BlinkDB:
Query Petabytes of Data in a Blink Time!

Barzan Mozafari
University of Michigan, Ann Arbor
Collaborators

Sameer Agarwal
Aurojit Panda
Henry Milner
Ion Stoica
Samuel Madden
My Research ...

Using statistics to build better data-intensive systems

1. More predictable
   - How to predict resources in a DB?
   - How to design a more predictable DB?

2. More scalable
   - How to scale crowdsourcing?
   - How to query petabytes of data in seconds?
Big Data
Online Media Websites
Real-time Ad-performance, Spam Detection
Big Data

Log Processing

Root-cause Analysis, A/B Testing
Overview

**Problem:** Need to compute aggregate statistics over massive sets of data

**Our Goal:** Support *interactive* ad-hoc analytical queries over these *large datasets*
100 TB on 1000 machines

1-2 Hours  →  25-30 Minutes  →  1 second

Hard Disks  →  Memory  →  ?
Target Workload

1. **Real-time latency** is valued over perfect accuracy

“On a good day, I can run up to 6 queries in Hive.”
- Anonymous Data Scientist at Facebook.
Target Workload

1. Real-time latency is valued over perfect accuracy: ≤ 10 sec for interactive experience

“On a good day, I can run up to 6 queries in Hive.”
- Anonymous Data Scientist at Facebook
Target Workload

1. **Real-time latency** is valued over perfect accuracy: \( \leq 10 \text{ sec for interactive experience} \)

2. Exploration is **ad-hoc**

3. Columns queried together (i.e., **Templates**) are **stable** over time
1. **Real-time latency** is valued over perfect accuracy: \( \leq 10 \) sec for interactive experience.

Exploration is ad-hoc.

Columns queried together (i.e., Templates) are stable over time.

<table>
<thead>
<tr>
<th>Queries</th>
<th>Templates</th>
</tr>
</thead>
<tbody>
<tr>
<td>68,785</td>
<td>( \approx ) 211</td>
</tr>
<tr>
<td>90%</td>
<td>( \approx ) 20%</td>
</tr>
</tbody>
</table>

Target Workload

Facebook Queries (1 week)
Target Workload

1. Real-time latency is valued over perfect accuracy: \( \leq 10 \) sec for interactive experience

Exploration is ad-hoc

Columns queried together (i.e., Templates) are stable over time

17,437 Queries \( \approx \) 108 Templates

90% Queries \( \approx \) 10% Templates
Target Workload

1. **Real-time latency** is valued over perfect accuracy: \( \leq 10 \) sec for interactive experience

2. Exploration is **ad-hoc**

3. Columns queried together (i.e., **Templates**) are **stable** over time

4. User defined functions (**UDF**) must be supported: **43.6% of Conviva’s queries**

5. Data is **high-dimensional & skewed**: \(+100\) columns
100 TB on 1000 machines

1-2 Hours  →  25-30 Minutes  →  1 second

Hard Disks  →  Memory  →  ?

One can often make perfect decision without perfect answers

Approximation using Offline Samples
BlinkDB Interface

```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
WITHIN 1 SECONDS
234.23 ± 15.32
```
BlinkDB Interface

```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
WITHIN 2 SECONDS
```

239.46 ± 4.96

```
SELECT avg(sessionTime)
FROM Table
WHERE city='San Francisco'
```

ERROR 0.1 CONFIDENCE 95.0%

234.23 ± 15.32

239.46 ± 4.96
BlinkDB Architecture

Offline sampling:
» Uniform
» Stratified on different sets of columns
» Different sizes

TABLE
Original Data

Sampling Module

On-Disk Samples
In-Memory Samples
BlinkDB Architecture

Predict time and error of the query for each sample type

TABLE
Original Data

Sampling Module

SELECT \textit{foo} (*)
FROM TABLE
IN TIME 2 SECONDS

Query Plan
Sample Selection

On-Disk Samples
In-Memory Samples
BlinkDB Architecture

SELECT $foo$ (*)
FROM TABLE
IN TIME 2 SECONDS

New Query Plan
Sample Selection

Parallel execution

Error Bars & Confidence Intervals

Result
182.23 ± 5.56
(95% confidence)
Three Key Sets of Challenges

1. How to accurately estimate the error?
   - What about UDFs? (43.6% of Conviva queries)
   - What if the error estimate itself is wrong?

2. Given a storage budget, which samples to build & maintain to support a wide range of ad-hoc exploratory queries?

3. Given a query, what should be the optimal sample type and size that can be processed to meet its constraints?
Closed-Form Error Estimates

Central Limit Theorem (CLT)

1. Count: $N(np, n(1-p)p)$
2. Sum: $N(n p \mu, n p (\sigma^2 + (1-p) \mu^2))$
3. Mean: $N(\mu, \sigma^2 / n)$
4. Variance: $N(\sigma^2, (\mu_4 - \sigma^4)/n)$
5. Stddev: $N(\sigma, (\mu_4 - \sigma^4)/(4\sigma^2 n))$

What about more complex queries?

- UDFs, nested queries, joins, ...
Bootstrap [Efron 1979]

Quantify accuracy of a sample estimator $f()$

Distribution $X \rightarrow f(X)$

random sample

$|S| = N \rightarrow f(S)$

what is $f(S)$'s error?

sampling with replacement

$S_1 \rightarrow f(S_1)$

$S_k \rightarrow f(S_k)$

$|S_i| = N$

• estimator: mean($f(S_i)$)
• error, e.g.: stdev($f(S_i)$)

can’t compute $f(X)$ as we don’t have $X$
Bootstrap

Quantify accuracy of a query on a sample table

Original Table

\[ T \]

\[ Q(T) \]

\[ Q(T) \] takes too long!

\[ S \] = \[ N \]

Sampling with replacement

\[ |S_i| = N \]

\[ S_1 \]

\[ \ldots \]

\[ S_k \]

\[ \ldots \]

\[ \ldots \]

\[ Q(S_1) \]

\[ Q(S_k) \]

\[ Q(S) \]

what is \[ Q(S) \]’s error?

\[ \text{estimator: } mean(f(S_i)) \]

\[ \text{error, e.g.: } stdev(f(S_i)) \]
Bootstrap

1. Bootstrap treats Q as a **black-box**
   - Can handle (almost) arbitrarily complex queries including UDFs!

2. **Embarrassingly Parallel**

![Diagram](image)

- Uses too many resources in the cluster
Error Estimation

1. CLT-based closed forms:
   - Fast but limited to simple aggregates

2. Bootstrap (Monte Carlo simulation):
   - Expensive but general

3. Analytical Bootstrap Method (ABM):
   - Fast and general
     - (some restrictions, e.g. no UDF, some self joins, ...)

Analytical Bootstrap Method*

Key Idea:

1. Annotate tuples w/ integer random variables
   - Probabilistic Multiset Database

2. Extend relational operators to manipulate these random variables

3. Use a single execution to estimate the empirical distribution

* The Analytical Bootstrap: A New Method for Fast Error Estimation in Approximate Query Processing, K. Zeng, G. Shi, B. Mozafari, C. Zaniolo, under submission
TPC-H Experiment

ABM is 2-4 orders of magnitude faster than simulation-based implementations of bootstrap
Three Key Sets of Challenges

1. How to accurately estimate the error?
   - What about UDFs? (43.6% of Conviva queries)
   - What if the error estimate itself is wrong?

2. Given a storage budget, which samples to build & maintain to support a wide range of ad-hoc exploratory queries?

3. Given a query, what should be the optimal sample type and size that can be processed to meet its constraints?
Problem with Uniform Samples

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Age</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYC</td>
<td>22</td>
<td>50,000</td>
</tr>
<tr>
<td>2</td>
<td>Ann Arbor</td>
<td>25</td>
<td>120,242</td>
</tr>
<tr>
<td>3</td>
<td>NYC</td>
<td>25</td>
<td>78,212</td>
</tr>
<tr>
<td>4</td>
<td>NYC</td>
<td>67</td>
<td>62,492</td>
</tr>
<tr>
<td>5</td>
<td>NYC</td>
<td>34</td>
<td>98,341</td>
</tr>
<tr>
<td>6</td>
<td>Ann Arbor</td>
<td>62</td>
<td>78,453</td>
</tr>
</tbody>
</table>

SELECT avg(salary) FROM table WHERE city = ‘Ann Arbor’
Problem with Uniform Samples

Larger Uniform Sample

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Age</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYC</td>
<td>22</td>
<td>50,000</td>
</tr>
<tr>
<td>2</td>
<td>Ann Arbor</td>
<td>25</td>
<td>120,242</td>
</tr>
<tr>
<td>3</td>
<td>NYC</td>
<td>25</td>
<td>78,212</td>
</tr>
<tr>
<td>4</td>
<td>NYC</td>
<td>67</td>
<td>62,492</td>
</tr>
<tr>
<td>5</td>
<td>NYC</td>
<td>34</td>
<td>98,341</td>
</tr>
<tr>
<td>6</td>
<td>Ann Arbor</td>
<td>62</td>
<td>78,453</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Age</th>
<th>Salary</th>
<th>Sampling Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>NYC</td>
<td>25</td>
<td>78,212</td>
<td>2/3</td>
</tr>
<tr>
<td>5</td>
<td>NYC</td>
<td>34</td>
<td>98,341</td>
<td>2/3</td>
</tr>
<tr>
<td>1</td>
<td>NYC</td>
<td>22</td>
<td>50,000</td>
<td>2/3</td>
</tr>
<tr>
<td>2</td>
<td>Ann Arbor</td>
<td>25</td>
<td>120,242</td>
<td>2/3</td>
</tr>
</tbody>
</table>

```
SELECT avg(salary) FROM table WHERE city = 'Ann Arbor'
```
Stratified Samples

Stratified Sample on City

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Age</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYC</td>
<td>22</td>
<td>50,000</td>
</tr>
<tr>
<td>2</td>
<td>Ann Arbor</td>
<td>25</td>
<td>120,242</td>
</tr>
<tr>
<td>3</td>
<td>NYC</td>
<td>25</td>
<td>78,212</td>
</tr>
<tr>
<td>4</td>
<td>NYC</td>
<td>67</td>
<td>62,492</td>
</tr>
<tr>
<td>5</td>
<td>NYC</td>
<td>34</td>
<td>98,341</td>
</tr>
<tr>
<td>6</td>
<td>Ann Arbor</td>
<td>62</td>
<td>78,453</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Age</th>
<th>Salary</th>
<th>Sampling Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>NYC</td>
<td>67</td>
<td>62,492</td>
<td>1/4</td>
</tr>
<tr>
<td>5</td>
<td>Ann Arbor</td>
<td>25</td>
<td>120,242</td>
<td>1/2</td>
</tr>
</tbody>
</table>

SELECT avg(salary) FROM table WHERE city = ‘Ann Arbor’ AND age > 60
Target Workload

1. Real-time latency is valued over perfect accuracy: ≤ 10 sec for interactive experience

2. Exploration is **ad-hoc**

3. Columns queried together (i.e., Templates) are **stable** over time

4. User defined functions (UDF) must be supported: 43.6% of Conviva’s queries

5. Data is **high-dimensional & skewed**: +100 columns
Which Stratified Samples to Build?

For \( n \) columns, \( 2^n \) possible stratified samples

Modern data warehouses: \( n \approx 100-200 \)

**Our solution**: Choosing the best set of samples by considering

1. Columns queried together
2. Data distribution
3. Storage costs
## Optimal Set of Samples

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Age</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYC</td>
<td>25</td>
<td>50,000</td>
</tr>
<tr>
<td>2</td>
<td>NYC</td>
<td>35</td>
<td>62,492</td>
</tr>
<tr>
<td>3</td>
<td>Ann Arbor</td>
<td>35</td>
<td>78,212</td>
</tr>
<tr>
<td>4</td>
<td>NYC</td>
<td>25</td>
<td>120,242</td>
</tr>
<tr>
<td>5</td>
<td>NYC</td>
<td>35</td>
<td>98,341</td>
</tr>
<tr>
<td>6</td>
<td>Berkeley</td>
<td>25</td>
<td>75,453</td>
</tr>
<tr>
<td>7</td>
<td>NYC</td>
<td>25</td>
<td>60,000</td>
</tr>
<tr>
<td>8</td>
<td>NYC</td>
<td>35</td>
<td>72,492</td>
</tr>
<tr>
<td>9</td>
<td>Berkeley</td>
<td>45</td>
<td>88,212</td>
</tr>
<tr>
<td>10</td>
<td>Berkeley</td>
<td>35</td>
<td>92,242</td>
</tr>
<tr>
<td>11</td>
<td>NYC</td>
<td>35</td>
<td>70,000</td>
</tr>
<tr>
<td>12</td>
<td>Ann Arbor</td>
<td>45</td>
<td>102,492</td>
</tr>
</tbody>
</table>

- [City]  
- [Age]  
- [Salary]  
- [City, Age]  
- [Age, Salary]  
- [City, Salary]  
- [City, Age, Salary]
Query Coverage

\[
\text{SELECT } \text{AVG (\ldots)} \\
\text{FROM Table} \\
\text{WHERE } \text{Age} = x
\]
Query Coverage

SELECT AVG (...) FROM Table WHERE Age = x

[City] 0%
[Age] 100%
[Salary] 0%
[City, Age] 100%
[Age, Salary] 100%
[City, Salary] 0%
[City, Age, Salary] 100%
Query Coverage

SELECT AVG (...) 
FROM Table 
WHERE Age = x AND 
  City = z
Query Coverage

SELECT AVG (...) 
FROM Table
WHERE Age = x  AND City = z
SELECT AVG (...)
FROM Table
WHERE Age = x AND City = z
**Query Coverage**

\[
\forall j: y_j \leq \max_{i: \phi_i \subseteq q_j' \cup i: \phi_i \supseteq q_j} (z_i \min 1, \frac{|D(\phi_i)|}{|D(q_j)|})
\]

**SELECT** AVG (...)  
**FROM** Table  
**WHERE** Age = x AND City = z

<table>
<thead>
<tr>
<th>[City]</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Age]</td>
<td>83%</td>
</tr>
<tr>
<td>[Salary]</td>
<td>0%</td>
</tr>
<tr>
<td>[City, Age]</td>
<td>100%</td>
</tr>
<tr>
<td>[Age, Salary]</td>
<td>100%</td>
</tr>
<tr>
<td>[City, Salary]</td>
<td>100%</td>
</tr>
<tr>
<td>[City, Age, Salary]</td>
<td>100%</td>
</tr>
</tbody>
</table>
### Cost of Stratification

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Age</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYC</td>
<td>25</td>
<td>50,000</td>
</tr>
<tr>
<td>2</td>
<td>NYC</td>
<td>25</td>
<td>80,000</td>
</tr>
<tr>
<td>3</td>
<td>Ann Arbor</td>
<td>35</td>
<td>80,000</td>
</tr>
<tr>
<td>4</td>
<td>NYC</td>
<td>25</td>
<td>120,000</td>
</tr>
<tr>
<td>5</td>
<td>NYC</td>
<td>25</td>
<td>80,000</td>
</tr>
<tr>
<td>6</td>
<td>Berkeley</td>
<td>25</td>
<td>80,000</td>
</tr>
<tr>
<td>7</td>
<td>NYC</td>
<td>25</td>
<td>60,000</td>
</tr>
<tr>
<td>8</td>
<td>NYC</td>
<td>25</td>
<td>70,000</td>
</tr>
<tr>
<td>9</td>
<td>Berkeley</td>
<td>30</td>
<td>80,000</td>
</tr>
<tr>
<td>10</td>
<td>Berkeley</td>
<td>25</td>
<td>90,000</td>
</tr>
<tr>
<td>11</td>
<td>NYC</td>
<td>40</td>
<td>80,000</td>
</tr>
<tr>
<td>12</td>
<td>Ann Arbor</td>
<td>45</td>
<td>100,000</td>
</tr>
</tbody>
</table>

### Stratified Sample on [City]

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Age</th>
<th>Salary</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYC</td>
<td>25</td>
<td>50,000</td>
<td>2/7</td>
</tr>
<tr>
<td>8</td>
<td>NYC</td>
<td>35</td>
<td>70,000</td>
<td>2/7</td>
</tr>
<tr>
<td>6</td>
<td>Berkeley</td>
<td>25</td>
<td>80,000</td>
<td>2/3</td>
</tr>
<tr>
<td>10</td>
<td>Berkeley</td>
<td>25</td>
<td>90,000</td>
<td>2/3</td>
</tr>
<tr>
<td>3</td>
<td>Ann Arbor</td>
<td>35</td>
<td>80,000</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Ann Arbor</td>
<td>45</td>
<td>100,000</td>
<td>1</td>
</tr>
</tbody>
</table>

**Cost = 6**
## Cost of Stratification

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Age</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYC</td>
<td>25</td>
<td>50,000</td>
</tr>
<tr>
<td>2</td>
<td>NYC</td>
<td>25</td>
<td>80,000</td>
</tr>
<tr>
<td>3</td>
<td>Ann Arbor</td>
<td>35</td>
<td>80,000</td>
</tr>
<tr>
<td>4</td>
<td>NYC</td>
<td>25</td>
<td>120,000</td>
</tr>
<tr>
<td>5</td>
<td>NYC</td>
<td>25</td>
<td>80,000</td>
</tr>
<tr>
<td>6</td>
<td>Berkeley</td>
<td>25</td>
<td>80,000</td>
</tr>
<tr>
<td>7</td>
<td>NYC</td>
<td>25</td>
<td>60,000</td>
</tr>
<tr>
<td>8</td>
<td>NYC</td>
<td>25</td>
<td>70,000</td>
</tr>
<tr>
<td>9</td>
<td>Berkeley</td>
<td>30</td>
<td>80,000</td>
</tr>
<tr>
<td>10</td>
<td>Berkeley</td>
<td>25</td>
<td>90,000</td>
</tr>
<tr>
<td>11</td>
<td>NYC</td>
<td>40</td>
<td>80,000</td>
</tr>
<tr>
<td>12</td>
<td>Ann Arbor</td>
<td>45</td>
<td>100,000</td>
</tr>
</tbody>
</table>

### Stratified Sample on [Salary]

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
<th>Age</th>
<th>Salary</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYC</td>
<td>25</td>
<td>50,000</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>NYC</td>
<td>25</td>
<td>60,000</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>NYC</td>
<td>25</td>
<td>70,000</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Ann Arbor</td>
<td>35</td>
<td>80,000</td>
<td>1/3</td>
</tr>
<tr>
<td>9</td>
<td>Berkeley</td>
<td>30</td>
<td>80,000</td>
<td>1/3</td>
</tr>
<tr>
<td>10</td>
<td>Berkeley</td>
<td>25</td>
<td>90,000</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Ann Arbor</td>
<td>45</td>
<td>100,000</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>NYC</td>
<td>25</td>
<td>120,000</td>
<td>1</td>
</tr>
</tbody>
</table>

**Cost = 8**
MILP Formulation

Maximize

\[ G = \sum_{j} p_j \cdot y_j \cdot \Delta(q_j, M) \]

subject to

\[ \sum_{i=1}^{m} |S(\phi_i, K)| \cdot z_i \leq C \]
MILP Formulation

Maximize

\[ G = \sum_j p_j \cdot y_j \cdot \Delta(q_j, M) \]

subject to

Cost of all Samples

\[ \sum_{i=1}^m |S(\phi_i, K)| \cdot z_i \leq C \]
MILP Formulation

Maximize

\[ G = \sum_j p_j \cdot y_j \cdot \Delta(q_j, M) \]

subject to

Cost of all Samples

\[ \sum_{i=0}^m V(q_i, K) \cdot z_i \leq C \]

\[ p_j = \text{Probability of each Query Type in the Workload} \]
**MILP Formulation**

Maximize

\[
G = \sum_{j} p_j \cdot y_j \cdot \Delta(q_j, M)
\]

subject to

\[
\sum_{i=1}^{m} \left| S(\phi_i, K) \right| \cdot z_i \leq C
\]

\[
\forall j: y_j \leq \max_{i: \phi_i \subseteq q_j \cup i: \phi_i \supsetneq q_j} \left( z_i \min \left( 1, \frac{|D(\phi_i)|}{|D(q_j)|} \right) \right)
\]

Cost of all Samples

Coverage Probability of each query Type
MILP Formulation

Maximize

\[ G = \sum_{j} p_j \cdot y_j \cdot \Delta(q_j, M) \]

subject to

Cost of all Samples

\[ \sum_{i=1}^{m} |\phi_i, K| \cdot z_i \leq C \]

\[ \Delta(q_j, M) = \text{Sparsity Function} \]
Experimental Setup

• **Conviva**: 30-day log of media accesses by Conviva users. Raw data 17 TB, partitioned this data across 100 nodes

• Log of 17,000 queries (a sample of 200 queries had 17 templates).

• 50% of storage budget: 8 Stratified Samples
Sampling Vs. No Sampling

![Bar chart showing query response time for different input data sizes (2.5TB and 7.5TB) and different data caching strategies (Fully Cached, Partially Cached). The chart compares Hive, Shark, and BlinkDB (1% relative error) performance.]
BlinkDB: Evaluation

Query Response Time (seconds)

Input Data Size (TB)

- Hive
- Shark

BlinkDB (1% relative error)
BlinkDB: Evaluation

Query Response Time (seconds)

Input Data Size (TB)

200-300x Faster!
Response Time vs. Error

![Graph showing the relationship between response time and statistical error for different sample types. The graph includes data for Uniform Samples, Single Column, and Multi-Column samples.](image-url)
Time Guarantees

![Graph showing the relationship between requested response time and actual response time. The x-axis represents requested response time in seconds, ranging from 2 to 10. The y-axis represents actual response time in seconds, ranging from 0 to 12. The graph indicates a linear increase in actual response time as requested response time increases.]
Error Guarantees
Related Work

Taxonomy of Workload Models
BlinkDB is Open Sourced!

http://blinkdb.org

Deployed and used by Facebook

Integrated into Presto
Conclusion

- **Approximation** is an important means to achieve **interactivity** in the big data age

- Ad-hoc exploratory queries on an optimal set of multi-dimensional stratified samples **converges to lower errors 2-3 orders of magnitude** faster than non-optimal strategies
References

• **Blink and It's Done: Interactive Queries on Very Large Data**, S. Agarwal, A. Panda, B. Mozafari, A. Iyer, S. Madden, I. Stoica, VLDB 2012 demo

• **BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data**, S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, I. Stoica, EuroSys 2013 [Best Paper Award]

• **The Analytical Bootstrap: A New Method for Fast Error Estimation in Approximate Query Processing**, K. Zeng, G. Shi, B. Mozafari, C. Zaniolo, under submission
Backup Slides