# RADOS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters

Sage Weil,
Andrew Leung, Scott Brandt, Carlos Maltzahn
{sage,aleung,scott,carlosm}@cs.ucsc.edu
*University of California, Santa Cruz*

# Outline

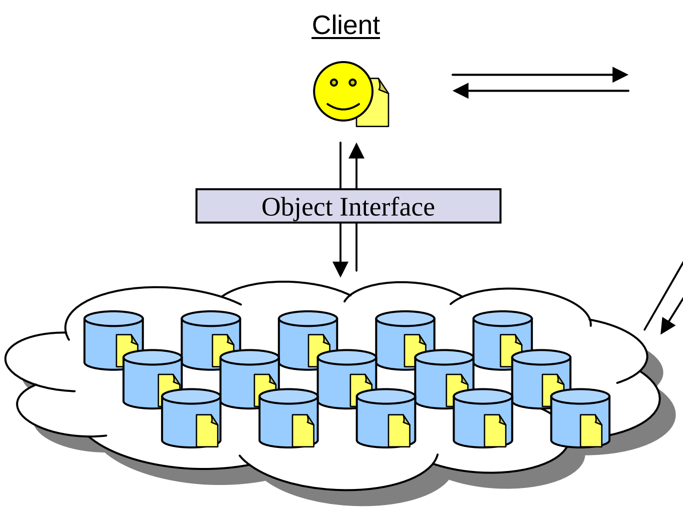
- Overview of architecture
- Scalable cluster management
- Current implementation
- Current research areas

# Intelligent Storage Clusters

- Current brick/object-based storage systems
  - Combine CPU, memory, network, disk/RAID
  - Move low-level allocation, security to devices
  - *Passive* storage targets
- RADOS
  - Reliable, Autonomic Distributed Object Store
    - Developed as part of Ceph distributed file system
  - Allows OSDs to *actively* collaborate with peers
    - Data replication (across storage nodes)
    - Failure detection
    - Data migration, failure recovery
  - High-performance, reliability, and scalability
    - Utilize peer-to-peer-like protocols
    - Strong consistency

# Overview

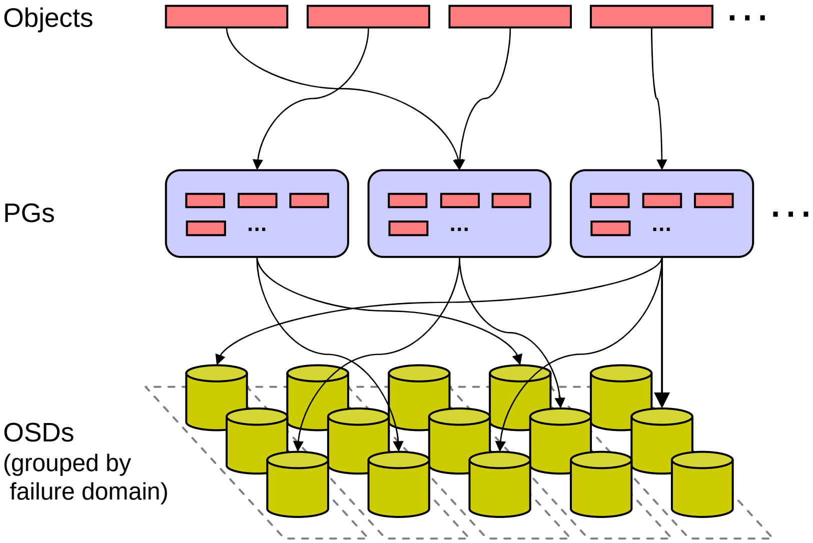
- Clients
  - Expose simple object-based interface to FS or application
  - Treat storage cluster as single logical object store
- Small monitor cluster
  - Stores configuration information and cluster state
  - Manages cluster by manipulating the *cluster map*
- Storage nodes (OSDs)
  - Store objects—local disk or RAID
  - Leverage information in cluster map to act semi-autonomously



# Data distribution

- Objects mapped to *placement groups* (PGs)
  - pg_id = hash(object_id) & mask
- PGs mapped to set of OSDs
  - PG contents replicated
  - ~100 PGs per OSD

- Compactly specify PG mapping with CRUSH *function*
  - Fast—even for huge clusters
  - Stable—adding/removing OSDs moves few PGs
  - Reliable—replicas span failure domains
  - ...everything you'd normally want to do, without and explicit table



# Cluster map

- Cluster map
  - State of storage node state (up/down, network address)
  - Complete (but compact) specification of data placement
  - Replicated by all nodes (OSDs, clients, and monitors)
- Managed by small, reliable monitor cluster
  - Paxos-based replication algorithm
  - Augmented to distribute map reads and updates across monitor cluster
- Monitors issue map updates in response to failures, node additions, etc.
  - Each map version has unique *epoch*
  - Small *incremental* updates describe changes between successive epochs
  - Updates are *lazily* propagated

# Map update distribution

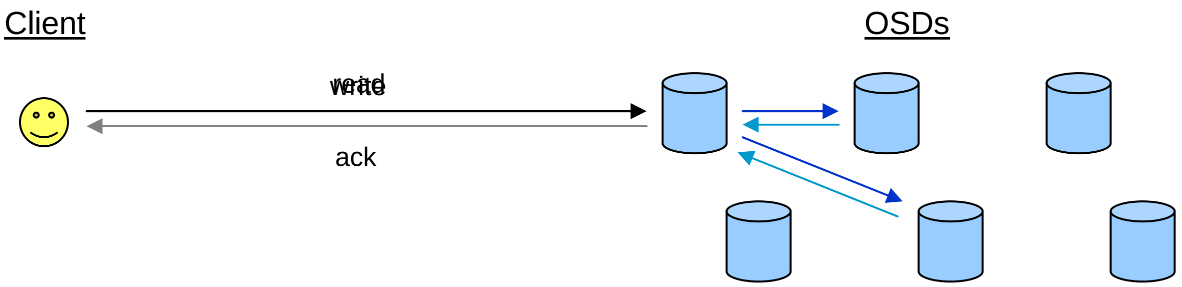
- All messages tagged with sender’s current map epoch
  - Peers agree on current data distribution, respective roles
  - Peers share incremental map updates when versions differ
- Clients
  - Direct requests based on current copy of the map
  - OSDs share recent map updates with client if it is out of date
    - Client redirects any affected outstanding requests
- OSDs
  - Remember last observed version at each OSD peer
  - Preemptively share updates when communicating
  - Periodic heartbeat messages between OSDs ensure that updates propagate in O(log n) time

# Scalable consistency

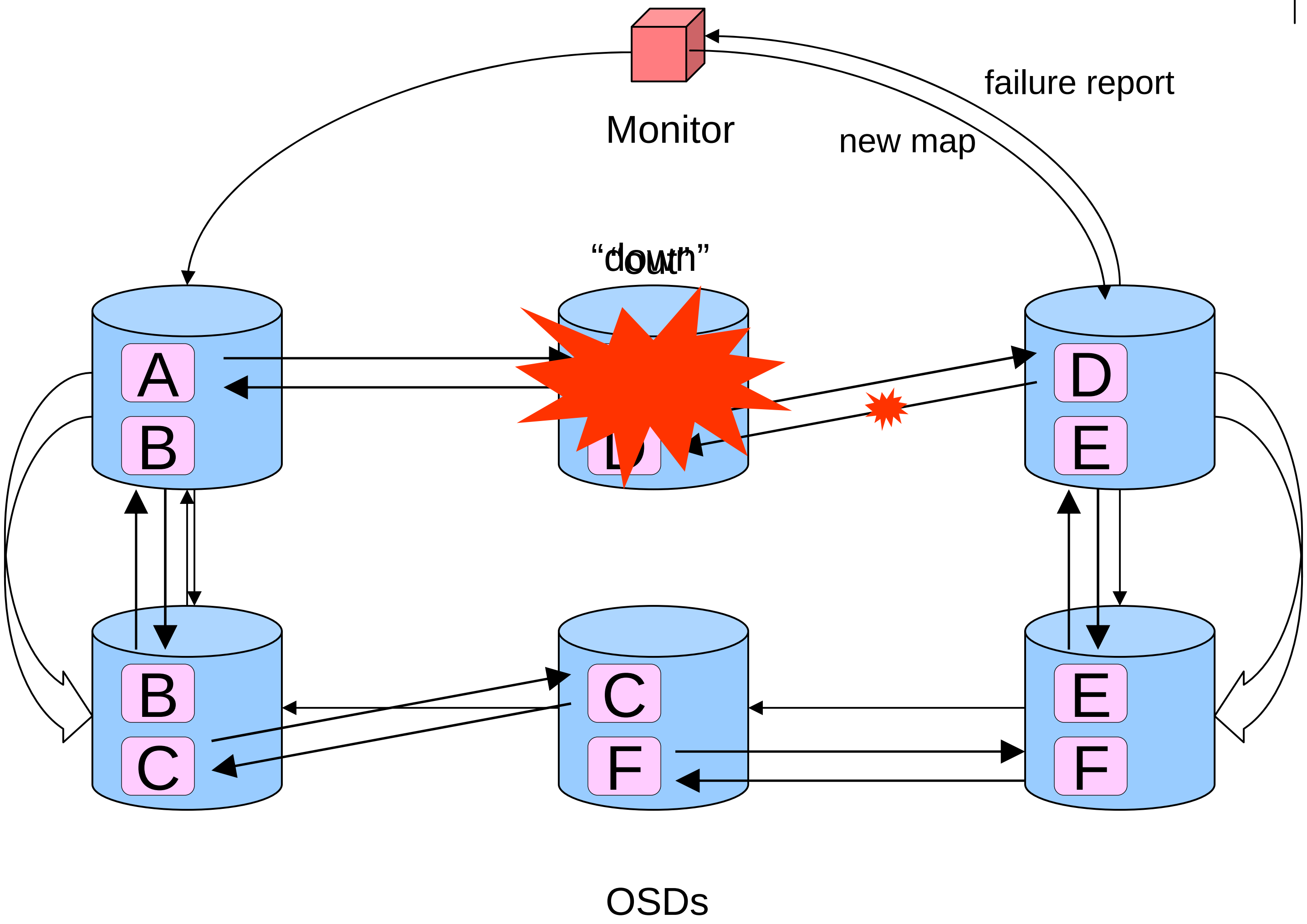
- Maps allow communicating OSDs and clients to agree on
  - Mapping of objects to PGs
  - Mapping of PGs to OSDs
- Map updates simply change specification of data distribution
- When OSDs receive a map update
  - Check if mapping for any locally store PGs has changed
  - Robust *peering* algorithm to resynchronize with new peers
  - Actively collaborate with peers to realize the new distribution
- OSDs autonomously manage
  - Data replication
  - Failure detection
  - Failure recovery

# Data Replication

- Reads are serviced by one OSD
- Writes are replicated by OSDs
  - Shifts replication bandwidth to OSD cluster
  - Simplifies client protocol
- OSDs know proper distribution of data from the cluster map



# Failure Detection and Recovery



# Current implementation

- Developed as part of the Ceph file system
  - Objects store blobs of binary data
  - Extent (start, length) based read/write interface
  - N-way object replication for data safety
    - Short-term PG logs for fast recovery from intermittent failures (e.g. node reboots)
  - Fast, simple, and scalable
  - LGPL: http://ceph.sourceforge.net/

# Current research

- Alternative object interfaces
  - Key/value storage
    - Dictionary/index management
    - Distributed B-tree or B-link trees
  - Scalable producer/consumer (FIFO) queues
- Fine-grained load distribution
  - CRUSH pseudorandom distribution subject to statistical variance in OSD utilizations
  - Offload read workload onto replicas for hot objects
  - Vary object replication level

# Current research (2)

- Alternative redundancy strategies
  - Parity-based (RAID) encoding of objects across OSDs in each PG
  - Dynamic adjustment of object encoding (replication vs RAID) in response to load
    - Replication for performance, RAID for space efficiency
- Object-granularity snapshot
  - As systems scale, volume-granularity snapshots are increasingly limiting
  - Facilitate fine-grained snapshot functionality
    - Introduce *clone()* with copy-on-write semantics
    - Part of effort to introduce snapshots on arbitrary subdirectories to Ceph file system

# Questions…

*Research supported by:*

*LLNL*

*DOE, NSF, ISSDM*

*http://ceph.sourceforge.net/*

# Scalability

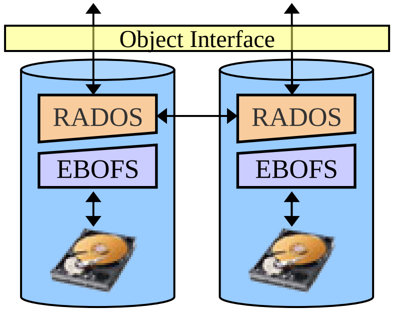
- Failure detection and recovery are distributed
  - Centralized monitors used *only* to update map
- Maps updates are propagated by OSDs
  - Epidemic-style propagation, with bounded overhead
  - No monitor broadcast necessary
- Monitor cluster can scale to accommodate flood of messages

- Identical “recovery” procedure used to respond to all map updates
  - (Node failure, cluster expansion, etc.)
  - OSDs always collaborate to realize the newly specified data distribution
  - Consistent data access preserved

# Intelligent OSD Cluster

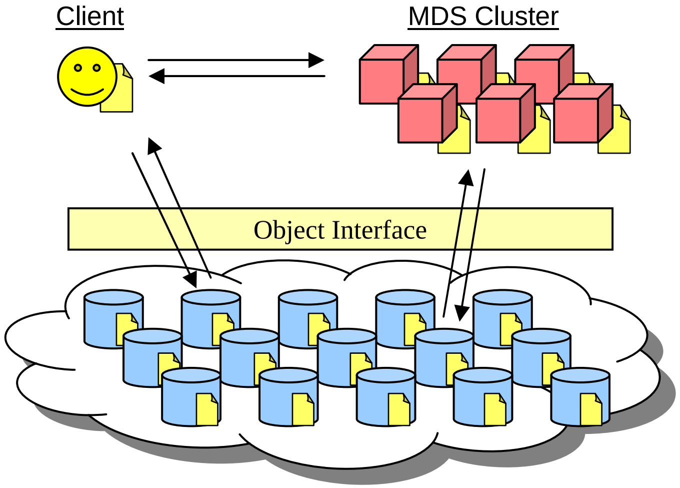
- Interface resembles T10, but similarity ends there
  - T10 OSDs are passive disks—only respond to read/write
  - Ceph OSDs actively collaborate with peers for failure detection, data replication, and recovery

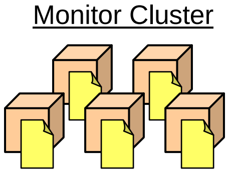
- Each OSD is a user-level process
  - EBOFS: object file system
    - Handles local object storage
    - Transactions and async commit notification
    - Extents, copy-on-write, shadowed BTrees, etc.
  - RADOS: reliable autonomic distributed object store
    - Exposes object interface
    - Manages consistent replication with peers
    - Re-replicates object data in response to failure



# Reliable, Distributed Object Store

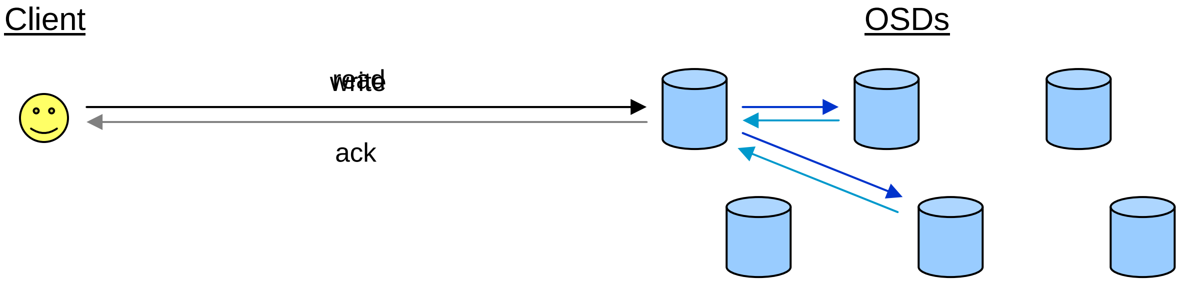
- Cluster viewed as a single reliable object store
  - Initiators (client, MDS) direct requests based on *cluster map*
    - Participating OSDs and their status (up/down)
    - CRUSH mapping function—consistent view of data layout
  - Small monitor cluster manages master copy of map, issues updates
    - Manage cluster response to failure
    - Lazily propagated
  - Cluster can act autonomously and intelligently
    - map update propagation, replication, failure detection, recovery





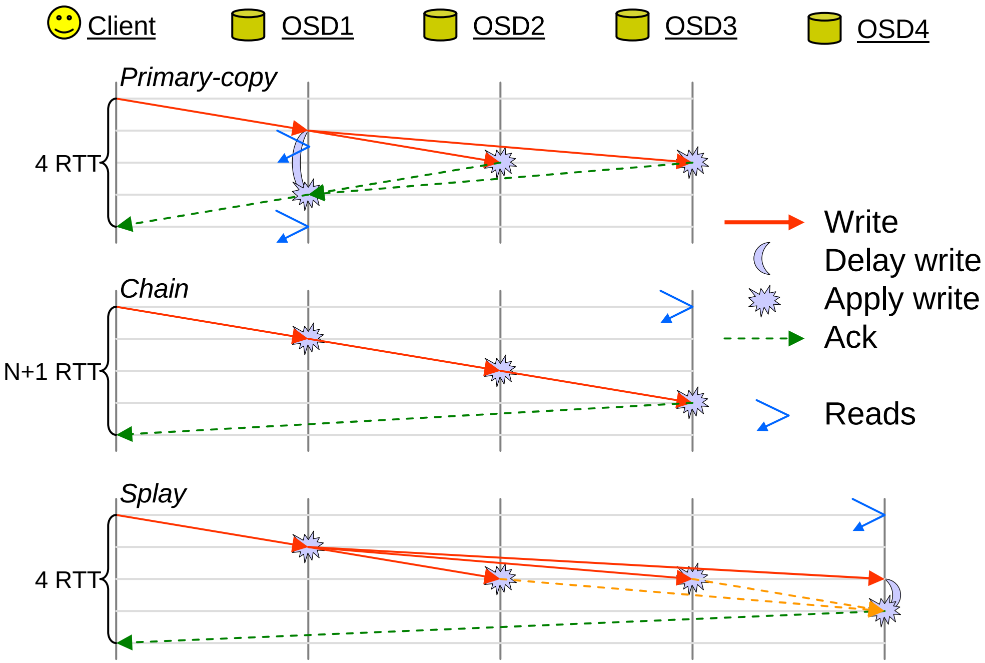
# Data Replication

- Reads are serviced by one OSD
- Writes are replicated by OSDs
  - Shifts replication bandwidth to OSD cluster
  - Simplifies client protocol
- OSDs know proper distribution of data from the cluster map



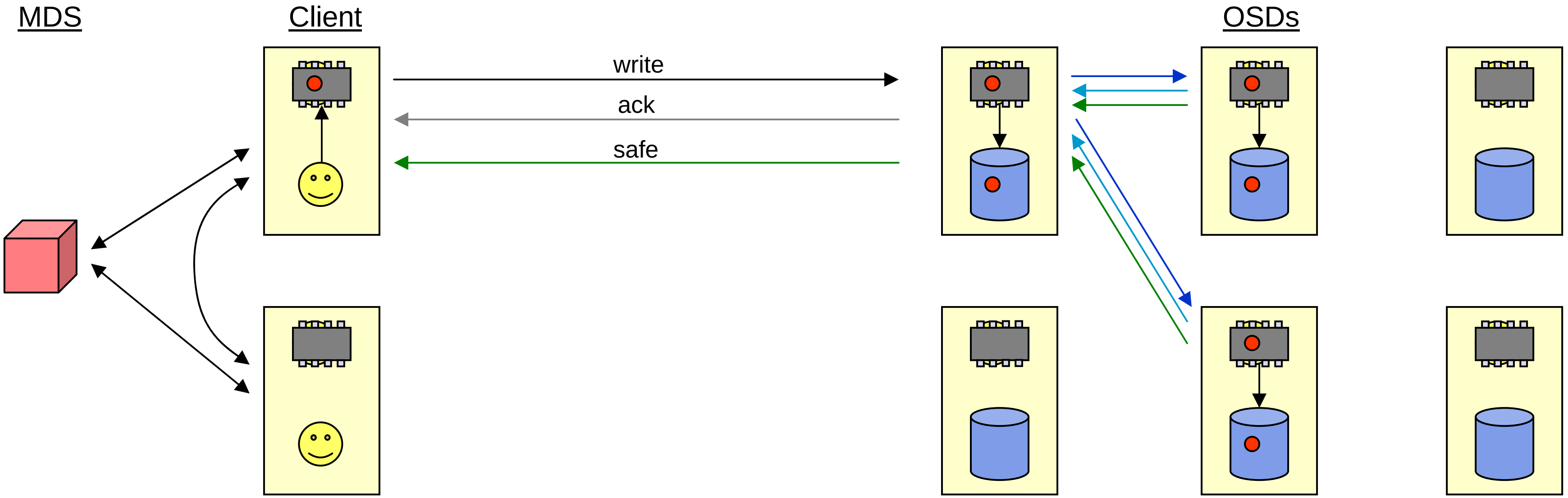
# Data Replication

- Three replication schemes are supported
  - Use different OSD to service read requests
  - Limit number of messages, lower update latency
- Fully consistent semantics in read/write write workloads, even with intervening failures



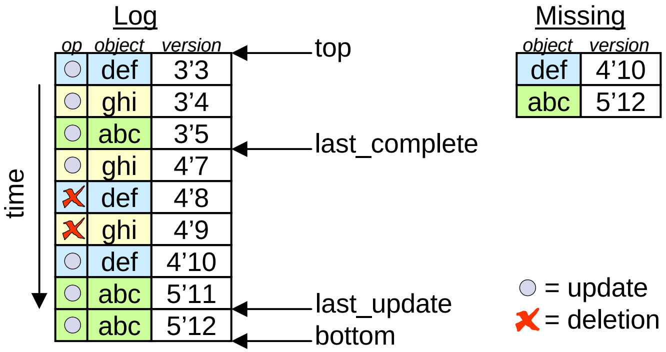
# Data Safety

- Two reasons we write data to a file system
  - **Synchronization** – so others can see it
  - **Safety** – so that data will be durable, survives power failures, etc.
- RADOS separates write acknowledgement into two phases
  - **ack** – write is serialized, and applied to all replica buffer caches
  - **safe** – all replicas have committed the write to disk



# Data Safety and Recovery

- Low-level object storage with EBOFS
  - Asynchronous notification of commits to disk
  - Copy-on-write
    - Failure reverts to fully consistent state—warp back in time
- Each PG maintains a log, object version
  - Replica resynchronization compares logs, identifies stale or missing objects
  - Fast recovery from intermittent failures (eg OSD reboots)



Log

| *op* | *object* | *version* |
|---|---|---|
| ○ | def | 3’3 |
| ○ | ghi | 3’4 |
| ○ | abc | 3’5 |
| ○ | ghi | 4’7 |
| ✗ | def | 4’8 |
| ✗ | ghi | 4’9 |
| ○ | def | 4’10 |
| ○ | abc | 5’11 |
| ○ | abc | 5’12 |

Missing

| *object* | *version* |
|---|---|
| def | 4’10 |
| abc | 5’12 |

○ = update
✗ = deletion