Lustre* - Fast Forward to Exascale

High Performance Data Division

Eric Barton

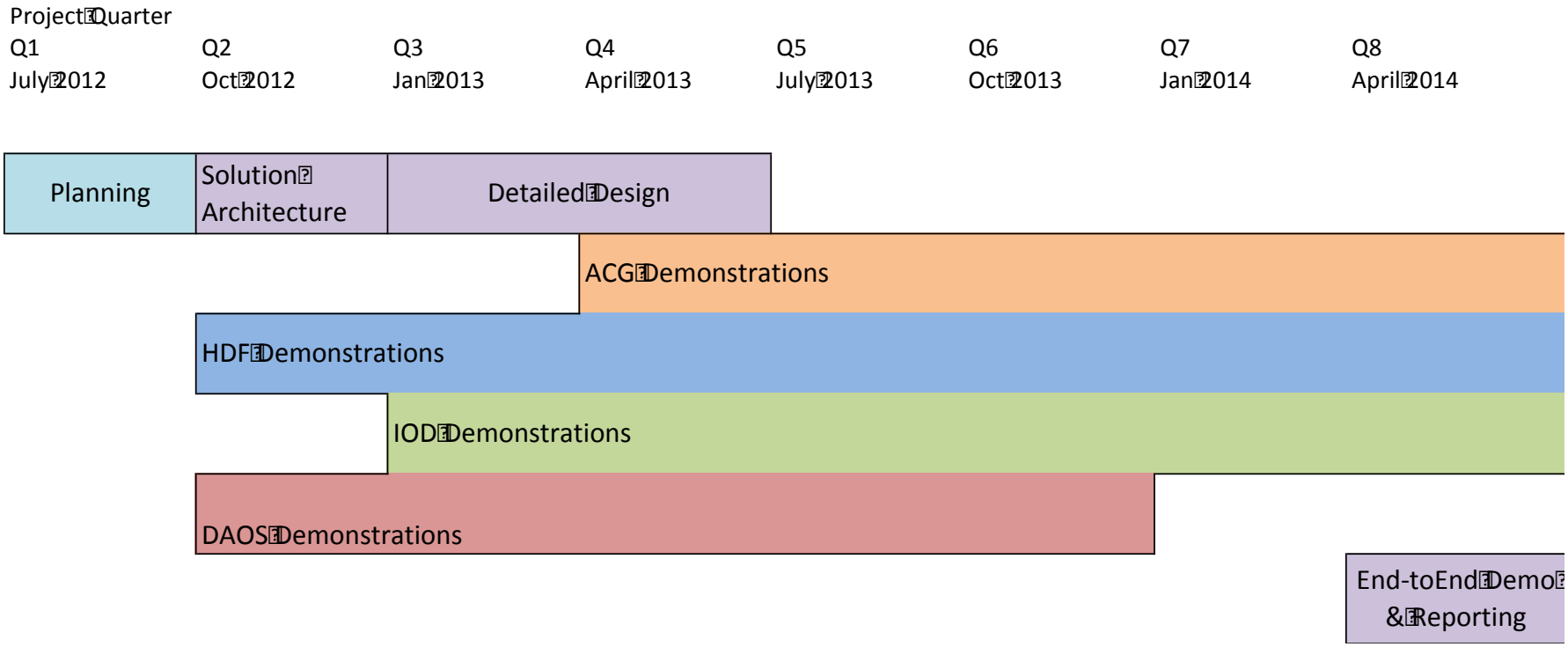
18th April, 2013

DOE Fast Forward IO and Storage

- Exascale R&D sponsored by 7 leading US national labs
  - Solutions to currently intractable problems of Exascale required to meet the 2020 goal of an Exascale system
- Whamcloud & partners won the IO & Storage contract
  - Proposal to rethink the whole HPC IO stack from top to bottom
    - Develop a working prototype
    - Demonstrate utility of prototype in HPC and Big Data
  - HDF Group – HDF5 modifications and extensions
  - EMC – Burst Buffer manager & I/O Dispatcher
  - Whamcloud – Distributed Application Object Storage (DAOS)
  - Cray – Scale out test
- Contract renegotiated on Intel acquisition of Whamcloud
  - Intel – Arbitrary Connected Graph computation
  - DDN – Versioning Object Storage Device

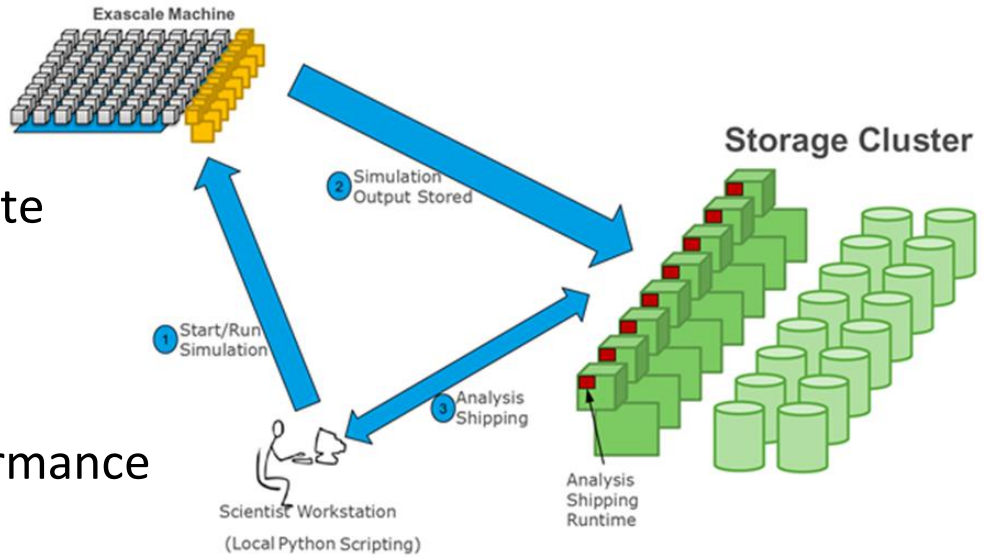
Project Schedule

- 8 project quarters from July 2012 through June 2014
  - Quarterly milestones demonstrate progress in overlapping phases
  - Planning – architecture – design – implementation – test – benchmark



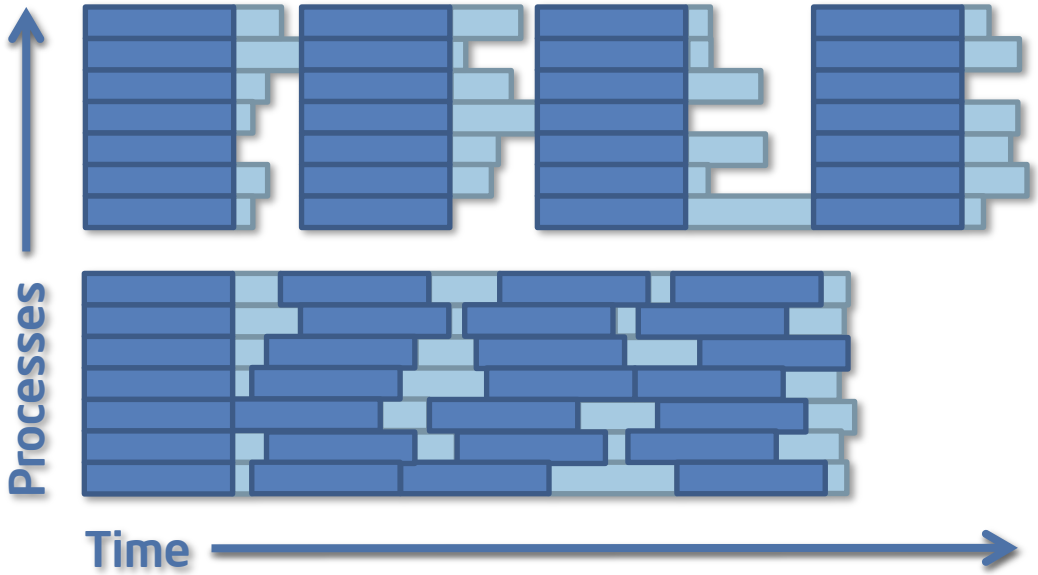
Project Goals

- Make storage a tool of the Scientist
  - Tractable management
  - Comprehensive interaction
  - Move compute to data or data to compute as appropriate



- Overcome today's IO limits
  - Multi-petabyte datasets
  - Shared file & metadata performance
  - Horizontal scaling & jitter
- Support unprecedented fault tolerance
  - Deterministic interactions with failing hardware & software
  - Fast & scalable recovery
  - Enable multiple redundancy & integrity schemes

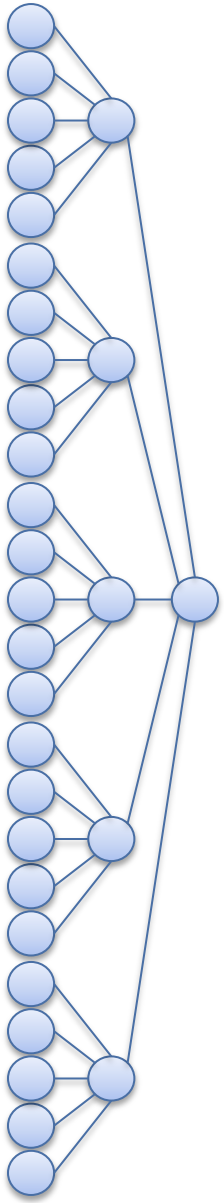
Non-blocking APIs



- Jitter
  - Scheduling noise
  - Power management
  - Dynamic load imbalance
- Tight coupling
  - Bulk synchronous programming simplifies application development
  - Makes strong scaling harder
- Loose coupling
  - Closes idle “gaps”
  - Requires non-blocking IPC **and IO**
- All IO non-blocking
  - Initiation procedure / completion event

Collectives

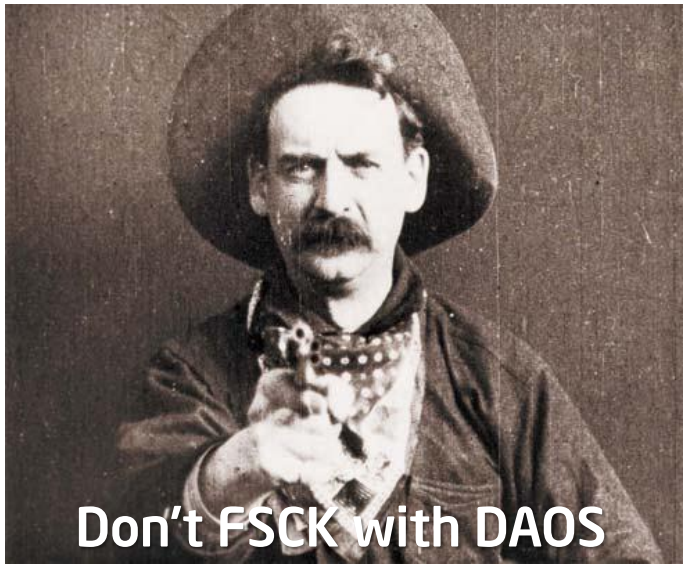
- Scalable O(log n) group operations
  - Open, commit…
- Push communications up the stack
  - Higher levels may…
    - Be able to piggyback on their own communications.
    - Have access to higher performance communications.
- local2global / global2local
  - Single process performs IO API call on behalf of process group
  - local2global creates opaque shareable buffer
  - global2local uses shareable buffer to bind local resources



Locking & Caching

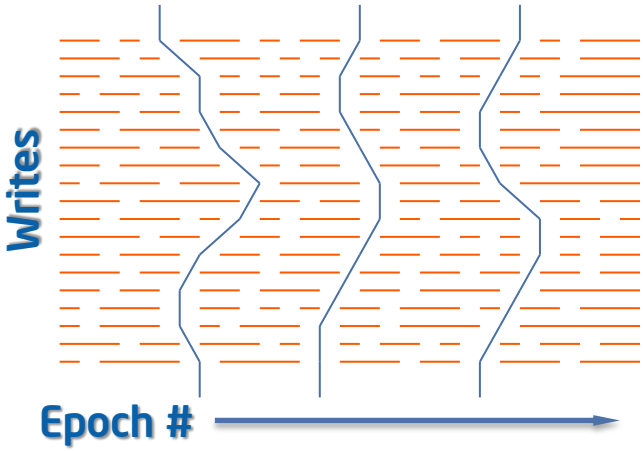
- Serialization kills scalability
  - It's not the storage system's responsibility
- Storage is not message passing
  - Tightly coupled processes should communicate directly
- Low-level IO should not predict high level caching requirements
  - Read-ahead / write behind is different from working set
- Avoid premature block alignment
  - It's a needless source of contention
- Don't let writers block readers
  - Or vice versa

Atomicity



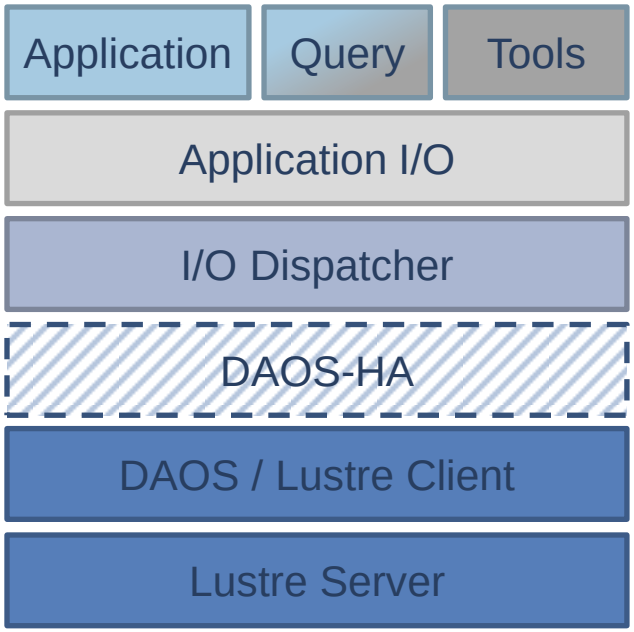
- Consistency & integrity guarantees
  - Required throughout the I/O stack
  - Required for data as well as metadata
    - Metadata is data to the level below
  - Cannot afford O(system size) recovery
- Transactions
  - Move storage system between consistent states
  - Recovery == rollback uncommitted transactions
    - Prefer O(0) v. O(transaction size) recovery time
  - Simplified interaction with failing subsystems for upper levels
  - Nestable transactions required in a layered stack
  - Scrub still required to protect against bitrot

Transactions



- Transactions ordered by epoch #
  - Writes apply in epoch order
  - All writes in an epoch committed atomically
  - All reads of an epoch see consistent data
- Applied within epoch scope
  - Container granularity
  - Multi-process and multi-object writes
  - Single committer for each open scope
- Arbitrary transaction pipeline depth
  - System may aggregate epochs
  - Highest Committed Epoch (HCE) determined on epoch scope open
  - “Slip” scope to check/wait for updates

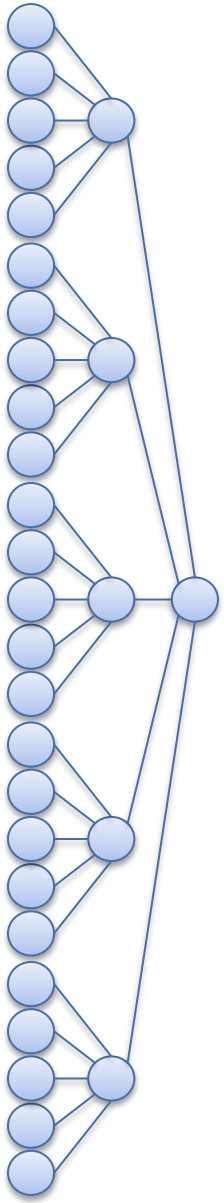
Layered I/O stack



- Applications and tools
  - Index, search, analysis, viz, browse, edit
  - Analysis shipping
  - In-transit analysis & visualization
- Application I/O API
  - Multiple domain-specific API styles & schemas
- I/O Dispatcher
  - Impedance match application requirements to storage capability
  - Burst Buffer manager
- DAOS-HA
  - High-availability scalable object storage
  - Follow-project from Fast Forward
- DAOS
  - Transactional scalable object storage

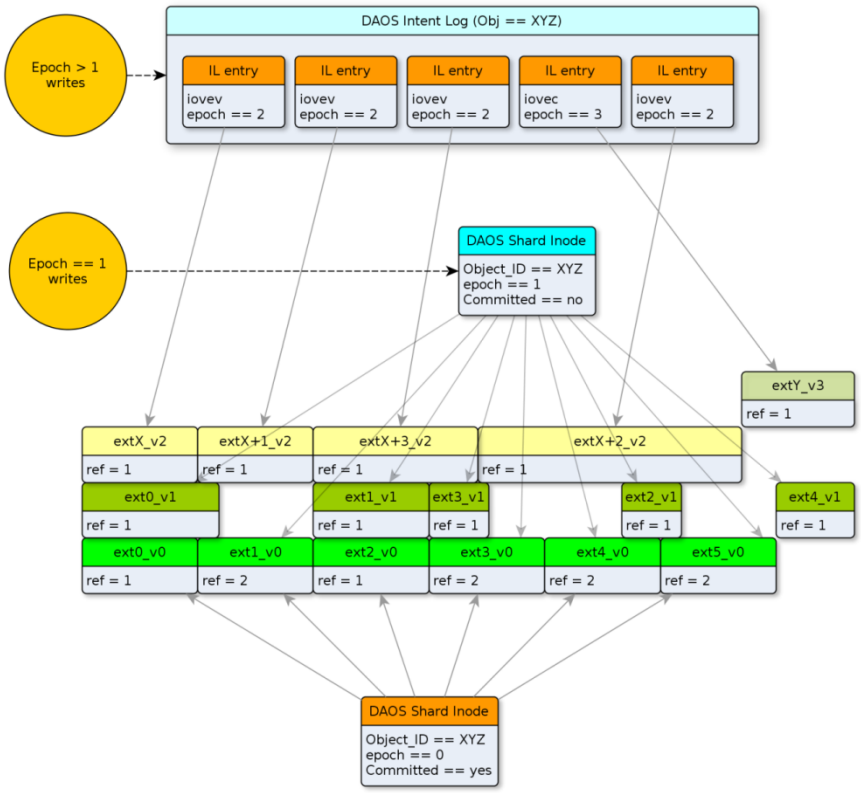
Scalable server health & collectives

- Health
  - Gossip distributes “I’m alive”
    - Fault tolerant
    - O(log n) latency
- Tree overlay networks
  - Fault tolerant
    - Collective completes with failure on group membership change
  - Scalable server communications
    - Scalable commit
    - Collective client eviction
    - Distributed client health monitoring



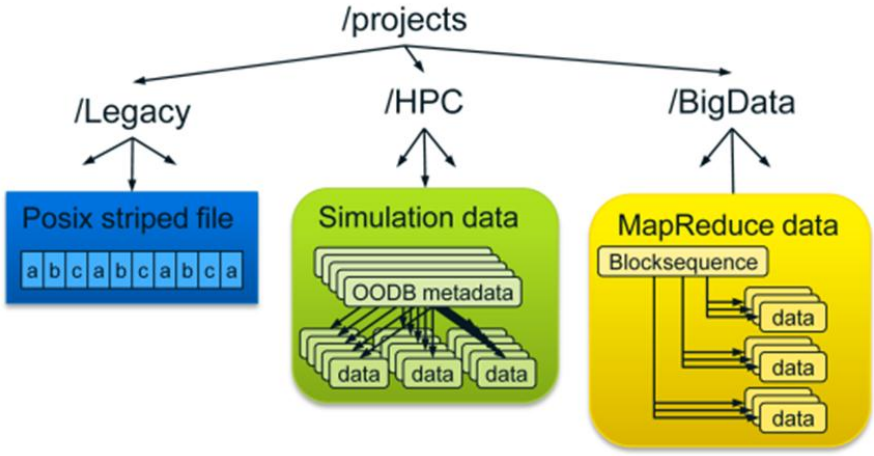
Versioned object storage

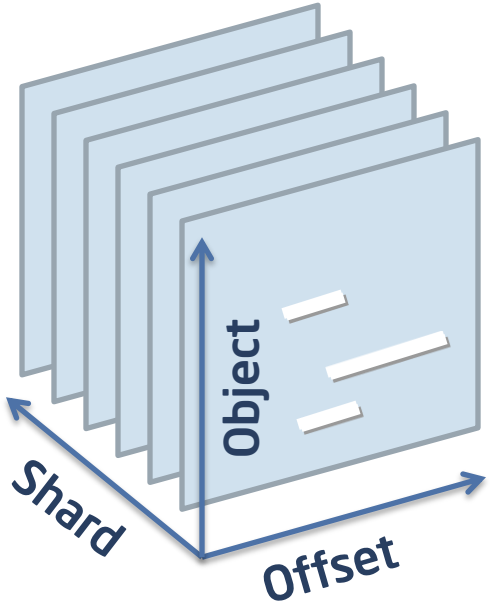
- COW & snapshot
  - Transaction rollback
- Version intent log
  - Applies writes in epoch order
- Writes persisted on arrival
  - No serialisation / backpressure
  - Full OSD blocks don’t have to move
- Extent metadata insert in epoch order
  - Start immediately previous epochs complete
  - On arrival when possible
  - Otherwise via version intent log



DAOS Containers

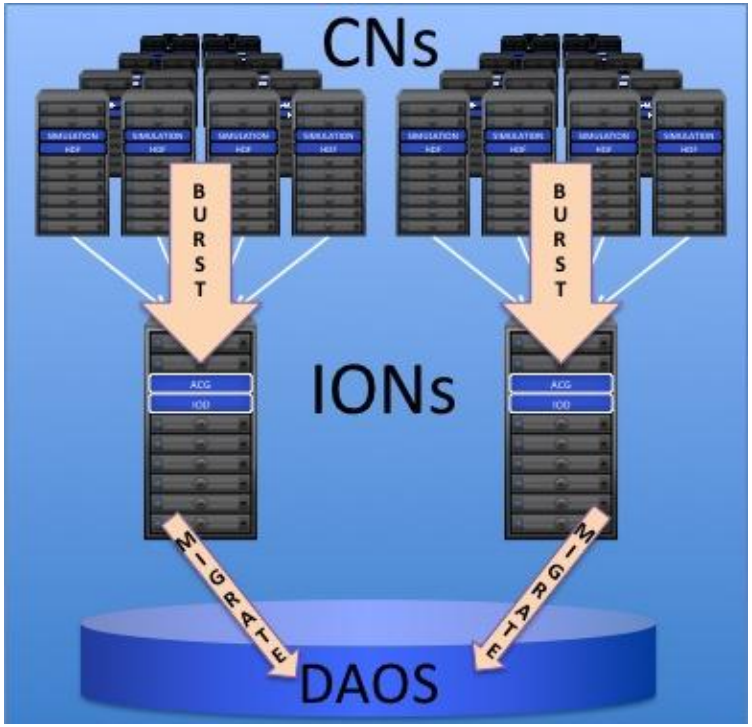
- Virtualizes Lustre's underlying object storage
  - Shared-nothing
    - 10s of billions of objects
    - Thousands of servers
- Private object namespace / schema
  - Filesystem namespace unpolluted
- Transactional PGAS
  - Baseline: addr = <shard.object.offset>
  - HA: addr = <layout.object.offset>
- Read & Write
  - No create/destroy
  - Punch == store 0s efficiently



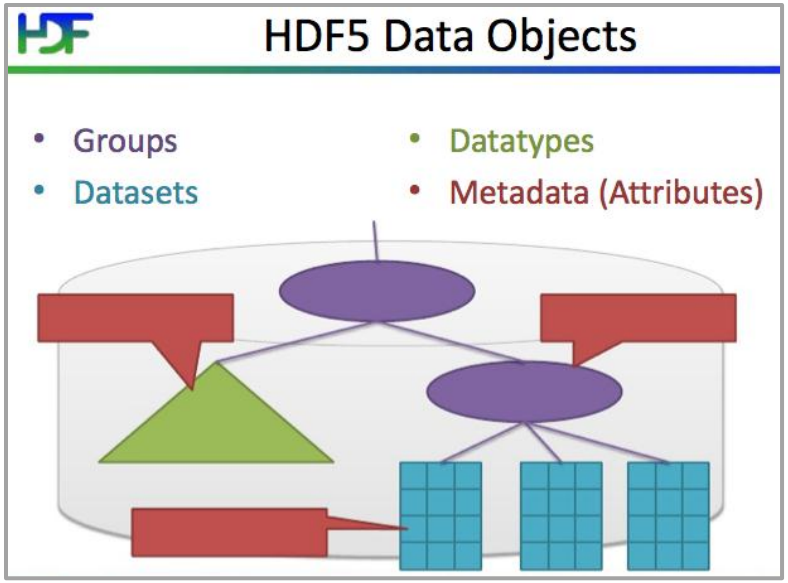


I/O Dispatcher

- Abstracts Burst Buffer (NVRAM) and global (DAOS) storage
- I/O rate/latency/bandwidth matching
  - Absorb peak application load
  - Sustain global storage performance
- Layout optimization guided by upper layers
  - Application object aggregation / sharding
    - Stream transformation
    - Semantic resharding
    - Multi-format semantic replication
- Buffers transactions
- Higher-level resilience models
  - Exploit redundancy across storage objects
- Scheduler integration
  - Pre-staging / Post flushing
- End-to-end data integrity



HDF5 Application I/O



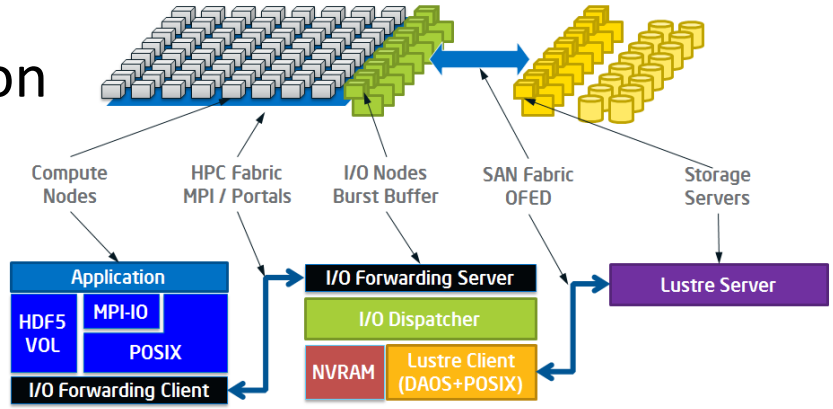
- Built-for-HPC object database
- New application capabilities
  - Non-blocking I/O
    - Create/modify/delete HDF5 objects
    - Read/write HDF5 Dataset elements
  - Atomic transactions
    - Group multiple HDF5 operations
- HDF5 Data Model Extensions
  - Pluggable Indexing + Query Language
  - Support for dynamic data structures
- New Storage Format
  - Leverage I/O Dispatcher/DAOS capabilities
  - End-to-end metadata+data integrity

Big Data – Arbitrary Connected Graphs

- HDF5 Adaptation Layer (HAL)
  - Storage API for ACG Ingress & Computation Kernel applications
  - Stores partitioned graphs and associated information using HDF5
- ACG Ingress
  - Extract meaningful information from raw data
  - Transform data dependency information into a graph
  - Partition graphs to maximize efficiency in handling and computation
- Graph Computation Kernel
  - Machine Learning: LDA, CoEM, ALS, etc.
  - Graph Analytics: PageRank, Triangle counting, Community structure

Follow-on development

- Productization & system integration
  - Job scheduler integration
    - In-transit analysis runtime
    - Analysis shipping runtime
  - Monitoring and management



- Btrfs VOSD – in-kernel (GPL) storage target
- DAOS-HA – Replication / erasure coding
  - IOD/HDF5-HA: Fault-tolerant Burst Buffer & IO forwarding
- Additional top-level APIs
  - Application domain-specific – e.g. OODB, MapReduce etc.
  - Layered over HDF5 or directly over IOD
  - Associated tools


intel