BlinkDB: Queries with Bounded Error and Bounded Response Times on Very Large Data

Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, Ion Stoica

Presented by Liqi Xu

Problem: very large data

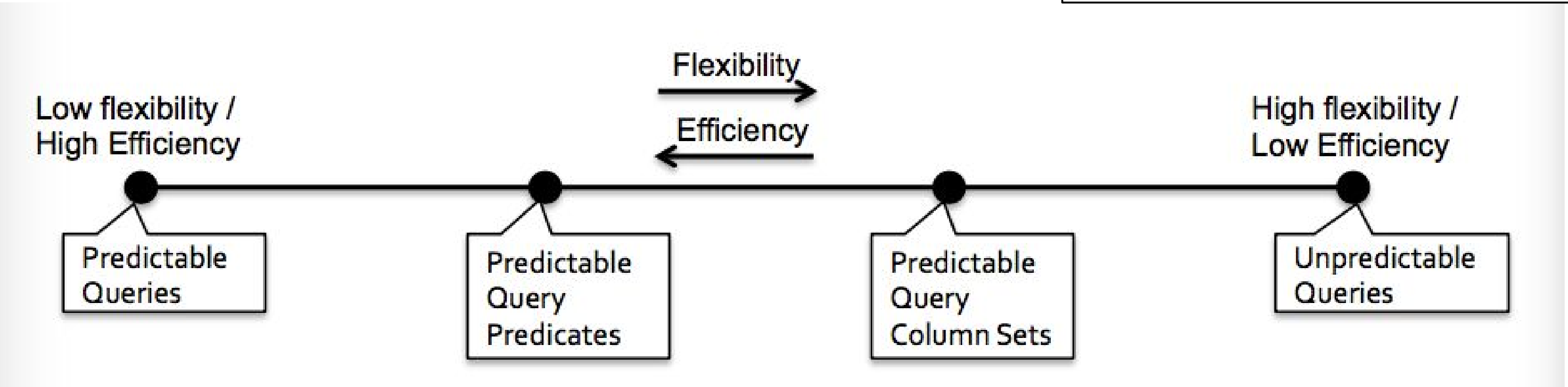
```
SELECT AVG(SessionTime)

FROM Sessions

WHERE City  = ‘New York’
```

- 100 million tuples for ‘New York’
- Problem:
    - High cost in execution *time* and *space*
- Idea: trade *result accuracy* for *response time and space*
- Sampling:
    - 10,000 tuples for ‘New York ’
    - return an approximate result (with error bound)
        - E.g. appox. avg 234.23 ± 5.32

Problems: approx. techniques

```
SELECT AVG(SessionTime)
FROM Sessions
WHERE City  = 'New York'
```

efficiency v.s. *flexibility* of the queries



All future queries are known in advance

Frequencies of group and filter predicates do not change over time

Frequencies of **set of columns used for** group and filter predicates do not change over time

No future queries are known in advance

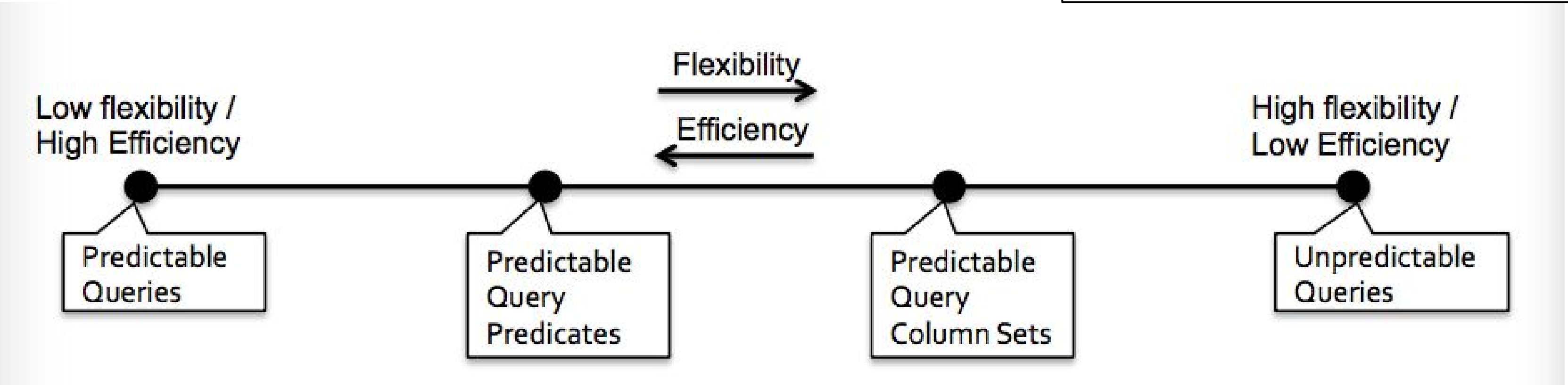
Problems: approx. techniques

SELECT AVG(SessionTime)

FROM Sessions

WHERE City = 'Urbana'

efficiency v.s. *flexibility* of the queries



All future queries are known in advance

Frequencies of group and filter predicates do not change over time

Frequencies of **set of columns used for** group and filter predicates do not change over time

No future queries are known in advance

'current' sampling

Online Aggregation

BlinkDB

- “a distributed sampling-based approximate query processing system”
- Efficient
  - ~TBs data in seconds
  - with meaningful error bounds

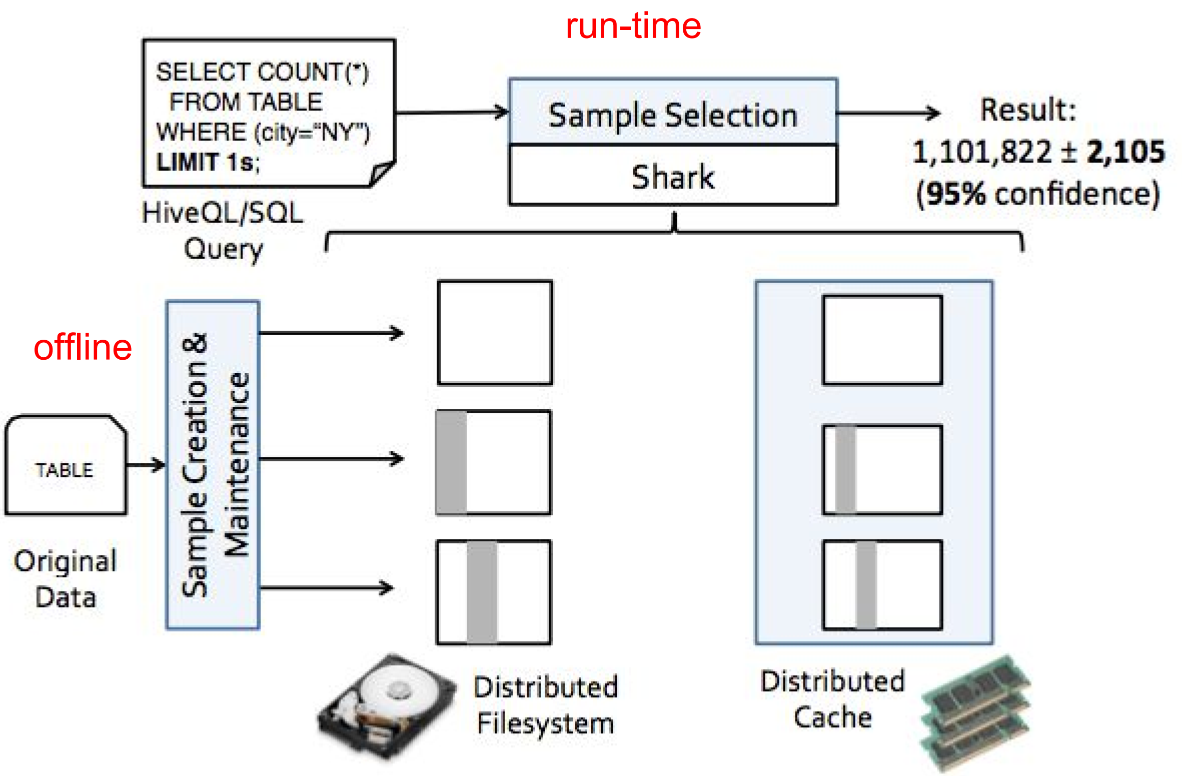
```
SELECT COUNT(*)
FROM Sessions
WHERE Genre = ‘western’
GROUP BY OS
ERROR WITHIN 10% AT CONFIDENCE 95%
```

```
SELECT COUNT(*)
FROM Sessions
WHERE Genere = ‘western’
GROUP BY OS
WITHIN 5 SECONDS
```

BlinkDB

- "a distributed sampling-based approximate query processing system"
- Efficient
  - ~TBs data in seconds
  - with meaningful error bounds
- More general queries
  - Only assumption:
    - *"query column sets"* (QCSs) are *stable*
    - QCSs: columns used for grouping and filtering (ie. in WHERE, GROUP BY, and HAVING)

BlinkDB Architecture



Sample creation

- Construct stratified samples

Problem with Uniform Samples

1. higher possibility of missing under-representing groups

```
SELECT AVG(SessionTime)
FROM Sessions
WHERE City = ‘Urbana’’
```

ID	City	Age	Session_Time
1	NYC	20	212
2	Urbana	40	532
3	NYC	30	243
4	Urbana	40	291
5	NYC	20	453
6	NYC	30	293

Sampling_rate = ⅓

ID	City	Age	Session_Time
3	NYC	30	243
5	NYC	20	453

Problem with Uniform Samples

1. higher possibility of missing under-representing groups
2. Error of each aggregate is NOT equal

ID	City	Age	Session_Time
1	NYC	20	212
2	Urbana	40	532
3	NYC	30	243
4	Urbana	40	291
5	NYC	20	453
6	NYC	30	293

Sampling_rate = ⅔

ID	City	Age	Session_Time
1	NYC	20	212
3	NYC	30	243
4	Urbana	40	291
6	NYC	30	293

Stratified Samples (on *City*)

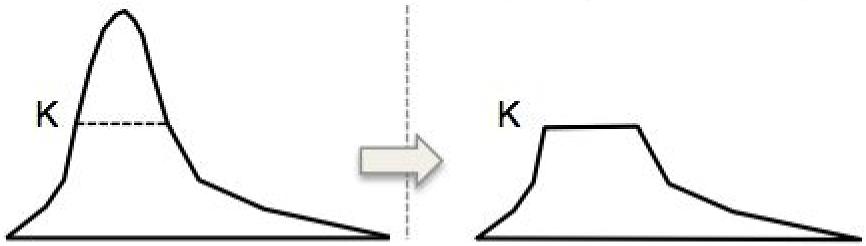
ID	City	Age	Session_Time
1	NYC	20	212
2	Urbana	40	532
3	NYC	30	243
4	Urbana	40	291
5	NYC	20	453
6	NYC	30	293

Sampling_rate(NYC) = 1/4
Sampling_rate(Urbana) = 1/2

ID	City	Age	Session_Time
3	NYC	30	243
4	Urbana	40	291

Assign equal sample size to each groups

Stratified Samples (on *City*)



ID	City	Age	Session_Time
1	NYC	20	212
2	Urbana	40	532
3	NYC	30	243
4	Urbana	40	291
5	NYC	20	453
6	NYC	30	293

Sampling_rate(NYC) = 3/4
Sampling_rate(Urbana) = 2/2

ID	City	Age	Session_Time
1	NYC	20	212
3	NYC	30	243
4	Urbana	40	291
5	NYC	20	453
6	NYC	30	293

Storage cost of stratified samples

- Build several multi-dimensional stratified samples
  - increase query accuracy and latency
- n columns ⟶ 2^n possible stratified samples

ID	City	Age	Session_Time
1	NYC	20	212
2	Urbana	40	532
3	NYC	30	243
4	Urbana	40	291
5	NYC	20	453

[City]
[Age]
[Session_Time]
[City, Age]
[City, Session_Time]
[Age, Session_Time]
[City, Age, Session_Time]

Storage cost of stratified samples

- Build several multi-dimensional stratified samples
  - increase query accuracy and latency
- n columns ⟶ 2^n possible stratified samples
- Solution:
  - Find subsets of column sets that maximize the weighted sum of coverage of the QCSs of the queries q_j

Optimization formulation

probability of a query type in workload

Coverage probability of each query type

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

Sparsity of the data

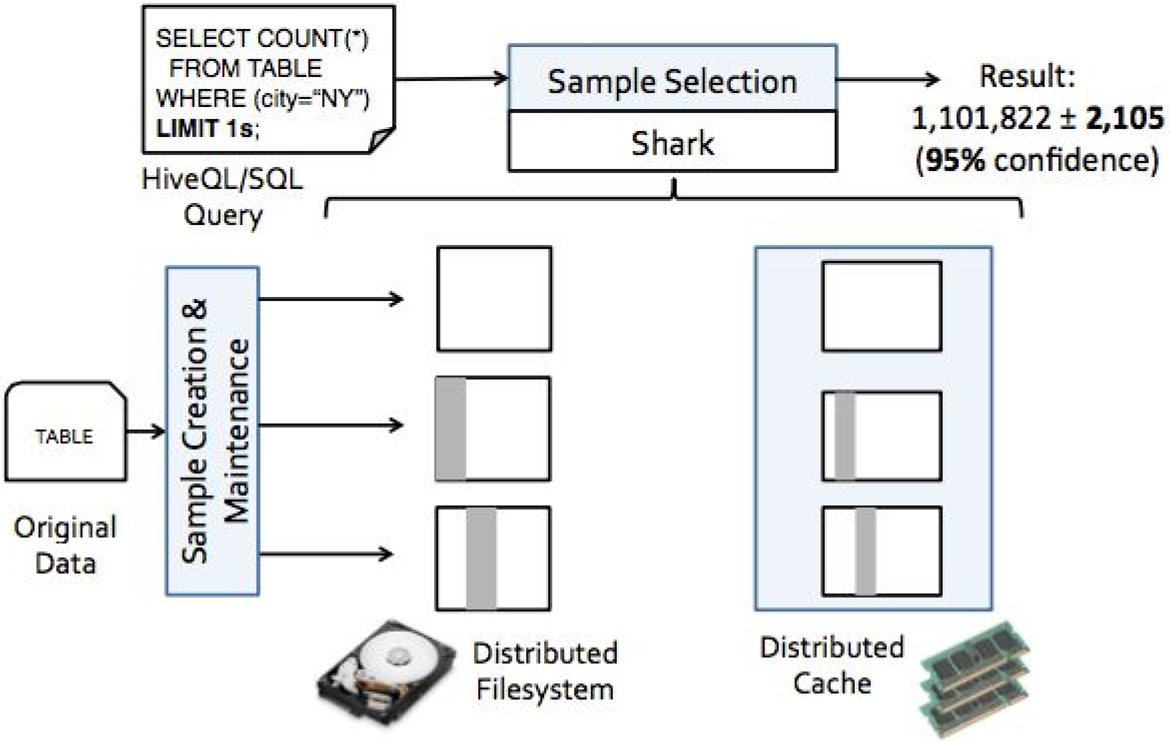
subject to

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

Overall storage capacity budget

storage cost of all samples

System Overview

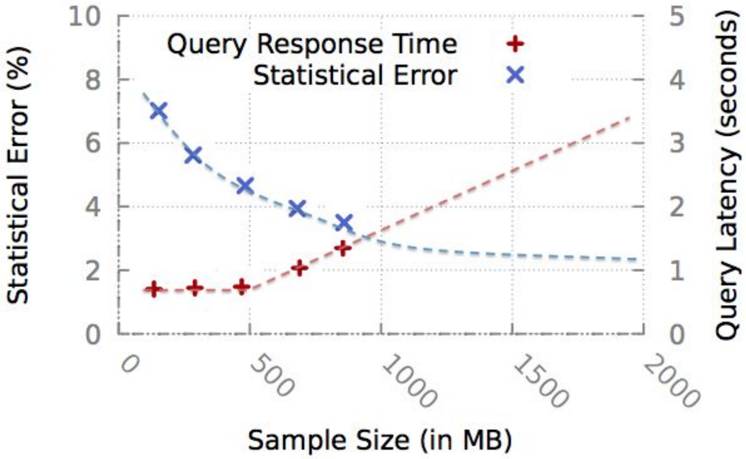


Online sample selection

- Given a Query Q with specified time/error constraints
    - BlinkDB generate different query plans for the same query Q
- How to pick the plan that best satisfies the time/error constraints?

Strategy

- Select appropriate sample(s)
- execute the query Q on small samples of those appropriate samples(s), in order to gather statistics about
  - query’s selectivity
  - complexity
  - underlying distribution of its query
- For each candidate sample
  - construct an *Error Latency Profile* (ELP)
  - statistically predict for larger samples



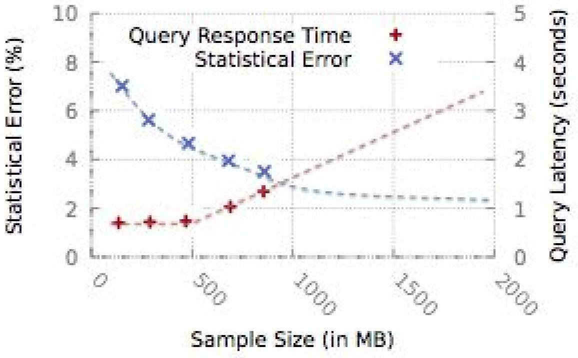
Example

SELECT AVG(SessionTime)

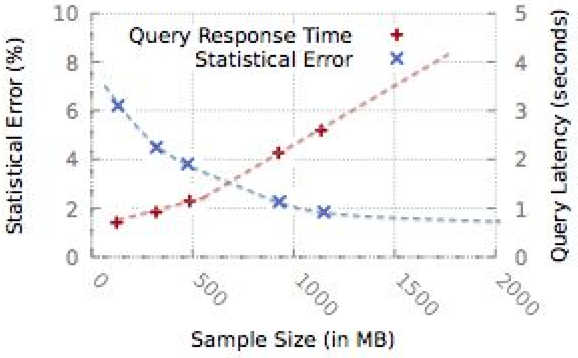
FROM Sessions

WHERE City = Galena'

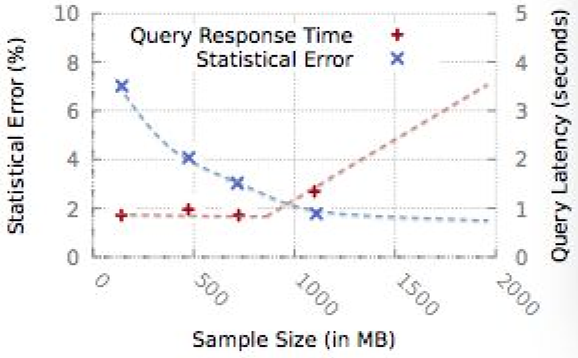
- System has 3 stratified samples
  - [date, country]
  - [date designated media area for a video
  - [date, ended_flag]
- Construct an ELP for each of the samples



(a) dt, country



(b) dt, dma



(c) dt, ended_flag

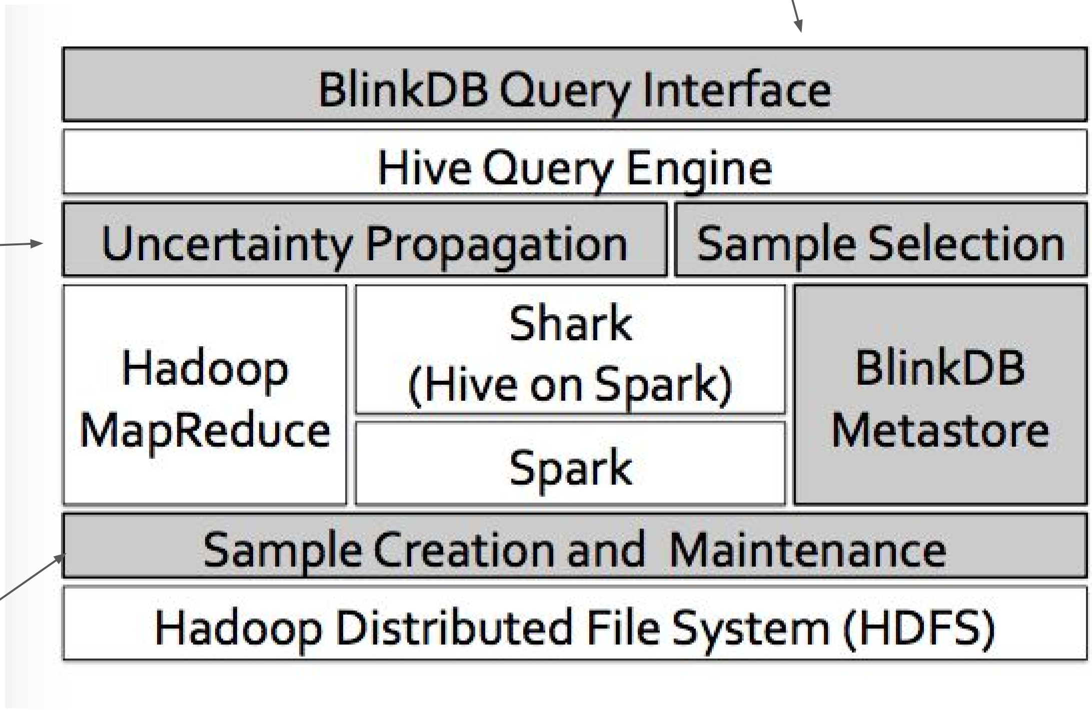
Implementation

enable queries with response time and error bounds

return error bars and confidence interval

assign query sized samples iteratively

create/update the set of random and multi-dimensional samples

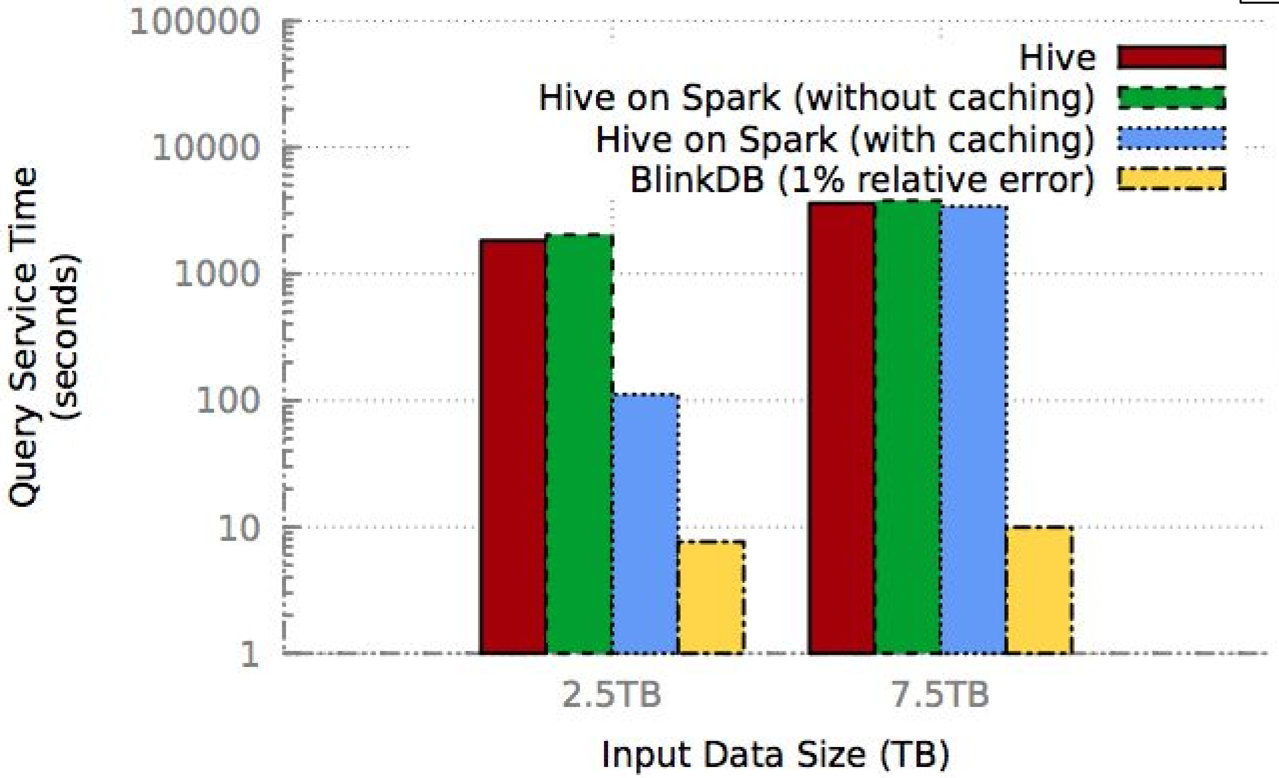


Evaluation Setting

- Conviva Workload
  - 17 TB in size
  - log of media accessed by Conviva users across 30 days
  - A sige big fact table with ~ 5.5 billion rows & 104 columns
  - raw query log constitutes 19,296 queries
- TPC-H workload
  - 1 TB of data
  - 22 benchmark queries
- For both of the workloads
  - partitioned data across 100 nodes
  - 50% storage budget

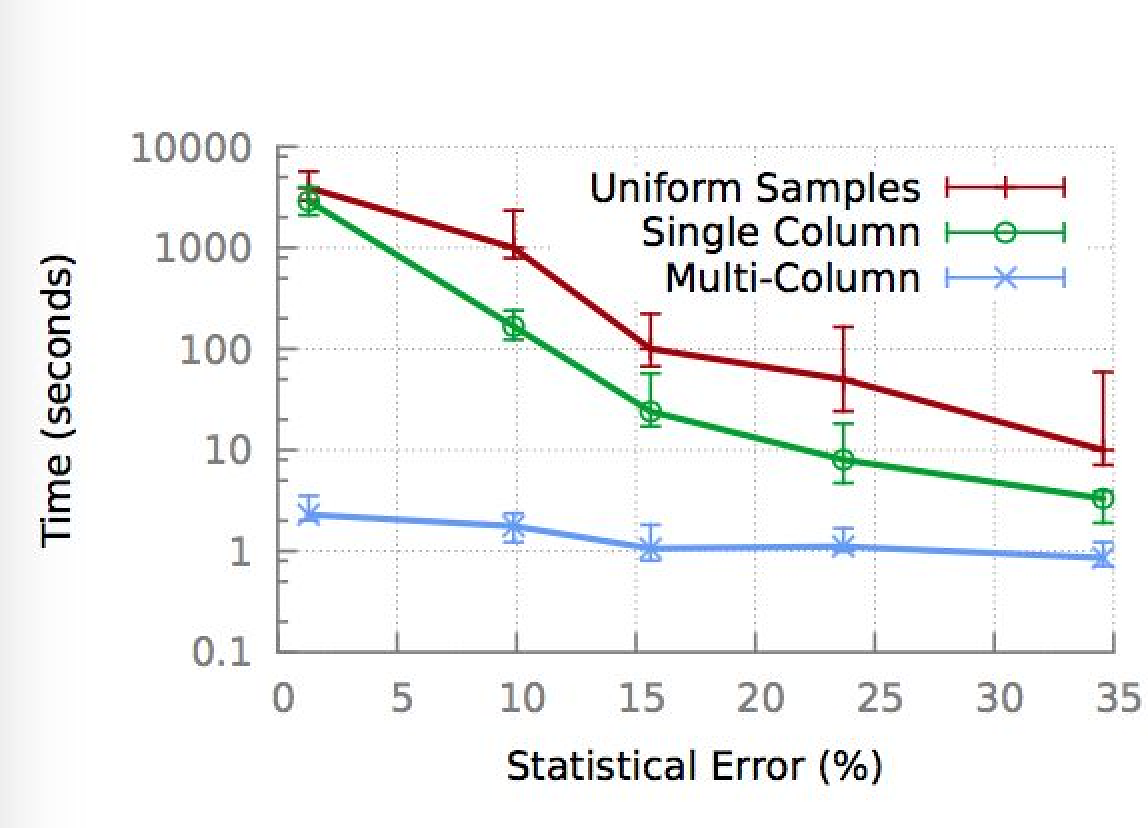
BlinkDB v.s. No Sampling

```
SELECT AVG(Session_Time)
FROM Sessions
WHERE date = …
GROUP BY City
```



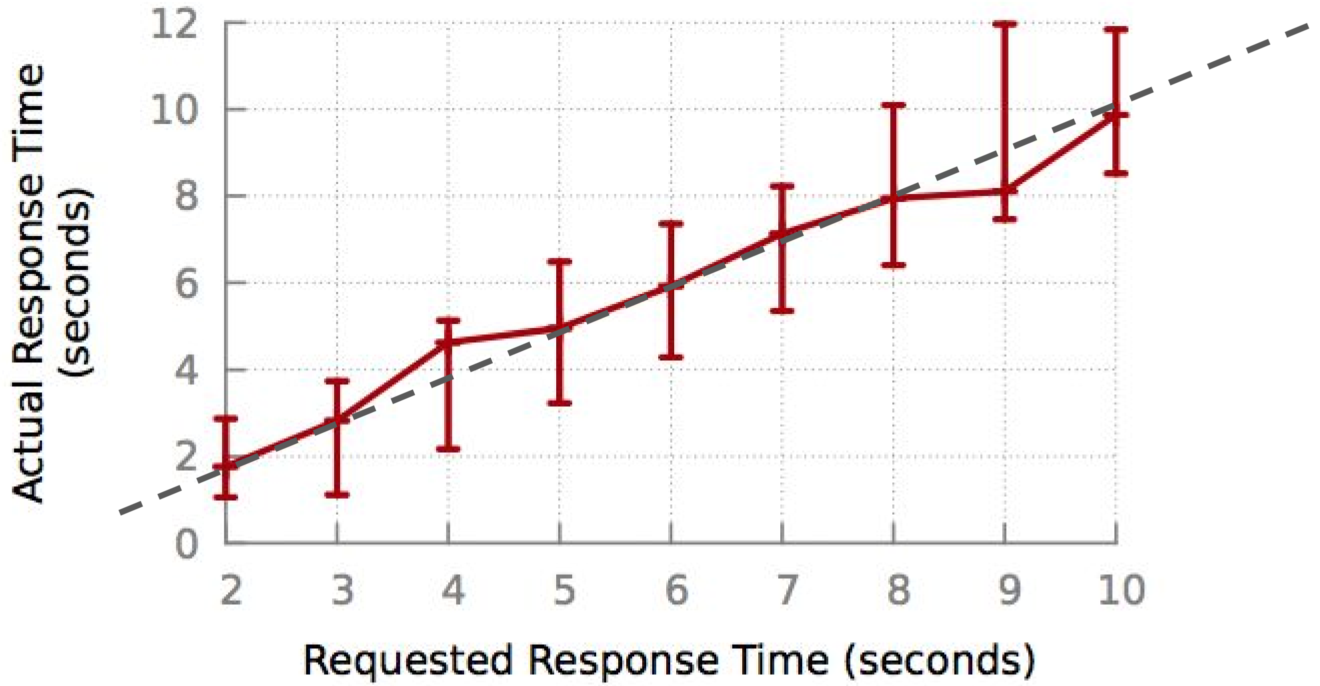
Response time v.s. Error

- Uniform samples: 50% of entire data
- Single Column: stratified on 1 column
- Multi-Column: stratifies on <= 3 columns



Time Guarantees

sample of 20 Conviva queries
ran each of them 10 times
on 17 TB data set



Error Guarantees

sample of 20 Conviva queries
ran each of them 10 times
on 17 TB data set