CS 764: Topics in Database Management Systems
Lecture 27: Pushdown DBMS

Xiangyao Yu
12/8/2021
Announcements

DAWN workshop
  – Reserve a presentation slot using the following google sheet
    https://docs.google.com/spreadsheets/d/1BkO3ZqxNXxHRk1-XTnHmvQ1z66sS4LUVvlJiHS6HIJl/edit?usp=sharing

Project report (DDL: Dec. 18)
  – **Submit to the hotcrp website** (like the proposal)

Submit course evaluation on [aefis.wisc.edu](http://aefis.wisc.edu)
**ICDE 2020**

**FlexPushdownDB: Hybrid Pushdown and Caching in a Cloud DBMS**

Yify Nong, Matt Youli, Matthew Youli, Yuhao Lai, Xiaoyang Yu, Marco Saerdom, Adafu Abubukar, Michael Stonebraker

University of Wisconsin Madison, Massachusetts Institute of Technology, and University of California, Berkeley

**ABSTRACT**

Cloud database systems adopt a straightforward architecture that separates the management of computation and storage. A common bottleneck is such an architecture in the network connecting the computation and storage layers. Two solutions have been explored to mitigate the bandwidth caching and computation pushdown. While both techniques can significantly reduce network traffic, existing DBMS consider them orthogonal techniques and support only one or the other, leaving potential performance benefits unexplored.

This paper explores FlexPushdownDB, an ACID cloud DBMS prototype that supports fine-grained hybrid query execution to coordinate the benefits of caching and computation pushdown in a storage-driven architecture. We build hybrid query execution based on a new concept called separate operators to combine the data from the cache and results from the pushdown processing. Our experimental evaluation on the Cloud Benchmark shows that the hybrid execution outperforms both the conventional caching-only architecture and pushdown-only architecture by 2.2x. In the hybrid architecture, we observe experiments show that Weighted IPU can outperforms the baseline (LP) by 35%.

**ICDE 2020**

**PushdownDB: Accelerating a DBMS Using S3 Computation**

Xiayang Yu, Matt Youli, Matthew Youli, Abraham Hameeni, Marco Saerdom, Adafu Abubukar, Michael Stonebraker

University of Wisconsin Madison, Massachusetts Institute of Technology

**ABSTRACT**

Cloud database systems adopt a straightforward architecture that separates the management of computation and storage. A common bottleneck in such an architecture is the network connecting the computation and storage layers. Two solutions have been explored to mitigate the bandwidth caching and computation pushdown. While both techniques can significantly reduce network traffic, existing DBMS consider them orthogonal techniques and support only one or the other, leaving potential performance benefits unexplored.

This paper explores PushdownDB, an ACID cloud DBMS prototype that supports fine-grained hybrid query execution to coordinate the benefits of caching and computation pushdown in a storage-driven architecture. We build hybrid query execution based on a new concept called separate operators to combine the data from the cache and results from the pushdown processing. Our experimental evaluation on the Cloud Benchmark shows that the hybrid execution outperforms the conventional caching-only architecture and pushdown-only architecture by 2.2x. In the hybrid architecture, we observe experiments show that Weighted IPU can outperforms the baseline (LP) by 35%.

**VLDB 2021**

**Data Management: S3-based Scheduling and Optimization in Cloud DBMSs**

Marcos E. Cardoso, Fabio A. Pires, Fernando A. Capone, Fabio A. Capone, Fabio A. Capone, Fabio A. Capone

**ABSTRACT**

Cloud database systems support efficient storage and computation by using S3. However, existing S3-based solutions for cloud DBMSs are not optimized to efficiently manage storage and computation. This paper presents a new S3-based architecture for cloud DBMSs that supports efficient S3-based scheduling and optimization. Our implementation shows that this approach can improve the performance of S3-based solutions by up to 30%.

**ICDE 2020**

**PushdownDB: Accelerating a DBMS Using S3 Computation**

Xiayang Yu, Matt Youli, Matthew Youli, Abraham Hameeni, Marco Saerdom, Adafu Abubukar, Michael Stonebraker

University of Wisconsin Madison, Massachusetts Institute of Technology

**ABSTRACT**

Cloud database systems adopt a straightforward architecture that separates the management of computation and storage. A common bottleneck in such an architecture is the network connecting the computation and storage layers. Two solutions have been explored to mitigate the bandwidth caching and computation pushdown. While both techniques can significantly reduce network traffic, existing DBMS consider them orthogonal techniques and support only one or the other, leaving potential performance benefits unexplored.

This paper explores PushdownDB, an ACID cloud DBMS prototype that supports fine-grained hybrid query execution to coordinate the benefits of caching and computation pushdown in a storage-driven architecture. We build hybrid query execution based on a new concept called separate operators to combine the data from the cache and results from the pushdown processing. Our experimental evaluation on the Cloud Benchmark shows that the hybrid execution outperforms the conventional caching-only architecture and pushdown-only architecture by 2.2x. In the hybrid architecture, we observe experiments show that Weighted IPU can outperforms the baseline (LP) by 35%.

**VLDB 2021**

**Data Management: S3-based Scheduling and Optimization in Cloud DBMSs**

Marcos E. Cardoso, Fabio A. Pires, Fernando A. Capone, Fabio A. Capone, Fabio A. Capone, Fabio A. Capone

**ABSTRACT**

Cloud database systems support efficient storage and computation by using S3. However, existing S3-based solutions for cloud DBMSs are not optimized to efficiently manage storage and computation. This paper presents a new S3-based architecture for cloud DBMSs that supports efficient S3-based scheduling and optimization. Our implementation shows that this approach can improve the performance of S3-based solutions by up to 30%.
Storage-Disaggregation Architecture

Features of disaggregation architecture

- Computation and storage layers are disaggregated
- Limited computation can happen in the storage layer
Storage-Disaggregation Architecture

Features of disaggregation architecture
- Computation and storage layers are disaggregated
- Limited computation can happen in the storage layer

Advantages
- Lower management cost
- Independent scaling of computation and storage

Disadvantages
- Network becomes a bottleneck
How to Mitigate the Network Bottleneck?

Solution 1: Move data to computation
- Cache storage data in the computation layer
- Example: Snowflake

Solution 2: Move computation to data
- Pushdown computation to the storage layer
- Example: PushdownDB
Key questions to address in this project:
- How to implement relational operators to leverage existing cloud services?
- What are the performance and cost tradeoffs?
PushdownDB implementation
- Single-node, multi-process Python-based database
- Ubuntu 16.04.5 LTS, Python version 2.7.12.

Source code: https://github.com/yxymit/s3filter.git
Simple Cloud Storage (S3)

Virtually infinite storage capacity with relatively low cost

Partition input relations into multiple shards, each shard is stored as a separate object in S3

S3 vs. elastic block store (EBS) vs. local store
- Virtually infinite capacity, shared across all nodes, lower cost, durable
S3 Select

Supports limited SQL queries on **CSV** and **Parquet** data format
- S3 Select recognizes database schema for both data formats
- **Simple queries with predicates and aggregation** (no join, no group-by, no sort, etc.)
PushdownDB – Supported Operators

S3 Select supports
- Filter
- Project
- Aggregate without group-by

PushdownDB supports
- Filter
- Project
- Top-K
- Join
- Group-by
Filter

Server-side filtering
– Compute server loads entire table from S3 and filters locally

Example query:
SELECT col1, col2
FROM R
WHERE col1 < 10
Filter

Server-side filtering
- Compute server loads entire table from S3 and filters locally

S3-side filtering
- Push down predicate evaluation using S3 Select

Example query:
```
SELECT col1, col2
FROM R
WHERE col1 < 10
```
Join

Baseline Join
- Server loads both tables from S3 and joins locally

```sql
SELECT SUM(O_TOTALPRICE)
FROM CUSTOMER, ORDER
WHERE
  O_CUSTKEY = C_CUSTKEY
  AND C_ACCTBAL <= upper_c_acctbal
  AND O_ORDERDATE < upper_o_orderdate
```
Join

Baseline Join
– Server loads both tables from S3 and joins locally

Filtered Join
– Server pushes filtering predicates to S3 to load both tables

```
SELECT SUM(O_TOTALPRICE)
FROM CUSTOMER, ORDER
WHERE
  O_CUSTKEY = C_CUSTKEY
  AND C_ACCTBAL <= upper_c_acctbal
  AND O_ORDERDATE < upper_o_orderdate
```
Bloom Join

- Step 1: Server loads the smaller table, builds a bloom filter using join key
- Step 2: Server sends the filter via S3 Select to load the bigger table
- Bloom filter is pushed down as a predicate

```
SELECT ... 
FROM S3Object
WHERE SUBSTRING('1000011...111101101',
    ((69 * CAST(attr as INT) + 92) % 97) % 68 + 1, 1 ) = '1'
```

```
SELECT SUM(O_TOTALPRICE)
FROM CUSTOMER, ORDER
WHERE
    O_CUSTKEY = C_CUSTKEY
    AND C_ACCTBAL <= upper_c_acctbal
    AND O_ORDERDATE < upper_o_orderdate
```
SELECT SUM(O_TOTALPRICE)
FROM CUSTOMER, ORDER
WHERE
  O_CUSTKEY = C_CUSTKEY
AND C_ACCTBAL <= upper_c_acctbal
AND O_ORDERDATE < upper_o_orderdate
Overall, PushdownDB reduces runtime by 6.7× and reduces cost by 30%
ICDE 2020

**PushdownDB: Accelerating a DBMS Using S3 Computation**

Xiaying Yu¹, Matt Youzil¹, Matthew Woicik¹, Abraham Ghanem¹, Marco Serfati³, Ashraf Abdalraoof¹, Michael Stonebraker¹

University of Wisconsin Madison | Massachusetts Institute of Technology

Tobias "Georgia" Computing Research Institute | University of Massachusetts Amherst

Email: yuxiaway@gmail.com, mattyouzil@gmail.com, matthewwoicik@gmail.com, abrahamghanem@gmail.com, marco.serfati@cs.wisc.edu, ashabraf@csail.mit.edu, stonebraker@umass.edu

---

**ABSTRACT**

Cloud data storage provides an abstract and decoupled architecture that separates management of computation and storage. A cloud database is such an architecture in the network connecting the computation and storage layers. Two solutions have been explored to mitigate the bandwidth caching and computing problem: PushdownDB [6] builds an S3 cloud storage system that supports fine-grained hybrid query execution to combine the benefit of caching and computing in pushdown systems in a cloud-computing architecture; We build hybrid query systems based on new concept called separate operators to combine the data from the cache and results from the pushdown processing. We also propose a novel Weighted LP cache replacement policy that takes into account the cost of pushdown computation. Our experimental evaluation on the Star-Templates benchmark shows that the hybrid execution outperforms both the conventional caching-only architecture and pushdown-only architecture by 2.2X. In the hybrid architecture, we experiment shows that Weighted LP can outperform the baseline LP by 1.7X.

**BIBLIOGRAPHIC FORMAT**

Yi Yang, Matt Youzil, Matthew Woicik, Xiaying Yu, Marco Serfati, Ashraf Abdalraoof, Michael Stonebraker.

PushdownDB: Hybrid Pushdown and Caching in a Cloud DBMS.


**FlexPushdownDB: Hybrid Pushdown and Caching in a Cloud DBMS**

Yi Yang¹, Matt Youzil¹, Matthew Woicik¹, Yuhao Liu¹, Xiaying Yu¹, Marco Serfati³, Ashraf Abdalraoof¹, Michael Stonebraker¹

University of Wisconsin Madison | Massachusetts Institute of Technology | University of Massachusetts Amherst

Tobias "Georgia" Computing Research Institute | University of Massachusetts Amherst

Email: yuxiaway@gmail.com, mattyouzil@gmail.com, matthewwoicik@gmail.com, abrahamghanem@gmail.com, marco.serfati@cs.wisc.edu, ashabraf@csail.mit.edu, stonebraker@umass.edu

---

**ABSTRACT**

Cloud data storage provides an abstract and decoupled architecture that separates management of computation and storage. A cloud database is such an architecture in the network connecting the computation and storage layers. Two solutions have been explored to mitigate the bandwidth caching and computing problem: PushdownDB [6] builds an S3 cloud storage system that supports fine-grained hybrid query execution to combine the benefit of caching and computing in pushdown systems in a cloud-computing architecture; We build hybrid query systems based on new concept called separate operators to combine the data from the cache and results from the pushdown processing. We also propose a novel Weighted LP cache replacement policy that takes into account the cost of pushdown computation. Our experimental evaluation on the Star-Templates benchmark shows that the hybrid execution outperforms both the conventional caching-only architecture and pushdown-only architecture by 2.2X. In the hybrid architecture, we experiment shows that Weighted LP can outperform the baseline LP by 1.7X.

**BIBLIOGRAPHIC FORMAT**

Yi Yang, Matt Youzil, Matthew Woicik, Xiaying Yu, Marco Serfati, Ashraf Abdalraoof, Michael Stonebraker.

FlexPushdownDB: Hybrid Pushdown and Caching in a Cloud DBMS.


---

**ICDE 2020**

---

**VLDB 2021**
Mitigate Network Bottleneck

**Baseline:** always load data from cloud storage (e.g., S3)
- Examples: default presto, hive, SparkSQL, etc.
Mitigate Network Bottleneck

**Baseline**: always load data from cloud storage (e.g., S3)

**Caching**: cache hot table data in the compute node
- Examples: Snowflake, redshift spectrum (static), Alluxio, etc.
Mitigate Network Bottleneck

**Baseline**: always load data from cloud storage (e.g., S3)

**Caching**: cache hot table data in the compute node

**Pushdown**: push down selection, projection, aggregation to storage
  - Examples: Redshift spectrum, Aqua, PushdownDB, etc.
Caching vs. Pushdown

Caching performance increases with a bigger cache.

Pushdown performance is independent of cache size.
Caching vs. Pushdown

**Caching** performance increases with a bigger cache

**Pushdown** performance is independent of cache size

A **hybrid** design may achieve the best of both worlds
Mitigate Network Bottleneck

Baseline (Pullup): always load data from cloud storage (e.g., S3)
Caching: cache hot table data in the compute node
Pushdown: push down selection, projection, aggregation to storage
Hybrid: hybrid caching and pushdown at fine granularity
FlexPushdownDB (FPDB) Overview

Design choices
– Cache table data rather than query results for simplicity

Source code: https://github.com/cloud-olap/FlexPushdownDB
FlexPushdownDB (FPDB) Overview

Design choices

– Cache table data rather than query results for simplicity
– Segment as the caching granularity

Source code: https://github.com/cloud-olap/FlexPushdownDB
FlexPushdownDB (FPDB) Overview

Design choices

– Cache table data rather than query results for simplicity
– Segment as the caching granularity

Source code: [https://github.com/cloud-olap/FlexPushdownDB](https://github.com/cloud-olap/FlexPushdownDB)
FlexPushdownDB (FPDB) Overview

Main modules

Hybrid executor → Cache manager
- Admission
- Eviction

Query plan

Local Cache

Caching request
FlexPushdownDB (FPDB)

Separable operators
- Can execute separately using cached segments and cloud storage
- Example: projection, selection, aggregation, hash join (partially)
FlexPushdownDB (FPDB)

Separable operators
– Can execute separately using cached segments and cloud storage
– Example: projection, selection, aggregation, hash join (partially)

Query execution
– Heuristic: exploit caching when possible, otherwise pushdown as much as possible
Separable Query Plan — Example

```
SELECT R.B, sum(S.D)
FROM R, S
WHERE R.A = S.C AND R.B > 10 AND S.D > 20
GROUP BY R.B
```
Cache Manager

Traditional caching assumption: **Equal-size cache misses incur the same cost**
Traditional caching assumption: Equal-size cache misses incur the same cost

In FPDB, misses that cannot exploit pushdown have higher cost, and should be considered for cached with higher priority
Traditional caching assumption: **Equal-size cache misses incur the same cost**

In FPDB, misses that cannot exploit pushdown have higher cost, and should be considered for cached with higher priority

**Weighted-LFU** cache replacement policy

- Increment the frequency counter with the estimate miss cost
- Estimated miss cost = network cost + scan cost + compute cost
Conclusion: FPDB outperforms baselines by 2.2x
Evaluation – Weighted-LFU

Weighted-LFU outperforms the baseline LFU by 37%
### Evaluation – Resource Usage

Table 2: Network Usage (GB) of different architectures.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Pullup</th>
<th>PD-only</th>
<th>CA-only</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>460.9</td>
<td>37.1</td>
<td>112.6</td>
<td>7.9</td>
</tr>
</tbody>
</table>
Table 2: Network Usage (GB) of different architectures.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Pullup</th>
<th>PD-only</th>
<th>CA-only</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>460.9</td>
<td>37.1</td>
<td>112.6</td>
<td>7.9</td>
</tr>
</tbody>
</table>

Table 3: CPU Usage (with dedicated compute servers) — CPU time (in minutes) of different architectures (normalized to the time of 1 vCPU).

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Pullup</th>
<th>PD-only</th>
<th>CA-only</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute</td>
<td>249.6</td>
<td>48.5</td>
<td>70.3</td>
<td>23.2</td>
</tr>
<tr>
<td>Storage</td>
<td>0.0</td>
<td>31.1</td>
<td>0.0</td>
<td>7.4</td>
</tr>
<tr>
<td>Total</td>
<td>249.6</td>
<td>79.6</td>
<td>70.3</td>
<td>30.6</td>
</tr>
</tbody>
</table>
Pushdown DBMS – Q/A

Caching query results better than caching input tables?
When it is able to pushdown, is it always better to pushdown?
ML-based model to replace benefit-based caching?
More accurate way to estimate the weights in WLFU?
Why caching and pushdown were orthogonal in other systems?
What is the most significant difficulty in this work?
Why use exiting storage services instead of inventing new ones?
Pre-known the R/W set for WLFU?
Compute layer’s disk used for anything by FPDB?
Pushdown DBMS – Q/A

Collaborative caching across multiple compute nodes?
Limitations of FPDB?
Can support other storage services? Why S3?
Hybrid caching and pushdown for OLTP workload?
Row-store vs. column-store database?
Next Week

DAWN Workshop

– Online workshop using the lecture zoom link
– Reserve a presentation slot using the following google sheet
  https://docs.google.com/spreadsheets/d/1BkO3ZqxNXxHRkI-XTnHmvQ1z66sS4LUVvJiHS6HIJI/edit?usp=sharing
– Each group has a 10 min slot: **8 min presentation + 2 min Q/A**
– Live presentation preferred, but recording is also ok

Submit course evaluation on [aefis.wisc.edu](http://aefis.wisc.edu)