UC SANTA CRUZ | Baskin Engineering

iris hep

SkyhookDM: Embedding Apache Arrow inside Storage Systems

Jayjeet Chakraborty, Ivo Jimenez, Sebastiaan Rodriguez Alvarez, Alexandru Uta, Jeff LeFevre, Carlos Maltzahn

CROSS

cross.ucsc.edu

Agenda

- Broader Problem and Solution
- Our approach
- Background
- Design and Architecture
- Evaluations

The Broader Problem

- With high speed storage devices like NVMe SSDs serving upto 3GB/s and networks supporting 25-100Gb/s, CPU is the new bottleneck
- Client-side computation of data and reading from efficient storage formats like Parquet, ORC exhausts the clients CPUs
- Leads to severely hampered scalability, latency, and throughput

Computational Storage as a Solution

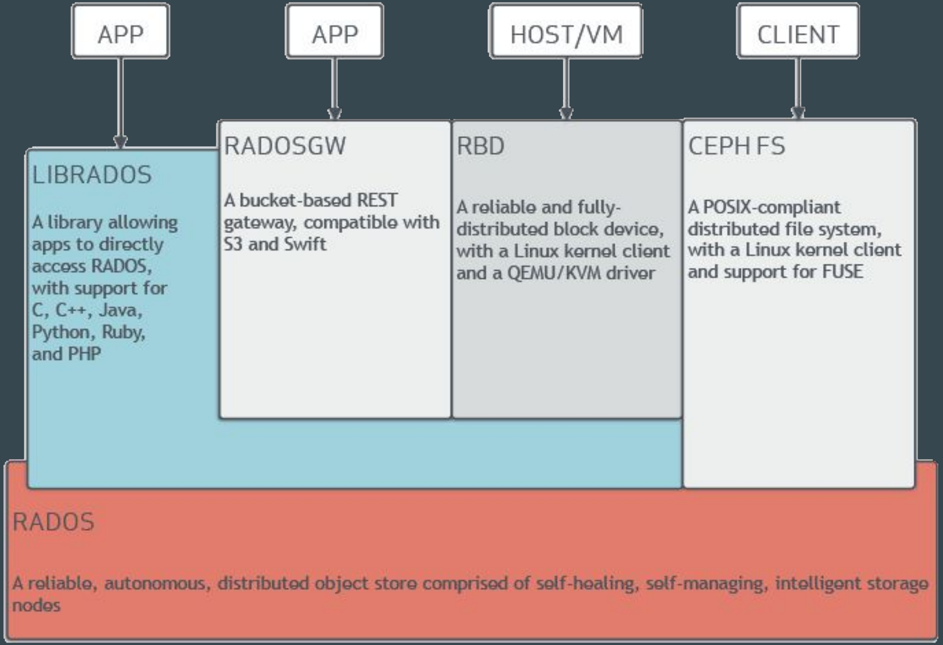
- Offload as much compute as possible to the storage layer
- Use the idle CPUs of the storage nodes to perform data filtering, decoding, decompression
- Accelerated queries due to reduced data movement and increased scalability

How is our approach (Skyhook) different ?

- Most computational storage systems require hardware support like SSDs embedded with FPGAs for offloading compute
    - No extra hardware, use storage nodes CPU, start offloading instantly
    - Does it work ? What are the challenges ?
- Enable compute offloading in existing programmable storage systems without code changes
    - Easily test out computational storage with your existing storage infrastructure
- Embed data access libraries directly into the storage system
    - Offload metadata management to the storage layer

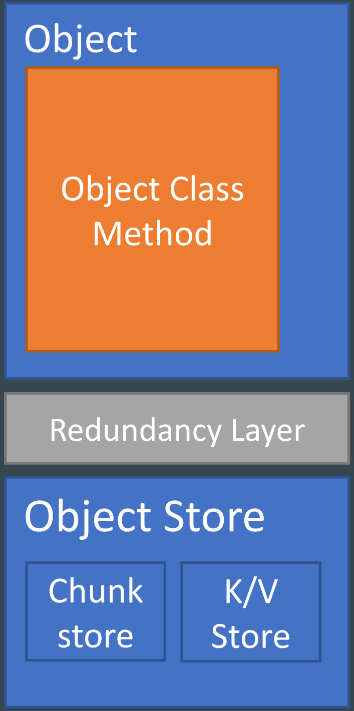
Ceph

- Provides 3 types of storage interface: File, Object, Block
- No central point of failure. Uses CRUSH maps that contains Object - OSD mapping
- Extensible Object storage layer via the Ceph Object Classes SDK


APP
APP
HOST/VM
CLIENT
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOSGW
A bucket-based REST gateway, compatible with S3 and Swift
RBD
A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver
CEPH FS
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE
RADOS
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

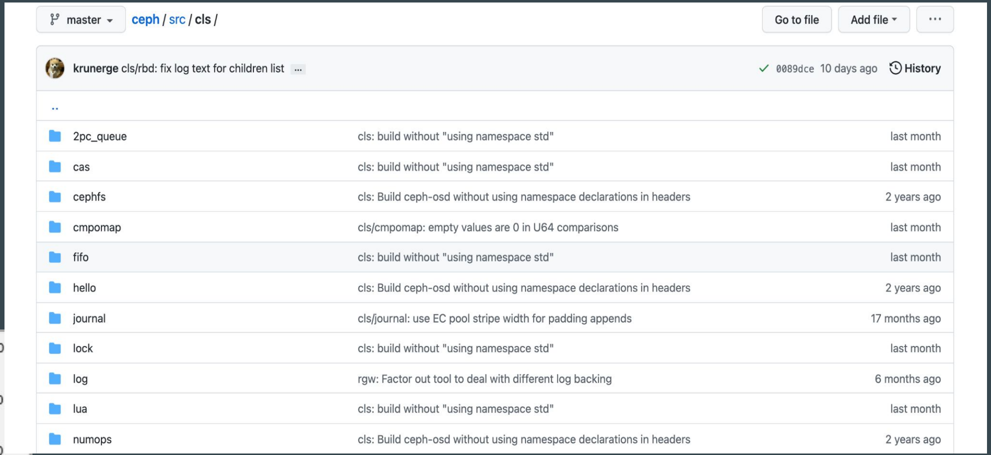
Object Class Mechanism

- Utilizing Ceph’s object class mechanism (“cls”)
  - Plugin-based object storage extension mechanism
  - Helps extend the Object store I/O path with User-defined functions
- Used by several Ceph internals
  - CephFS, RGW (Rados Gateway), RBD (Rados Block Device)



Object Class Mechanism

Object classes in Ceph

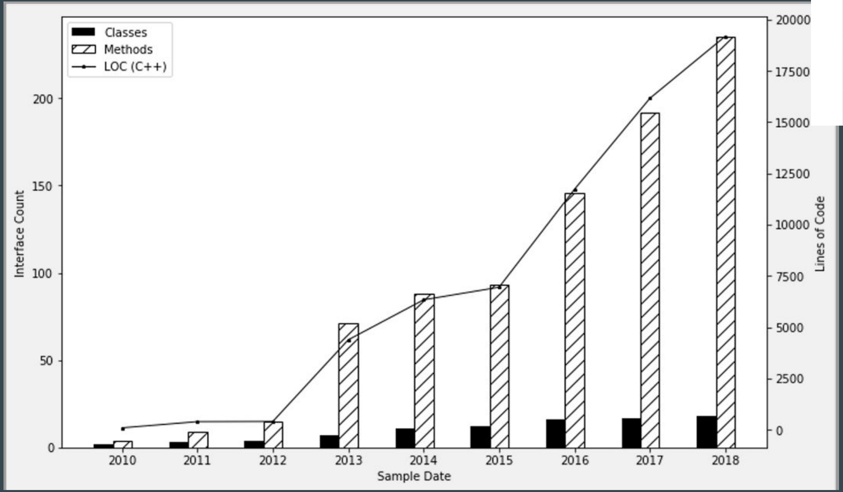


master | ceph / src / cls / | Go to file | Add file

krunerge cls/rbd: fix log text for children list … ✓ 0089dce 10 days ago History

..

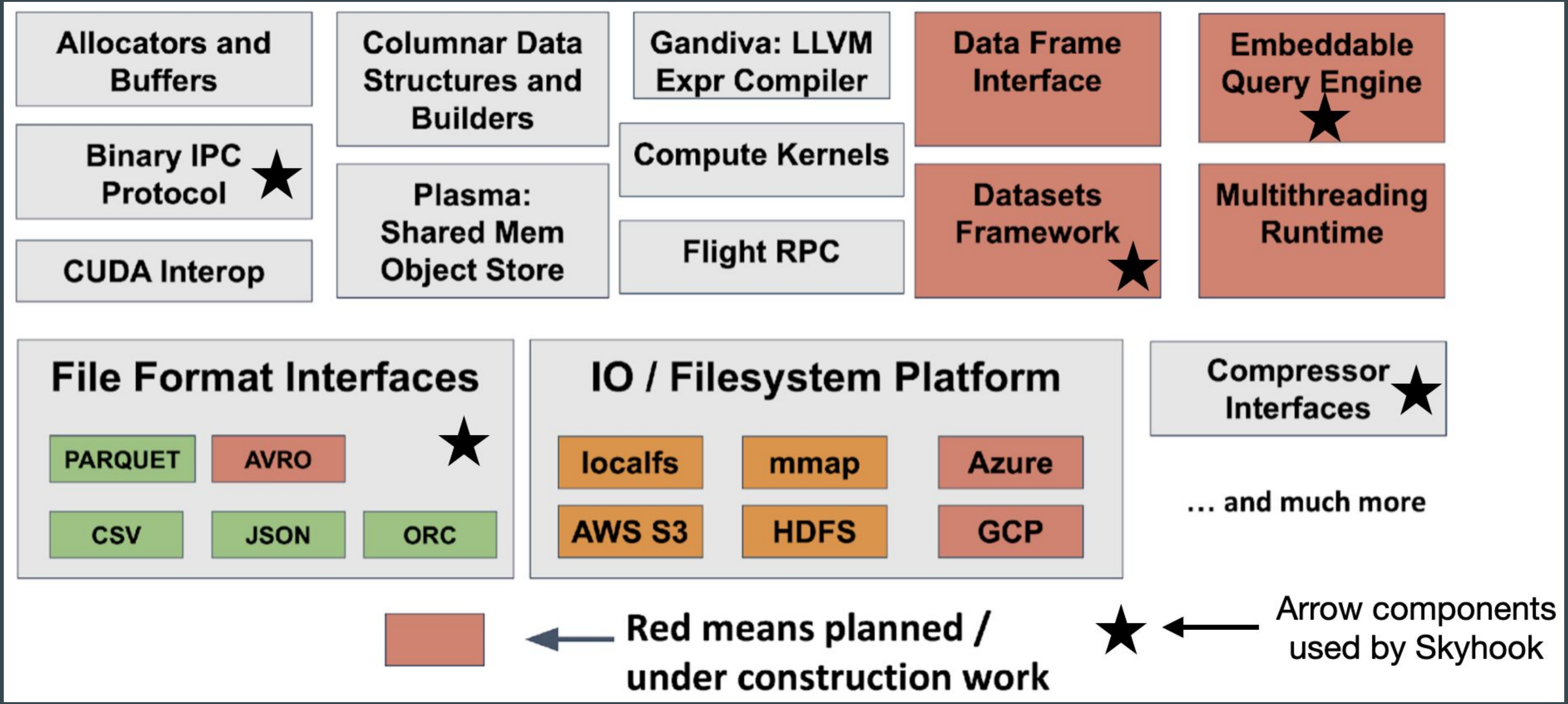
2pc_queue	cls: build without "using namespace std"	last month
cas	cls: build without "using namespace std"	last month
cephfs	cls: Build ceph-osd without using namespace declarations in headers	2 years ago
cmpomap	cls/cmpomap: empty values are 0 in U64 comparisons	last month
fifo	cls: build without "using namespace std"	last month
hello	cls: Build ceph-osd without using namespace declarations in headers	2 years ago
journal	cls/journal: use EC pool stripe width for padding appends	17 months ago
lock	cls: build without "using namespace std"	last month
log	rgw: Factor out tool to deal with different log backing	6 months ago
lua	cls: build without "using namespace std"	last month
numops	cls: Build ceph-osd without using namespace declarations in headers	2 years ago



Growth of object classes in Ceph

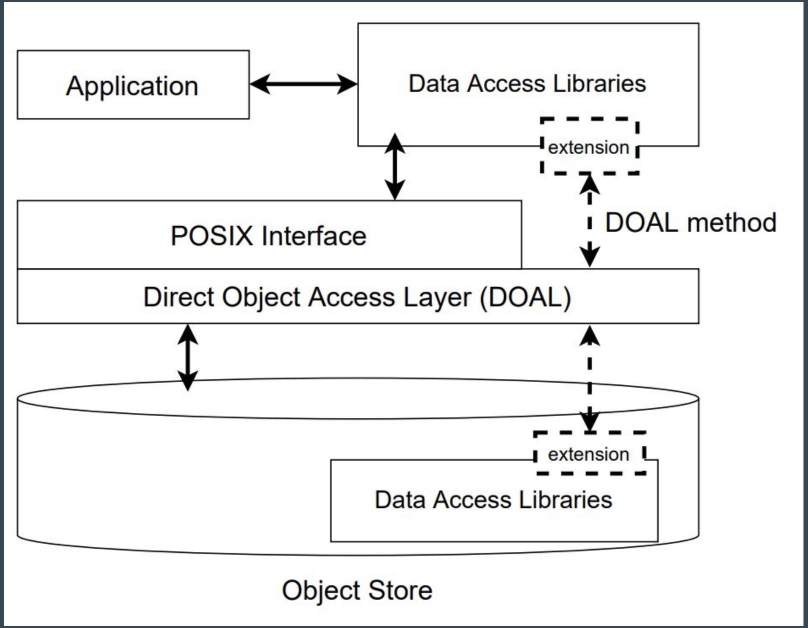
Apache Arrow

Rich collection of pluggable components for building data processing systems



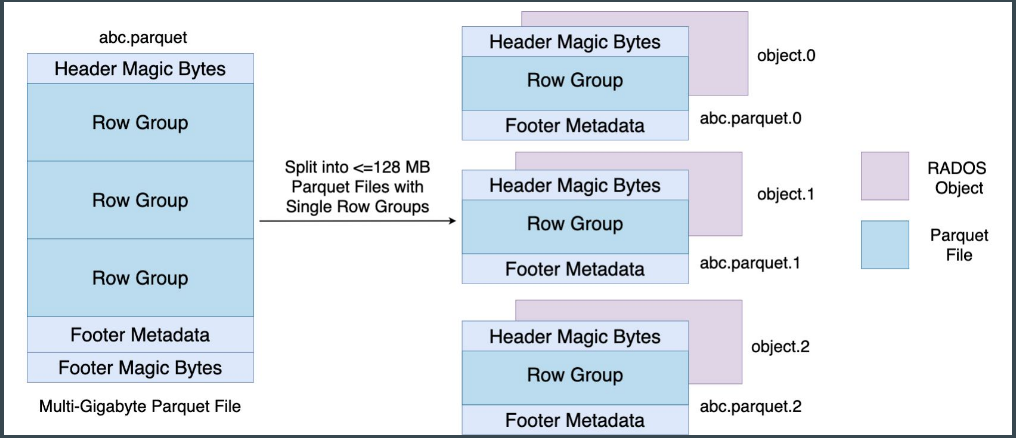
Design Paradigm

- Extend client and storage layers of programmable storage systems with data access libraries
- Embed a FS shim inside storage nodes to have file-like view over objects
- Make object class extensions directly available to the clients without having to change the FS

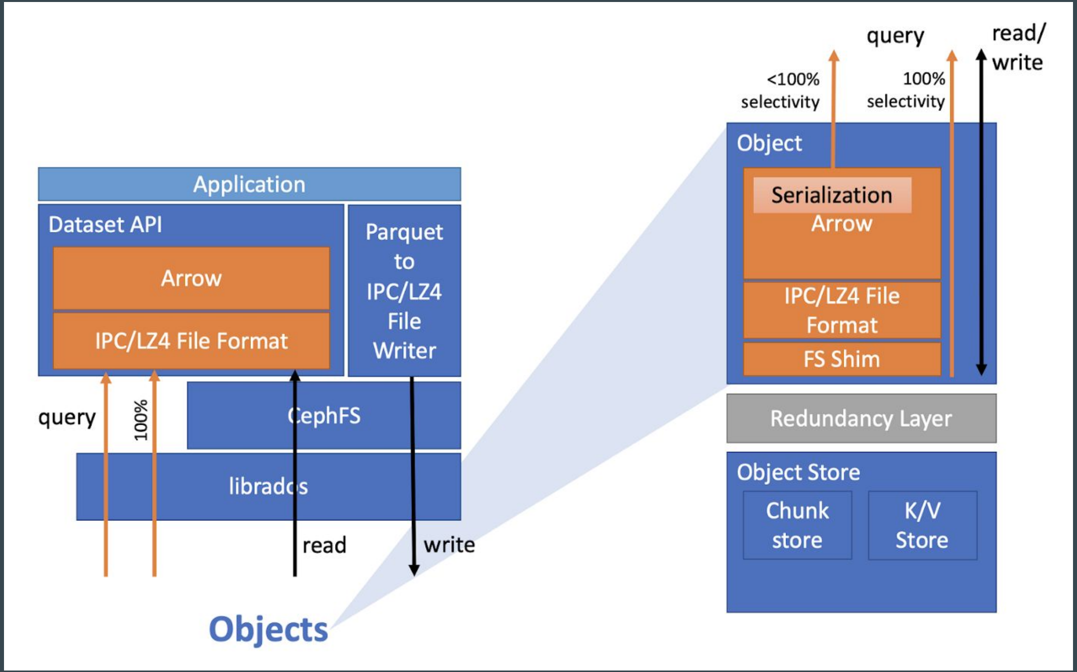


File Layout

- Max object size allowed by Ceph is 128MB
    - In this work, we use 64MB files as we found out it to provide the best performance
- Files larger than 64MB are split into ~64MB files
    - Each partition goes into a single RADOS object
- This `1:1` mapping between files and objects facilitates the direct translation from filenames to Object IDs



Architecture



Using Skyhook from Arrow

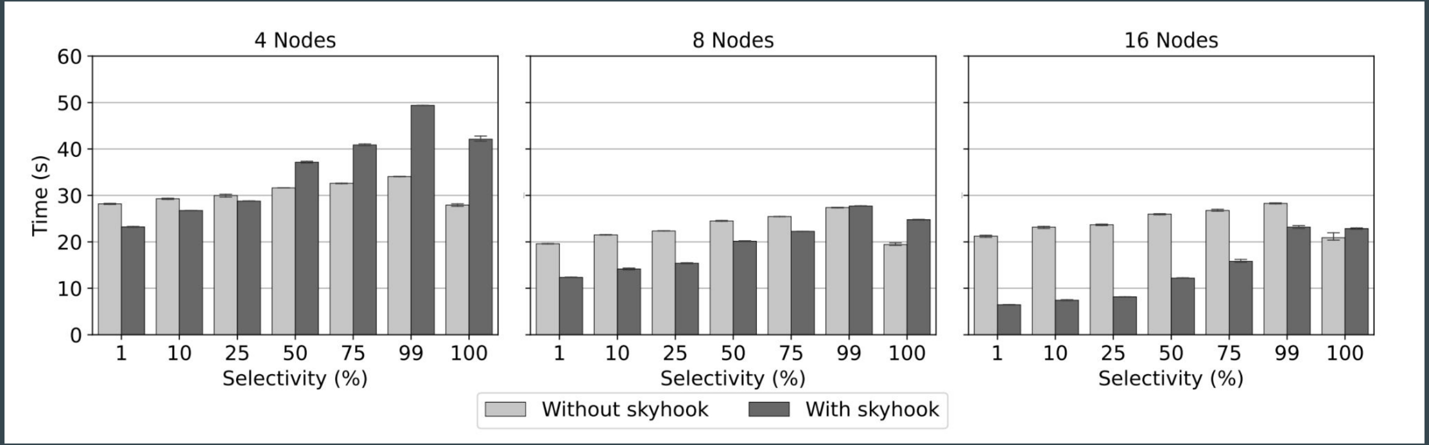
```
# Reading from Parquet
import pyarrow.dataset as ds
format_ = "parquet"
dataset = ds.dataset(
    "/dataset", format=format_
)
dataset.to_table()

# Reading from Parquet using Skyhook
import pyarrow.dataset as ds
format_ = ds.SkyhookFileFormat(
    "parquet", "/ceph.conf"
)
dataset = ds.dataset(
    "/dataset", format=format_
)
dataset.to_table()
```

Skyhook supports file formats that are supported by Arrow out-of-the-box

- Parquet, CSV, JSON, Feather

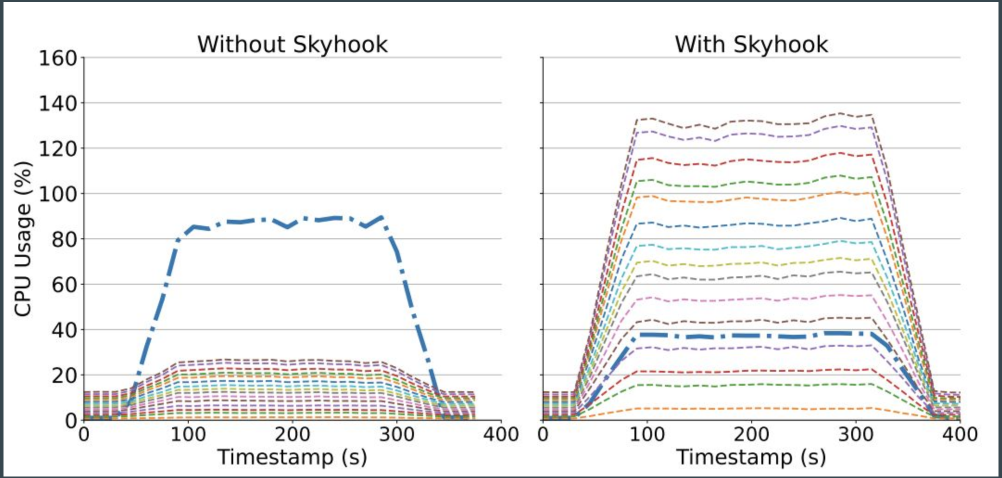
Evaluations

Query Latency



- Skyhook scales with cluster size but Parquet does not
- When Skyhook cannot benefit from scale out, serialization overhead dominates
- 100% selectivity of Skyhook results in a unnecessary packing/unpacking of Parquet files inside the storage nodes
    - We can simply detect 100% selectivity queries and avoid offloading to Skyhook

CPU usage

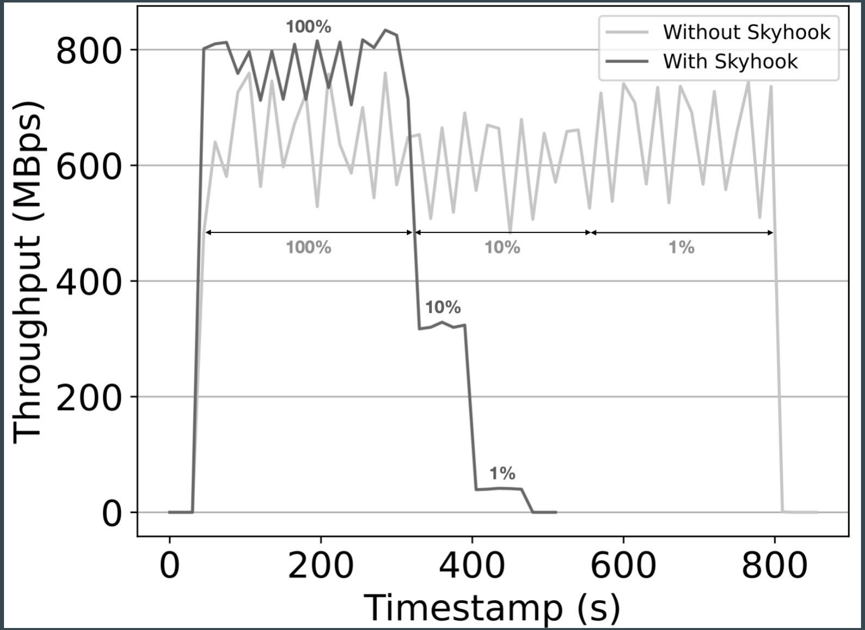


Client: Thick blue line
Storage: Other lines

- Skyhook provides compute access to more CPU resources improving scalability and performance
- Decompression of LZ4 compressed batches uses some CPU on the client side in Skyhook
- The simplicity of offloading using storage nodes CPUs trades off total CPU usage

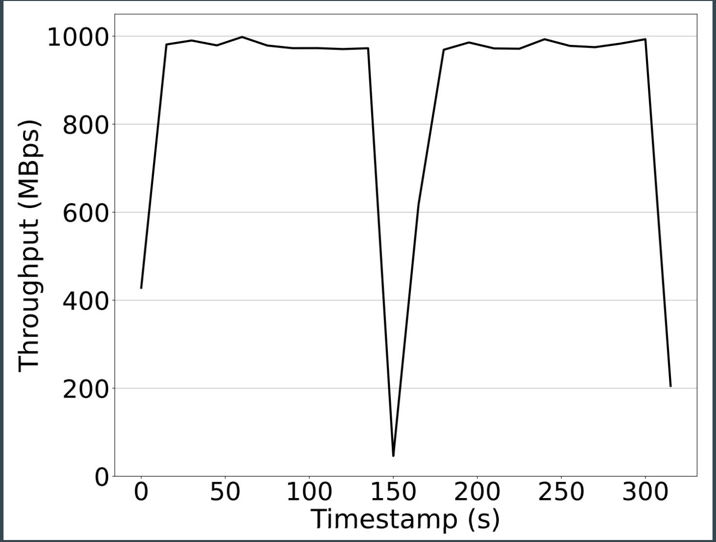
Network Traffic

- Skyhook prevents unnecessary bandwidth wastage making room for other applications
- Queries with 10% and 1% row selectivity are much faster in Skyhook than without Skyhook
- Bandwidth usage during 100% selectivity of Skyhook is little more than without skyhook as Skyhook transfers data in slightly larger LZ4-compressed Arrow IPC format

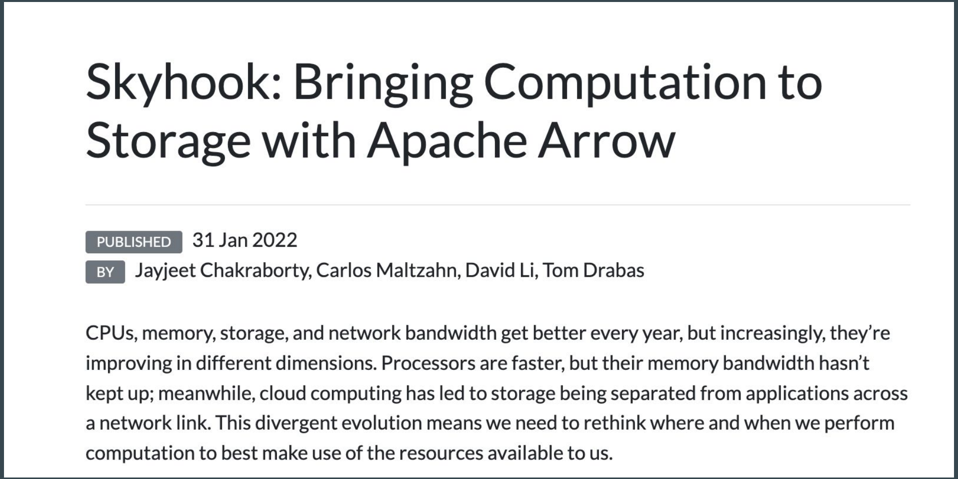


Crash Recovery

- Skyhook queries are fault tolerant due to the integration of compute with storage
- Object method calls are sent to object names regardless of their physical location
- When a server crashes, method class are automatically restarted on another server with a redundant copy of that object



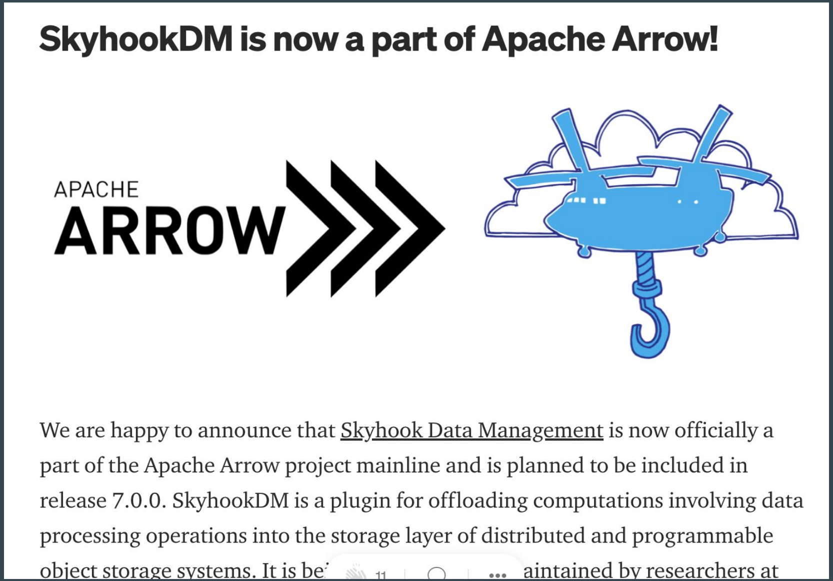
Skyhook upstreamed in Apache Arrow !



Skyhook: Bringing Computation to Storage with Apache Arrow

PUBLISHED 31 Jan 2022
BY Jayjeet Chakraborty, Carlos Maltzahn, David Li, Tom Drabas

CPUs, memory, storage, and network bandwidth get better every year, but increasingly, they're improving in different dimensions. Processors are faster, but their memory bandwidth hasn't kept up; meanwhile, cloud computing has led to storage being separated from applications across a network link. This divergent evolution means we need to rethink where and when we perform computation to best make use of the resources available to us.



SkyhookDM is now a part of Apache Arrow!

We are happy to announce that Skyhook Data Management is now officially a part of the Apache Arrow project mainline and is planned to be included in release 7.0.0. SkyhookDM is a plugin for offloading computations involving data processing operations into the storage layer of distributed and programmable object storage systems. It is be aintained by researchers at

Future Work

- Support RDMA to transfer result Arrow Record Batches from storage to the client
  - This is to avoid the memory-to-wire serialization overhead
- Move to embedding the Arrow Streaming Compute Engine instead of the Arrow Dataset API to support offloading more complex compute operations
  - Requires having a streaming interface in Ceph Object Class SDK
- Use Gandiva to accelerate Arrow query processing inside the storage layer
  - Leverage SIMD processing capabilities of modern processors

Thank You

jayjeetc@ucsc.edu

https://iris-hep.org/projects/skyhookdm.html