

Ceph: A Scalable, High-Performance Distributed File System

Sage A. Weil, Scott A. Brandt,
Ethan L. Miller, Darrel D. E. Long

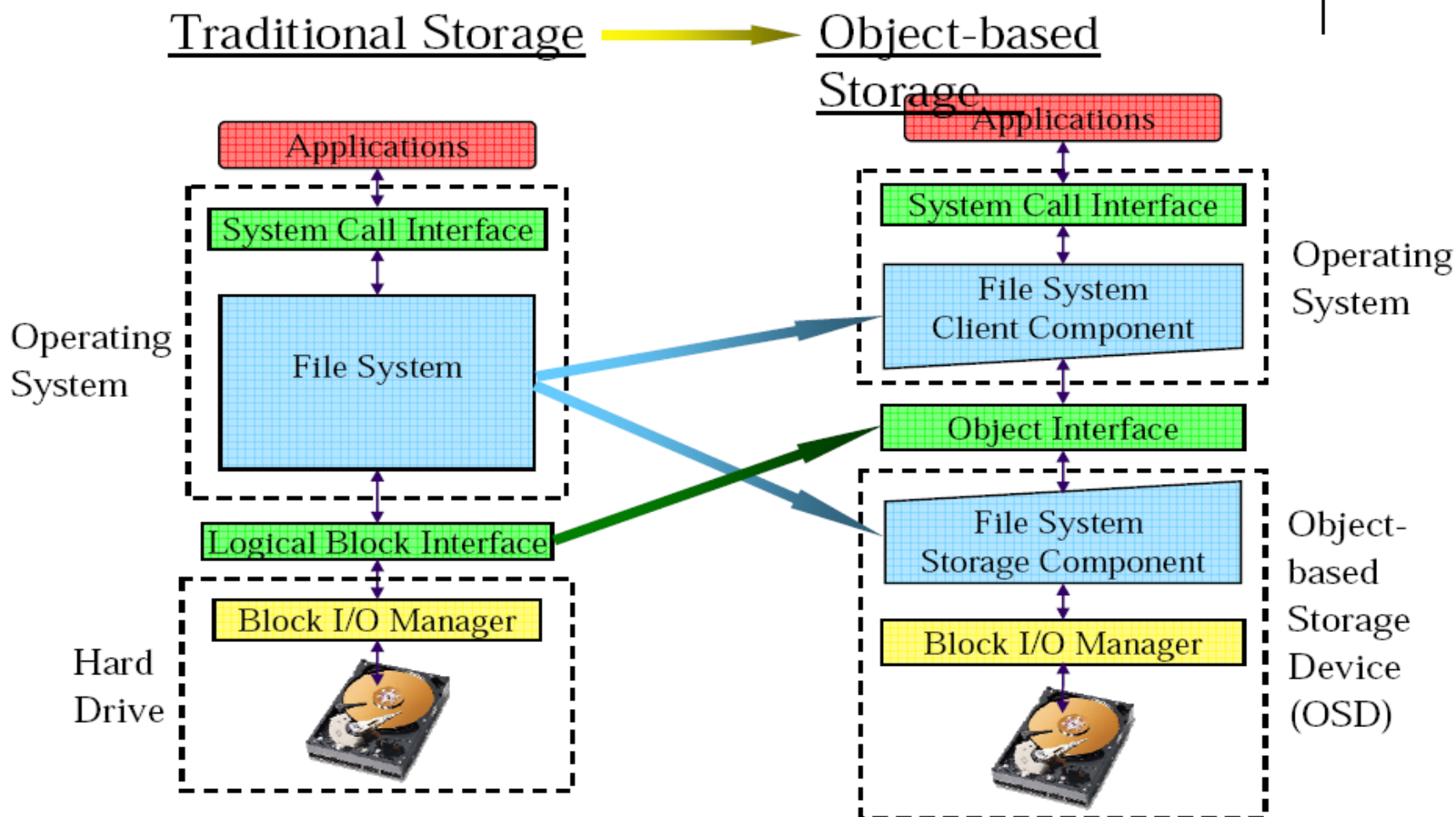
Contents

- Goals
- System Overview
- Client Operation
- Dynamically Distributed Metadata
- Distributed Object Storage
- Performance

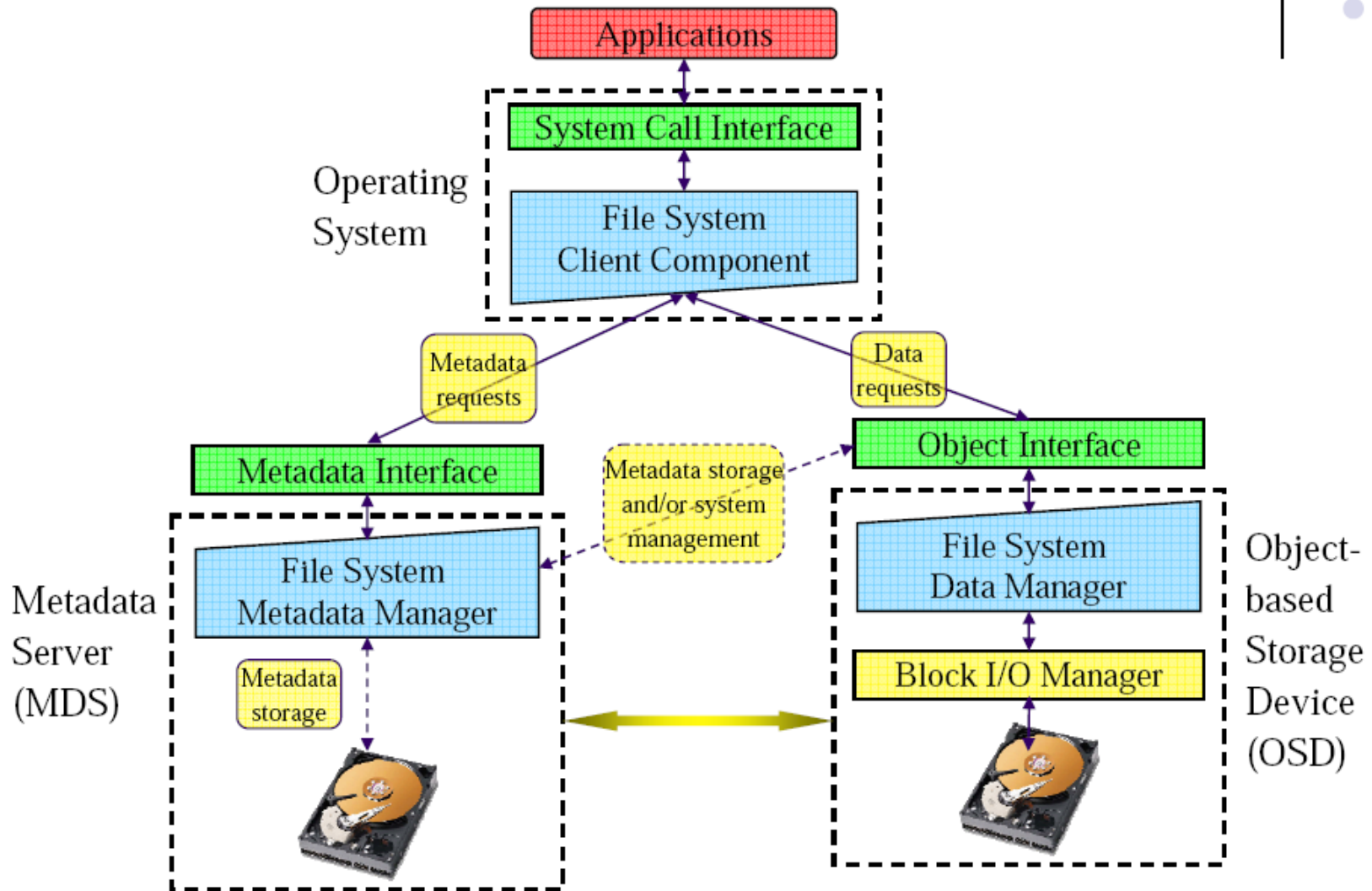
Goals

- Scalability
 - Storage capacity, throughput, client performance. Emphasis on HPC.
- Reliability
 - “...failures are the norm rather than the exception...”
- Performance
 - Dynamic workloads

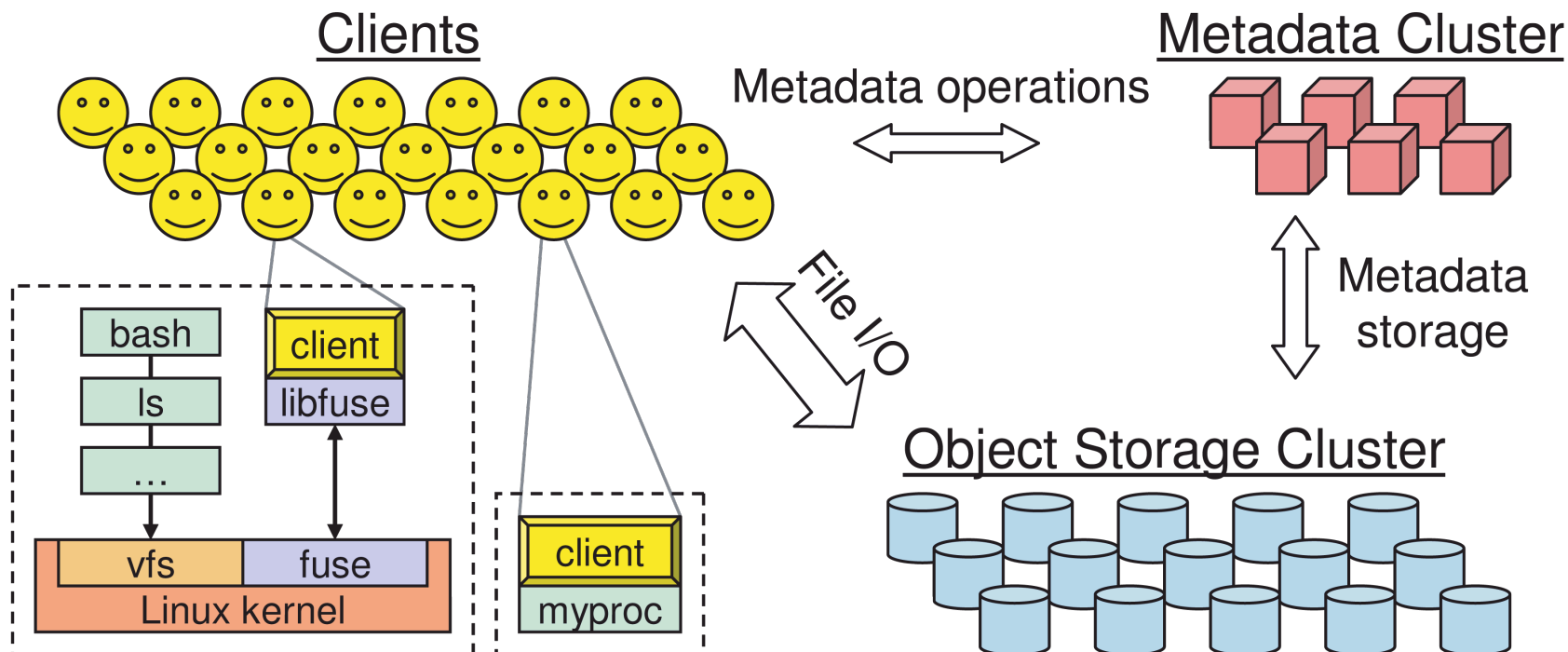
First Key Idea: Object-based Storage



Second Key Idea: Decoupled Data and Metadata



System Overview



Key Features

- Decoupled data and metadata
 - CRUSH
 - Files striped onto predictably named objects
 - CRUSH maps objects to storage devices
- Dynamic Distributed Metadata Management
 - Dynamic subtree partitioning
 - Distributes metadata amongst MDSs
- Object-based storage
 - OSDs handle migration, replication, failure detection and recovery

Client Operation

- Ceph interface
 - Nearly POSIX
 - Decoupled data and metadata operation
- User space implementation
 - FUSE or directly linked

Client Access Example

1. Client sends *open* request to MDS
2. MDS returns capability, file inode, file size and stripe information
3. Client read/write directly from/to OSDs
4. MDS manages the capability
5. Client sends *close* request, relinquishes capability, provides details to MDS

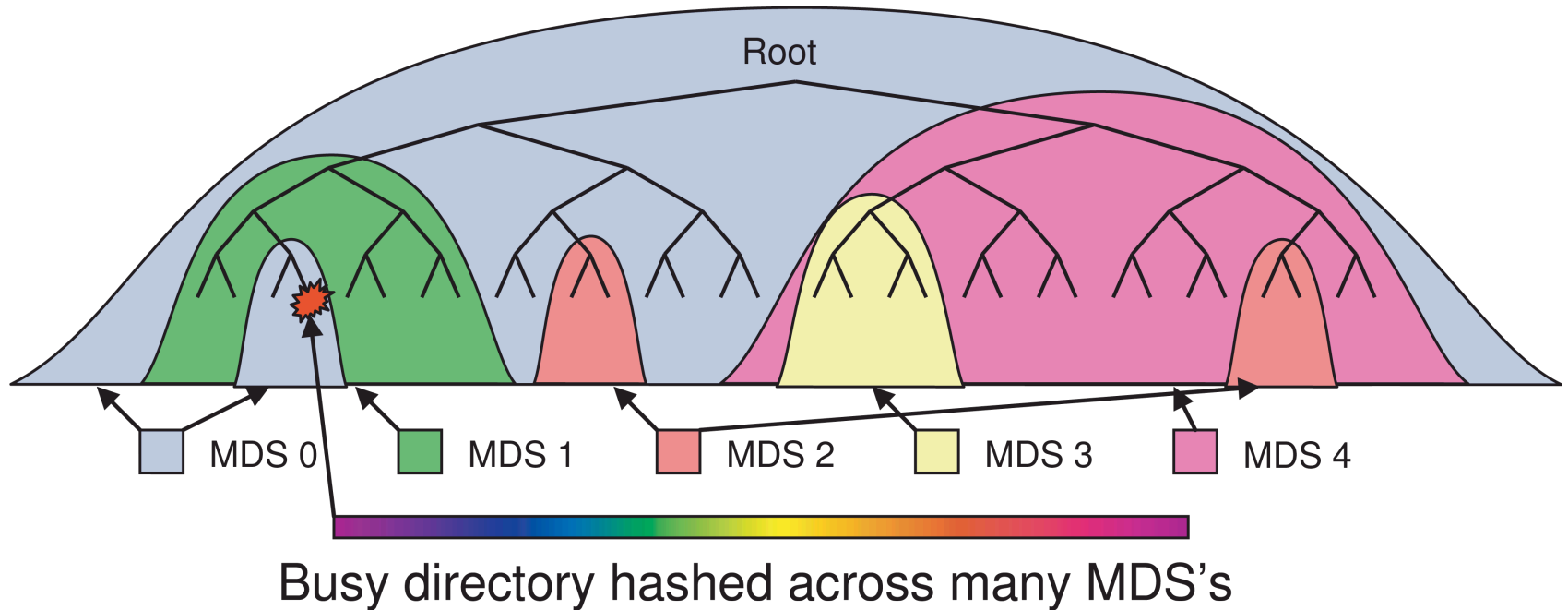
Synchronization

- Adheres to POSIX
- Includes HPC oriented extensions
 - Consistency / correctness by default
 - Optionally relax constraints via extensions
 - Extensions for both data and metadata
- Synchronous I/O used with multiple writers or mix of readers and writers

Distributed Metadata

- “Metadata operations often make up as much as half of file system workloads...”
- MDSs use journaling
 - Repetitive metadata updates handled in memory
 - Optimizes on-disk layout for read access
- Adaptively distributes cached metadata across a set of nodes

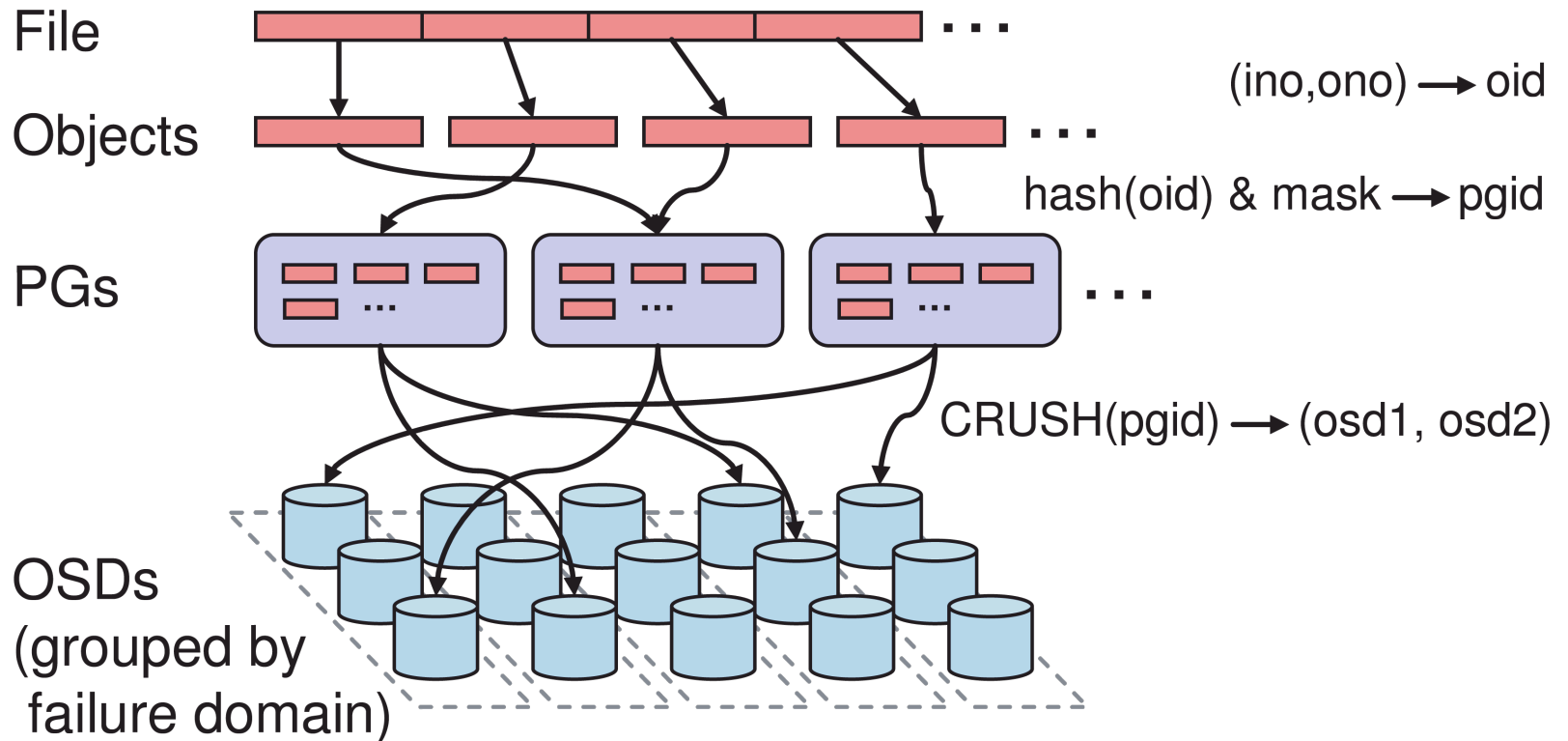
Dynamic Subtree Partitioning



Distributed Object Storage

- Files are split across objects
- Objects are members of placement groups
- Placement groups are distributed across OSDs.

Distributed Object Storage

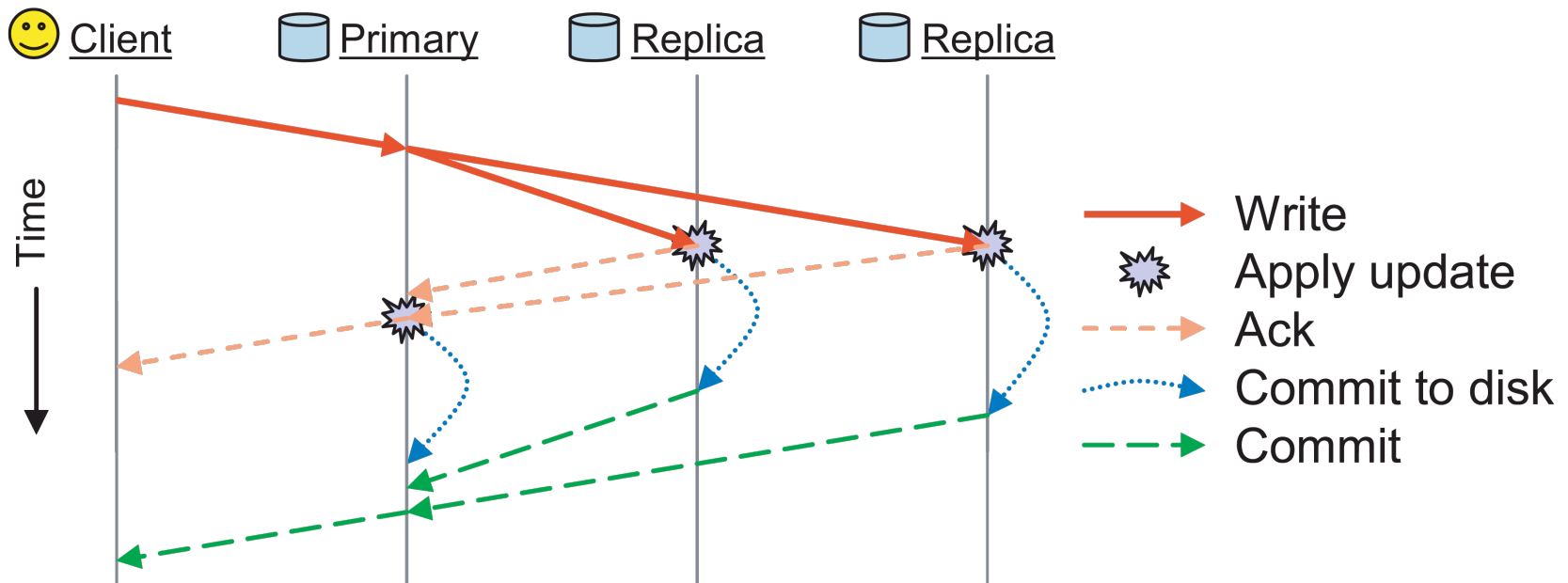


CRUSH

- $\text{CRUSH}(x) \rightarrow (\text{osd}_{n1}, \text{osd}_{n2}, \text{osd}_{n3})$
 - Inputs
 - x is the placement group
 - Hierarchical cluster map
 - Placement rules
 - Outputs a list of OSDs
- Advantages
 - Anyone can calculate object location
 - Cluster map infrequently updated

Replication

- Objects are replicated on OSDs within same PG
 - Client is oblivious to replication



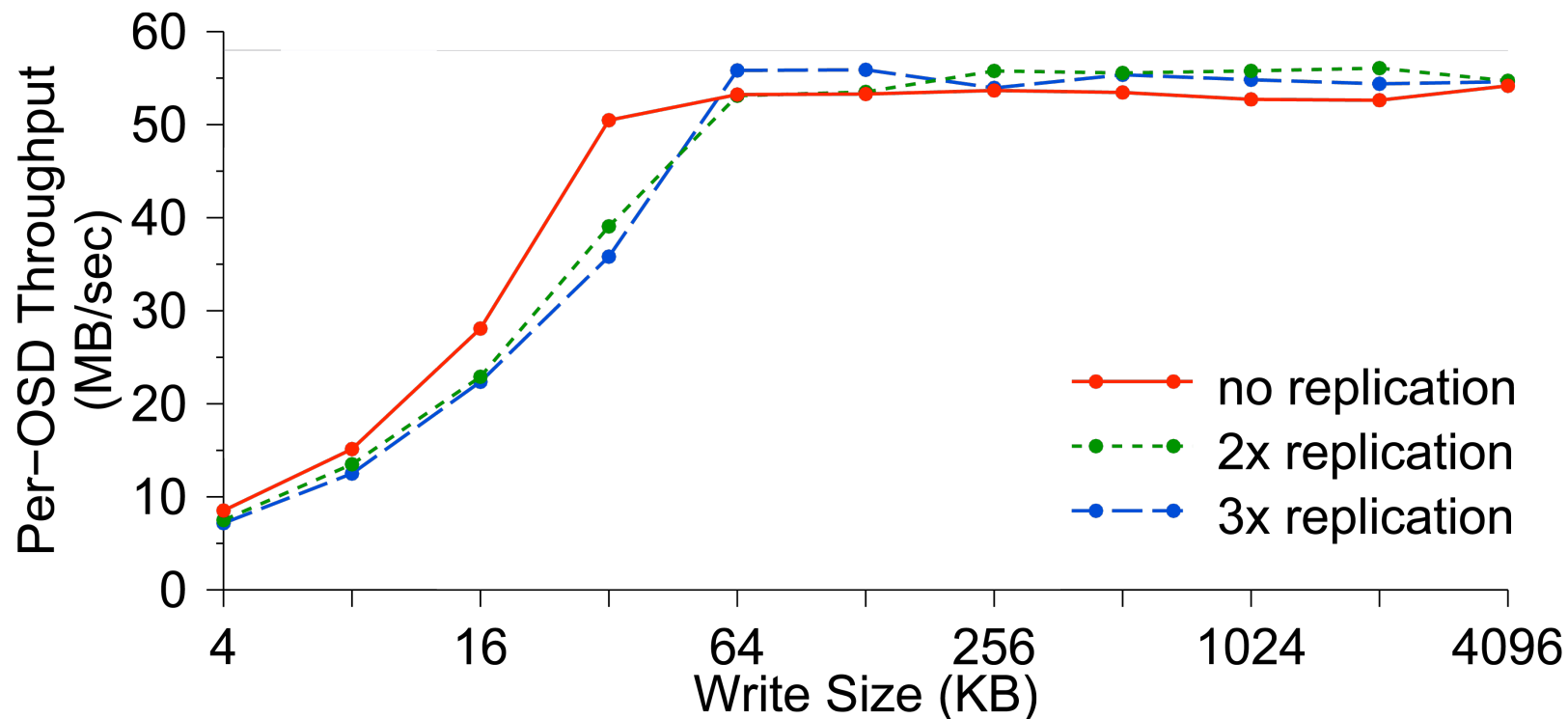
Failure Detection and Recovery

- *Down and Out*
- Monitors check for intermittent problems
- New or recovered OSDs peer with other OSDs within PG

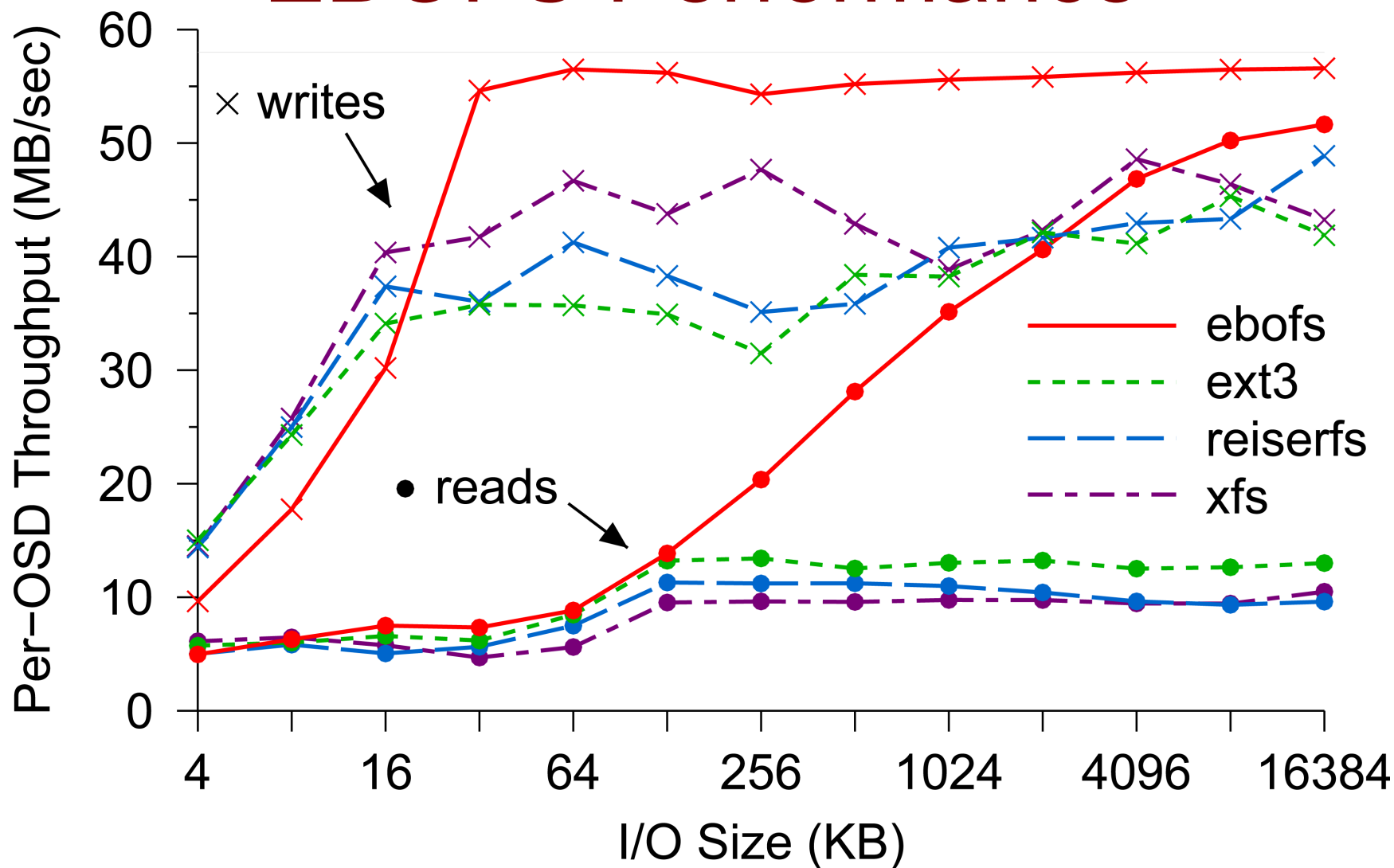
Conclusion

- Scalability, Reliability, Performance
- Separation of data and metadata
 - CRUSH data distribution function
- Object based storage

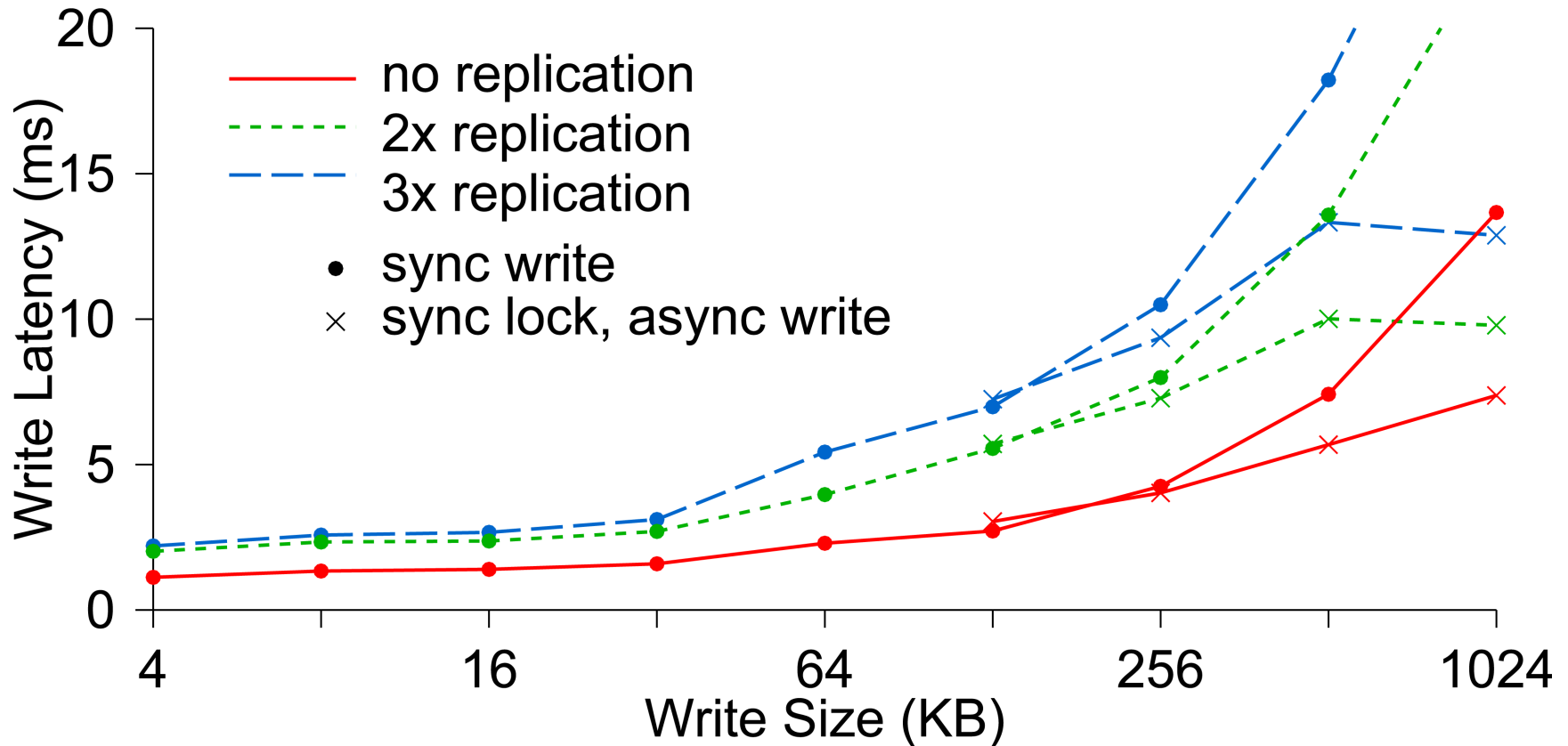
Per-OSD Write Performance



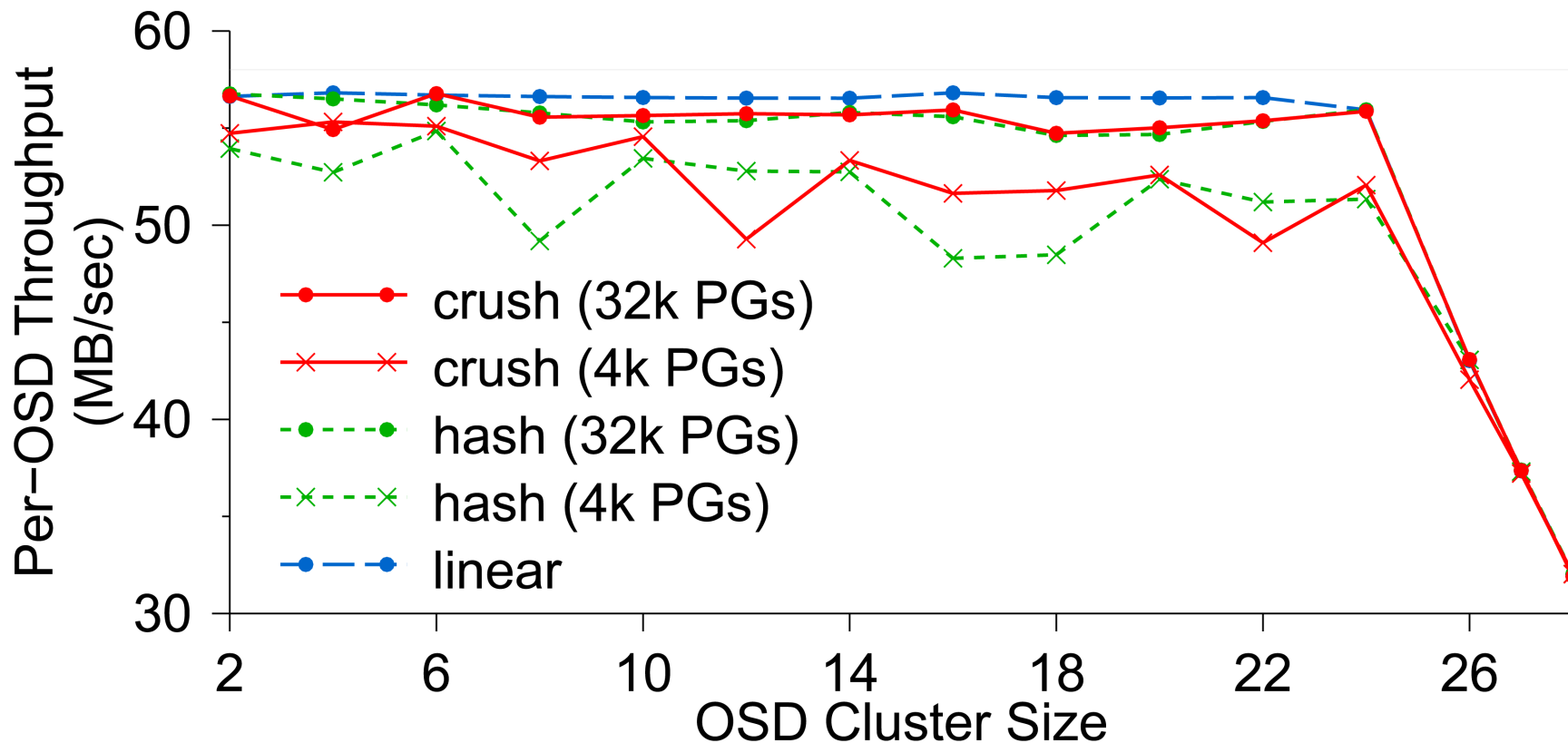
EBOFS Performance



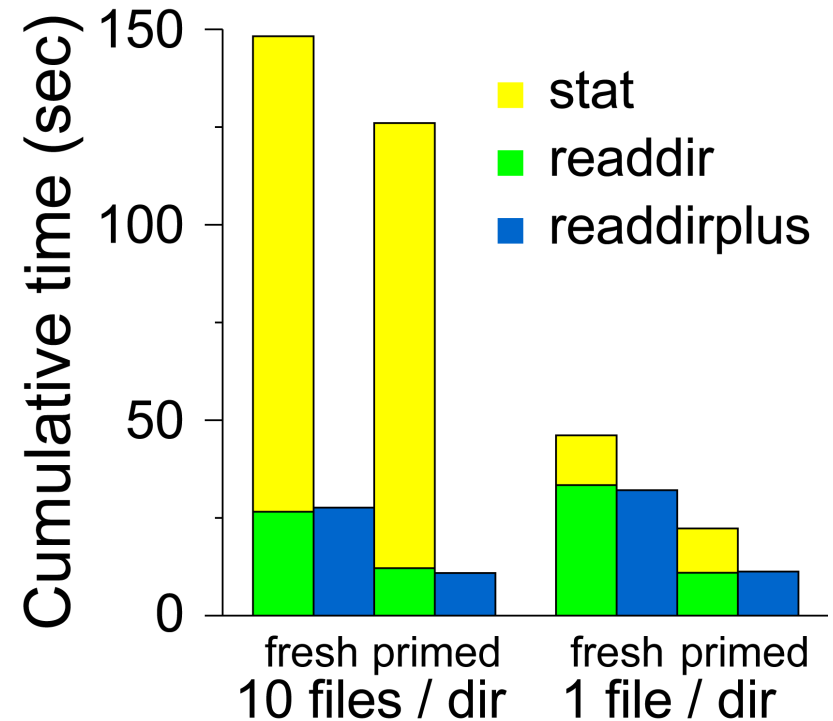
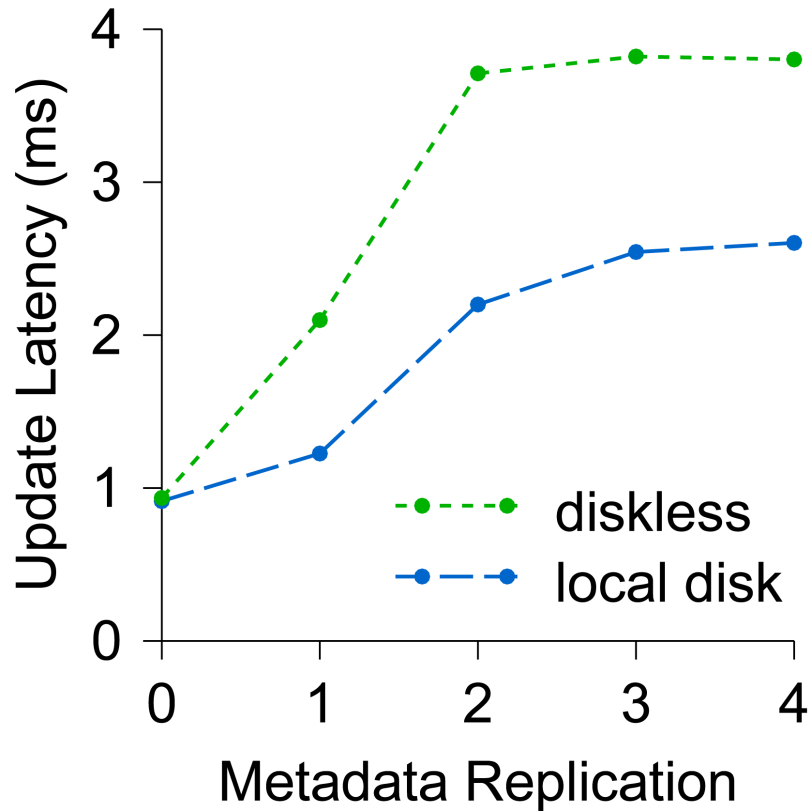
Write Latency



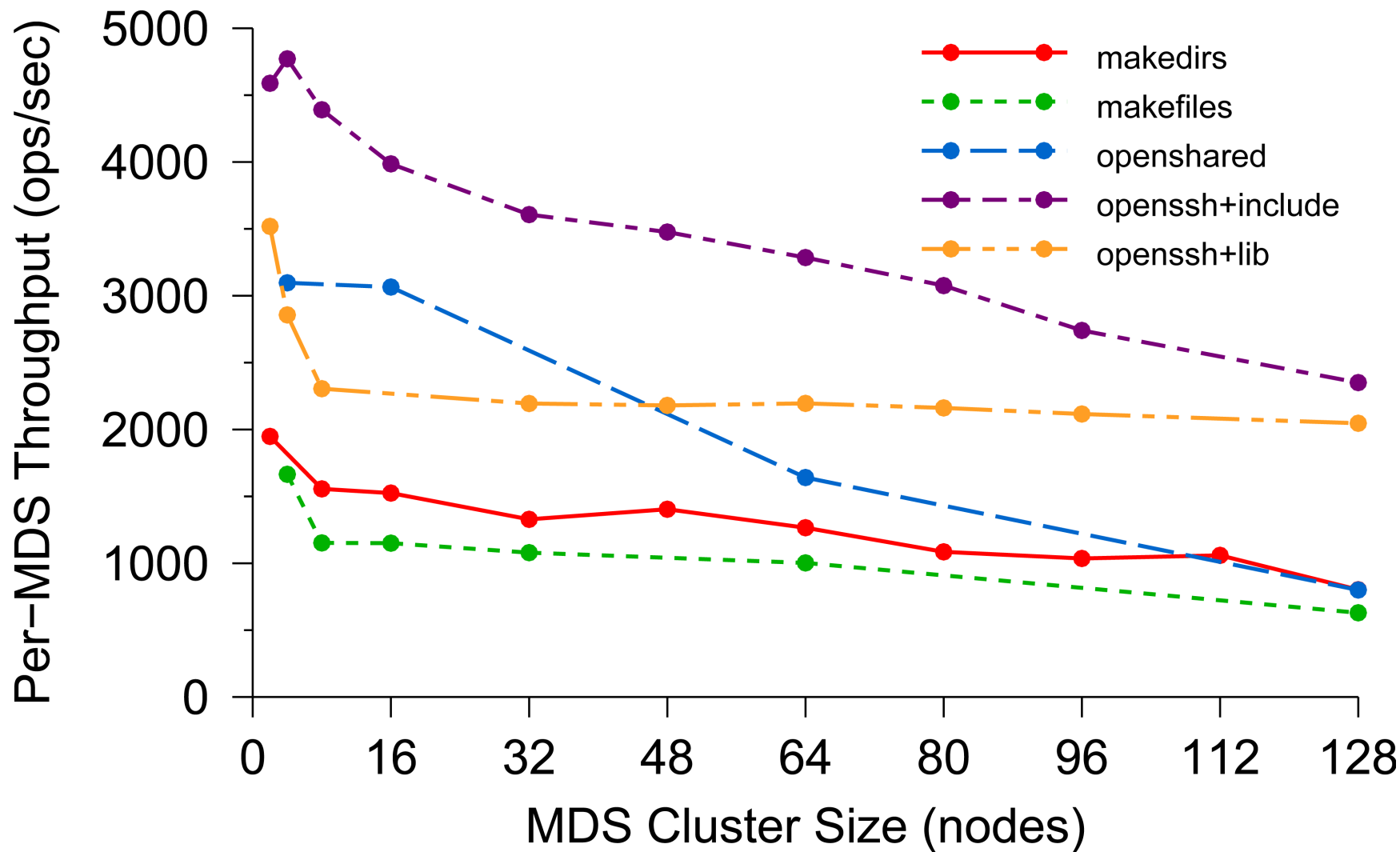
OSD Write Performance



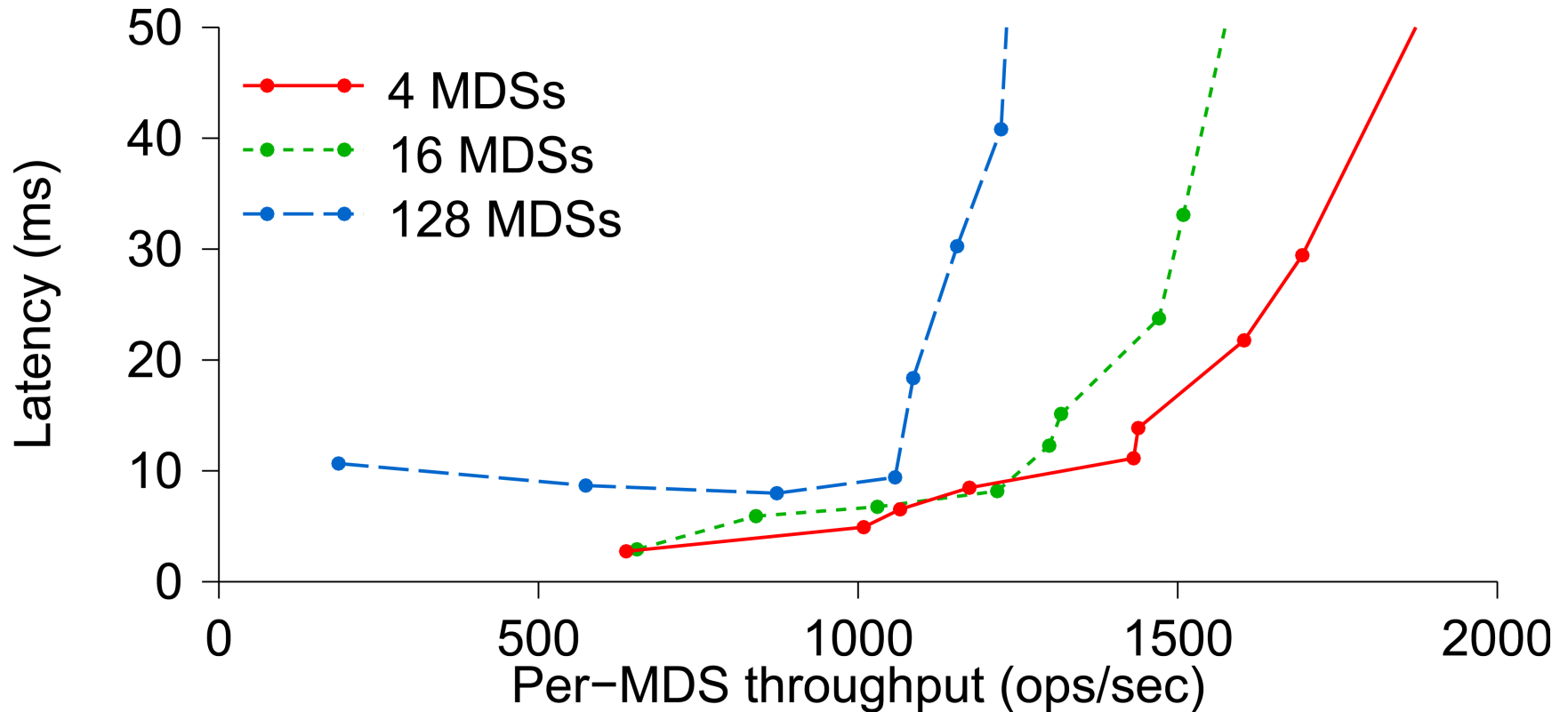
Diskless vs. Local Disk



Per-MDS Throughput



Average Latency



Related Links

- OBFS: A File System for Object-based Storage Devices
 - ssrc.cse.ucsc.edu/Papers/wang-mss04b.pdf
- OSD
 - www.snia.org/tech_activities/workgroups/osd/
- Ceph Presentation
 - <http://institutes.lanl.gov/science/institutes/current/ComputerScience/ISSDM-07-26-2006-Brandt-Talk.pdf>
 - Slides 4 and 5 from Brandt's presentation

Acronyms

- **CRUSH**: Controlled Replication Under Scalable Hashing
- **EBOFS**: Extent and B-tree based Object File System
- **HPC**: High Performance Computing
- **MDS**: MetaData server
- **OSD**: Object Storage Device
- **PG**: Placement Group
- **POSIX**: Portable Operating System Interface for uniX
- **RADOS**: Reliable Autonomic Distributed Object Store