Los Alamos
NATIONAL LABORATORY

Kinetic Pipe: LANL-Seagate’s Early Prototype for Near-Data, SQL-Like Query Processing

Qing Zheng, Scientist, Los Alamos National Laboratory

3/15/2023

LA-UR-23-22598

Overview

Problem

Scientific analytics increasingly bottlenecked on large data transfers

Goal

Evaluate how/how much in-drive analytics can help

Kinetic Pipe

An early prototype for assessing gains

Results

Sizeable speedups even when data transfer is not the primary bottleneck

Background: Scientific Datasets

Resemble tables with rows and columns

- Rows: records
- Columns: attributes

Traditional HPC data formats: HDF5, NetCDF (self describing, parallel io, offset-based query interface)

We are also looking at leveraging industrial data formats (such as Apache Parquet, ORC, Avro) and analytics stacks to enable richer query types beyond offsets (e.g.: SQL)

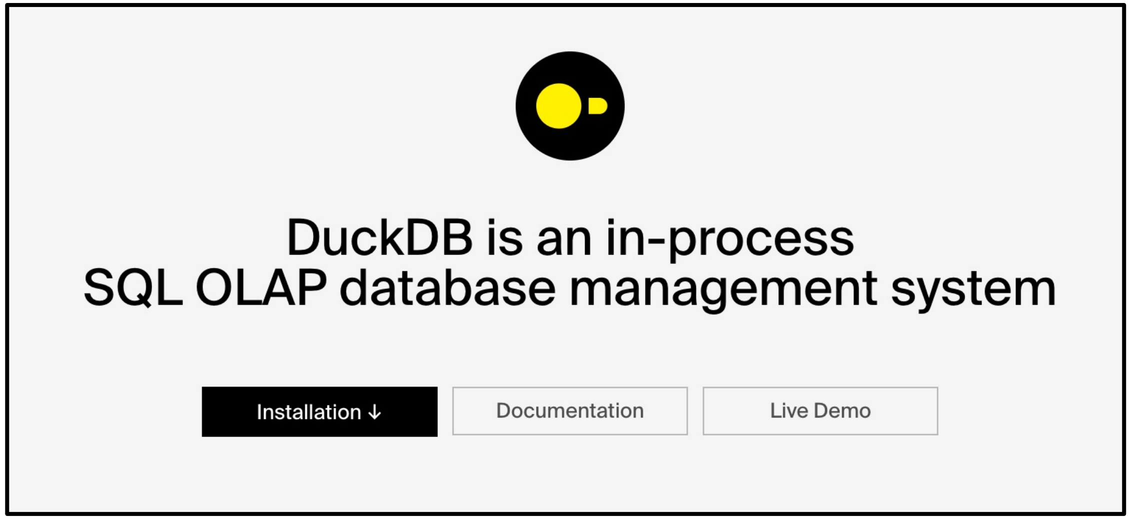
An Example

Store data in Parquet

- Columnar data model
  - Row Groups
    - Column Chunks
- Self describing
- Lightweight min-max indexes per column per row group

Run Queries using DuckDB

- Supports SQL
- Understands Parquet




```
SELECT * FROM 'test.parquet'
WHERE X>Y
```

HPC Workflows

Simulation Phase

- User submits jobs
- Jobs run on compute nodes
- Jobs generate data (e.g.: in Parquet)
- Data is written to backend storage
- Storage likely tiered

Analytics Phase

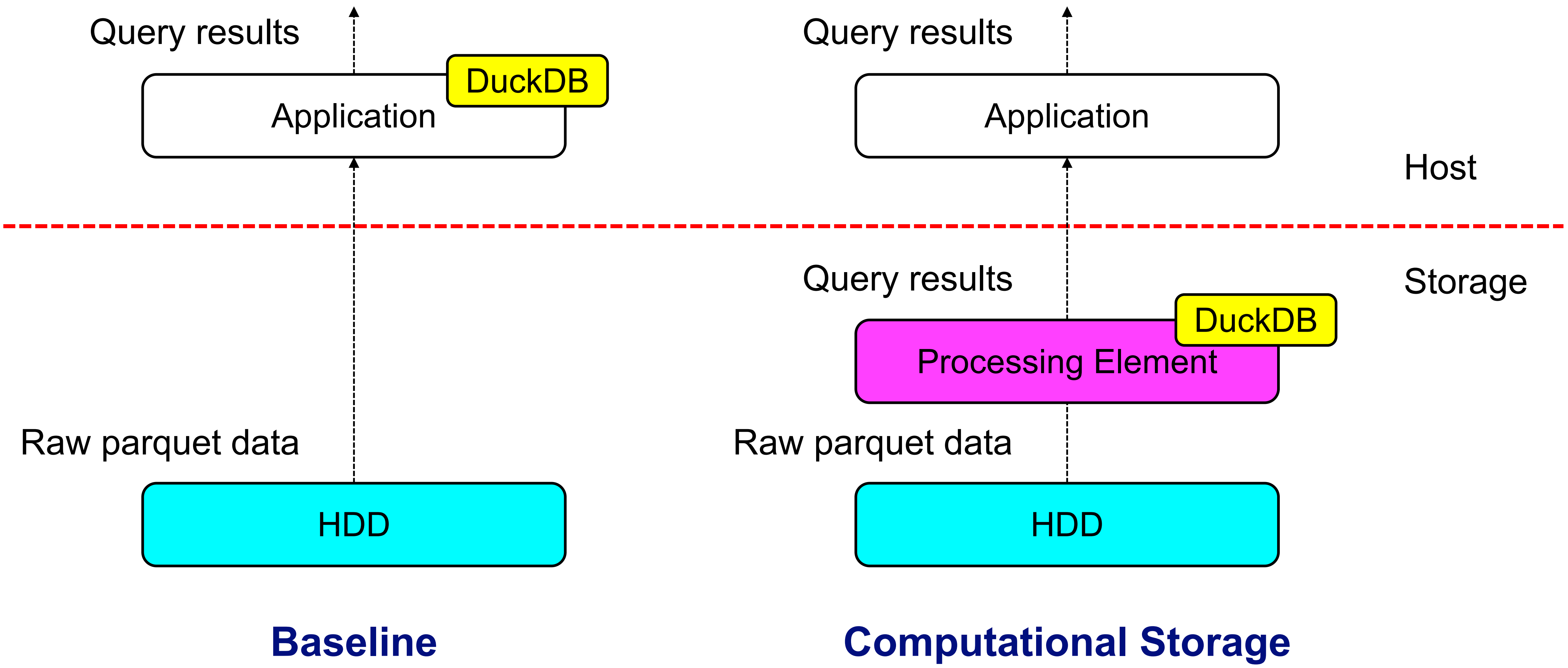
- User runs queries against their data (e.g.: DuckDB)
- **A query may select only a tiny amount of data from a large dataset**
- Queries run *slowly* when a large amount of data is moved

Can we return only data that is selected by a query?

Time to Read Back 1PB of Data

Burst Buffer	Local Scratch	Shared Scratch	Campaign Storage	Tape Archive
3.2TB/s	1.2TB/s	300GB/s	100GB/s	10GB/s
312s	**14min**	**56min**	**2.8hr**	**28hr**

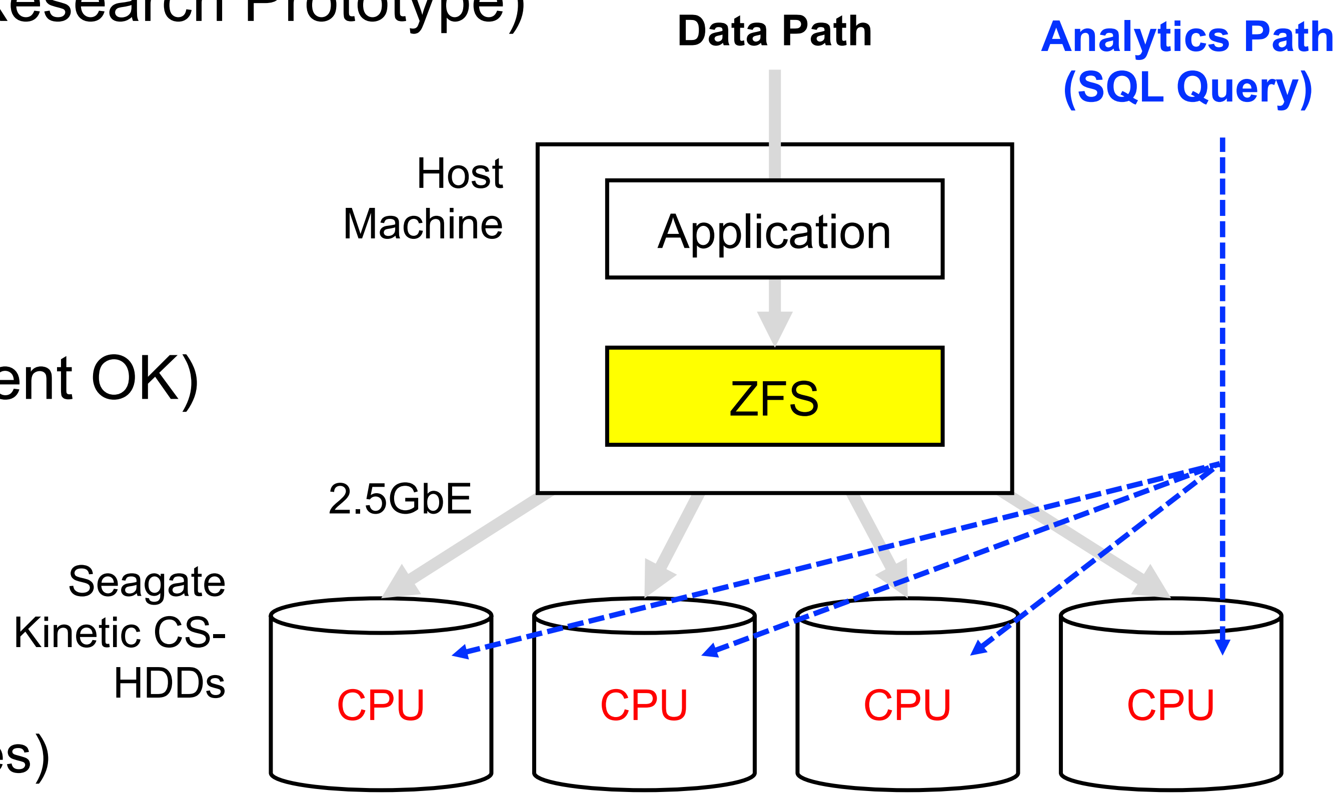
Why Computational Storage Might Help



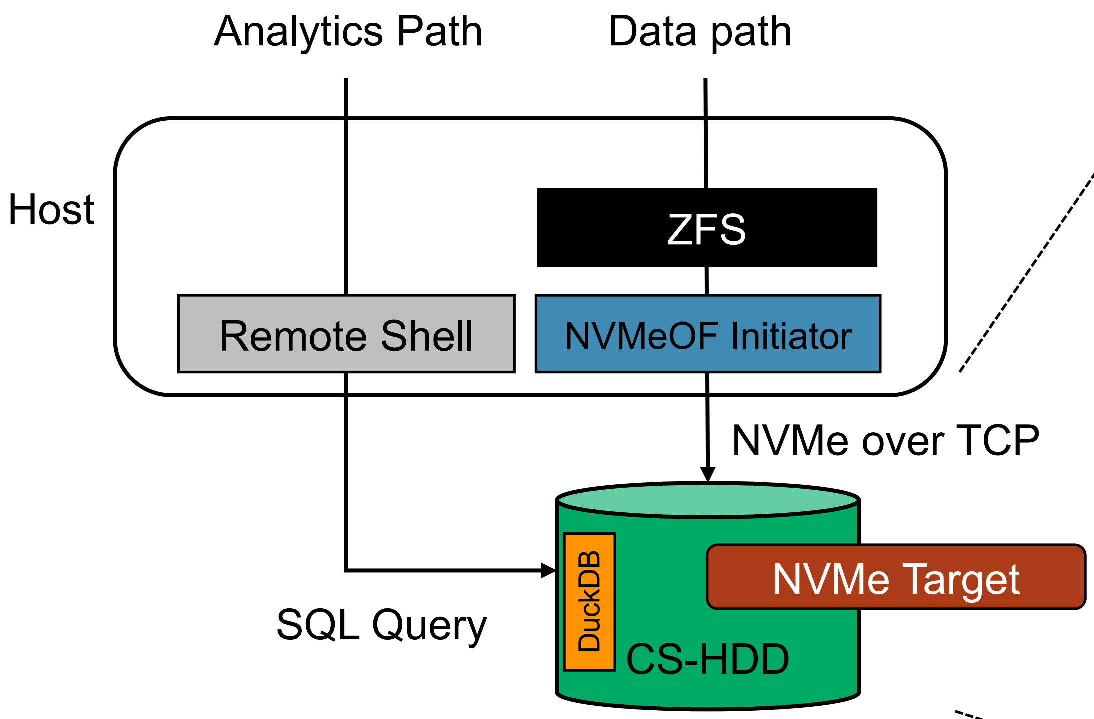
Kinetic Pipe

Our first near-data analytics prototype for cool storage tiers

- **Disk**: Kinetic CS-HDDs (Seagate's Research Prototype)
  - CPU: 2x ARM Cortex-A53 cores
  - RAM: 1GB
  - OS: Ubuntu Linux (C++ development OK)
  - Network: 2x 2.5GbE
- **Host Filesystem**: ZFS
  - Data protection: **RAID** (1, 2, or 3 parities)



A Close Look


Analytics Path
Data path
Host
ZFS
Remote Shell
NVMeOF Initiator
NVMe over TCP
DuckDB
NVMe Target
CS-HDD
SQL Query


SEAGATE
EXOS
ENTERPRISE
16TB
Exos™ X16
ST16000NM001G
SECURE
HDD Mfg by Seagate Technology LLC

Two Challenges

Drives have no knowledge of FS file-to-block mapping

- Solution: LibZDB (allow querying ZFS for mapping information)

A data row may be split over multiple drives

- Data alignment control

Evaluation

3 Scenarios

A) Host network is a bottleneck

- Can in-drive analytics improve performance?

B) Host CPU is a bottleneck

- Can in-drive analytics improve performance?

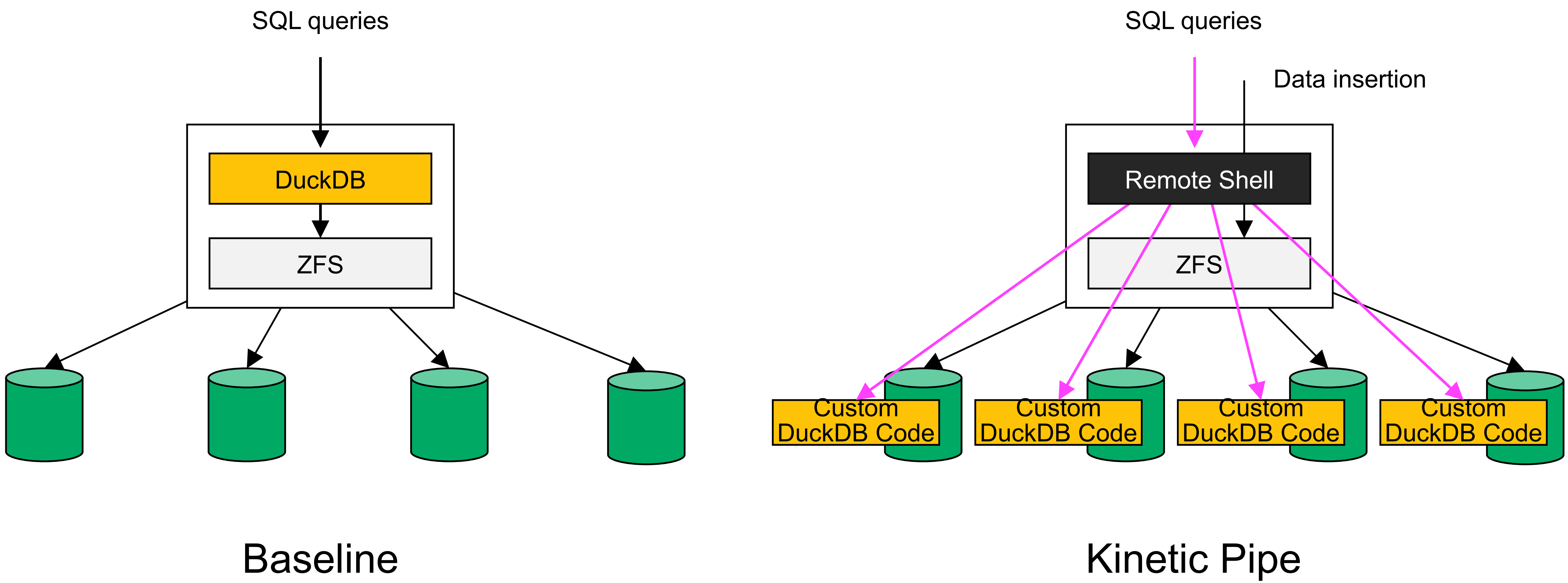
C) Host has abundant CPU & network

- Can in-drive analytics continue to improve performance?

Experiment Setup

- **1** ZFS host (32 AMD CPU cores)
- **38** CS-HDDs
  - 2x 16+3 **RAID** Pools
- **50GB** dataset from a real particle simulation
  - 2 billion rows (in Parquet fmt)
    - Columns: ID, x, y, z, ke
- **Two DuckDB queries**
  - SELECT ***** WHERE **ke>X**
  - SELECT **sum(ke)** WHERE **ke>X**

Baseline vs. Kinetic Runs

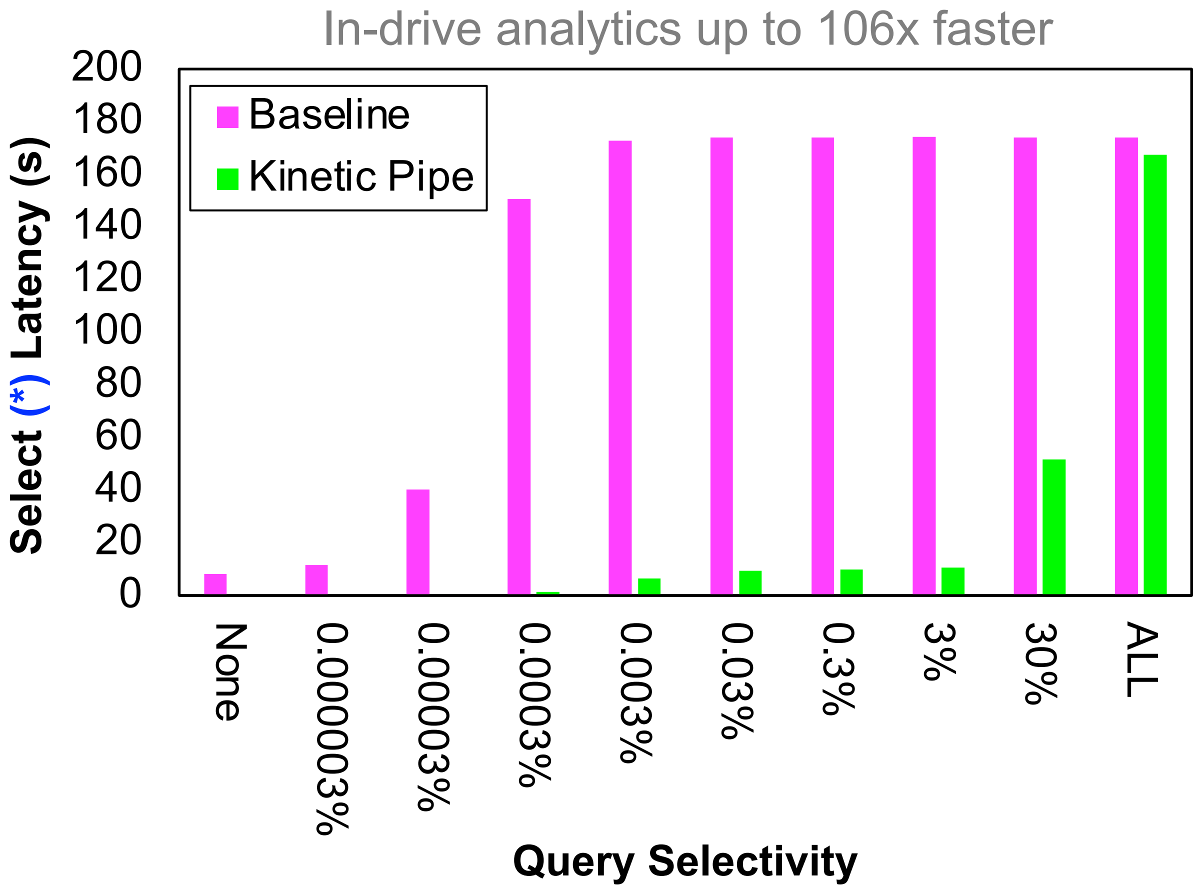


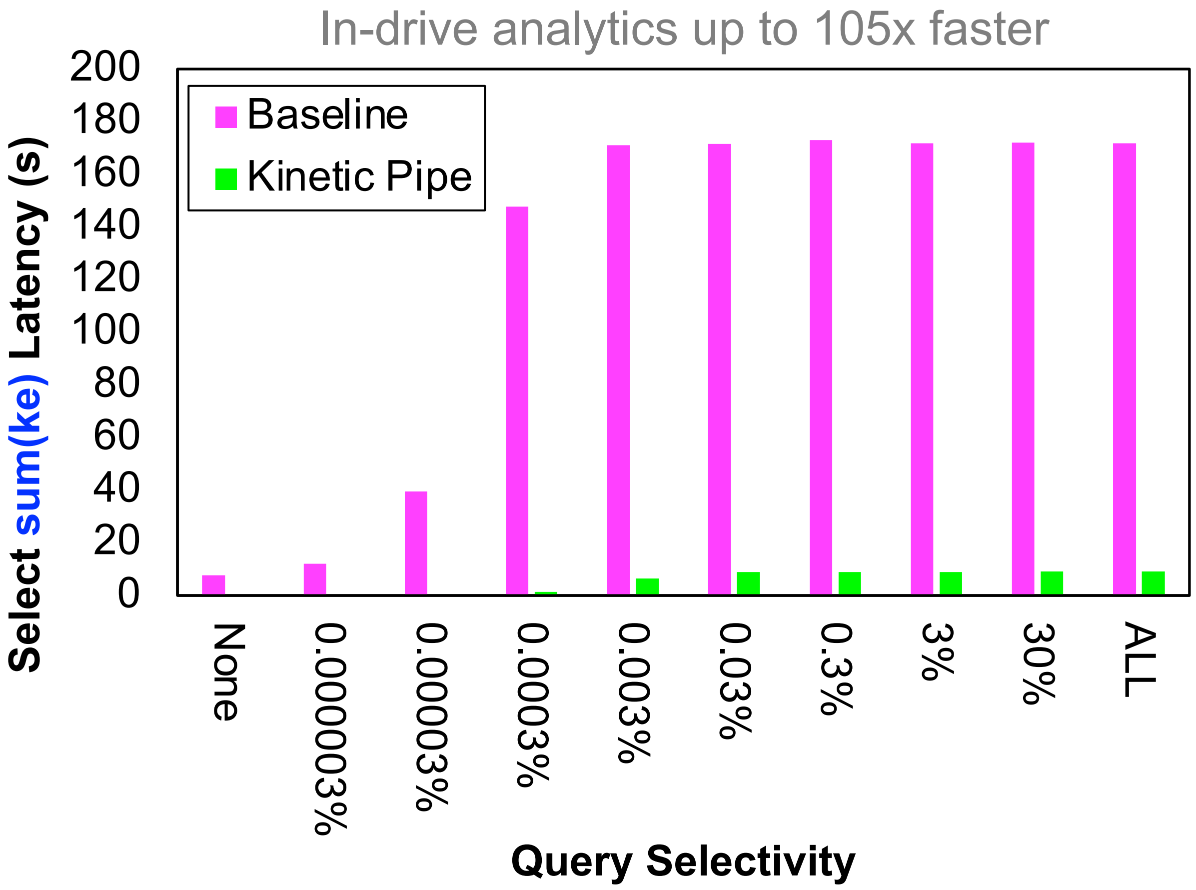
OS page cache purged at the beginning of each run

Result

In-drive analytics allow sending less data over the network

Case #1: Host network was the bottleneck



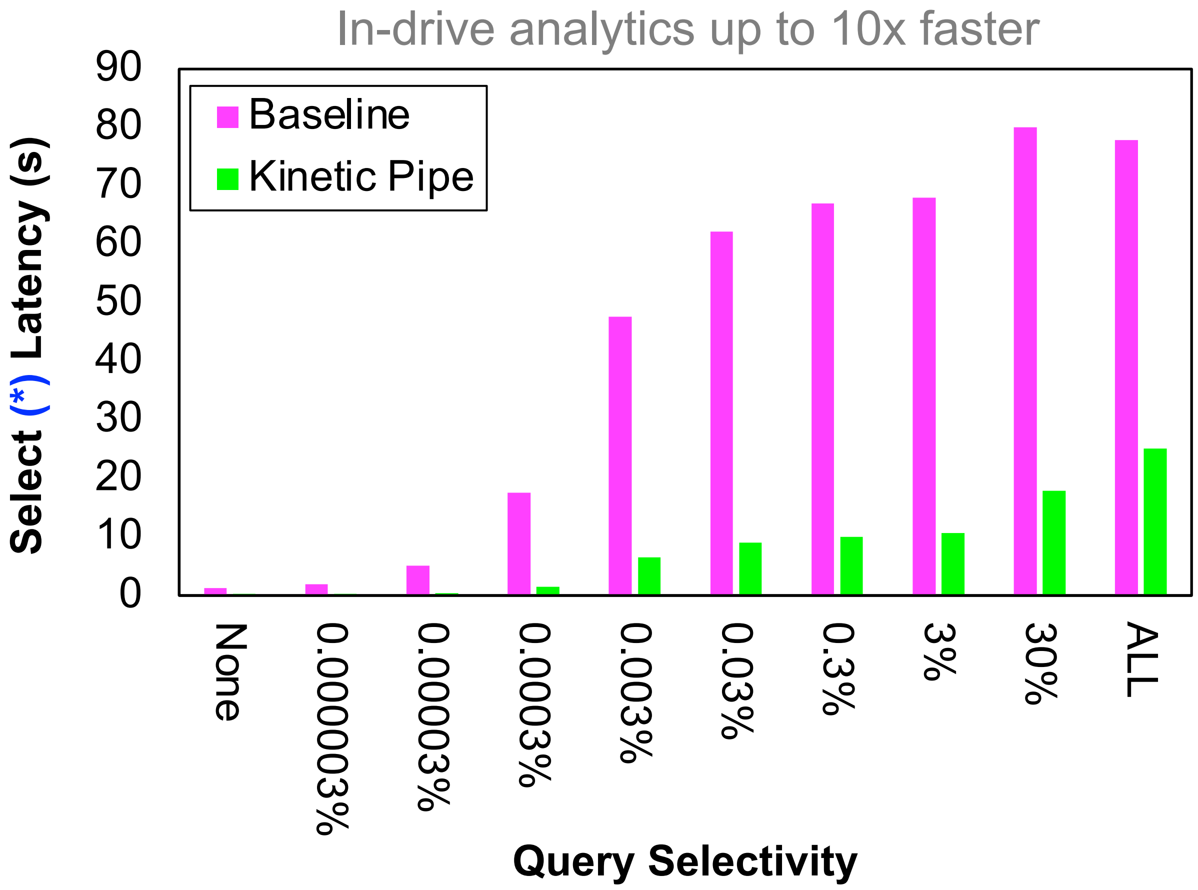


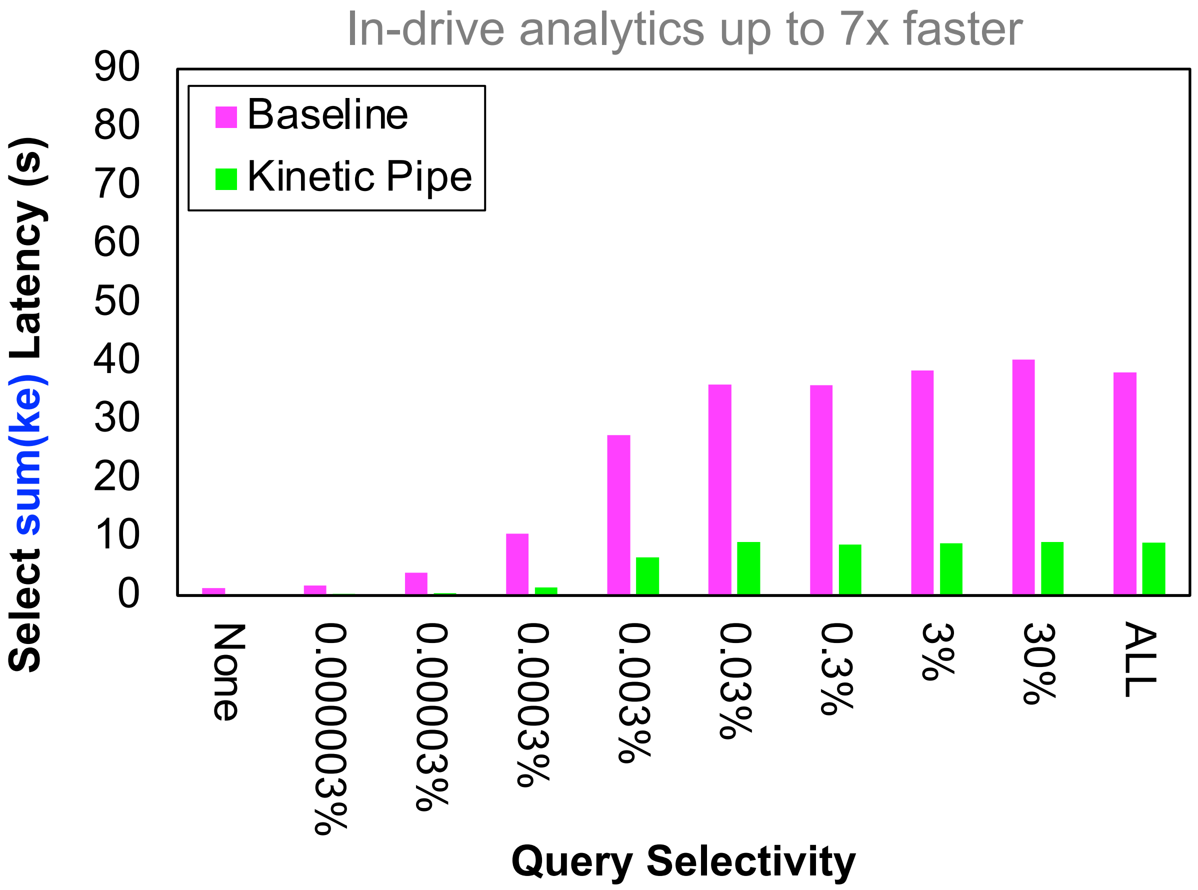
Host NIC bandwidth: 2.5gbit, CS-HDDs: 38x2.5=95gbit

Result

In-drive analytics allow massively parallel computing across drives

Case #2: Host CPU was the bottleneck



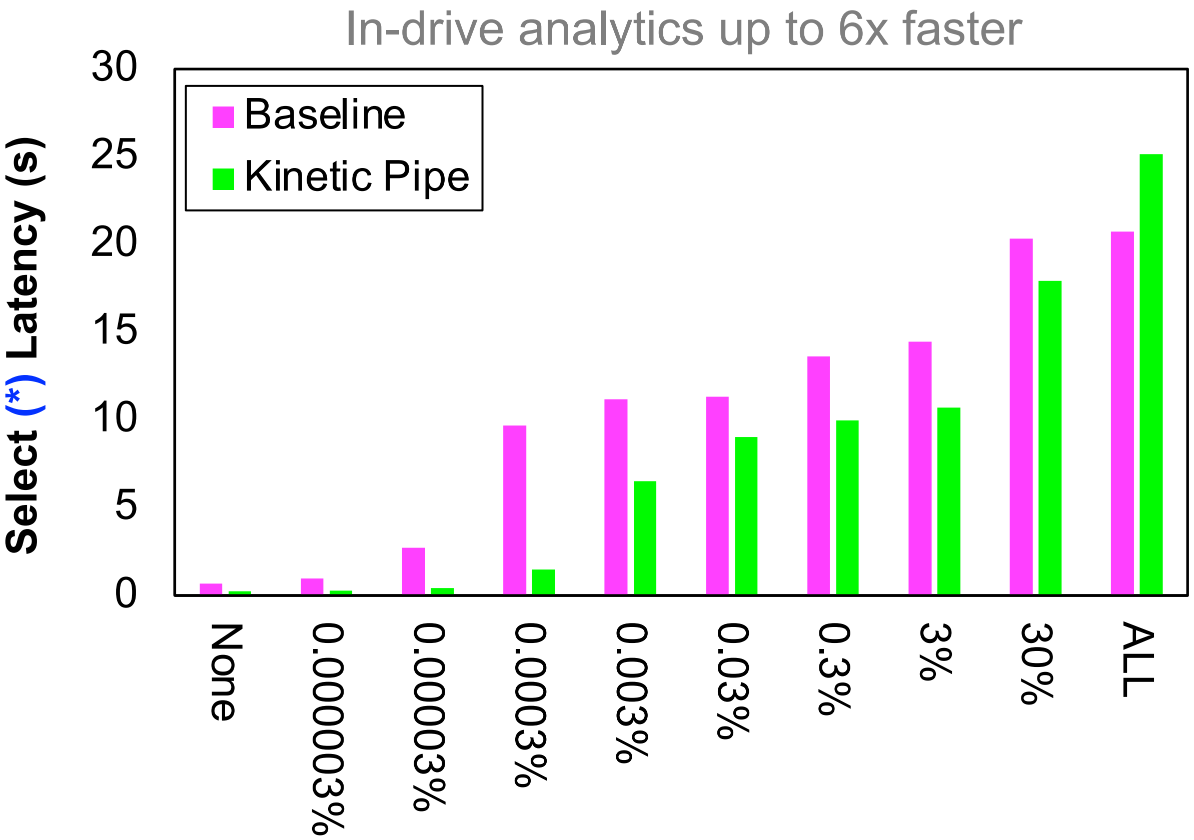


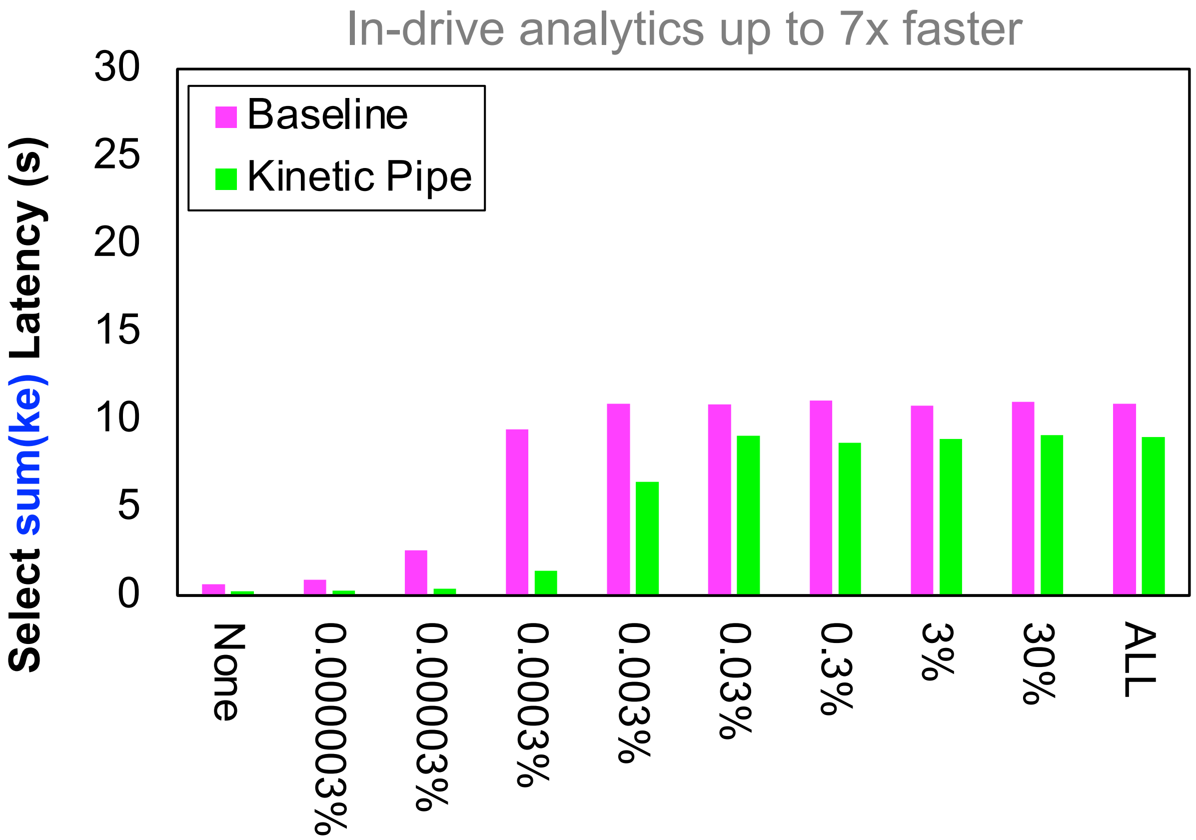
Host CPU: limited to 2 cores (AMD), CS-HDDs: 38x2=76 cores (ARM)

Result

In-drive analytics allow more fully utilizing disk bandwidth

Case #3: Host had abundant network & CPU





Host NIC bandwidth: 4x40=160gbit, CS-HDDs: 38x2.5=95gbit

Conclusion

Computational storage provides new ways of accelerating data-intensive applications

In-drive data management schemes matter (O_DIRECT, clustered index)

Layer violation: “cheating” one filesystem may be possible; cheating multiple layers of filesystems is hard (FS internal load balancing, fail over, compression, concurrency control)

Future directions: Block-based acceleration to object-based acceleration

Acknowledgement

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)

Evan Burgess (evan.burgess@seagate.com)

Ivan Rodriguez (ivan.rodriguez@seagate.com)

David Allen (david.j.allen@seagate.com)

John Bent (john.bent@seagate.com)

Bradley Settlemyer (bsettlemyer@nvidia.com)

Thank you!