Pinot: Realtime OLAP for 530 Million Users

Presented By: Sushant Raikar
Challenge

100 QPS  ->  100000 QPS
Pinot

- Single system for all OLAP needs
- Ability to serve up to 50000 QPS
- Flexible
- Near-Realtime data
History

RDBMS
History

RDBMS

Cubing

Offline
# Data in Pinot

<table>
<thead>
<tr>
<th>Doc Id</th>
<th>Model</th>
<th>Year</th>
<th>Price</th>
<th>tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Toyota</td>
<td>1997</td>
<td>7500</td>
<td>moon-roof, reliable</td>
</tr>
<tr>
<td>2</td>
<td>Toyota</td>
<td>2001</td>
<td>7500</td>
<td>mp3, compact</td>
</tr>
<tr>
<td>3</td>
<td>Mazda</td>
<td>2005</td>
<td>1500</td>
<td>Manual, compact</td>
</tr>
<tr>
<td>4</td>
<td>Honda</td>
<td>2010</td>
<td>8000</td>
<td>leather, cool</td>
</tr>
<tr>
<td>5</td>
<td>Honda</td>
<td>2011</td>
<td>9000</td>
<td>automatic</td>
</tr>
<tr>
<td>10000</td>
<td>bmw</td>
<td></td>
<td>25000</td>
<td>moon-roof, reliable</td>
</tr>
</tbody>
</table>

**RAW DATA**

**INDEX GENERATION**

<table>
<thead>
<tr>
<th>Doc Id</th>
<th>Model</th>
<th>Year</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Toyota</td>
<td>1997</td>
<td>7500</td>
</tr>
<tr>
<td>2</td>
<td>Toyota</td>
<td>2001</td>
<td>7500</td>
</tr>
<tr>
<td>3</td>
<td>Mazda</td>
<td>2005</td>
<td>1500</td>
</tr>
<tr>
<td>4</td>
<td>Honda</td>
<td>2010</td>
<td>8000</td>
</tr>
<tr>
<td>5</td>
<td>Honda</td>
<td>2011</td>
<td>9000</td>
</tr>
<tr>
<td>10000</td>
<td>bmw</td>
<td></td>
<td>25000</td>
</tr>
</tbody>
</table>

**COLUMNAR DATA**
Architecture

Controller

Object Store
Architecture

- Servers
- Controller
- Object Store
Architecture

- Brokers
- Servers
- Controller
- Object Store
- Queries
Uploading Data

1. HTTP POST Data Segment to Controller
2. Controller uploads it to persistent object store
3. Controller asks servers to download segment
4. Server downloads data from Object Store
5. Tells Controller that it finished downloading
6. Brokers update routing Tables
Running Queries

1. Queries arrive at broker get analyzed
2. Routing table is picked
3. Multiple servers are contacted
4. Servers generate logical and physical query plans and execute the query.
5. Broker merges data

Routing is important, especially for large clusters

- Multiple routing tables for same table
- Randomly select one routing table for a query
- Load balancing
Hybrid Queries

To handle near real time queries

- "Online" Server
  - Consumes from Kafka
  - Stops based on time or number
  - Contacts controller for consensus
  - Forms indexes similar to offline

- Broker splits query
  - Based on Time
  - Into offline and online
  - Merges result

```sql
SELECT SUM(foo)
FROM T
WHERE date > 'Jul 31'
```

```sql
SELECT SUM(foo) WHERE date >= 'Aug 2'
```

```sql
SELECT SUM(foo) WHERE date > 'Jul 31' AND date < 'Aug 2'
```

```sql
SELECT SUM(foo) FROM T
WHERE date > 'Jul 31'
```
Optimization: Star Tree Index

<table>
<thead>
<tr>
<th>docid</th>
<th>country</th>
<th>browser</th>
<th>... other dimensions</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>al</td>
<td>ie</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>ca</td>
<td>safari</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>us</td>
<td>chrome</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>us</td>
<td>chrome</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>us</td>
<td>ie</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>N</td>
<td>us</td>
<td>safari</td>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>

Raw records

1. Multidimensional sort
2. Split on the column and create a node for each value (+raw doc range)
3. Create star node (contains aggregated docs after removing the split column)
4. Apply 1,2,3 for each node recursively until the number of records in node < SplitThreshold
Optimization: Star Tree Query

```
select sum(Views) from T
where country = CA
```

```
select sum(X)
from T
where d1 = v1 and d2 = v2 and ...
```

Any query pattern will scan less than $SplitThreshold$ records.
**Optimization: Star Tree Query**

```
select sum(X) from T
where d1=v1 and d2=v2 and ...
```

Any query pattern will scan less than *SplitThreshold* records
Results

- Single infrastructure for all OLAP needs
- Support for user facing analytical queries
  - 50000 Queries per Second
- Query Flexibility
  - High throughput, low complexity
  - Low throughput, complex queries
- Pluggable Indexing
  - Star Tree Index
Discussion

- Limitations of Pinot?
- How does Pinot differ from in-memory OLAP like Hyper or Hekaton?
- Pinot is becoming Apache, with support from companies like Uber, Slack. What does LinkedIn gain or lose from this move?
- Is Pinot basically Druid with better optimizations?
- Comments about the paper