Los Alamos NATIONAL LABORATORY

Analytics Query Pushdown Using Object-Based Computational Storage

Qing Zheng, Scientist, Los Alamos National Laboratory

6/3/2024

LA-UR-24-25418

Overview

Goal

- Rapid insight generation

Problem

- Scientific analysis often read more data than is necessary

Approach

- Execute queries closer to data using computational storage

OCS

Object-based Computational Storage

An effort in exploring an **open** object-based computational storage API for analysis query pushdown

A collaboration between SK hynix, Airmettle, Neuroblade, and Versity

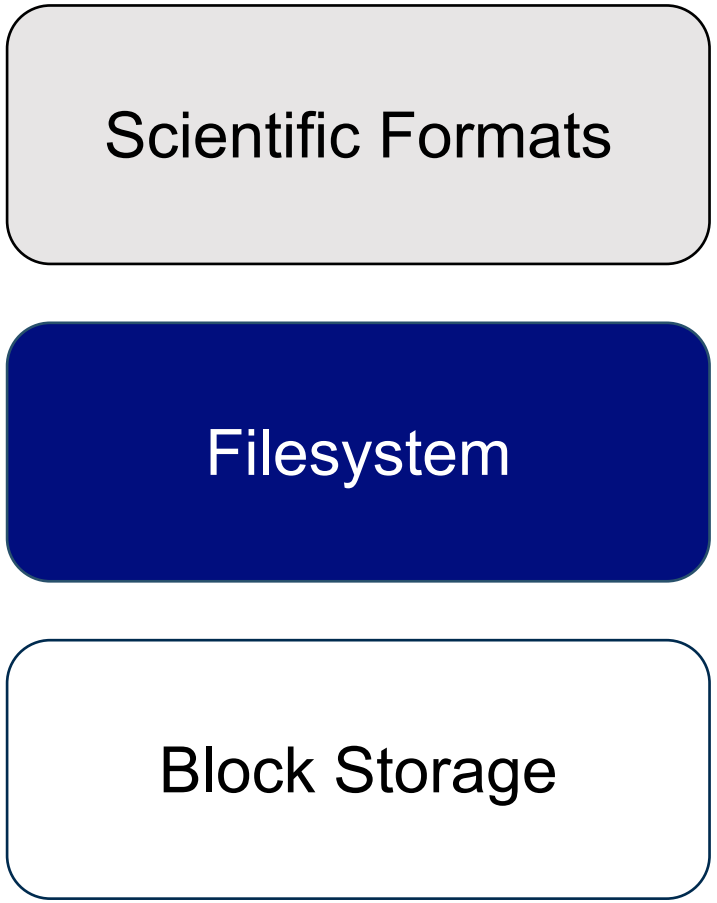
Background: Scientific Storage I/O Stack

Filesystem over blocks

Popular data formats: VTK, HDF5, NetCDF, …

- Self-describing
- Columnar (each column is a data array)
- Offset-based data access methods
- Geometry data (points, cells)

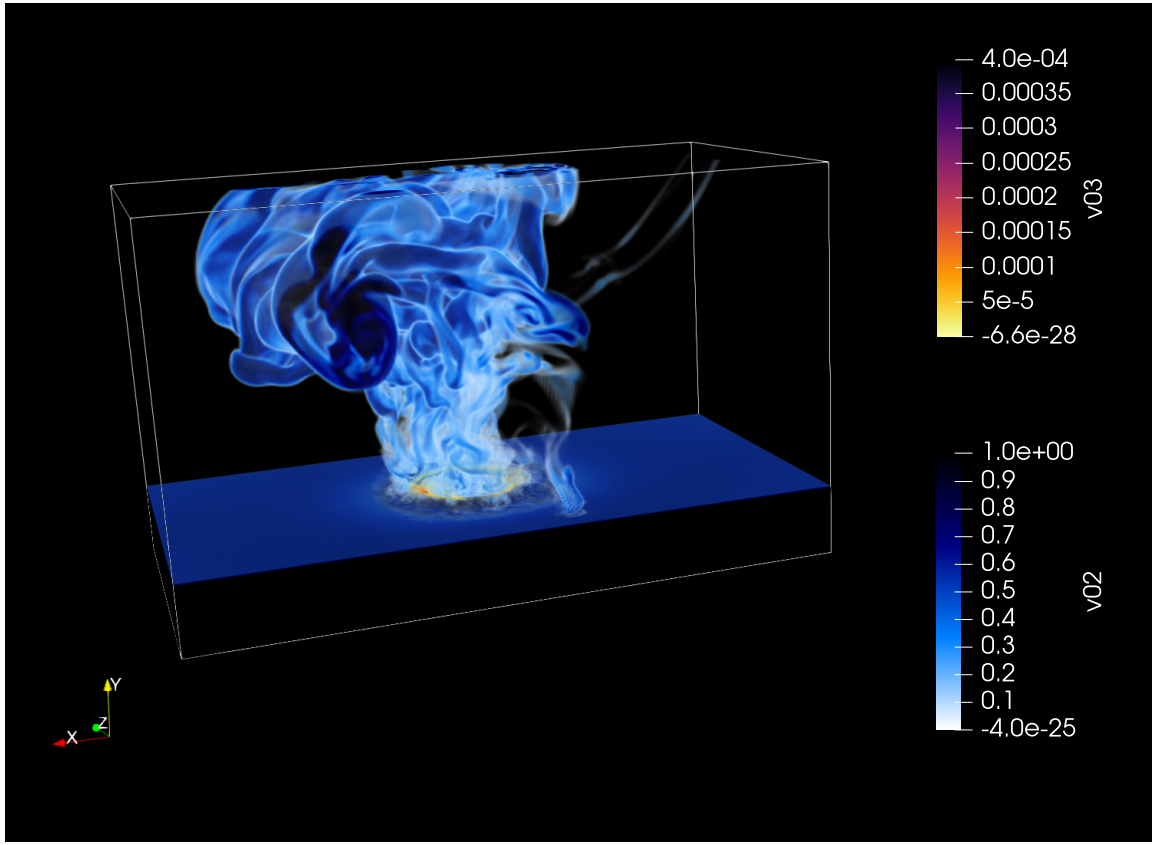
Data agnostic erasure protection at filesystem level



Example: Deep Water Asteroid Impact

Unstructured grid, 216M points, 182M cells, 11 data arrays (columns), 182M rows

	Column	Type	Description
1	rho	float	density in grams per cubic centimeter
2	prs	float	pressure in microbars
3	tev	float	temperature in electronvolt
4	xdt	float	x component vectors in centimeters per second
5	ydt	float	y component vectors in centimeters per second
6	zdt	float	z component vectors in centimeters per second
7	snd	float	sound speed in centimeters per second
8	grd	float	AMR grid refinement level
9	mat	float	material number id
10	v02	float	volume fraction of water
11	v03	float	volume fraction of asteroid

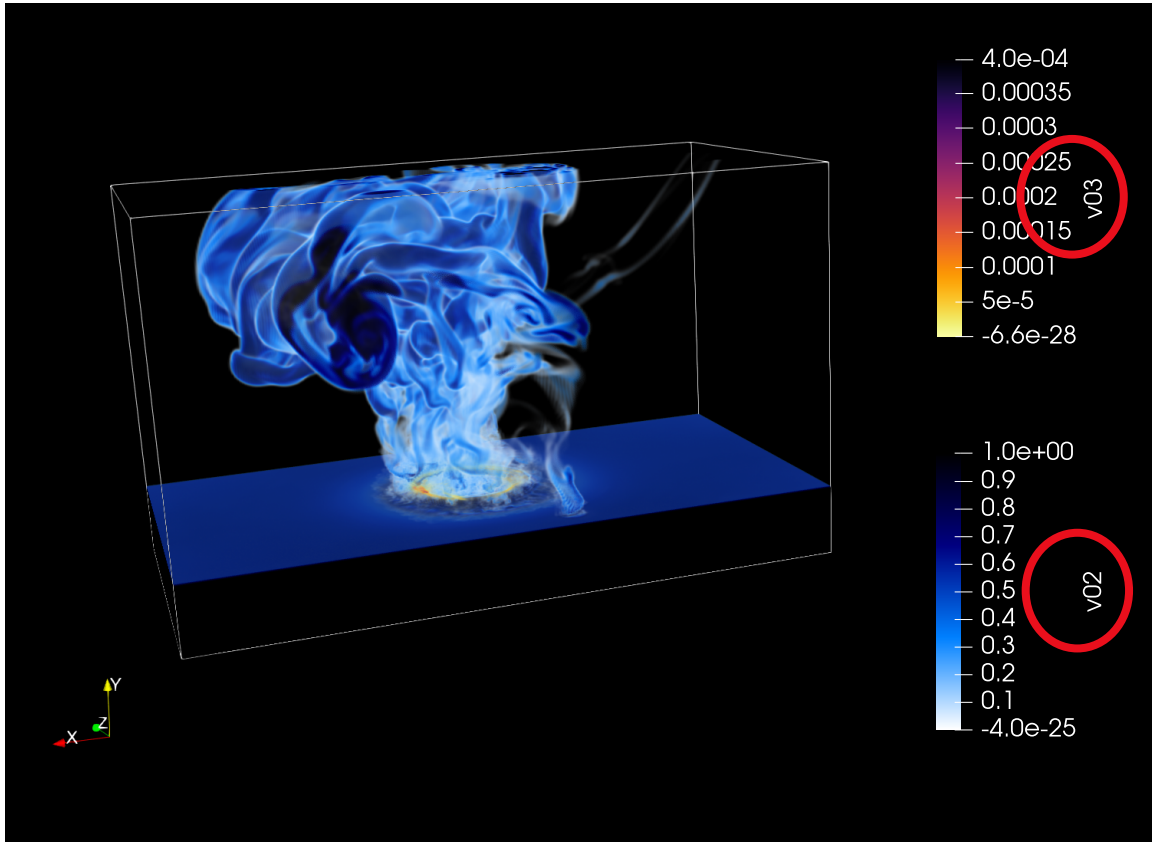


8GB/timestep

Analysis Rarely Uses All Columns

Existing scientific formats support efficient column skipping (by being columnar)

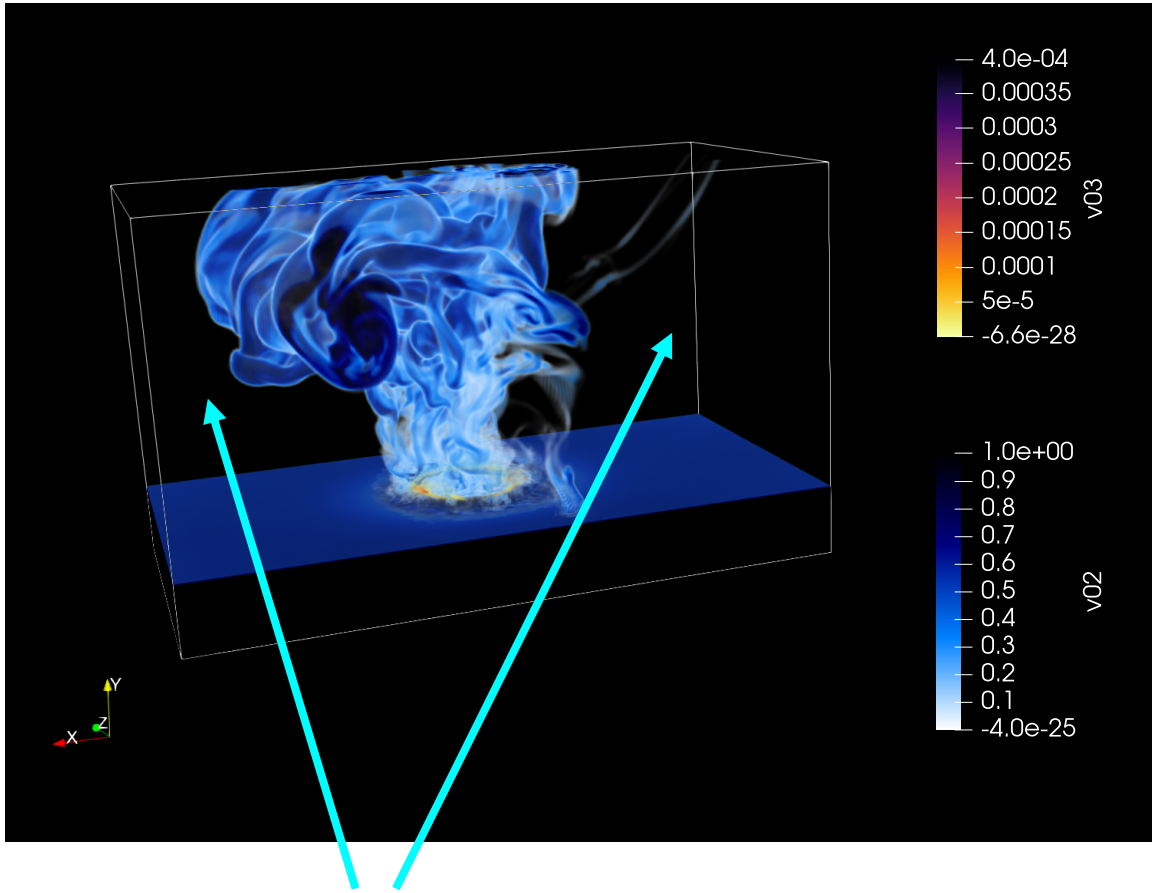
	Column	Type	Description
1	rho	float	density in grams per cubic centimeter
2	prs	float	pressure in microbars
3	tev	float	temperature in electronvolt
4	xdt	float	x component vectors in centimeters per second
5	ydt	float	y component vectors in centimeters per second
6	zdt	float	z component vectors in centimeters per second
7	snd	float	sound speed in centimeters per second
8	grd	float	AMR grid refinement level
9	mat	float	material number id
10	v02	float	volume fraction of water
11	v03	float	volume fraction of asteroid



How About Row Skipping?

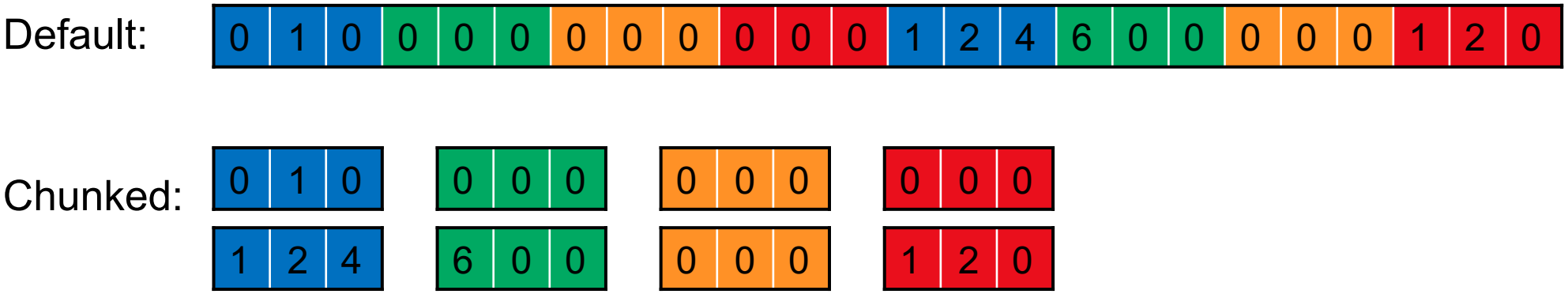
Today, all rows are read even when only a few are needed

	Column	Type	Description
1	rho	float	density in grams per cubic centimeter
2	prs	float	pressure in microbars
3	tev	float	temperature in electronvolt
4	xdt	float	x component vectors in centimeters per second
5	ydt	float	y component vectors in centimeters per second
6	zdt	float	z component vectors in centimeters per second
7	snd	float	sound speed in centimeters per second
8	grd	float	AMR grid refinement level
9	mat	float	material number id
10	v02	float	volume fraction of water
11	v03	float	volume fraction of asteroid



For example: it'd be nice if we could skip reading those "empty" cells

Mimicking Row Skipping With Chunking & Compression



The hope is for compression to reduce "0 0 0" chunks to almost nothing

- So that we don't pay much reading them

Purpose of chunking is to allow efficient subarray access (arr[n:m])

- Each chunk can be independently de/compressed

Current state-of-the-art in scientific world

Real World Predicates Are Often More Complex Than Skipping 0's

Challenges and opportunities: Complex queries tend to have higher selectivity

- Leaving opportunity for **more aggressive data reduction**
- And demand for **more advanced readers** to maximize reduction ratio

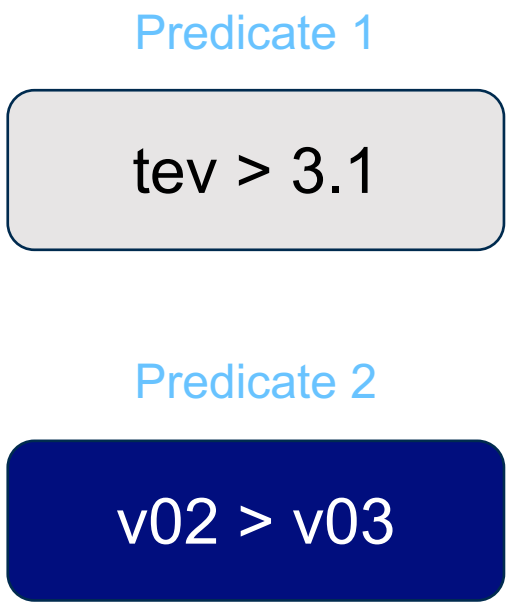
For example: SELECT v03 WHERE v03 > 9

- A reader can leverage per-chunk statistics (min/max values) to skip as many chunks as possible
- Existing scientific codes don't always perform this optimization

Real World Predicates Are Often More Complex Than Skipping 0's

Another example: SELECT v03 WHERE tev > 3.1 AND v02 > v03

- A reader may leverage per-column statistics to **estimate the selectivity of each predicate** and decide which one to serve as the primary filter
  - Use the primary filter to skip as many chunks as possible
- If user-supplied per-column indexes are available, also use them



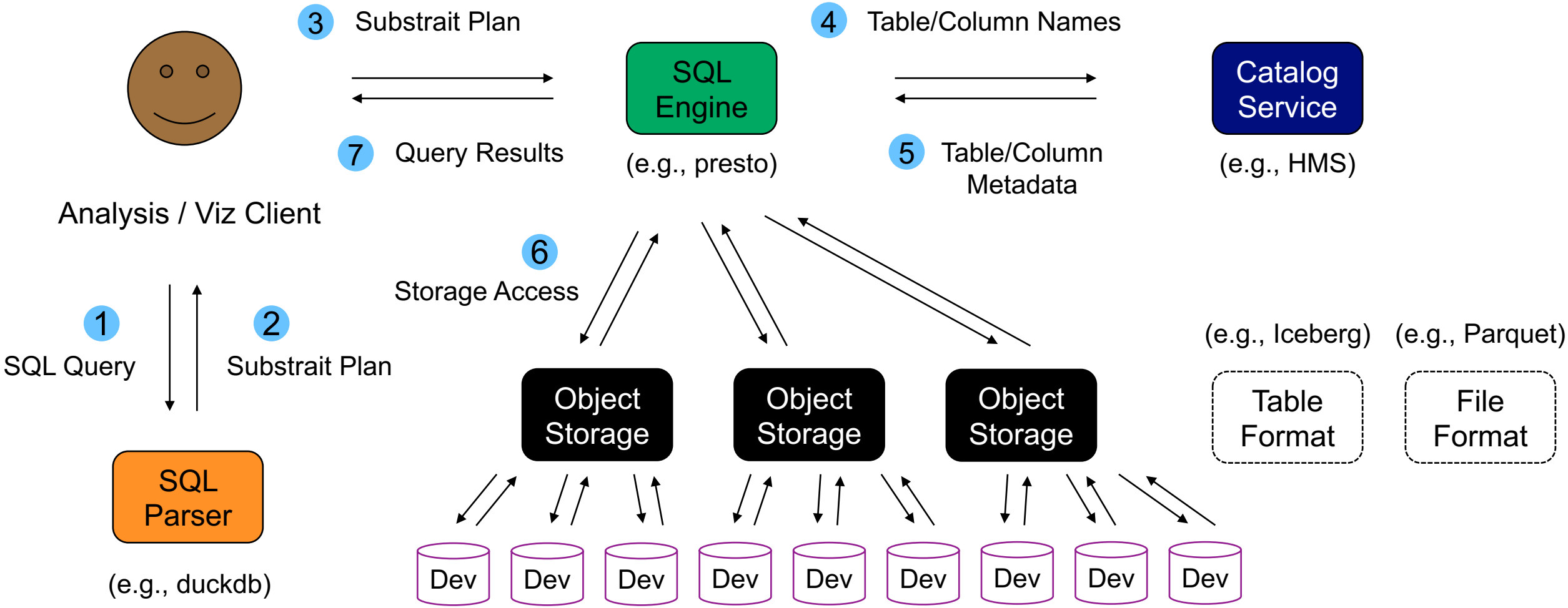
Not Reinventing the Wheel

Databases know how to best execute a SQL query

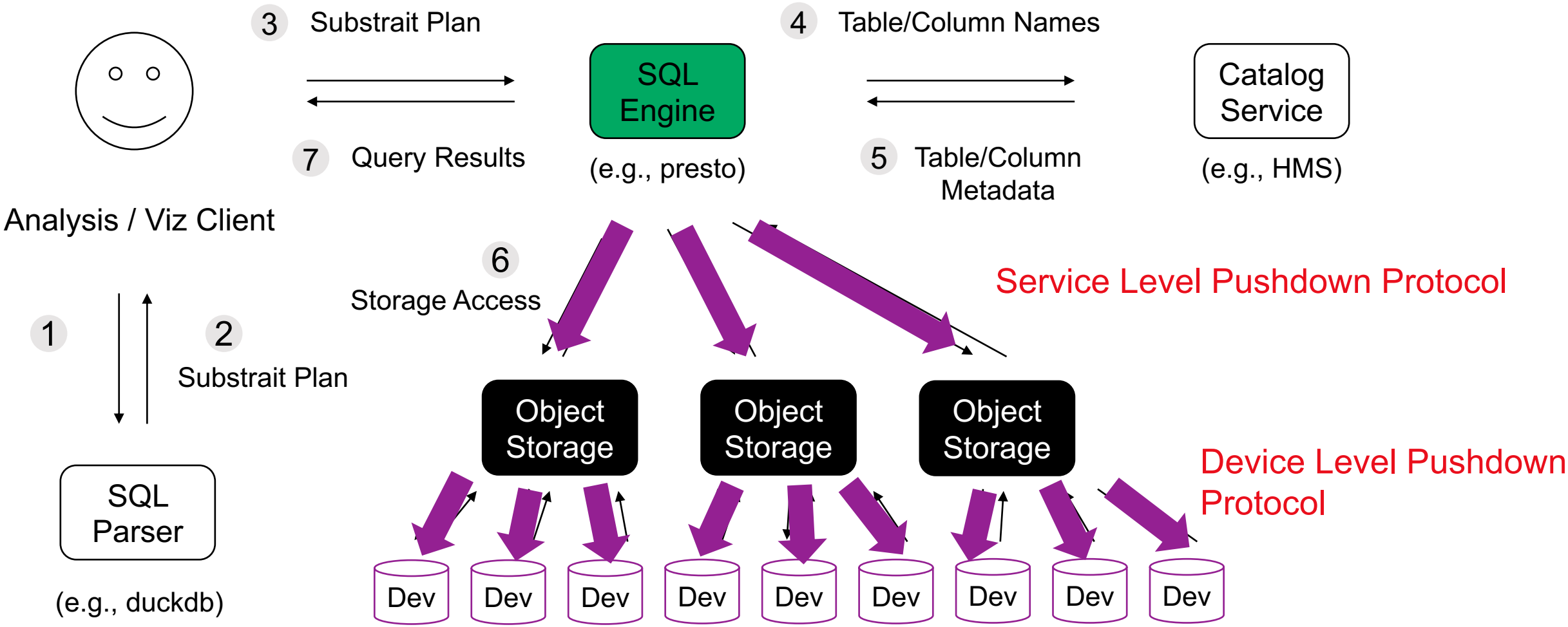
Composable databases are increasingly a thing

- SQL parser: Calcite
- Vendor neutral SQL query representation: Substrait
- Open-source SQL engines: Presto, Drill, Spark, Impala, Hive, DuckDB
- Catalog services: HMS (hive metastore)
- Open table formats: Iceberg, Hudi, Delta
- Columnar data formats: Parquet, Arrow, ORC
- Storage: S3 API
- Open query pushdown API for computational storage: ?

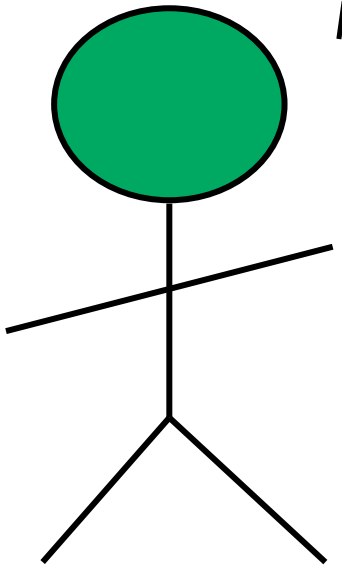
Converged, Open Analytics Stack for HPC and Non-HPC



Open Query Pushdown API for Computational Storage



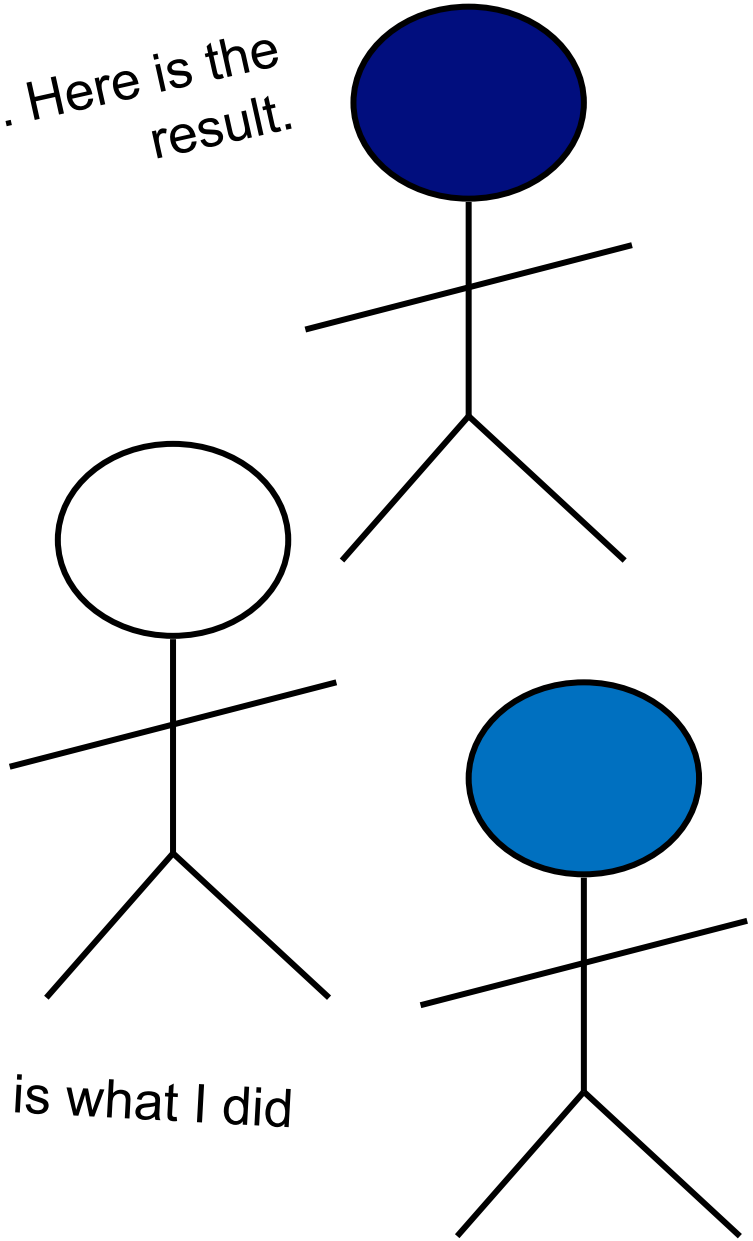
Division of Labor



Here is my substrait plan. Please run it.

I did nothing. Here is all the data that you might need.

I did everything. Here is the result.

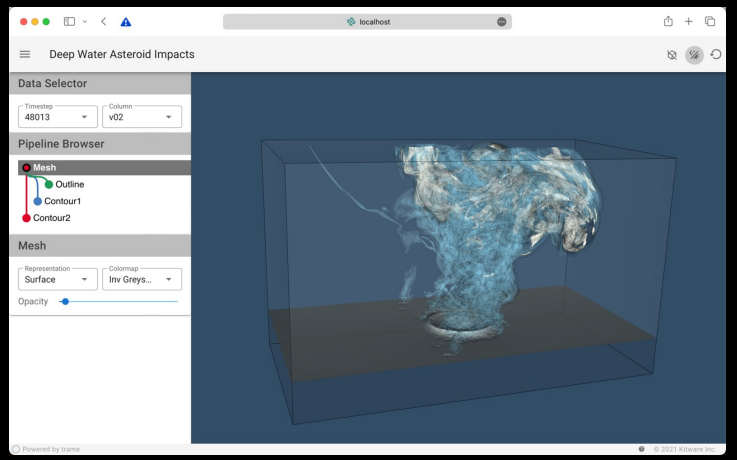


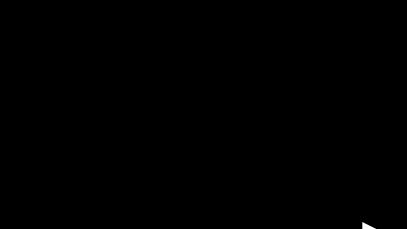
I did something. Here is what I did and my partial result.

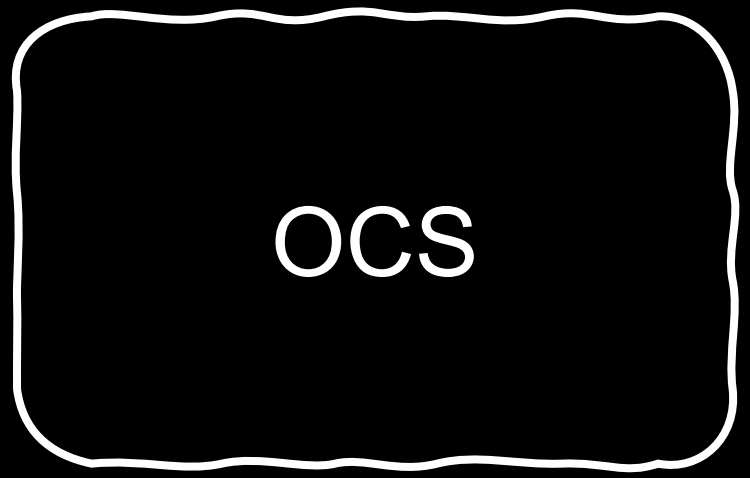
Upper Layer

Lower Layer

LANL/SK hynix FMS 2024 Live Demo @ Santa Clara Convention Center

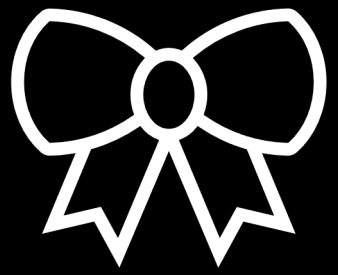




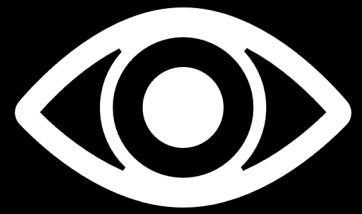


Real-world viz pipeline modified to leverage OCS open analytics stack to coordinate, plan, and run queries

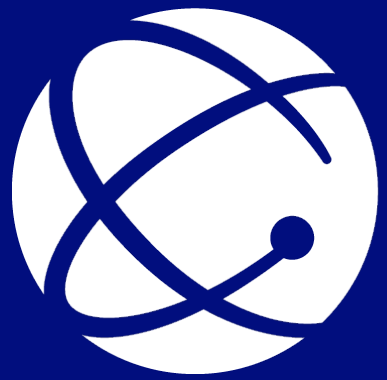
SK hynix's prototype OCS system implementing OCS pushdown APIs



Thank our collaborators: SK hynix, Airmettle, Neuroblade, and Versity



Look forward to seeing you all at FMS 24



Los Alamos

NATIONAL LABORATORY