SkyhookDB: leveraging programmable storage toward DB elasticity

CROSS Incubator Project

Jeff LeFevre and Noah Watkins
Growing your database now

• Single node architecture
Growing your database now

• Single node architecture
  – Bigger machine!
  – No changes to DB app
• Challenge
  – Changing hardware
  – Big enough?
Growing your database now

- Multi-node architecture (Parallel DB / MPP)
Growing your database now

- Multi-node architecture (Parallel DB / MPP)
  - More machines!
  - More Parallelism
- Challenge
  - Data redistribution
  - Complexity
Growing your database now

• Hybrid Architecture
  – Link independent DBs!
  – More parallelism
• Challenges
  – Coordination
  – Data redistribution
Growing your database now

• Hybrid Architecture
  – Link independent DBs!
  – More parallelism
• Challenges
  – Coordination
  – Data redistrib
Cloud: single node architectures

- **AWS Aurora**, AWS Redshift Spectrum, Snowflake Computing

![Diagram](image-url)
Cloud: MPP architectures

- AWS Aurora, AWS Redshift Spectrum, Snowflake Computing
Cloud: MPP architectures

- AWS Aurora, AWS Redshift Spectrum
- Snowflake Computing
Cloud: DB architectures on S3

- AWS Aurora, AWS Redshift Spectrum, Snowflake Computing

NO DB PROCESSING IS DONE “IN” S3
SkyhookDB

Ceph Distributed Object Store

DB application

CPU

DOMA Workshop 2017 - Flatiron
SkyhookDB

• Single node architecture
  – Simple
• Distrib Object storage
  – Parallel IO, Extensible
SkyhookDB

- Single node architecture
  - Simple
- Distrib Object storage
  - Parallel IO, Extensible
SkyhookDB

- Single node architecture
  - Simple
- Distrib Object storage
  - Parallel IO, Extensible
- Dynamically scale out
  - CPU with storage!
SkyhookDB

• Single node architecture
  – Simple
• Distrib Object storage
  – Parallel IO, Extensible
• Dynamically scale out
  – CPU with storage!
Leverage+Extend Object Storage

• Object-based *data layout*
  – Horizontal partitioning (i.e., sets of rows)
• Object-based *processing operations*
  – SQL, complex functions
• Object-based *metadata management*
  – Indexes, statistics, collection-level optimizations
Utilize trusted open source software

PostgreSQL database

Ceph object storage
Table (rows)

Ceph Server / OSD

Query-able Metadata
- Index values
- Counts
- Min/max/size
- Stats, etc...

Disk
RocksDB

object-1
object-2
object-3

DOMA Workshop 2017 - Flatiron
Unit of Parallelism is an Object

Table (rows)

Ceph Server / OSD

- RocksDB

Query-able Metadata
- Index values
- Counts
- Min/max/size
- Stats, etc...

- object-1
- object-2
- object-3

CPU

DOMA Workshop 2017 - Flatiron
Experimental Setup

• Dataset:
  – TPC lineitem table, 1 billion rows ~140GB
  – Ceph 10K objects ~4MB each (each with index)
• Queries:
  – Range query, regex query (LIKE), point query
• Machines:
  – Ubuntu16, 20 Cores, 500GB SSDs, 10GbE (Cloudlab)
• Skyhook parallelism - 12/node IO dispatch
Range query (selectivity=10%)
Point query (unique row)

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

- **Client-side**
- **Server-side**
- **Server-side + index**

![Bar chart showing execution time](chart.png)

Execution Time (seconds)

<table>
<thead>
<tr>
<th>Number of OSDs</th>
<th>Client-side</th>
<th>Server-side</th>
<th>Server-side + index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.9</td>
<td>2.9</td>
<td>1.4</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>190</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
<td>100</td>
<td>1.4</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>50</td>
<td>1.4</td>
</tr>
<tr>
<td>16</td>
<td>50</td>
<td>25</td>
<td>1.4</td>
</tr>
</tbody>
</table>
Average execution time for CPU intensive (regex) scan (s=10%) query, client-side vs. server-side processing (pushdown)

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

- **client-side**
- **server-side**

<table>
<thead>
<tr>
<th>Number of OSDs</th>
<th>Execution Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>300</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>80</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
</tr>
</tbody>
</table>
Memory Pressure: Data > Memory

Client side processing

Server side processing

Optimized with local knowledge
SkyhookDB Summary

• New architecture for DB scaling
  – Better matches the cloud storage model
  – Take some processing away from DB, push into storage layer
  – More OSDs == More parallelism
• Pushing functionality \textit{directly into the Storage Layer}
  – Processing, indexing, statistics, counts, \textit{metadata querying}, batching, …, others?
    • Domain specific custom functions
    • Works well for highly parallelizable tasks
References (1)


[Bor] H. Boral, D. DeWitt, Database machines, an idea whose time has passed?, Workshop on Database Machines 1983.


[Teradata 2016] Marketing statement: “Multi-dimensional scalability through independent scaling of processing power and storage capacity via our next generation massively parallel processing (MPP) architecture”


References (2)


References (3)


[Kudu 2015] Lipcon et. al *“Kudu: Storage for Fast Analytics on Fast Data”*, white paper.

[Moatti 2017] Too Big to Eat: Boosting Analytics Data Ingestion from Object Stores with Scoop, ICDE 2017
Backup slides
Queries

Qa: **Count query** with 10% selectivity:
SELECT count(*) FROM lineitem1B WHERE l_extendedprice > 71000.0

Qb: **Range query** with 10% selectivity:
SELECT * FROM lineitem1B WHERE l_extendedprice > 71000.0

Qd: **Point query** (unique row) issued with and without index:
SELECT l_extendedprice FROM lineitem1B WHERE l_orderkey=5 AND l_linenumber=3

Qf: **Regex query** with 10% selectivity:
SELECT * FROM lineitem1B WHERE l_comment iLIKE '%uriously%'
Initial comparison with single-node Postgres (selectivity=10%)

Filter: l_extendedprice > 71000

Regex: l_comment like %uriously%

![Graph showing execution time for different numbers of storage servers (OSDs)]
Solving the single node bottleneck - use with MPP arch

- Solves: scale up without adding new MPP nodes
  - Does not solve data redistrib/reassign prob when adding new MPP node
Solving the single node bottleneck - use with Hybrid MPP arch

- Solves Data redistrib when adding new single node
  - Does not solve multi-node xct problem