

DAOS

An Architecture for Extreme Scale Storage

Eric Barton
High Performance Data Division
Intel

Legal Information

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation.

- Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at <http://www.intel.com/content/www/us/en/software/intel-solutions-for-lustre-software.html>.
- Intel may make changes` to specifications and product descriptions at any time, without notice.
- Any code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.
- Intel, Intel Inside, the Intel logo and Xeon Phi are trademarks of Intel Corporation in the United States and other countries.
- Material in this presentation is intended as product positioning and not approved end user messaging.

*Other names and brands may be claimed as the property of others.

© 2015 Intel Corporation

Emerging Trends

Increased computational power...

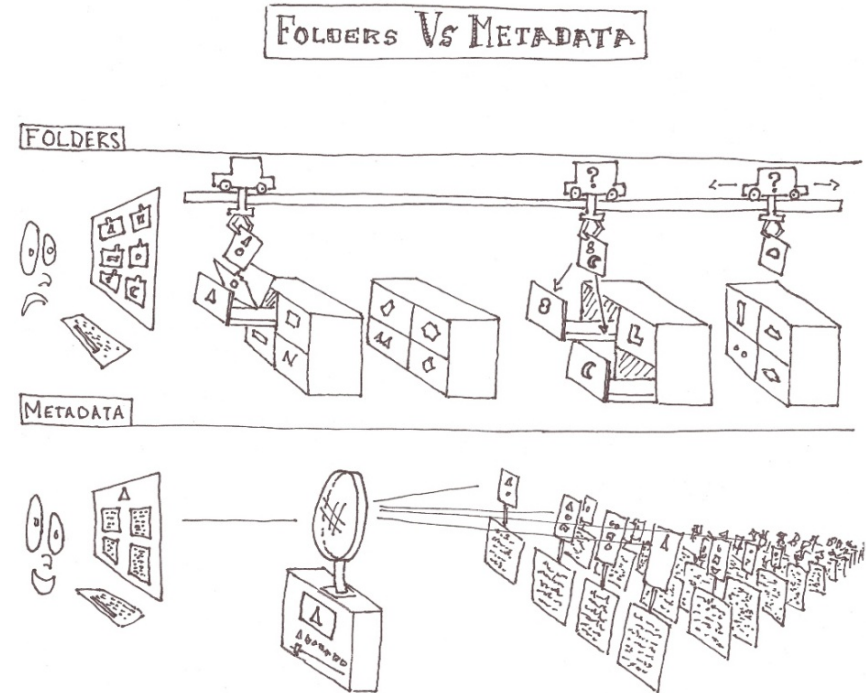
- Huge expansion of simulation data volume & metadata complexity
- Complex to manage and analyze

...achieved through parallelism

- 100,000s nodes with 10s millions cores
- More frequent hardware & software failures

Tiered storage architectures

- High performance fabric & solid state storage on-cluster
- Low performance, high capacity disk-based storage off-cluster



john-norris.net CC SA-BY 2.0

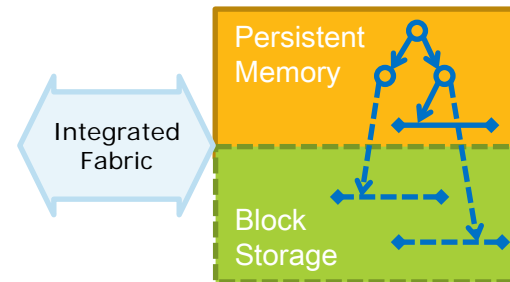
Disruptive Change

NVRAM + Integrated fabric

- Byte-granular storage access
- Sub- μ S storage access latency
- μ S network latency

Conventional storage software

- Block granular access limits scaling
- High overhead
 - 10s μ S lost to communications S/W
 - 100s μ S lost to F/S & block I/O stack



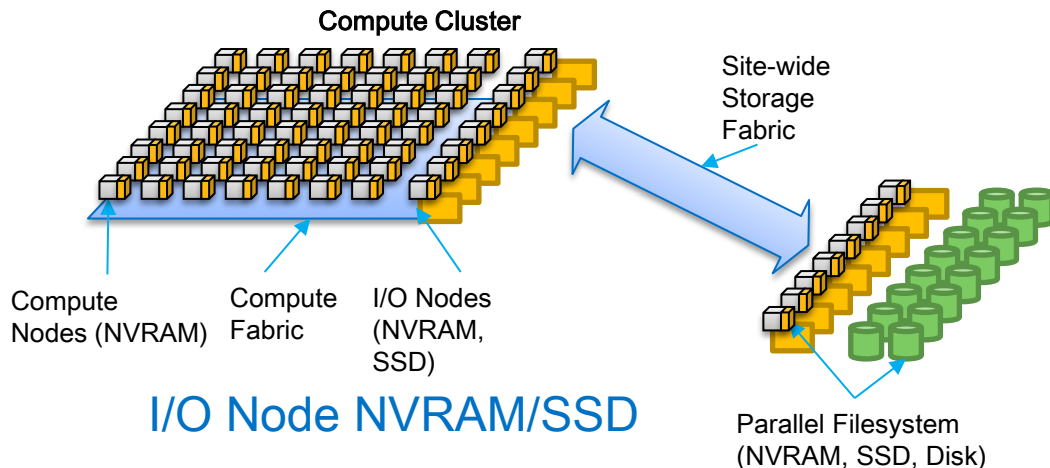
I/O stack requirements

- Minimal software overhead
 - OS bypass
 - Communications
 - Latency sensitive I/O
- Fail-out resilience
- Persistent Memory storage
 - Filesystem & application metadata / hot data
- Block storage
 - SSD – warm data / Disk – lukewarm data

Storage Architecture

Compute Node NVRAM

- Hot data
 - High valence & velocity
 - Brute-force, ad-hoc analysis
 - Extreme scale-out
- Full fabric bandwidth
 - $O(1PB/s) \rightarrow O(10PB/s)$
- Extremely low fabric & NVRAM latency
 - Extreme fine grain
 - New programming models



I/O Node NVRAM/SSD

- Semi-hot data / staging buffer
- Fractional fabric bandwidth
 - $O(10TB/s) \rightarrow O(100TB/s)$

Parallel Filesystem NVRAM/SSD/Disk

- Site-wide shared warm storage
 - SAN limited – $O(1TB/s) \rightarrow O(10TB/s)$
- Indexed data

On-cluster (hot) storage requirements

Scalable capacity

- O(10x) system DRAM

Scalable throughput

- Significant fraction of fabric bandwidth
- Significant fraction of fabric injection rate

Data integrity & consistency

- Tunable resilience/availability
- No silent failures
- Safe overwrite

Minimal usage constraints

- Global namespaces
 - System namespace shared across jobs
 - Connected workflows
 - Object namespaces shared across processes
 - Encapsulating entire simulation datasets
- Fine grained, random, massively concurrent

Minimal interference

- Data movement by unrelated workflows

Security

- Authorized/authenticated access

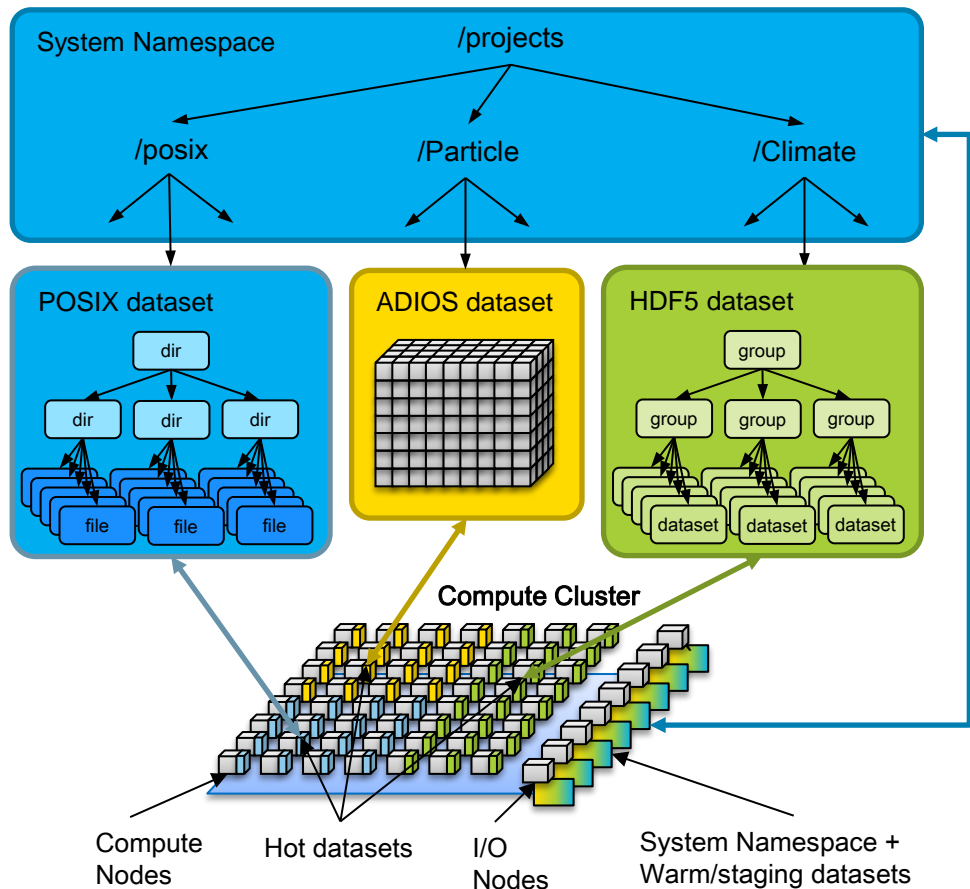
Global Namespaces

Containers

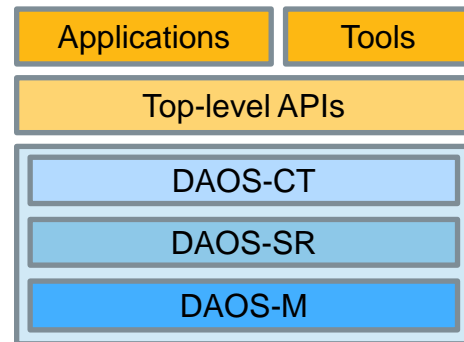
- Shared System Namespace
 - “Where’s my stuff”
- Private Namespaces
 - “My stuff”
 - Entire simulation datasets

Multiple Schemas

- POSIX*
 - Shared (system) & Private (legacy datasets)
 - No discontinuity for application developers
- Scientific: HDF5*, ADIOS*, SciDB*, ...
- Big Data: HDFS*, Spark*, Graph Analytics, ...



Distributed Application Object Storage



Exascale I/O stack

- Extreme scalability, ultra fine grain
- Integrity, availability, resilience
- Unified model over all storage tiers site-wide

Multiple Top Level APIs

- Domain-specific APIs
- High-level data models

DAOS-CT: Caching and Tiering

- Data migration over storage tiers
 - Guided by usage metadata
 - Driven by system resource manager

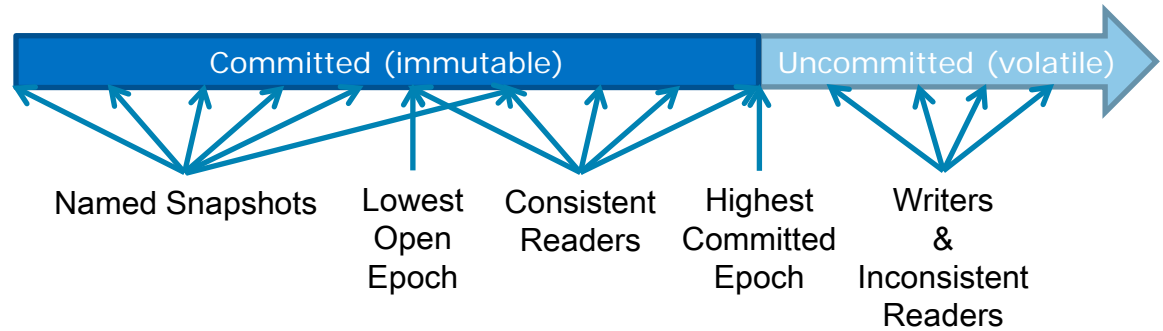
DAOS-SR: Sharding and Resilience

- Scaling throughput over storage nodes
- Fail-out resilience across storage nodes

DAOS-M: Persistent Memory Object Storage

- Ultra-low latency / fine grain I/O
- Fine-grain versioning & global consistency
- Location (latency & fault domain) aware

Transactions



Why

- Simplify application development
 - Safe update in-place
 - Guaranteed data model consistency
 - Concurrent producer/consumer workflows
- Support resilience schemas
 - Guaranteed consistency for redundant distributed data
- Support tiered storage
 - Preserve integrity on data migration

How

- Multi-version concurrency control
 - Snapshot consistency on read
 - Maximize concurrency/asynchrony
- Process groups
 - Arbitrary numbers of collaborating processes
 - Arbitrary numbers of storage targets
 - Leader commit/snap/migrate

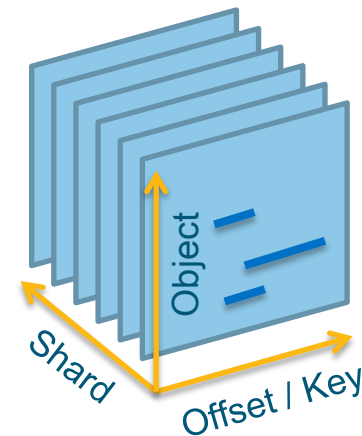
DAOS-M Object Storage

Multiple Independent Object Address Spaces

- Versioning Object PGAS
- Container = {container shards} + metadata
 - Container Shard = {objects}
 - Object = KV store or byte array
 - Sparsity exposed
 - Metadata = {shard list, commit state}
 - Minimal
 - Resilient (Replicated state machine)

Maximum concurrency

- Byte-granular MVCC
- Deferred integration of mutable data
- Writes eagerly accepted in arbitrary order
- Reads sample requested version snapshot



Distributed Transactions

- Prepare: Send updates tagged with version 't'
- Commit: Mark version 't' committed in container MD
 - Version 't' now immutable and globally consistent
- Abort: Discard version 't' updates everywhere

Low latency

- End-to-end OS bypass
- Persistent Memory server
- Userspace fabric drivers

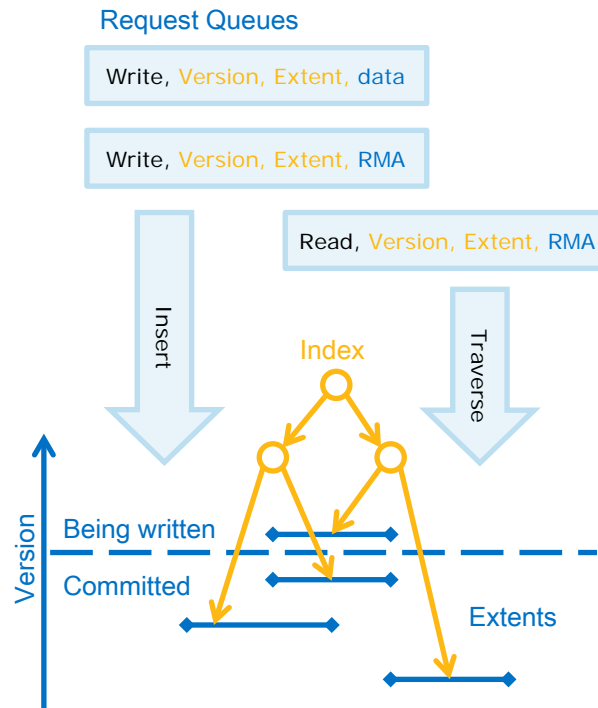
DAOS-M latency sensitive server operations

Byte array objects

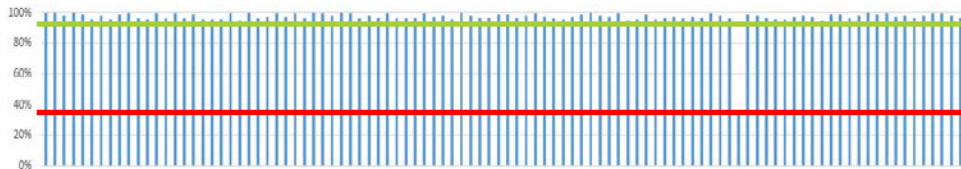
- Write (log data)
 - Allocate extent buffer in NVRAM
 - Copy immediate / RDMA READ remote
 - Insert into persistent extent.version index
- Read (data integration)
 - extent.version index traversal => gather descriptor
 - RDMA WRITE remote

Key-Value objects

- Insert/remove/retrieve value into key.version table



Sharding & Resilience



Multiple mixed schemas

Performance schemas

- Scale IOPs & bandwidth

Resilience schemas

- Data integrity
 - Checksums + data stored separately
- N-way replication
 - High performance for shared write objects
- Erasure codes
 - High efficiency for non-shared objects
- Asynchronous refactor, scrub & repair
 - Exploit immutability of committed data

Leverage DAOS-M

- Distributed consistency
- Sparsity

Scaling Requirements

- Onerous!
 - Aliasing of access & distribution patterns
 - Bulk synchronous workload == Amdahl's law vector
- Extreme object size dynamic range
 - “Megaliths” v. “grains of sand”
- Declustering
 - Rebalance on node addition
 - Distributed rebuild on node failure

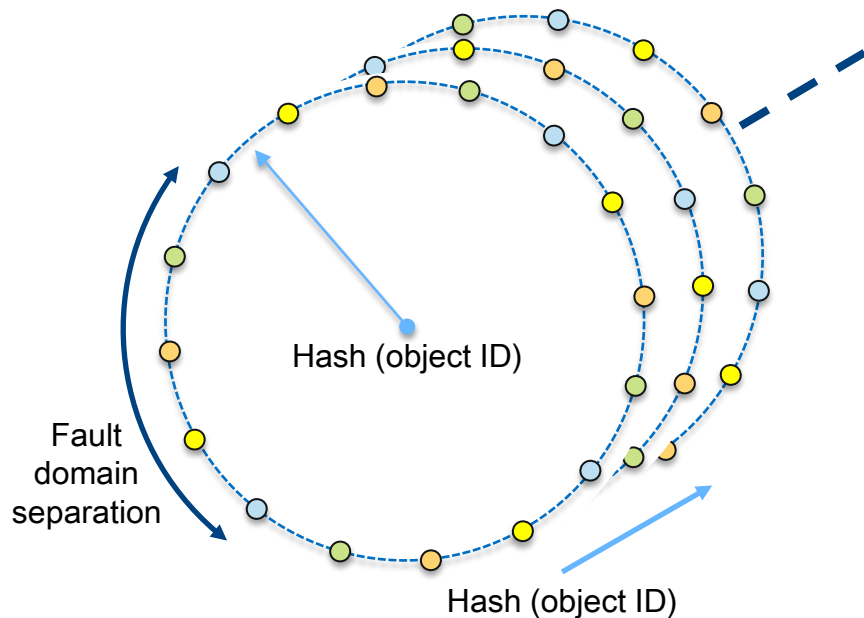
Sharding & Resilience

Algorithmic ($O(0)$) layout metadata

- Consistent hash randomizes placement
- Replicas placed adjacently
 - Hash must guarantee minimum separation of nodes in same fault domain
- Multiple hash rings for declustering

Explicit ($O(n)$) layout metadata

- Layout responsive to usage
 - Preserve locality / feed “hungriest mouth”
- $O(0)$ structures used to store layout



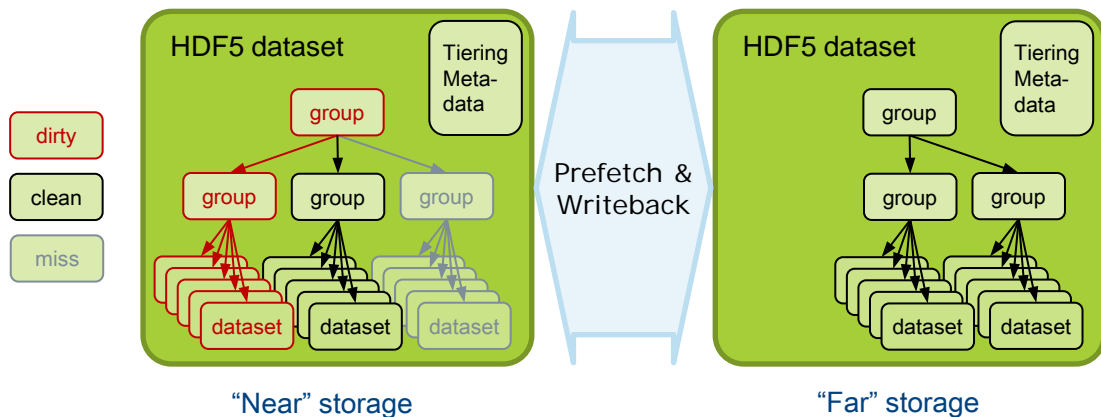
Caching & Tiering

Metadata

- Residence maps
 - Whole object maintained directly
 - Sub-object leverages lower layers
- Access patterns
 - Historical
 - Explicit notification by upper levels
 - Data “colouring”

Data migration

- Resharding between tiers
 - Maintain distributed object semantics
 - Maximize performance on subsequent access
 - Select appropriate resiliency schemas



Explicit control

- Persist & prestage APIs / JCL
- System resource manager driven migration
 - Rebalance & minimize interference

Transparent caching

- Write-back & demand cache
- Prefetch guided by usage metadata
- Residence maps

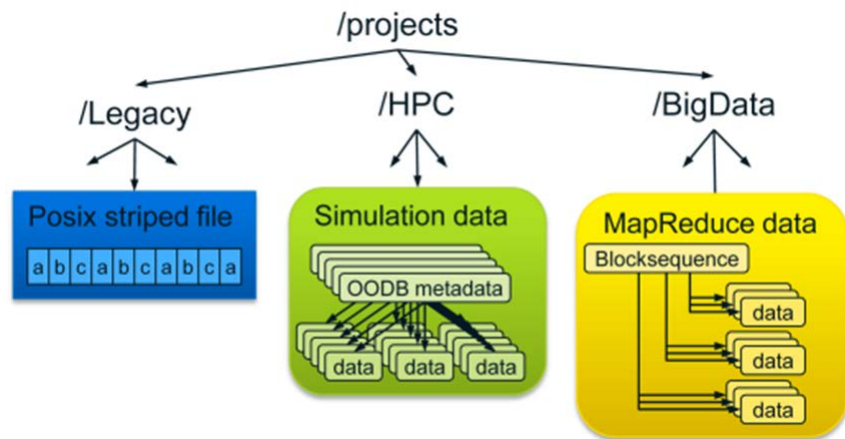
Top level I/O APIs

POSIX Containers

- POSIX namespace over DAOS-HA objects
 - Dynamically sharded directories & files
- Private POSIX namespaces
 - Library for parallel applications and middleware targeting POSIX
- System POSIX namespace
 - Parallel server exporting shared namespace

DAOS for application programmers

- Simplified APIs
- Distributed Persistent Memory



High level HPC object databases

- Complex application datatypes & metadata
- HDF5 + derivatives / ADIOS / SciDB etc...

Big Data

- HDFS compatibility layer
 - Hadoop ecosystem
- Spark / Graph Analytics etc...

NVRAM Storage Revolution

Cost-effective storage & fabric integration

- Challenge: Extreme scale-out
 - Amdahl's law
 - Fault Tolerance
- Reward: Storage @ full fabric bandwidth
 - $O(1000)$ increase in data velocity

Byte-granular data access @ uS latency

- Challenge: Deliver benefit to applications
 - Software overhead of conventional storage & communications stacks
- Reward: Ultra fine-grain access
 - Remove constraints on applications
 - Enable new programming models

