

Active Object Storage to Enable Scalable and Reliable Distributed File Systems

John A. Chandy

Department of Electrical and Computer Engineering



University of
Connecticut

Active Object Storage

- Active Disks
 - Intelligence at the disk can distribute computation to parallel disks
- Active Object Storage for Parallel File Systems
 - Active Disks for OSDs
 - Use Active Storage to improve parallel file system performance
 - Use Active Storage to improve parallel file system reliability
 - Application aware storage and autonomic storage using active OSDs.

Active Disks

- We already have intelligence at the disk
 - Block management
 - Arm scheduling
- Can we use that intelligence for computation?
 - Disk controller can potentially optimize data layout and retrieval since it knows how the data is stored
 - Avoid extra data transfer across networks, buses, or interconnect
 - Parallel disks get you concurrency

Active Disks

- Vendors haven't provided mechanisms for active disks yet
- Complexity not worth it in commodity disks
- Clusters provide the parallelism right now

Active Disks

- Can we use OSDs to make Active Disks a reality?
 - Application-aware storage
 - Object attributes can give hints to the disk
 - Application specific
 - Parallel File Systems
 - Felix et al. added a filtering layer to Lustre to provide active processing
 - T10 OSD?

Active Disks using OSD

- Previous Implementation
 - Based on disc-osd
 - Object-oriented (Java)
 - Attach object types to storage objects
 - Define methods for object types
- New Implementation
 - Based on osc-osd (supported by Panasas)
 - RPC - Call functions on OSD remotely
 - Execute Engines – C, Java, Perl, etc.

Active Disks using OSD

- How do you move code from client to target within OSD framework?
 - Create an object with the code
 - Each function object has a special attribute that defines the type of associated execute engine
 - OSD can support multiple execute engines

Active Disks using OSD

- How do you execute the method remotely within the OSD framework?
 - New EXECUTE FUNCTION command so that we can invoke a function
 - We use the CDB continuation to specify the parameters
 - Results (if any) returned directly or written to a new object

From T10/08-185r5 changes to OSD-2

Active Disks using OSD

- Code fragment

```
read(oid, buff, offset, nbytes);
```

```
// Do computation
```

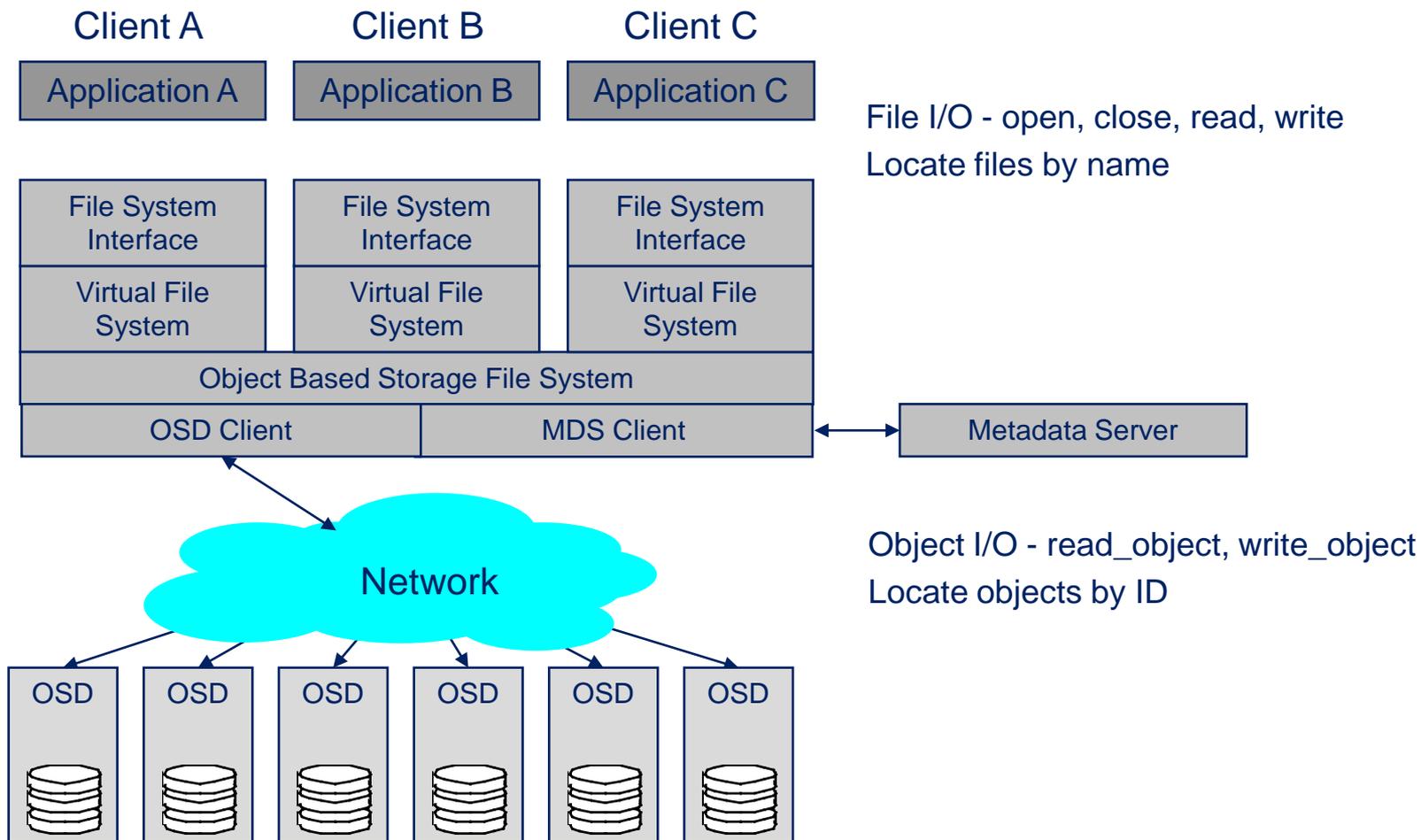
```
newoid = create();
```

```
write(oid, buff, offset, nbytes);
```

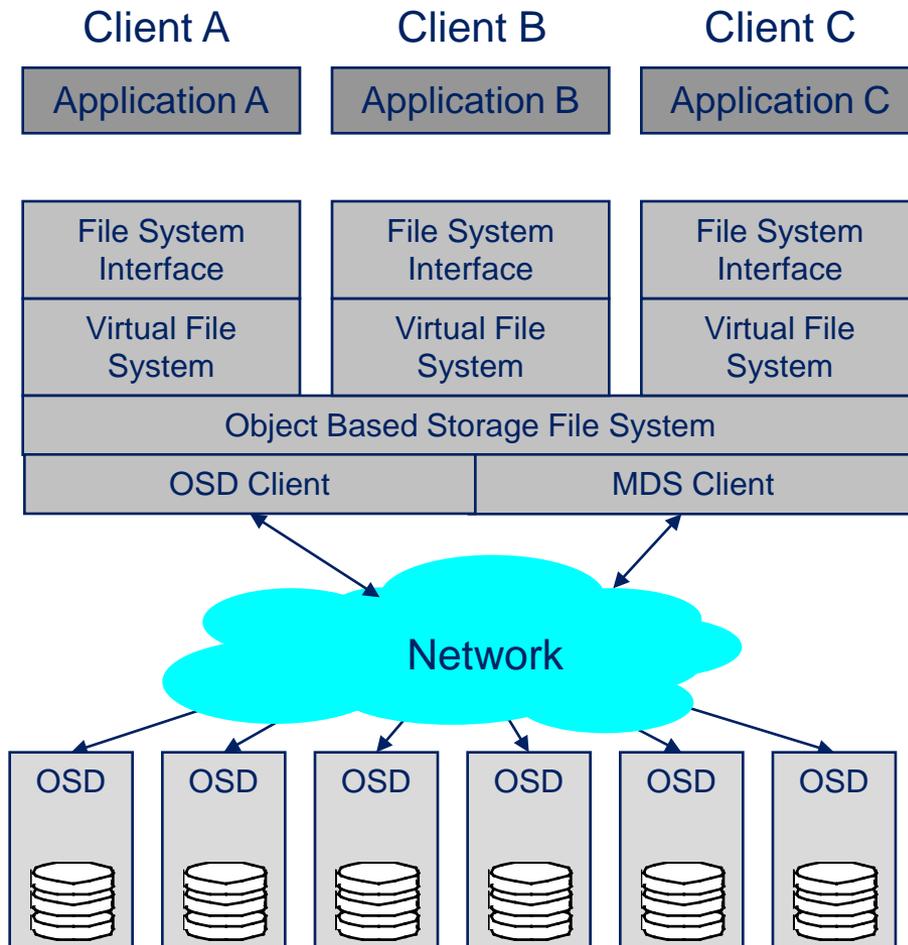
Active Disks using OSD

- Issues
 - Atomicity
 - Can we guarantee atomicity of operations?
 - Safety
 - Should we place limits on active disk computation?
 - CPU, memory, disk
 - What system services should the OSD target provide?
 - Local file system access?
 - Process/Thread management?
 - cron/background?
 - Application adoption is needed
 - Database is the most promising
 - Reductions
 - Update operations

Parallel File Systems



Parallel File Systems



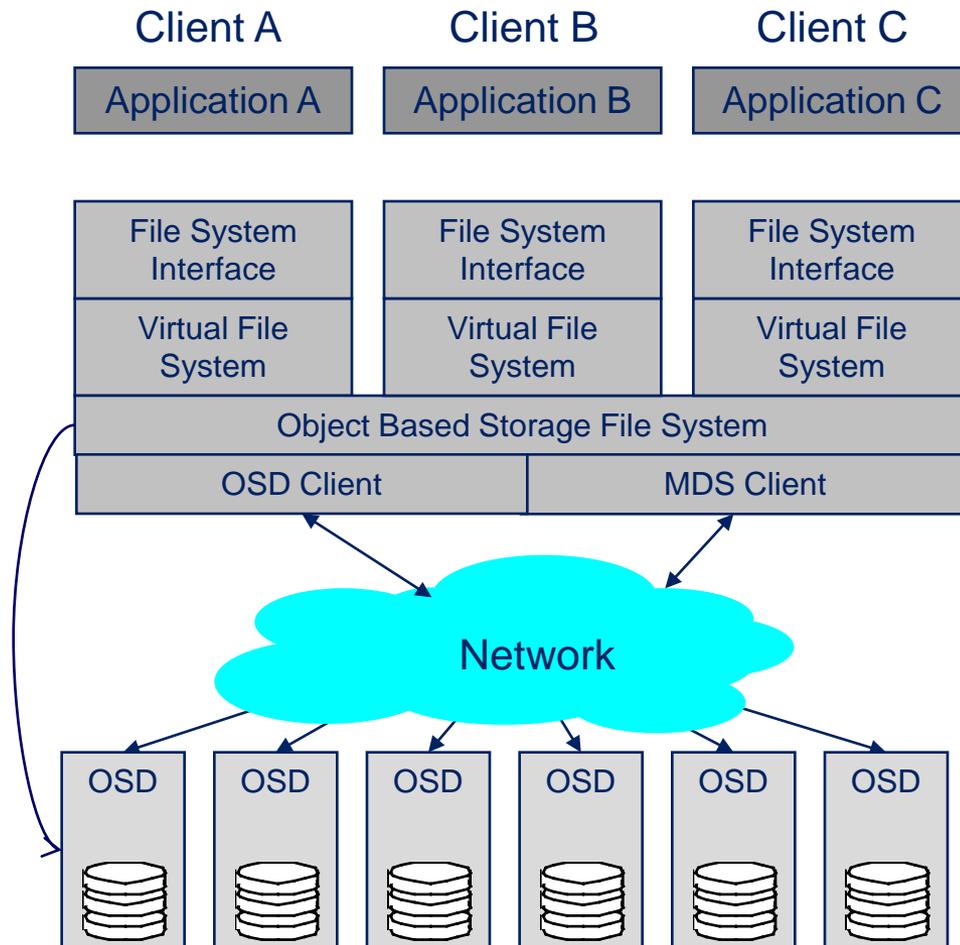
File I/O - open, close, read, write
Locate files by name

[Ali et al. Cluster 2008]

Object I/O - read_object, write_object
Locate objects by ID

Metadata – locate with hashes, use
atomic access protocol

Parallel File Systems



File I/O - open, close, read, write
Locate files by name

[Ali et al. Cluster 2008]

Object I/O - read_object, write_object
Locate objects by ID

Metadata – functionality implemented
at Active OSD
- e.g. directory lookup

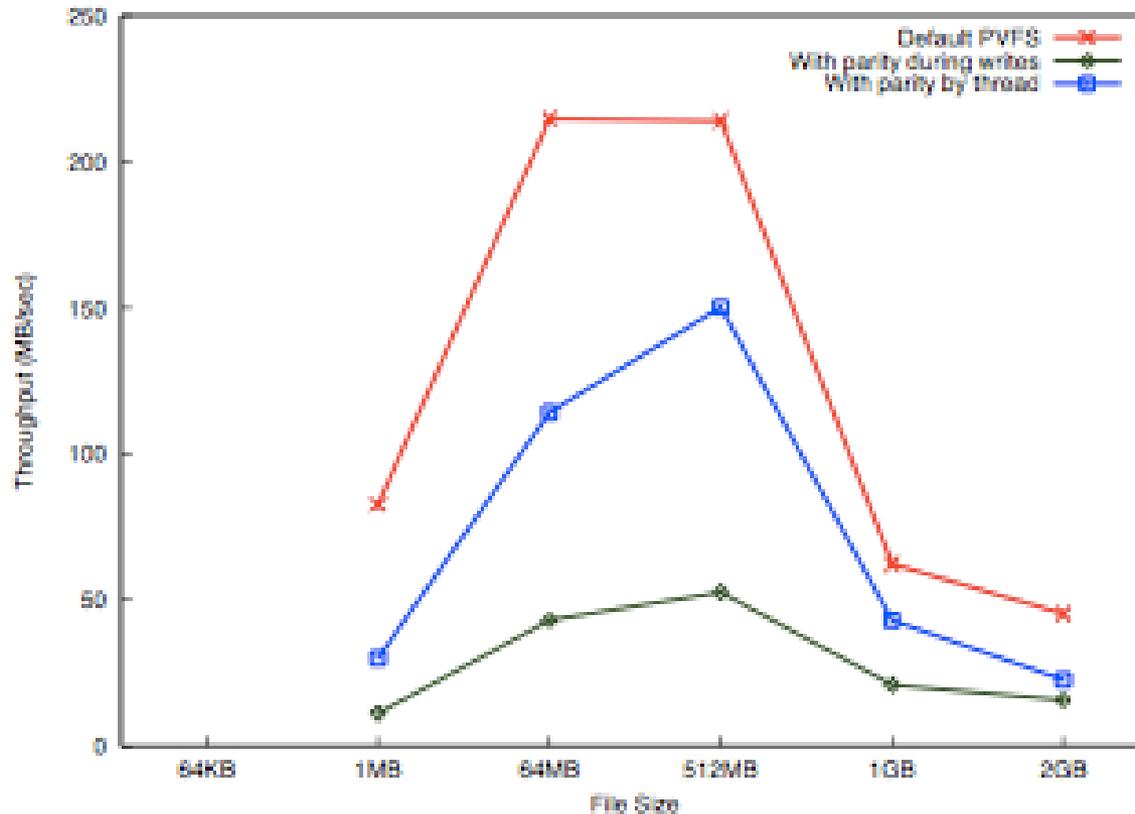
Active OSD for File Systems

- Move some file system functionality to OSD
- Allow applications to directly access active OSD
 - Override file system functionality
 - e.g. – data attributes lookup, colocation of related files
- Metadata partitioning and migration
- Node to node communications

Active OSD for Reliability

- XDWRITE/XDPARITY SCSI commands allow for parity calculations at the disk rather than at the client
- Same technique can be used with OSD
 - Use active code to calculate parity or more complicated redundancy mechanism (e.g. erasure codes)
- Handle client failures
 - Inconsistency in the middle of stripe write
 - OSDs can maintain information about “dirty” data

Active OSD for Reliability



Application aware storage

- Application-specific policy decisions
 - Redundancy, backup, QoS, encryption, etc.
 - Enabled with attributes
 - ATtribute-based Extendable Storage
 - Enabled with active storage
 - Applications can download code to enforce particular policy decisions

Active OSD for Backup

- Use attributes to specify files that need to be backed up
- Use active code on OSD to transfer backup data directly to backup server
 - No need to read data to client – third-party data transfer (NDMP)

Autonomic storage using Active OSD

- Automatic rebalancing and migration
- Quality of service monitoring
- Application-level optimization
 - Database tuning

Summary

- Develop active object storage architecture
- Active storage for improved file system performance
- Active storage for improved file system reliability
- Application aware storage using OSD/active storage
- Acknowledgements: NSF HECURA CCF-093787