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

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

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>2pc_queue</td>
<td>rgw/notifications: persistency - cleanup stale reservations</td>
<td>19 days ago</td>
</tr>
<tr>
<td>cas</td>
<td>cls/cas: replace bool get() with void get()</td>
<td>3 months ago</td>
</tr>
<tr>
<td>cephfs</td>
<td>cls: Build ceph-ossd without using namespace declarations in headers</td>
<td>5 months ago</td>
</tr>
<tr>
<td>cmpomap</td>
<td>cls/cmpomap: add cls module for CMPXATTR-like functionality in omap</td>
<td>5 months ago</td>
</tr>
<tr>
<td>helo</td>
<td>cls: Build ceph-ossd without using namespace declarations in headers</td>
<td>5 months ago</td>
</tr>
<tr>
<td>journal</td>
<td>cls/journal: use EC pool stripe width for padding appends</td>
<td>5 months ago</td>
</tr>
<tr>
<td>lock</td>
<td>cls, rados, rbd, rds, common: Avoid name collision with Windows headers</td>
<td>2 months ago</td>
</tr>
<tr>
<td>log</td>
<td>cls: Build ceph-ossd without using namespace declarations in headers</td>
<td>5 months ago</td>
</tr>
<tr>
<td>lua</td>
<td>librados: add symbol versioning to the C++ API</td>
<td>2 years ago</td>
</tr>
<tr>
<td>rnumops</td>
<td>cls: Build ceph-ossd without using namespace declarations in headers</td>
<td>5 months ago</td>
</tr>
<tr>
<td>otp</td>
<td>cls: Build ceph-ossd without using namespace declarations in headers</td>
<td>5 months ago</td>
</tr>
<tr>
<td>queue</td>
<td>cls/rgw_qc: Clearing off urgent data in bufferlist, before</td>
<td>3 months ago</td>
</tr>
<tr>
<td>rbd</td>
<td>librbd: track in-progress migration aborting operation</td>
<td>22 days ago</td>
</tr>
<tr>
<td>rafcount</td>
<td>cls: Build ceph-ossd without using namespace declarations in headers</td>
<td>5 months ago</td>
</tr>
<tr>
<td>rgw</td>
<td>Merge pull request #31058 from cbradley/wip-rgw-skip-bilog</td>
<td>4 days ago</td>
</tr>
<tr>
<td>rgw_qc</td>
<td>cls/rgw_qc: Fixing carriage returns in log statement</td>
<td>3 months ago</td>
</tr>
<tr>
<td>sdk</td>
<td>cls/sdk: Update cls_sdk.cc to work without using namespace</td>
<td>2 years ago</td>
</tr>
<tr>
<td>timetree</td>
<td>cls: Build ceph-ossd without using namespace declarations in headers</td>
<td>5 months ago</td>
</tr>
<tr>
<td>user</td>
<td>rgw: introduce safe user-reset-stats</td>
<td>2 months ago</td>
</tr>
<tr>
<td>version</td>
<td>cls: Build ceph-ossd without using namespace declarations in headers</td>
<td>5 months ago</td>
</tr>
</tbody>
</table>
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_getvals(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_getvals(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

Object

Sub-Table-1  Sub-Table-2

RocksDB

Row contains=“hat”
Logical location (e.g., row num)
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

![Diagram showing object and RocksDB with "label=XYZ"]
Example Use-case: Custom cls Read

- Filter data by label="XYZ"
Create and Register Class/Method

C++ snippet

```cpp
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

Ceph cluster with SkyhookDM cls extensions

Data Management Application

rados exec(obj, class, method, in, out)

Objects provide local data processing

Query-able metadata

rados exec(obj, class, method, in, out)
SkyhookDM Architecture

Ceph cluster with SkyhookDM cls extensions

Data Management Application

rados exec(obj, class, method, in, out)

Objects provide local data processing

Query-able metadata

rados exec(obj, class, method, in, out)
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

Ceph cluster with SkyhookDM cls extensions

Data Management Application

IO Parallelism

CPU Parallelism

rados exec(obj, class, method, in, out)

Objects provide local data processing

Query-able metadata

rados exec(obj, class, method, in, out)
'Multi-tenancy' of cls computations

Data Management Application-A

rados exec(obj1, *myclass*, *mymethod*, in, out)

Data Management Application-B

rados exec(obj2, *yourclass*, *yourmethod*, in, out)

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

**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

2020 Storage Developer Conference. © CROSS@UCSC. All Rights Reserved.
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

  - Schema
  - Record Batch
  - Record Batch
  - Record Batch
Data Layouts within Object

- Consider Arrow table format

```
<table>
<thead>
<tr>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Batch</td>
</tr>
<tr>
<td>Record Batch</td>
</tr>
<tr>
<td>Record Batch</td>
</tr>
</tbody>
</table>
```

Object

```
Sub-Table-1
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>
Sub-Table-2

RocksDB
```
Data Layouts within Object

- Consider Arrow table format
Data Layouts within Object

- Consider Arrow table format
Local vs. Distributed Transformations

Local

Object

Object'

Distributed

Object

Object-1

Object-2
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
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

SELECT extended_price FROM lineitem

Execution Time (seconds)

- Before transform
- After transform
- After transform with PROJECT extended_price
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
- 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
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!
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

SELECT 1%
Offload to storage
Scaling the number of OSDs

ARROW FORMAT

- NO PROCESSING
- SELECTIVITY 1%

Execution Time (seconds)

Number of Storage Servers (OSDs)

1 2 4 8 16
Local vs. Distributed Transform

4 OSDs, Transform cost, row to column format

Execution time (seconds)

- Local transform
- Distributed transform
- Client transform

Outside storage

In-storage
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

![Chart showing execution time for different number of OSDs, comparing client-side, server-side, and server-side+index processing.](chart.png)