DAOS Feature Update

DAOS User Group – SC 2020

Liang.zhen@intel.com

Notices and Disclaimers

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 product or component can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. For more complete information about performance and benchmark results, visit http://www.intel.com/benchmarks .

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/benchmarks .

Intel Advanced Vector Extensions (Intel AVX) provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

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 does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Outline

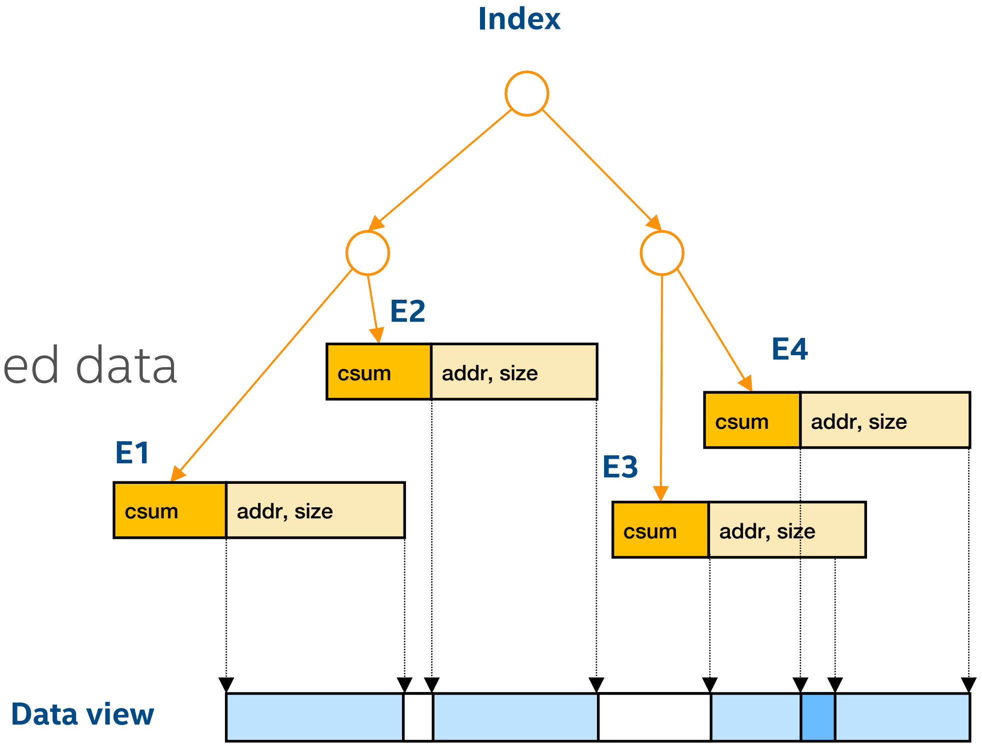
- Background: Data protection and Self-healing
- End-to-end data integrity
- Data reconstruction and Online Server Addition
- Erasure Coding
- Distributed Transaction

Background: Data protection and Self-healing

- Data protection
  - Replication
    - Simple, high capacity and bandwidth overhead
  - Erasure coding (EC)
    - Complex, high computation overhead, low capacity and bandwidth overhead
- Resilience for node failure or media corruption
  - Object shards are stored across multiple storage nodes
  - Degraded mode I/O
  - Background data recovery by self-healing system

End-to-end data integrity

- Calculated internally by the client library
  - Server-side verification is optional
- Stored persistently along with the data
- Detect silent data corruption on fetch
  - Server: verify and recompute checksum only for misaligned fetch
  - Client: verify checksum for data from server
  - Client: switch to degraded mode fetch for corrupted data
- Future works
  - Checksum scrubbing

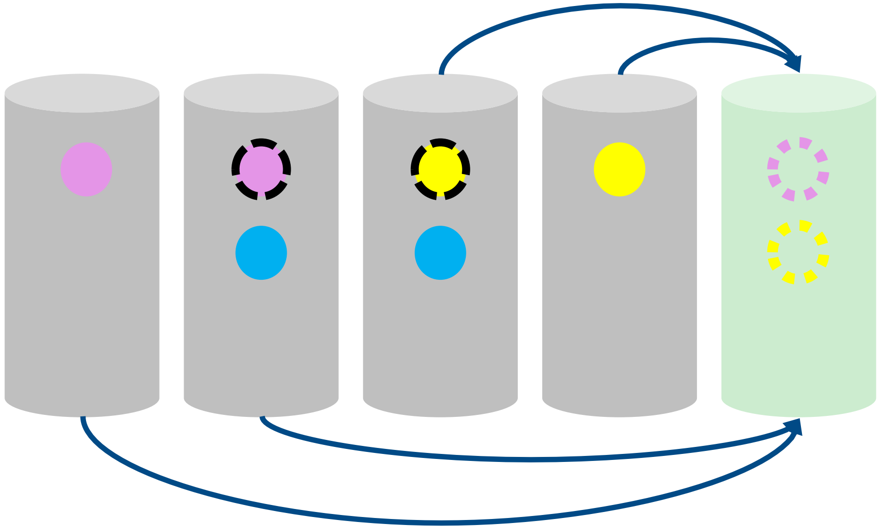


Data reconstruction and Online Server Addition

- Rebuild
  - Health monitoring, auto eviction and self-healing
- Drain
  - Manually evict storage target(s)
- Reintegration
  - Add evicted storage target(s) back
- Addition
  - Extend storage pool by adding more targets

Online Server Addition

- Data migration service
  - Input: object IDs (algorithmic object placement)
  - Action: data movement
- Generic service for data movement in the system
  - All data movement activities share the same protocol and service
    - Rebuild, reintegration, addition, drain
  - Scan objects
    - Call different placement APIs for different data movement activities
  - Pull and reconstruct
    - Data migration service



Erasure coding (EC)

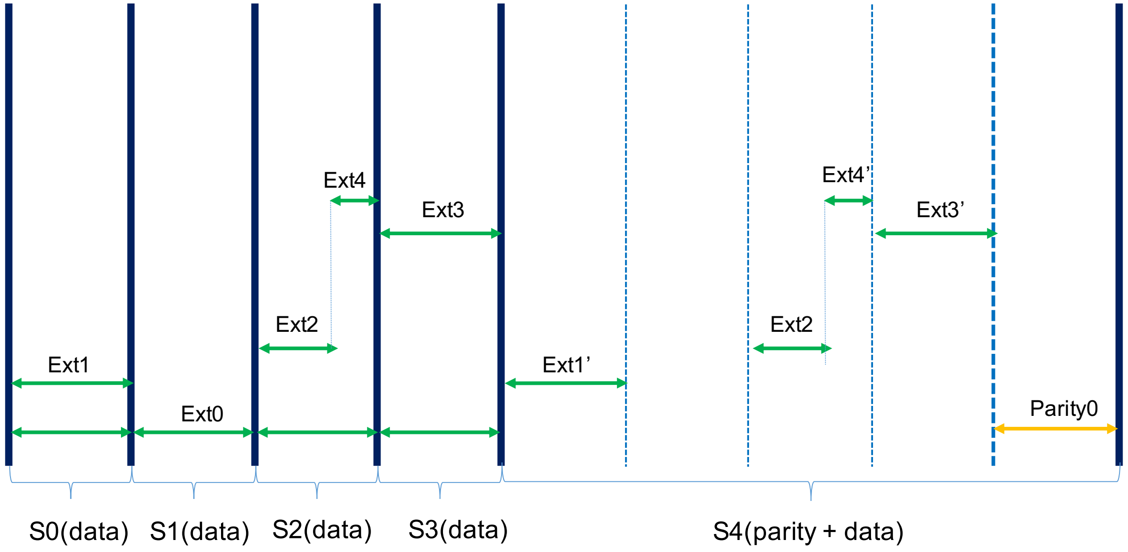
- Replication has high storage and bandwidth overhead
  - N-way replication
    - Overhead == (N – 1) * size
  - Efficient recovery
- Erasure coding is more space efficiency
  - EC(N + M)
    - overhead == M/N * size
  - Expensive recovery

- EC Functionalities
  - Reed-Solomon based EC
  - Data recovery
    - Degraded mode (client): inflight data reconstruction
    - Rebuild (server): background data recovery
  - Aggregation (server): background encoding and space reclaim

Erasure coding – Read and write protocols

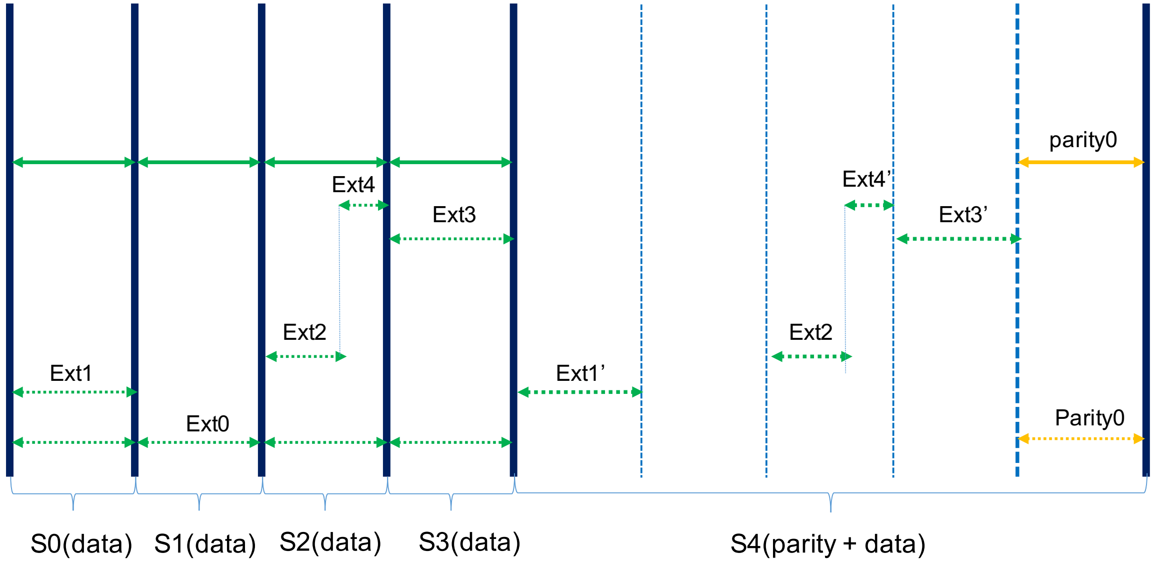
- Full stripe write
  - Client-side encoding
  - Client sends RPC to parity target (store parity)
  - Parity target forwards RPC to all the data targets (store data)
- Read
  - Client sends RPC to data targets
  - Transaction status should be considered

- Partial write
  - No encoding
  - Client sends RPC to parity target (store data)
  - Parity target forwards it to corresponding data targets (store data)



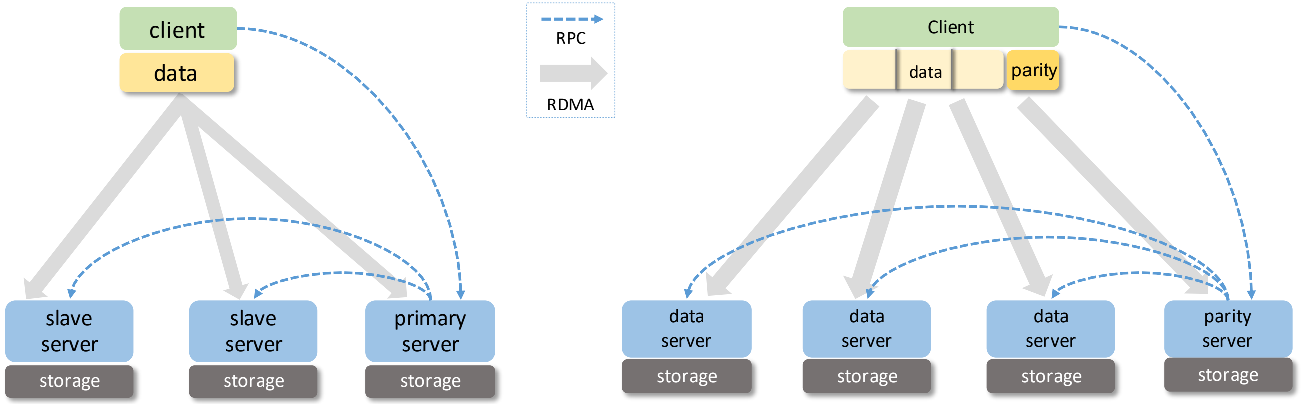
Erasure coding – Data recovery and space reclaim

- Degraded mode
  - Client: reconstruct data inflight (read extra data/parity)
- Rebuild
  - Server: reconstruct data in the background
- Aggregation
  - Server: merge overwrites and compute parity for merged data



Distributed transaction – Distributed I/O

- Both replication and EC updates are distributed I/O
  - Atomicity of distributed I/O
  - 2-phase commit protocol
- Consistency of conditional operations
  - Update
  - Insert
  - Punch



(a) Replicated write

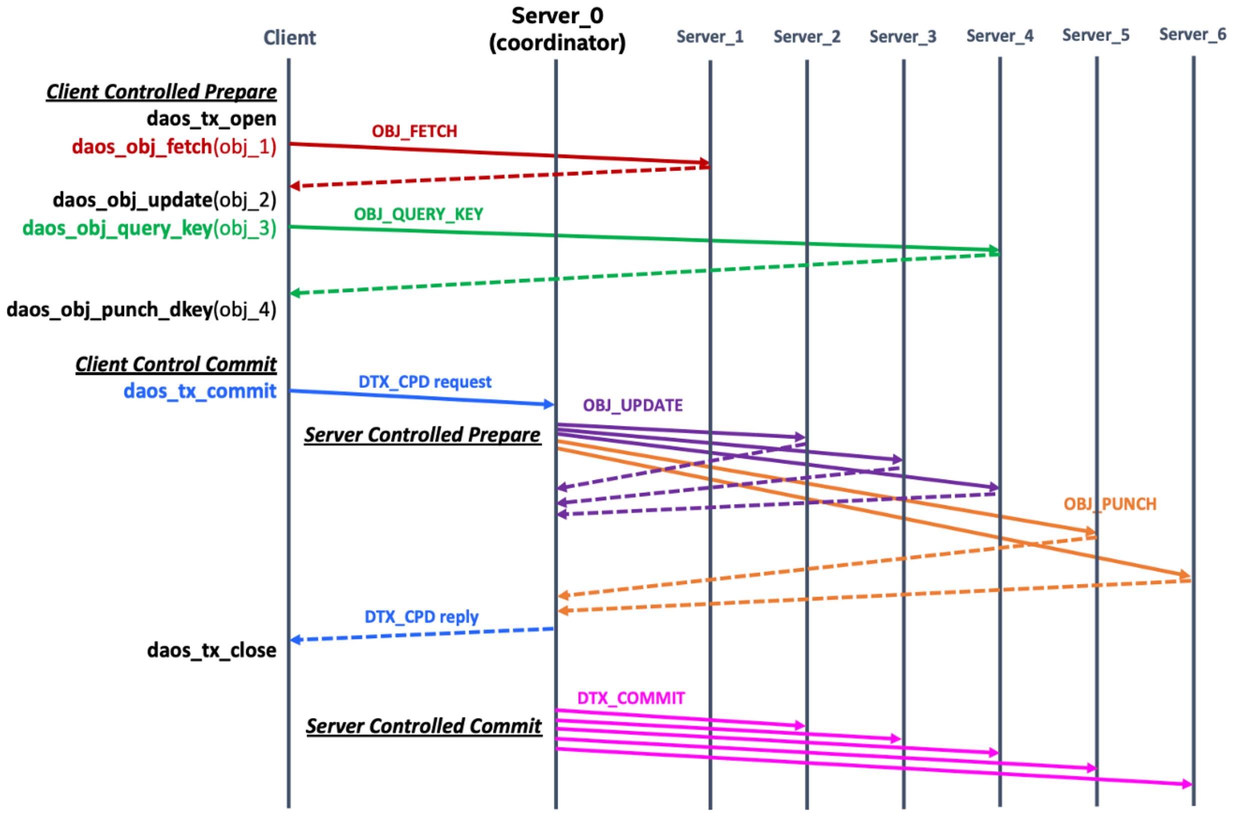
(b) Erasure coding write

Distributed transaction – Transactional API

- Transactional operations
  - Atomicity of multiple operations (e.g. rename)
  - Exported by API

- Client cached transaction
  - Client: submit reads within transaction
  - Client: cache update/punch (no RPC to server)
  - Client: send compound RPC to “commit” update/punch
  - Server: unpack the compound RPC and run

    2-PC protocol for operations of compound RPC



- Multi-version concurrency control (MVCC)
  - Ensure that transactions execute as if they are serialized in time order
  - Transaction involving both reads and writes must follow all rules
  - When a transaction is rejected, it restarts with the same transaction ID but a higher timestamp