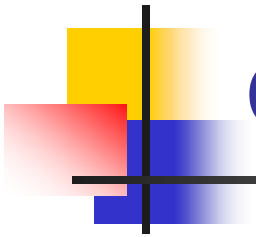
Design and Evaluation of Oasis: An Active Storage Framework Based on T10 OSD Standard

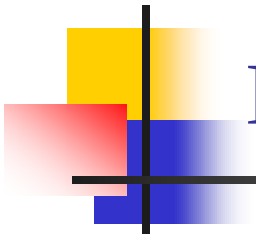
Yulai Xie[12], Kiran-Kumar Muniswamy-Reddy[3], Dan Feng[1], Darrell D. E. Long[2]
Yangwook Kang[2], Zhongying Niu[1], Zhipeng Tan[1]

[1]School of Computer, Huazhong University of Science and Technology
Wuhan National Laboratory for Optoelectronics
[2]University of California, Santa Cruz
[3]Harvard University



Outline

- Background and Motivation
- Oasis Design and Implementation
- Evaluation
- Conclusions and Future Work

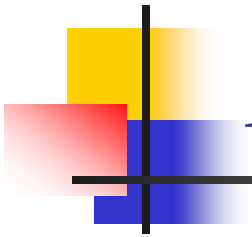


Background and Motivation

- Object-based storage devices are more intelligent than block-based storage devices.
  - manage object data itself & rich semantic object attributes

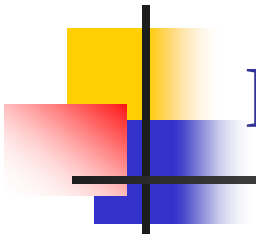
- Active storage can make devices more intelligent by enabling computation inside storage device.
  - filter data on the device side & provide aggregation processing capabilities

- Why not using object-based storage devices to do active storage jobs?



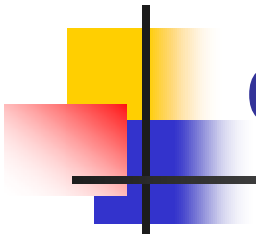
Another aspect, as for T10 OSD Standard…

- T10 OSD Standard
  - Being developed by the OSD Technical Work Group within the SNIA and the INCITS T10 Technical Committee
  - Has defined four kinds of OSD objects, OSD commands, OSD attributes page and basic capability-based OSD security model
- The existing T10 OSD standard (V2) does not sufficiently expose the intelligence/capabilities of object-based storage device.



Efforts to integrate active storage into OSD

- Existing Work
  - Huston et al. (FAST'07)
  - Piernas et al. (Super Computing'07)
  - Qin et al. (AINA'06)
  - John et al. (High Performance I/O Systems and Data Intensive Computing'08)
  - Devulapalli et al.
- Main shortcomings of these existing work
  - Not designed for general object-based storage platform
  - Lots of modifications to the file system or operating system
  - Not practical for use
  - Lack of comprehensive evaluation on real world applications



Our contributions

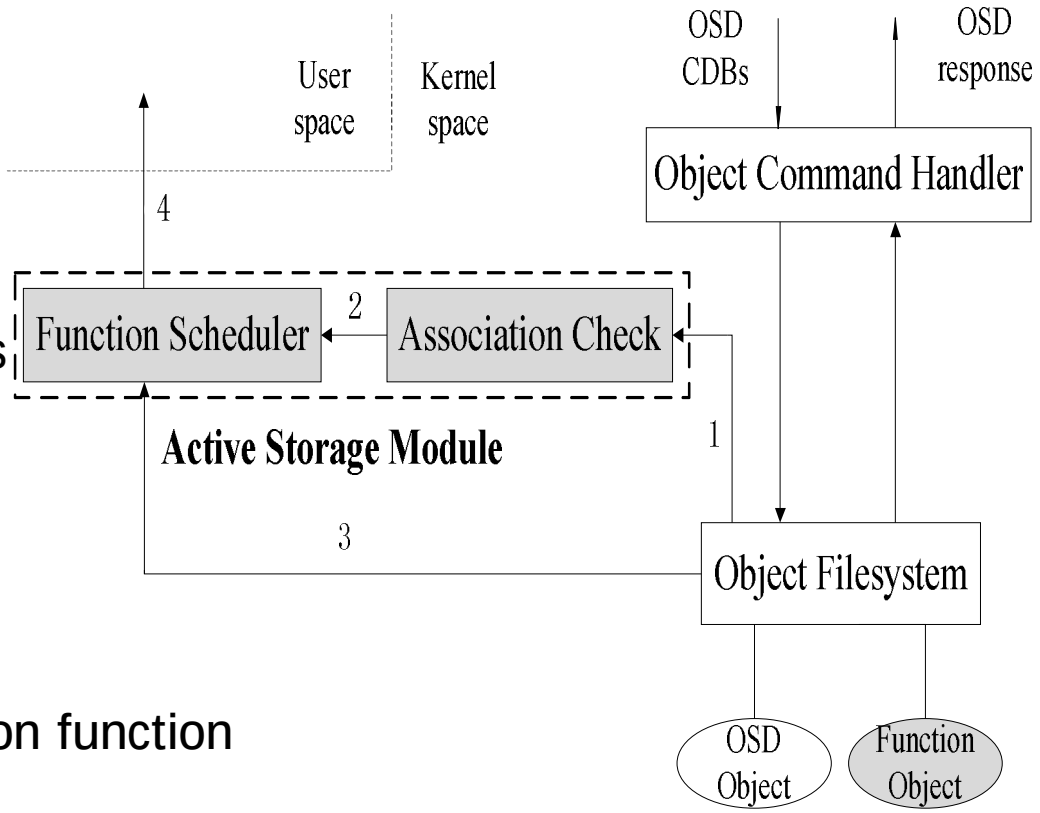
- Propose **An active storage framework based on T10 OSD standard, called Oasis that aims to be for practical use.**
- Design in terms of user case.

  Transparent processing, Multi-granularity processing, Flexible management, preliminary security for execution
- Very small modifications to the T10 OSD standard, aims to be accepted by the vendors

  We have added another kind of object, two parameters in the OSD attribute page and one permission bit in the capability
- Evaluate the performance, scalability and implementation overhead of Oasis on three typical real-world applications: database selection, blowfish decryption and edge detection.

Oasis Architecture Overview

- Object Command Handler
  - Get and analyze OSD command
- Object Filesystem
  - Manage various objects
- Association Check
  - check whether any function objects are associated with an OSD object
- Function Schedule
  - schedule function object to execute

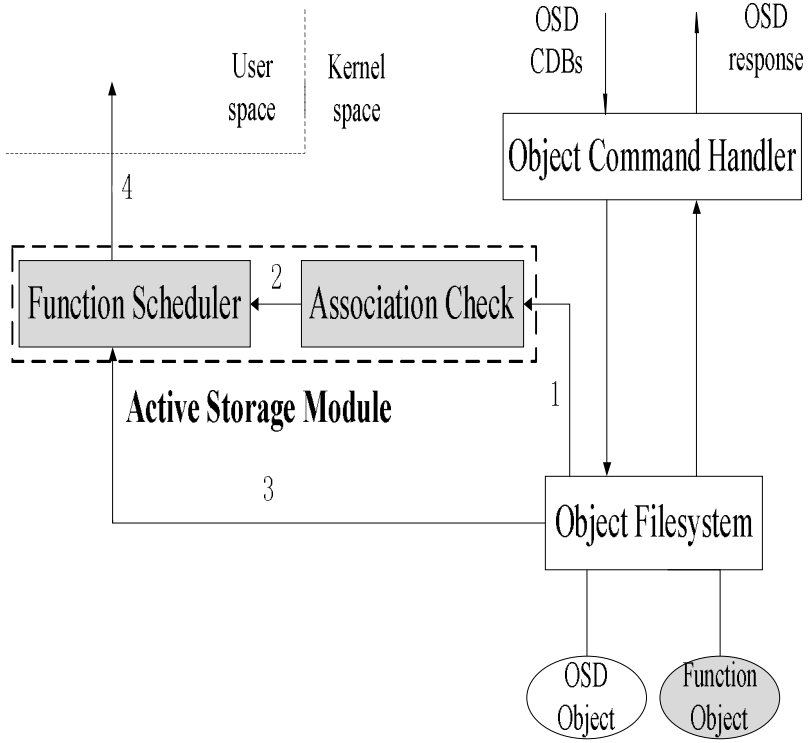


- Function Object
  - Used to hold the offloaded application function (e.g., compression、encryption, etc)
  - A piece of code that can be executed in OSD to perform operations on certain user objects

How we use this system?

If a user want to read an encrypted file from an OSD, what should he do?

- Create a function object that represent Decryption application in the OSD
- Associate this function object with an OSD object
- Send a READ command to the OSD object
- Then the associated function object that represent decryption will be scheduled to execute.



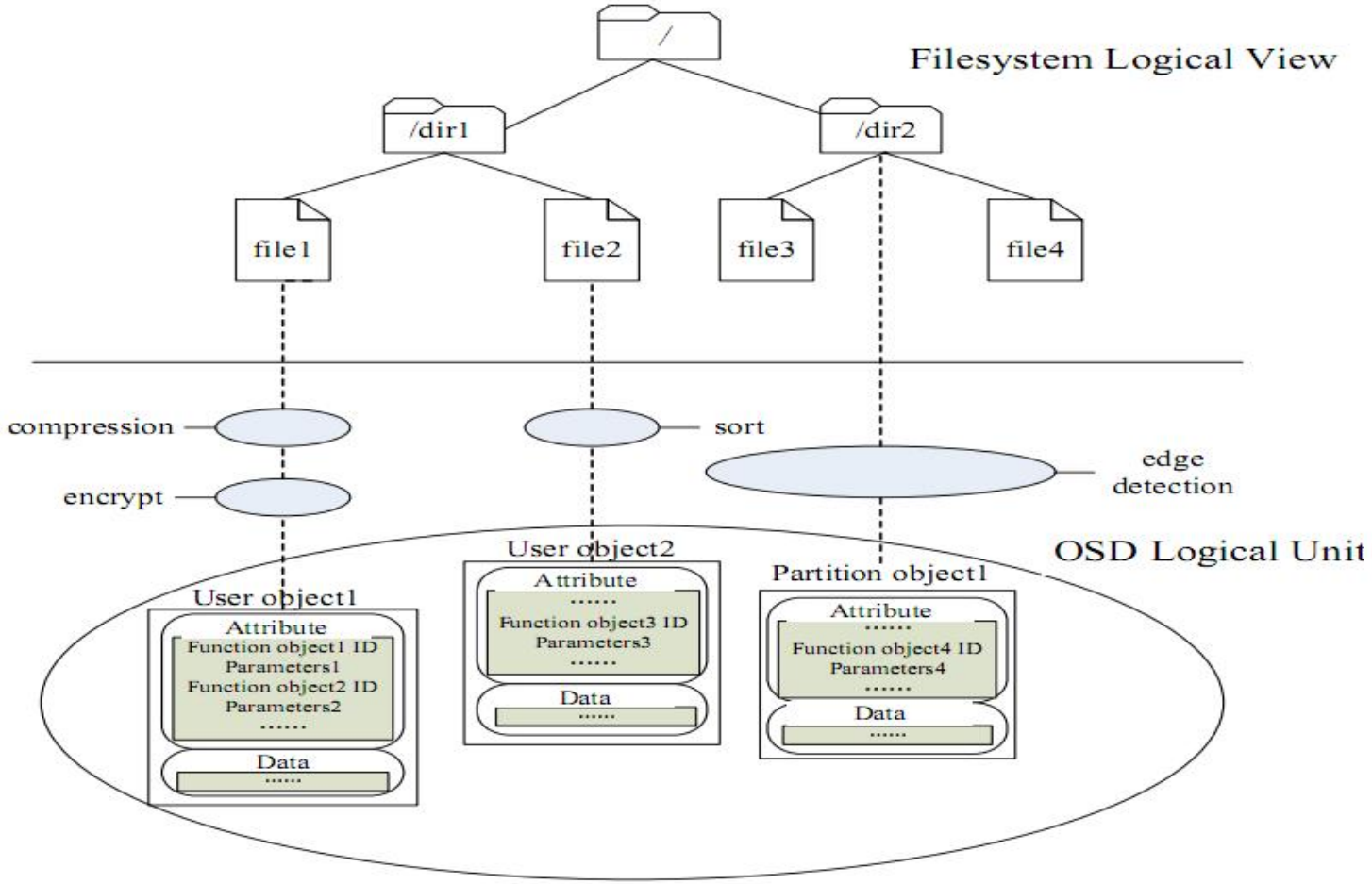
Critical characteristics for practical use: -- Transparent and Multi-granularity processing

- Associate a function object with an OSD object
  - The function object will be invoked to execute during the read or write process
  - We can flexibly apply different application function to different kinds of files
  - Support different processing granularity

Table 2: The extended User Object Information attributes page contents

Attribute Number	Length (bytes)	Attribute	Application Client Settable	OSD Logical Unit
0h	40	Page identification	No	Yes
1h	8	Partition_ID	No	Yes
2h	8	User_object_ID	No	Yes
3h	**8**	**Function_Object_ID**	**No**	**Yes**
4h	**8**	**Parameter**	**Yes**	**No**
5h to 8h		Reserved	No	Yes
9h	variable	Username	Yes	No
10h to FFFF FFFEh	...	...	...	...

An association example



Critical characteristics for practical use: -- Flexible and efficient management

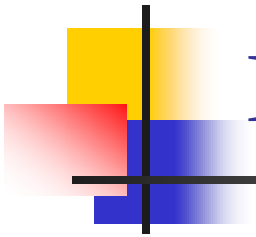
- We use a separate partition to store function object
- We use OSD commands that manage user objects to manage function objects by specifying the partition ID that holds the function objects.
  - Download a function object to the storage device.
    ------ CREATE AND WRITE command
  - Remove a function object from the storage device.
    ------ REMOVE command
  - Conveniently view which function objects are there in the storage device.
    ------ LIST command
  - A user can know which function objects are associated with an OSD object.
    ------ GET ATTRIBUTES command

Critical characteristics for practical use: -- Preliminary security consideration

- Function object should be developed by vendors
  - The vendor has professional knowledge and tools to write and validate code.
- We can use public&private keys to prevent function object code from being modified
  - The vendor encrypts the code with a private key, both the user and OSD have a public key.
- We add a permission bit called FUN_EXE into the capability to prevent unauthorized access
  - Two users may both have authority to set the attributes of an OSD object, but only the user that downloaded function object into OSD can invoke the function object to execute .

Table 3: Permissions bit mask format

Bit	7	6	5	4	3	2	1	0
49	READ	WRITE	GET_ATTR	SET_ATTR	CREATE	REMOVE	OBJ_MGMT	APPEND
50	DEV_MGMT	GLOBAL	POL/SEC	M_OBJECT	QUERY	GBL_REM	**FUN_EXE**	Reserved



Evaluation

- Experimental setup
  - A host and 1, 2 or 4 OSDs, all machines run Redhat linux 2.4.20 and are connected via 1Gbps Ethernet.
  - Oasis is developed based on Intel OSD reference implementation (REFv20).
- Workload

Application	size of dataset	% of data filtering
Database Selection	1.77GB (33 million line records)	87.4%
Edge Detection	584MB(10000 images)	96.7%
Blowfish Decryption	800MB(100 million line records)	0

Evaluation

- Performance improvement

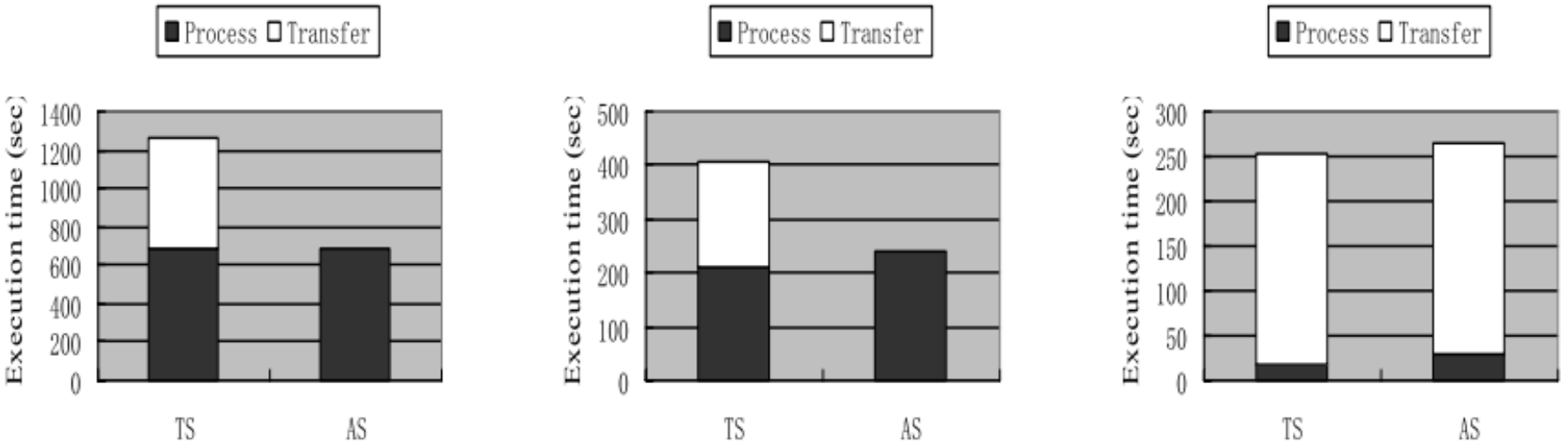
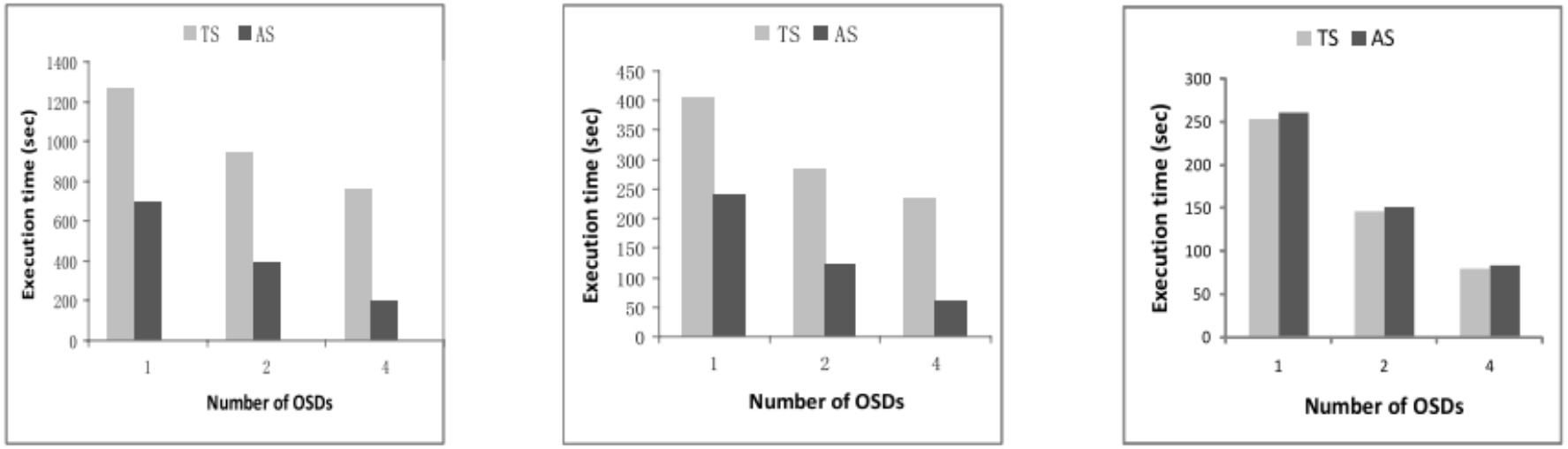


Fig. 4. Execution time breakdown for different applications with one OSD

- TS: Traditional Storage
- AS: Active Storage

Evaluation

- Scalability



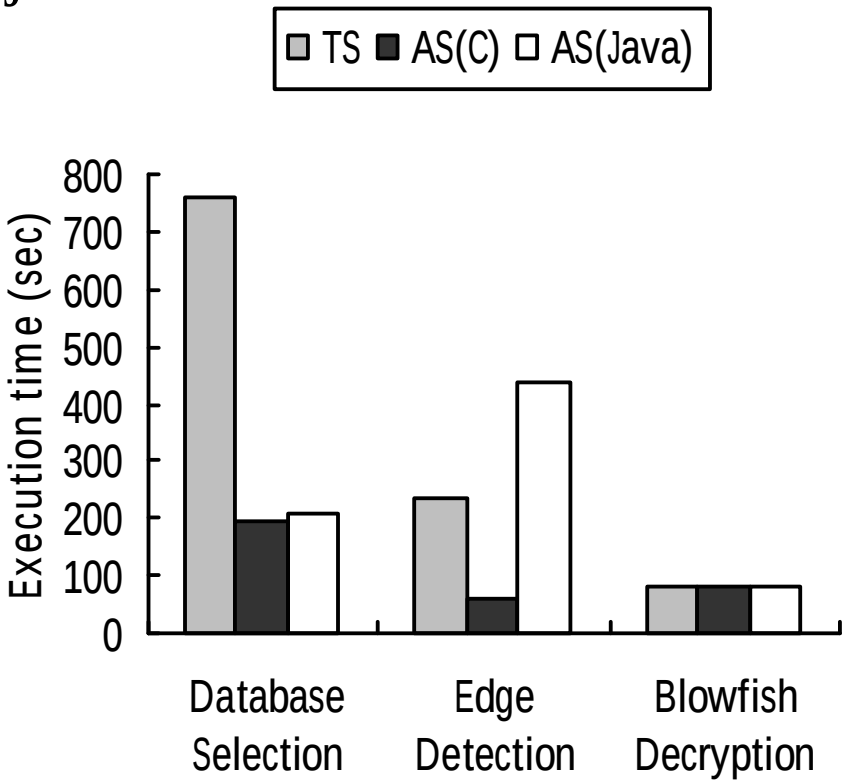
(a) Database Selection (b) Edge Detection (c) Blowfish Decryption

Fig. 5. Execution time for different applications with different number of OSDs

- The performance of AS and TS are both consistent with the increase in the OSDs.
- TS and AS are comparable in the Blowfish Decryption as no data reduction exists in this application.

Evaluation

- Impact of language of function objects
- A large number of I/O operations are required for the Edge Detection algorithm to generate the output image.
- Edge Detection algorithm implementation using the Java language is significantly slower than the implementation using the C language.
- Even such performance degradation with the Java implementation may compromise the benefits of data reduction in the Edge Detection application achieved by the active storage technology.

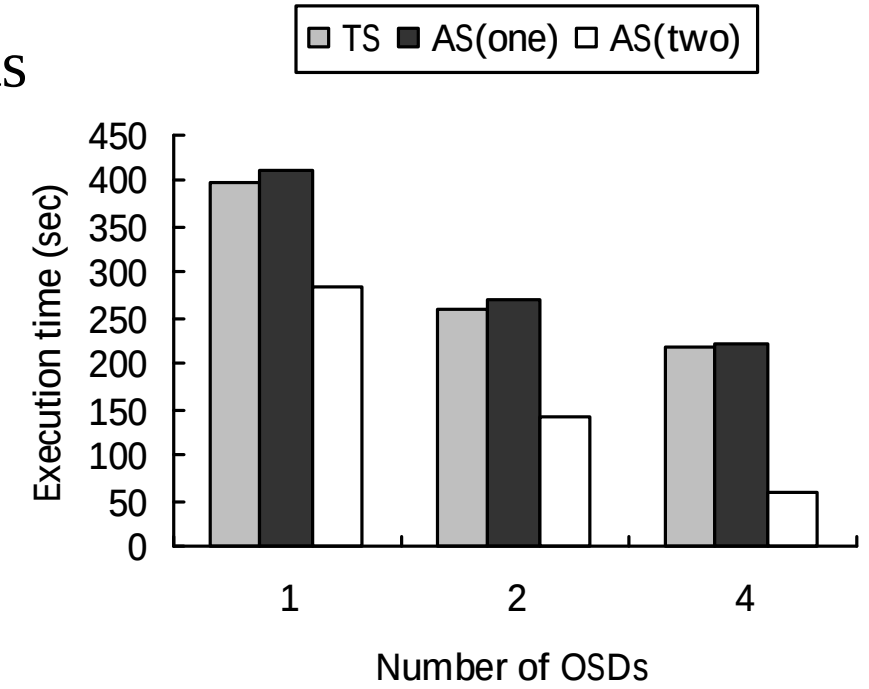


- **I/O intensive application would incur a performance bottleneck with Java implementation.**

Evaluation

- Impact of multiple function objects

  - For a hybrid application that is composed of multiple applications, only applications that can make data reduction across the I/O interconnect can really benefit system performance.



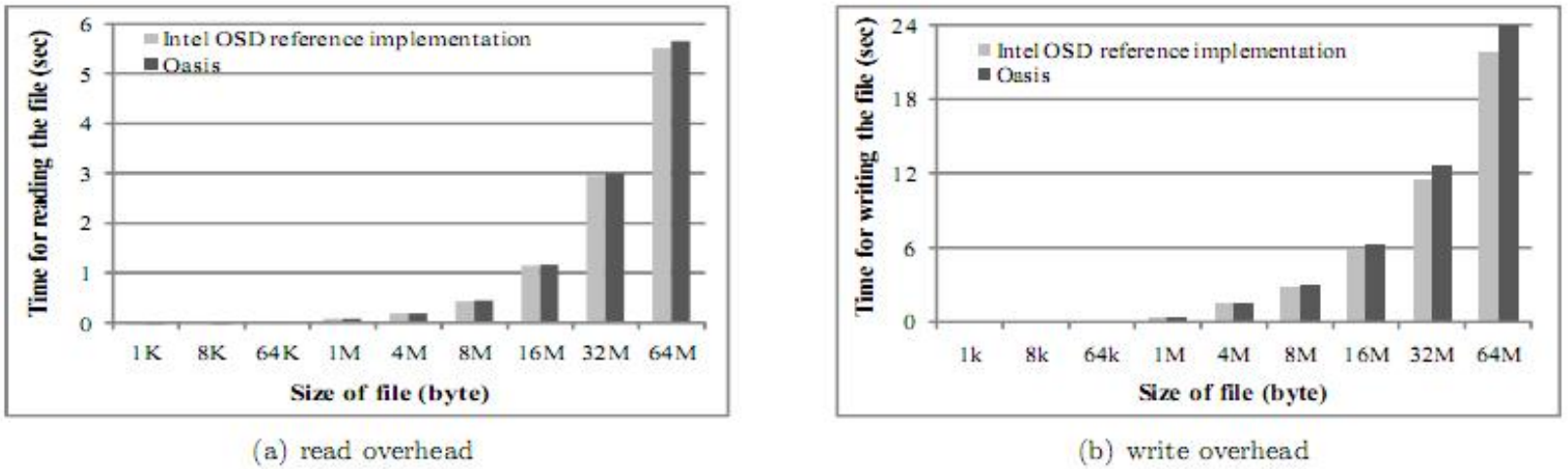
A hybrid application that stacks a Database Selection service on a Blowfish Decryption service

AS(one): first decryption on OSD, then selection on host

AS(two): process hybrid application on OSDs

Evaluation

- Implementation overhead



(a) read overhead (b) write overhead

Figure 8: Implementation overhead of Oasis over the Intel OSD reference implementation

- During every Read or Write, the system has to check whether there exists any function object associated with the OSD object that is being read or written.
- The overhead is small, 1.2%-5.9% for read and 0.6%-9.9% for write.

Evaluation

- Management overhead

Number	Management Description	Object Commands	Completion Time (ms)
1	Create a 1KB function object	CREATE AND WRITE	13.6
2	Associate a function object	SET ATTRIBUTES	2.8
3	Retrieve a 1KB association information	GET ATTRIBUTES	12.1
4	List 512 bytes function objects	LIST	4.5
5	Delete a 1 KB function object	REMOVE	7.8

Conclusions and Future work

- We develop a system that is aimed to be practically used:
  - Small modifications to the existing T10 OSD standard
    Another kind of object, two parameters added, one permission bit added
  - Four kinds of critical characteristics in terms of user case
    Transparent and multiple granularity processing, flexible management, preliminary security consideration
  - System demonstration on three real world applications in terms of performance, scalability, language, etc.

- Future work
  - Concurrent execution of multiple function objects by employing sandbox technology
  - Dynamic workload partition when host or OSDs are heavily loaded.

Thanks!