HPC for Science Based Motivations for Computation Near Storage

04/2022

LA-UR-22-23025

Gary Grider

HPC Division

Los Alamos National Laboratory

Eight Decades of Production Weapons Computing to Keep the Nation Safe

Maniac

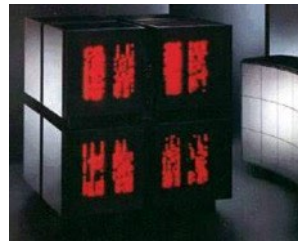
IBM Stretch

CDC

Cray 1

Cray X/Y

CM-2



CM-5

SGI Blue Mountain

DEC/HP Q

IBM Cell Roadrunner

Cray XE Cielo

Cray Intel KNL Trinity

Ising DWave

Cross Roads

CROSS ROADS
An APEX Collaboration

intel

NGP-1

NGP-1

NVIDIA

HPC Scientific Simulation Systems

Trinity – circa 2016

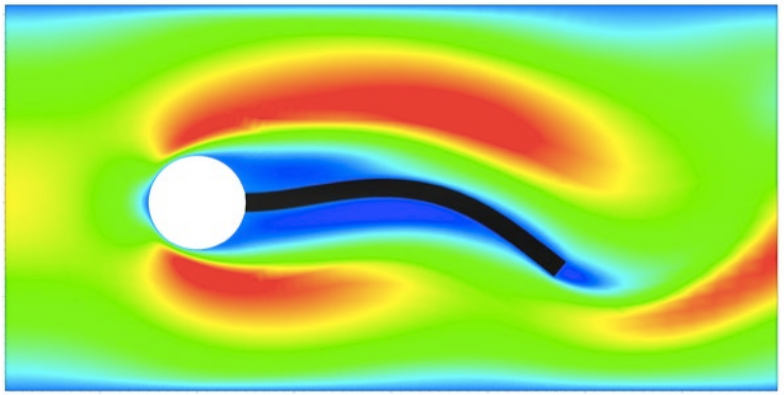
- Haswell and KNL
- 20,000 Nodes
- Few Million Cores
- 2 PByte DRAM
- 4 PByte NAND Burst Buffer ~ 4 TByte/sec
- 100 Pbyte Scratch PMR Disk File system ~1.2 TByte/sec
- 60PByte/year Sitewide Campaign Store ~ 50 GByte/sec
- 60 PByte Sitewide Parallel Tape Archive ~ 3 Gbyte/sec

Circa 2023

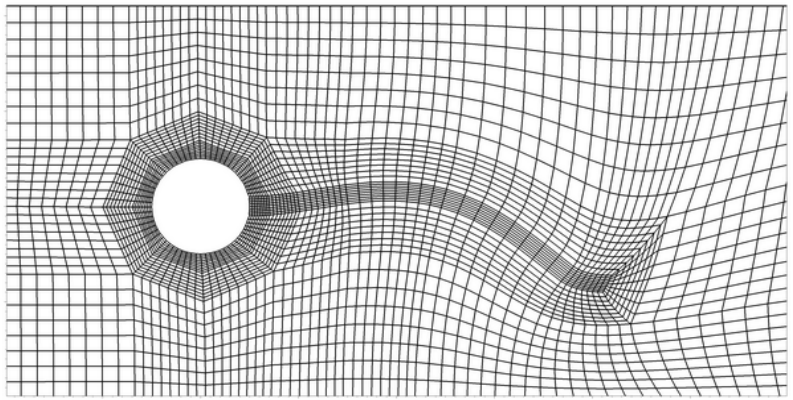
- 10 PB DRAM
- 100 PB Flash
- Half Exabyte spinning disk

I know its not Tier1 sized but at LANL its for one job for several years.
10 PB files and 200 PB Campaigns
For a single user/small user team

And when10PB DRAM is 100X too small HPC Scientific Simulation Data Techniques

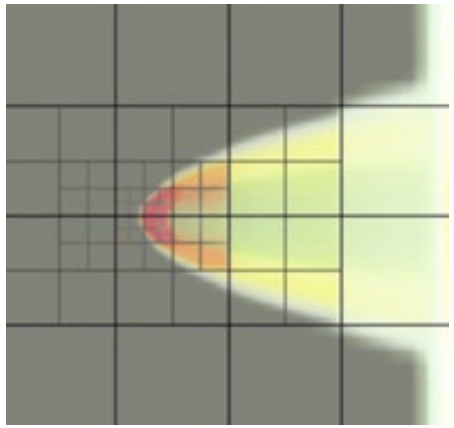


- Wont it compress? Floating point high entropy nope (1.2:1)
- But it does compress computationally, we are only really interested in the fun parts
- We just don't know where those fun parts are a priori and the move around FAST



ALE – Advanced Lagrangian Eulerian

http://web.cs.ucdavis.edu/~ma/VolVis/amr_mesh.jpg



Eulerian AMR

- But there is opportunity in this.
- The capacity is what it is
- But
- The results from query/analysis is a small % of the total data
- And
- While scientific data is files today, that is an artifact of 60 years with disks, its really record oriented data, 5-50 floats per "cell"

Now that you are all Computational Scientists How Does this Motivate Computing near Storage/Network?

- It's a LOT of data
  - Compute near storage
    - Making routine but intensive data management tasks more efficient/cost effective
    - Making routine data analysis tasks more efficient/cost effective

    and
    - Change fundamentally how we do data analysis (no longer influenced by disk storage)
      - Use the fact that the data is really records
      - We may need the network to help too

Making routine but intensive data management tasks more efficient/cost effective

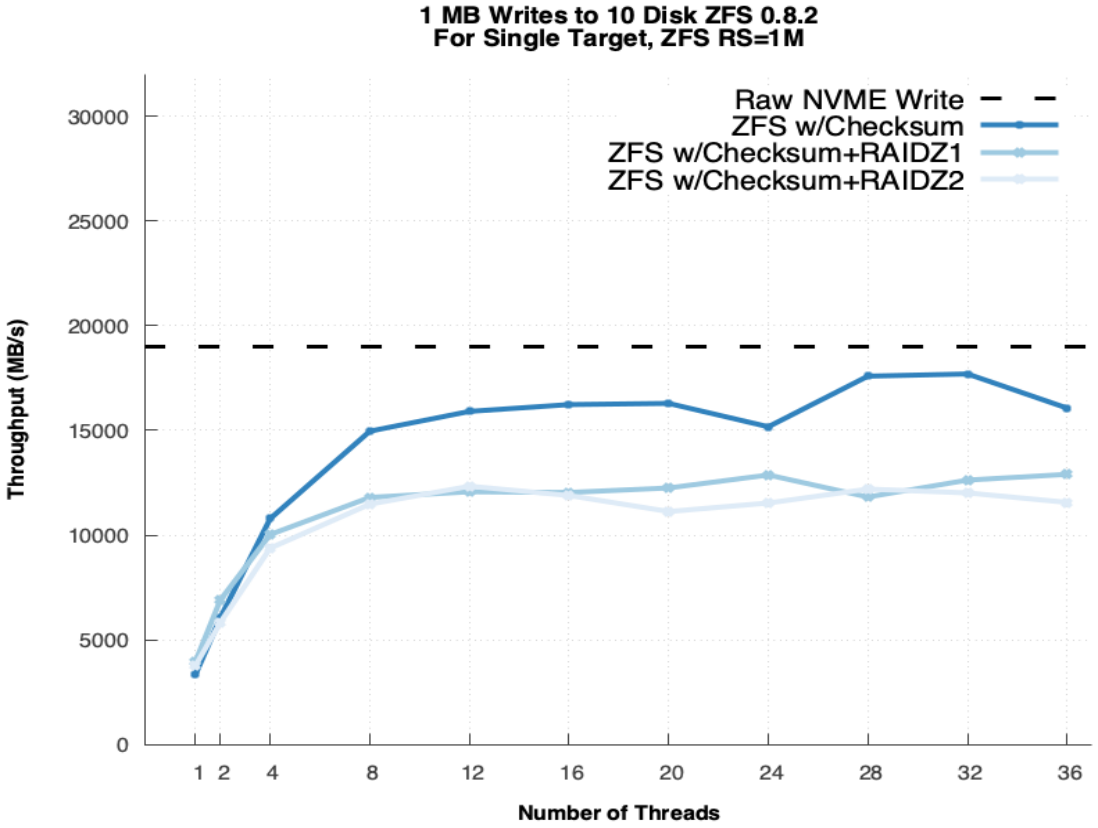
Data Management
Memory Bandwidth Intensive Operations

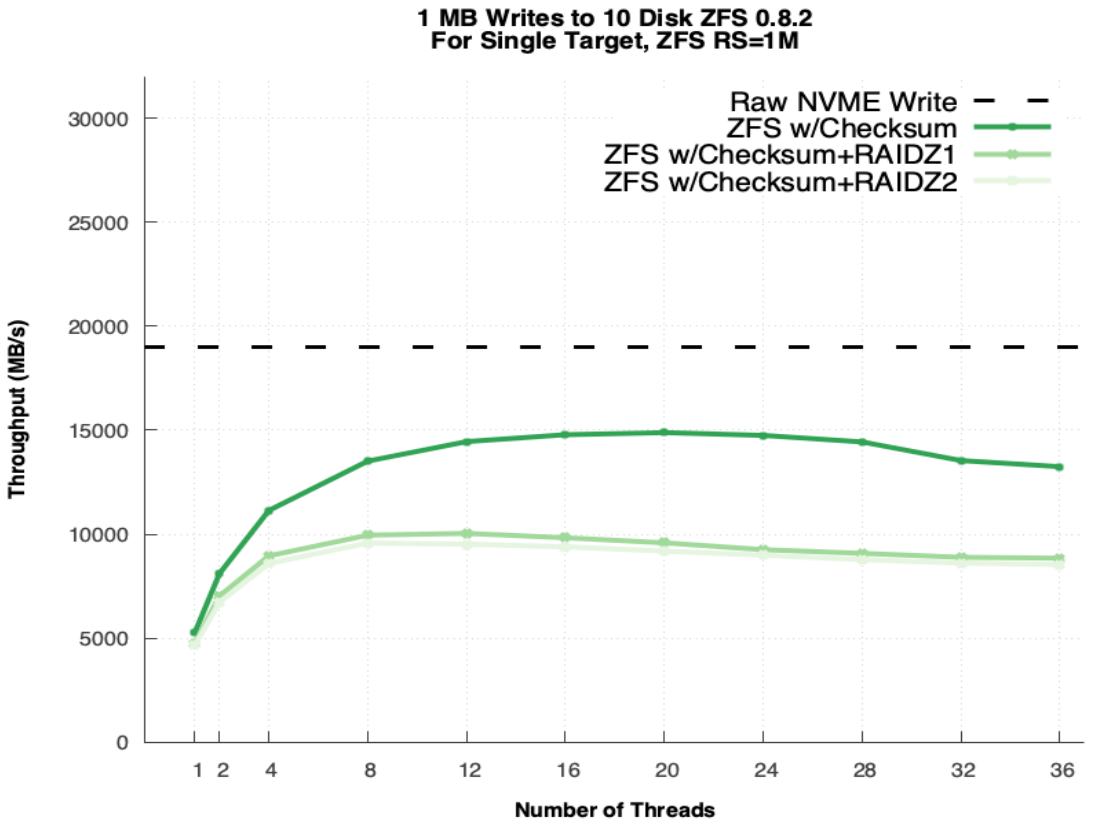
File System Services Offload

- Non-obvious requirements
  - Require transparent data placement for disaster recovery
  - Require parallel file system support
  - Not just Read – in a mixed simulation/AI/ML site - Write-dominant workload for simulation (defensive I/O: write once, read never) still are important
- Computational Storage Benefits/Opportunities
  - Increase compression rates from 1.06:1 -> 1.3:1 for scientific data
  - Enable expensive coding/decoding to protect against correlated failures
  - Achieve higher per-server and per-device bandwidths
  - Lower server costs and quantities
- Overcome poor server memory BW imbalance
- Less expensive file oriented solutions
- Use a commonly used HPC file system to leverage offload (ZFS)
- Break the bonds of block and file being the only interfaces (distributed heterogeneous computing)

Data Agnostic fixed function offloads

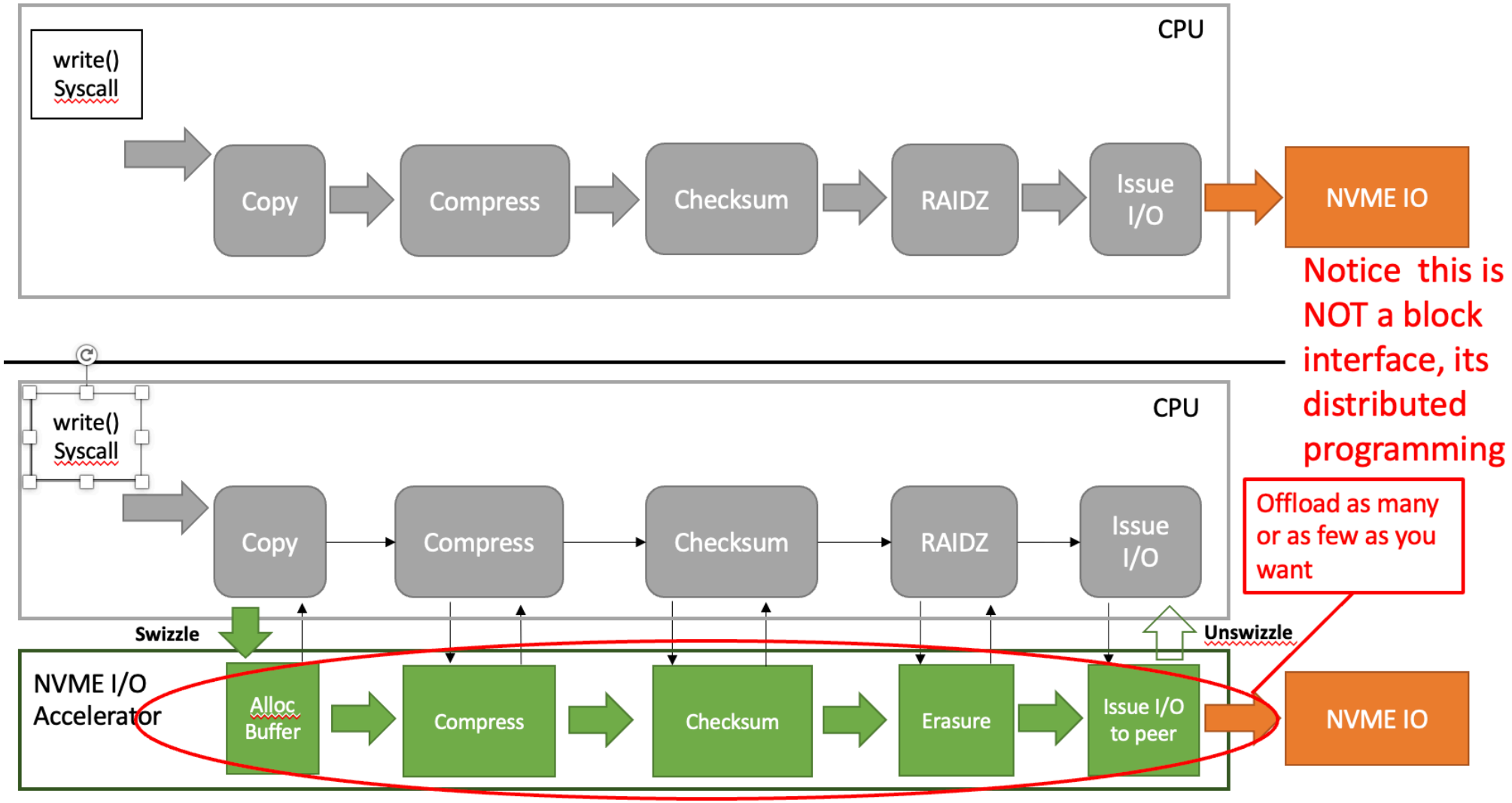
ZFS Checksums, RAIDZ1 (9+1), RAIDZ2 (8+2) Server memory bandwidth is problematic and expensive





- Intel Platinum (Dual Socket)
- AMD EPYC (2nd Gen)

Notional fixed function offloads in ZFS



Accelerated Box of Flash: Powerful Computational Storage for Big Data Projects

Radically new approach to storage acceleration aids data manipulation for research and discovery

MARCH 21, 2022

Recent Press Release

Partnership to demonstrate NVME Computational Storage based ZFS accelerated ABOF (distributed popular host kernel based FS app distributing functions to ABOF smart nic and NVME accelerator)

Eideticom – NoLoad™ software
AEON – enclosure hardware design/engineering
NVIDIA – Bluefield 2 technology
SK hynix – Flash Storage
LANL – ZFS, ZFS offload interface, Linux Kernel offload layer

https://discover.lanl.gov/news/0321-computational-storage

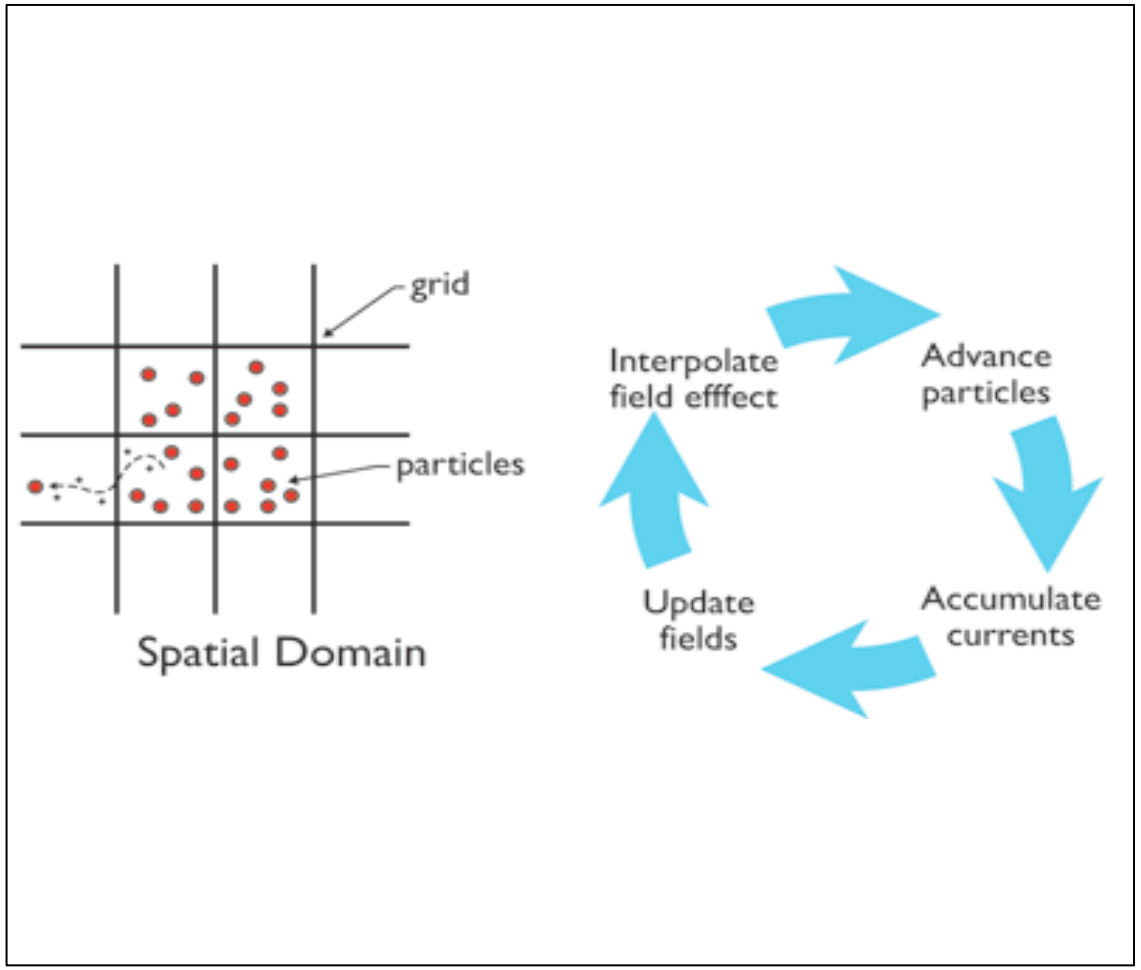
Hopefully will help guide NVME Computational Emerging Standards.

Making routine but intensive data analysis tasks more efficient/cost effective

Change fundamentally how we do data analysis
(no longer influenced by disk storage)

Single Dimensional Indexing - Brief VPIC Overview

- Particle-in-cell MPI code (scales to ~100K processes)
  - Fixed mesh range assigned to each process
  - 32 – 64 Byte particles
  - Particles move frequently between 10's of thousands of processes
  - Million particles per node (Trillion particle in target simulation)
  - Interesting particles identified at ***simulation end (say 1000 interesting particles)***

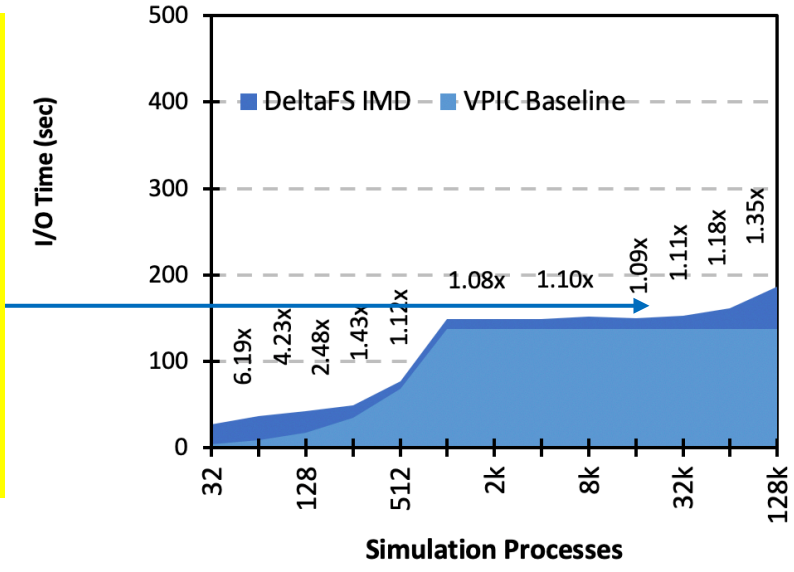


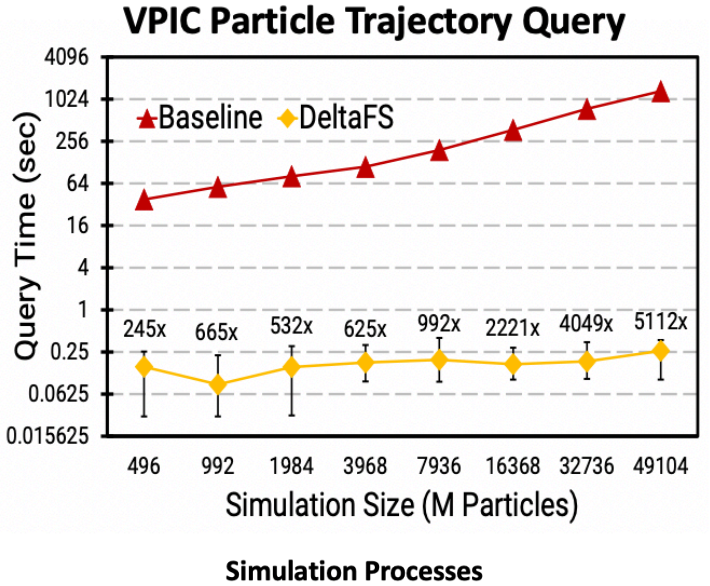
DeltaFS - Near-device Indexing and Analytics

- Non-obvious requirements
  - Simulations run under intense memory pressure (app may use 90%)
  - In-situ indexing runs into scaling limitations
- Computational Storage Benefits/Opportunities
  - Speedups for post-hoc analysis (1000x speedup demonstrated)
  - Less reliance on massive compute tier as a large merge sort space

Get efficiency and lower time to solution (1000X)

Add a little time indexing on the way out and get 1000X on analysis step (the indexing must scale and be efficient (perfect offload opportunity)





Application thought it was writing/reading from 1 file per trillion particles, really writing records to massive parallel distributed KVS!
8 Billion Particle Ops/Sec.
(yes Billion)

Collaboration of CMU, LANL, ANL, HDF Group

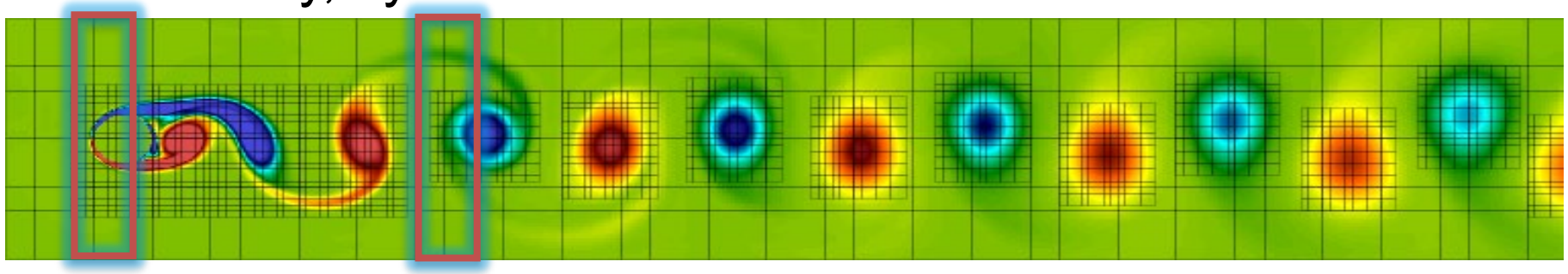
(papers at PDSW 15, PDSW 17, SC19 (Best Student Paper)

1000X Speed Up by Leveraging Records and Not Files

- Isn't 8 Billion Metadata ops/sec good enough?
- Well maybe, but this is a low dimensional highly structured fixed mesh and the sharding was simple to shard (particle ID)
- What about higher dimensional problems, unstructured and ever changing meshes?
- What if one of the indexing dimensions is pressure or energy and we don't know the key distribution, how do we come up with a sharding strategy?
- In the Particle problem, "now that I know where the "interesting particles were" what was going on around those interesting particles

Indexing for Unstructured Meshes

- How do you store/represent an AMR mesh?
  - In memory, dynamic tree and nested list structures are common

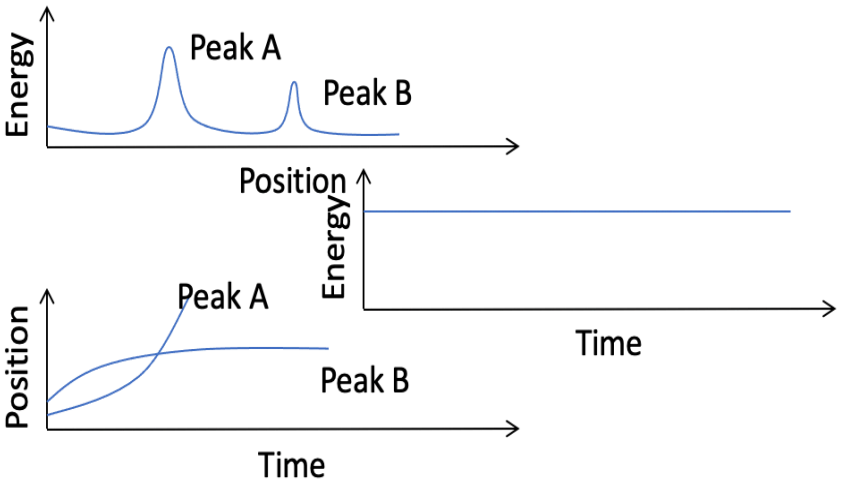


How many rows are in each of these columns?

(For that matter, how many columns are in each of these columns?!)

How do you store this kind of time series data in a usable form?

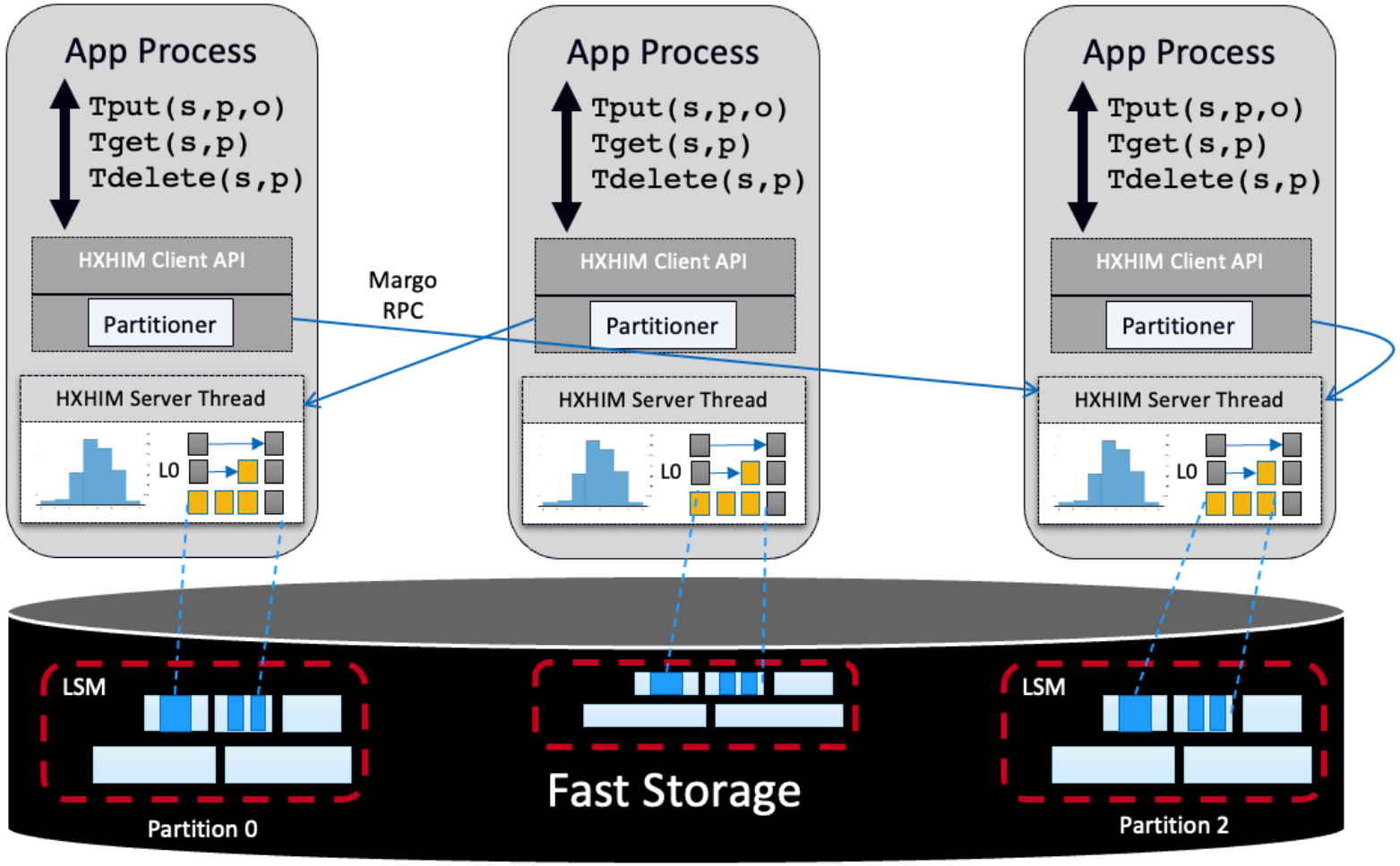
- **Key Value Store?**
- **Key-value exposes the structure in the data**
- **Key-value allows fine-grained data annotation**
- **Need to add some HPC research to make efficient for HPC platforms**



Imagine if we could track a shockwave going through a material by only retrieving 1/1000th of the data!

Multi-Dimensional Motivation - MDHIM/XDHIM

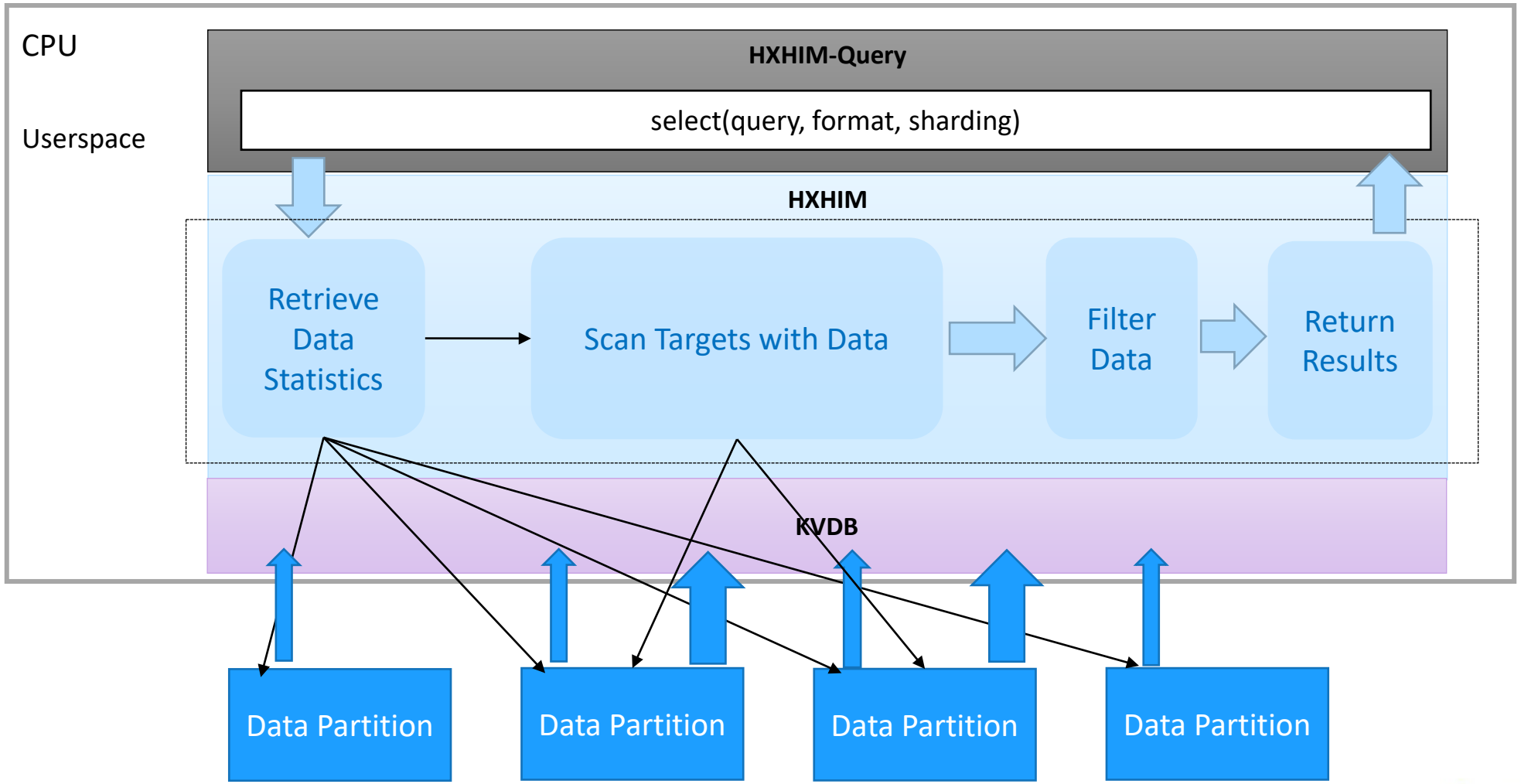
An application linkable parallel KVS Framework KVS is linked in the application, bulk put/get ops, uses key range sharding and server side stored procedures, X Dimensional Sharded Index



Distributed key sharding when we don't know the key distribution and it is changing all the time

- Remember the golden rules
  - **We can't use a bunch of compute node memory**
  - **We can't increase the write time by very much**
- Can you discover a key distribution of the data with only compute application hints?
  - In simulation, each process is responsible for a small fraction of the physical space, typically a volume
  - Every time step, processes must communicate surface/boundary conditions to its neighbors.
    - The network gets to see 10-20% of the data every time step
    - The part of the mesh the network sees changes all the time due to AMR and the network sees more of the "fun parts" due to AMR
  - For Checkpoint/RecoveryEvery dump of state, the network gets to see all the data
  - Can the network keep histograms of important fields to help guess at the distribution? (Compute in the network!)

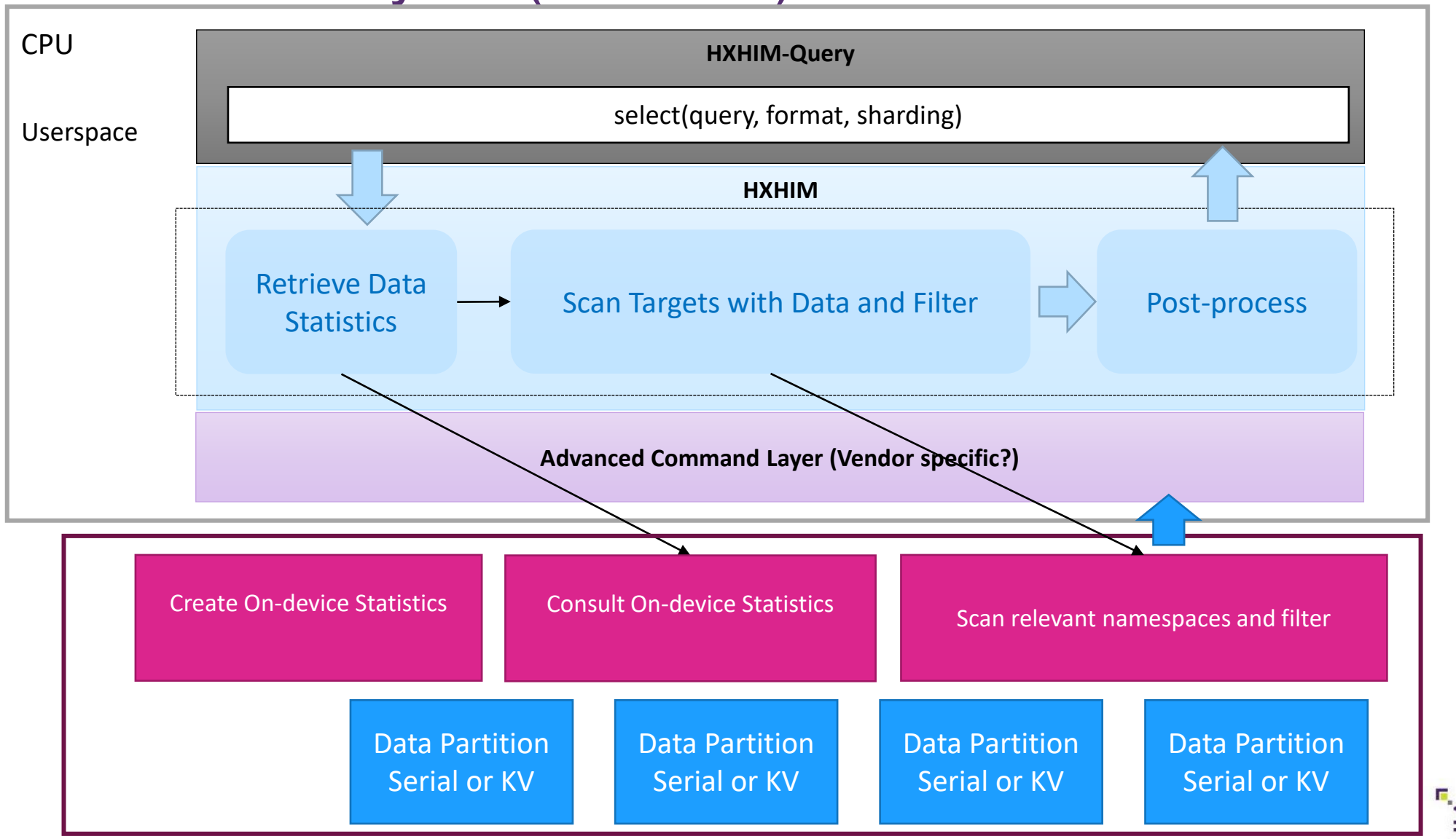
Host Based User-space Analytics Application Software Layers



This is the read path only.

Statistics must be generated during output phase

Offloaded to Computational Storage Analytics Application Software Layers (Notional)

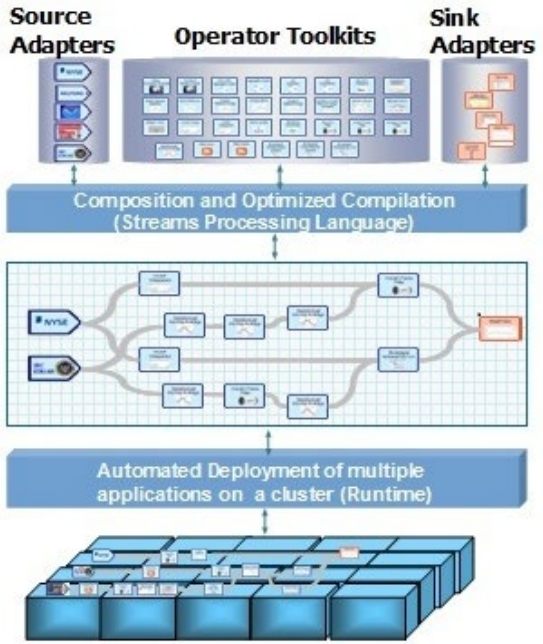


This is the read path only.

Statistics can be generated at the devices

How do we think about using computational storage/offloading functions/programming?

- Offload intensive data management functions, minimal changes to storage stack, no changes to apps
- Middleware like Hxhim (distributed parallel KVS framework (ordered))
- Emerging standards? NVME Computational Storage
- Different storage paradigm than block!
- Learn lessons from streams programming paradigm?
  - System S (DOD/IBM)
  - Netsketch (CMU)



Thanks for your time!

Ultra-Scale Systems Research Center

Los Alamos NATIONAL LABORATORY EST. 1943

NNSA

HIGH PERFORMANCE COMPUTING

The Efficient Mission Centric Computing Consortium

Please take a moment to rate this session.

Your feedback is important to us.

