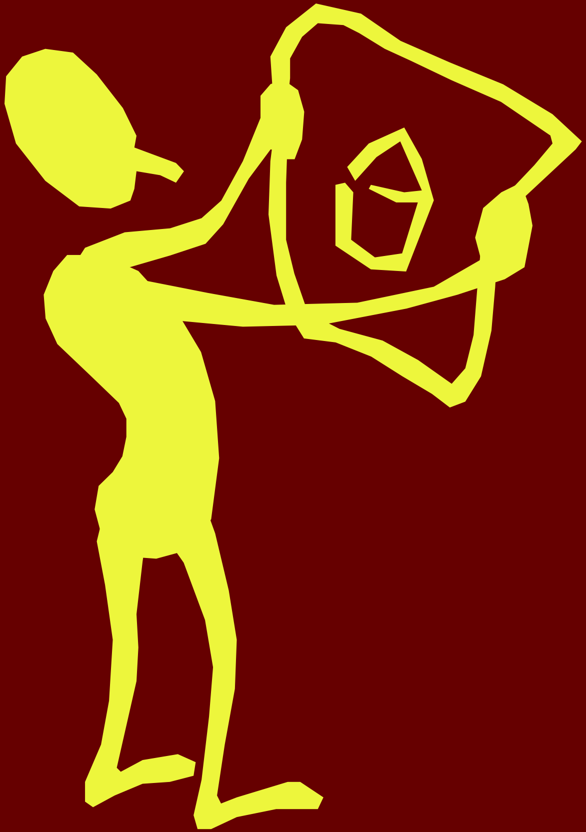
Testing Suite



- .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

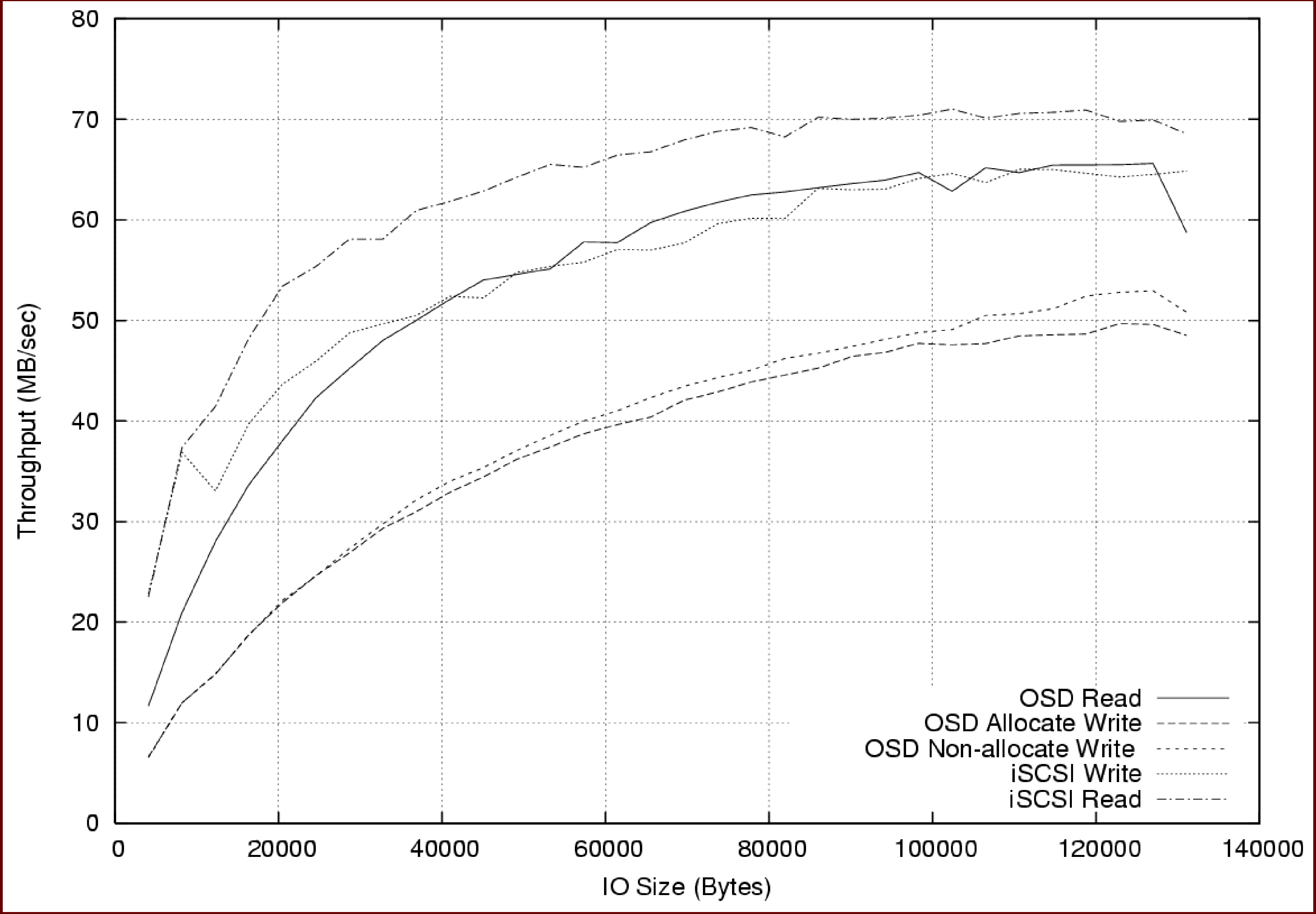
- Background
  - Object Based Storage
  - Existing Object Interfaces
  - Object Based Storage Ecosystem
- Motivation
- The DISC-OSD Implementation
  - Overview
  - Target
  - Client
  - Security Model
  - Test Suite
- Performance Evaluation
- Future work



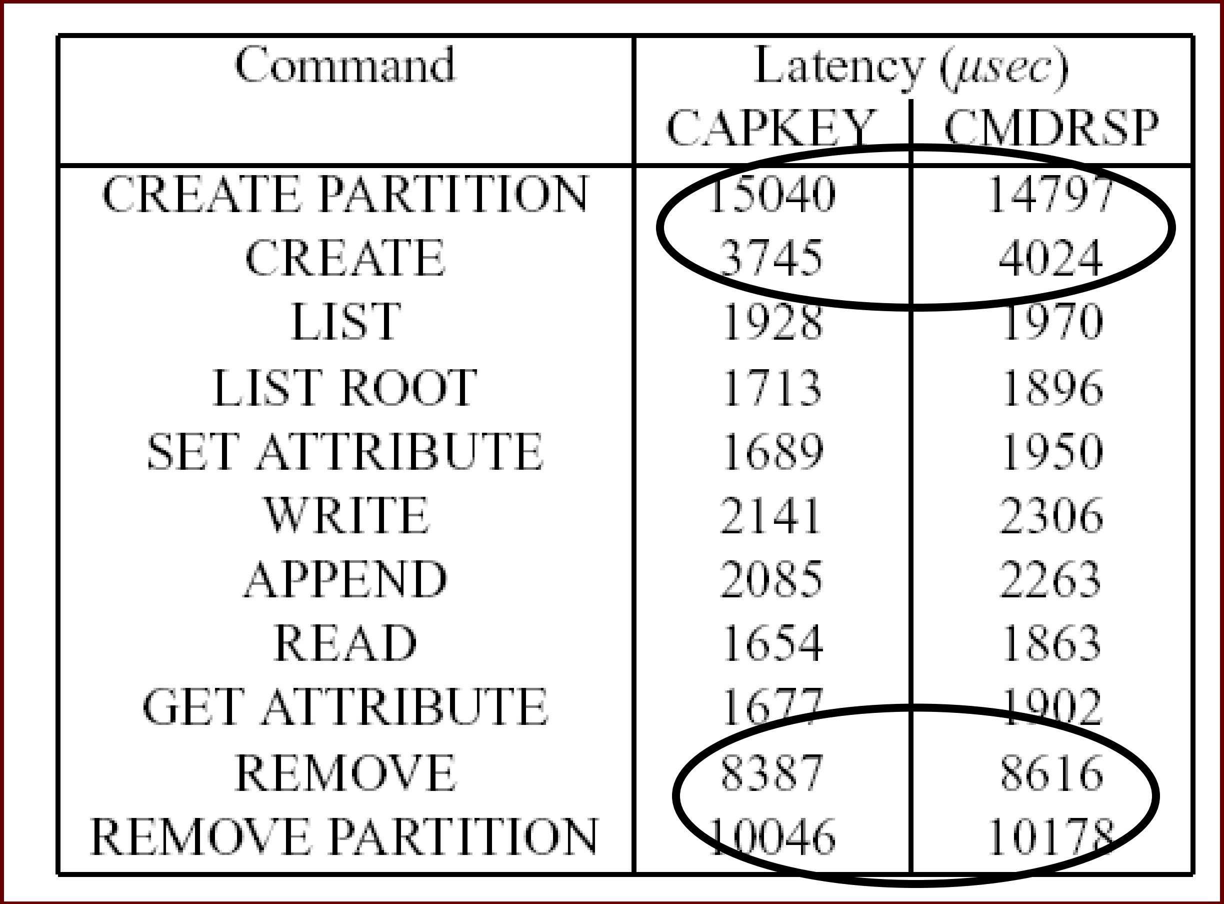
Performance Testing

DISC

Raw Throughput Comparison of OSD and iSCSI



Latency of OSD Commands



Command	Latency (μsec)	
	CAPKEY	CMDRSP
CREATE PARTITION	15040	14797
CREATE	3745	4024
LIST	1928	1970
LIST ROOT	1713	1896
SET ATTRIBUTE	1689	1950
WRITE	2141	2306
APPEND	2085	2263
READ	1654	1863
GET ATTRIBUTE	1677	1902
REMOVE	8387	8616
REMOVE PARTITION	10046	10178

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

Current Projects @ DISC

- SQUAD: A unified framework for storing and querying structured and unstructured data
  - Storage perspective on managing heterogeneous data
    - Efficient storage
    - Querying
- OSD based hierarchical storage management system
  - OSD based tape library
- QoS Specification and Enforcement on Storage

Acknowledgements

- Mike Mesnier
- DISC consortium members
  - Engenio, SUN, Symantec, ETRI(Korea), ITRI(Taiwan),
- DISC Faculty members
- Testing and Implementation Support
  - Girish Moodalbail
  - Pramod Mandagere
  - Biplob Debnath
  - Sojeong Hong
- Logistics and Management support
  - Cory Devor

DISC

Thank You !
Questions / Comments ?