ESSIO
Extreme Scale Storage & I/O

Eric Barton
Intel

ENTERPRISE & HPC PLATFORM GROUP
Data Center Group

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.

Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

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.

Emerging trends

Increased computational power…

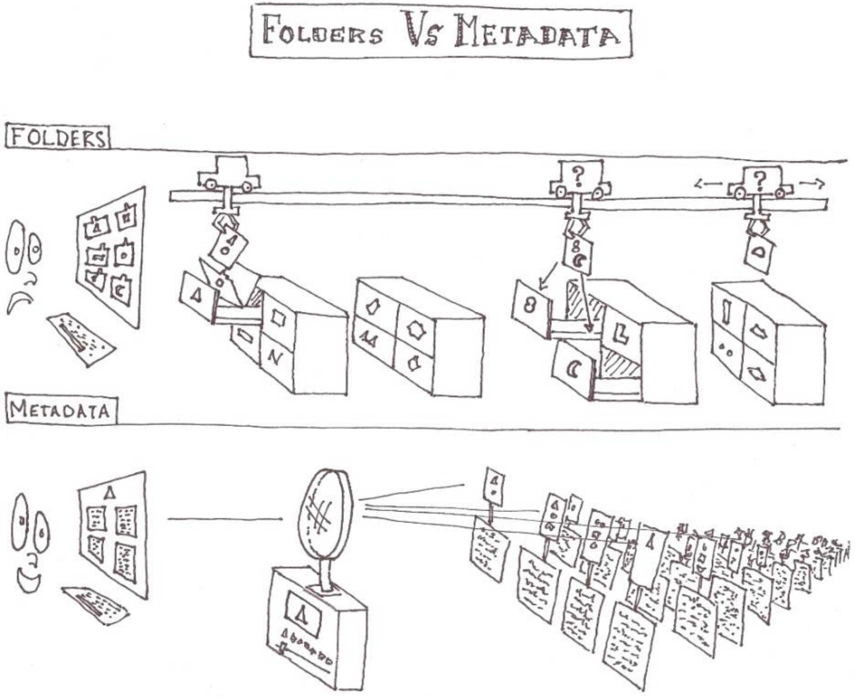
- Huge expansion of data volumes & metadata complexity
- Complex to manage and analyze

…achieved through parallelism…

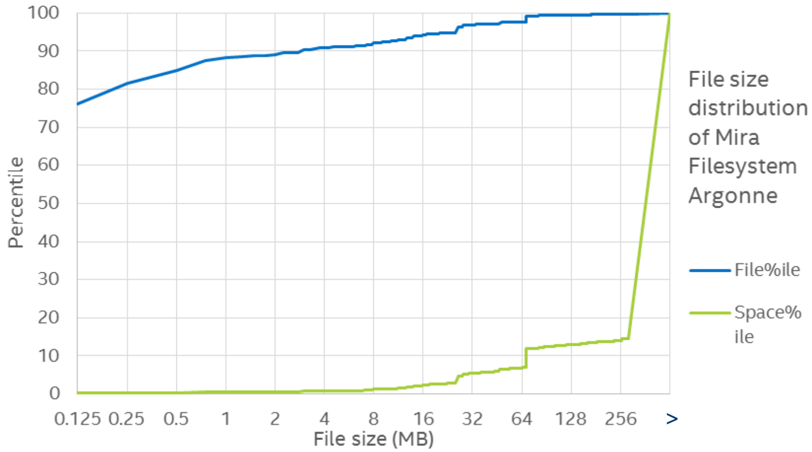
- 100,000s nodes with many millions of cores
- Routine, frequent hardware & software failures
- Extreme scalability challenge

…with tiered storage architectures

- Economics of performance v. capacity



Emerging trends



Vast majority of storage objects are tiny…

- Kilobytes and smaller
- Trending to larger proportion of system capacity
- High performance => Billions of IOPs @ lowest possible latency
- Resilience => Replication improves concurrency/scalability @ acceptable storage overhead
- **Poorly supported by today's filesystems**

…but vast majority of space is used by large storage objects

- Megabytes and larger
- High performance => efficient streaming @ Terabytes per second
- Resilience => Erasure codes required for space efficiency & limit system cost

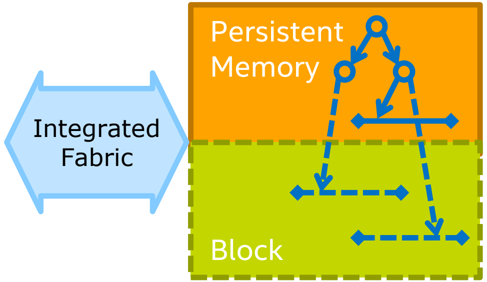
Disruptive change

3D XPoint™ + Intel® Omni-Path Architecture

- Byte-granular storage
- Sub-µS storage latency
- µS network latency

Conventional storage software

- High overhead
  - 10s µS lost to communications S/W
  - 100s µS lost to F/S & block I/O stack
- Block/page granular & alignment sensitive
  - False sharing limits scalability



Exascale storage

- Low overhead
  - OS bypass comms + storage
- Byte granular & alignment insensitive
  - Ultra low latency enables fine granularity
- Persistent Memory
  - All metadata & fine-grained application data
- Block storage
  - NAND: High performance bulk data
  - Disk: High capacity cold data

Project History

DOE Fast Forward Storage & I/O

- 2 year project – prototype demonstrated June 2014
- Intel / HDF Group* / EMC* / DDN* / Cray*
- Prototype Exascale tiered storage system
  - IOD – based on PLFS (performance tier)
  - DAOS – based on Lustre* / ZFS* (long term storage)

DOE Extreme Scale Storage & I/O

- 2 year project – prototype to be demonstrated June 2017
- Intel / HDF Group
- Unified storage model over all storage tiers
- Working with DOE labs to port and evaluate applications

Exascale storage requirements

Routine failures – MTTI O(hours)

- Application failure
  - Complex data models have complex consistency / cleanup requirements
  - Application development intractable without system consistency guarantees
- Storage system failure
  - Fail-out resilience to simplify & cost-reduce storage hardware
  - End-to-end integrity checks required to eliminate silent data/metadata corruption
  - Workflow driven resilience schemas
    - None (cheapest) / Replication (high performance) / Erasure Codes (space efficient)

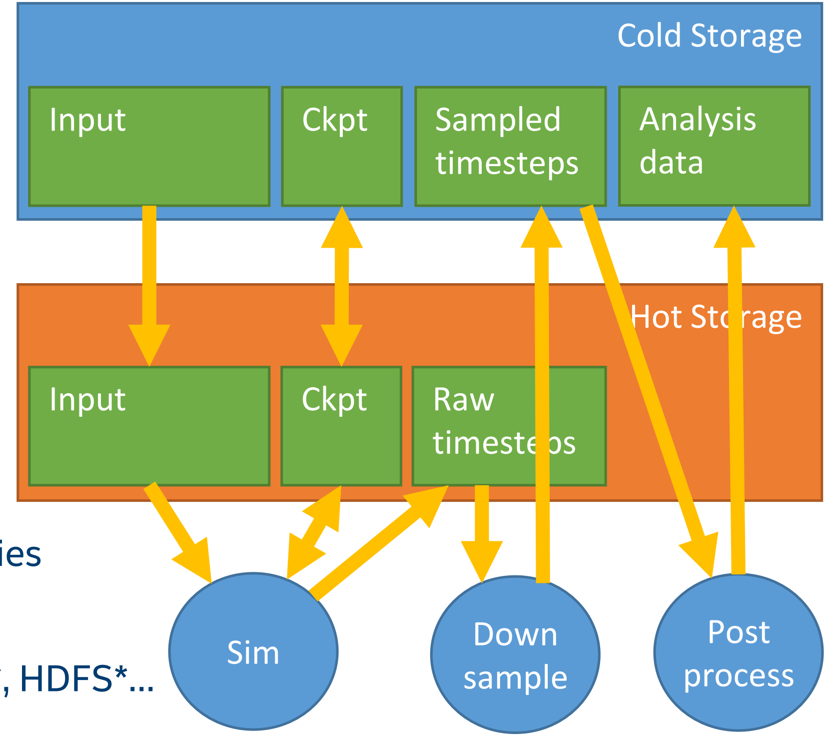
Unprecedented scalability

- Shared-nothing, lockless distribution schemas
- Arbitrary alignment and granularity without sacrificing consistency

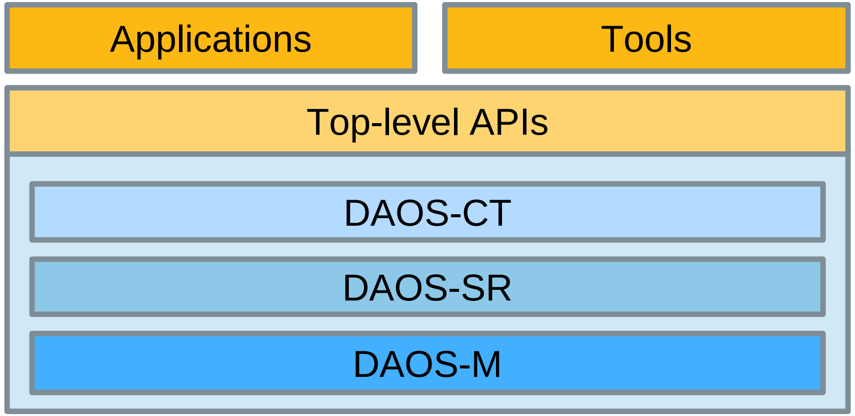
Exascale storage requirements

Simplify application & workflow development

- Transparently span multiple storage tiers
  - From cheap high capacity storage to expensive performance storage
- Support producer/consumer workflow pipelines
  - Simplify coordination between jobs sharing data
  - Connect jobs at any storage tier
- Simplify data management
  - Snapshots
  - Aggregate related datasets into manageable entities
- Support multiple storage APIs & data models
  - POSIX, HDF5*, pNetCDF*, ADIOS*, Legion*, Spark*, HDFS*…



Distributed Asynchronous Object Storage



Multiple Top Level APIs

- Domain-specific APIs
- High-level data models

DAOS-CT: Caching and Tiering

- Transparent caching
- Efficient staging

DAOS-SR: Sharding and Resilience

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

DAOS-M: Memory class object storage

- Ultra-low latency / fine grain object I/O
- Multi-version concurrency control
- Global consistency model

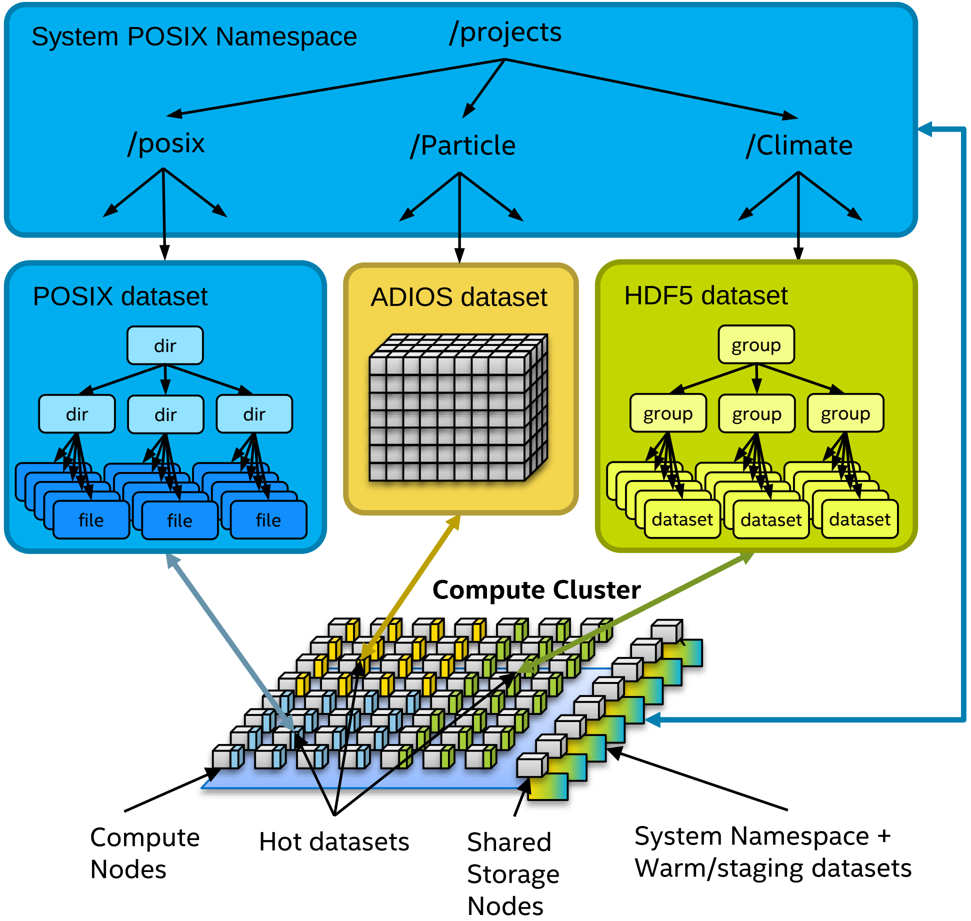
Global Namespaces

Containers

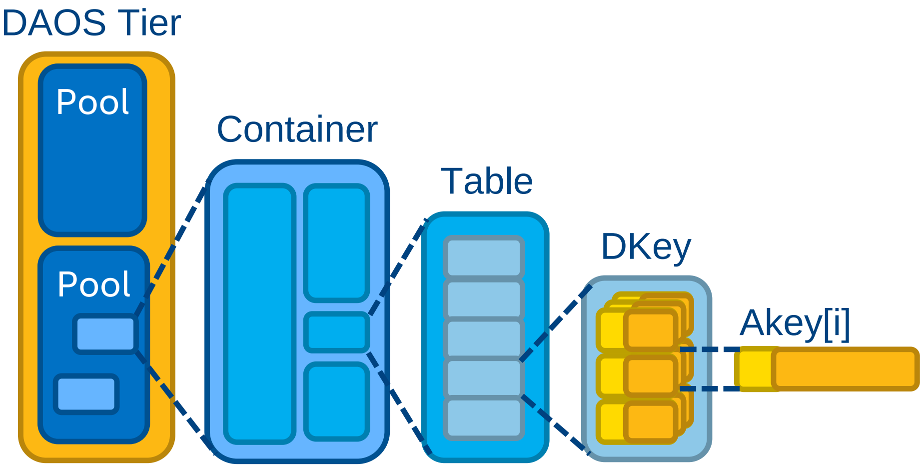
- Shared System Namespace
  - "Where's my stuff"
- Private Namespaces
  - "My stuff"
  - Entire simulation datasets

Multiple Schemas

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



DAOS Storage Model



Storage Pool

- Reservation of storage within a tier

Container

- Related data & metadata distributed across an entire storage pool

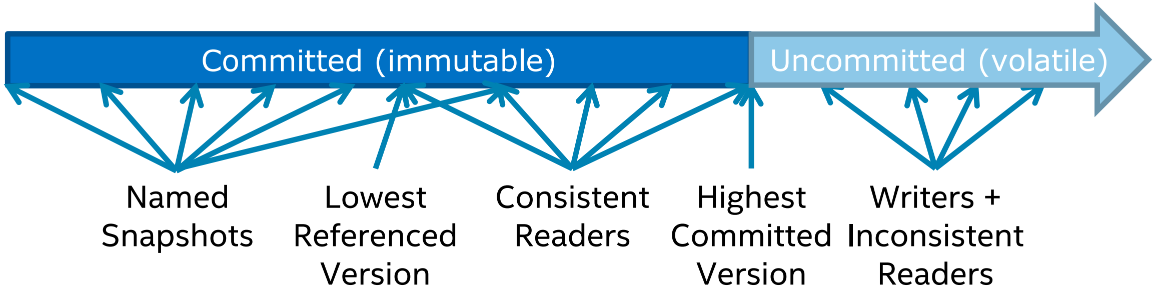
Table

- Collection of related arrays/values with own distribution/resilience schema
- Dkey determines placement
- Akey selects named array
- Index [lo…hi] selects array extent

Array Element

- Arbitrary binary blob
  - Single byte to several Mbytes
  - Atomic unit of versioning
- Non-destructive write: log blob@version
- Consistent read: blob@(max(version <= v))

Transactions



Why

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

How

- Versioned storage
  - Non-destructive (logging) write
  - Snapshot consistency on read
  - Maximize concurrency/asynchrony/scalability
- Process groups
  - Arbitrary numbers of collaborating processes
  - Leader commit/snap/migrate
- Conflict resolution in top-level client library
  - No system-imposed, worst-case serialization

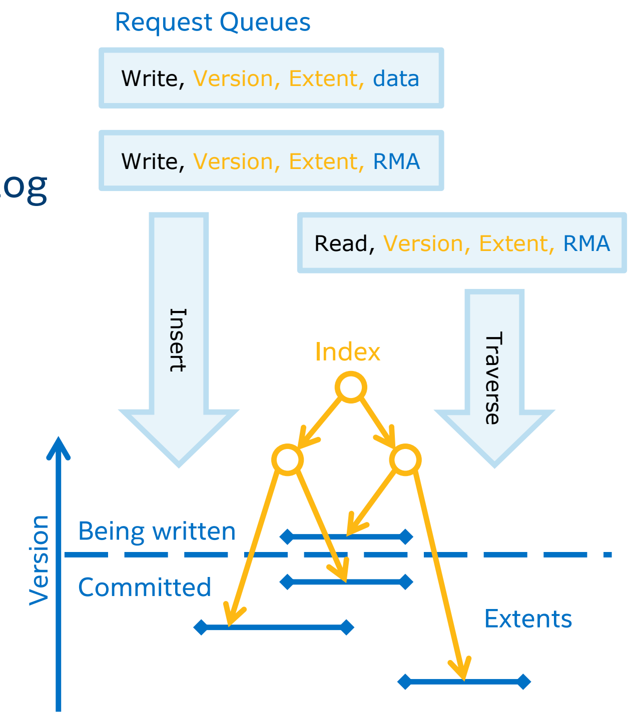
DAOS-M fundamental operations

Write (version, {array elements})

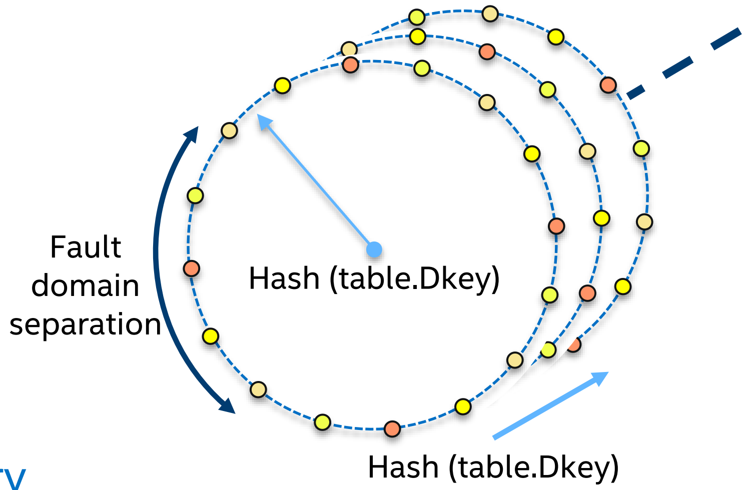
- Insert set of array elements with given version into searchable log
  - Allocate extent buffer in NVRAM
  - RDMA READ client data
  - Insert into persistent index
- Commutative – writes may arrive in any order

Read (version, {array elements})

- Return array element values with highest version <= given
  - Traverse index to create gather descriptor
  - RDMA WRITE client data
- Committed version => Immutable, consistent snapshot across storage nodes



Sharding & Resilience



Algorithmic layout metadata

- Scales with # storage nodes
- Consistent hash randomizes placement
  - Disperses fault domains
- Multiple hash rings for declustering*

Explicit layout metadata

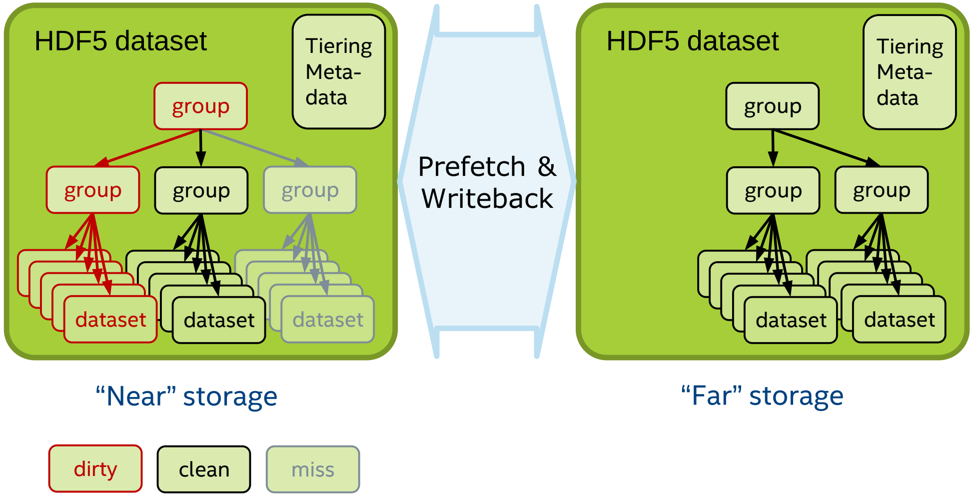
- Scales with volume of data
- Layout responsive to usage
  - Progressive & locality driven layouts
- Stored using algorithmic layout (bootstrap)

Recovery

- Failed target evicted & recover targets selected on each ring
- Storage targets hosting shards of damaged tables send table IDs to recovery targets
- Recovery targets “pull” data from peers until rebuild complete
- Clients may now read from recovery targets

* Concept developed in collaboration with Argonne National Laboratory, see preliminary work in *P. Carns et al, “Consistent Hashing Distance Metrics for Large-Scale Object Storage,” Poster at the ACM/IEEE SuperComputing Conference (SC’15), Nov. 2015.*

Caching & Tiering



Staging controls

- Prestage populates cache
  - Select using top-level data abstraction
    - POSIX directory subtrees
    - HDF5 query/view
- Persist flushes committed data
- Staging optimized to aggregate & stream I/O on lower performance tier

Transparent caching

- Fetch-on-miss / policy driven eviction
- Slowly evolving working sets

Data migration

- Resharding between tiers
  - Maintains distributed object semantics
  - Tier-appropriate resiliency schemas
- Transactional

Storage Revolution

Cost-effective storage & fabric integration

- Challenge: Extreme scale-out
  - Scalability (Amdahl's law)
  - Fault Tolerance
- **Reward: 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: Efficient ultra fine-grain I/O**
  - Simplify applications & enable new programming models