ceph distributed file system

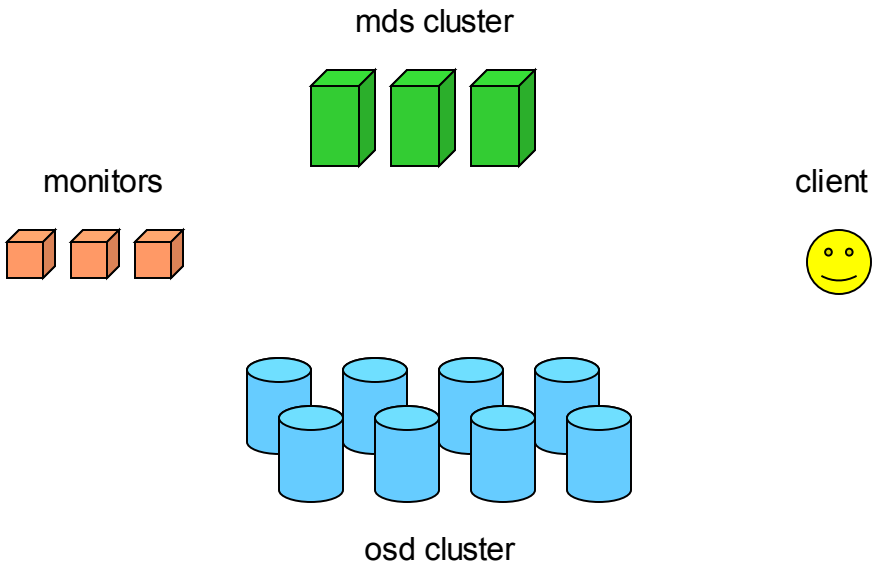
sage weil

lsf'08

outline

- quick overview
- kernel client
- object storage
- roadmap

architecture



- userspace daemons
  - monitors – config/state info
  - mds – namespace
  - osds – data/metadata
- cluster sizes are dynamic
- anything can(/will) fail

- osds are “intelligent”
  - consistent/safe replicated writes
  - failure recovery/data migration
- mds cluster
  - fast
  - scalable
  - highly available

mds

- embed inodes inside directory (first dentry)
  - load entire directory's inodes with single io
- long (100MB+) journal to aggregate changes and limit directory metadata writes
- adaptive hierarchical partitioning of workload across mds nodes
- replication across mds caches

osd

- objects replicated across dynamic cluster
  - flexible specification mapping *n* replicas across failure domains, tiered storage, etc.
- osds actively collaborate to handle
  - consistent replication
  - failure detection
  - data migration/recovery
  - strong replication consistency
- minimal central management
  - monitors update osdmap to reflect osd state changes
  - map distributed efficiently and lazily
- mds, clients treat cluster as single logical object pool

kernel client

- basically functional
  - still missing
    - parts of io path
    - sync mode (e.g. write sharing)
  - metadata behavior initially similar to nfs
    - lookup/revalidate succeed based on timeout
    - moving toward more stateful protocol with strong consistency
- lots to come later…
  - locking, directio, …
  - behave on clients with 32bit inos

some issues

- near-oom
  - more complicated communication model makes memory reservation difficult
  - semi-arbitrary osd cluster topology changes are possible
  - need to receive/process map updates
- dual write ack
  - ack -> serialized, in osd ram
  - commit -> safe on disk

object storage

- requirements
  - compound atomic transactions
  - async notification of commits
    - i.e. immediate return when cache is updated + eventual callback indicating update has flushed to disk
- currently use ebofs
  - works, performs well
  - generic auxiliary journaling
    - low latency commits, esp with NVRAM
  - will currently fall over for large volumes of small objects
  - more code to maintain
- alternatives
  - layer over existing file systems
    - fsync incurs degenerate behavior in most fs's
    - something more like a sync inotify?
  - libbtrfs, libzfs

roadmap

- “complete” kernel client
- usability, tools

- quotas
  - voucher based
  - distributed, scalable, etc.
- directory- or file-granularity snapshots
  - rely on per-object COW semantics in object store
  - versioned dentries in MDS

- http://ceph.sf.net
- ceph-devel@lists.sf.net
- #ceph on oftc.net