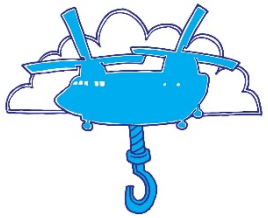
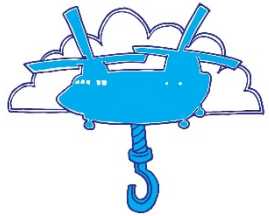
SDC 20
BY Developers FOR Developers
Storage Developer Conference
September 22-23, 2020

SkyhookDM: Storage and Management of Tabular Data in Ceph

Jeff LeFevre, Carlos Maltzahn
Center for Research in Open Source Software
University of California, Santa Cruz



Skyhook Data Management



- Open source software, LGPL 2.1 License
- Built on Ceph distributed object storage
- Computational storage for tabular data
- Extensible, scalable, generic

ceph

CROSS | CENTER FOR RESEARCH IN OPEN SOURCE SOFTWARE

APACHE ARROW

Google FlatBuffers: Memory Efficient Serialization Library

Approach

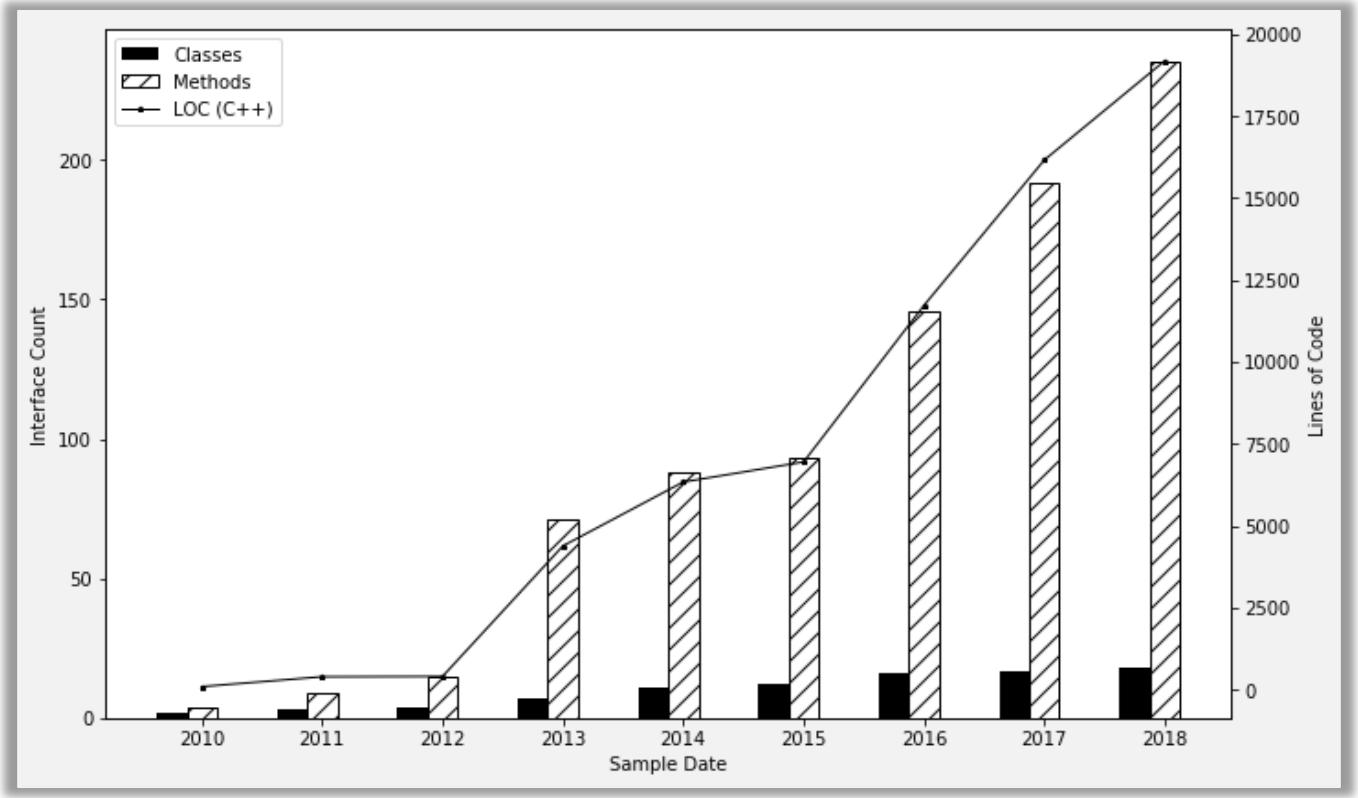
- *Programmable* storage (See programmability.us)
  - Combine, expose, or extend existing storage services toward new functionality
- In-storage execution of data management tasks
  - Embed external libraries in storage
- Dynamically offload computation to storage servers
- Dynamically reorganize physical data configuration
- ***Reduce CPU and Network resources for client apps***

But How?

User-defined extensions to Ceph

- Utilize Ceph's existing *object class* mechanism ('cls')
  - Extensible framework for objects
  - ceph/src/cls
- Methods executed directly by objects
  - Shared libraries available on all OSDs
- Utilized by Ceph internals
  - CephFS, rgw, rbd, others…

CLS growth in Ceph



Growth of object classes and methods in Ceph mainline

Snapshot of '*cls*' Classes in Ceph

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

ivancich Merge pull request #31058 from cbodley/wip-rgw-skip-bilog ... 86df8b9 4 days ago History

..

2pc_queue	rgw/notifications: persistency - cleanup stale reservations	19 days ago
cas	cls/cas: replace bool get() with void get()	3 months ago
cephfs	cls: Build ceph-osd without using namespace declarations in headers	5 months ago
cmpomap	cls/cmpomap: add cls module for CMPXATTR-like functionality in omap	5 months ago
hello	cls: Build ceph-osd without using namespace declarations in headers	5 months ago
journal	cls/journal: use EC pool stripe width for padding appends	5 months ago
lock	cls,rados,rbd,mds,common: Avoid name collision with Windows headers	2 months ago
log	cls: Build ceph-osd without using namespace declarations in headers	5 months ago
lua	librados: add symbol versioning to the C++ API	2 years ago
numops	cls: Build ceph-osd without using namespace declarations in headers	5 months ago
otp	cls: Build ceph-osd without using namespace declarations in headers	5 months ago
queue	cls/rgw_gc: Clearing off urgent data in bufferlist, before	3 months ago
rbd	librbd: track in-progress migration aborting operation	22 days ago
refcount	cls: Build ceph-osd without using namespace declarations in headers	5 months ago
rgw	Merge pull request #31058 from cbodley/wip-rgw-skip-bilog	4 days ago
rgw_gc	cls/rgw_gc: Fixing carriage returns in log statement.	3 months ago
sdk	cls/sdk: Update cls_sdk.cc to work without using namespace	2 years ago
timeindex	cls: Build ceph-osd without using namespace declarations in headers	5 months ago
user	rgw: introduce safe user-reset-stats	2 months ago
version	cls: Build ceph-osd without using namespace declarations in headers	5 months ago

Read/Write Interface

- Ceph objects can access their local data via two interfaces within ‘cls’
  1. Chunkstore – raw device access
  2. KVstore – Local instance of RocksDB on OSD (omap interface)
- For us –
  1. Map tabular data to a device and offset
  2. Consider storing tabular data and/or metadata

CLS Interface Examples

- Some functions available within a cls method
  - // read/write
    - cls_cxx_read(ctx, off, len, buf)
    - cls_cxx_write(ctx off, len, buf)
    - cls_cxx_replace(ctx, off, len, buf)
  - // metadata
    - cls_cxx_setaxxtr(ctx, name, buf)
    - cls_cxx_stat(ctx, size_t, NULL)
  - // utilize local RocksDB instance on the OSD
    - cls_cxx_map_setvals(ctx, map<str, buf>)
    - cls_cxx_map_getval(ctx, key, buf)
    - cls_cxx_map_getval***s***(ctx, keystart, nkeys, map<str,buf>, more)

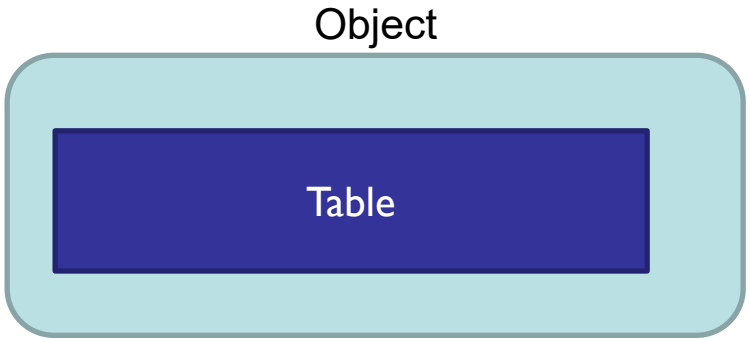
CLS Interface Examples

- Some functions available within a cls method
  - // read/write
    - cls_cxx_read(ctx, off, len, buf)
    - cls_cxx_write(ctx off, len, buf)
    - cls_cxx_replace(ctx, off, len, buf)
  - // metadata
    - cls_cxx_setaxxtr(ctx, name, buf)
    - cls_cxx_stat(ctx, size_t, NULL)
  - // utilize local RocksDB instance on the OSD
    - cls_cxx_map_setvals(ctx, map<str, buf>)
    - cls_cxx_map_getval(ctx, key, buf)
    - cls_cxx_map_getval***s***(ctx, keystart, nkeys, map<str,buf>, more)

Notice the use of offset/length allows partial read/write of objects

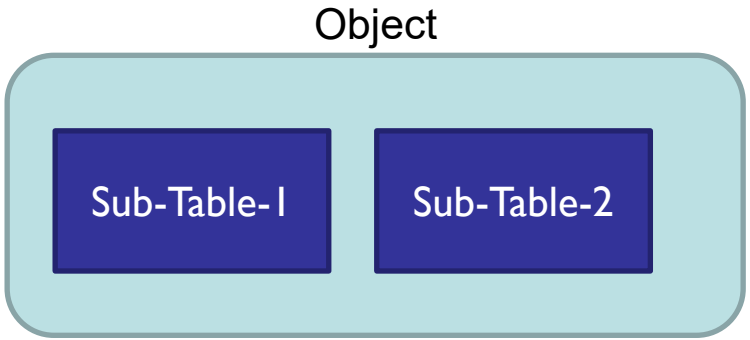
Why use partial read/write of an object?

- Enables great flexibility for physical data layout within each object



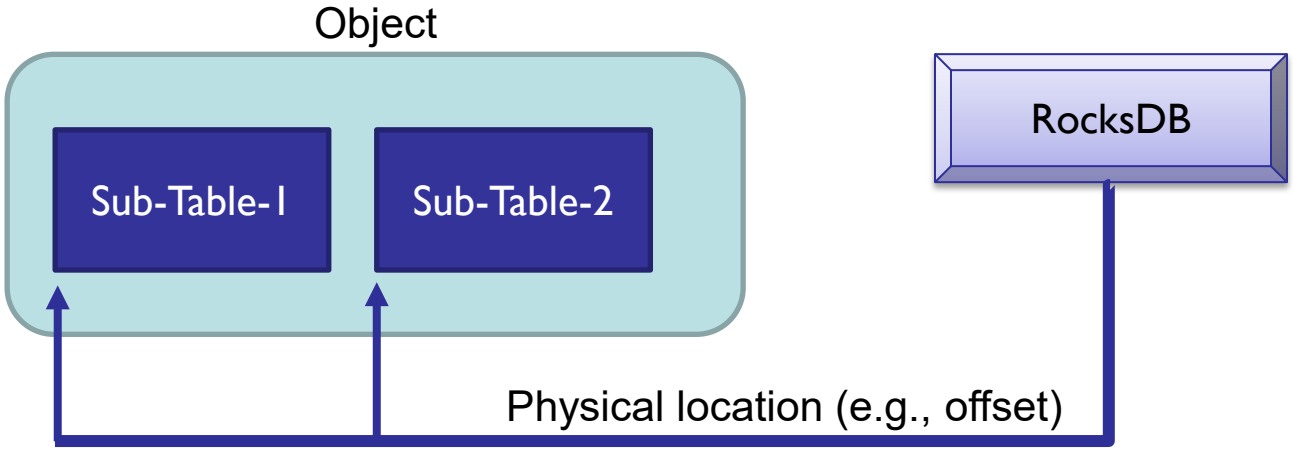
Why use partial read/write of an object?

- Enables great flexibility for physical data layout within each object



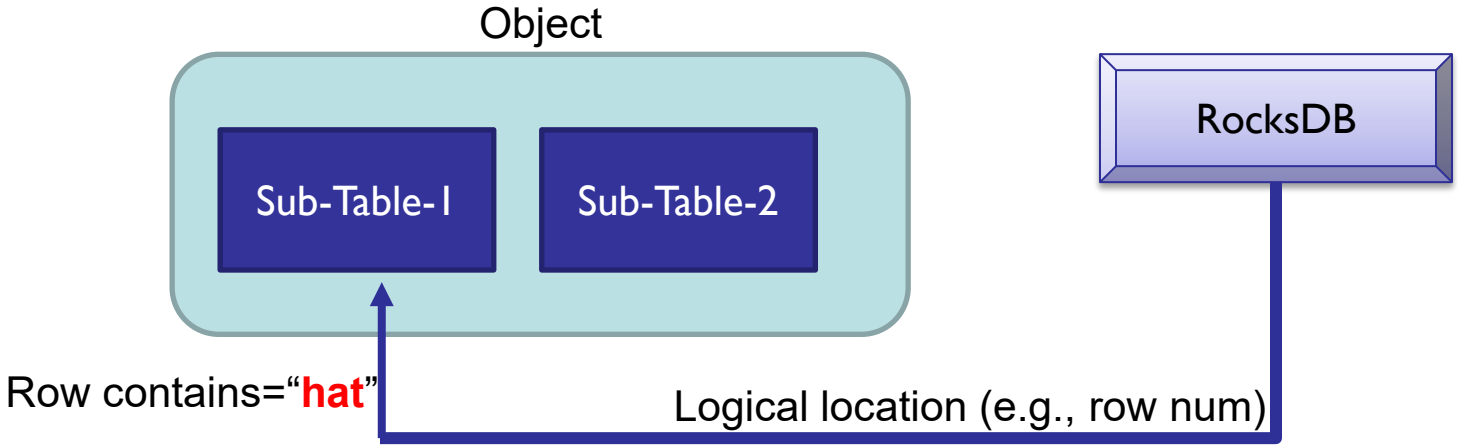
Why use partial read/write of an object?

- Enables great flexibility for physical data layout within each object
- RocksDB enables query-able metadata



Why use partial read/write of an object?

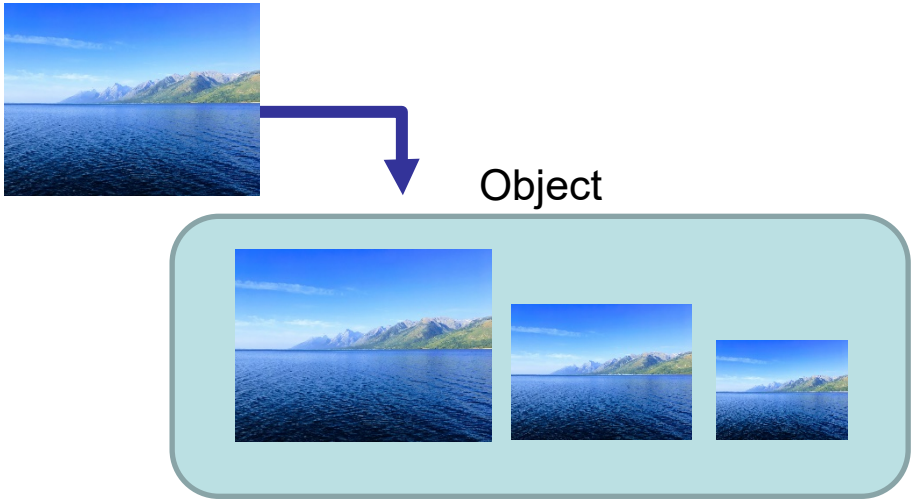
- Enables great flexibility for physical data layout within each object
- RocksDB enables query-able metadata



Example Use Case

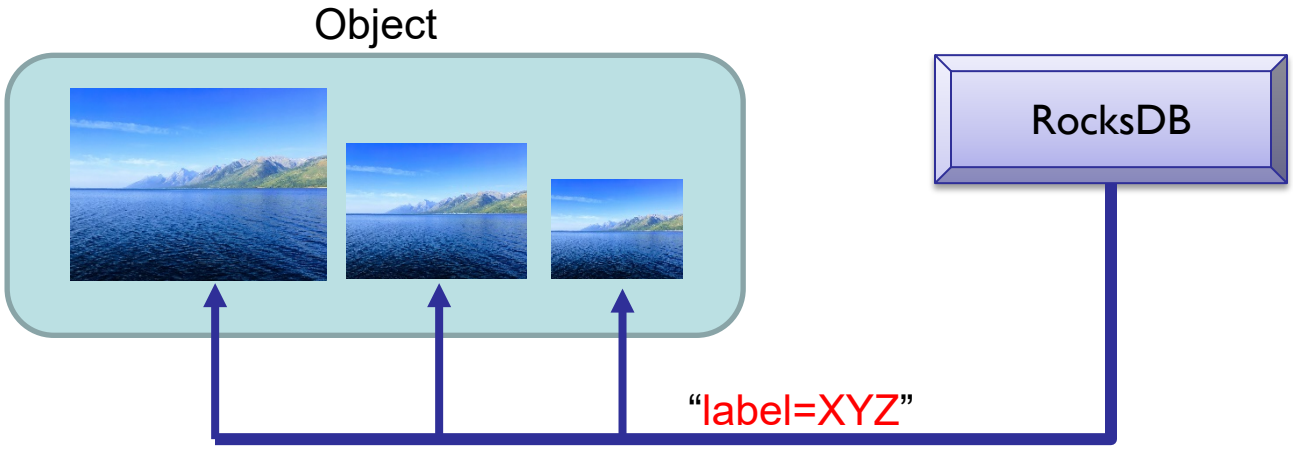
Example Use-case: Custom cls Write

- Write original image, create thumbnails during write



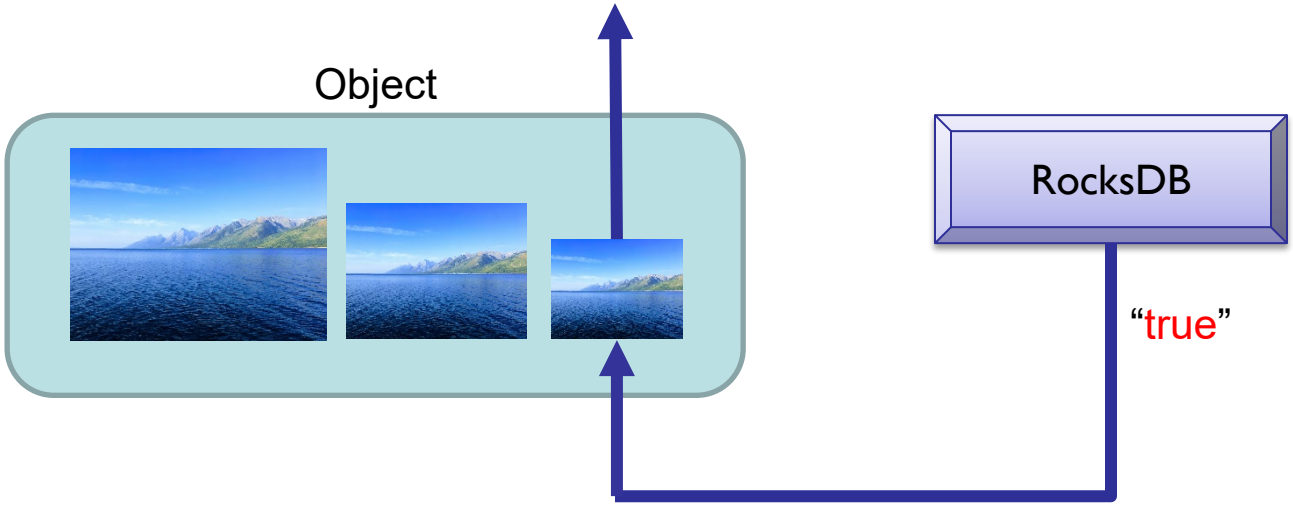
Example Use-case: Custom cls Metadata

- Create metadata – generate image labels, store as metadata in local RocksDB



Example Use-case: Custom cls Read

- Filter data by label="XYZ"



Create and Register Class/Method

C++ snippet

```
CLS_VER(1,0)
CLS_NAME(tabular)


cls_handle_t h_class;
cls_method_handle_t h_exec_query_op;
cls_method_handle_t h_build_index;


void __cls_init()
{
 CLS_LOG(20, "Loaded tabular class!");
 cls_register("tabular", &h_class);


 cls_register_cxx_method(h_class, "exec_query_op",
    CLS_METHOD_RD, exec_query_op, &h_exec_query_op);


 cls_register_cxx_method(h_class, "build_index",
    CLS_METHOD_RD | CLS_METHOD_WR, build_index, &h_build_index);
}
```

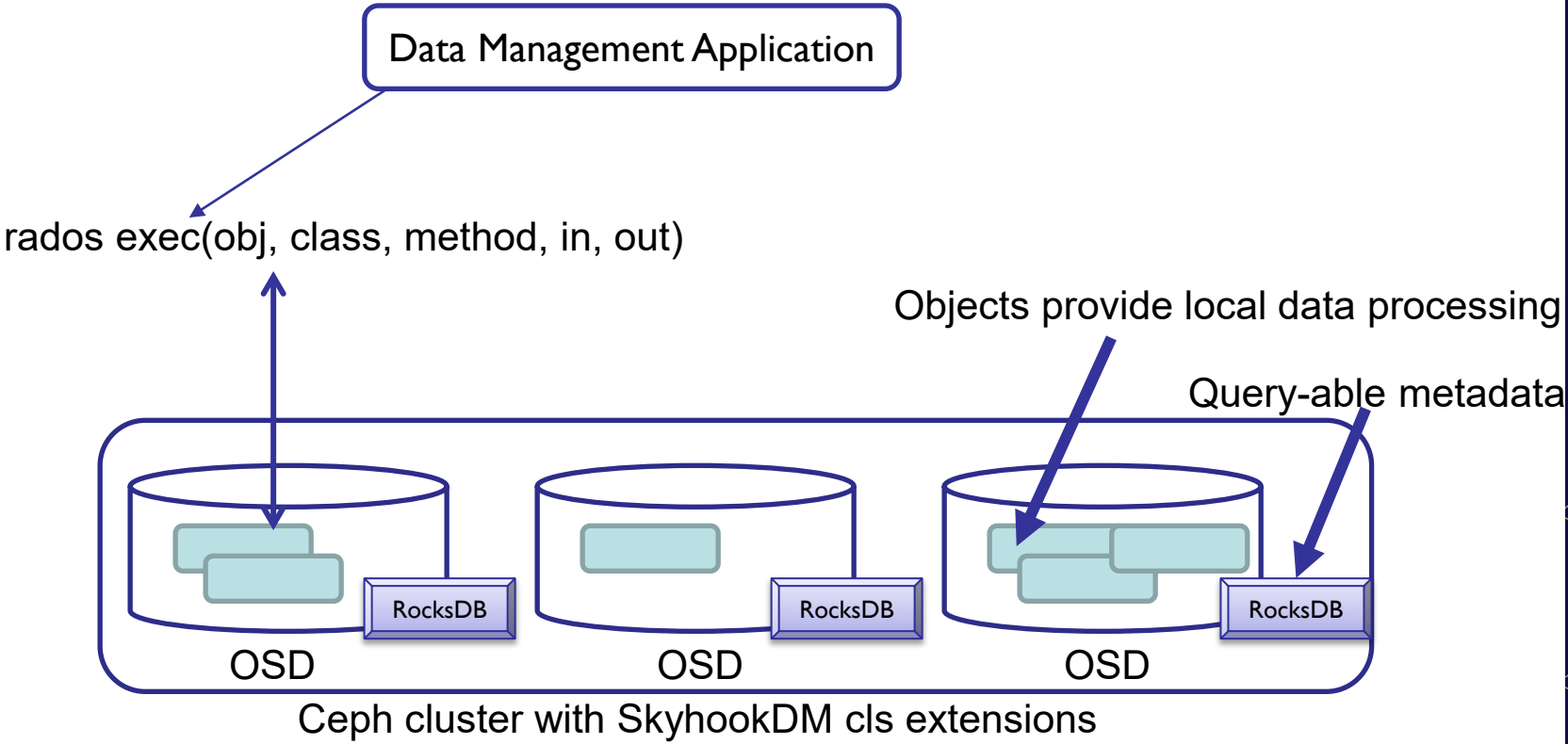
```
static int exec_query_op(cls_method_context_t hctx,
                bufferlist *in,
                bufferlist *out)
{

// contains the serialized user request.
query_op op;

// decode the query op to get the query params
bufferlist::const_iterator it = in->begin();
ceph::decode(op, it);

...

ceph::bufferlist buf;

// lookup metadata
cls_cxx_map_getval(hctx, key, buf)

...

// read local data
int ret = cls_cxx_read(hctx, off, len, &buf);

// process data

...

out->append(result, sizeof(result))

return 0;
}
```

‘cls’ for SkyhookDM

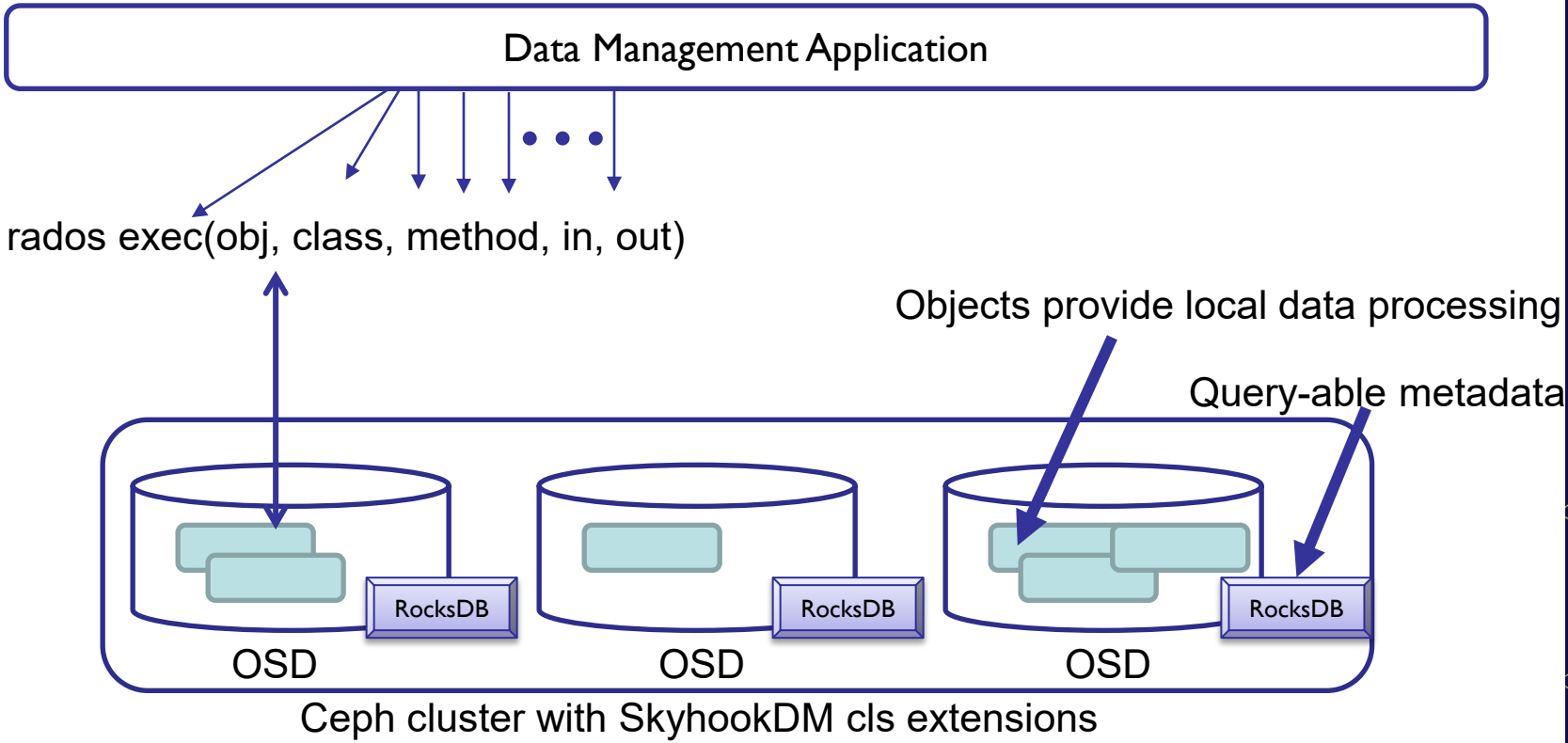
- Note that CLS mechanism ***already exists*** in Ceph
  - Used heavily by Ceph internals as shown
- We create custom read/write methods
  - Our methods are ***not*** Ceph specific
  - C++ code, Arrow library
- We simply utilize Chunk store and KV store interfaces
  - Approach is applicable ***to any system*** that offers such interfaces for objects

SkyhookDM Architecture

SkyhookDM Architecture



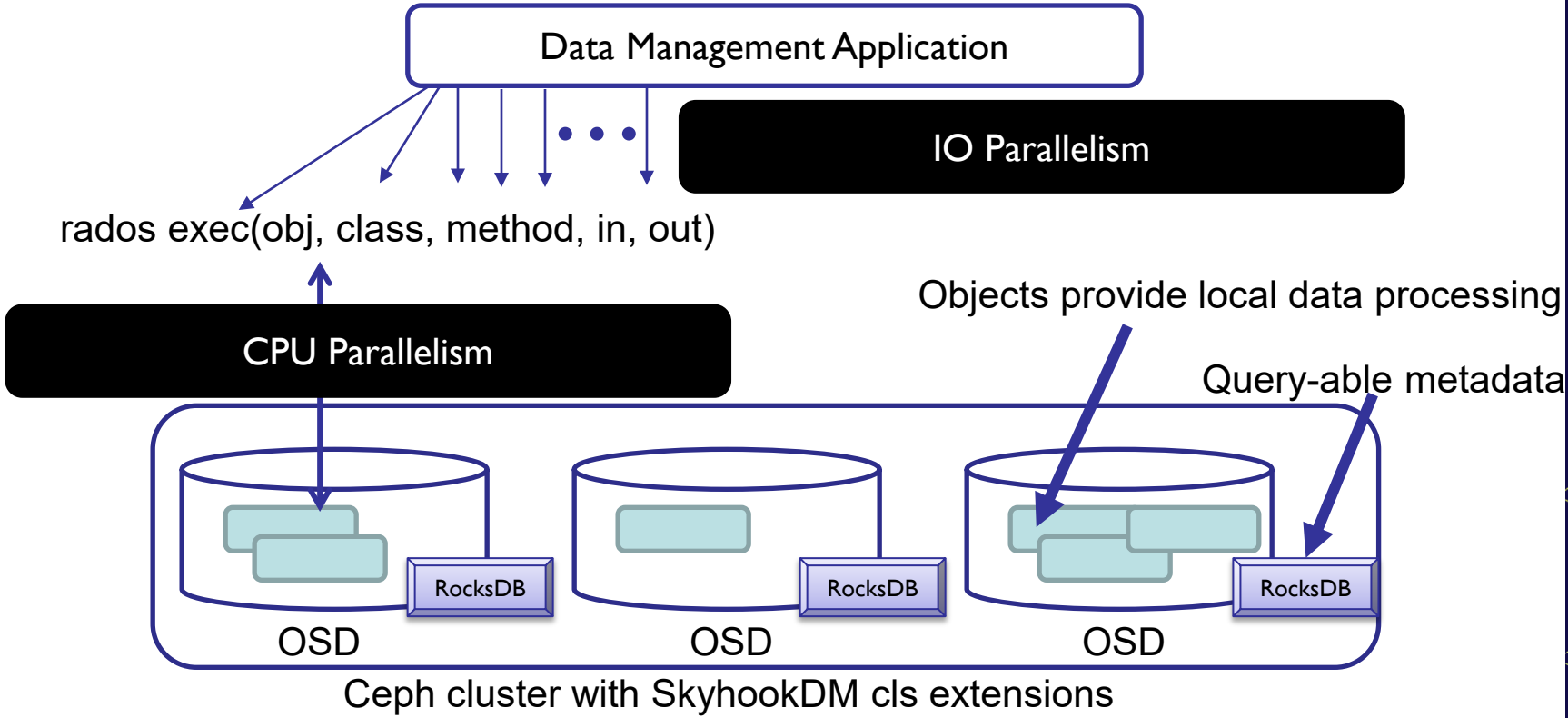
SkyhookDM Architecture



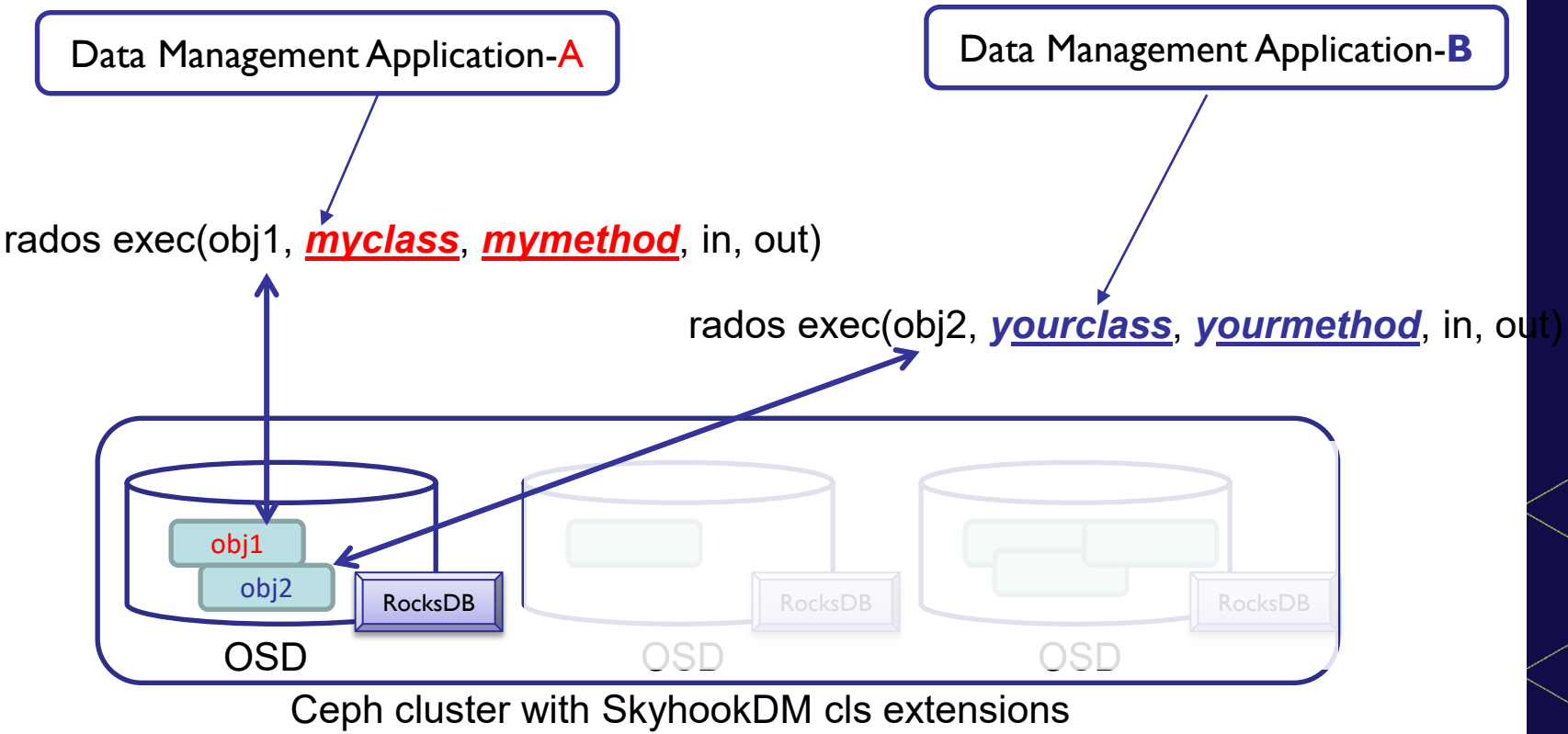
Data Management Application

- Client-side interface to SkyhookDM's LIBRADOS object classes
- Can consider several approaches
  - Distributed processing application frameworks
    - Spark, Dask, others
  - Database External Table interface (widely avail)
    - e.g., PostgreSQL foreign data wrapper
  - FileAPIs that map onto themselves/pass thru
    - e.g., HDF5 Virtual Object Layer (VOL)

SkyhookDM Architecture



‘Multi-tenancy’ of cls computations



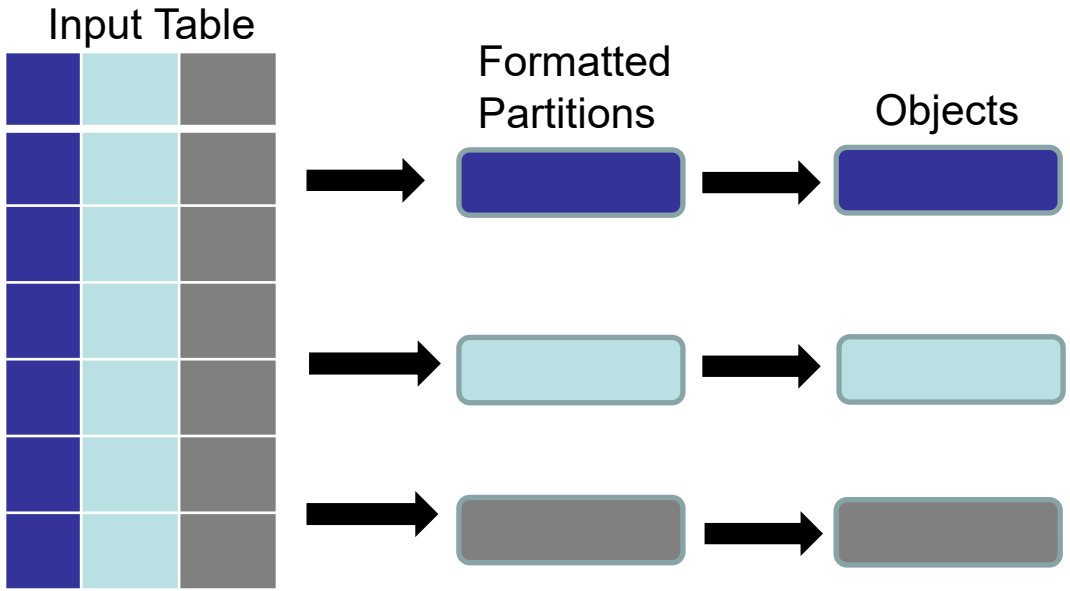
Ceph cluster with SkyhookDM cls extensions

Data Format, Partitioning, Access Methods

Data Formats in SkyhookDM

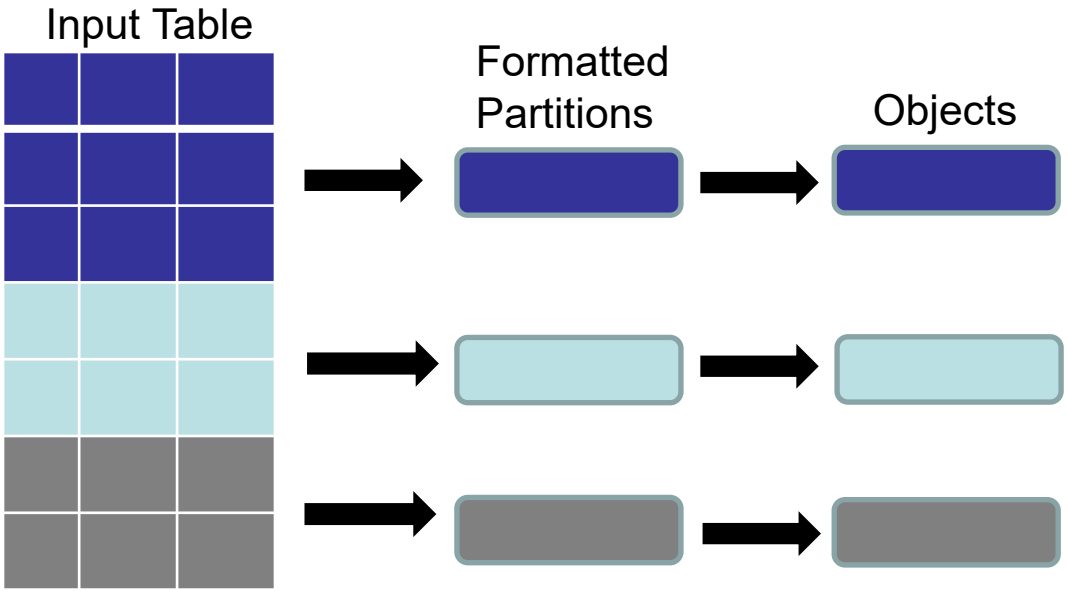
- Utilize fast in-memory serialization formats
  - Google Flatbuffers/Flexbuffers
    - Fields can be accessed without parsing / copying / object allocation.
  - Apache Arrow – Provide “Arrow Native” storage + processing!
    - Very popular in big data world and for data exchange
    - Recent stable release of version 1.0 (July 24, 2020)
    - ***Compute API*** for Arrow tables
    - Recent ***Dataset API***, provides table abstraction over a collection of remote files/fragments

Partitioning in SkyhookDM



Vertical (col) Partitioning

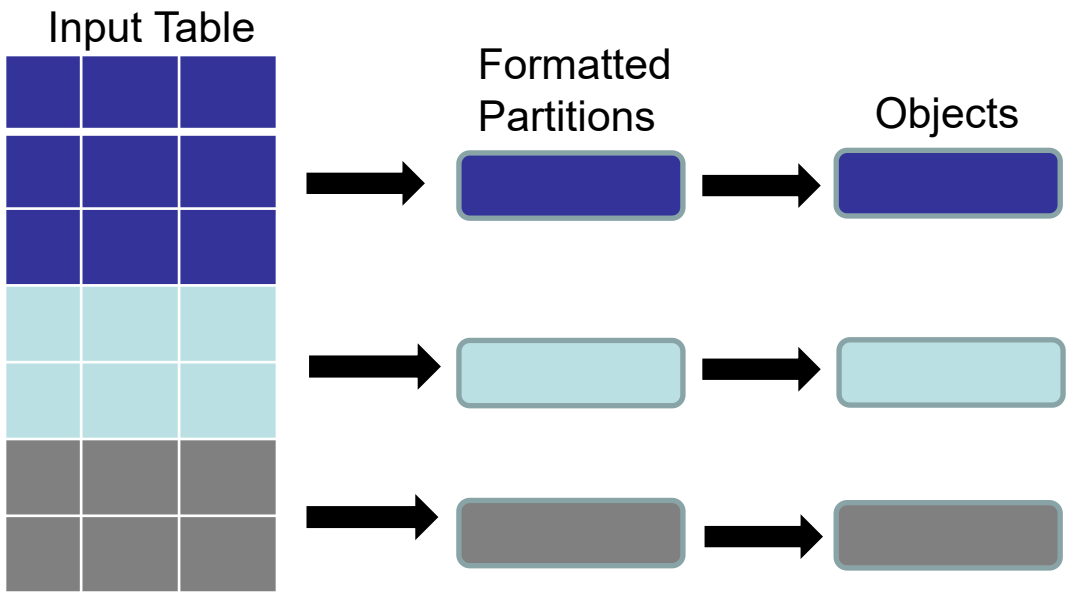
Partitioning in SkyhookDM



Horizontal (row) Partitioning

*uses JumpConsistentHash

Partitioning in SkyhookDM



Horizontal (row) Partitioning

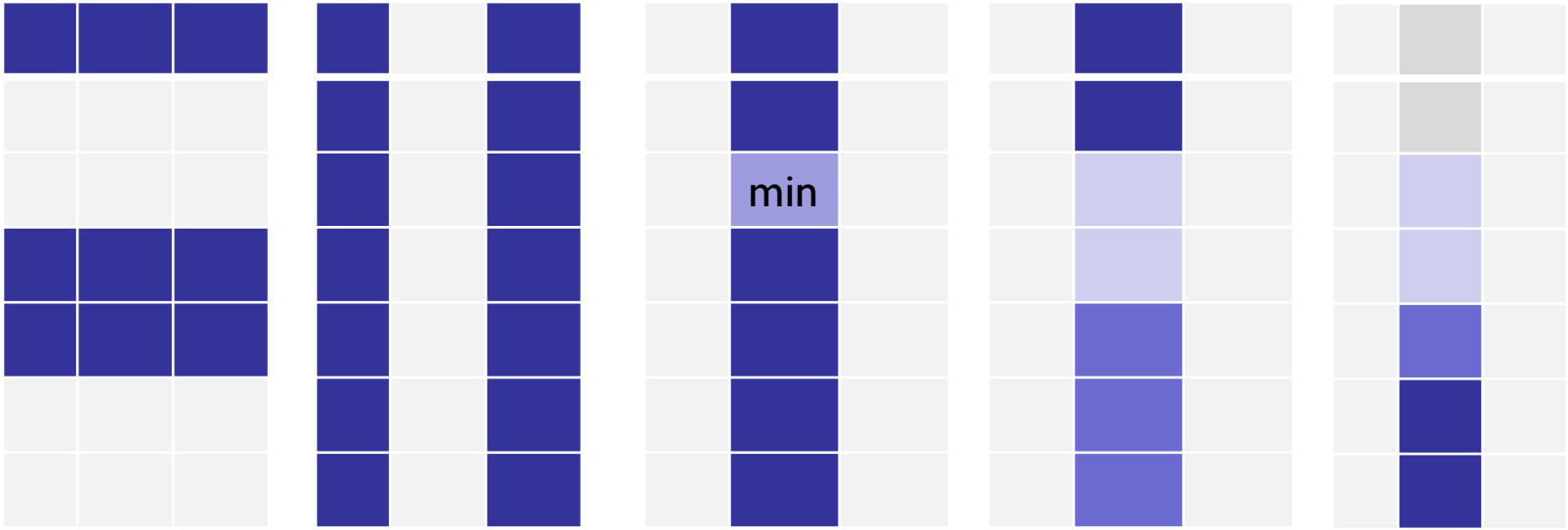
Key properties of partitions

- Format retains data's semantics (data schema)
- Object names are generated
- Objects are distributed by Ceph based on name
- Object location not stored by Skyhook

*uses JumpConsistentHash

Data Processing

- SELECT, PROJECT, Aggregate, Groupby, Sort



- Moving to Arrow Compute API, will support more

What Metadata to Store in RocksDB?

- Physical offsets
- Logical content location
  - Create index on various columns
  - Can consider text indexing as well
- Column statistics, access patterns
  - Value distribution important for query optimization
  - Estimate selectivity, then choose scan or index
- Object-local metadata, so each object can optimize itself

When to Pushdown (offload) Processing

- Currently this is binary to pushdown into storage or not
- Can be a runtime decision by query optimizer based on cluster knowledge
- What if a storage server is overloaded?
  - Object may reject processing, “pushback”
    - We are working on this mechanism
    - Key consideration when offloading in our framework

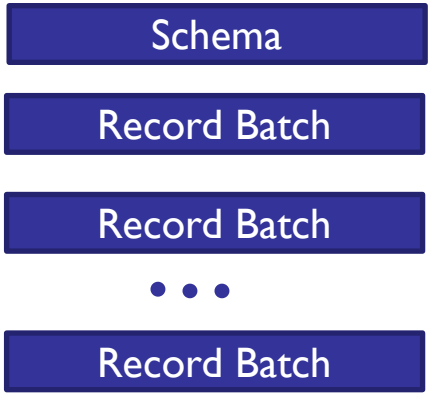
Physical Design Optimizations

Physical Design

- Long-studied problem in databases
- Good physical design is crucial for workload performance
- Includes partitioning, data format, data layout, indexing, materialized views
- In our case
  - Map table data structure to physical layout on disk

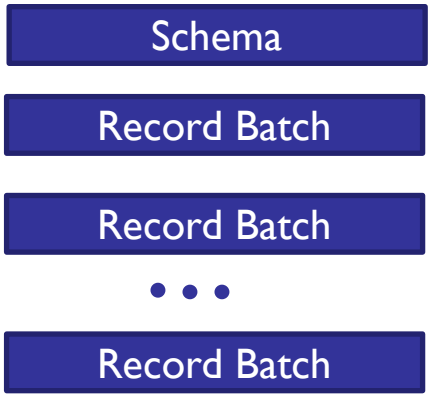
Data Layouts within Object

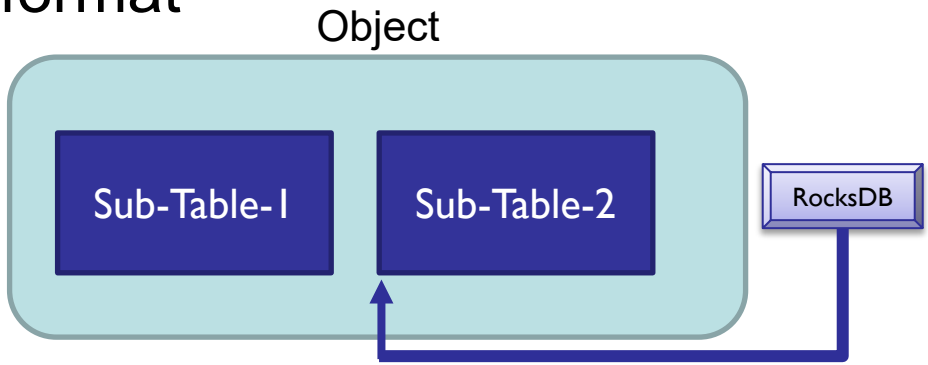
- Consider Arrow table format



Data Layouts within Object

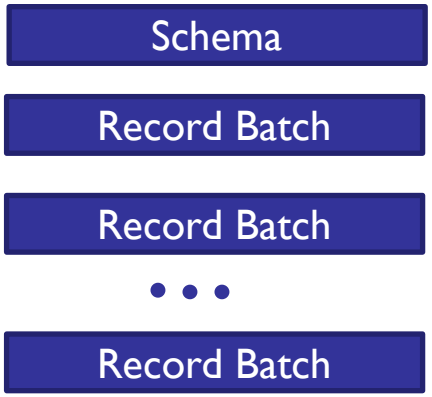
- Consider Arrow table format

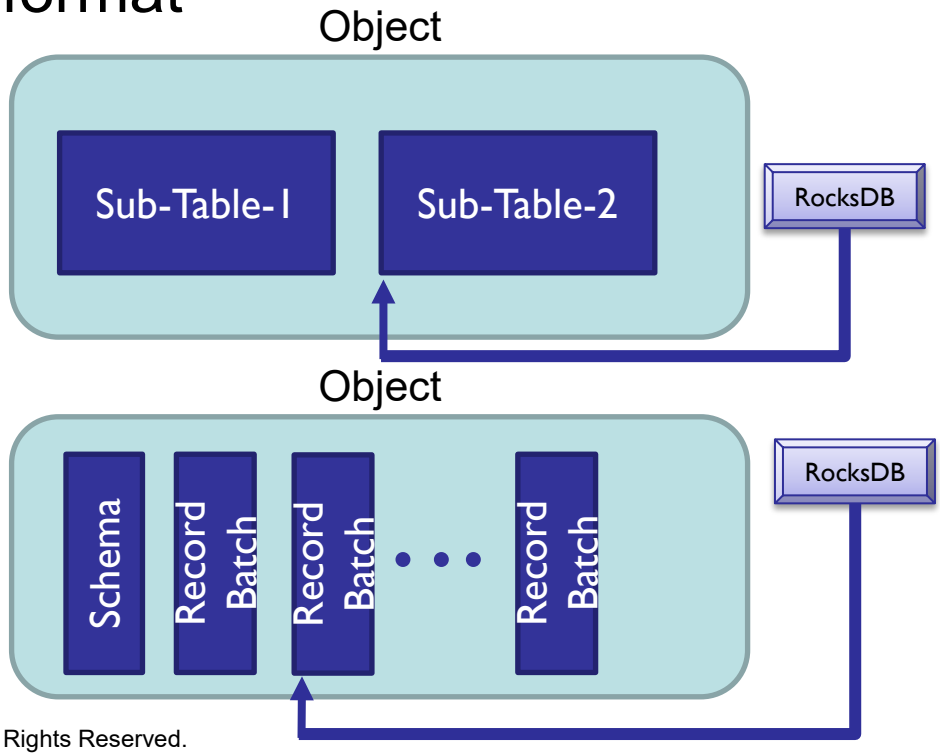




Data Layouts within Object

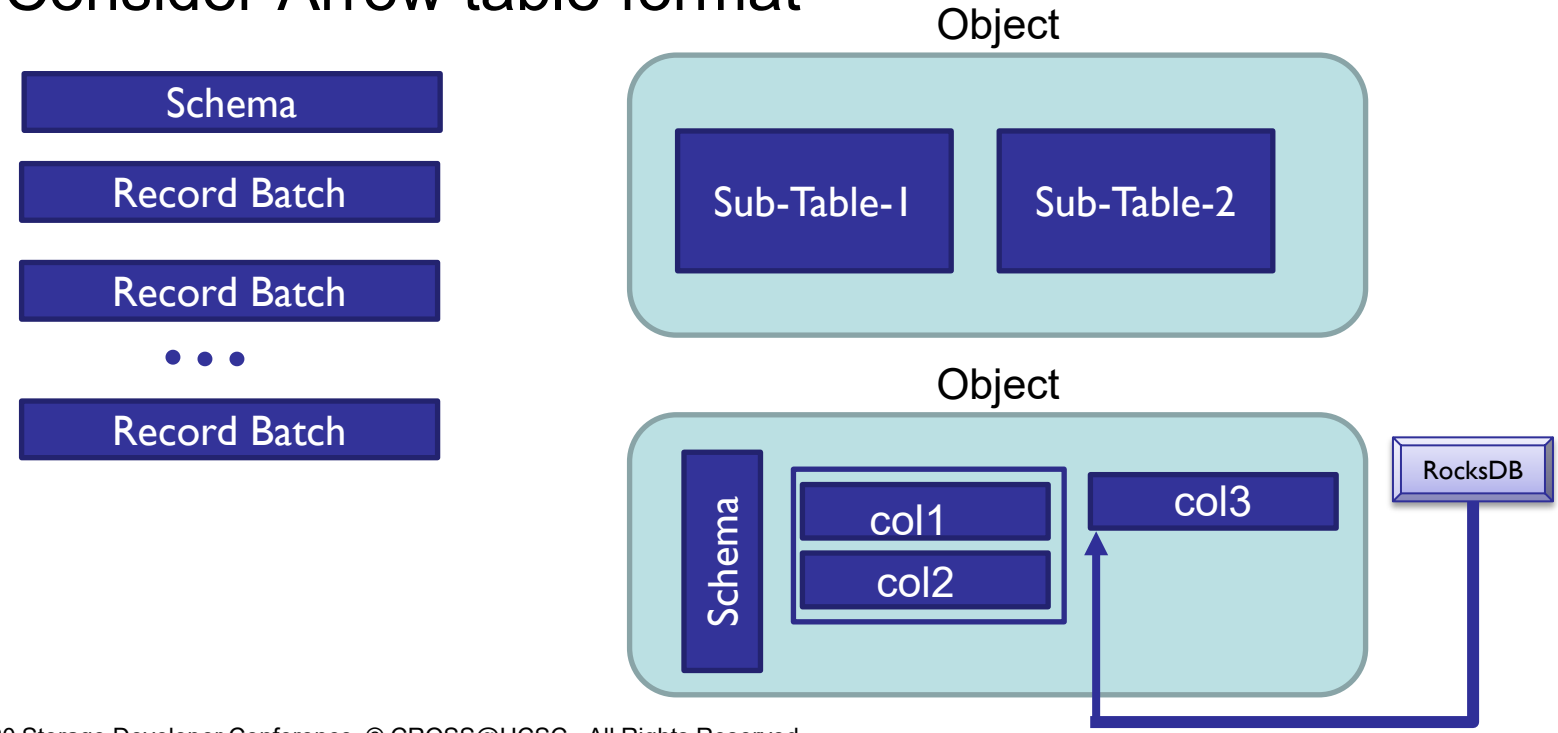
- Consider Arrow table format



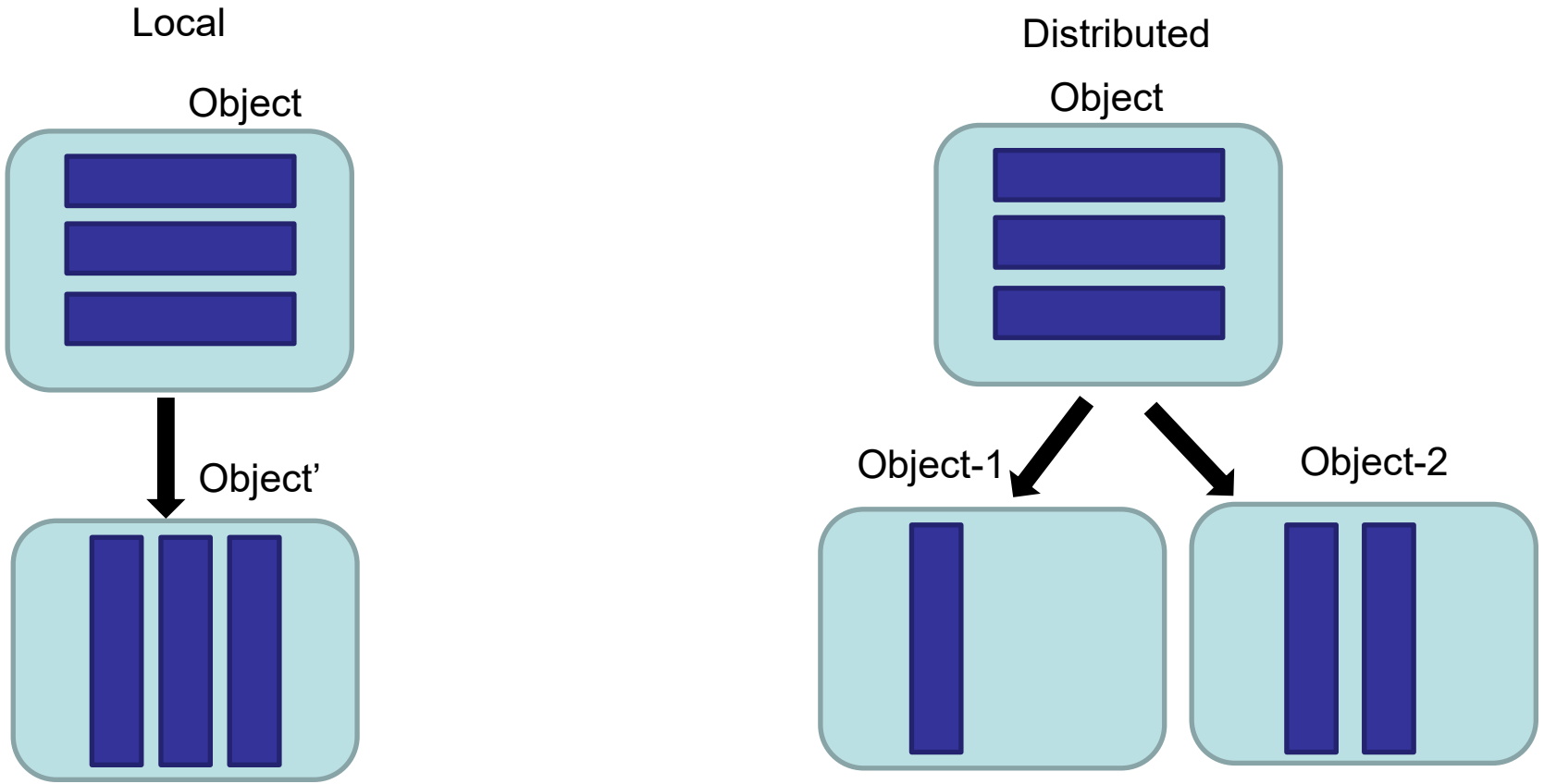


Data Layouts within Object

- Consider Arrow table format



Local vs. Distributed Transformations



Experimental Results

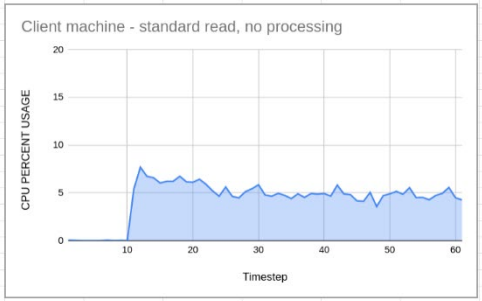
Experimental Setup

- Data: TPC-H Lineitem table 750M rows
  - 10,000 objects, 75,000 rows/object
- Formats: Flatbuffer/Flexbuffer (row), Arrow (col)
- Queries: select and project
  - SELECT * FROM lineitem WHERE extended_price > 91,500.00
  - SELECT extended_price FROM lineitem
- Hardware: NSF Cloudlab 40 core, 10GbE, 1TB HDD
- Ceph with SkyhookDM extensions, 8 OSDs, 1 client machine
- Simple client side driver, process in client or pushdown to storage using SkyhookDM extensions

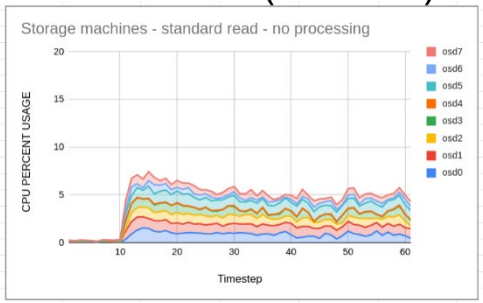
Benefit of Offloading: CPU – 8 OSDs

BASELINE
Read cost only, no query

Client-side



Server-side (stacked)



Benefit of Offloading: CPU – 8 OSDs

BASELINE
Read cost only, no query

SELECT 1%
No offload to storage

Benefit of Offloading: CPU – 8 OSDs

BASELINE
Read cost only, no query

SELECT 1%
No offload to storage

SELECT 1%
Offload to storage

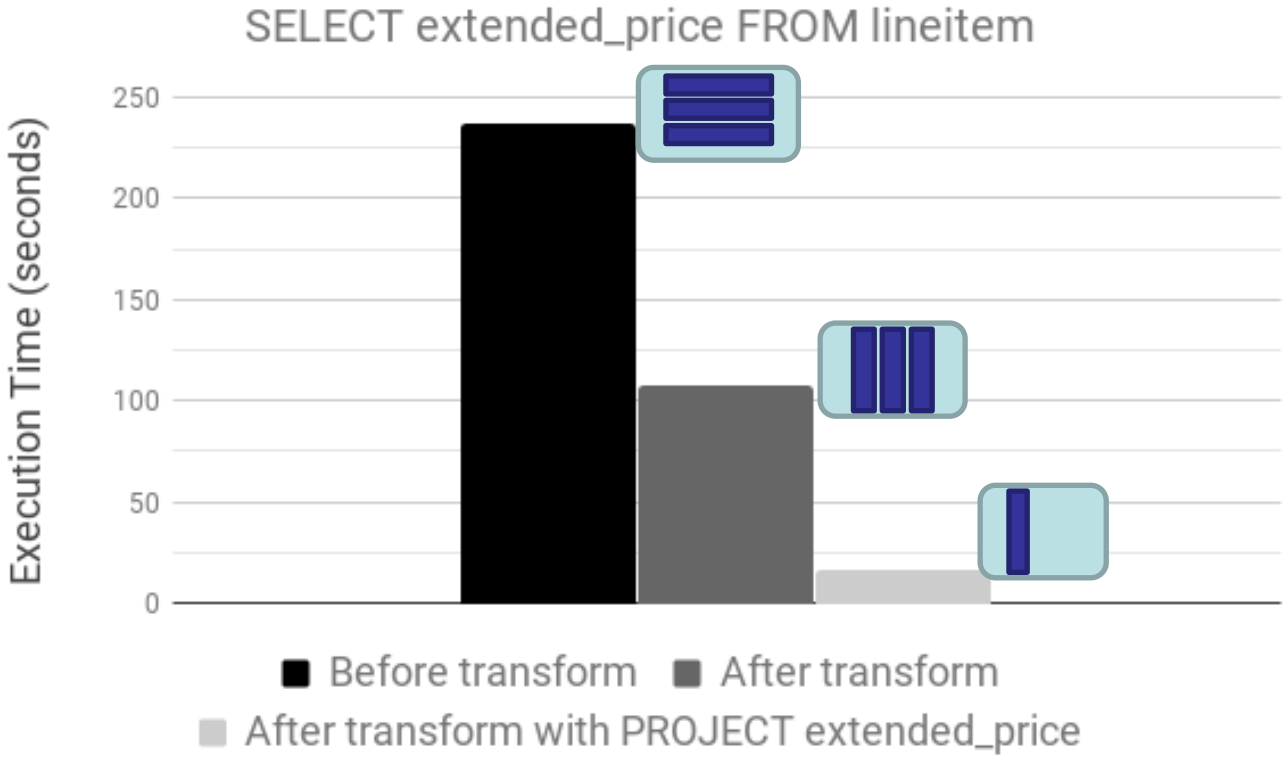
Benefit of Offloading: CPU – 8 OSDs

BASELINE
Read cost only, no query

SELECT 1%
No offload to storage

SELECT 1%
Offload to storage

Benefit of Transform Phys Layout



Ongoing Work

- Adapting to Apache Arrow Dataset API
  - Interacting with Arrow community for feedback
  - Creating a LIBRADOS Fragment, hopefully push upstream (just starting implementation now)
- RADOS read from remote object (collect)
- Deployment now via Kubernetes and Rook

Plug!

CROSS

CENTER FOR RESEARCH IN OPEN SOURCE SOFTWARE

- Bridges gap between student research & open source projects
- Funded by endowment from Sage Weil (Ceph founder) & corporate memberships.
  - **Fujitsu Laboratories**, **Kioxia**, **Seagate**
- Supports graduate research & Incubates work beyond graduation to reach critical mass
  - Skyhookdm project (skyhookdm.com) – store & manage tabular data in Ceph
  - Popper project (getpopper.io) – container native workflow execution engine
- Directed by Carlos Maltzahn carlosm@ucsc.edu
- cross.ucsc.edu for more information

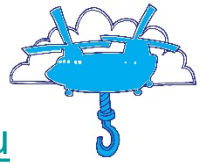
UNIVERSITY OF CALIFORNIA
SANTA CRUZ

Summing it up - SkyhookDM

- ***Data partitioning and layout***
  - Physical mapping of data onto objects
- ***Offload processing***
  - Custom `cls' methods to execute query ops
- ***Offload physical design***
  - Format transformations, indexing, and query-able metadata
- ***Not hacking Ceph***, just using existing mechanism
  - CLS methods updated by copying .so file to OSDs

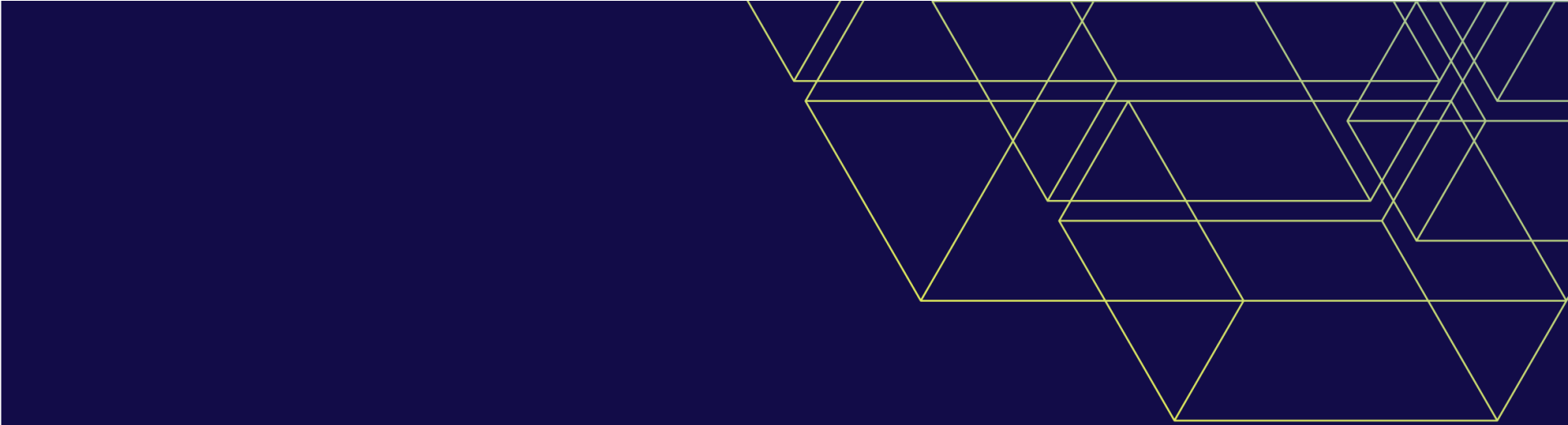
Acknowledgements

- Center for Research in Open Source Software at UCSC (CROSS)
- NSF Grant OAC-1836650, CNS-1764102, CNS-1705021
- IRIS-HEP Software Institute
- Current and previous CROSS Corporate Member companies
- Everyone who has contributed to SkyhookDM project!
  - Esp. Noah Watkins, Michael Sevilla, Ivo Jimenez, Ken Iizawa
  - Many internal and external students, Google Summer of Code fellows, IRIS-HEP fellows, Master's projects and theses
- Thank you!



SkyhookDM.com
jlefevre@ucsc.edu

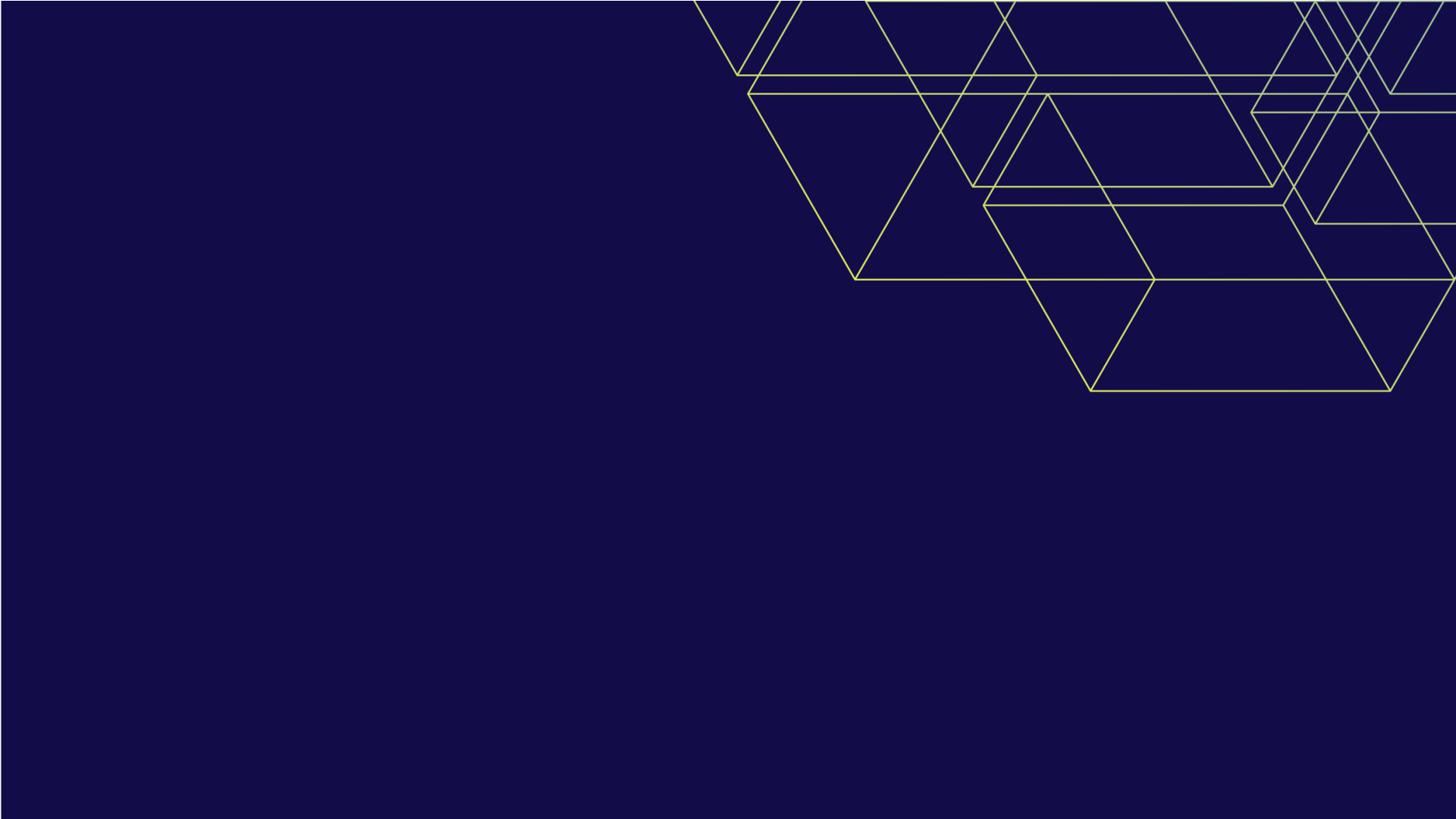
CROSS | CENTER FOR RESEARCH IN OPEN SOURCE SOFTWARE



Please take a moment
to rate this session.

Your feedback matters to us.

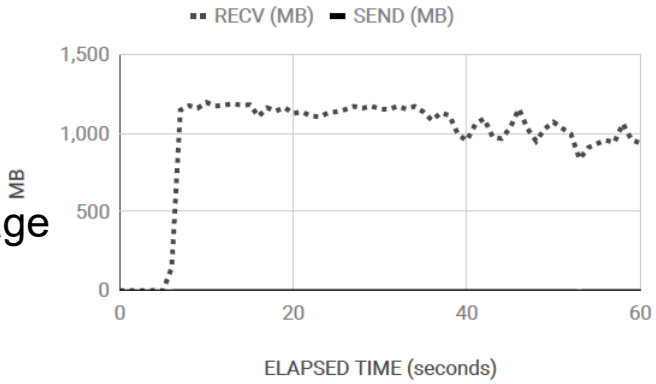
Thank you.



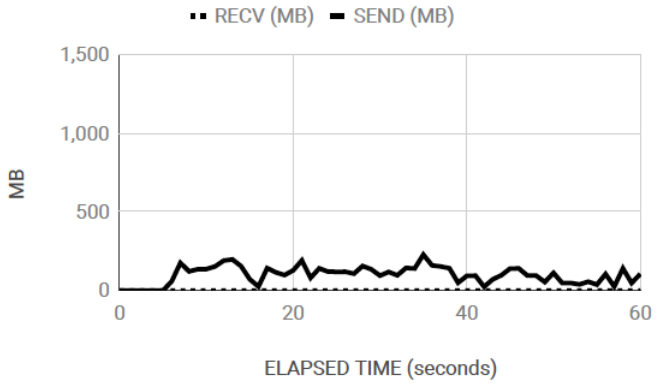
Network Resources 8 OSDs

SELECT 1%
No offload to storage

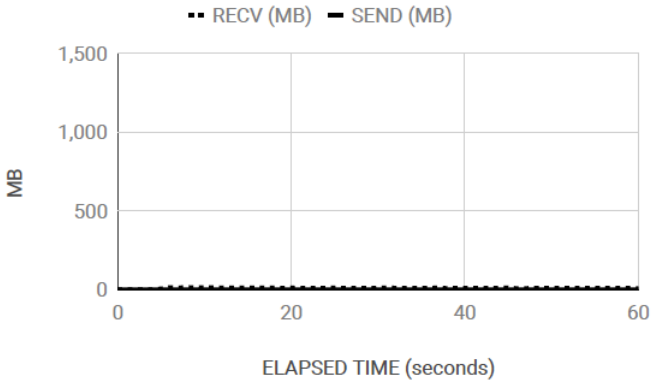
Client

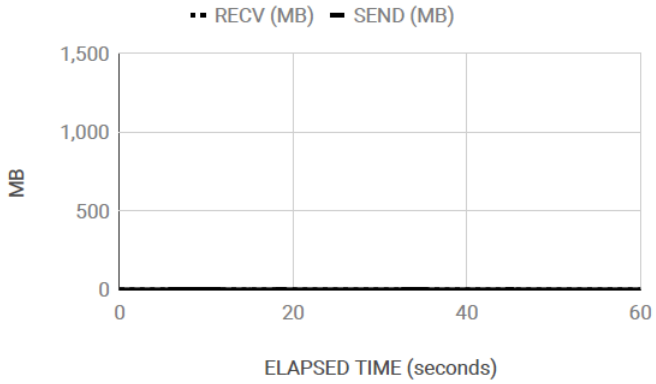


1 Server

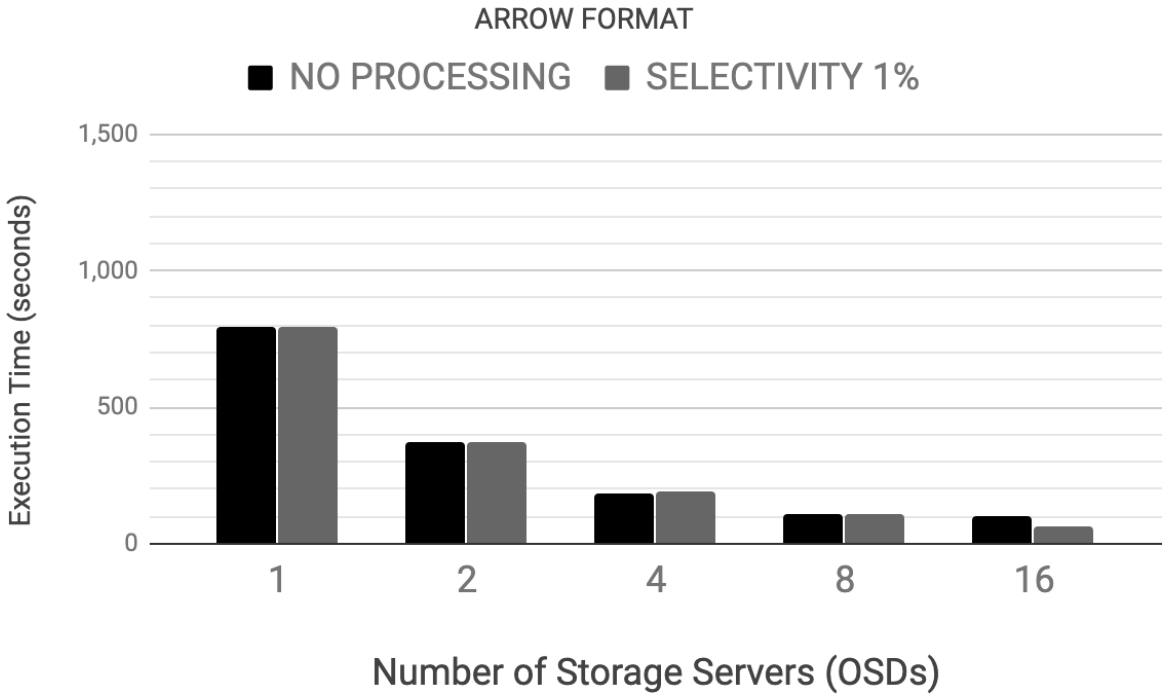


SELECT 1%
Offload to storage

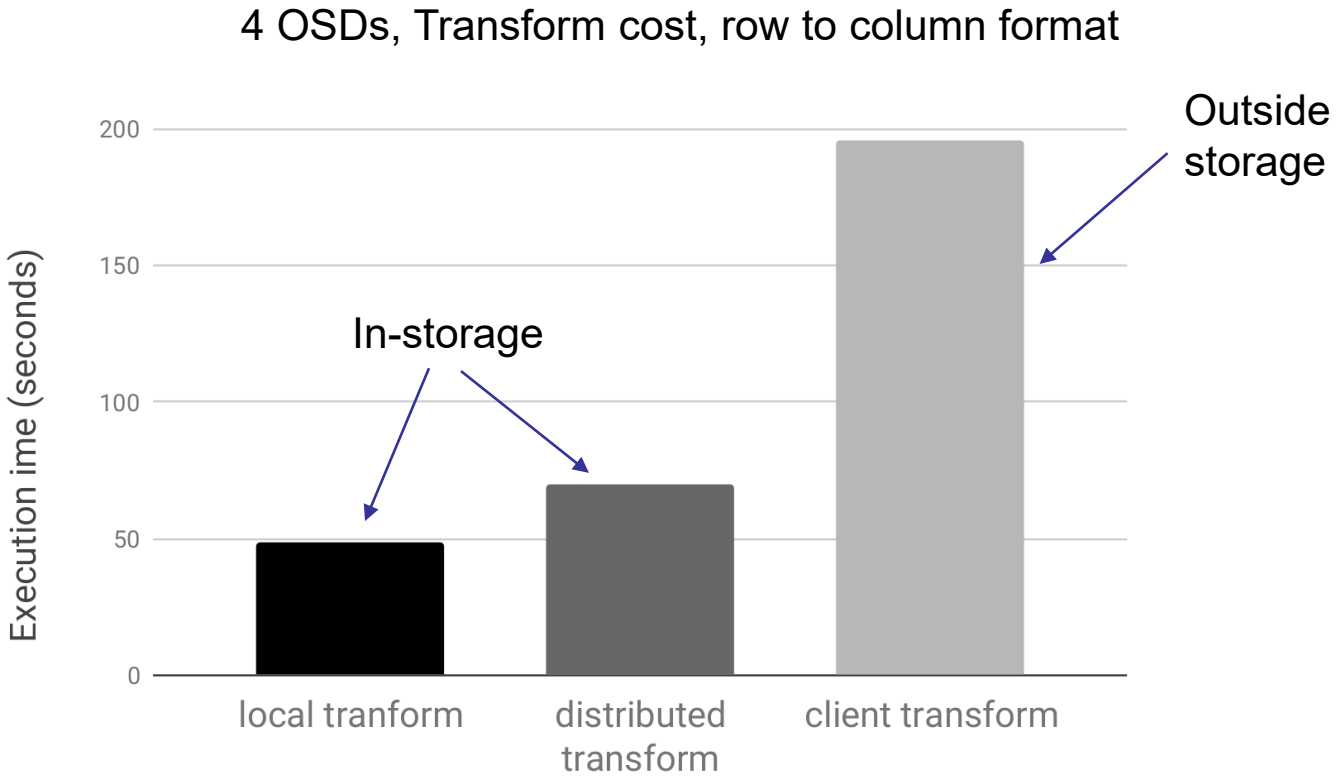




Scaling the number of OSDs



Local vs. Distributed Transform



Average execution time for point query (unique record), client-side vs. server-side processing (pushdown)

Cloudlab c220g2 machines, SSDs, 1 Billion rows, 10K objects, 1x replication, cold cache