
whamcloud

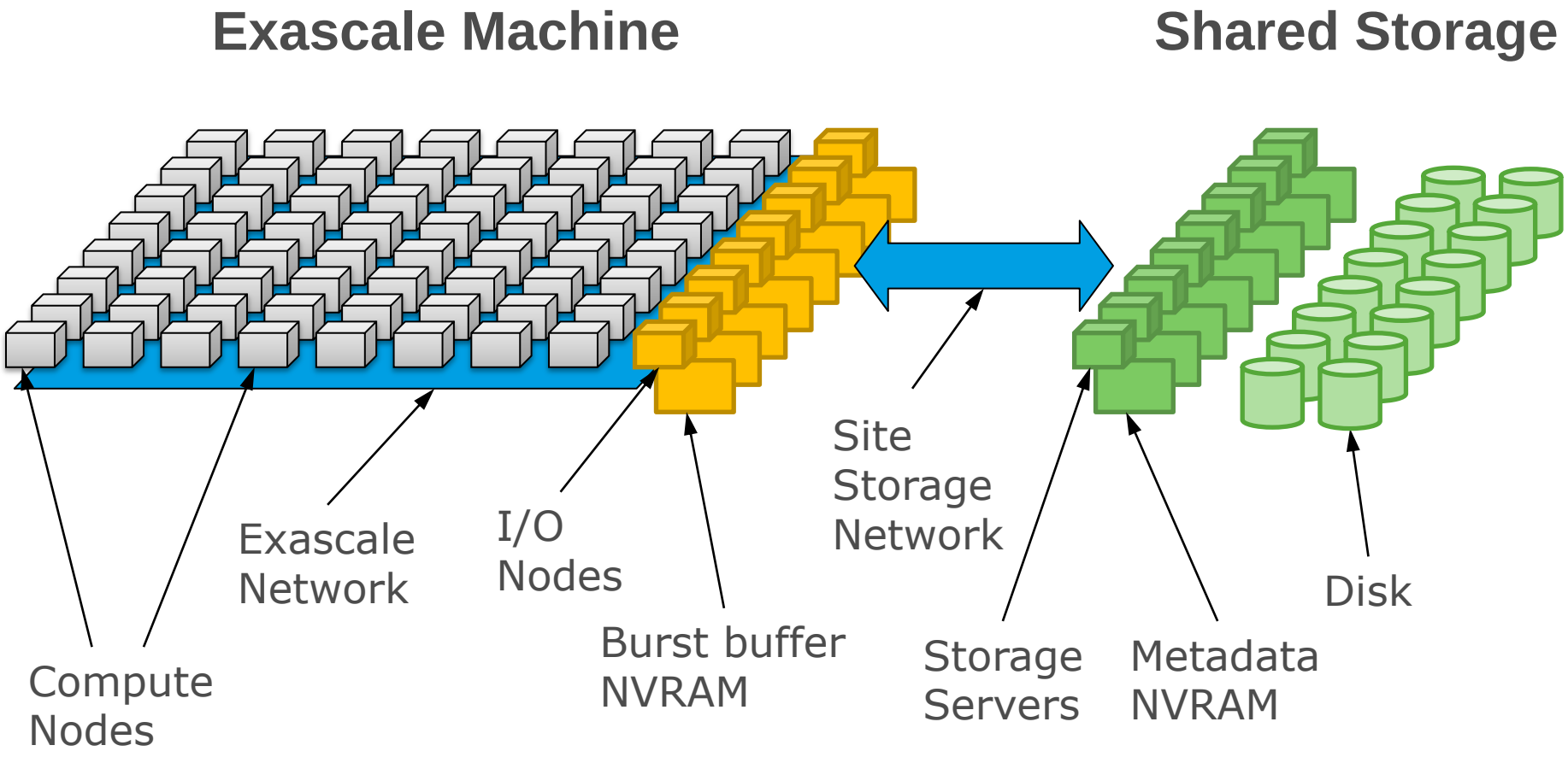
whamcloud

OpenSFS TWG
12th July 2012

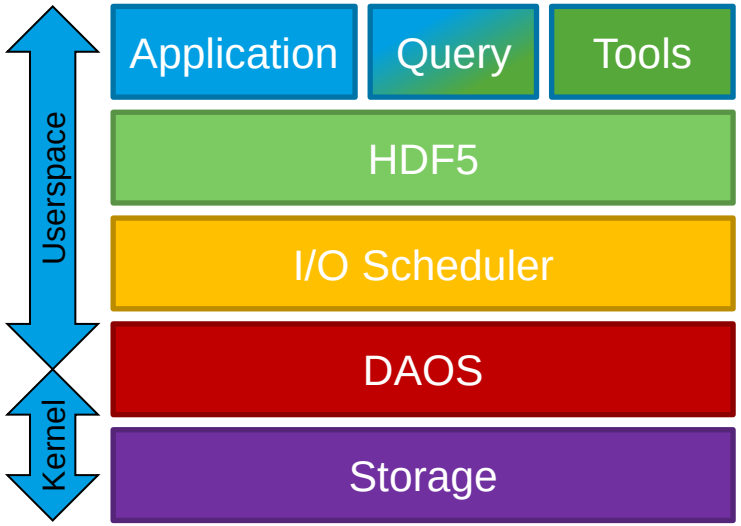
Fast Forward Exascale I/O

- Eric Barton
  CTO
  Whamcloud, Inc
  eeb@whamcloud.com

Exascale I/O Hardware Architecture

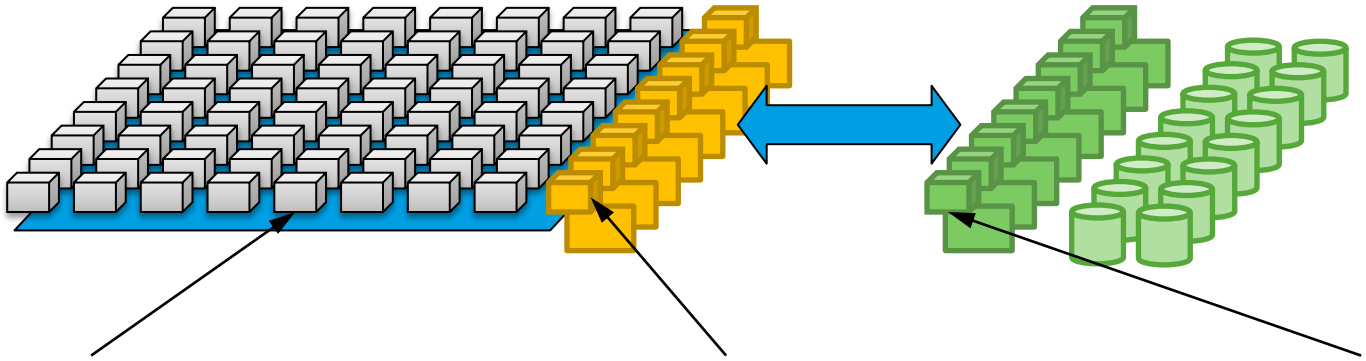


I/O stack



- HDF5
  - Object based application I/O
    - Application data + metadata
  - Indexes
    - Fast query / analysis
  - End-to-end application integrity
- I/O Scheduler
  - Object aggregation / layout optimization / caching
    - Application
      - Fragmented / misaligned application data
      - Massive burst data bandwidth / metadata operation rates
    - Storage system
      - Streaming I/O
- Distributed Application Object Storage
  - Scalable object storage
  - Guaranteed consistency thru failure

Exascale I/O Software Architecture



Compute Nodes

Application

HDF5 API

Function Shipping Client

I/O Nodes

Function Shipping Server

I/O Scheduler / BBMgr

DAOS API

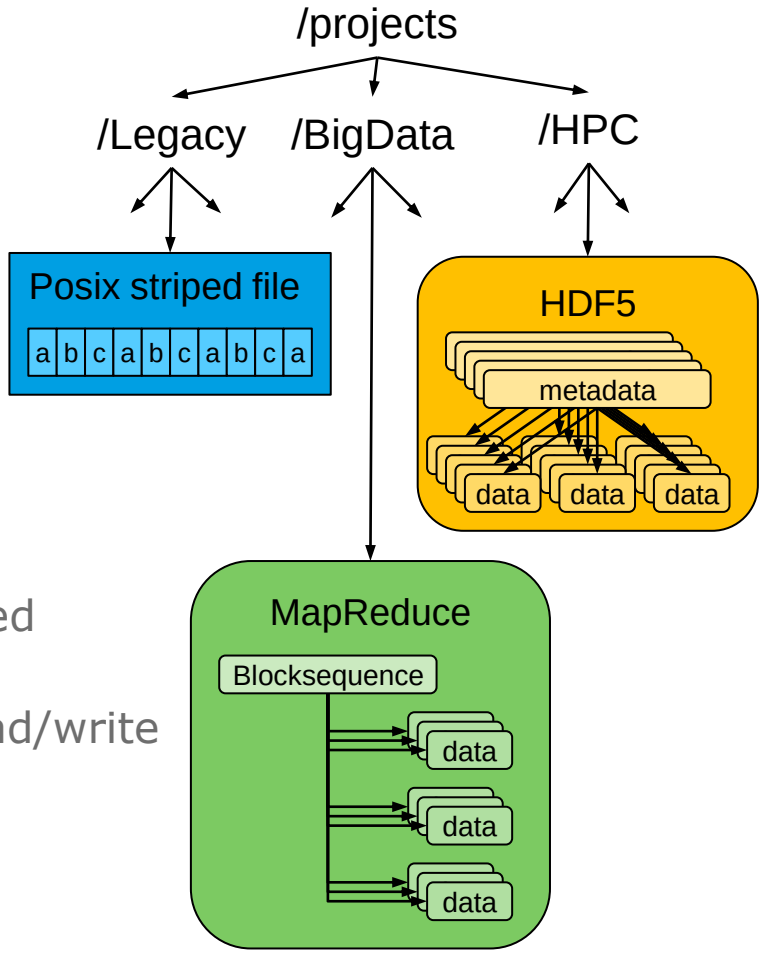
Lustre Client

Storage Servers

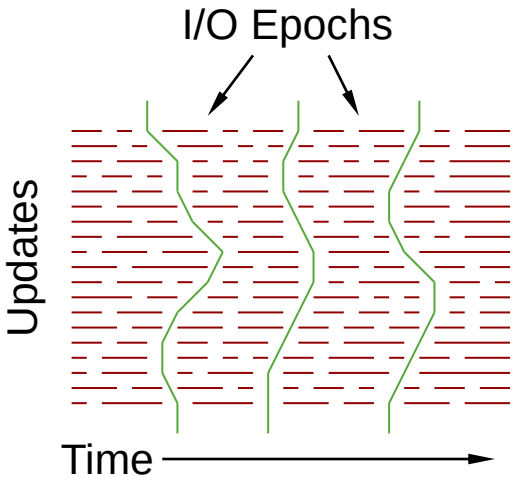
Lustre Server

Exascale Filesystem

- Conventional namespace
  - Administration, security, accounting
- Posix files
  - Legacy data and applications
  - Libraries, binaries, text
- DAOS Containers
  - Scalable object creation & I/O
    - Sharded object namespace
    - 10s of billions of objects distributed over thousands of OSSs
    - Share-nothing create/destroy, read/write
  - Resilient
    - Consistent / thru failure



Transactions



- (Meta)data consistency / integrity
  - Metadata at one level is data in the level below
    - Consistency required at all levels
  - Consistent thru any/all failures
    - Foundational component of system resilience
- I/O Epochs
  - Epochs demark globally consistent snapshots
    - Copy-on-write OSD (ZFS)
    - Roll back to last globally consistent snapshot
    - Guarantee updates in one epoch atomic
  - No blocking / barriers
    - Non-blocking on each OSD
      - Next epoch fills cache while previous epoch snapshots
    - Non-blocking across OSDs
      - Epochs advance independently on each OSD
      - Epoch snapshots freed after next snapshot persistent on all OSDs

DAOS API

- Collective Container Open
  - Single process opens and broadcasts handle
  - All processes use handle for operations on sub-objects
- Shard aware
  - Container layout exposed by API
- Non-blocking
  - Initiation procedures
  - Completion events
    - By initiated action
    - By epoch
- Transactional
  - Writes tagged by epoch
    - All writes in the same epoch atomic
  - Get Epoch: I intend to write in this epoch
  - Put Epoch: I've finished writing to this epoch
  - Slip Epoch: I want my writes to slip to a later epoch
    - 'n' concurrent/distributed threads participate in the same epoch

DAOS API

- Prototype implementation on POSIX
  - Early prototyping target for project partners
  - Use POSIX subtree to for object index
    - DNE Phase I: separate directory per shard
    - DNE Phase II: single sharded directory
- Implementation on Lustre
  - Open by FID
    - Collective open on DAOS container
  - Unified Target
    - Index and data objects on same OST
    - Export OST object creation to client
  - Sharded index
    - Leverage DNE Phase II
  - Filesystem metadata
    - Per-shard accounting
    - Aggregation on stat

Epochs

- Scalable server collectives network
  - Uncongested virtual LNET
    - Extension of LND RDMA setup / zero-copy completion
    - Rate limit ingest: underutilization provides latency guarantee
  - Collective network establishment
    - Gossip protocols / accrual failure detection
    - Consistent spanning tree
  - Collectives protocols
    - Aggregate on path to root
    - Replicate on path to leaves
  - Uses
    - Epoch generation
    - Most recent globally stable epoch
    - Eviction barriers
- OST
  - Fence writes by epoch
- ZFS
  - Fine grained (per transaction group) snapshots
  - Eliminate I/O pipeline stall

Thank You

- Eric Barton
  CTO
  Whamcloud, Inc.
  eeb@whamcloud.com