Phoenix

We put the SQL back in NoSQL

James Taylor
jtaylor@salesforce.com

Your success. Our cloud.

salesforce.com
In the dawn of time...
Relational Databases were invented
But we all know the problems folks ran into
And then there was HBase
And it was good

1. Horizontally scalable
And it was good

1. Horizontally scalable
2. Maintains data locality
And it was good

1. Horizontally scalable
2. Maintains data locality
3. Runs on commodity hardware

Leverage the NoSQL boom
But somewhere, something terrible went wrong.
But somewhere, something terrible went wrong

1. It takes too much expertise to write an application
But somewhere, something terrible went wrong

1. It takes too much expertise to write an application
2. It takes too much code to do anything
But somewhere, something terrible went wrong

1. It takes too much expertise to write an application
2. It takes too much code to do anything
3. Your application is tied too closely with your data model
What is Phoenix?

- SQL skin for HBase
What is Phoenix?

- SQL skin for HBase
- An alternate client API
What is Phoenix?

- SQL skin for HBase
- An alternate client API
- An embedded JDBC driver that allows you to run at HBase native speed
What is Phoenix?

- SQL skin for HBase
- An alternate client API
- An embedded JDBC driver that allows you to run at HBase native speed
- Compiles your SQL into native HBase calls
What is Phoenix?

- SQL skin for HBase
- An alternate client API
- An embedded JDBC driver that allows you to run at HBase native speed
- Compiles your SQL into native HBase calls

so you don’t have to!
Phoenix Performance

- Phoenix (key filter)
- Phoenix (full table)
- Hive over HBase

Duration (sec) vs. Table Row Count

- 10M to 100M rows

Graph showing performance comparison.
Why SQL for HBase?

- Broaden HBase adoption
  - Give folks an API they already know
Why SQL for HBase?

- Broaden HBase adoption
  - Give folks an API they already know
- Reduce the amount of code users need to write
  ```sql
  SELECT TRUNC(date,'DAY'), AVG(cpu_usage)
  FROM web_stat
  WHERE domain LIKE 'Salesforce%'
  GROUP BY TRUNC(date,'DAY')
  ```
Why SQL for HBase?

- Broaden HBase adoption
  - Give folks an API they already know
- Reduce the amount of code users need to write
  
  ```sql
  SELECT TRUNC(date,'DAY'), AVG(cpu_usage)
  FROM web_stat
  WHERE domain LIKE 'Salesforce%'
  GROUP BY TRUNC(date,'DAY')
  ```
- Performance optimizations transparent to the user
  - Aggregation
  - Skip Scan
  - Secondary indexing (soon!)
Why SQL for HBase?

- Broaden HBase adoption
  - Give folks an API they already know
- Reduce the amount of code users need to write
  
  ```
  SELECT TRUNC(date,'DAY'), AVG(cpu_usage)
  FROM web_stat
  WHERE domain LIKE 'Salesforce%'
  GROUP BY TRUNC(date,'DAY')
  ```
- Performance optimizations transparent to the user
  - Aggregation
  - Skip Scan
  - Secondary indexing (soon!)
- Leverage existing tooling
  - SQL client/terminal
  - OLAP engine
Query Processing

Over metrics data for clusters of servers with a schema like this:

<table>
<thead>
<tr>
<th>Server Metrics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>RESPONSE_TIME</td>
<td>INTEGER</td>
</tr>
<tr>
<td>GC_TIME</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CPU_TIME</td>
<td>INTEGER</td>
</tr>
<tr>
<td>IO_TIME</td>
<td>INTEGER</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Query Processing

With data that looks like this:

<table>
<thead>
<tr>
<th>HOST</th>
<th>DATE</th>
<th>RESPONSE_TIME</th>
<th>GC_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ny1-s1</td>
<td>Jun 5 10:10:10.234</td>
<td>1234</td>
<td></td>
</tr>
<tr>
<td>ny1-s1</td>
<td>Jun 5 11:18:28.456</td>
<td>4560</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sfl-s1</td>
<td>Jun 5 10:10:10.234</td>
<td>2345</td>
<td></td>
</tr>
<tr>
<td>sfl-s1</td>
<td>Jun 6 12:46:19.123</td>
<td>1003</td>
<td></td>
</tr>
<tr>
<td>sf7-s20</td>
<td>Jun 4 08:23:23.456</td>
<td>5002</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scenario 1
Chart Response Time Per Cluster

SELECT host, trunc(date,'HOUR'),
    min(response_time), max(response_time)
FROM server_metrics
WHERE date > CURRENT_DATE() – 1
AND substr(host, 1, 3) IN ('sf1', 'sf3', 'sf7')
GROUP BY substr(host, 1, 3), trunc(date,'HOUR')
Phoenix Roadmap

- Secondary Indexing
- Hash Joins
- Apache Drill integration
- Count distinct and percentile
- Derived tables
  - SELECT * FROM (SELECT * FROM t)
- Cost-based query optimizer
- OLAP extensions
  - WINDOW, PARTITION OVER, RANK
- Monitoring and management
- Transactions
Thank you!
Questions/comments?