Scaling Databases and File APIs with programmable Ceph Object Storage

VAULT 20

Jeff LeFevre jlefevre@ucsc.edu
Carlos Maltzahn carlosm@ucsc.edu
Bridges gap between student research & open source projects
Funded by Sage Weil endowment & corporate memberships

Teach students on how to productively engage in open source communities
Fund high-impact research with plausible path to successful open source projects
Incubate developer communities around research prototypes

Modeled after NSF’s I/UCRCs. Adds open source software focus. Sustained through membership fees
Postdocs building dev communities for their research prototypes

Graduated with Ph.D. and is well-published expert
Starts out with a significant code base from Ph.D. project
Leverages at least one well-established OSS community
Wants to become an OSS leader

Community seeding via “Research Experience” Programs:
- Tap into pool of students who need project topics
- CROSS is Google Summer of Code Mentor Organization
- Great community management training
- Great driver for community infrastructure

Incubator Fellows

Jeff LeFevre:
SkyhookDM – Programmable Storage for Databases
skyhookdm.com

Kate Compton:
Tracery 2 & Chancery – Getting poets to program AI
tracery.io

Ivo Jimenez:
Black Swan – The Practical Reproducibility Platform
falsifiable.us

Evaluation metric: number of contributors from number of organizations
Exit: when external funding becomes available or project fails reviews
Expected runtime: 2-4 years
Cutting-edge research projects with plausible paths to successful open source software projects

Addresses
a fundamental research question
Is advised by UC Santa Cruz faculty
Is not required to create any software
Opens a plausible path to open source software that might be widely adopted
Has completed coursework required by UC Santa Cruz Ph.D. program

Research Fellows

Xiaowei Chu:
Mapping Datasets to Object Storage (Advisor: Carlos Maltzahn)

Akhil Dixit:
CAvSAT - A System for Query Answering over Inconsistent Databases (Advisor: Phokion Kolaitis)

Jianshen Liu:
Eusocial Storage Devices (Advisor: Carlos Maltzahn)

Sheng Hong Wang:
Lgraph - An Open Source Multi-Language Synthesis and Simulation Infrastructure (Advisor: Jose Renau)

Graduates

Ivo Jimenez (now incubator fellow): Popper - Practical Falsifiable Research (Advisor: Carlos Maltzahn)


Brendan Short: Strong Consistency in Dynamic Wireless Networks for Better Navigation of Autonomous Vehicles (Advisor: Ricardo Sanfelice)

cross.ucsc.edu/symposium

Annual 2-day event with 2 tracks of program and “Systems Oktoberfest”, next event: Oct 7-8, 2020
Centers technical program around current CROSS research and incubator projects
Shows off student work and research programs
Establishes interested communities of students, industry, government, and faculty
Located at Baskin School of Engineering on UC Santa Cruz campus
Skyhook Data Management

- Presented last year at Vault19
- Scaling storage to support database processing
  - Storage layer extensions to Ceph object classes
  - In-storage execution via data access libraries and their APIs
This Talk

• Overview + New developments since Vault19
  – Extensions for Column-oriented storage
    • Apache Arrow Format
  – Extensions for backend plugin support
    • HDF5 Virtual Object Layer
    • High Energy Physics (ROOT) data format
  – Extensions for Physical Design reorganizations
    • Data layouts
Data management in Storage?

• Not a new concept
  – “database machines” of the 1980’s era
    • Customized HW/SW for data management
  – Research today on embedding functions in disks/SSDs/FTLs/FPGAs
• Distributed file systems and customizable software make exploring this a bit easier now
Overview of Our Approach

• Software based
• Open source Ceph object classes extensions
  – User-defined functions (C++, Lua)
  – Customized read/write methods
• Provide data semantics to storage system
• Enable storage to understand and process data locally
Pushdown Processing is an old concept

• Reduce cardinality as early as possible
• Typically processing is done in application layer
Pushdown Processing is an old concept

- Reduce cardinality as early as possible
- Typically processing is done in application layer
Pushdown Processing is an old concept

- Reduce cardinality as early as possible
- Typically processing is done in application layer
Pushdown Processing is an old concept

- Reduce cardinality as early as possible
- Typically processing is done in application layer
Pushdown Processing is an old concept

- Reduce cardinality as early as possible
- Typically processing is done in application layer
What about data management?

• Data reliability concerns
  – Replication, consistency, access control
• Physical design concerns
  • Indexes, materialized views,
  • Partitioning, file format
• Data skew? (object size)

JSON, Protocol Buffers, Parquet, Arrow, Flatbuffers, Avro, Binary Proprietary,...
SkyhookDM Architecture

Application (DB, HDF5 library)

Skyhook: Access layer uses External Data interface

Storage system (Ceph object store)

Skyhook: Objects use our Ceph extensions

Objects contain custom file partitions and native data access methods

Local RocksDB instances provide query-able metadata and indexing
SkyhookDM (now)

- Data storage and processing inside storage software layer via Ceph extensions
- Dynamic reorganization of the physical design
  - Each object independently transformed (or not)
- Adapt to changing workloads
  - Transform row ↔ column formats dynamically
- Support elasticity
  - Repartition objects
Previously Row-oriented

Table data

Row partitions*

part-1 ➔ Obj-1
part-2 ➔ Obj-2
part-3 ➔ Obj-3

Formatted data**

- Data format retains data’s semantics (schema)
- Semantics are interpreted by custom object classes
- We use generated object names
- No location info stored by Skyhook

*Partition rows with JumpConsistentHash

**Partitions formatted as Google Flatbuffers
## (1) Support for Column Processing

<table>
<thead>
<tr>
<th>Table data</th>
<th>Row partitions*</th>
<th>Formatted data**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>part-1</td>
<td>Obj-1</td>
</tr>
<tr>
<td></td>
<td>part-2</td>
<td>Obj-2</td>
</tr>
<tr>
<td></td>
<td>part-3</td>
<td>Obj-3</td>
</tr>
</tbody>
</table>

*Partition rows by Column

**Partitions formatted as Apache Arrow

- Data format retains data’s semantics (schema)
- Semantics are interpreted by custom object classes
- We use generated object names
- No location info stored by Skyhook
- GSoC project
Processing Types

(1 partition/object)  (row-operation)  (col-operation)  (set operation)
ORDER BY/SORT
How to Embed Semantics?

- Flatbuffers and Arrow APIs have extensible schema metadata
  - Column type, name, version, length, etc.
- How to determine which API to use?
  - Enable storage to check data format
  - Flatbuffer metadata wrapper
Data Partition Metadata Required

- **Flatbuffer metadata wrapper** per partition
  - Enables each partition to understand its properties
  - Important for dynamic scalability
  - Database/client app doesn’t need to know state of all objs

```c
table FB_Meta {
  blob_format : int32;  // enum SkyFormatType of contents stored in data blob
  blob_data  : [ubyte];  // formatted data (any supported format)
  blob_size  : uint64;   // number of bytes in data blob
  blob_deleted : bool;    // has this data been deleted?
  blob_orig_off : uint64=0; // optional: offset of blob data in orig file
  blob_orig_len : uint64=0; // optional: num bytes in orig file
  blob_compression : int=0;  // optional: populated by enum {none, lzw, ...}
}
```
Data Partition Metadata Required

- **Flatbuffer metadata wrapper** per partition
  - Enables each partition to understand its properties
  - Important for dynamic scalability
  - Database/client app doesn’t need to know state of all objs

```sql
table FB_Meta {
    blob_format : int32;  // enum SkyFormatType of contents stored in data blob
    blob_data : [ubyte];  // formatted data (any supported format)
    blob_size : uint64;  // number of bytes in data blob
    blob_deleted : bool;  // has this data been deleted?
    blob_orig_off : uint64=0;  // optional: offset of blob data in orig file
    blob_orig_len : uint64=0;  // optional: num bytes in orig file
    blob_compression : int=0;  // optional: populated by enum {none, lzw, ...}
}
```
(2) Scalable APIs

• SkyhookDM object extensions and data format metadata enable multiple formats
• Can now store and process custom formats
• Typically DB layer supports backends via external table interface (foreign data wrapper)
• Scientific file formats
  – HDF w/VOL, ROOT file format (physics)
ROOT FILE Format

Dataset → File(s)

File(s) → TreeName → TBranch → DS.Tr.Br-A.0

File(s) → TreeName → TBranch → DS.Tr.Br-B.0

File(s) → TreeName → TBranch → DS.Tr.Br-C.0

File(s) → TreeName → TBranch → DS.Tr.Br-C.1
ROOT access -> obj access

• Data is stored into objects in a common format
  – Apache Arrow
• Original file replaced by collection of objects
• Objects are accessed in parallel
  – Pushdown select and project
ROOT access -> obj access

- Data is stored into objects in a common format
  - Apache Arrow
- Original file replaced by collection of objects
- Objects are accessed in parallel
  - Pushdown select and project
- Scalable file access AND processing via storage
Python Interface for Scientists

- Python library for ROOT data
- Commonly used by analysts in Jupyter notebooks
- Issues SkyhookDM reads/writes
  - Data returned as pyarrow or dataframes
- Scalable Architecture design
SkyhookDM Python Client Library

User Query → Calculate object names → Generate Skyhook tasks

DASK Scheduler

Dask worker 0

SkyhookDM-read()

dask calls

task_0

Dask worker m

SkyhookDM-read()

dask calls

task_m

Ceph Cluster w/SkyhookDM extensions
SkyhookDM Python Client Library

User Query → Calculate object names → Generate Skyhook tasks → DASK Scheduler

Dask node(s)

- Dask worker 0
task_0
- Dask worker m
task_m

Ceph Cluster w/SkyhookDM extensions

SkyhookDM-read()
calls

SkyhookDM-read()
calls

https://github.com/uccross/skyhookdm-pythonclient
(3) Physical Design Management

• Physical design management (PDSW19)
• Dynamically transform data between row <-> column
  – Match current workload needs
• Very large space of design choices
  – Consider replication, format, num objects, size,...
Results

- Data: TPC-H Lineitem table, 750M rows
- Queries: select and project over lineitem
  - SELECT * from lineitem WHERE extended_price > 91,500.00
  - SELECT extended_price from lineitem WHERE extended_price > 91,500.00
- Hardware: NSF Cloudlab 40 core, 10GbE, 1TB HDD
- App: Postgres 10+, Ceph with Skyhook extensions
Transform row to column

4 node storage cluster (Ceph), 1 node client machine, 750M rows TPC-H Lineitem table

Execution time (seconds)

- Local transform
- Distributed transform
- Client transform

In storage

Outside storage
PROJECT before/after transform

SELECT extended_price FROM lineitem

- ROWs - flatbuffer
- COLs - arrow
- 1 COL - arrow

Execution Time (seconds)

- Before transform
- After transform
- After transform with PROJECT extended_price
Scalability

FLATBUFFER-FLEXBUFFER FORMAT

- NO PROCESSING
- SELECTIVITY 1%

Execution Time (seconds)

Number of Storage Servers (OSDs)
Thank you

Questions please
Acknowledgements

• Center for Research in Open Source Software at UCSC
• NSF Grant OAC-1836650, CNS-1764102, CNS-1705021
• Everyone who has contributed to SkyhookDM project!