LANL's Journey Toward Computational Storage

Computational Storage For Simulation Science

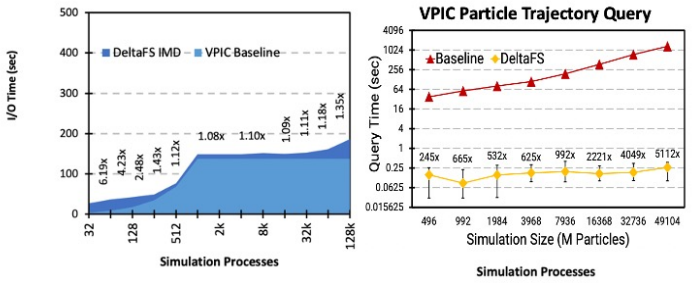
03/2024

Gary Grider
HPC Division Leader
Los Alamos National Laboratory

LA-UR-24-22150

Computational Storage Why?
Data Gravity is an Important Consideration

- Need flexibility in where computation is done (host, network, device) as economics will change over time
- Its not just energy and time to insight, some of these analytics require the same size analytics footprint as the simulation footprint (petabyte of ram) making analytics not always as feasible
- Data Agnostic Offloads
  - Server memory BW does not allow many passes over streaming data
- Data Aware Offloads
  - Analytics is often multiple orders of magnitude less reading than writing
  - You just have a hard time finding what you are looking for (filter/index/histogram/etc.)
  - Can we add metadata/indexing/ordering to data as it is written with almost no overhead and reap huge wins on read (time, hdwr resources, energy)
  - For Science
    - Particle methods -> “Ordered” row-based analytics (KV)
    - Grid methods -> columnar-based analytics
    - Large Complex Grid methods -> THE KITCHEN SINK



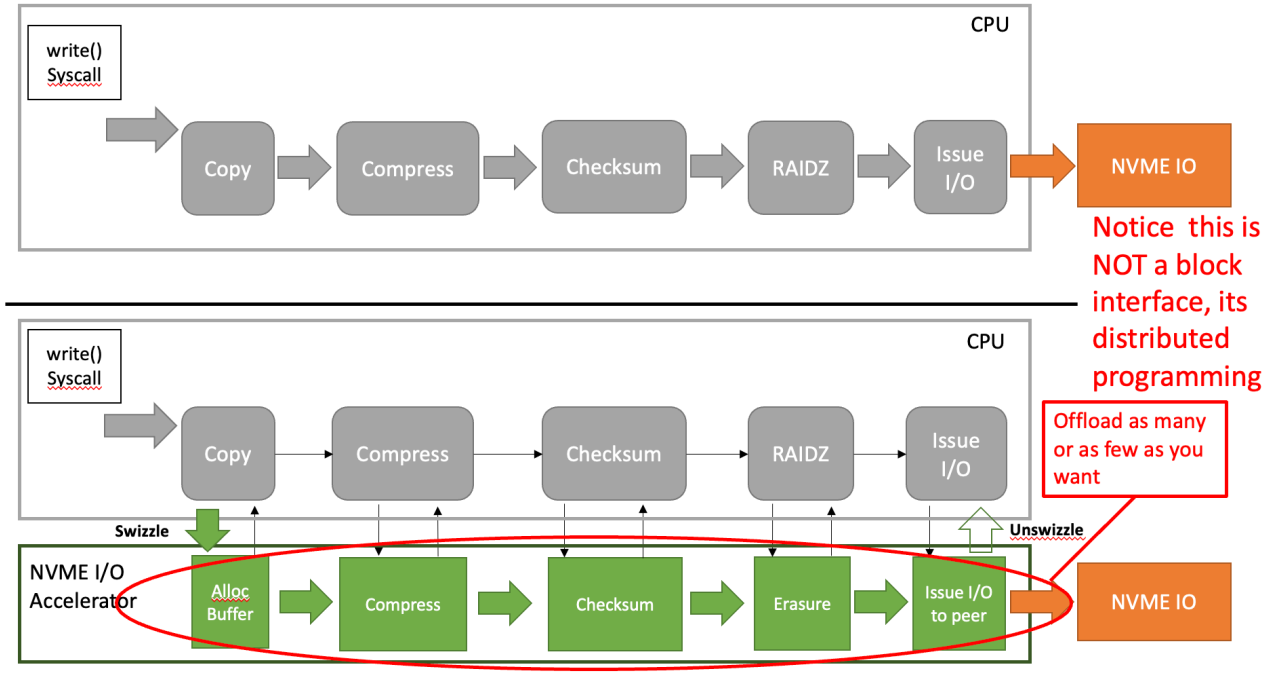
(Papers at PDSW 15, PDSW 17, SC19 (Best Student Paper)

Data Agnostic Offload (ABOF)

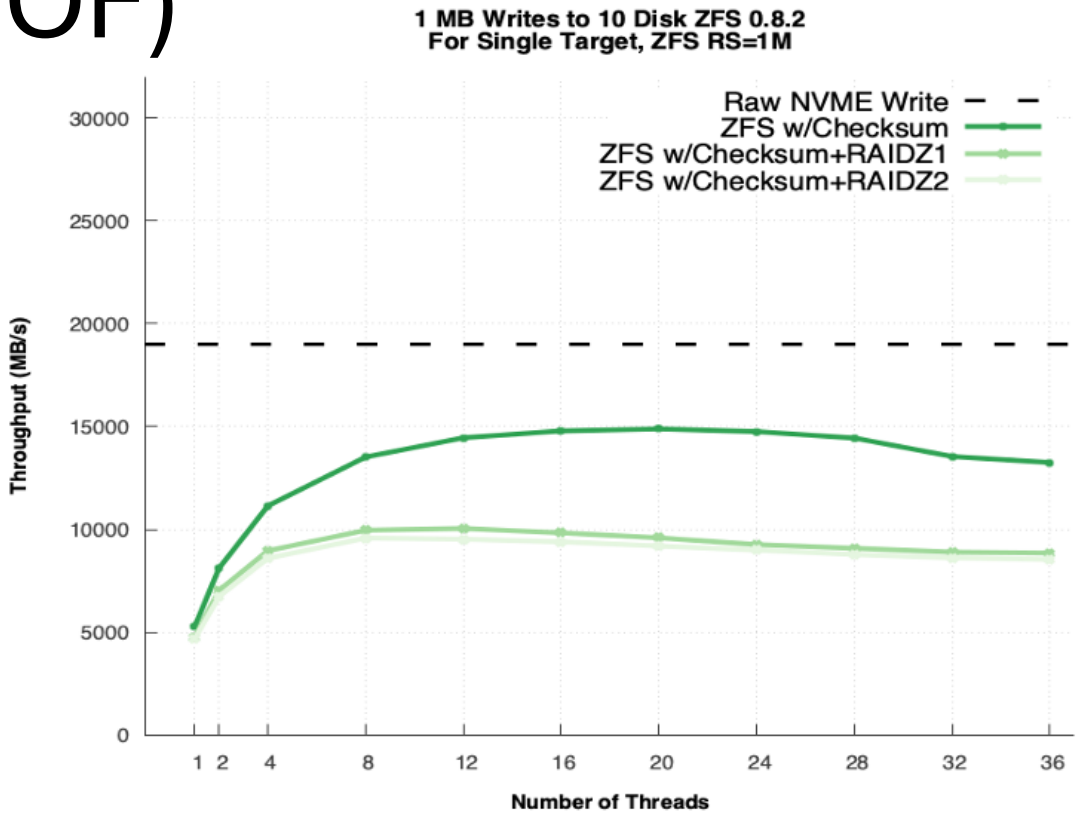
Offload erasure, encoding, compression and make it possible to run each anywhere you like (host, network/DPU, storage/CSA/CSP/CSD)

Consumable:

- Open ZFS
- Kernel module for offloads to register into
- ZFS mods unstreamed
- Tested under Lustre
- Data written is same, upon failure just fall back to host
- No app changes other than faster on less capable host



Notice this is NOT a block interface, its distributed programming

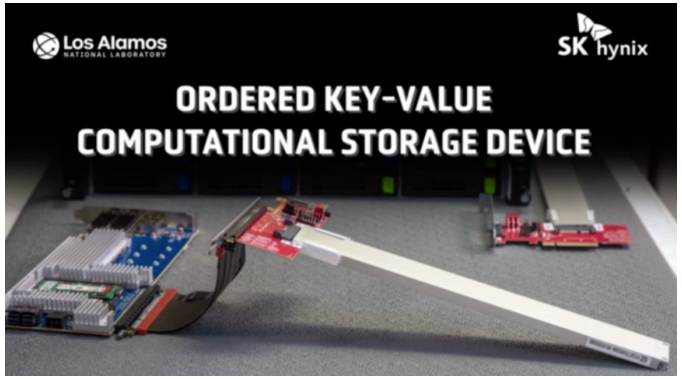


- Not a block interface, a programming interface
- Remote malloc, copy, operate on …

Great partnership

NVIDIA AEON COMPUTING EIDETICOM SK hynix

Data Aware Row And Col Based Analytics Offload Trials



Carnegie Mellon University

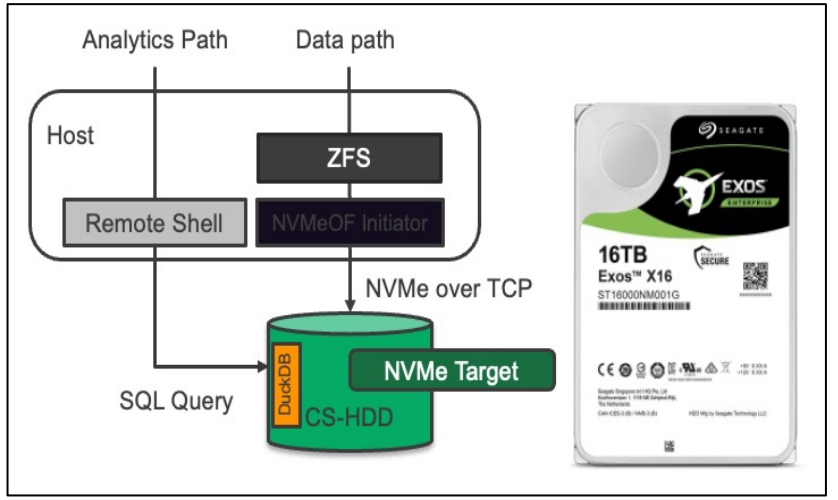
SK hynix

Row (Point methods in Science)

- Early KV-CSD work was hashed based but many KV workloads require “order from chaos”
- Leverage LANL/CMU (DeltaFS (Best Student Paper SC19) trace 1K particles in 1T moving in 1 M cells based on LSM/Rocks
- WHY KV-CS: Performance/Energy win. Few sstables have filter matches but also very few records needed per sstable
- Consumable: (Pretty consumable)
  - User facing Rocks API
  - Extension to SNIA NVME-KV interface
  - Could be accelerator under Rocks

Columnar (Grid methods in Science)

- Leverage
  - Apache ecosystem columnar technology (Parquet/etc.)
  - LANL ZFS knowledge what blocks are part of a parquet file and DuckDB
- Why CSHDD: Performance/Energy win. Many row groups selected (due to AMR) but big variety of number of records needed from those row groups
- Consumable: (Not very consumable - due to inside filesystem implementation)
  - Special knowledge of ZFS file/erasure, something of a layer violation
- Tiny proc/mem performed simple reduction slightly faster than host
- With many drives behind host, scaling beats host by itself (frees host)



Analytics Under Erasure CSHDD

SEAGATE

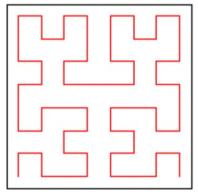
CS data agnostic/data aware learnings

- CSA/CSP's: multi-device/big resource ops (erasure, compression, pipelined functions, etc.)
- CSD's: single-device ops, wins on reductions (requires analytics friendly stripe/erasure)
- Consumption models and broader use cases matter
  - Accelerate under popular server apps: Rocks (rows) / Object Servers (cols) (select on cols/rows)
  - Leveraging 

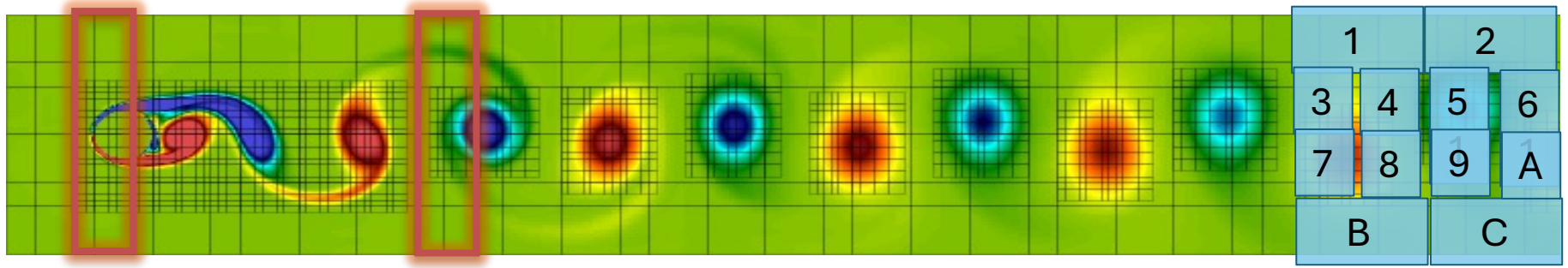
 apache ecosystems seems wise
  - Consumers of large scale data live on one of two basic camps
    - Don't know what you are looking for, want a compact representation (ML training/AI)
    - Know what you are looking for, finding is difficult (due to data size and/or complexity)
      - this is the most obvious place where CS helps and can be implemented in mostly north-south communication patterns
- File systems can be a bit heavy for devices and blocks are useless, perhaps Object is the compromise , with erasure analytics chunks intact on single devices
- At rest compression/erasure seems doable but encryption will be interesting

Why Columnar and why Offload to near Storage?

Multi-dimensional Unstructured Adaptive Meshes (grid methods) use distributed arrays/columns



Single process Hilbert order

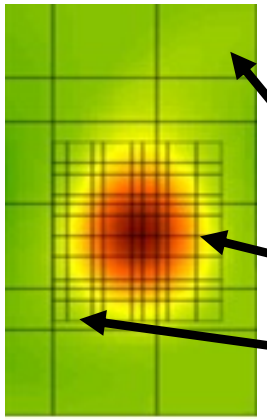


Processes have roughly same number of cells for comp/mem balance but must shuffle cells for AMR

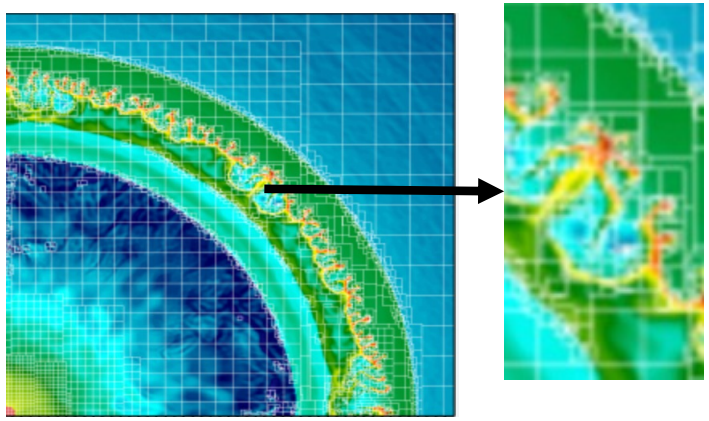
Find outer edge of eddy's (light blue and yellow). Can light weight indexing near storage yield nnnX less data?

- 1 PB file per time step contains all the state (for restart) (think 1 PB) (and thousands of time step)
- Each cell has 10-100 state variables (64float) (temp, pressure, energy, momentum, differentials (for gradients)…)
- In 3D – applications view this as 10-100 distributed arrays (COLUMNS), serialize in Hilbert space filling curve order

Many selected row groups will yield only some rows and most queries will exclude many columns

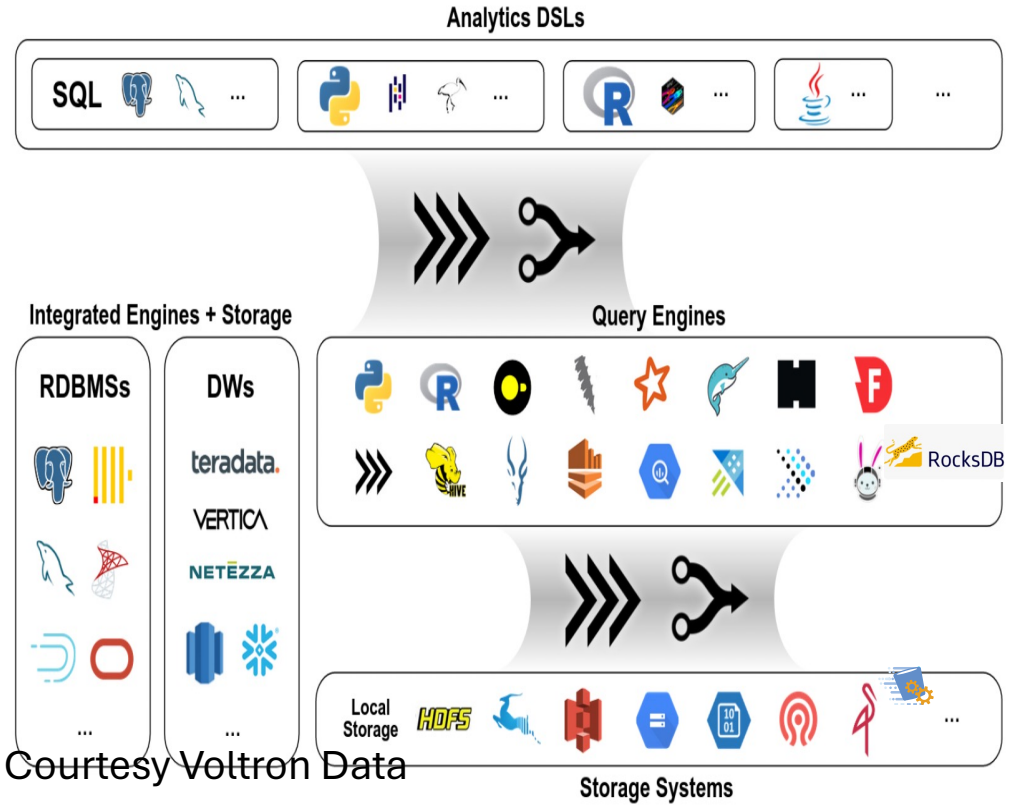


- If you need all row groups, or all rows of groups, or all columns of groups – why offload?
- We don't
  - Reduce trivially by excluding entire row groups
  - Need entire row group (all rows (subset of column)s
  - Need only some rows of the row group

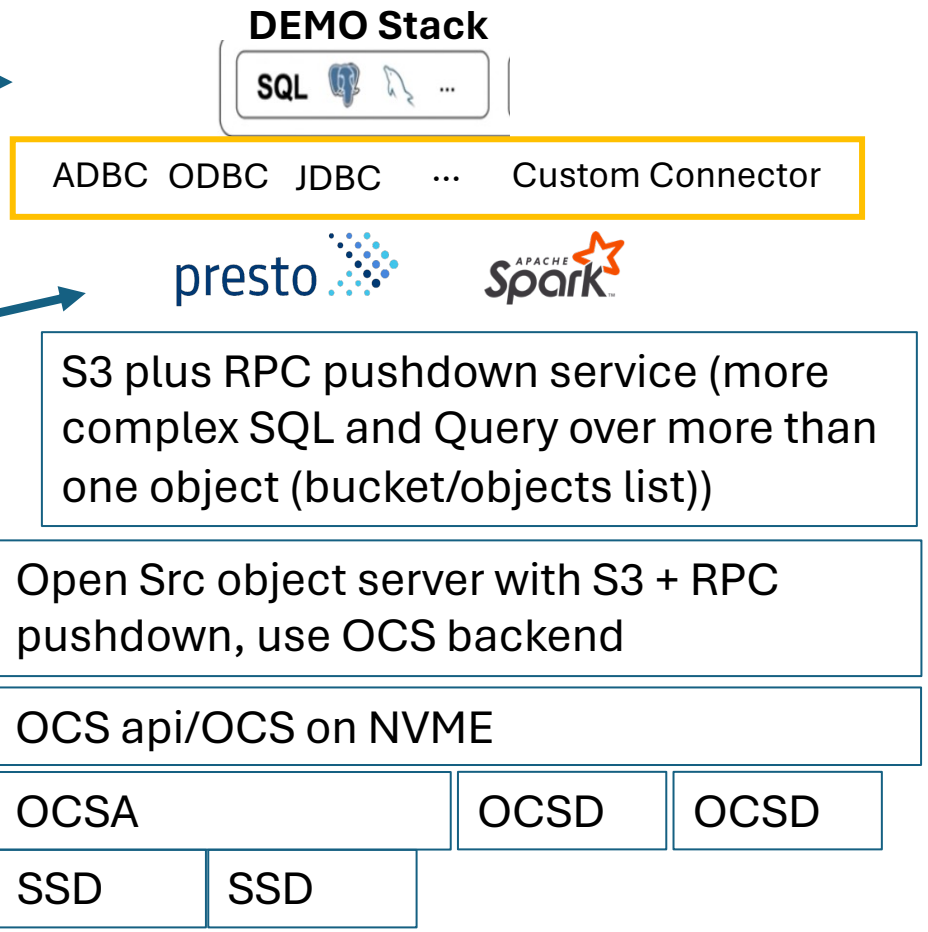


Its never that simple

A Columnar end-to-end demo with Object CS Offloads



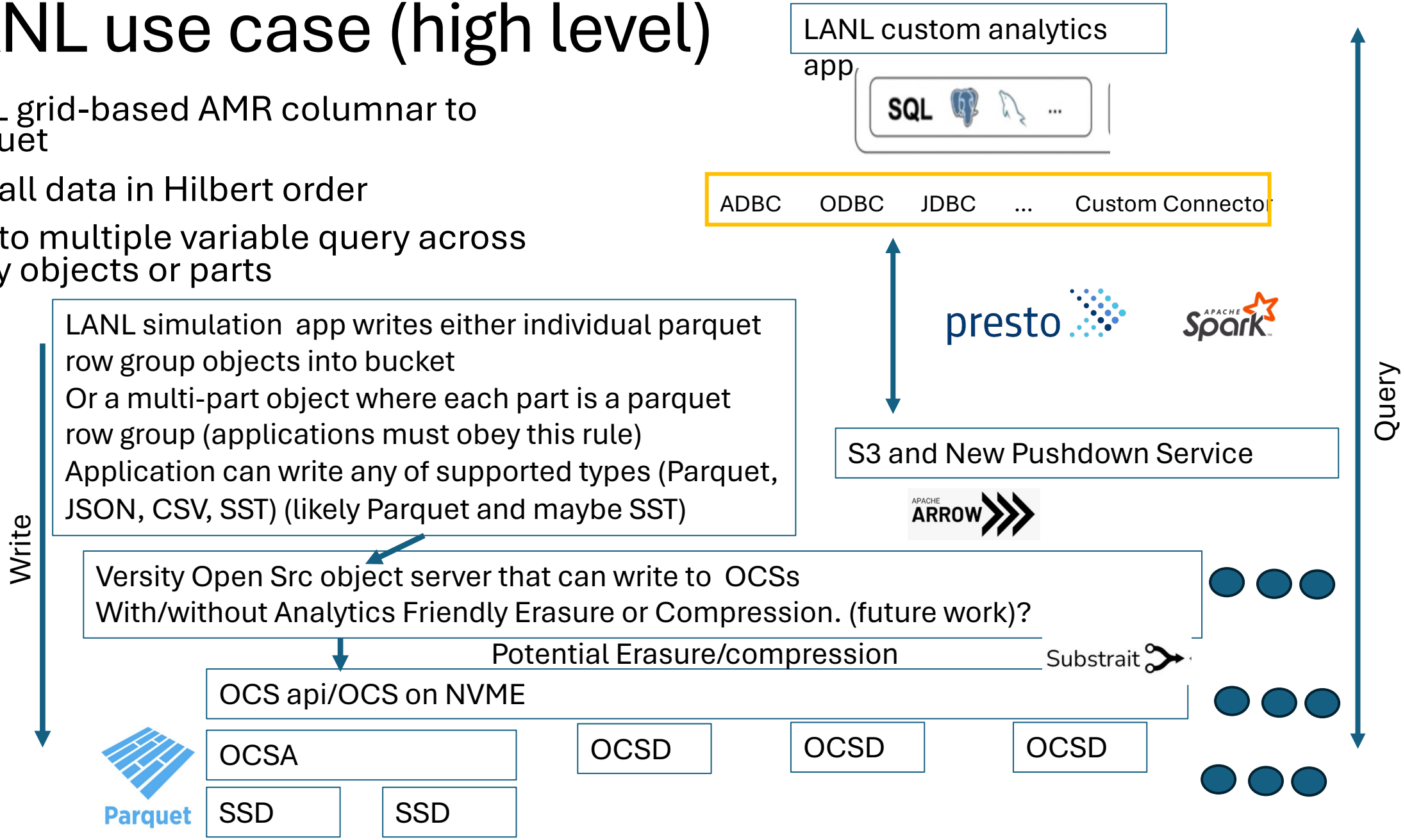
Courtesy Voltron Data



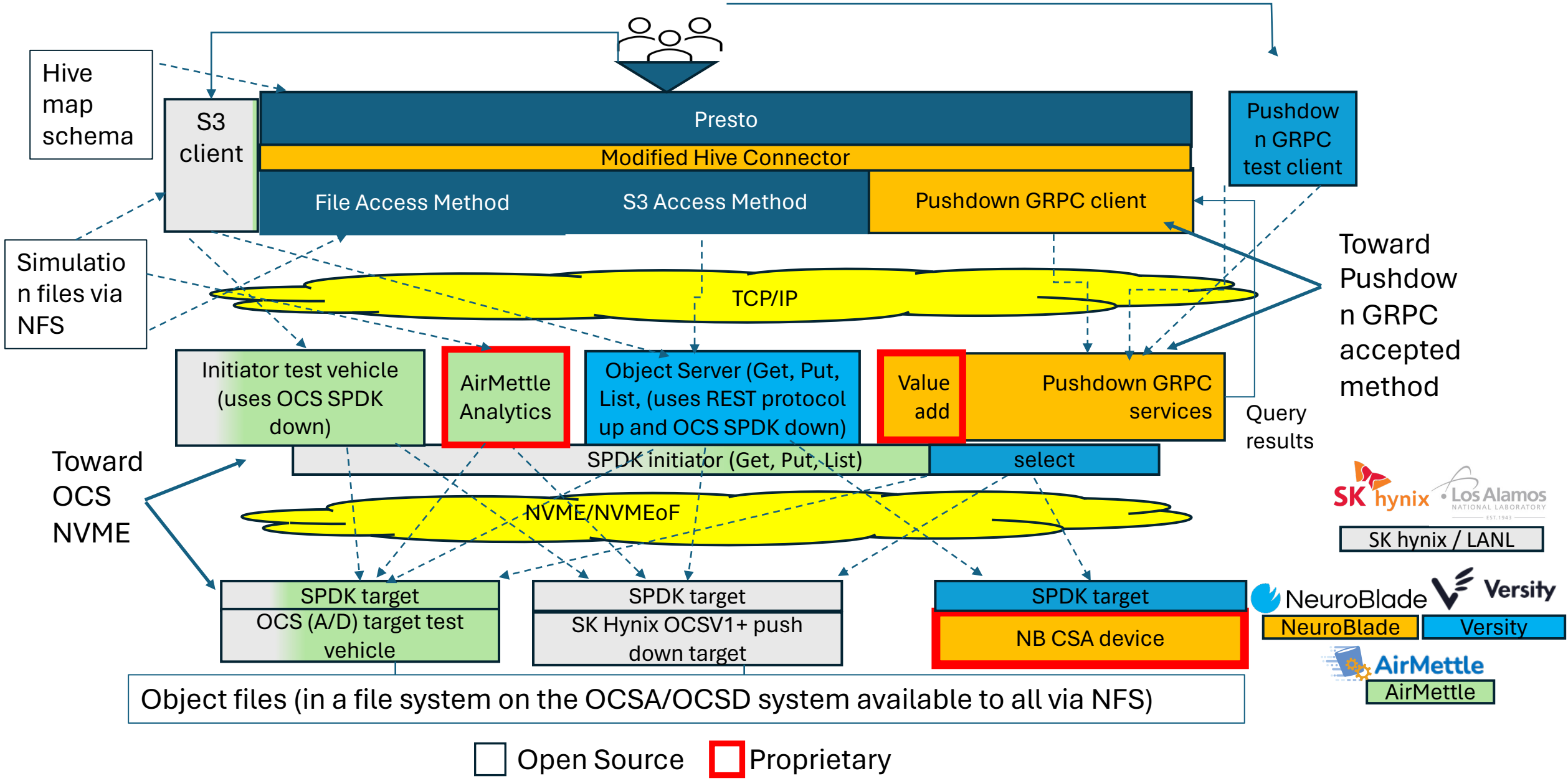
- Object enables same view of “analytic chunk” everywhere (App, obj server, CSA and CSD. (**object not block!**)
- Object Future proofs CS: if CXL becomes how to address smart storage, offload will be based on memory objects, and if file everywhere ever wins, files and objects are close cousins.
- Assists with reality that economics will insist on north/south bw devices

LANL use case (high level)

- LANL grid-based AMR columnar to Parquet
- Overall data in Hilbert order
- One to multiple variable query across many objects or parts



OCS Initiators/Targets Open Ecosystem Demonstration(s)



What if you don’t want/need objects and you want to do this analytics reduction pushdown using files?

- Concept is to use NFS as the mechanism to write the data, read the data, pass the query, and read the results.
- Want to enable use with pNFS to make this all parallel in a totally standards compliant way using NFS/pNFS and Posix security etc.
- Want it to be efficient especially on write and read (indexing mostly done on HPC clients during write not slowing down the data dump time)
- Can be slightly more inefficient on query because the concept is to retrieve many times/orders of magnitude less data than the total data set.

NFS File Query Pushdown

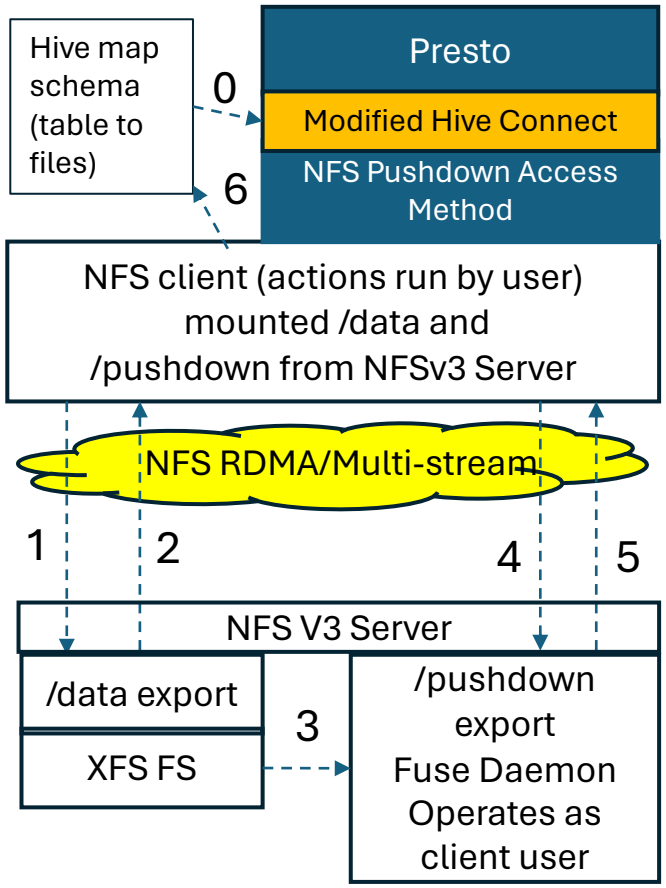
Mkdir /data/d1
1 mkdir to data server
2 response

Write /data/d1/parq1
1 create/write to data server
2 responses

Read /data/d1/parq1
1 read to data server
2 results stream on handle
2 responses

Prep for analytics /data/d1
1 readdir to data server
2 results stream on handle
2 responses
6 load hive table to file map
and schema

Query
0 read table to files from hive
4 open /pushdown/specialfile
4 Write pushdown to handle
(query on files from hive list)
3 Fuse (as user) reads files locally (but
through 5 query results stream on handle
5 responses



Parallel NFS File Query Pushdown

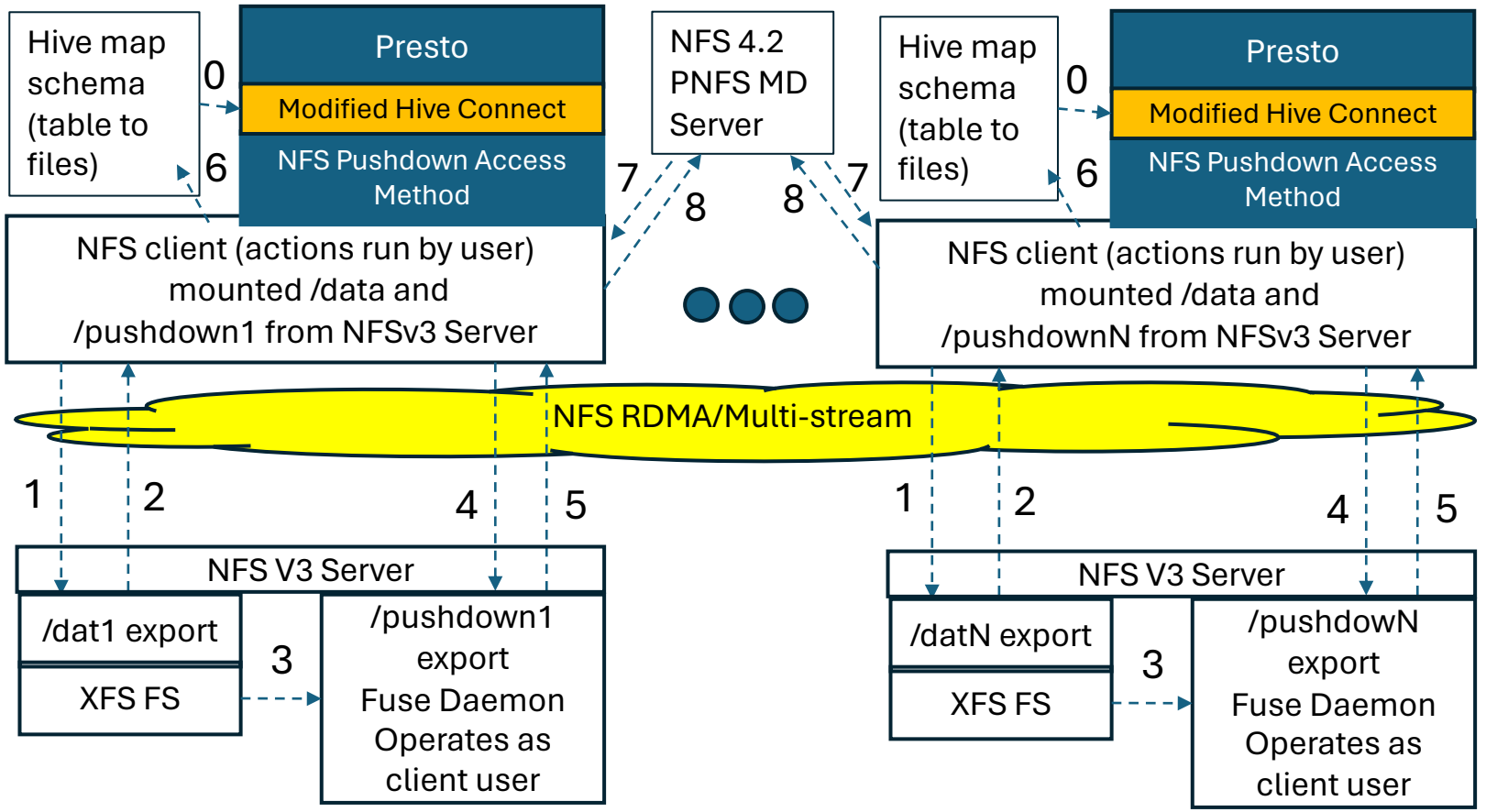
Mkdir /data/d1
7 mkdir to MD server
8 response

Read /data/d1/parq1
7 open to MD server 8 response
1 read to data server (via map)
2 results stream on handle
2 responses

Write /data/d1/parq1 to parqN
7 create to MD server 8 response
1 write data to data server directed by map from md server
(MDS will spread files to different data servers (no in file striping)
2 responses

Prep for analytics /data/d1
7 readdir to MD server
8 results stream on handle
8 responses
6 load hive table to file map and schema (include map location data server 1-N)

Query
0 read table to files from hive
4 open all /pushdownN/specialfile(s)
4 Write pushdown to handle(s)
(query on files from hive list) to each data server where there are files from hive
3 Fuse reads files locally (but through 4.2 local copy read so it an happen as user)
5 query results stream on handle from each
5 responses from each



NFS/PNFS Pushdown seems relatively simple to demo

- NFS4.2 server will give you the map and you can figure out the data server where the file is
- NFS4.2 PNFS server will spread the files out over data servers
- FUSE daemon can be written to accept open of “special files” which interact with fuse to accept pushdown and return answers/results in the normal nfs read/write handle)
- FUSE daemon could have access to the files locally but would use nfs4.2 to read the file with the local bypass so that this can honor user/grp permissions
- Fuse daemon is really not much different than the gprc server code only there is no need for buffering and all that, as its all handled in the nfs file handle but the pushdown and results in arrow is all the same. The SK hynix demo of pushdown code could easily be modified to be the fuse daemon
- Fuse performance might not be perfect but the expectation is that the Fuse daemon will be reading a lot more data from the local files than sending to the client (reduction)

Partnering has been the key to this exploration!

Thanks for your time!