Kinetic Campaign

Speeding Up Scientific Data Analytics with Computational Storage Drives and Multi-Level Erasure Coding

Qing Zheng
Scientist, Los Alamos National Laboratory (LANL)
LA-UR-22-29500
About Me

- **HPC Storage Scientist** at Los Alamos National Laboratory

- I received my PhD at Carnegie Mellon University in 2021

- I do distributed filesystem metadata management, KV stores, scientific data analytics

- [https://zhengqmark.github.io](https://zhengqmark.github.io)
Agenda

- **Why computational storage**: large-scale data analytics challenges in HPC
- **MarFS**: LANL’s current archival storage using erasure coding
- **Kinetic**: Seagate’s novel active disk research platform
- **C2**: LANL’s next-gen archival storage combining in-drive computing and erasure coding for cost-effective data protection, storage, and rapid queries
Large-Scale Data Analytics Challenges in HPC

Why computational storage?
Typical HPC Simulation Workflow at LANL

- Simulation **writes** state to storage periodically
- Analysis code later **reads** data back for in-mem operations (e.g.: movie making)
- Data may not compress
- Performance depends on available storage bandwidth
Trend #1: Multi-Tiering for Cost-Effective High Bandwidth

- Tackling larger data requires higher bandwidth
- No single media type can provide both speed & capacity at the same time (given a cost budget)
- Result: storage increasingly tiered
Challenge #1: Asymmetric Read-Write Performance

- **Writing** data to storage may continue to be **fast** thanks to multi-tiering.

- **Reading** data from storage can be much **slower** (when data is streamed from the slow tiers)

A more recent HPC platform
Trend #2: Analysis Increasingly Selective

- **Analysis** used to read back an entire dataset

- **Today**: queries tend to only target a small subset of data

- Need to avoid excessive data reads (especially when reading from a slow storage media)

Example: SELECT X, Y, Z FROM particles **WHERE** Ke >= 1.5
Less than 0.1% or 0.00001% needs to be read from storage

Image from LANL VPIC simulation done by L. Yin, et al at SC10. X, Y, Z are 3D locations of a particle. Ke is a particle’s energy.
Challenge #2: How to Read Back Just Interesting Rows

- Data known to be interesting only at simulation end
- Indexing only works when all rows are indexed at all simulation timesteps
- Compute node resources are limited
- Sorting only helps 1 query

Can I just create a small index for those interesting rows?
Existing Solutions Fall Short in Different Ways

**Post-processing**
- 1. Simulation
- 2. Indexing
- 3. Analysis

*Excessive data movement*
*Requires additional compute nodes than the job*
*Does not work for large jobs*

**In-transit processing**
- 1. Simulation
- 2. Indexing
- 3. Analysis

**In-situ processing**
- 1. Simulation
- 2. Analysis

*Can only index a single column*
*Does not work for multi-dimensional queries*
Opportunities for Rapid Query Acceleration

- **Today**: all computation takes place on compute nodes
- Excessive data movements or reduced index quality or increased per-job resource footprint
- **Computational storage** allows for overcoming existing solution limitations (by offloading compute to storage reducing data movement)
MarFS: LANL’s Current Cool Storage Tier

Multi-level erasure coding
Overview of LANL’s Multi-Tier Storage Infrastructure

- **Burst buffer**
  - 3.2TB/s
- **Parallel FS**
  - 1.2TB/s
- **Campaign storage**
  - 50-100GB/s
- **Tape**
  - 10GB/s
Multi-Level Erasure Coding in MarFS

- Layered data protection domains for localized repair
- Data & parity round-robin to storage nodes
- Multiple JBODs per storage node using SMR drives
Cost-Effective Data Protection Through Localization

Most rebuilds are done within a single OSD without being limited by inter-OSD bandwidth

Increased storage overhead for parity

Performance depends on having a high inter-OSD bandwidth

Lower parity overhead
Towards a Computational Campaign Storage Tier

- Upper layer writes data in a **columnar format** with lightweight indexes (min, max) every MBs of data
- Campaign constructs detailed row-level indexes offline
- Queries run on storage incurring **minimal data movement**
Kinetic: A Seagate’s Research Active Disk Platform

CS-HDD for in-drive computation
Computational Storage for High-Density Disk Drives

- **Kinetic HDD** = Disk + an envoy card
- **Envoy:**
  - CPU: 2x ARM Cortex-A53 cores
  - RAM: 1GB
  - OS: Ubuntu 20.04.4 (Linux 5.16.17)
  - Network: 2x Ethernet ports
    - 2.5Gb/s per port
- Ethernet uses the standard SAS connector interface with repurposed pin outs
96 Drives Per Kinetic Chassis

- **Total capacity:** 1.5PB (96 x 16TB)
- **Each drive has an IP** (acts like a server)
- **Total bandwidth:**
  240Gb/s (96 x 2.5Gb/s)
  - Fully subscribed
- **HA:** 2 sets of switches per chassis, 2 NIC ports per drive
Ordered KV Based Storage

- **Kinetic API**
  - Ping
  - Device Erase (like mkfs)
  - Put, Get, Delete
  - Iteration: GetPrev, GetNext, GetRange
  - Exec (eBPF or C++ progs)
- More info: https://gitlab.com/kinetic-storage/libkinetic

Production version may differ
Transform MarFS to a Computational Campaign

- **Kinetic HDDs** for near-storage data computation (async index creation & query processing)
- Latest MarFS & ZFS file mapping for locating per-dataset disk blocks
- A shim layer that translates disk LBAs to Kinetic KVs

![Diagram showing data path and analytics path with Seagate Kinetic HDDs, Intel ISA-L, ZFS, and analysis code (eBPF) stores results as new KV pairs on storage for later retrieval.]
C2: LANL’s Next-Gen Campaign Storage

The world’s first storage system to achieve analytics on disk under erasure
**Prototype Implementation**

- **LibZDB** (LANL’s stripped-down version of ZDB) for mapping ZFS filenames to disk LBAs
- Drive exposed as an NVMeOF block device to ZFS
- Custom C++ code for in-drive analytics

![Diagram showing the integration of LibZDB, ZFS, and NVMeOF with analytics and data path]

Current Prototype

 Longer-term Design

-Multi-level Erasure Coding

C2 Gateway

Intel ISA-L

C2 OSD ZFS RAID

C2 OSD ZFS RAID

C2 OSD ZFS RAID

Analytics Path

Data Path

LibZDB

ZFS

custom client

/dev/nvme1n1

NVMeOF

Kinetic HDD

NVMe Target

LibZDB

ZFS

/dev/sda

Kinetic Block Device (Kernel)

KVCV

KVCV

STORAGE DEVELOPER CONFERENCE
Data Format & Alignment

- Many apps store data as arrays of indivisible units (that must be read in their entirety for analytics)

- Storage writes data to drives without necessarily considering unit boundaries

- A drive may not see an entire record
Co-Design Data with ZFS RAID Schemes

- ZFS divides files into records (size configurable via ZFS_recordsize)
- Records are individually RAID’ed over disk drives
- Co-designing ZFS records with RAID configurations enables alignment control
RAID-Aligned Parquet Row Groups

- App writes columnar data to C2
- C2 re-formats it to RAID-aligned Parquet row group for in-drive computation
- Each drive sees entire Parquet row groups

C2 formats each row group to be exactly 1MB and puts per row group metadata at the end of that 1MB

On ZFS, each parquet file is physically stored as a directory of 2 container files

Storage overhead due to formatting: 0.4%
Experiment: One ZFS Host, 5 Kinetic HDDs

- A particle dataset from a real-world scientific simulation: 500 million particles (24 bytes each, 12GB total)

- 11K Parquet row groups (around 2.3K row groups per drive)

Query: SELECT * FROM particles WHERE Ke>Value

<table>
<thead>
<tr>
<th>Ke</th>
<th>Ke</th>
<th>Ke</th>
<th>Ke</th>
<th>Ke</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;0.7</td>
<td>&gt;0.6</td>
<td>&gt;0.5</td>
<td>&gt;0.4</td>
<td>&gt;0.3</td>
</tr>
<tr>
<td>0</td>
<td>80</td>
<td>2.5K</td>
<td>63K</td>
<td>1104K</td>
</tr>
</tbody>
</table>

Hits per query

Particle Schema

<table>
<thead>
<tr>
<th>ID</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Ke</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint64</td>
<td>float</td>
<td>float</td>
<td>float</td>
<td>float</td>
</tr>
</tbody>
</table>
Kinetic In-Drive Analytics Up to 5x Faster

OS page cache purged before each query

Query Conditions

High query selectivity

Low query selectivity

Baseline is limited by data movement

C2-Kinetic is limited by CPU

C2-Kinetic 5X Faster

C2-Kinetic 4.5X Faster

C2-Kinetic 2X Faster

Ke>0.7

Ke>0.6

Ke>0.5

Ke>0.4

Ke>0.3
Future Work

- Larger scale, more drives
- Asynchronous index construction in drives
- More levels of erasure coding
Conclusion

- Massive data movement has become a key bottleneck for large-scale scientific data analytics
- Near-data computation provides opportunities for rapidly searching big data with minimal data movement
- Having the flexibility to move compute to where it performs the best will become increasingly important as market evolves quickly
- C2 just demonstrated that in-drive data computation can co-exist with erasure coding while speeding up scientific discovery
LANL-Seagate Campaign Storage Design Team

- Jason Lee (jasonlee@lanl.gov)
- Brian Atkinson (batkinson@lanl.gov)
- Jarrett Crews (jarrett@lanl.gov)
- David Bonnie (dbonnie@lanl.gov)
- Dominic Manno (dmanno@lanl.gov)
- Gary Grider (ggrider@lanl.gov)

- Philip Kufeldt (philip.kufeldt@seagate.com)
- Ivan Rodriguez (ivan.rodriguez@seagate.com)
- Evan Burgess (evan.burgess@seagate.com)
- David Allen (david.j.allen@seagate.com)
- Bradley Settlemyer (bsettlemyer@nvidia.com)
Please take a moment to rate this session.

Your feedback is important to us.