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**

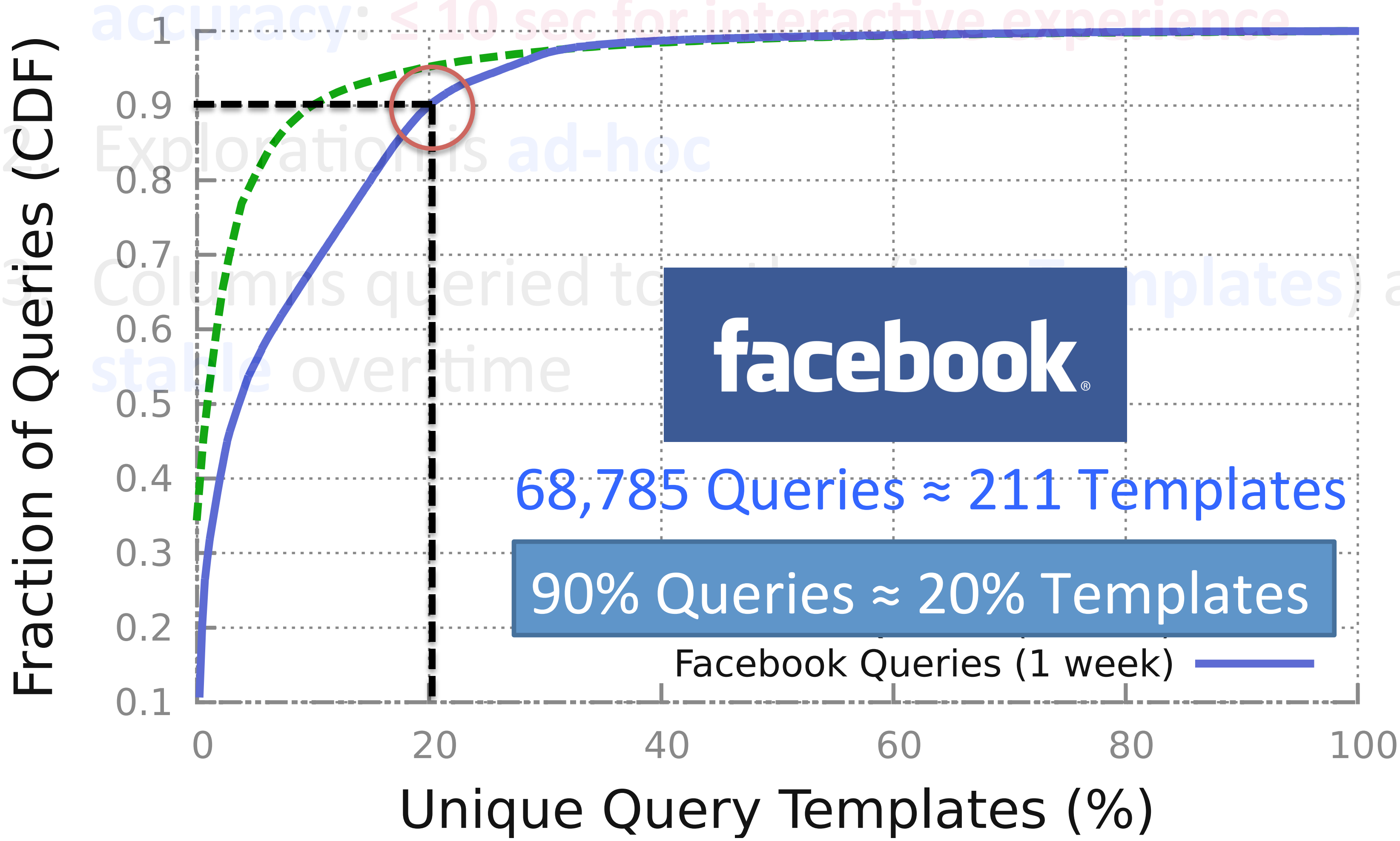
"On a good day, I can run up to 6 queries in Hive."
- Anonymous Data Scientist at facebook

Target Workload

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

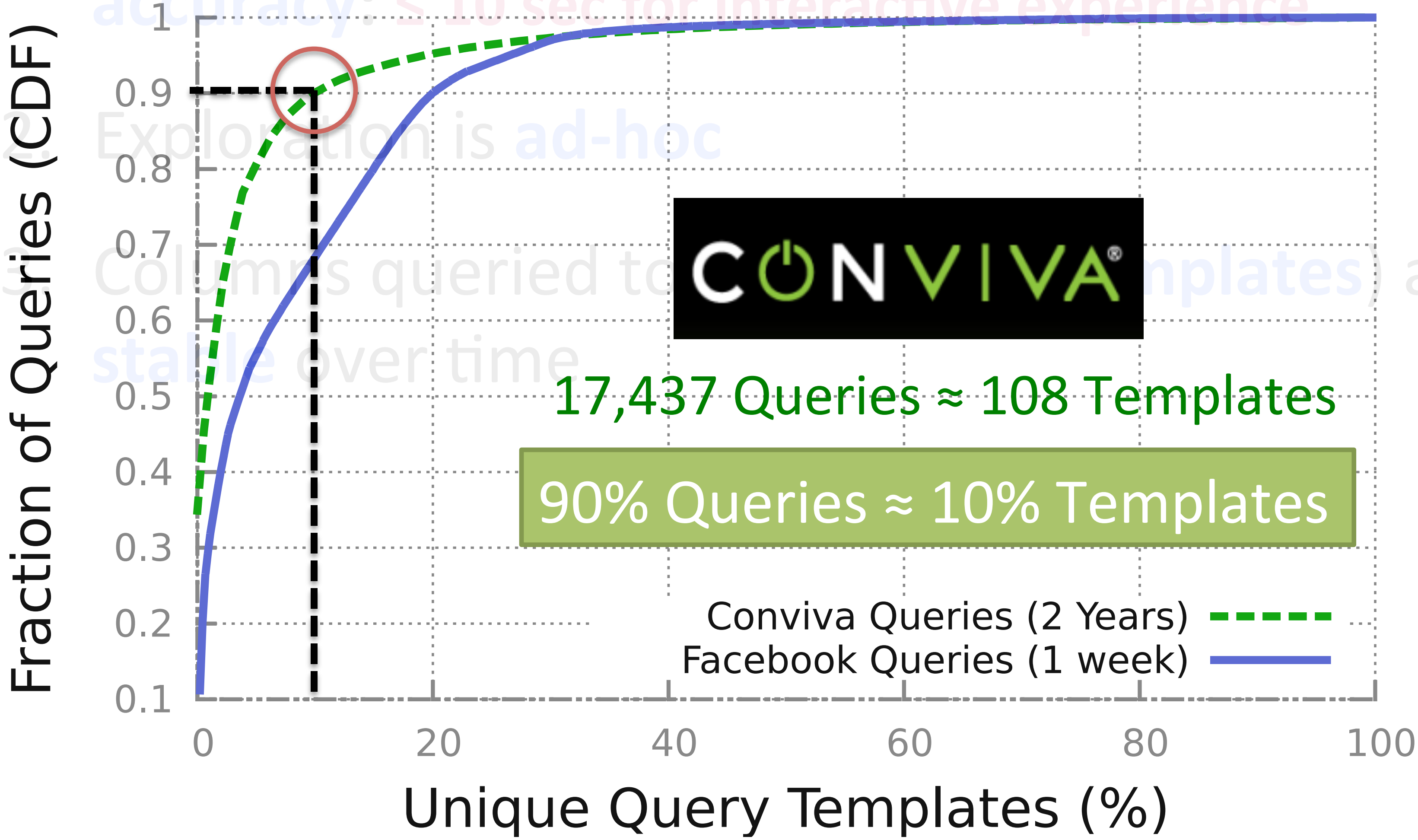
Target Workload

1. **Real-time latency** is valued over perfect accuracy: ≤ 10 sec for interactive experience
2. Exploration is ad-hoc
3. Columns queried to… Templates) are stable over time



Target Workload

1. **Real-time latency** is valued over perfect accuracy: ≤ 10 sec for interactive experience
2. Exploration is ad-hoc
3. Columns queried to … are stable over time



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**

100 TB on 1000 machines

1-2 Hours

Hard Disks

25-30 Minutes

Memory

1 second

?

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** → ~~234.23 ± 15.32~~

239.46 ± 4.96

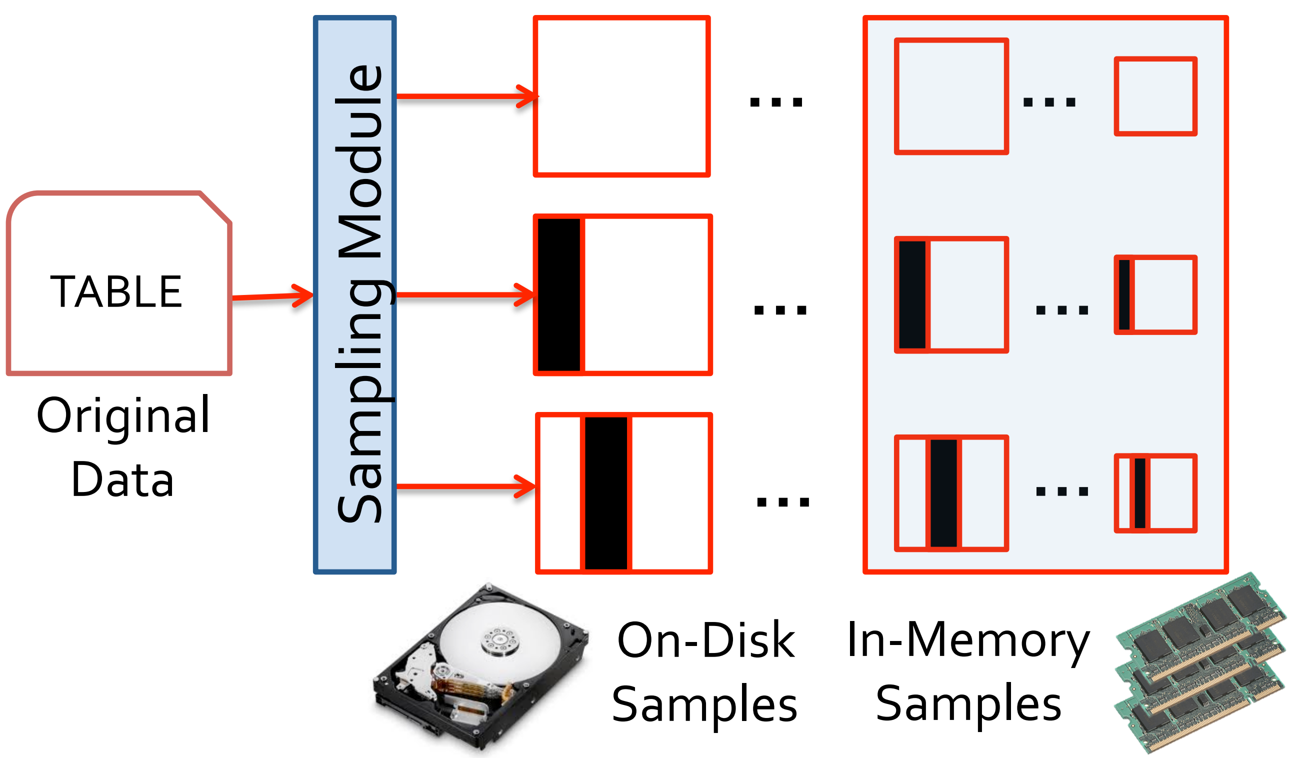
SELECT avg(sessionTime)

FROM Table

WHERE city='San Francisco'

ERROR 0.1 **CONFIDENCE** 95.0%

BlinkDB Architecture



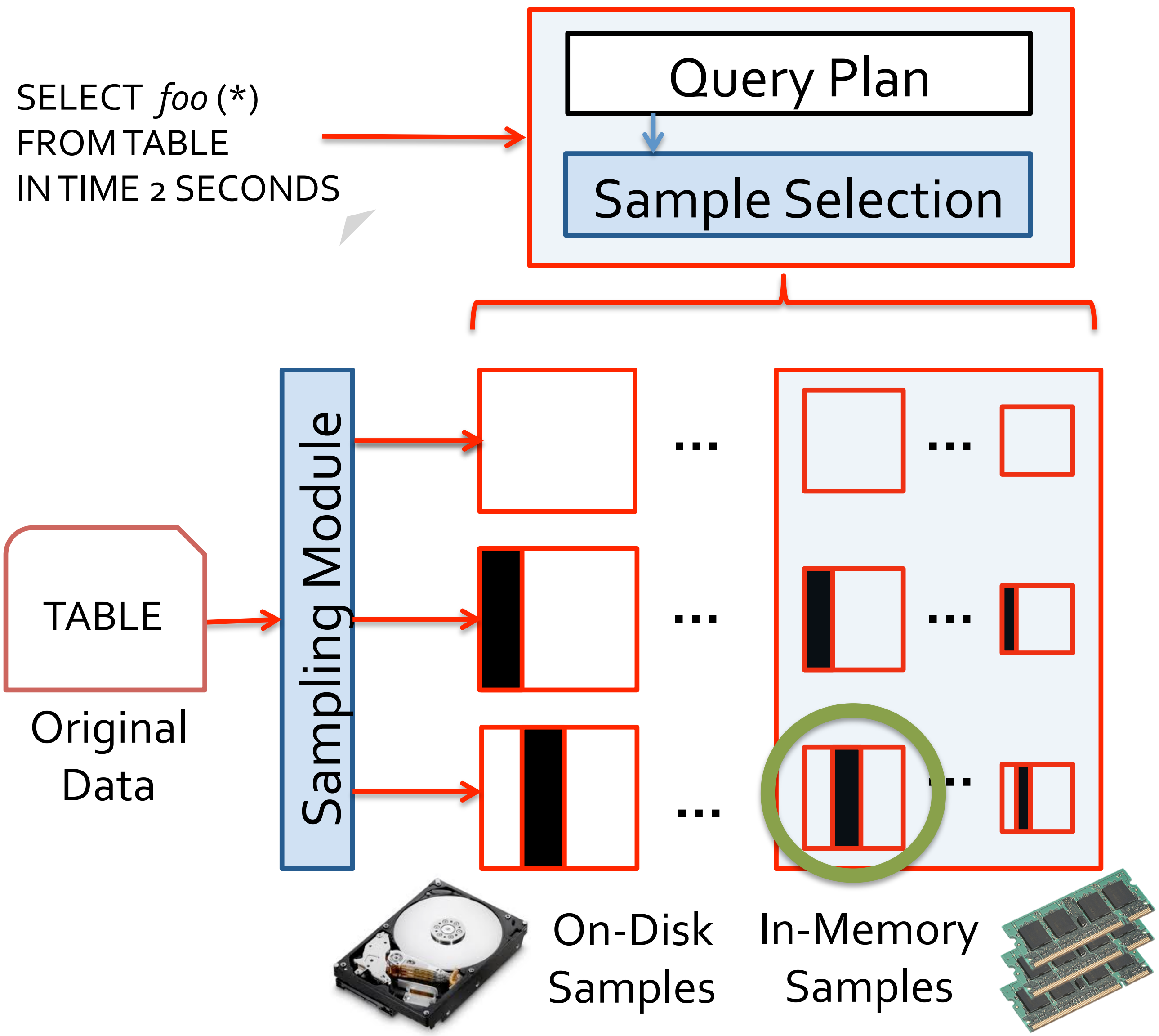
Offline sampling:

» **Uniform**

» **Stratified** on different sets of columns

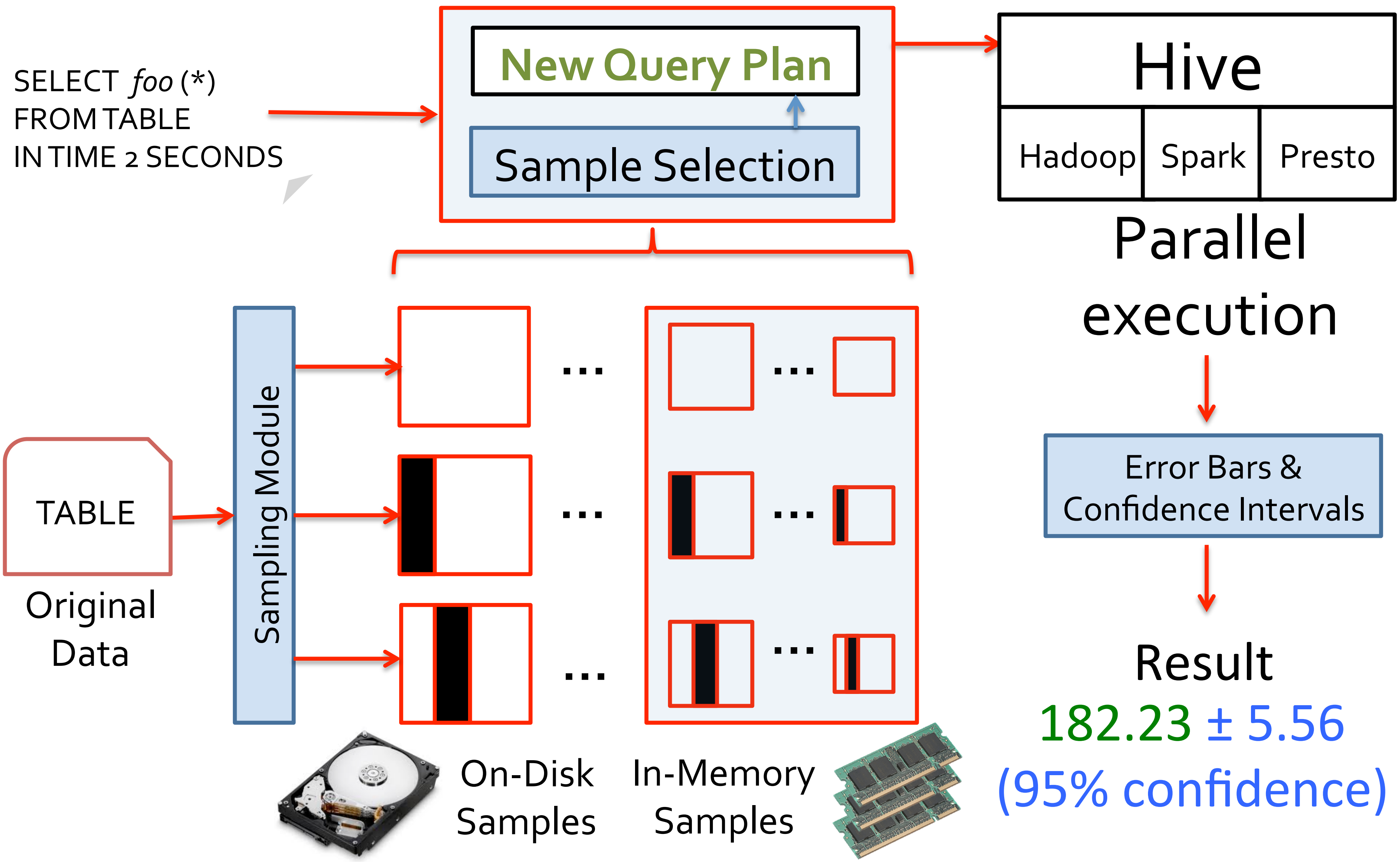
» Different sizes

BlinkDB Architecture



Predict **time** and **error** of the query for each sample type

BlinkDB Architecture



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)

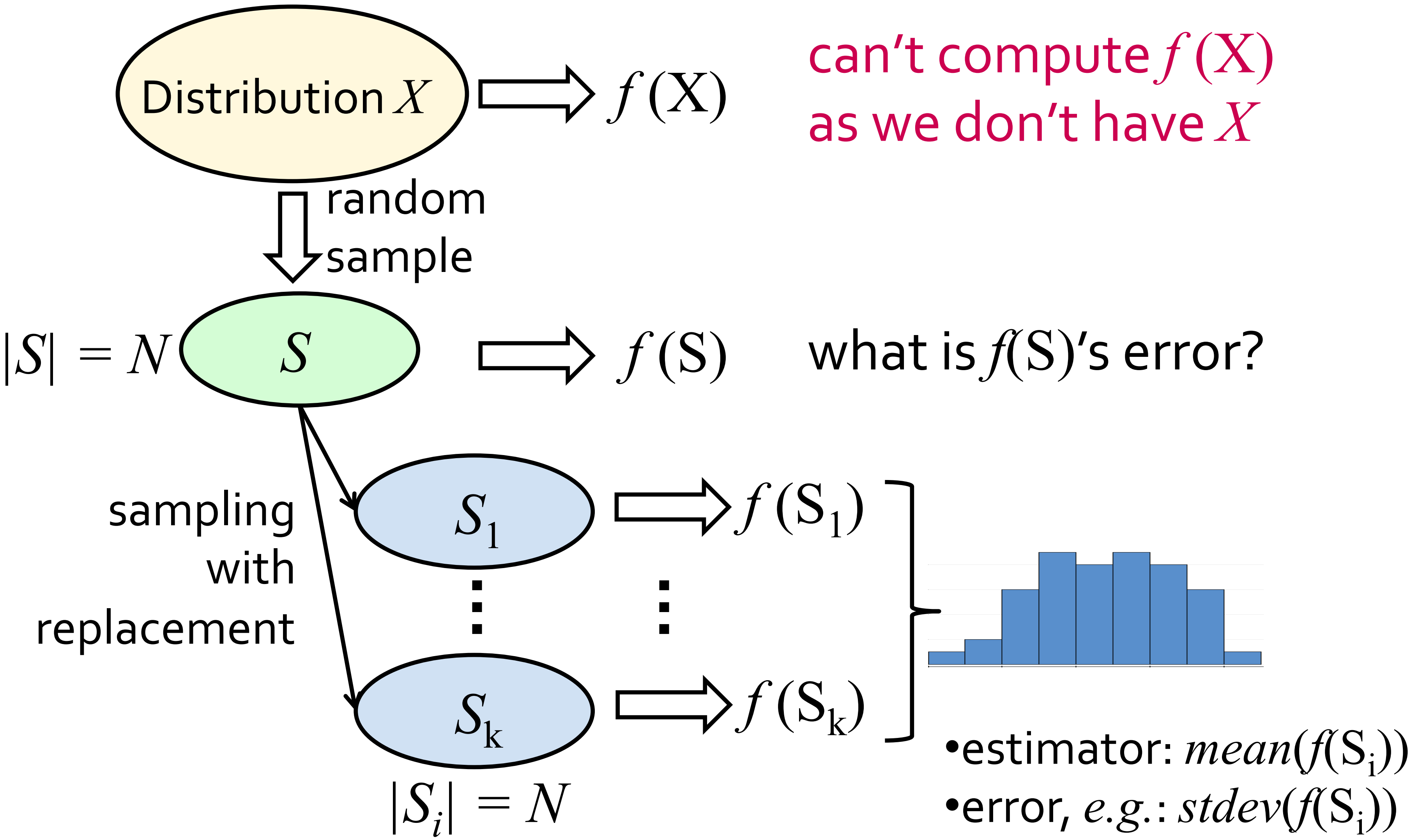
1. Count: $N(np, n(1-p)p)$
2. Sum: $N(np\mu, np(\sigma^2 + (1-p)\mu^2))$
3. Mean: $N(\mu, \sigma^2/n)$
4. Variance: $N(\sigma^2, (\mu_4 - \sigma^4)/n)$
5. Stddev: $N(\sigma, (\mu_4 - \sigma^4)/(4\sigma^2 n))$

What about more complex queries?

- UDFs, nested queries, joins, ...

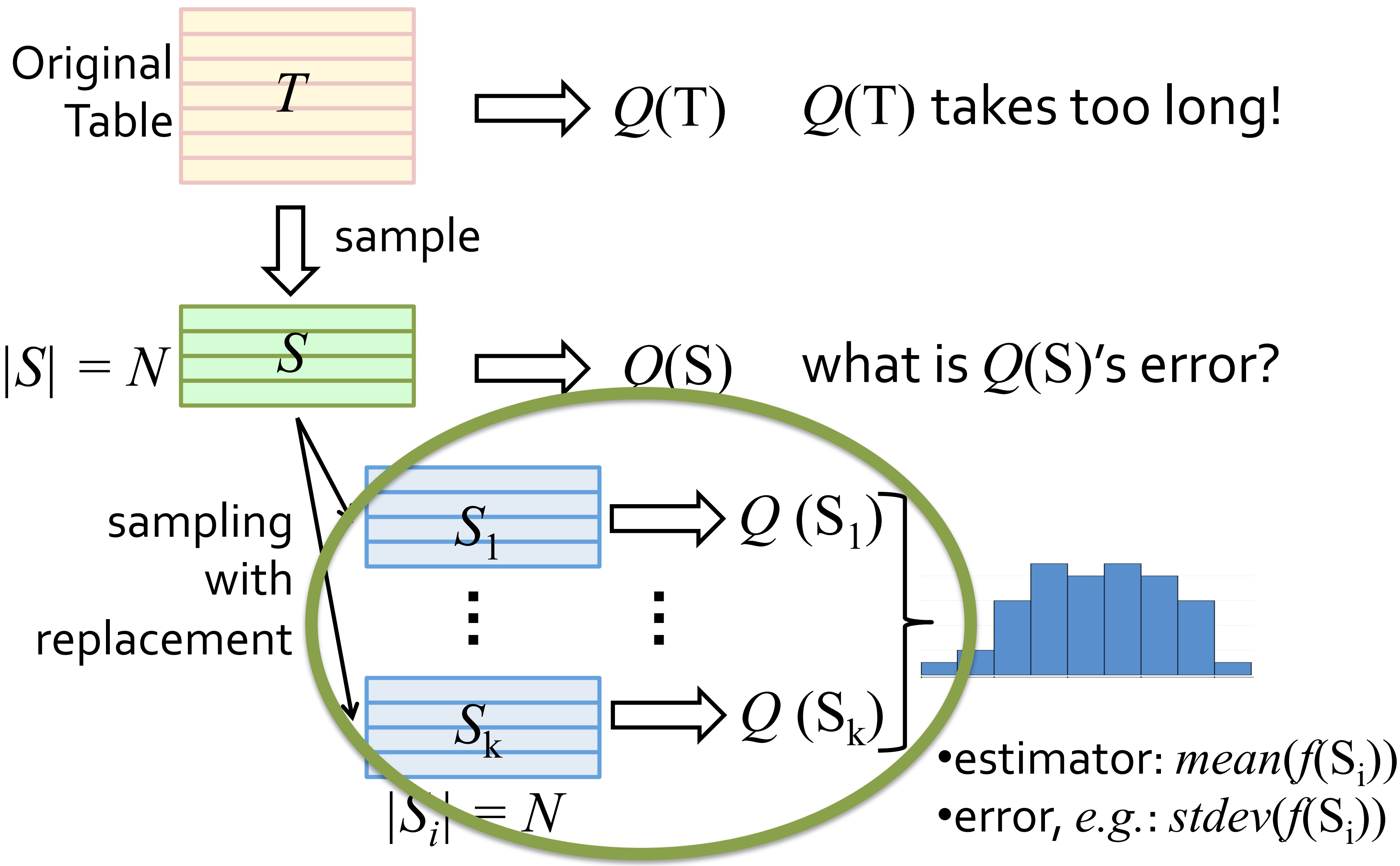
Bootstrap [Efron 1979]

Quantify accuracy of a sample estimator $f()$



Bootstrap

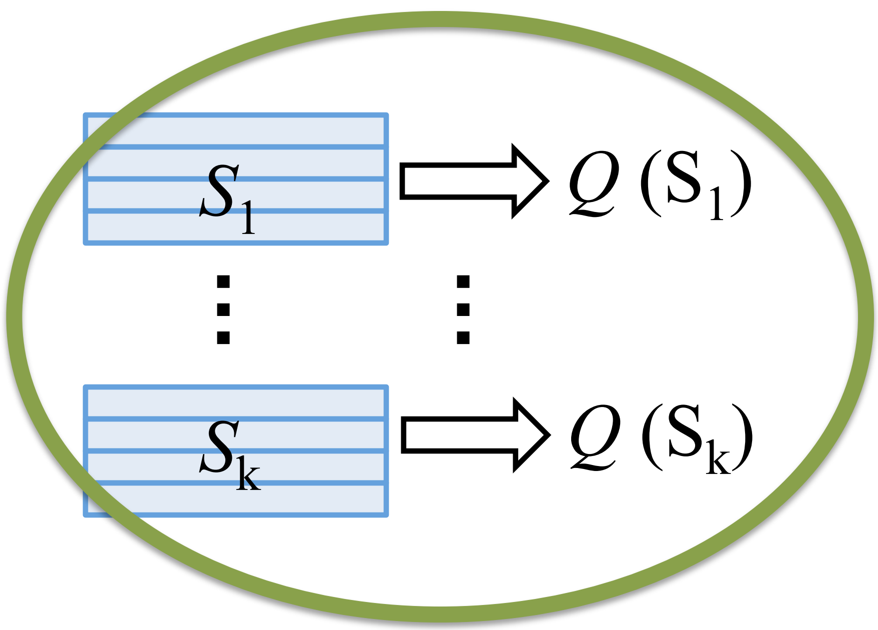
Quantify accuracy of a query on a sample table



Bootstrap

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

2. **Embarrassingly Parallel**



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, ...)

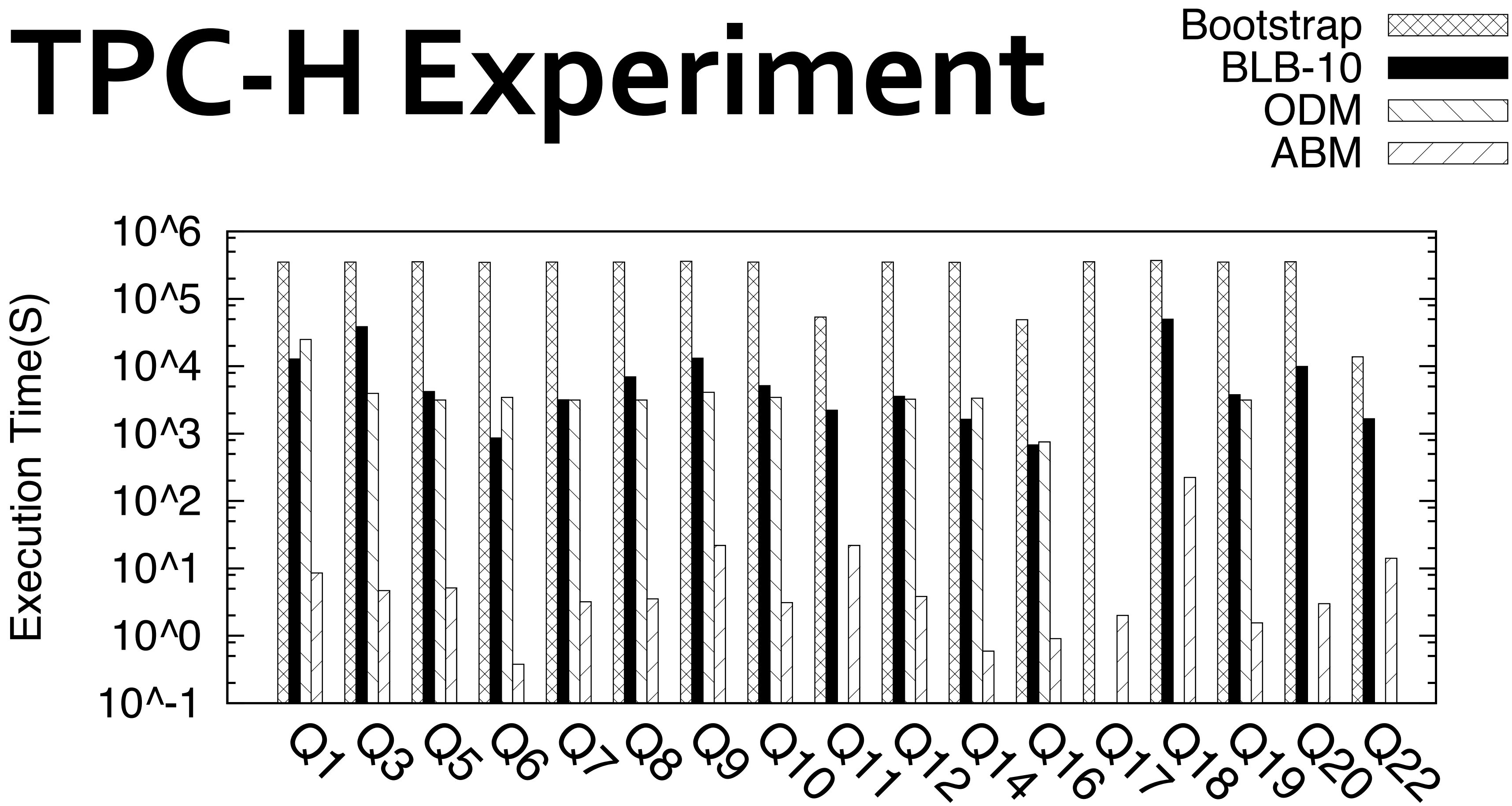
Analytical Bootstrap Method*

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

ID	City	Age	Salary
1	NYC	22	50,000
2	Ann Arbor	25	120,242
3	NYC	25	78,212
4	NYC	67	62,492
5	NYC	34	98,341
6	Ann Arbor	62	78,453

Uniform Sample

ID	City	Age	Salary	Sampling Rate
3	NYC	25	78,212	1/3
5	NYC	34	98,341	1/3

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

Problem with Uniform Samples

ID	City	Age	Salary
1	NYC	22	50,000
2	Ann Arbor	25	120,242
3	NYC	25	78,212
4	NYC	67	62,492
5	NYC	34	98,341
6	Ann Arbor	62	78,453

Larger Uniform Sample

ID	City	Age	Salary	Sampling Rate
3	NYC	25	78,212	2/3
5	NYC	34	98,341	2/3
1	NYC	22	50,000	2/3
2	Ann Arbor	25	120,242	2/3

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

Stratified Samples

ID	City	Age	Salary
1	NYC	22	50,000
2	Ann Arbor	25	120,242
3	NYC	25	78,212
4	NYC	67	62,492
5	NYC	34	98,341
6	Ann Arbor	62	78,453

Stratified Sample on City

ID	City	Age	Salary	Sampling Rate
3	NYC	67	62,492	1/4
5	Ann Arbor	25	120,242	1/2

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 ≈ 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

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

[City]

[Age]

[Salary]

[City, Age]

[Age, Salary]

[City, Salary]

[City, Age, Salary]

Query Coverage

SELECT AVG (…)

FROM Table

WHERE Age = *x*

[City]

[Age]

[Salary]

[City, Age]

[Age, Salary]

[City, Salary]

[City, Age, Salary]

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*

[City]

[Age]

[Salary]

[City, Age]

[Age, Salary]

[City, Salary]

[City, Age, Salary]

Query Coverage

SELECT AVG (…)

FROM Table

WHERE **Age** = *x* AND

City = *z*

[City]	
[Age]	
[Salary]	
[City, Age]	**100%**
[Age, Salary]	
[City, Salary]	
[City, Age, Salary]	**100%**

Query Coverage

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

[City]	**?**
[Age]	**?**
[Salary]	**0%**
[City, Age]	**100%**
[Age, Salary]	**?**
[City, Salary]	**?**
[City, Age, Salary]	**100%**

Query Coverage

SELECT AVG (…)

FROM Table

WHERE **Age** = *x* AND

City = *z*

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

[City]	50%
[Age]	83%
[Salary]	0%
[City, Age]	100%
[Age, Salary]	100%
[City, Salary]	100%
[City, Age, Salary]	100%

Cost of Stratification

ID	City	Age	Salary
1	NYC	25	50,000
2	NYC	25	80,000
3	Ann Arbor	35	80,000
4	NYC	25	120,000
5	NYC	25	80,000
6	Berkeley	25	80,000
7	NYC	25	60,000
8	NYC	25	70,000
9	Berkeley	30	80,000
10	Berkeley	25	90,000
11	NYC	40	80,000
12	Ann Arbor	45	100,000

Stratified Sample on [City]

ID	City	Age	Salary	Ratio
1	NYC	25	50,000	2/7
8	NYC	35	70,000	2/7
6	Berkeley	25	80,000	2/3
10	Berkeley	25	90,000	2/3
3	Ann Arbor	35	80,000	1
12	Ann Arbor	45	100,000	1

Cost = 6

Cost of Stratification

ID	City	Age	Salary
1	NYC	25	50,000
2	NYC	25	80,000
3	Ann Arbor	35	80,000
4	NYC	25	120,000
5	NYC	25	80,000
6	Berkeley	25	80,000
7	NYC	25	60,000
8	NYC	25	70,000
9	Berkeley	30	80,000
10	Berkeley	25	90,000
11	NYC	40	80,000
12	Ann Arbor	45	100,000

Stratified Sample on [Salary]

ID	City	Age	Salary	Ratio
1	NYC	25	50,000	1
7	NYC	25	60,000	1
8	NYC	25	70,000	1
3	Ann Arbor	35	80,000	1/3
9	Berkeley	30	80,000	1/3
10	Berkeley	25	90,000	1
12	Ann Arbor	45	100,000	1
4	NYC	25	120,000	1

Cost = 8

MILP Formulation

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

subject to

$$\sum_{i=1}^{m} |S(\phi_i, K)| \cdot z_i \leq \mathbb{C}$$

MILP Formulation

Maximize $$G = \sum_j p_j \cdot y_j \cdot \Delta(q_j, M)$$

subject to

$$\sum_{i=1}^{m} |S(\phi_i, R)| \cdot z_i \leq \mathbb{C}$$

Cost of all Samples

MILP Formulation

Maximize $$G = \sum_j \boxed{p_j \cdot}\, y_j \cdot \Delta(q_j, M)$$

subject to

Cost of all Samples

$$\sum_{i=1}^{m} \left|S(\varphi_i, K)\right| \cdot z_i \leq \mathbb{C}$$

p_j **=** **Probability of each Query Type in the Workload**

MILP Formulation

Maximize $$G = \sum_{j} \boxed{p_j \cdot} \boxed{y_j \cdot} \Delta(q_j, M)$$

subject to

Cost of all Samples

$$\sum_{i=1}^{m} |S(\phi_i, K)| \cdot z_i \leq \mathbb{C}$$

Coverage Probability of each query Type

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

MILP Formulation

Maximize

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

subject to

Cost of all Samples

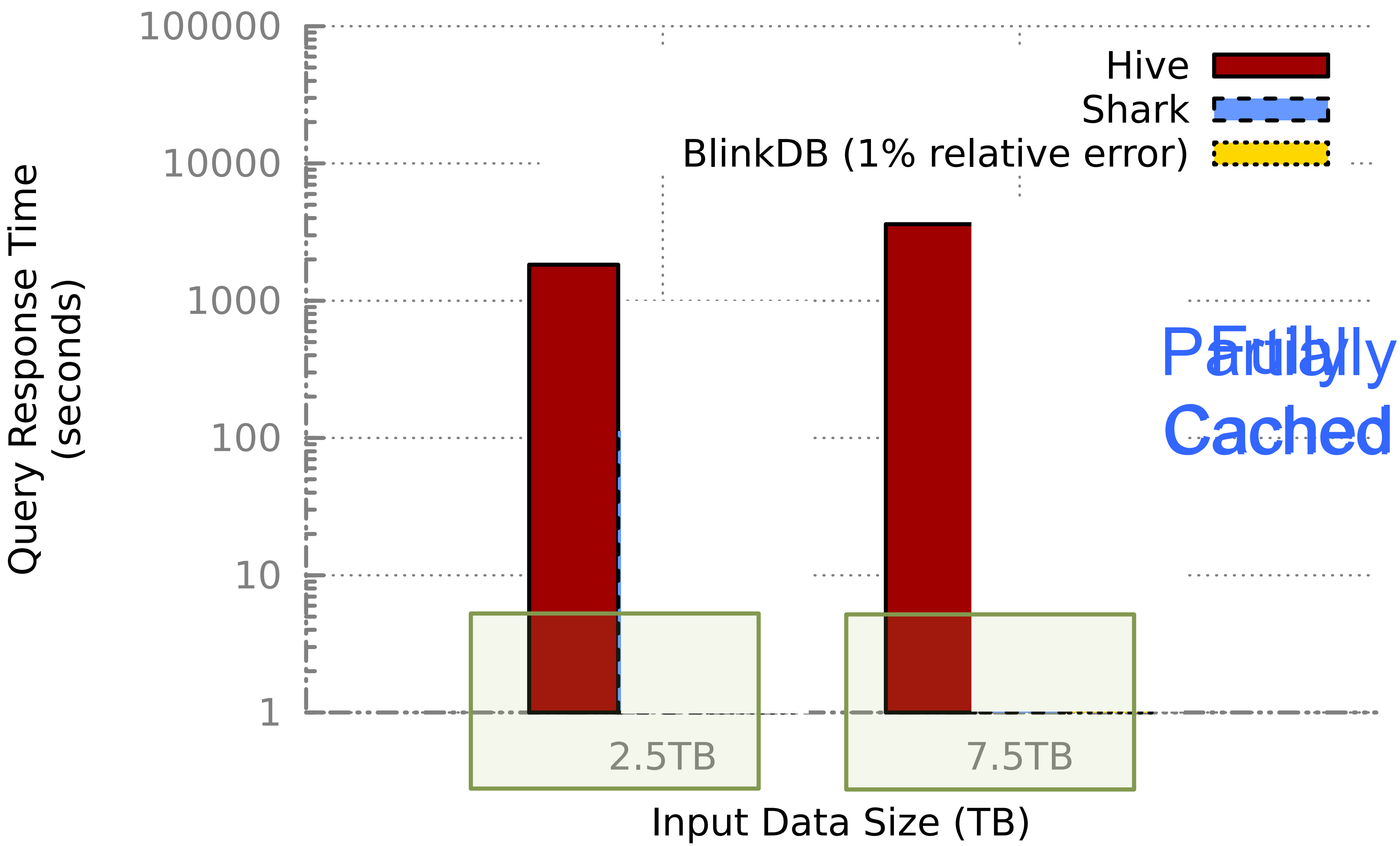
$$\sum_{i=1}^{m} |S(\varphi_i, R)| \cdot z_i \leq \mathbb{C}$$

$\Delta(q_j, M)$ **=** **Sparsity Function**

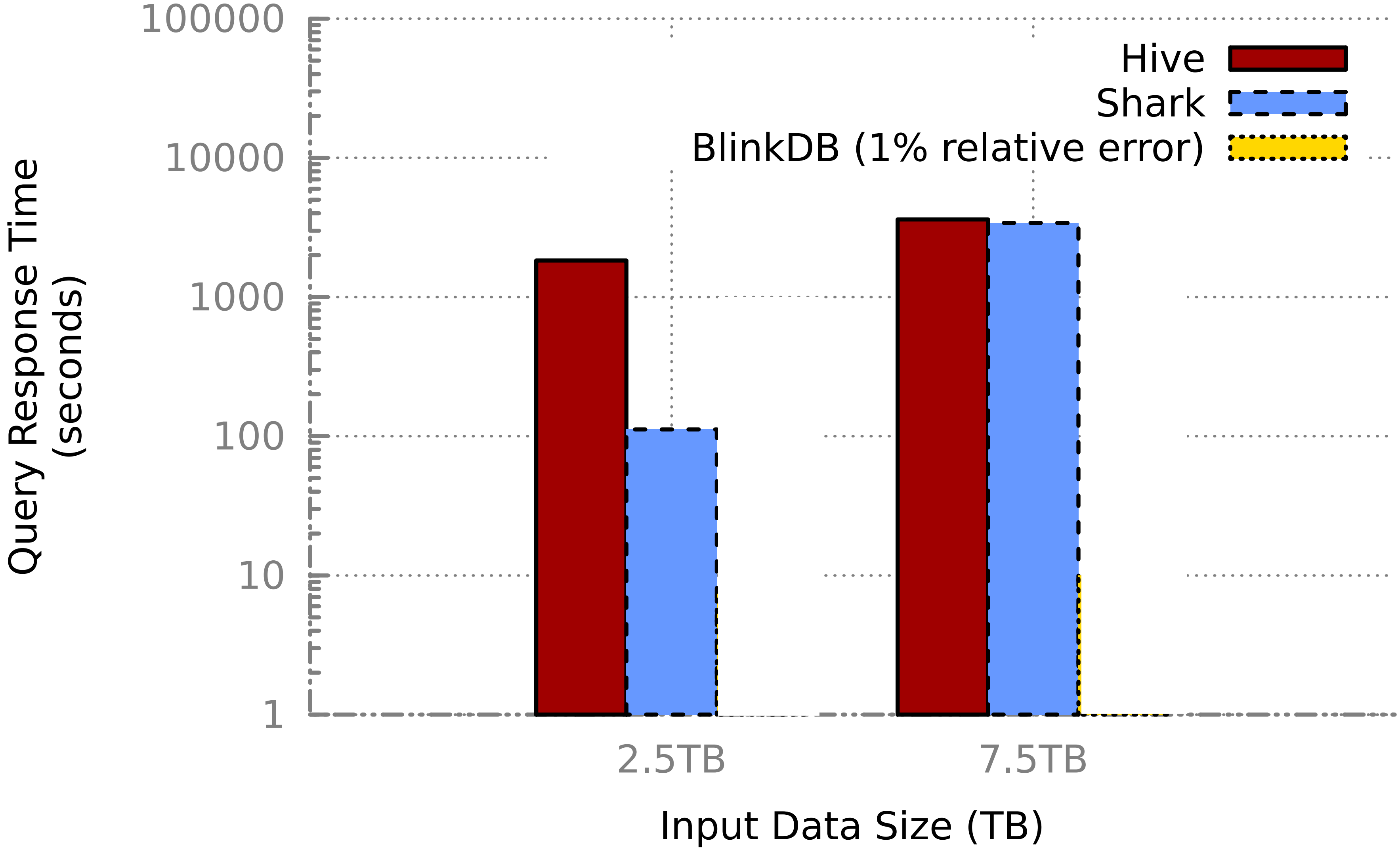
Experimental Setup

- **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

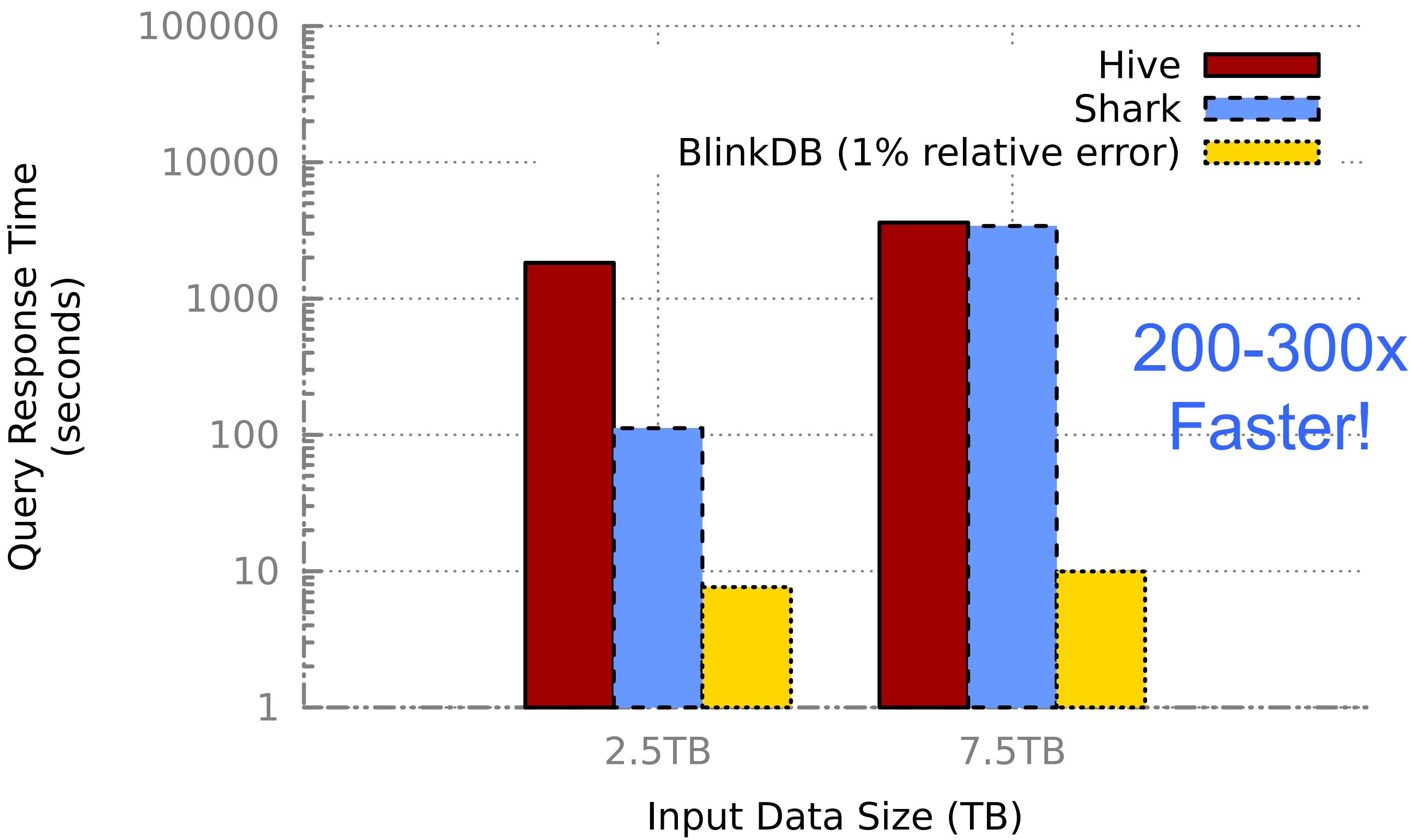
Sampling Vs. No Sampling



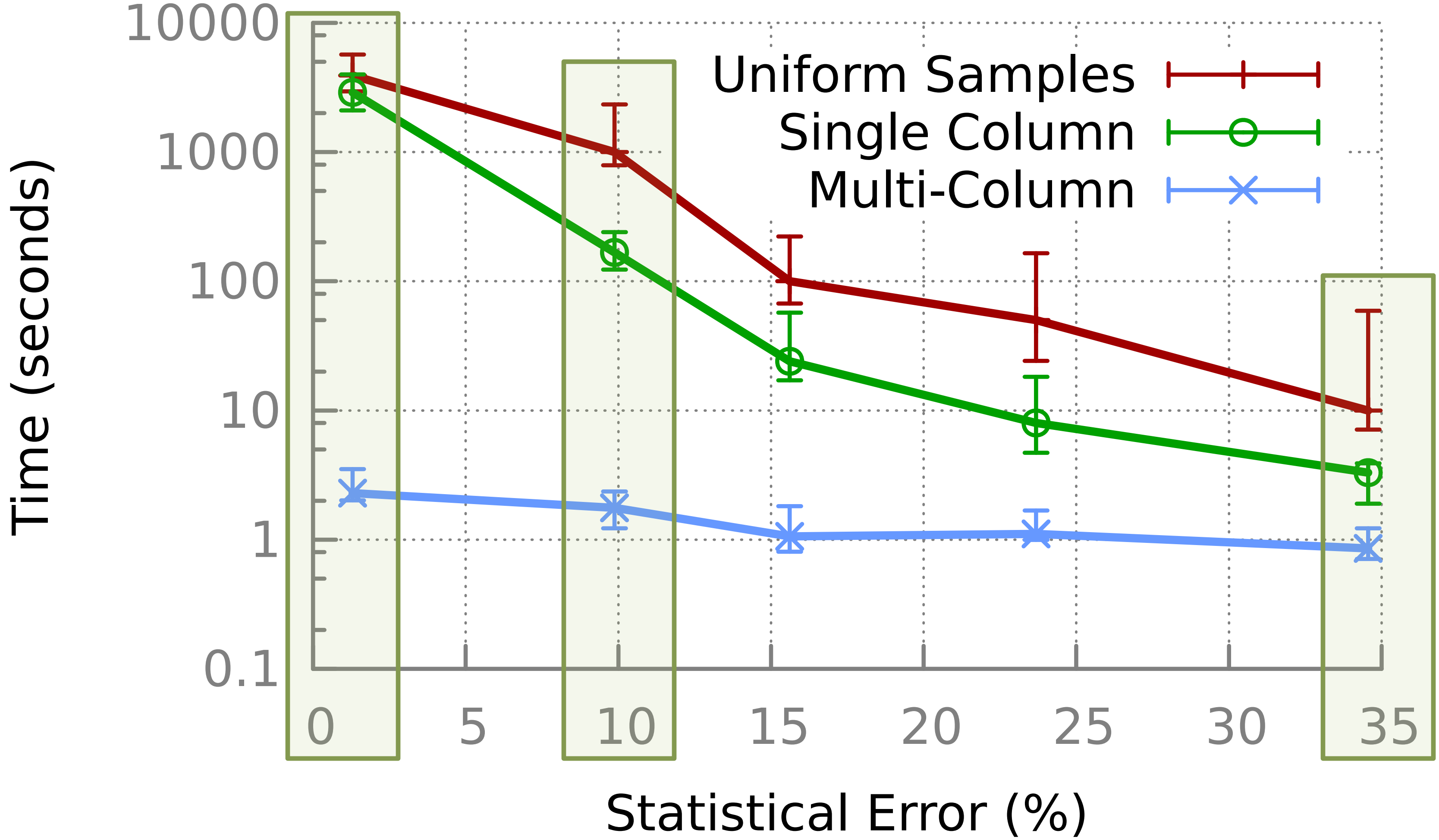
BlinkDB: Evaluation



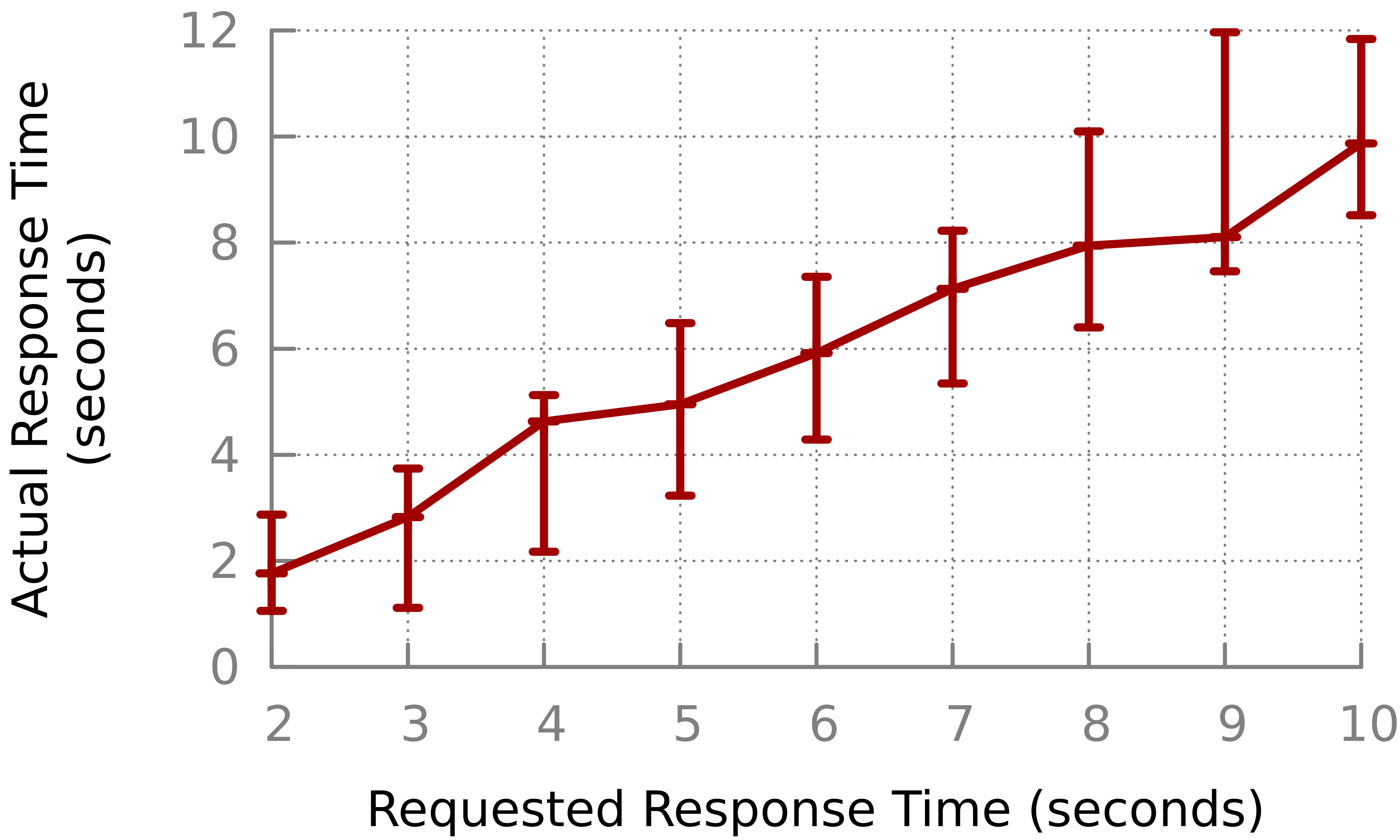
BlinkDB: Evaluation



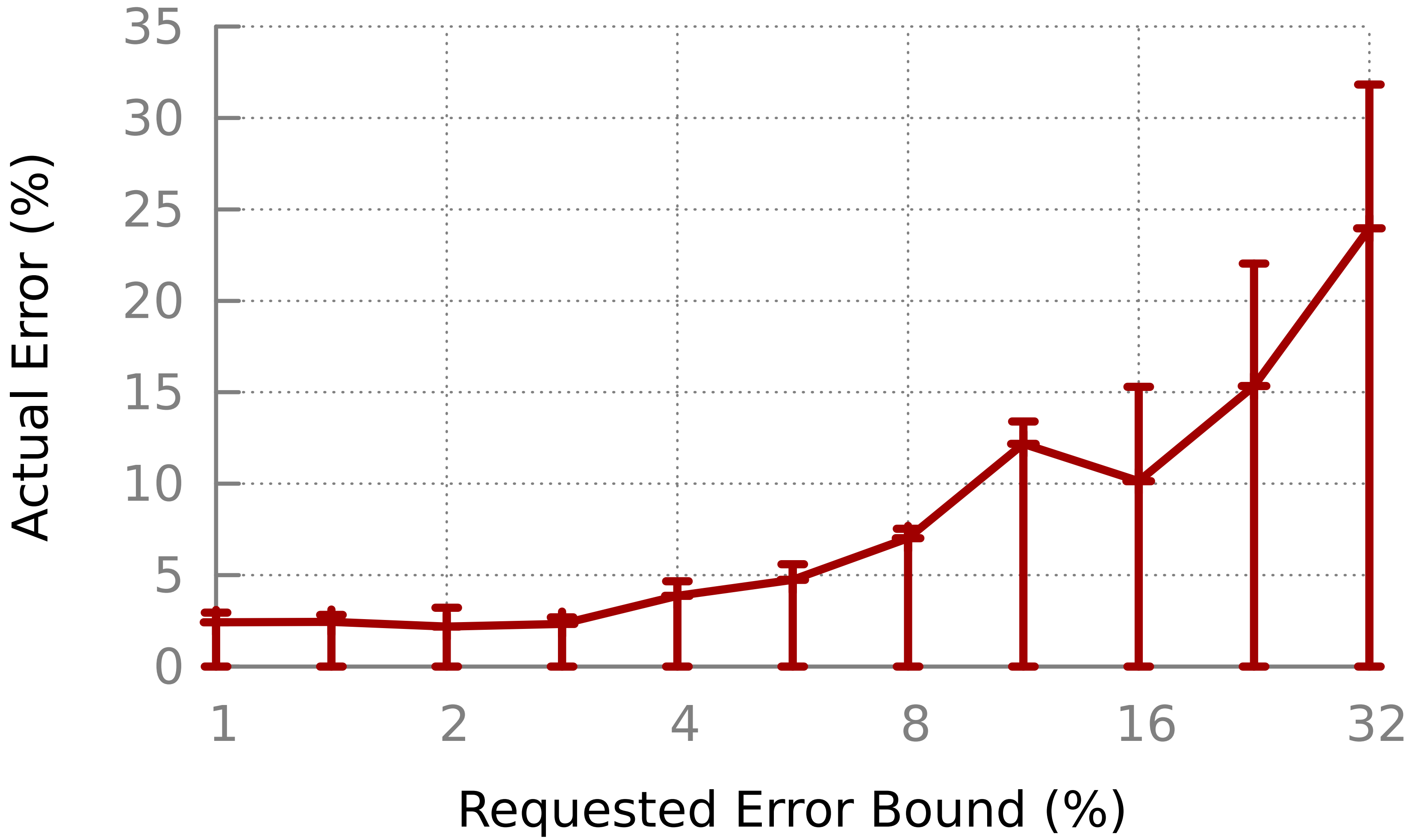
Response Time vs. Error



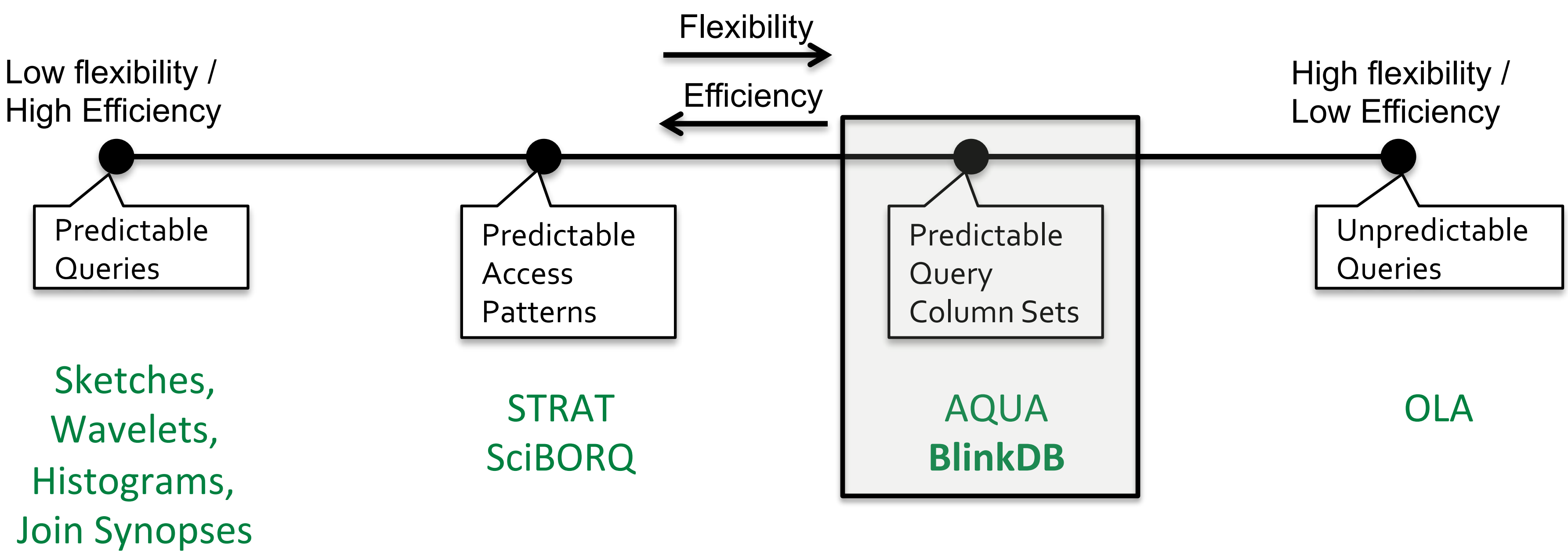
Time Guarantees



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