Benchmarking and Testing an OSD for Correctness and Compliance

Allon Shafrir

Dalit Naor, Petra Reshef, Ohad Rodeh, Allon Shafrir, Adam Wolman and Eitan Yaffe

Reliable Systems Technologies
IBM Haifa Research Lab

IBM Labs in Haifa
IBM Verification Conference 2005

© 2005 IBM Corporation

Outline of talk

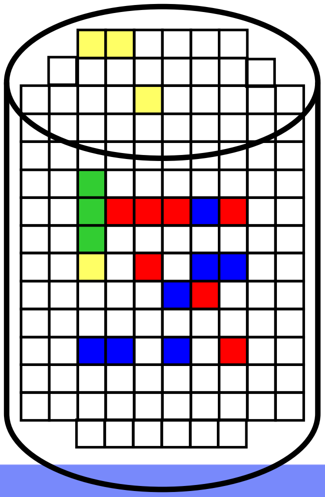
- OSD: a real-life example in need for software testing
- What is an OSD - Object Storage Device?
  - Concept and functionality
  - Motivation for work: object storage in IBM Haifa Lab
- Testing Challenges, briefly
  - Correctness
  - Compliance
  - Security
- Tools, infrastructure
- In more details:
  - Testing OSD security mechanisms
  - Benchmarking
- Future

© 2005 IBM Corporation

Object Storage – the concept

- Raise level of abstraction to present object and not individual blocks
  - Objects are containers of data and meta-data
  - Fine grain, object-level security

Today's Block Device



Operations

read block at address…

write block

Security

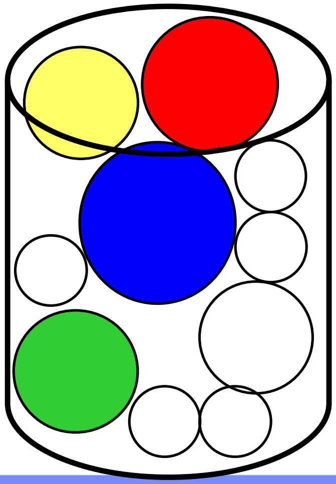
Weak

Full disk

Allocation

External

Object Store



Operations

create object

delete object

read object offset

write object offset

Set/Get attribute

Security

Strong

Per Object

Allocation

Local

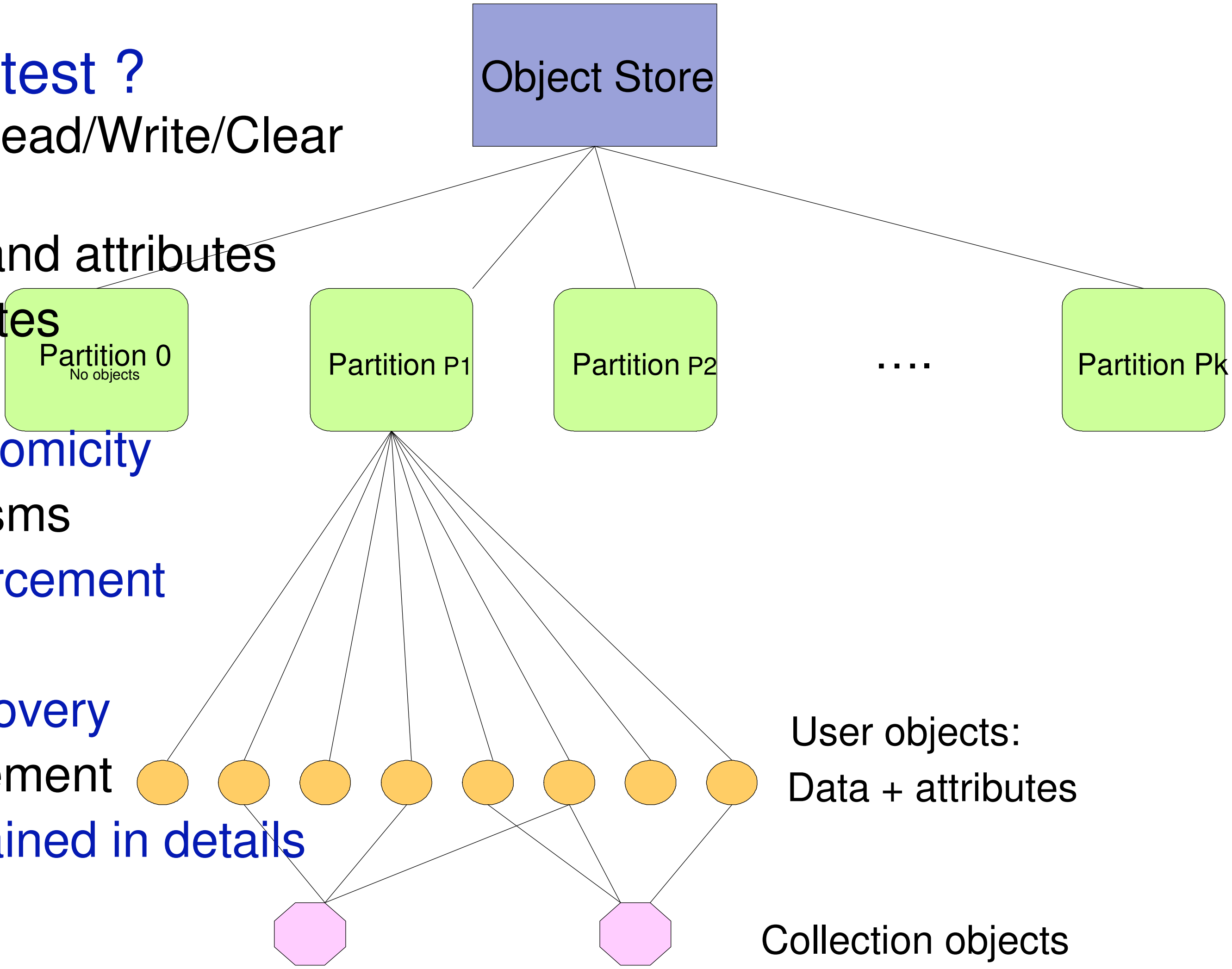
Standard specification

- T10 standard specification was approved on 2004.
  - Industry-driven working group
  - Ongoing work on version 2.
  - Extends the SCSI interface

- OSD is a “hot” buzzword in the storage industry.

© 2005 IBM Corporation

- OSD structure: 2 levels hierarchy
- So, what do we test ?
  - Create/Delete/Read/Write/Clear /Append/List
  - Retaining data and attributes
  - Updating attributes
  - Concurrency:
    - Isolation, atomicity
  - Quota mechanisms
    - Fuzzy enforcement
  - Persistency
    - Correct recovery
  - Security enforcement
    - To be explained in details



© 2005 IBM Corporation

Our OSD software code structure

Test

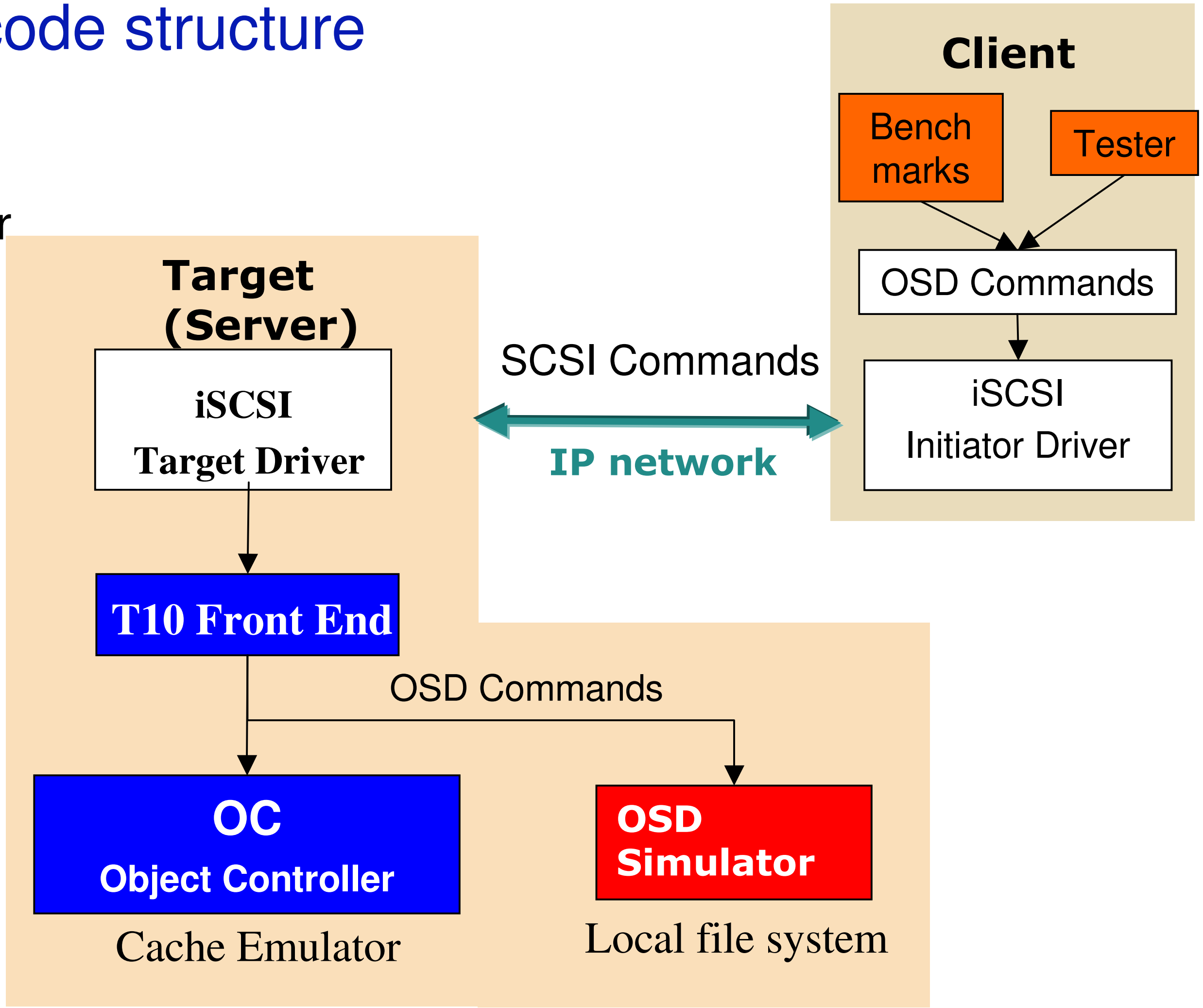
- OC – object controller
- T10 front end

Approach

- Gray-box testing
  - Mostly black box

Tools

- “Simulator”, OSD reference imp.
- Tester
  - Script language
- Harness
- Benchmarks
- Script generator (coverage)



© 2005 IBM Corporation

Testing security mechanisms

- A New and Critical functionality
- Difficult to test
  - Sometimes Invisible

© 2005 IBM Corporation

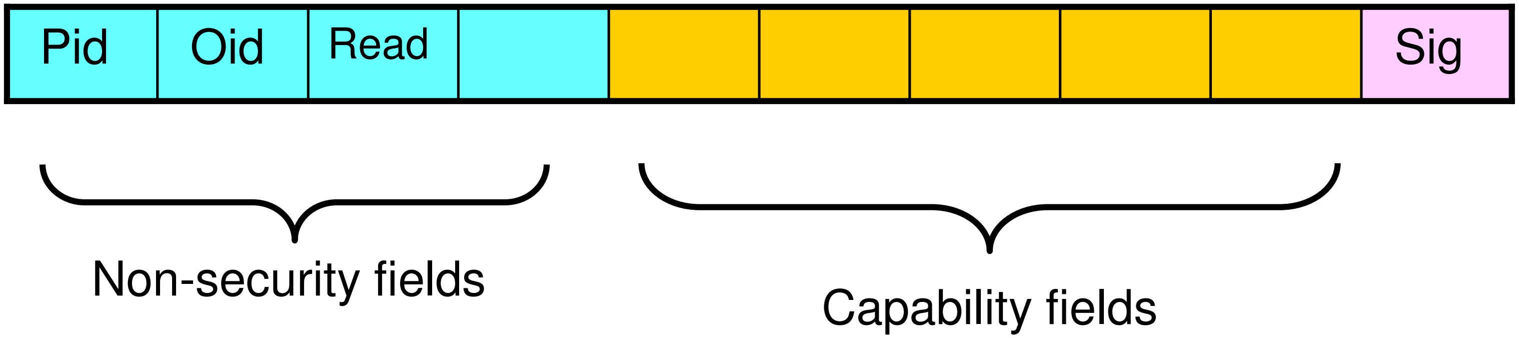
Functional Coverage of Capability Testing

- Capability based security
  - Every command comes with a capability
    - A set of bits permitting the operation
    - Signed cryptographically with a key
  - The device should allow or disallow commands by the standard protocol.
- Security Library
  - Code that generates capabilities for a specific command
    - The minimal capability that permits the command
    - Generation of many different legal capabilities.
    - Generation of many different illegal capabilities
  - Key management
    - Refreshing keys using a special keys hierarchy
  - Revocation scenarios

© 2005 IBM Corporation

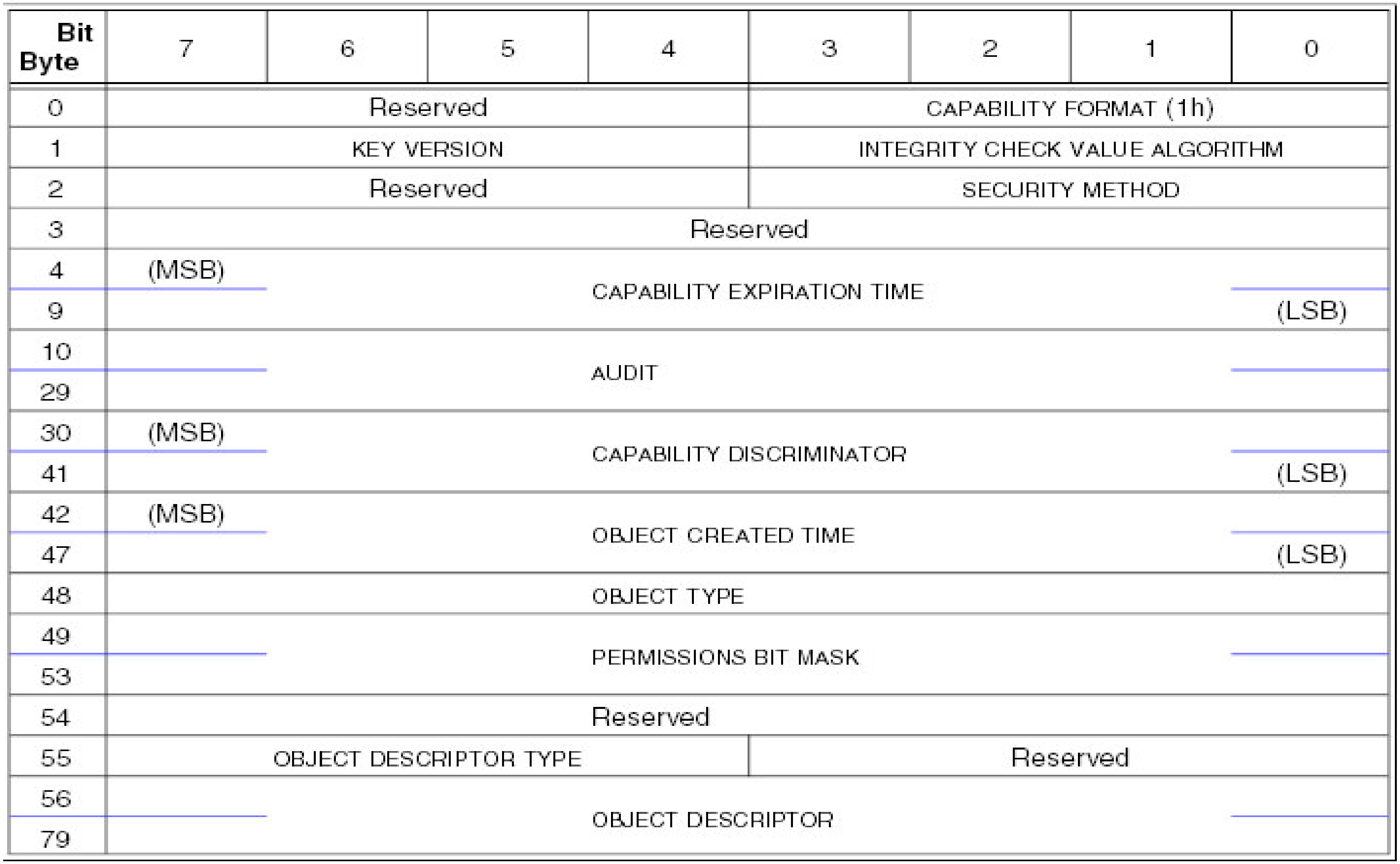
Constructing legal and illegal capabilities

- What makes a capability legal ?
  - Non-security fields of the command (Service Action, partition-id, object-id) are considered input parameters.
  - Capability fields are considered variables.
  - Several rules must be satisfied to allow the command.
  - Given a command, a legal capability is one that allows the command.
    - E.g. A read command



© 2005 IBM Corporation

Table 6 — Capability format



Bit Byte	7	6	5	4	3	2	1	0
0	Reserved				CAPABILITY FORMAT (1h)			
1	KEY VERSION				INTEGRITY CHECK VALUE ALGORITHM			
2	Reserved				SECURITY METHOD			
3	Reserved							
4	(MSB)	CAPABILITY EXPIRATION TIME						
9								(LSB)
10	AUDIT							
29								
30	(MSB)	CAPABILITY DISCRIMINATOR						
41								(LSB)
42	(MSB)	OBJECT CREATED TIME						
47								(LSB)
48	OBJECT TYPE							
49	PERMISSIONS BIT MASK							
53								
54	Reserved							
55	OBJECT DESCRIPTOR TYPE				Reserved			
56	OBJECT DESCRIPTOR							
79								

Constructing legal and illegal capabilities

- Defines constraints (rules) for each rule specified by the standard.
- Combine the constraints to achieve a single rule for each field
- For each capability field:
  - Generate the minimal "legal" value.
    - (e.g. Just the read permission bit set)
  - Add random "noise" within the legal range.
    - (e.g. allow other, unused operations)
- A legal capability is transformed to an illegal one by selecting one field, and randomly changing it to make it illegal.
  - For bit fields – we random a single bit from those that should be '1' and make it '0'.
  - We can repeat this for more than one field.
- Testing a given legal sequence of commands.
  - Running the sequence where each command is sent with many illegal-capabilities and many legal capabilities.
  - Commands are not idem potent.
  - We found a way to do this without breaking the legality of the sequence.

© 2005 IBM Corporation

Benchmarks

- OSD is a new technology; no workloads yet
- Different than file-system benchmarks
  - Richer commands
  - Different compositions
  - Yet to be determined
- Basis for a general-purpose benchmark suite for OSDs in the future
  - Synthetic benchmarks
    - e.g. sequential short writes
  - “system workloads” in the spirit of File-System benchmarks.
- We found that detailed analysis can be used to identify bugs

© 2005 IBM Corporation

Linux SCSI System Bug

- Observation:
  - On long benchmarks, plotted maximum latency of commands
    - There is always a small (statistically negligible) # commands with very large latency
- In Linux SCSI implementation,
  - SCSI commands are submitted in LIFO instead of FIFO order
    - It’s that way for a good reason but it’s bad for intensive workloads.
  - Causes starvation
- We patched the kernel

© 2005 IBM Corporation

Summary

- OSD is a new standard technology
  - Will require a common evaluation and testing criteria for implementations to come
- First reported effort in this direction
  - Some characteristics are unique to this technology
- Spent a substantial amount of effort
  - Our tools proved extremely useful in an interoperability demo with Seagate
- Many possible extensions:
  - Richer functionalities
  - Improved coverage
  - Enriched benchmarks as the field matures

© 2005 IBM Corporation

Thank You

© 2005 IBM Corporation

Security Testing Mode

- First Phase:
- For each command
  - For i=1 to N
    - Send command with an illegal capability C; expect failure
  - Send command with a legal capability C; expect success
- Second Phase
- For i=1 to N
  - Execute the script using an illegal credential on each command

```
create_part pid=q1;
create pid=q1 oid=o1;
write pid=q1 oid=o1 ofs=0 len=2000;

write pid=q1 oid=o1 ofs=2214 len=31287;
write pid=q1 oid=o1 ofs=61384 len=41663;
write pid=q1 oid=o1 ofs=50056 len=8994;
write pid=q1 oid=o1 ofs=22424 len=65466;
write pid=q1 oid=o1 ofs=61008 len=41783;
write pid=q1 oid=o1 ofs=158970 len=34521;
write pid=q1 oid=o1 ofs=136394 len=52936;
write pid=q1 oid=o1 ofs=33300 len=12597;

snapshot_osd; // succesful

clear pid=q1 oid=o1 ofs=7748 len=7499;

snapshot_osd;
```

© 2005 IBM Corporation