Insights of Approximate Query Processing Systems

Presented by:  Huanyi Chen
               Ruoxi Zhang
Agenda

- Introduction
- Background
- VerdictDB & SnappyData
- Experiment Setup
- Evaluation
- Insights
Why AQP?

<table>
<thead>
<tr>
<th># of Day</th>
<th>Income (CAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>2</td>
<td>240</td>
</tr>
<tr>
<td>3</td>
<td>180</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>230</td>
</tr>
<tr>
<td>6</td>
<td>190</td>
</tr>
<tr>
<td>7</td>
<td>180</td>
</tr>
</tbody>
</table>

Avg(Income) = 195.71

shop income
Why AQP?

more efficient (50% rows)
accuracy > 95%

<table>
<thead>
<tr>
<th># of Day</th>
<th>Income (CAD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>2</td>
<td>240</td>
</tr>
<tr>
<td>3</td>
<td>180</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>230</td>
</tr>
<tr>
<td>6</td>
<td>190</td>
</tr>
<tr>
<td>7</td>
<td>180</td>
</tr>
</tbody>
</table>

Avg(Income) = 195.71

Shop income = 186.67
Why AQP?

99.9% Identical
100x-200x Faster

Answer by **Impala** in 350 seconds

Answer by **VerdictDB** in 2 seconds
Sampling Based AQP

Uniform (Random) Sampling

Original Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Advertiser</th>
<th>Geo</th>
<th>Bid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>adv10</td>
<td>NY</td>
<td>0.0001</td>
</tr>
<tr>
<td>2</td>
<td>adv10</td>
<td>VT</td>
<td>0.0005</td>
</tr>
<tr>
<td>3</td>
<td>adv20</td>
<td>NY</td>
<td>0.0002</td>
</tr>
<tr>
<td>4</td>
<td>adv10</td>
<td>NY</td>
<td>0.0003</td>
</tr>
<tr>
<td>5</td>
<td>adv20</td>
<td>NY</td>
<td>0.0001</td>
</tr>
<tr>
<td>6</td>
<td>adv30</td>
<td>VT</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Uniform Sample

<table>
<thead>
<tr>
<th>ID</th>
<th>Advertiser</th>
<th>Geo</th>
<th>Bid</th>
<th>Sampling Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>adv20</td>
<td>NY</td>
<td>0.0002</td>
<td>1/3</td>
</tr>
<tr>
<td>5</td>
<td>adv20</td>
<td>NY</td>
<td>0.0001</td>
<td>1/3</td>
</tr>
</tbody>
</table>

```
SELECT avg(bid)
FROM AdImpresssions
WHERE geo = 'VT'
```
Sampling Based AQP

Stratified Sampling

**Original Table**

<table>
<thead>
<tr>
<th>ID</th>
<th>Advertiser</th>
<th>Geo</th>
<th>Bid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>adv10</td>
<td>NY</td>
<td>0.0001</td>
</tr>
<tr>
<td>2</td>
<td>adv10</td>
<td>VT</td>
<td>0.0005</td>
</tr>
<tr>
<td>3</td>
<td>adv20</td>
<td>NY</td>
<td>0.0002</td>
</tr>
<tr>
<td>4</td>
<td>adv10</td>
<td>NY</td>
<td>0.0003</td>
</tr>
<tr>
<td>5</td>
<td>adv20</td>
<td>NY</td>
<td>0.0001</td>
</tr>
<tr>
<td>6</td>
<td>adv30</td>
<td>VT</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

**Stratified Sample on Geo**

<table>
<thead>
<tr>
<th>ID</th>
<th>Advertiser</th>
<th>Geo</th>
<th>Bid</th>
<th>Sampling Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>adv20</td>
<td>NY</td>
<td>0.0002</td>
<td>1/4</td>
</tr>
<tr>
<td>2</td>
<td>adv10</td>
<td>VT</td>
<td>0.0005</td>
<td>1/2</td>
</tr>
</tbody>
</table>

**Query Column Set (QCS)**

```sql
SELECT avg(bid)
FROM AdImpresssions
WHERE geo = 'VT'
```
# Why SnappyData & VerdictDB?

- Spark
- Open-source*

<table>
<thead>
<tr>
<th>Name</th>
<th>Online/Offline</th>
<th>Distributed/Standalone</th>
<th>Platform</th>
<th>Algorithm</th>
<th>Skewed</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlinkDB</td>
<td>Offline</td>
<td>Distributed</td>
<td>Hive/Hadoop (Shark)</td>
<td>Stratified sampling</td>
<td>Yes</td>
</tr>
<tr>
<td>Sapprox</td>
<td>Online</td>
<td>Distributed</td>
<td>Hadoop</td>
<td>Distribution-aware Online sampling</td>
<td>No</td>
</tr>
<tr>
<td>Approxhadoop</td>
<td>Online</td>
<td>Distributed</td>
<td>Hadoop</td>
<td>Approximation-enabled MapReduce</td>
<td>No</td>
</tr>
<tr>
<td>Quickr</td>
<td>Online</td>
<td>Distributed</td>
<td>N/A</td>
<td>ASALQA algorithm</td>
<td>No</td>
</tr>
<tr>
<td><strong>SnappyData</strong></td>
<td><strong>Online</strong></td>
<td><strong>Distributed</strong></td>
<td><strong>Spark and GemFire</strong></td>
<td><strong>Spark as a computational engine; GemFire as transactional store</strong></td>
<td>No</td>
</tr>
<tr>
<td>FluoDB</td>
<td>Online</td>
<td>Distributed</td>
<td>Spark</td>
<td>Mini-batch execution OLA Model</td>
<td>No</td>
</tr>
<tr>
<td>XDB</td>
<td>Online</td>
<td>Standalone</td>
<td>PostgreSQL</td>
<td>Wander join</td>
<td>No</td>
</tr>
<tr>
<td><strong>VerdictDB</strong></td>
<td><strong>Online</strong></td>
<td><strong>Standalone</strong></td>
<td><strong>Spark SQL</strong></td>
<td><strong>Database learning</strong></td>
<td>No</td>
</tr>
<tr>
<td>IDEA</td>
<td>Online</td>
<td>Standalone</td>
<td>N/A</td>
<td>Reuse answers of past overlapping queries for new query</td>
<td>No</td>
</tr>
<tr>
<td>BEAS</td>
<td>Online</td>
<td>Standalone</td>
<td>Commercial DBMS</td>
<td>Approximability theorem</td>
<td>No</td>
</tr>
<tr>
<td>ABS</td>
<td>Online</td>
<td>Standalone</td>
<td>N/A</td>
<td>Bootstrap</td>
<td>No</td>
</tr>
</tbody>
</table>
SnappyData

Spark Jobs, Scala/Java/Python/R API, JDBC/ODBC, Object API (RDD, DataSets)

Spark API
(Streaming, ML, Graph)

Data Frame, RDD, DataSets

Transactions, Indexing

Full SQL

HA

SDE is NOT open source

SnappyData

IN-MEMORY

Spark Cache

Unified Data Access
(Virtual Tables)

Unified Catalog

Shared Nothing Disk Persist

HDFS/HBASE

S3

JSON, CSV, XML

SQL db

Cassandra

MPP DB

Stream sources
SnappyData

+ WITH ERROR

Sample Creation

Sample Selection

RESULTS

QCS

FRACTION
VerdictDB

user/app

SQL

ResultSet

VERDICTDB

SQL

ResultSet

database
Figure 1: VERDICTDB’s offline and online workflow: sample preparation (in gray) and query processing (in green).
Experiment Setup

- Cluster Setup
  - SnappyData: 1 locator, 1 lead, and 2 servers
Experiment Setup

- Cluster Setup
  - SnappyData: 1 locator, 1 lead, and 2 servers
  - VerdictDB on Spark: 1 master and 2 executors

- Each Node
  - 24/32 GB memory used
  - 500 GB HDD
Experiment Setup

- TPC-H Benchmark
  - OLAP
  - 22 queries includes Aggregation, Join, etc.
  - Well known and standard
  - Customizable

- Data
  - 1GB and 10GB
  - Uniformly distributed
Evaluation

SnappyData
- Stratified Sampling
- In-memory

VerdictDB
- Uniform Sampling
- Not in-memory (bug?)
**SnappyData - Latency**

Execution time (ms) using TPC-H (SF=10, fraction 0.1)

Q1: Up to **3.6X** speedup  
~0.0001 Error

![Graph showing execution time for TPC-H queries with SnappyData and SnappyData_AQP](image)

- **Q1**: Up to **3.6X** speedup with ~0.0001 Error
- **Q6**: SnappyData: 1832 ms, SnappyData_AQP: 1399 ms
- **Q14**: SnappyData: 6629 ms, SnappyData_AQP: 3870 ms
SnappyData - Accuracy

Actual Error for TPC-H Q14 result (SF=10) given different sample tables (fraction)

- fraction 0.01
- fraction 0.1
- fraction 0.2
- fraction 0.3

Base Table
Sample Tables ...

Time (ms) for TPC-H Q14 result (SF=10) given different sample tables (fraction)

- fraction 0.01
- fraction 0.1
- fraction 0.2
- fraction 0.3
- Snappy
SnappyData- Creating Sample Tables

Time (ms) for creating SnappyData sample tables with different fractions

- fraction 0.01
- fraction 0.1
- fraction 0.2
- fraction 0.3
VerdictDB - Latency

Execution time (ms) using TPC-H (SF=10, fraction 0.1)

Up to ~11X speedup!
VerdictDB - Speedup

Speedup for TPC-H (SF=1, fraction=0.1)

- Q1
- Q6
- Q14

Speedup for TPC-H (SF=10, fraction=0.1)

- Q1
- Q6
- Q14
VerdictDB - Creating Sample Tables

Time (ms) for creating VerdictDB sample tables with different fraction

- Fraction 0.01
- Fraction 0.1
- Fraction 0.2
- Fraction 0.3
**VerdictDB - Accuracy**

Actual Error for TPC-H Q14 result (SF=10) given different sample tables (fraction)

- **Base Table**
- **Sample Tables** ...

**Time (ms) for TPC-H Q14 result (SF=10) given different sample tables (fraction)**

- converge!
Other Queries?

Q14
Error: ~ 1.7%
Speedup: ~1.7X

```sql
select
    100.00 * sum(case
        when p_type like 'PROMO%'
        then l_extendedprice * (1 - l_discount)
        else 0
    end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
    and l_shipdate >= date '1993-10-01'
    and l_shipdate < date '1993-10-01' + interval '1' month;
```

Q19
Error: ~ 80%
Speedup: ~5.5X

```sql
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#32'
    and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
    and l_quantity >= 7 and l_quantity <= 7 + 10
    and p_size between 1 and 5
    and l_shipmode in ('AIR', 'AIR REG')
    and l_shipinstruct = 'DELIVER IN PERSON';
```
Other Queries?

```
select
  supp_nation,
  cust_nation,
  l_year,
  sum(volume) as revenue
from
  (
    select
      n1.n_name as supp_nation,
      n2.n_name as cust_nation,
      year(l_shipdate) as l_year,
      l_extendedprice * (1 - l_discount) as volume
    from
      supplier,
      lineitem,
      orders,
      customer,
      nation n1,
      nation n2
    where
      s_suppkey = l_suppkey
      and o_orderkey = l_orderkey
      and c_custkey = o_custkey
      and s_nationkey = n1.n_nationkey
      and c_nationkey = n2.n_nationkey
      and (n1.n_name = 'GERMANY' and n2.n_name = 'KENYA')
      or (n1.n_name = 'KENYA' and n2.n_name = 'GERMANY')
      and l_shipdate between date '1995-01-01' and date '1996-12-31'
  ) as shipping
group by
  supp_nation,
  cust_nation,
  l_year
order by
  supp_nation,
  cust_nation,
  l_year;
```

Key missing in sample tables!

Careful design of sample table or original table!

Q7

AQP not working!
Insights

- AQP performs well:
  - For aggregate functions such as SUM, AVG and COUNT
  - When WHERE is simple
- Users’ foreseen is important!
  - for both query and original table
Future Work

- Test error estimation in sampling
- Other sampling techniques
  - Biased Sampling
- Database learning
- Approximate hardware