VerdictDB

Universalizing Approximate Query Processing

Yongjoo Park
Barzan Mozafari
Joseph Sorenson
Junhao Wang

CSE Computer Science and Engineering, University of Michigan

Universal

Approximate Query Processing

VerdictDB

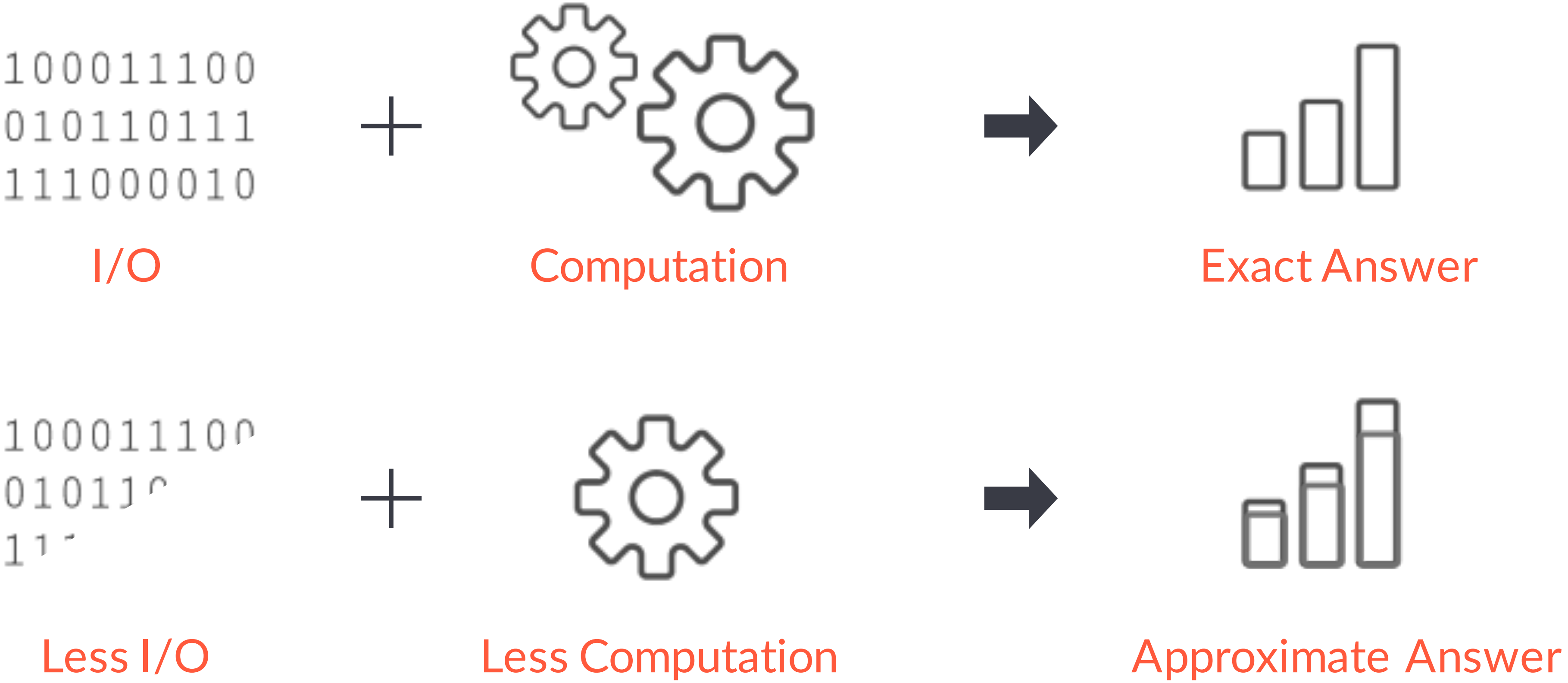
Universal

Approximate Query Processing

What is Approximate Query Processing (AQP)?

I/O + Computation → Exact Answer

What is Approximate Query Processing (AQP)?



Why AQP?

Higher Productivity

Numerous studies:

A latency >2 seconds is no longer interactive and negatively affects creativity!

Why AQP?

Higher Productivity

Numerous studies:

A latency >2 seconds is no longer interactive and negatively affects creativity!

Lower Cost (Time + Resources)

Human time: Money

Machine time: No one loves their EC2 bill!

Why AQP?

Higher Productivity

Numerous studies:

A latency >2 seconds is no longer interactive and negatively affects creativity!

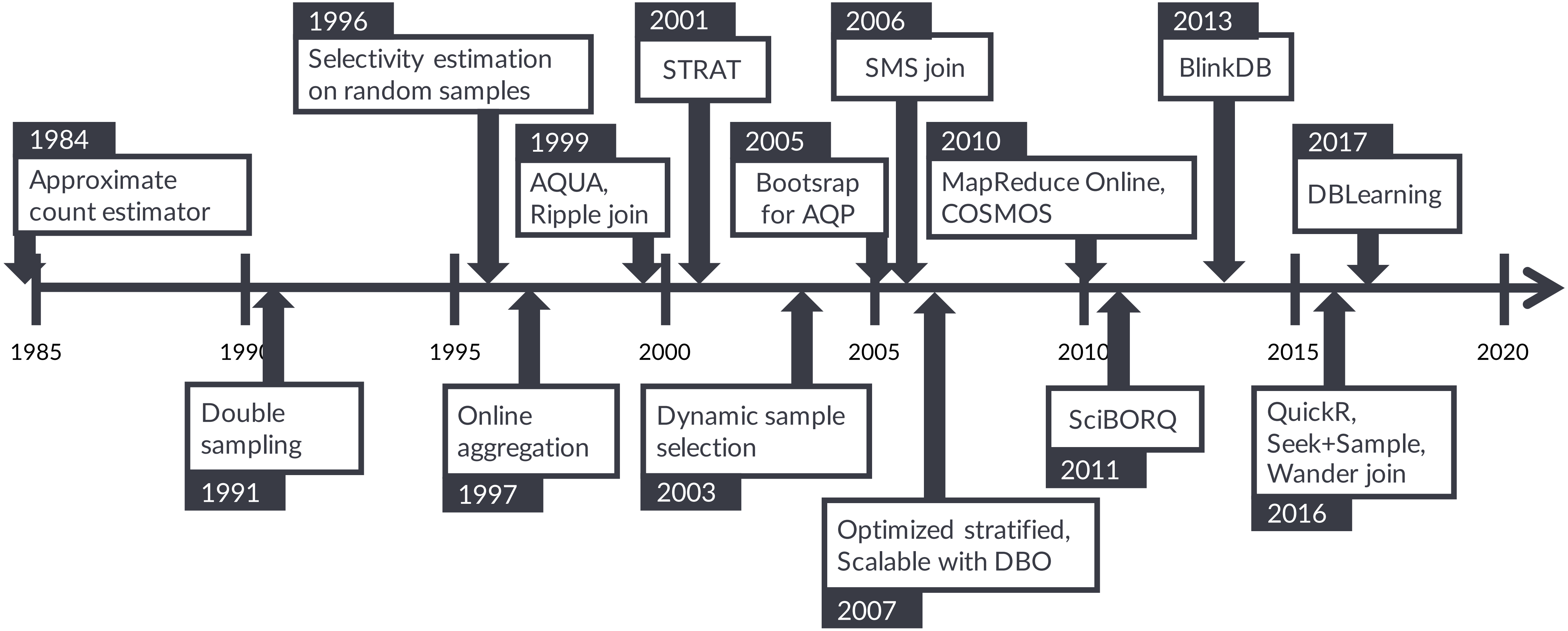
Lower Cost (Time + Resources)

Human time: Money

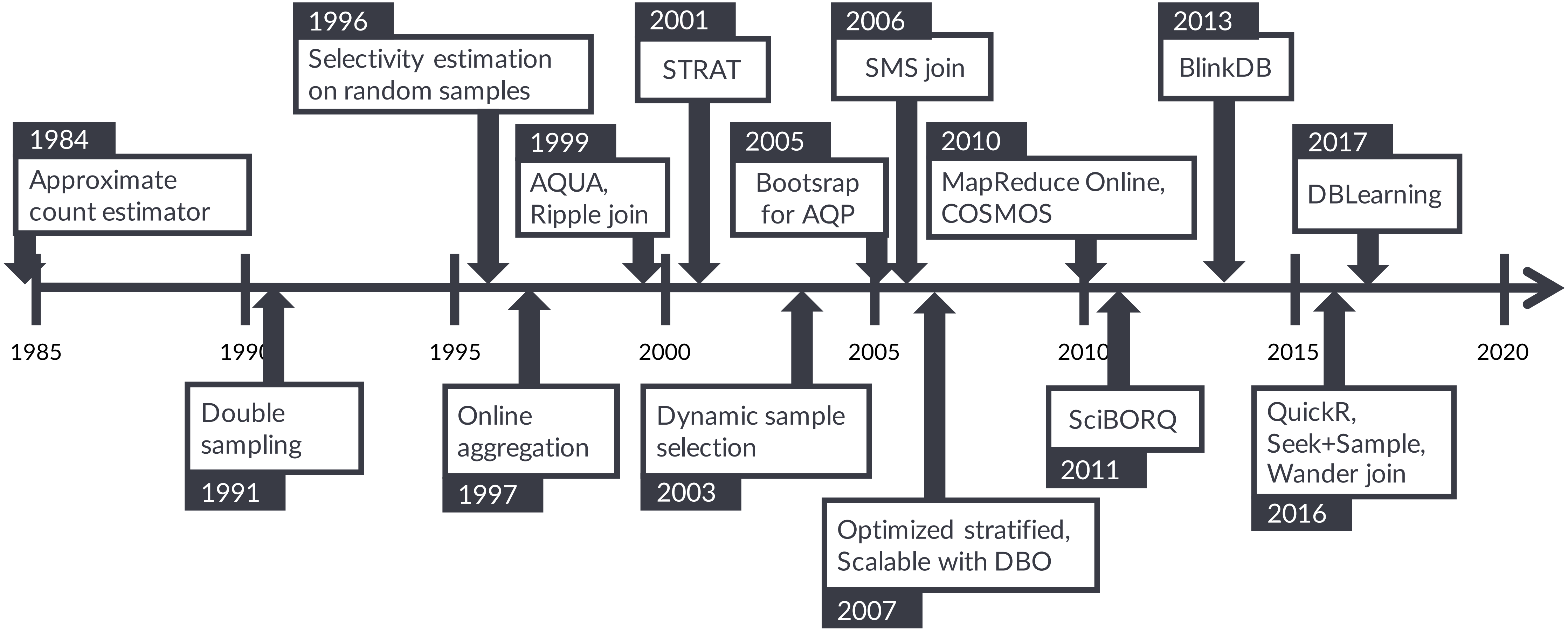
Machine time: No one loves their EC2 bill!

Jeff Bezos

AQP research in academia



AQP research in academia



35 years of research, little industry adoption

AQP is hard to adopt

AQP typically requires significant modifications of DBMS internals

- **Error estimation:** [BlinkDB ‘13], [G-OLA ’15], …
- **Query evaluation:** [Online ‘97], [Join Synopses ‘99], …
- **Relational operators:** [ABM ‘14], …

AQP is hard to adopt

AQP typically requires significant modifications of DBMS internals

- **Error estimation:** [BlinkDB '13], [G-OLA '15], …
- **Query evaluation:** [Online '97], [Join Synopses '99], …
- **Relational operators:** [ABM '14], …

Traditional DBMS vendors

- Stable codebase, reluctant to make major changes
- Slow in adopting ANYTHING :-)

AQP is hard to adopt

AQP typically requires significant modifications of DBMS internals

- **Error estimation:** [BlinkDB '13], [G-OLA '15], …
- **Query evaluation:** [Online '97], [Join Synopses '99], …
- **Relational operators:** [ABM '14], …

Traditional DBMS vendors

- Stable codebase, reluctant to make major changes
- Slow in adopting ANYTHING :-)

Newer SQL-on-Hadoop systems: implementing standard features

AQP is hard to adopt

AQP typically requires significant modifications of DBMS internals

- **Error estimation:** [BlinkDB '13], [G-OLA '15], …
- **Query evaluation:** [Online '97], [Join Synopses '99], …
- **Relational operators:** [ABM '14], …

Traditional DBMS vendors

- Stable codebase, reluctant to make major changes
- Slow in adopting ANYTHING :-)

Newer SQL-on-Hadoop systems: implementing standard features

Users won't abandon their existing DBMS just to use AQP.

Built-in AQP functions in OLAP engines

amazon REDSHIFT

APPROXIMATE
PERCENTILE_DISC

ORACLE

approx_count_distinct
approx_percentile

APACHE Spark

approxCountDistinct
approxQuantile

cloudera IMPALA

NDV
APX_MEDIAN

count-distinct or quantile

Built-in AQP functions in OLAP engines

APPROXIMATE
PERCENTILE_DISC

approx_count_distinct
approx_percentile

approxCountDistinct
approxQuantile

NDV
APX_MEDIAN

count-distinct or quantile

Good progress!
But, too little, too slow

Built-in AQP functions in OLAP engines

amazon REDSHIFT

APPROXIMATE
PERCENTILE_DISC

ORACLE

approx_count_distinct
approx_percentile

APACHE Spark

approxCountDistinct
approxQuantile

cloudera IMPALA

NDV
APX_MEDIAN

count-distinct or quantile

Good progress!
But, too little, too slow

Limitations

1. Good only when the data does not fit in memory
2. Good only for flat queries: *no error propagation*
3. Applicable only for order statistics: *no support for UDAs or arithmetic aggregates*

Built-in AQP functions in OLAP engines

APPROXIMATE
PERCENTILE_DISC

approx_count_distinct
approx_percentile

approxCountDistinct
approxQuantile

NDV
APX_MEDIAN

count-distinct or quantile

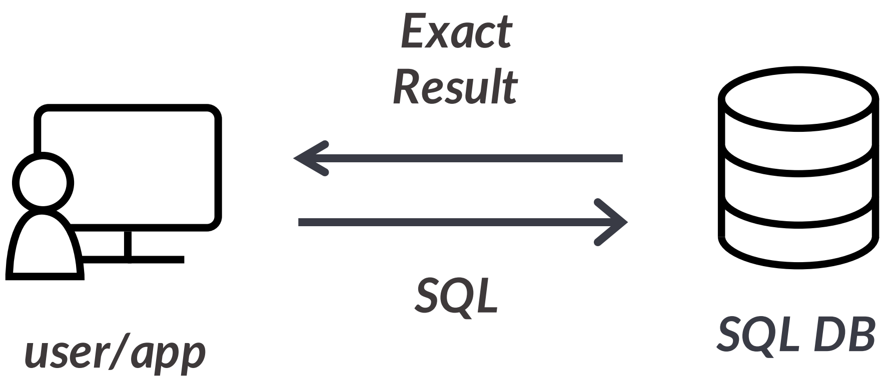
Good progress!
But, too little, too slow

Limitations

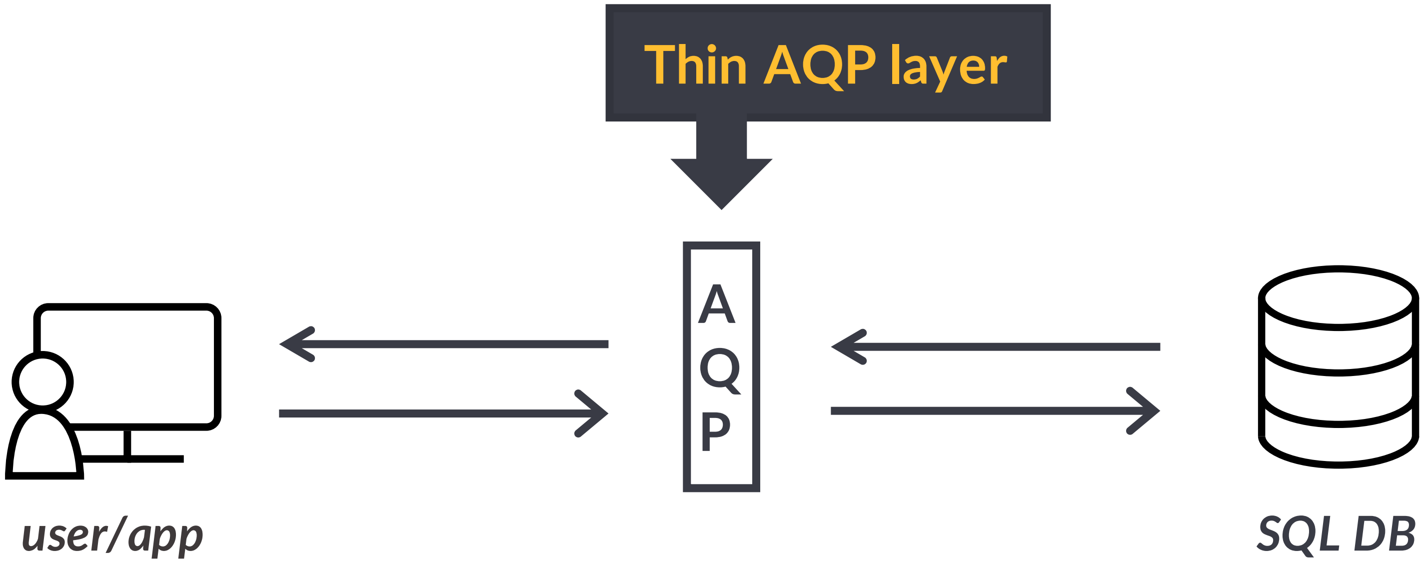
1. Good only when the data does not fit in memory
2. Good only for flat queries: *no error propagation*
3. Applicable only for order statistics: *no support for UDAs or arithmetic aggregates*

Need for complete AQP solutions that are easy to adopt

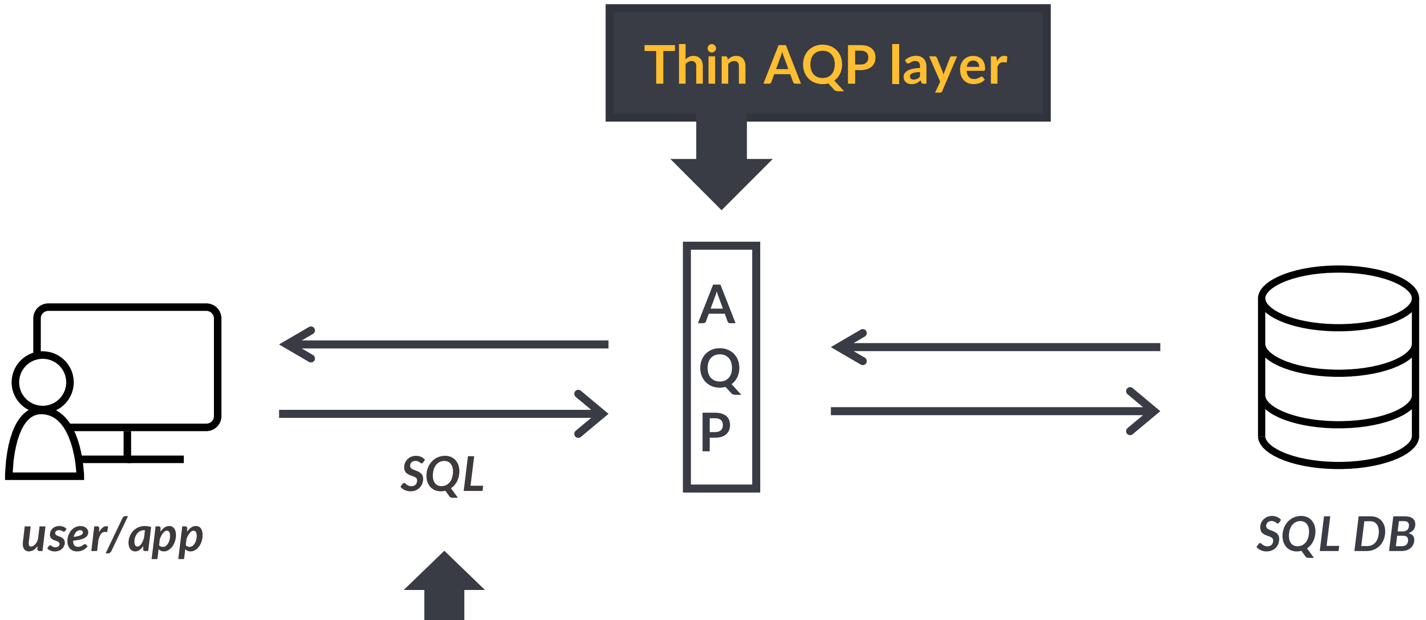
Our proposal: Universal AQP



Our proposal: Universal AQP

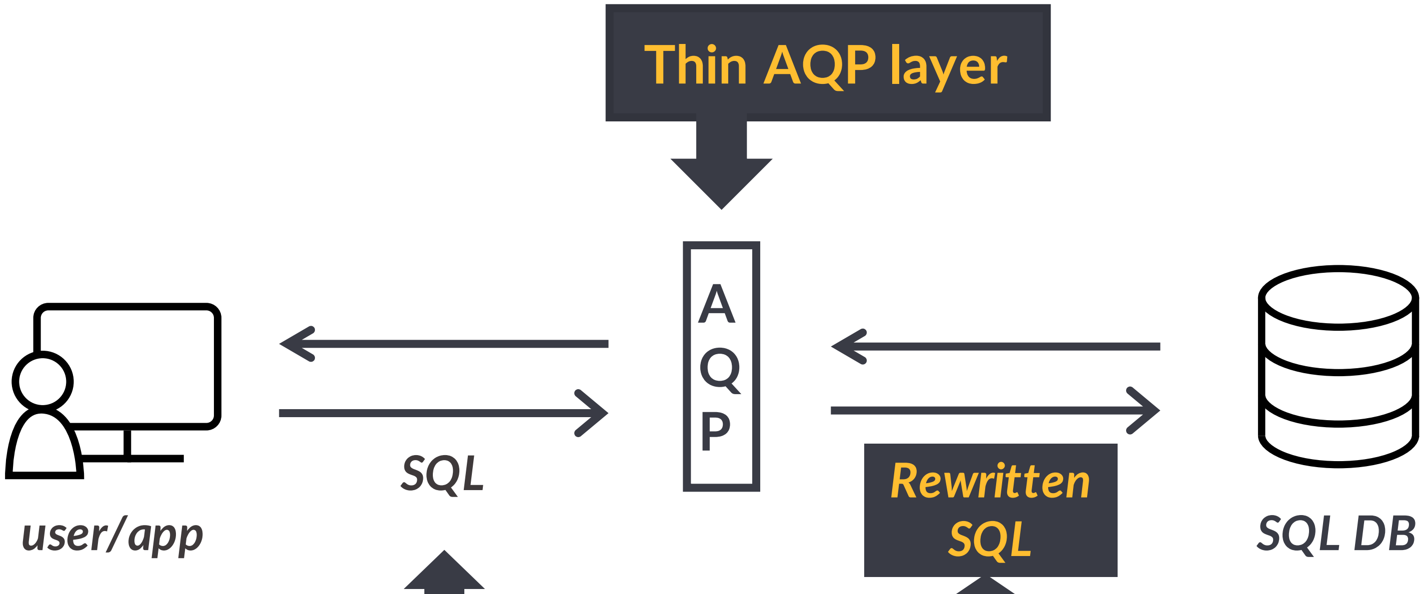


Our proposal: Universal AQP




```
select avg(price)
from sales
where channel = 'online'
```

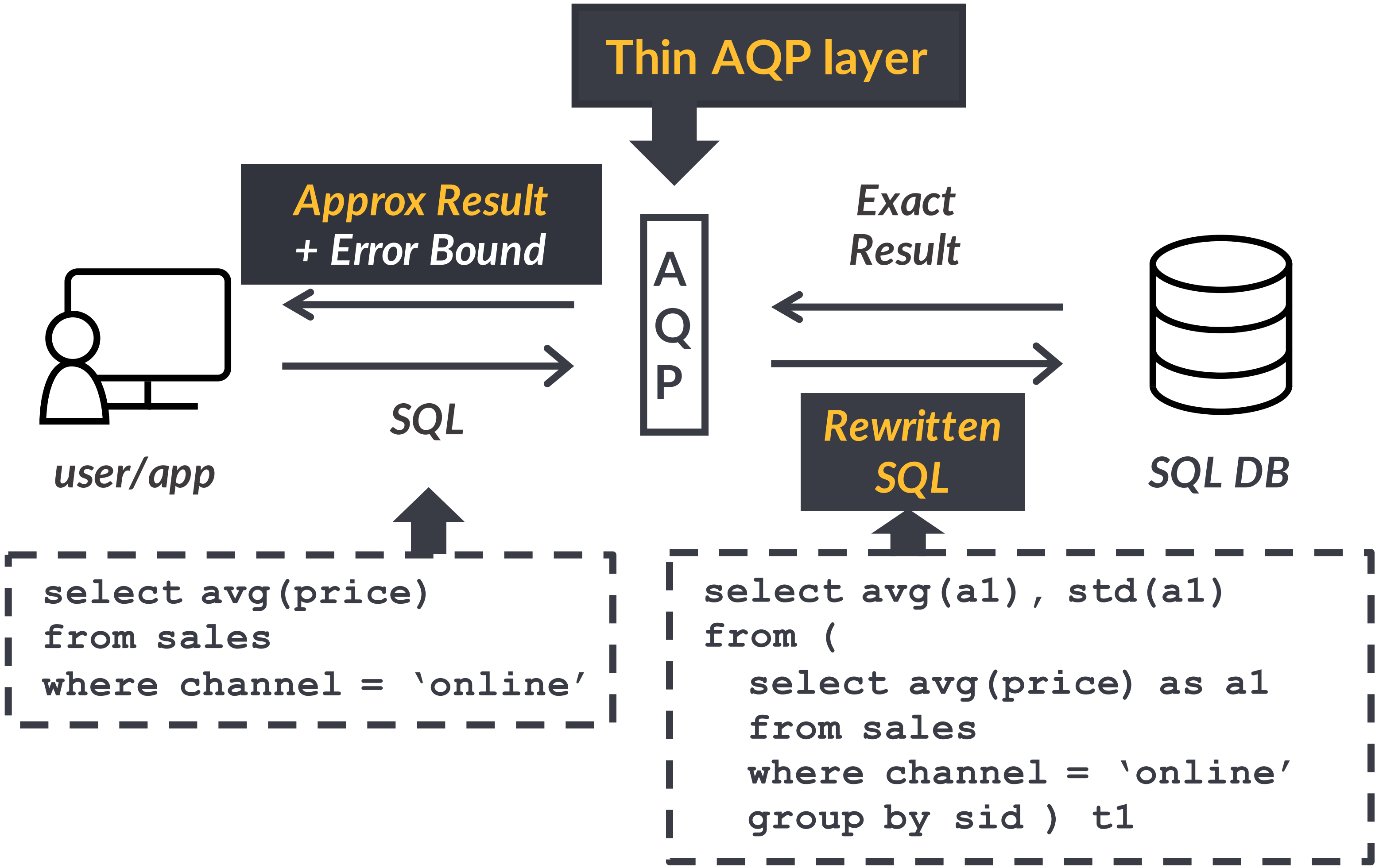
Our proposal: Universal AQP



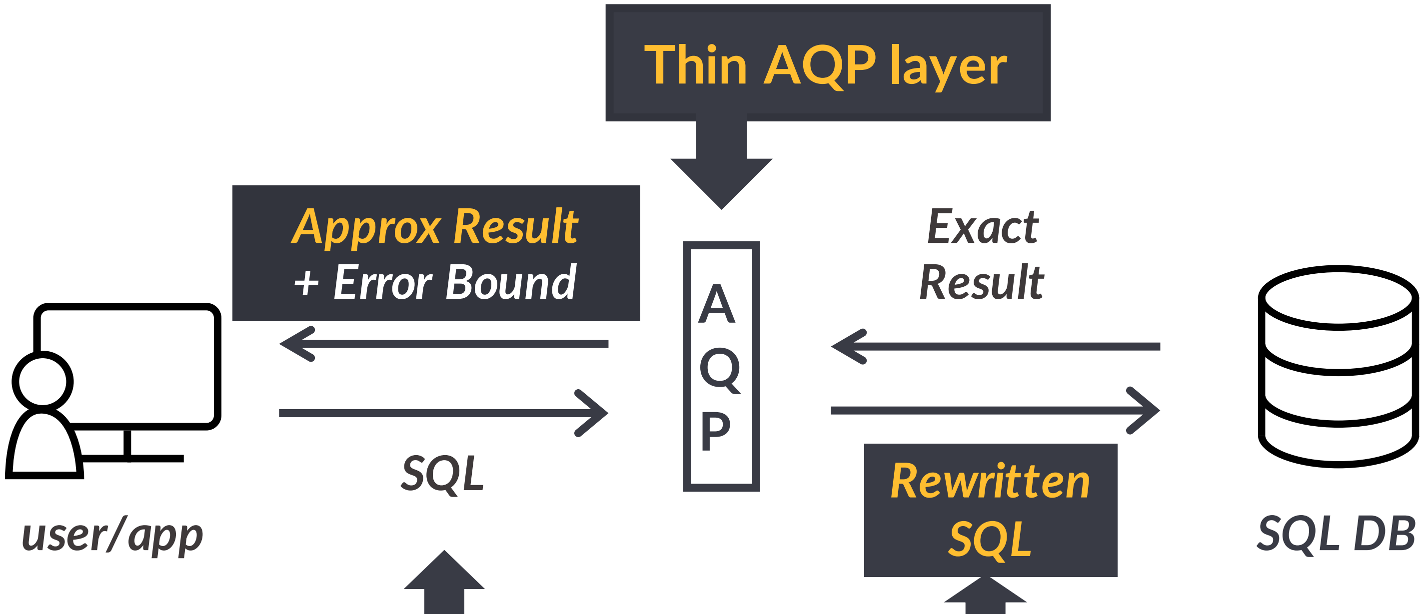

```
select avg(price)
from sales
where channel = 'online'
```

```
select avg(a1), std(a1)
from (
  select avg(price) as a1
  from sales
  where channel = 'online'
  group by sid ) t1
```

Our proposal: Universal AQP



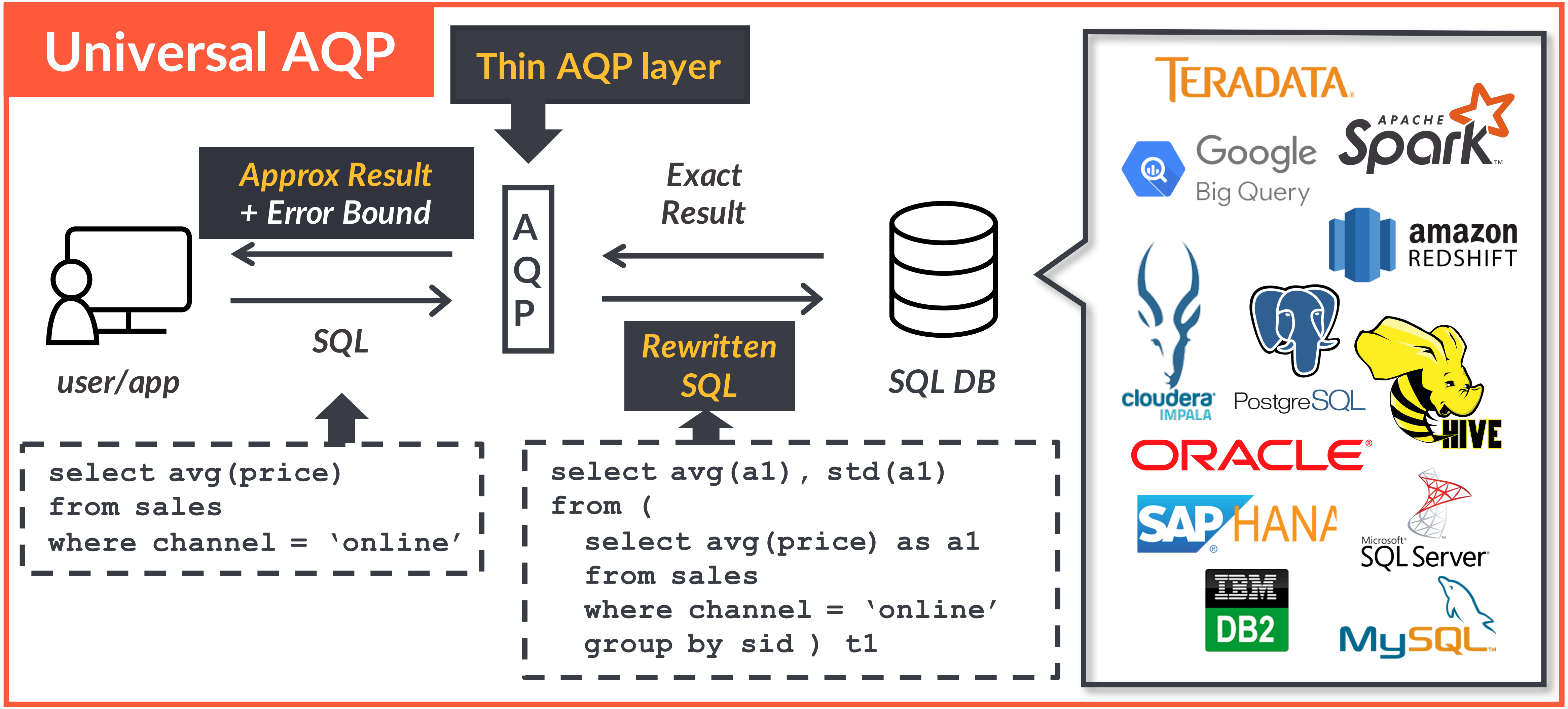
Our proposal: Universal AQP




```
select avg(price)
from sales
where channel = 'online'
```

```
select avg(a1), std(a1)
from (
  select avg(price) as a1
  from sales
  where channel = 'online'
  group by sid ) t1
```

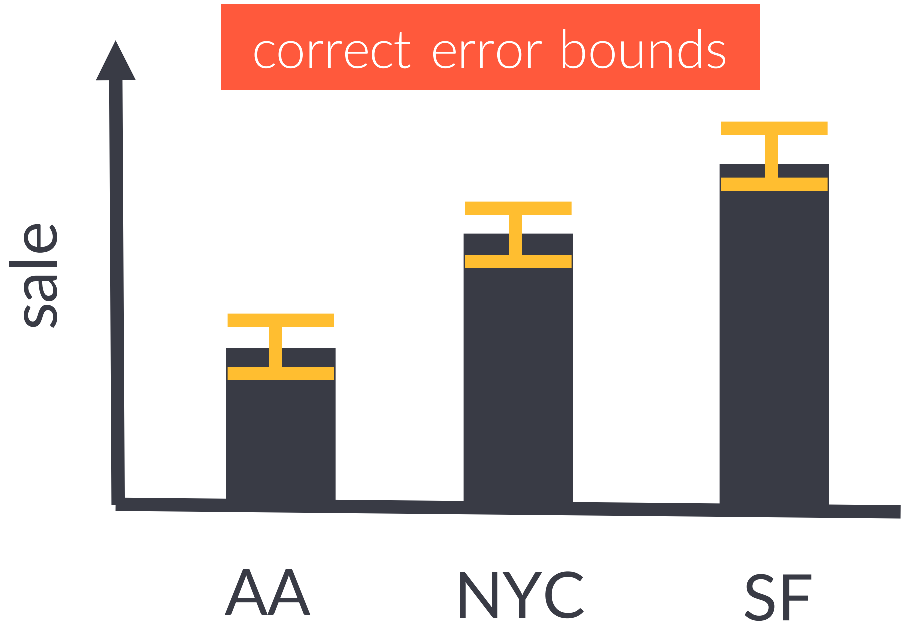
Our proposal: Universal AQP



Challenges of Universal AQP

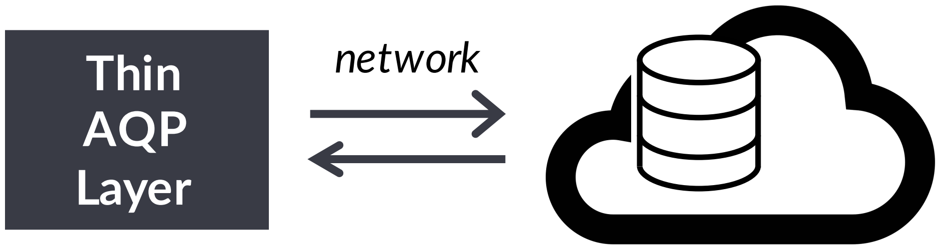
Challenges of Universal AQP

1. **Statistical correctness (inter-tuple correlations)**
    - Foreign-key constraints [Join Synopses '99]
    - Modifying join algorithm [Wander Join '16]
    - Modifying the query plan [Quickr '16]



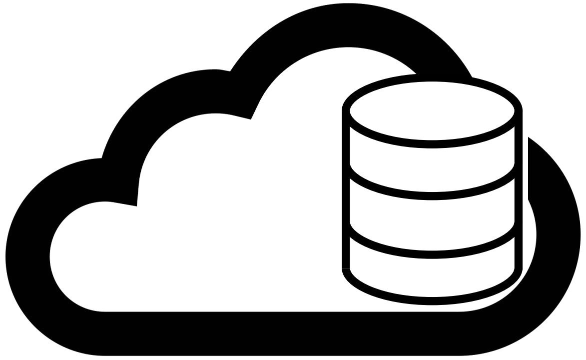
Challenges of Universal AQP

1. Statistical correctness (inter-tuple correlations)
    - Foreign-key constraints [Join Synopses '99]
    - Modifying join algorithm [Wander Join '16]
    - Modifying the query plan [Quickr '16]
2. Middleware efficiency
    - Lack of access to DBMS machinery



Challenges of Universal AQP

1. **Statistical correctness (inter-tuple correlations)**
   - Foreign-key constraints [Join Synopses '99]
   - Modifying join algorithm [Wander Join '16]
   - Modifying the query plan [Quickr '16]
2. **Middleware efficiency**
   - Lack of access to DBMS machinery
3. **Server efficiency**
   - Resampling-based techniques [Pol and Jermaine '05, BlinkDB '14]
   - Intimate integration of err est. logic into scan operators [Quickr '16, SnappyData]
   - Overriding the relational operators altogether [ABM '14]

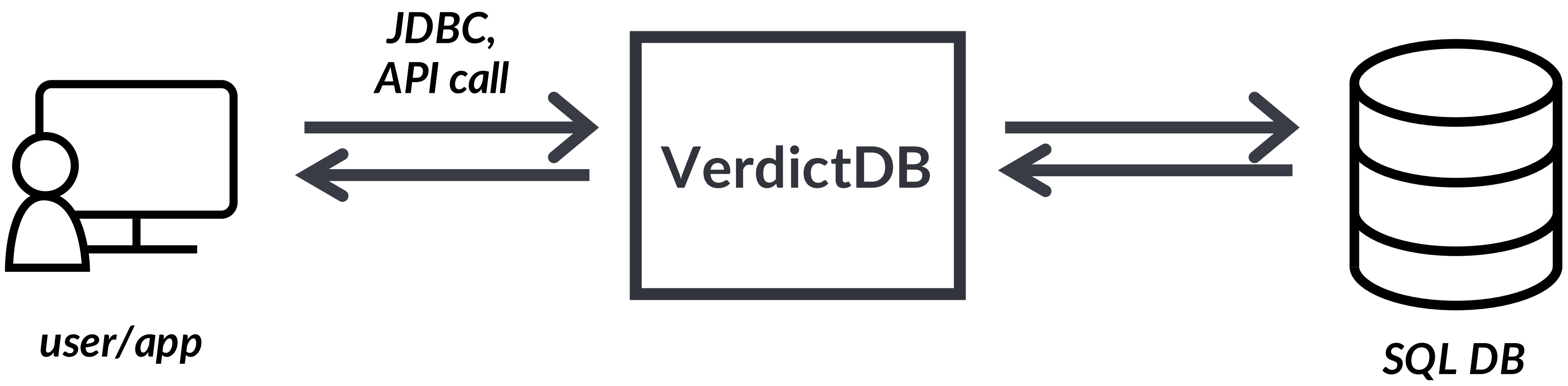


VerdictDB Overview

First Universal AQP system

Deployment

Deployment



Deployment

Deployment

Stores (1) offline-created samples, and (2) VerdictDB-managed metadata

Deployment

Stores (1) offline-created samples, and (2) VerdictDB-managed metadata

The only requirements:

- create table as select ...
- rand(), agg(col) over (partition by ...)

Deployment

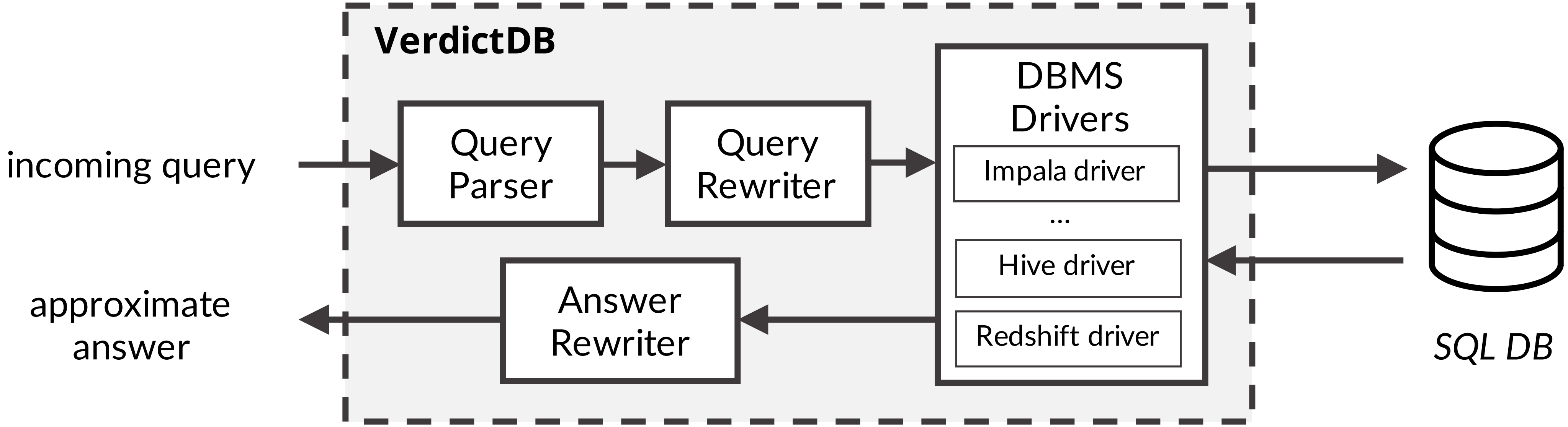
Stores (1) offline-created samples, and (2) VerdictDB-managed metadata

The only requirements:

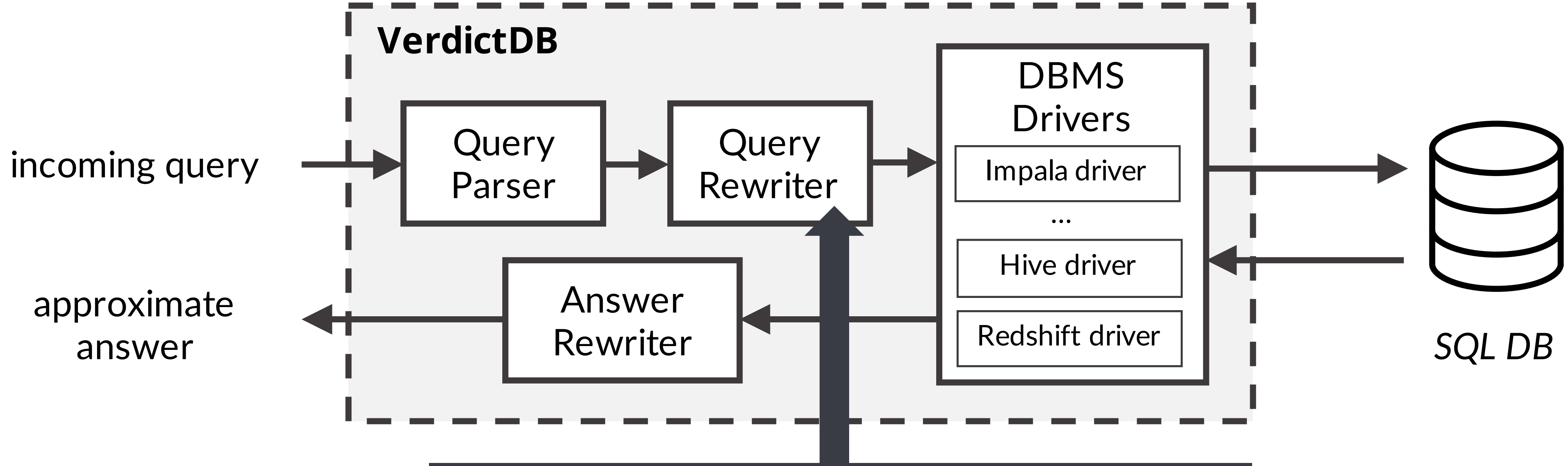
- create table as select …
- rand(), agg(col) over (partition by …)

supported by almost any SQL engines

Architecture



Architecture



Crucial component

1. Chooses an optimal set of samples
2. Scales values appropriately
3. **Inserts an error estimation logic**

Error estimation in VerdictDB

Error estimation in general

User interested in $Q(T)$

We compute $Q(S)$ where S is a sample of T

Error estimation in general

User interested in $Q(T)$

We compute $Q(S)$ where S is a sample of T

Main question: how close is $Q(S)$ to $Q(T)$?

Error estimation in general

User interested in $Q(T)$

We compute $Q(S)$ where S is a sample of T

Main question: how close is $Q(S)$ to $Q(T)$?

	Fast?	**General?**
Closed-form (CLT, Hoeffding, HT)		
Existing Resampling (subsampling, bootstrap)		
Ours (variational subsampling)		

Error estimation in general

User interested in $Q(T)$

We compute $Q(S)$ where S is a sample of T

Main question: how close is $Q(S)$ to $Q(T)$?

	Fast?	**General?**
Closed-form (CLT, Hoeffding, HT)	**YES**	NO (no UDAs)
Existing Resampling (subsampling, bootstrap)		
Ours (variational subsampling)		

Error estimation in general

User interested in $Q(T)$

We compute $Q(S)$ where S is a sample of T

Main question: how close is $Q(S)$ to $Q(T)$?

	Fast?	**General?**
Closed-form (CLT, Hoeffding, HT)	**YES**	NO (no UDAs)
Existing Resampling (subsampling, bootstrap)	NO (can be slow in SQL)	**YES** (Hadamard differentiable)
Ours (variational subsampling)		

Error estimation in general

User interested in $Q(T)$

We compute $Q(S)$ where S is a sample of T

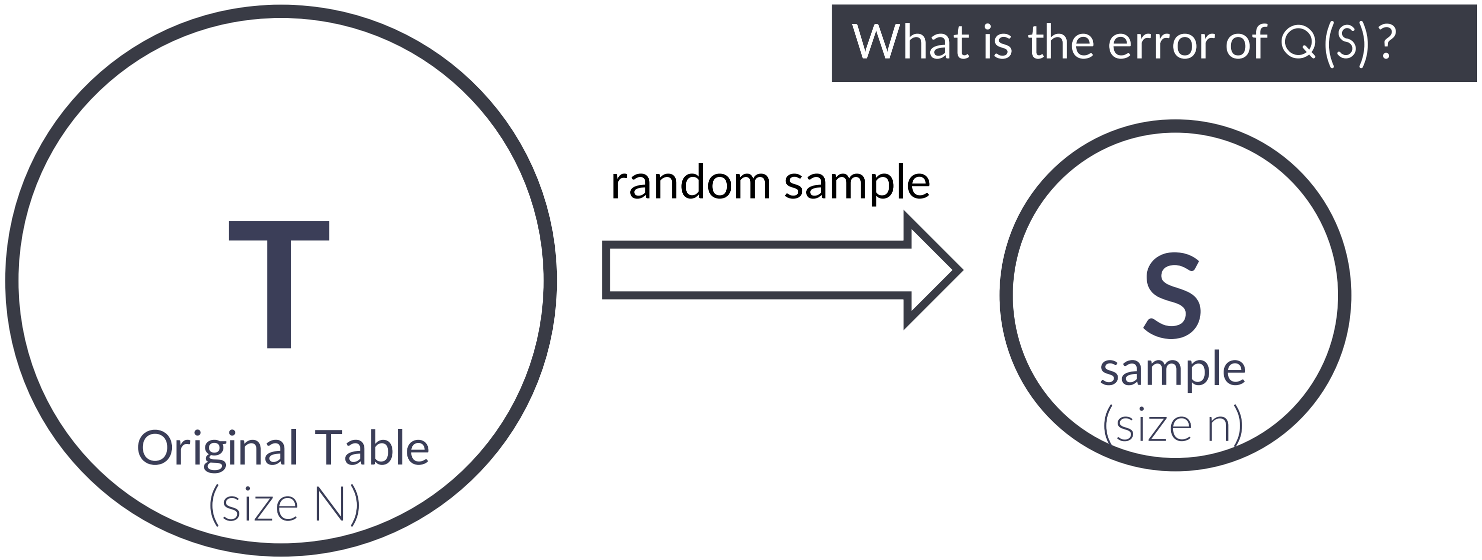
Main question: how close is $Q(S)$ to $Q(T)$?

	Fast?	**General?**
Closed-form (CLT, Hoeffding, HT)	**YES**	NO (no UDAs)
Existing Resampling (subsampling, bootstrap)	NO (can be slow in SQL)	**YES** (Hadamard differentiable)
Ours (variational subsampling)	**YES**	**YES** (Hadamard differentiable)

Recap: traditional subsampling

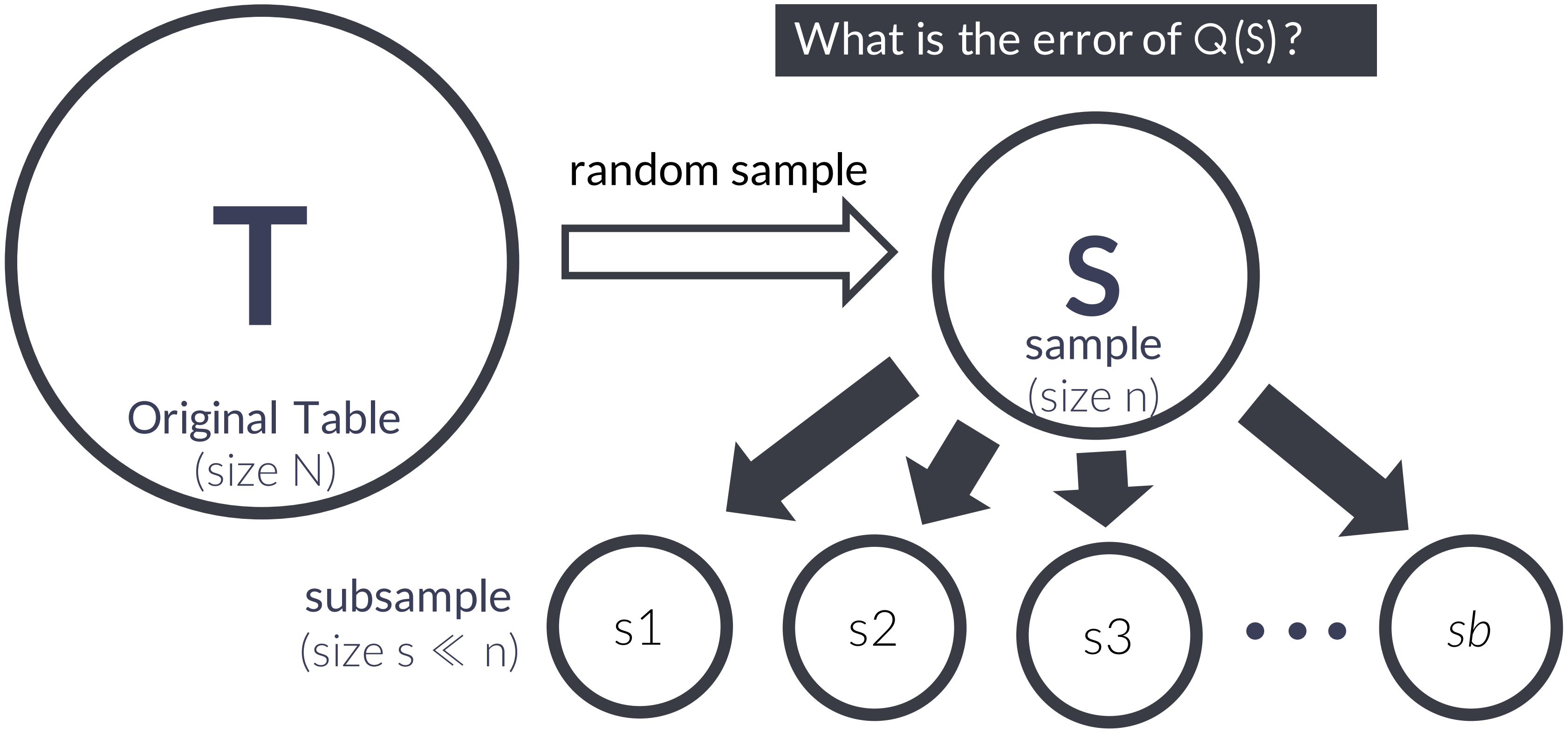
Q(T) is slow / expensive

Recap: traditional subsampling



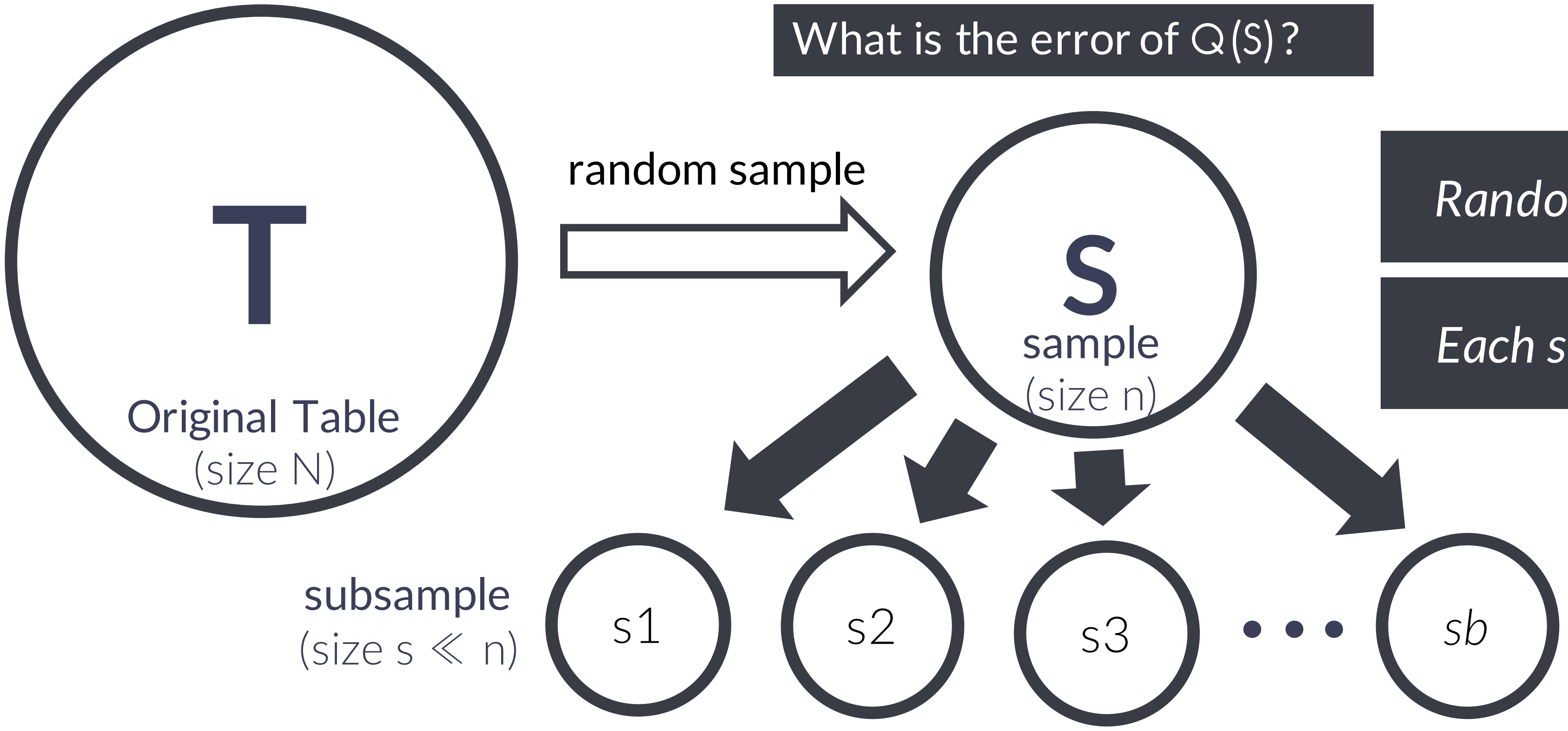
Recap: traditional subsampling

What is the error of Q(S)?



Recap: traditional subsampling

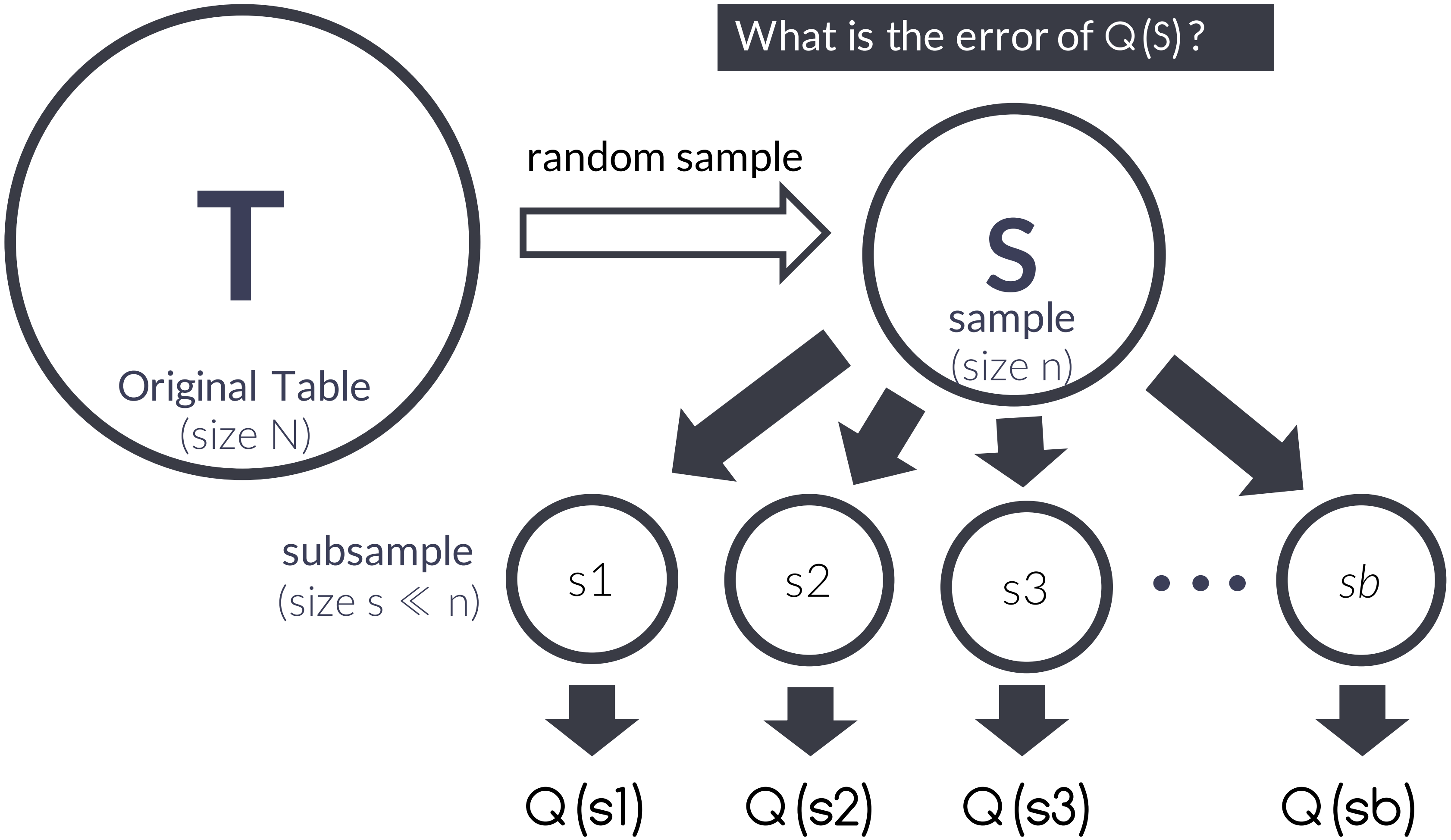
What is the error of Q(S)?



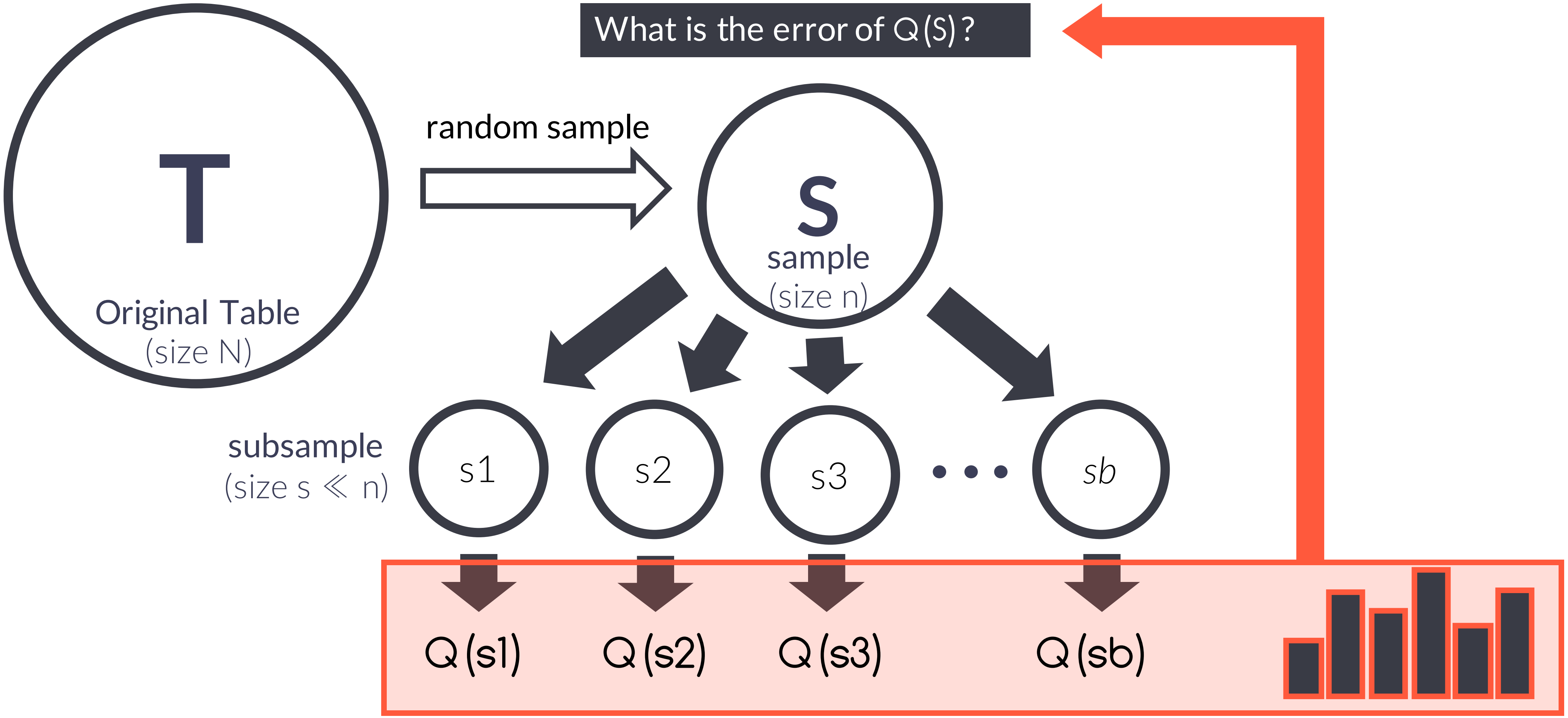
Random sample without replacement

Each subsample is independent

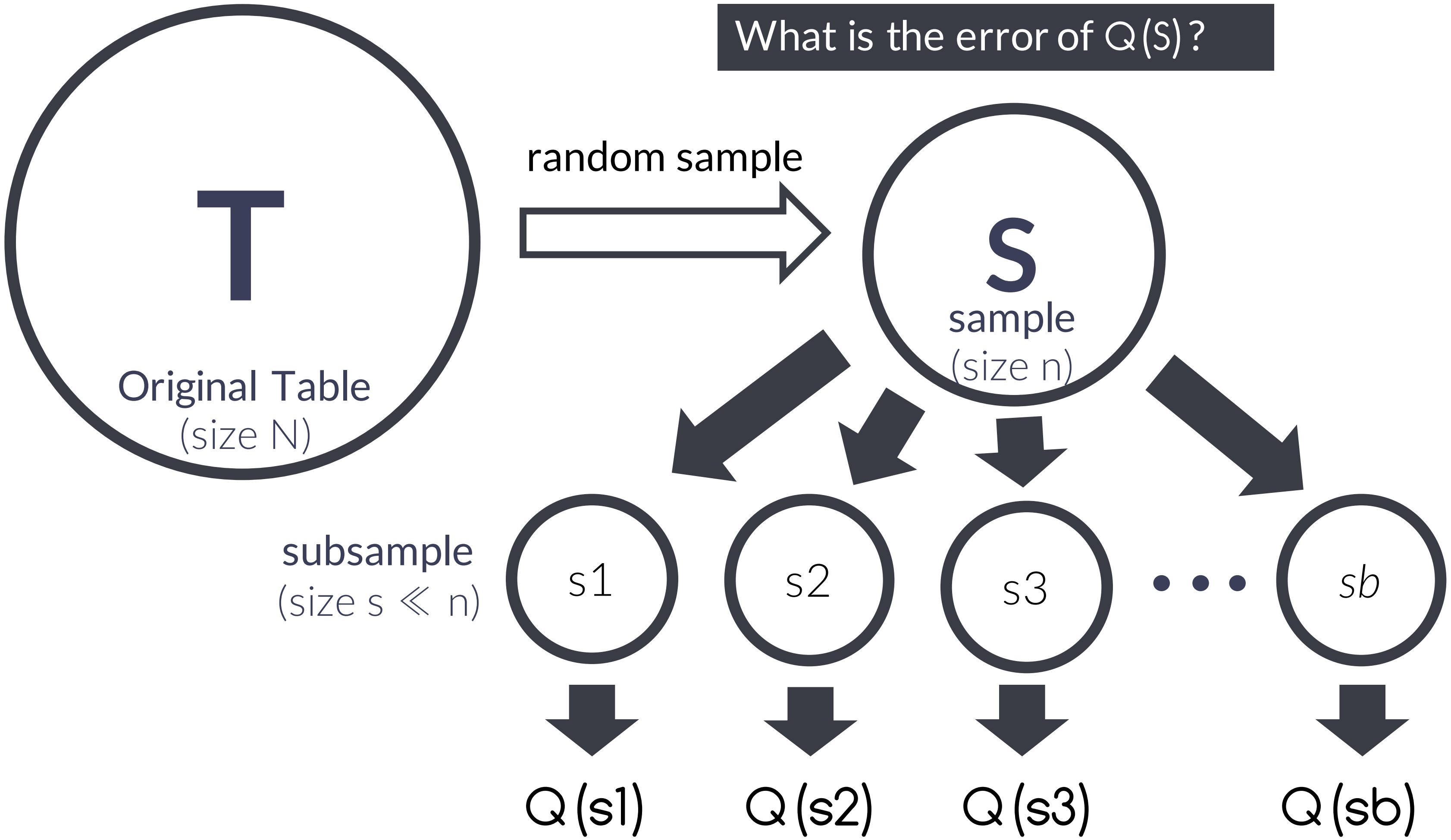
Recap: traditional subsampling



Recap: traditional subsampling



Recap: traditional subsampling



Important properties

1. A tuple may belong to multiple subsamples.
2. The size of every subsample is s.

Traditional subsampling in SQL is slow

subsample ID

n tuples

CITY	PRODUCT	PRICE	1	2	···	b
AA	egg	$3.00	1	0		1
AA	milk	$5.00	0	1		0
AA	egg	$3.00	0	0		1
NYU	egg	$4.00	0	1		0
NYU	milk	$6.00	0	0		1
NYU	candy	$2.00	1	0		0
SF	milk	$6.00	0	1		0
SF	egg	$4.00	0	0		0
SF	egg	$4.00	0	1		1
			sum = s			*sum = s*

Traditional subsampling in SQL is slow

subsample ID

n tuples

CITY	PRODUCT	PRICE	1	2	...	b
AA	egg	$3.00	1	0		1
AA	milk	$5.00	0	1		0
AA	egg	$3.00	0	0		1
NYU	egg	$4.00	0	1		0
NYU	milk	$6.00	0	0		1
NYU	candy	$2.00	1	0		0
SF	milk	$6.00	0	1		0
SF	egg	$4.00	0	0		0
SF	egg	$4.00	0	1		1
			sum = s			*sum = s*

```
Algorithm:
for i = 1, ..., n
  for j = 1, ..., b
    if sid[i,j] == 1
      sum[j] += price[i]
```

Traditional subsampling in SQL is slow

CITY	PRODUCT	PRICE	1	2	...	b
AA	egg	$3.00	1	0		1
AA	milk	$5.00	0	1		0
AA	egg	$3.00	0	0		1
NYU	egg	$4.00	0	1		0
NYU	milk	$6.00	0	0		1
NYU	candy	$2.00	1	0		0
SF	milk	$6.00	0	1		0
SF	egg	$4.00	0	0		0
SF	egg	$4.00	0	1		1

subsample ID

n tuples

sum = s   *sum = s*

```
Algorithm:
for i = 1, ..., n
  for j = 1, ..., b
    if sid[i,j] == 1
      sum[j] += price[i]
```

Time Complexity: O(n·b)

Traditional subsampling in SQL is slow

subsample ID

n tuples

CITY	PRODUCT	PRICE	1	2	···	b
AA	egg	$3.00	1	0		1
AA	milk	$5.00	0	1		0
AA	egg	$3.00	0	0		1
NYU	egg	$4.00	0	1		0
NYU	milk	$6.00	0	0		1
NYU	candy	$2.00	1	0		0
SF	milk	$6.00	0	1		0
SF	egg	$4.00	0	0		0
SF	egg	$4.00	0	1		1
			sum = s			*sum = s*

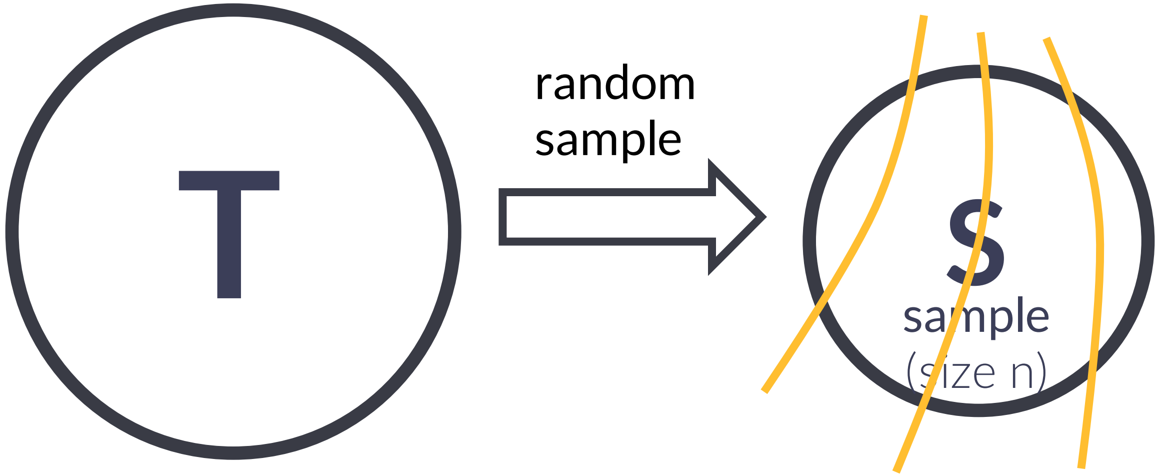
```
Algorithm:
for i = 1, ..., n
  for j = 1, ..., b
    if sid[i,j] == 1
      sum[j] += price[i]
```

Time Complexity: O(n·b)

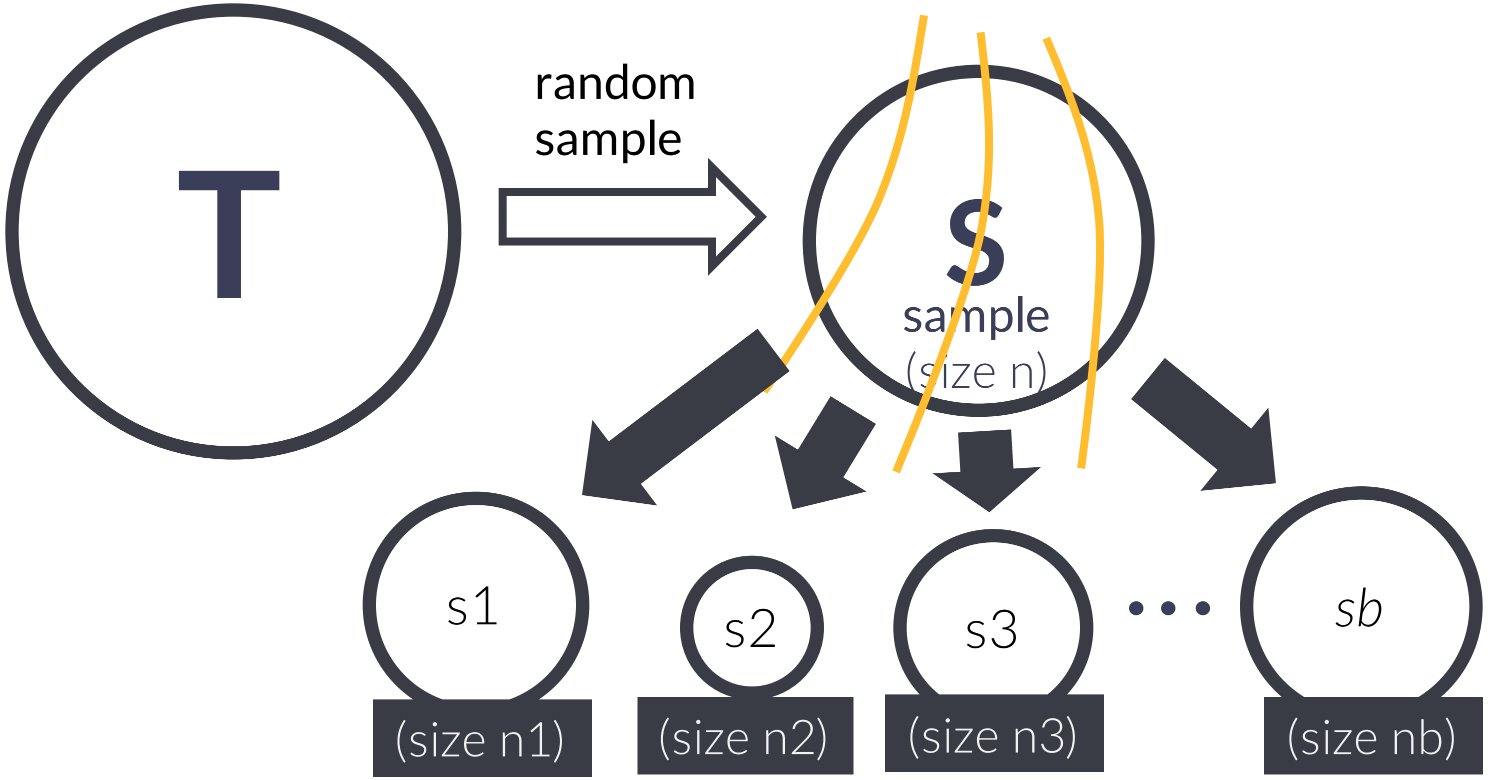
No error est: 0.35 sec
Trad. subsampling: 118 sec
337x slower

(based on 1G sample, Impala)

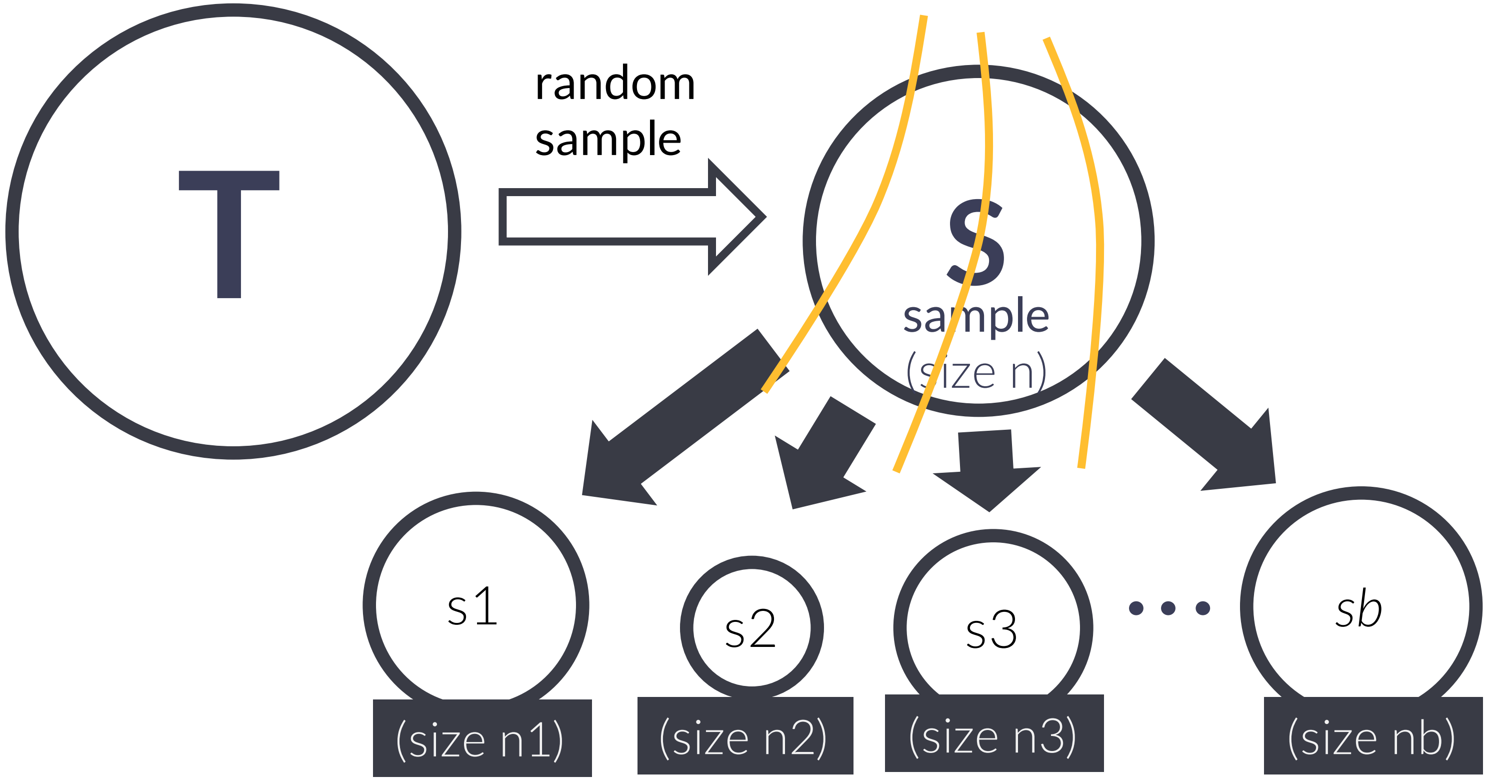
Our approach: variational subsampling



Our approach: variational subsampling



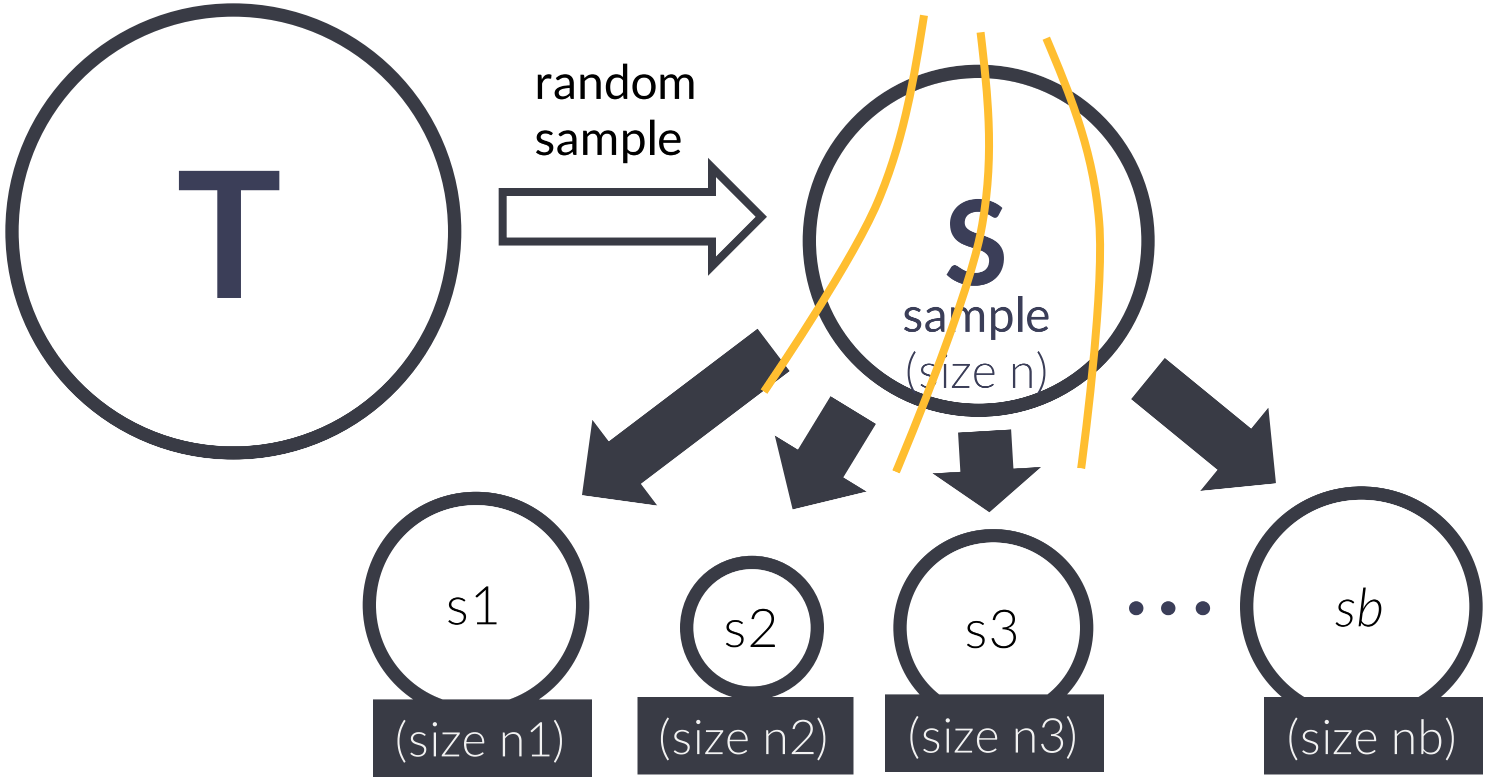
Our approach: variational subsampling



Important properties

1. A tuple may belong to multiple subsamples.
2. The size of every subsample is *s*.

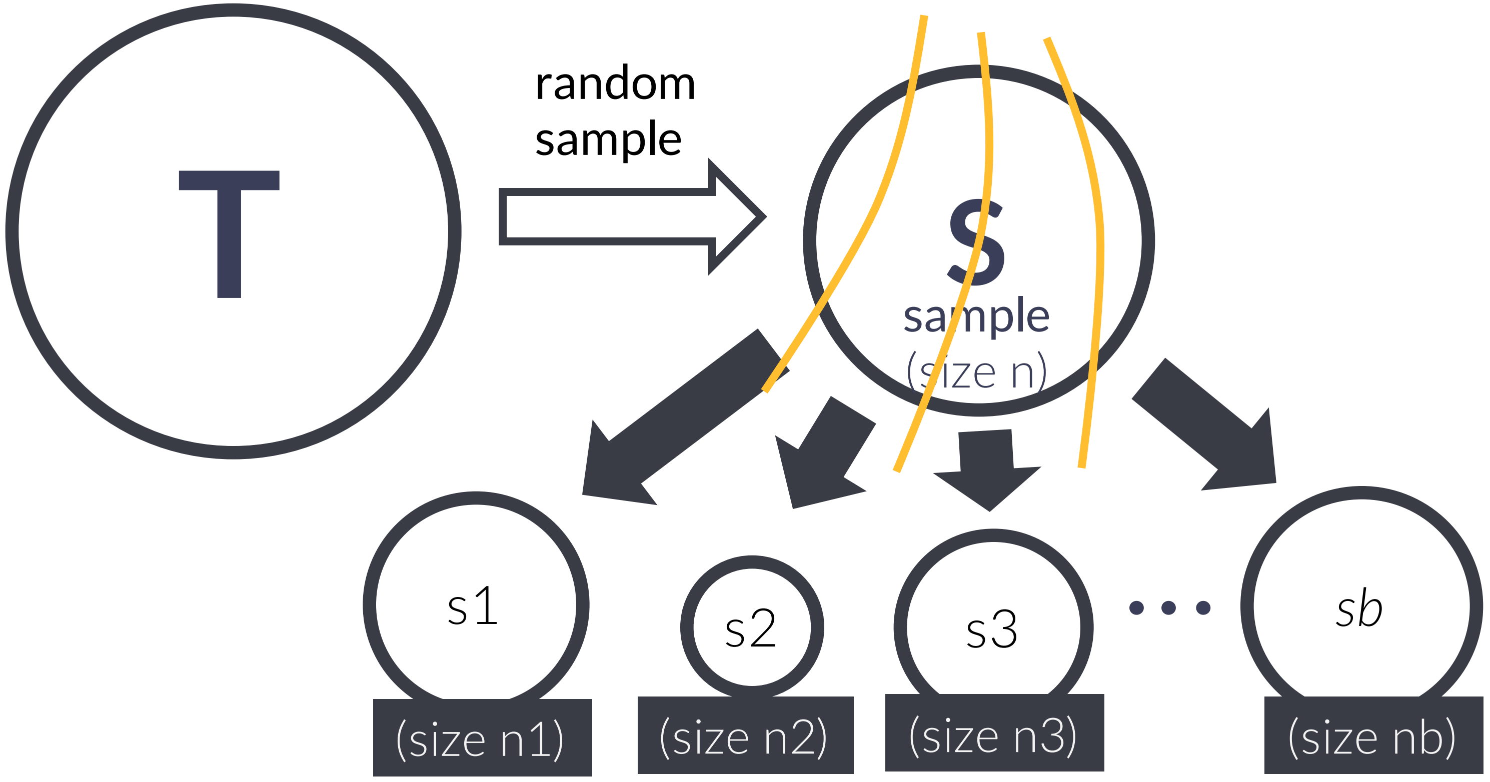
Our approach: variational subsampling



Important properties

1. ~~A tuple may belong to multiple subsamples.~~
   Each sampled tuple can belong to at most one subsample
2. The size of every subsample is *s*.

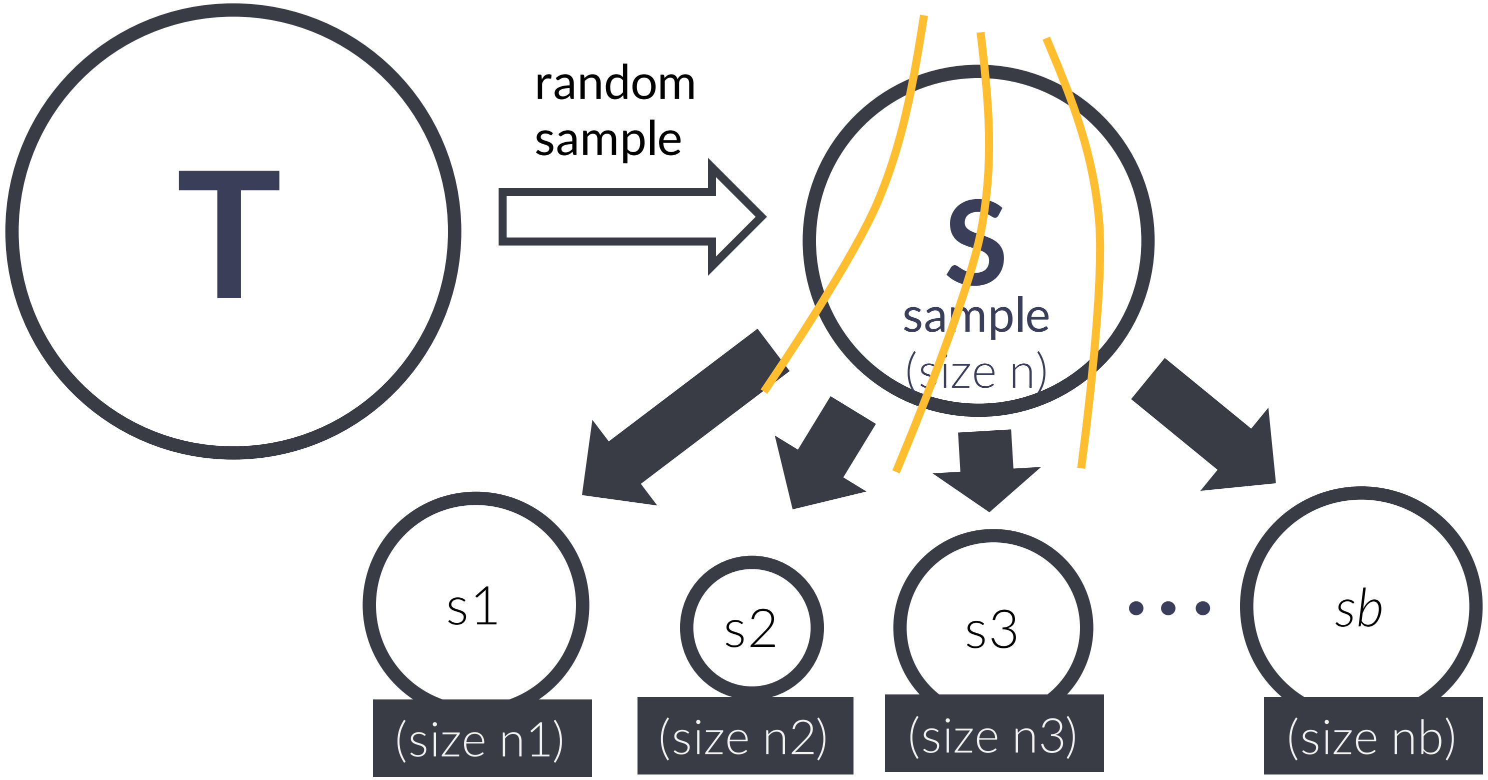
Our approach: variational subsampling



Important properties

1. ~~A tuple may belong to multiple subsamples.~~
   Each sampled tuple can belong to at most one subsample
2. ~~The size of every subsample is s.~~
   Allow subsamples to differ in size.

Our approach: variational subsampling



Important properties

1. ~~A tuple may belong to multiple subsamples.~~
   Each sampled tuple can belong to at most one subsample
2. ~~The size of every subsample is s.~~
   Allow subsamples to differ in size.

*Can be implemented in SQL as **a single group-by query!***

Variational subsampling in SQL is fast

CITY	PRODUCT	PRICE	subsample ID
AA	egg	\$3.00	1
AA	milk	\$5.00	3
AA	egg	\$3.00	2
NYU	egg	\$4.00	4
NYU	milk	\$6.00	3
NYU	candy	\$2.00	1
SF	milk	\$6.00	5
SF	egg	\$4.00	4
SF	egg	\$4.00	5

`randint(1,b)`

n tuples

We call this augmented table, a variational table

Variational subsampling in SQL is fast

CITY	PRODUCT	PRICE	subsample ID
AA	egg	$3.00	1
AA	milk	$5.00	3
AA	egg	$3.00	2
NYU	egg	$4.00	4
NYU	milk	$6.00	3
NYU	candy	$2.00	1
SF	milk	$6.00	5
SF	egg	$4.00	4
SF	egg	$4.00	5

n tuples

randint(1,b)

We call this augmented table, a variational table

```
Algorithm:
for i = 1, ..., n
  sum[sid] += price[i]
```

Variational subsampling in SQL is fast

CITY	PRODUCT	PRICE	subsample ID
AA	egg	$3.00	1
AA	milk	$5.00	3
AA	egg	$3.00	2
NYU	egg	$4.00	4
NYU	milk	$6.00	3
NYU	candy	$2.00	1
SF	milk	$6.00	5
SF	egg	$4.00	4
SF	egg	$4.00	5

n tuples

`randint(1,b)`

We call this augmented table, a variational table

```
Algorithm:
for i = 1, ..., n
  sum[sid] += price[i]
```

Time Complexity: O(n)

Variational subsampling in SQL is fast

CITY	PRODUCT	PRICE	subsample ID
AA	egg	$3.00	1
AA	milk	$5.00	3
AA	egg	$3.00	2
NYU	egg	$4.00	4
NYU	milk	$6.00	3
NYU	candy	$2.00	1
SF	milk	$6.00	5
SF	egg	$4.00	4
SF	egg	$4.00	5

randint(1,b)

n tuples

We call this augmented table, a variational table

```
Algorithm:
for i = 1, ..., n
  sum[sid] += price[i]
```

Time Complexity: O(n)

No error est: 0.35 sec

Trad. subsampling: 118 sec

Var. subsampling: 0.73 sec

162× faster than traditional

(based on 1G sample, Impala)

Main results

<u>Theorem 1 (Consistency)</u> ***The distribution*** *of the aggregates of* ***variational subsamples****, after appropriate scaling,* ***converges to the true distribution*** *of the aggregate of a sample as* $n \to \infty$.

Main results

Theorem 1 (Consistency) ***The distribution*** *of the aggregates of* ***variational subsamples****, after appropriate scaling,* ***converges to the true distribution*** *of the aggregate of a sample as* $n \to \infty$.

Theorem 2 (Convergence Rate) *The convergence rate of variational subsampling is equal to that of traditional subsampling* ***when b is finite***.

$$O\left(n_s^{-1/2} + \frac{n_s}{n} + \underline{b^{-1/2}}\right)$$

The error term from the finite b
(The Dvoretzky–Kiefer–Wolfowitz inequality)

Experiments

Experiments

1. Does VerdictDB provide enough speedup?

Experiments

1. Does VerdictDB provide enough speedup?
2. Is VerdictDB (UAQP)'s performance comparable to a tightly-integrated AQP engine?

Experiments

1. Does VerdictDB provide enough speedup?
2. Is VerdictDB (UAQP)'s performance comparable to a tightly-integrated AQP engine?
3. Is variational subsampling statistically correct?

Experiments

1. Does VerdictDB provide enough speedup?
2. Is VerdictDB (UAQP)'s performance comparable to a tightly-integrated AQP engine?
3. Is variational subsampling statistically correct?

Yes

Experiments

1. Does VerdictDB provide enough speedup?
2. Is VerdictDB (UAQP)'s performance comparable to a tightly-integrated AQP engine?
3. Is variational subsampling statistically correct?

Yes

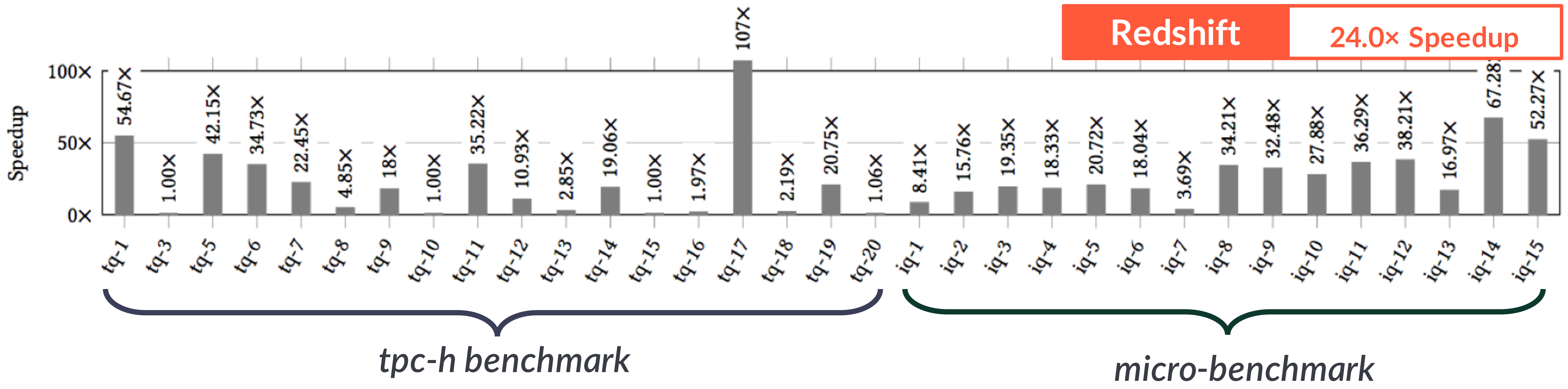
Datasets:

- 500GB TPC-H benchmark / 200GB Instacart dataset / synthetic datasets

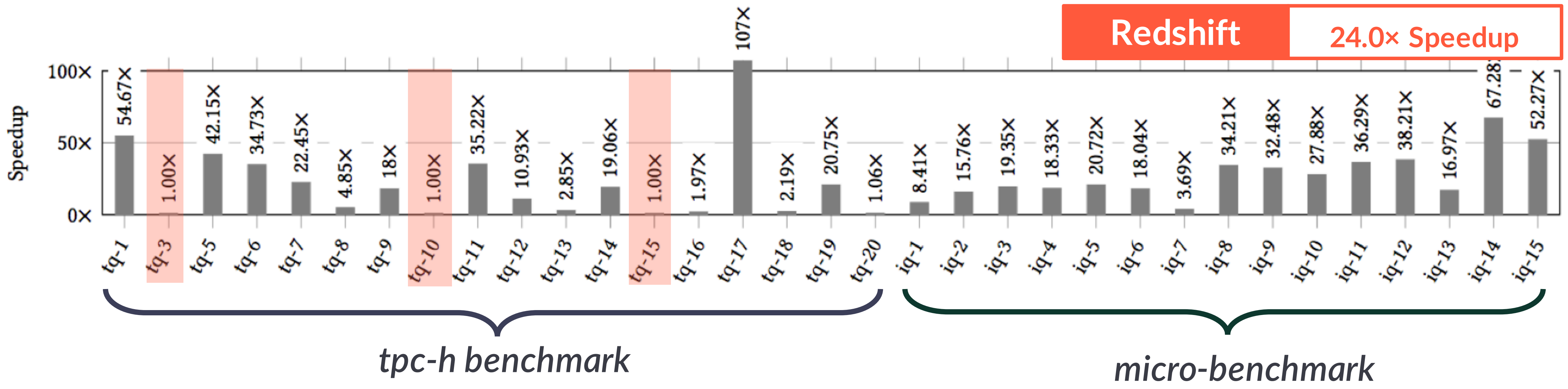
Underlying databases

- Amazon Redshift, Apache Spark SQL, Apache Impala on 10+1 r4.xlarge cluster

Speedup for Redshift

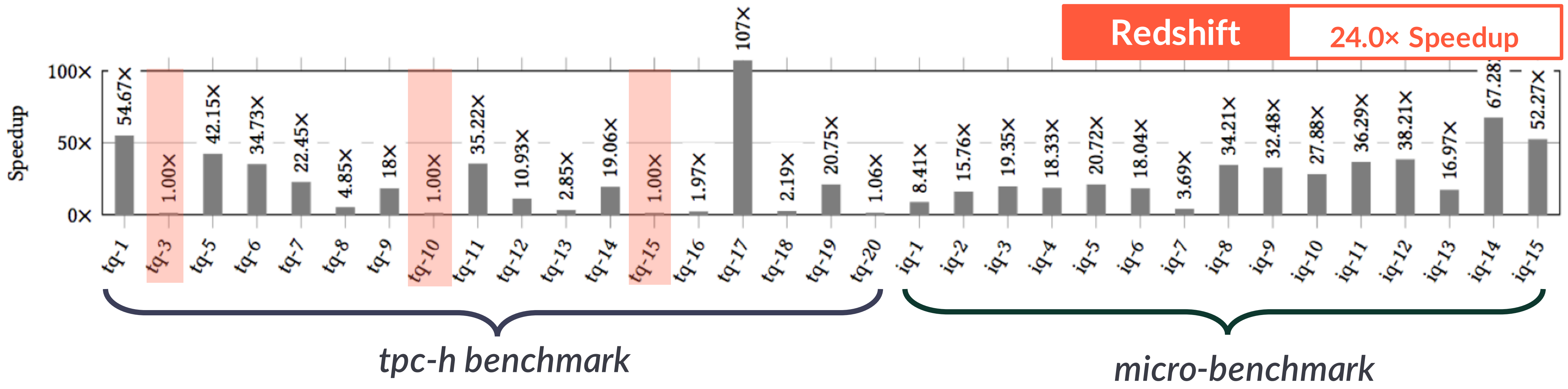


Speedup for Redshift



t3, t10, t15: no speedup (i.e., 1×) due to high-cardinality grouping attributes

Speedup for Redshift

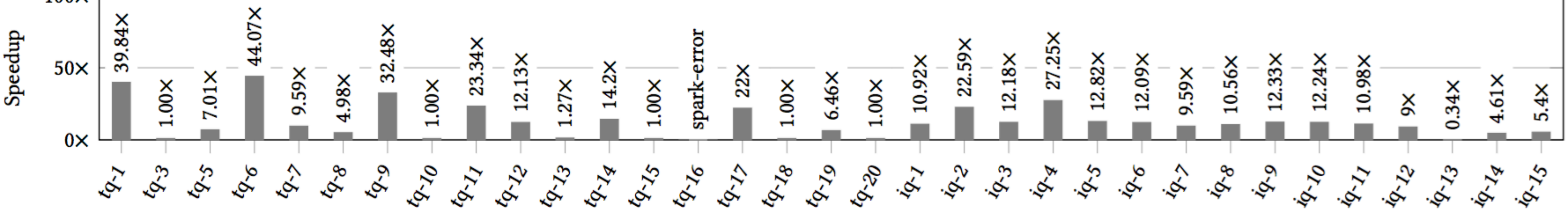


t3, t10, t15: no speedup (i.e., 1×) due to high-cardinality grouping attributes

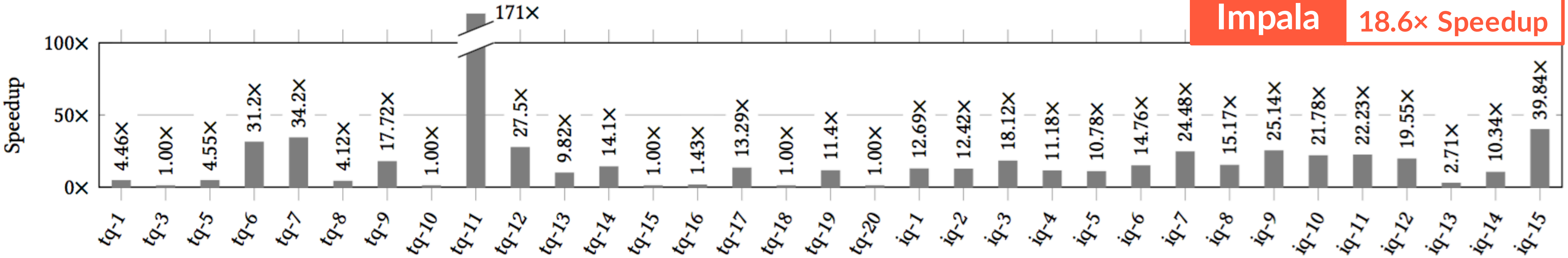
Other queries: 26.3× speedups (relative errors were 2%)

Speedup for Apache Spark & Impala

Spark SQL 12.0× Speedup

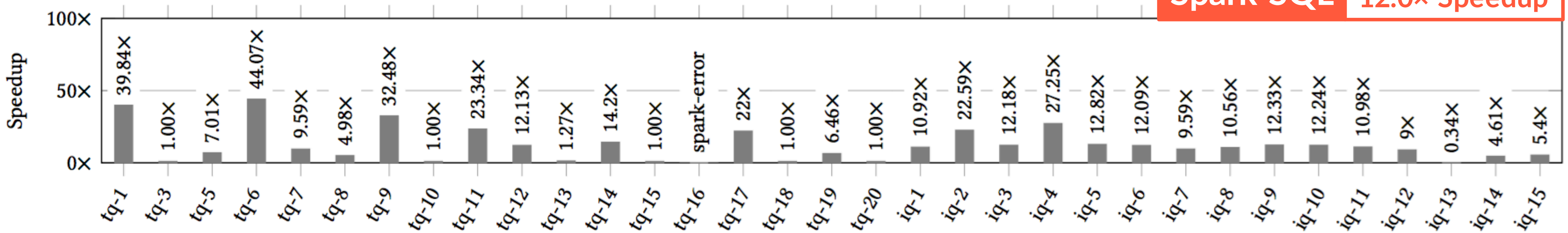


Impala 18.6× Speedup

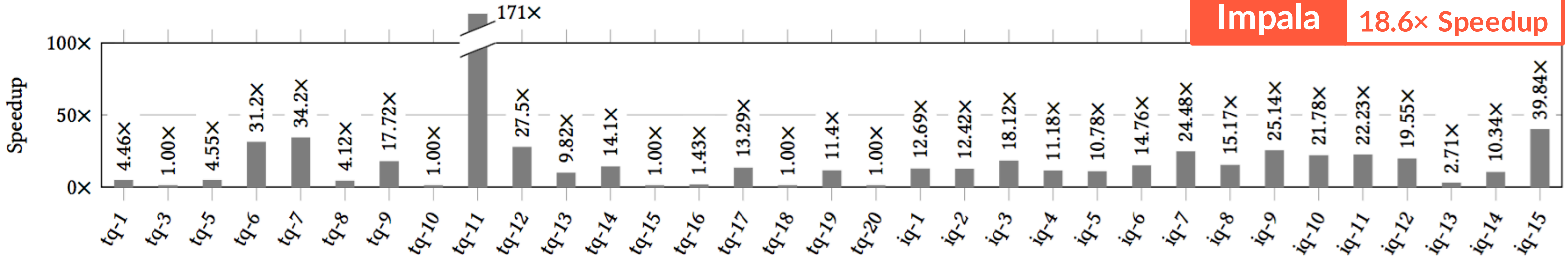


Speedup for Apache Spark & Impala

Spark SQL 12.0× Speedup



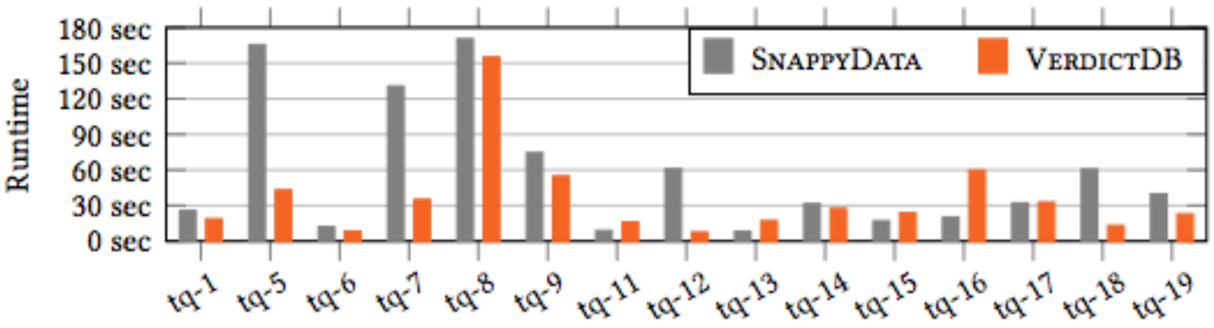
Impala 18.6× Speedup

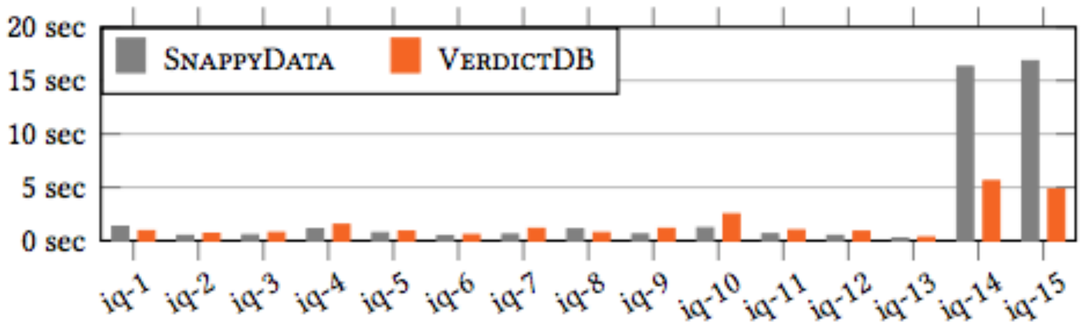


$$speedup = \frac{overhead + processing}{overhead + (sample\ processing)}$$

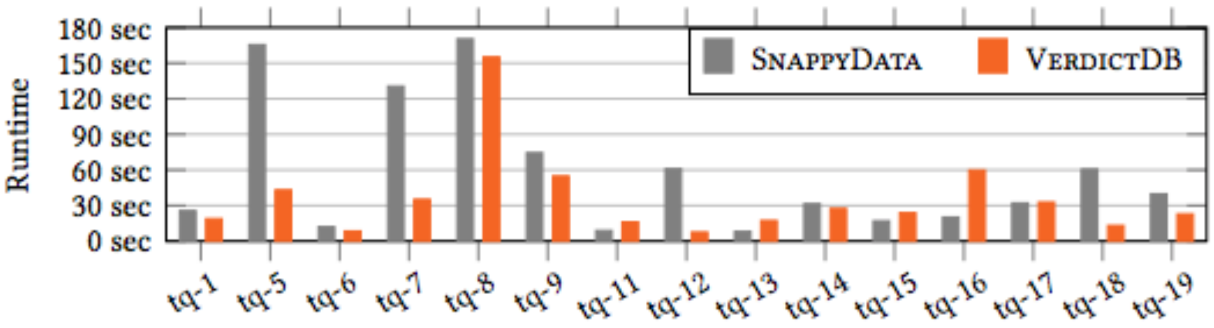
Lower overhead →
Larger speedup

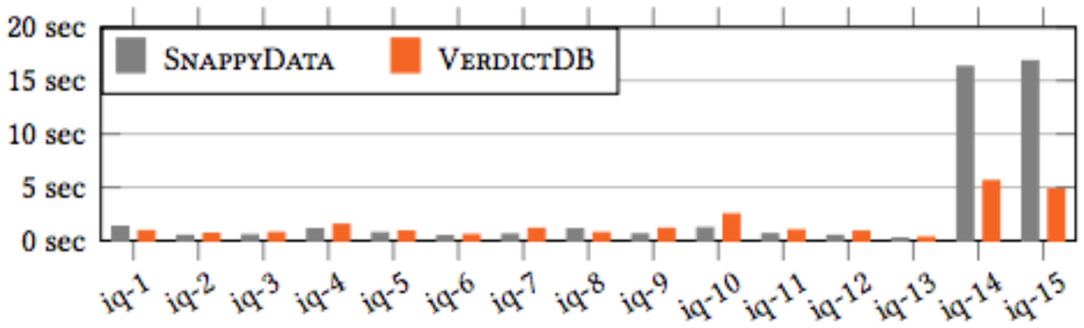
UAQP vs. Tightly-integrated AQP





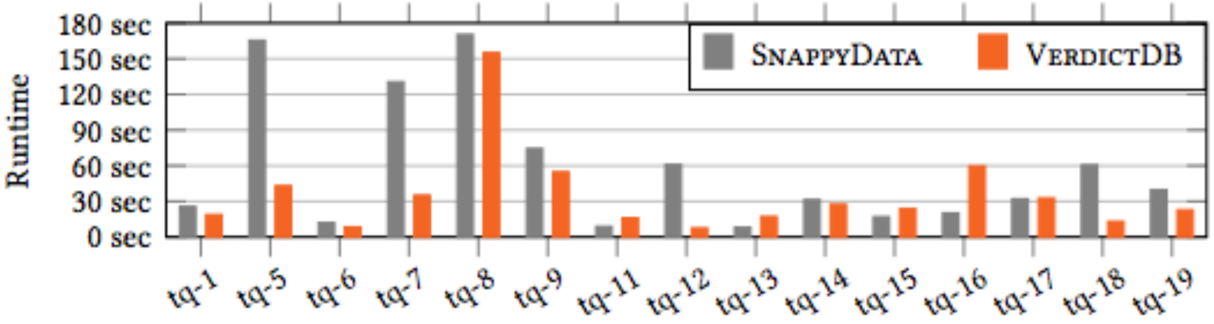
UAQP vs. Tightly-integrated AQP

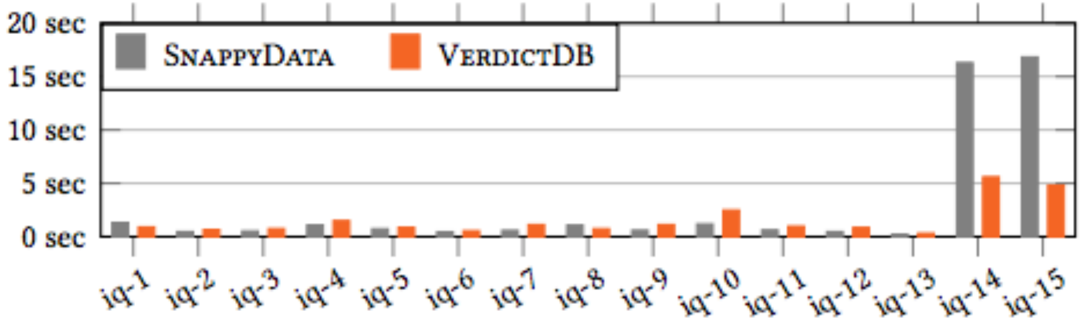




VerdictDB was comparable to SnappyData.

UAQP vs. Tightly-integrated AQP



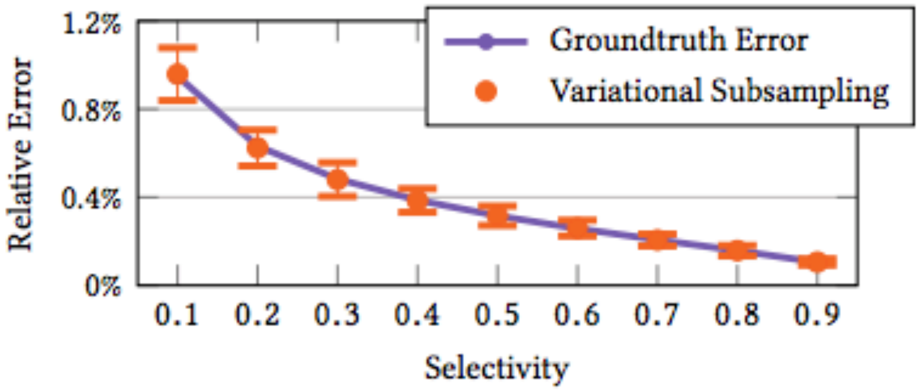


VerdictDB was comparable to SnappyData.

SnappyData ver 0.8 didn't support the join of two sample tables.

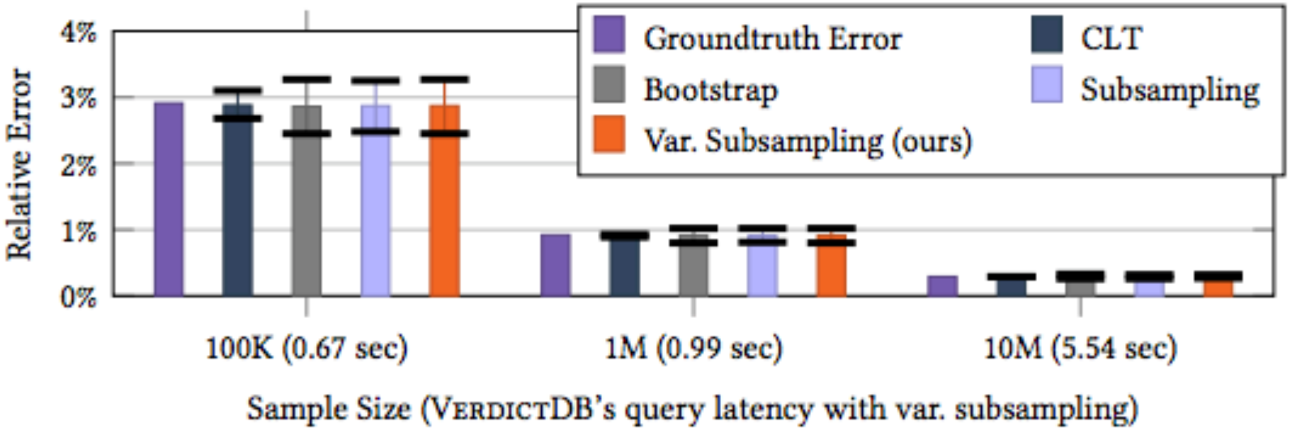
Variational subsampling: correctness

Rel. err. naturally become smaller for higher selectivity.



(a) Estimated error for different selectivity

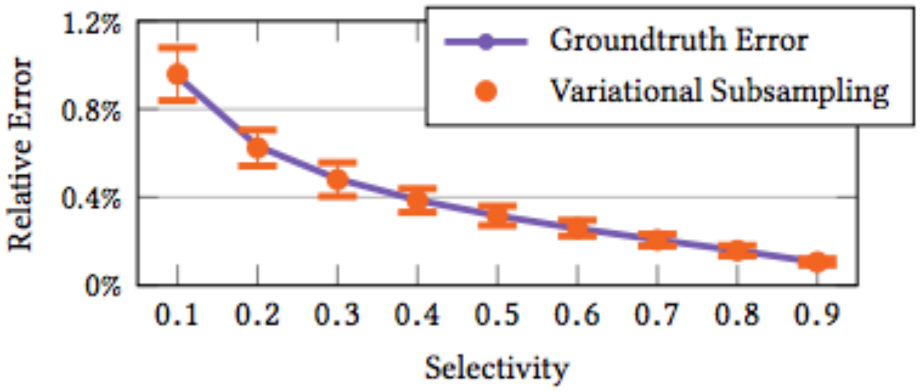
The bars are 5th and 95th percentiles.



(b) Estimated error for different sample sizes

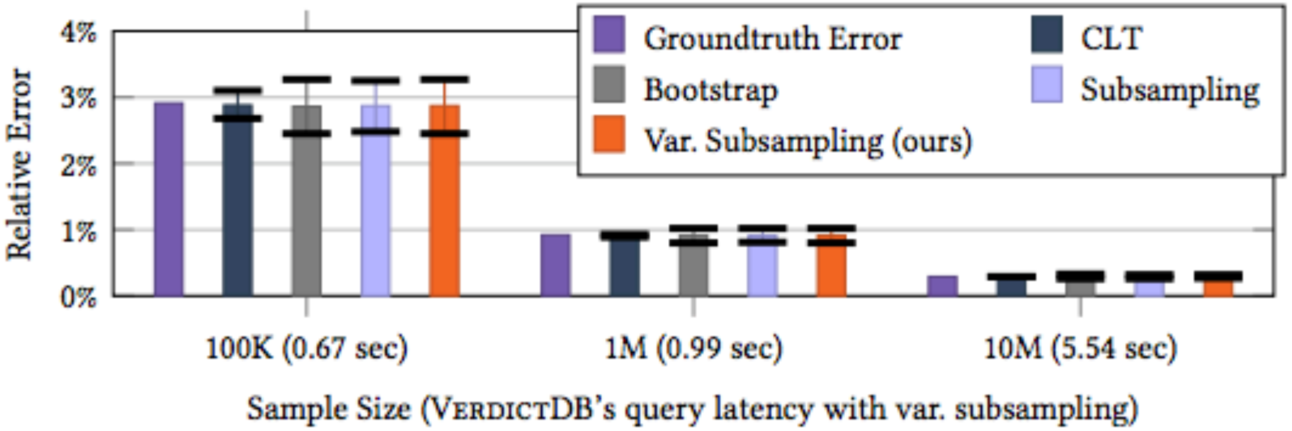
Variational subsampling: correctness

Rel. err. naturally become smaller for higher selectivity.



(a) Estimated error for different selectivity

The bars are 5th and 95th percentiles.

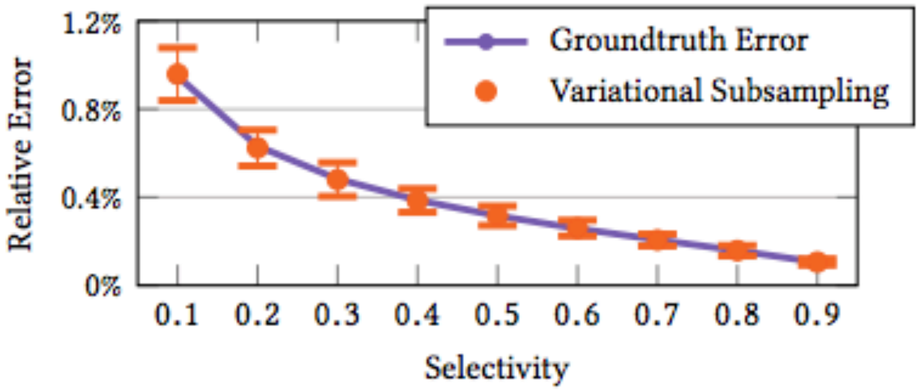


(b) Estimated error for different sample sizes

The estimated errors close to true errors.

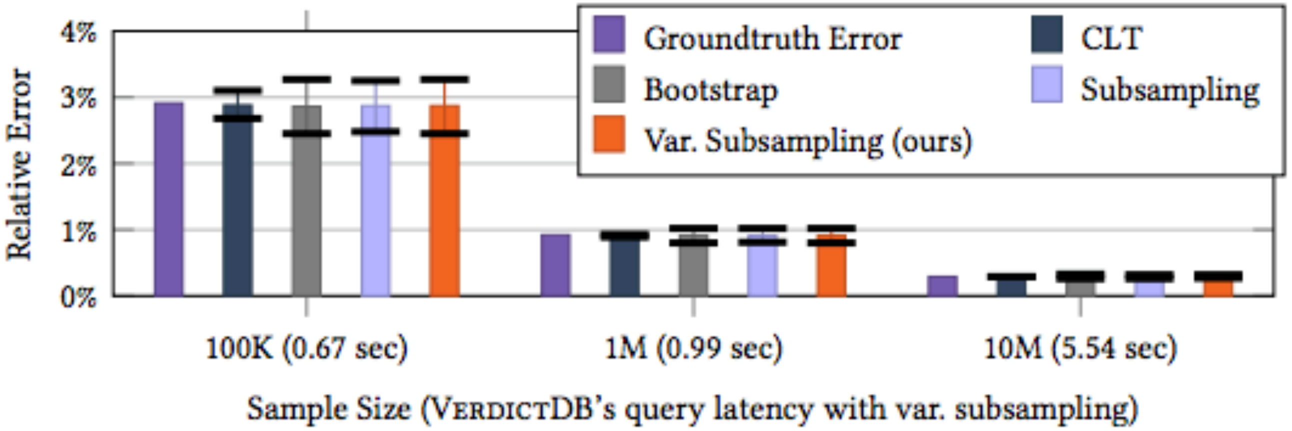
Variational subsampling: correctness

Rel. err. naturally become smaller for higher selectivity.



(a) Estimated error for different selectivity

The bars are 5th and 95th percentiles.

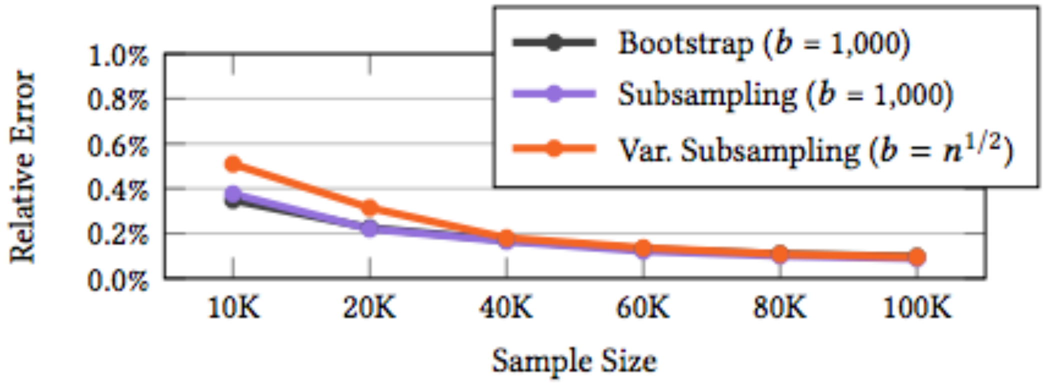


(b) Estimated error for different sample sizes

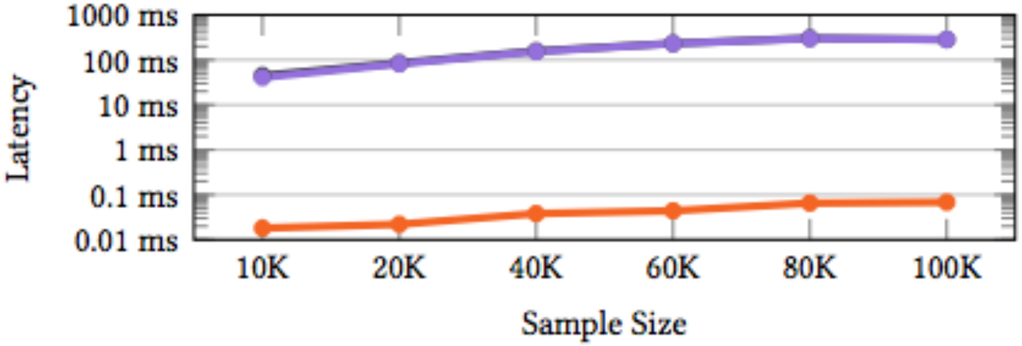
The estimated errors close to true errors.

The accuracy of var. subsampling ≈ (a) bootstrap and (b) trad. subsampling

Variational subsampling: convergence rate

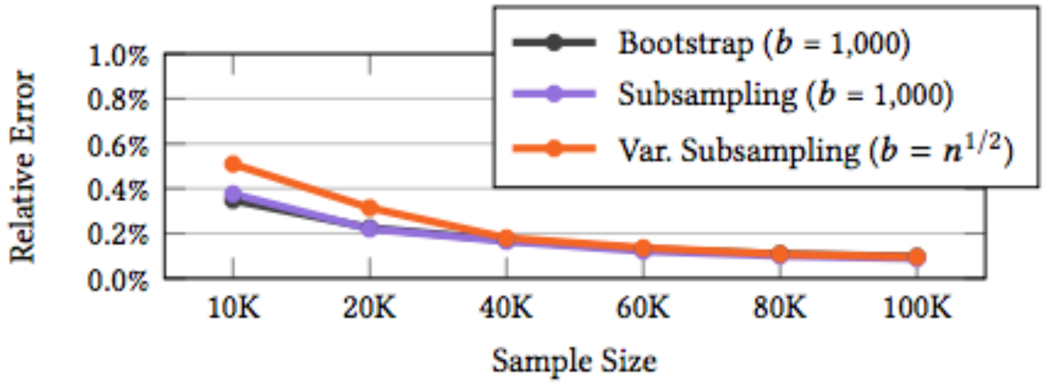


(a) Accuracy of error bound estimation

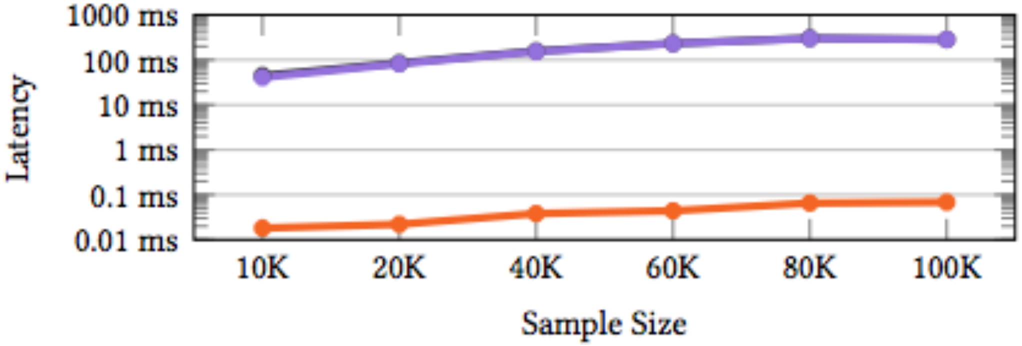


(b) Latency of error bound estimation

Variational subsampling: convergence rate



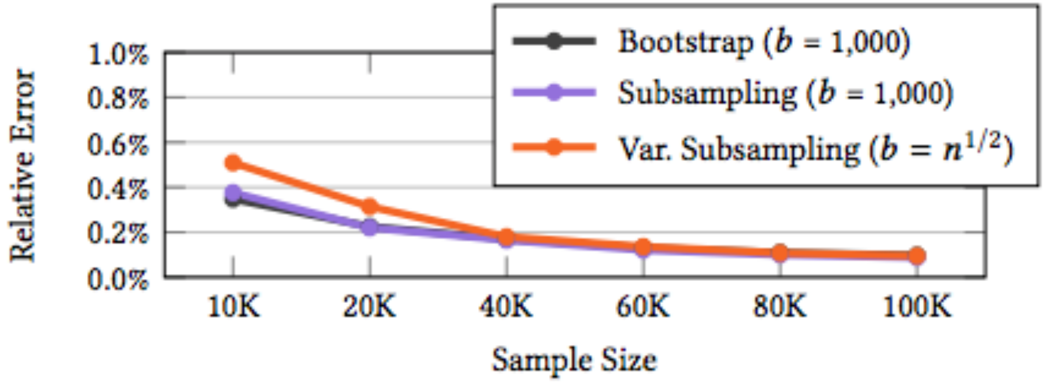
(a) Accuracy of error bound estimation



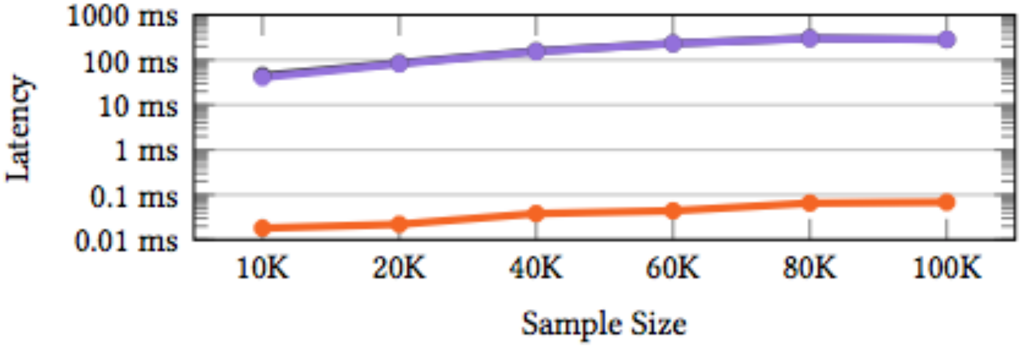
(b) Latency of error bound estimation

The accuracy was almost the same for relatively large samples.

Variational subsampling: convergence rate



(a) Accuracy of error bound estimation



(b) Latency of error bound estimation

The accuracy was almost the same for relatively large samples.

Variational subsampling was significantly faster.

Conclusion: Universal AQP is viable

Conclusion: Universal AQP is viable

1. Comparable performance to a fully-integrated solution

Conclusion: Universal AQP is viable

1. Comparable performance to a fully-integrated solution

2. New error estimation technique: *variational subsampling*

Conclusion: Universal AQP is viable

1. Comparable performance to a fully-integrated solution
2. New error estimation technique: *variational subsampling*
    1. Generality and computational efficiency

Conclusion: Universal AQP is viable

1. Comparable performance to a fully-integrated solution
2. New error estimation technique: *variational subsampling*
    1. Generality and computational efficiency
    2. The first subsampling-based error estimation technique for AQP

Conclusion: Universal AQP is viable

1. Comparable performance to a fully-integrated solution
2. New error estimation technique: *variational subsampling*
    1. Generality and computational efficiency
    2. The first subsampling-based error estimation technique for AQP
3. Offers considerable speedup
   (18.45× on average, up to 171×, less than 2-3% errors)

Conclusion: Universal AQP is viable

1. Comparable performance to a fully-integrated solution
2. New error estimation technique: *variational subsampling*
    1. Generality and computational efficiency
    2. The first subsampling-based error estimation technique for AQP
3. Offers considerable speedup
   (18.45× on average, up to 171×, less than 2-3% errors)

Open-sourced (Apache v2.0): http://verdictdb.org

Future Work

Development

Research

Future Work

Development

- Adding more drivers (Presto, Teradata, Oracle, SQL Server, …)

Research

Future Work

Development

- Adding more drivers (Presto, Teradata, Oracle, SQL Server, ...)

Research

- Support for online sampling

Future Work

Development

- Adding more drivers (Presto, Teradata, Oracle, SQL Server, ...)

Research

- Support for online sampling
- Robust physical designer (see CliffGuard @ SIGMOD 15)

Future Work

Development

- Adding more drivers (Presto, Teradata, Oracle, SQL Server, ...)

Research

- Support for online sampling
- Robust physical designer (see CliffGuard @ SIGMOD 15)
- Integration with ML libraries (sampling-based model tuning)

VerdictDB

http://verdictdb.org

Thank You

VerdictDB: current status

- We support
  - aggregates: sum, count, avg, count-distinct, quantiles, UDAs
  - sources: base table, derived table, equi-join
  - filters: comparison, some subquery
  - others: group-by, having, etc.

VerdictDB: current status

- We support
  - aggregates: sum, count, avg, count-distinct, quantiles, UDAs
  - sources: base table, derived table, equi-join
  - filters: comparison, some subquery
  - others: group-by, having, etc.
- Open-sourced under Apache License version 2.0
  - http://verdictdb.org for code and documentation

VerdictDB: current status

- We support
  - aggregates: sum, count, avg, count-distinct, quantiles, UDAs
  - sources: base table, derived table, equi-join
  - filters: comparison, some subquery
  - others: group-by, having, etc.
- Open-sourced under Apache License version 2.0
  - http://verdictdb.org for code and documentation
- Upcoming features
  - Online sampling, automated physical designer

Example of query rewriting

original

```
select l_returnflag , count (*) as cc
from lineitem
group by l_returnflag ;
```

rewritten

```
select vt1 .` l_returnflag ` AS `l_returnflag `,
          round ( sum (( vt1 .`cc ` * vt1 .` sub_size `)) / sum ( vt1 .` sub_size `)) AS `cc `,
          (stddev ( vt1 .` count_order `) * sqrt ( avg ( vt1 .` sub_size `)))
          / sqrt ( sum ( vt1 .` sub_size `)) AS `cc_err `
from (select vt0 .` l_returnflag ` AS `l_returnflag `,
           (( sum ((1.0 / vt0 .` sampling_prob `)) / count (*))
           * sum ( count (*)) OVER ( partition BY vt0 .` l_returnflag `)) AS `cc `,
           vt0 .`sid ` AS `sid `, count (*) AS `sub_size `
        from lineitem_sample vt0
        GROUP BY vt0 .` l_returnflag `, vt0 .`sid `) AS vt1
GROUP BY vt1 .` l_returnflag `;
```

Bibliography

[Pol and Jermaine ‘05] Pol, Abhijit, and Christopher Jermaine. "Relational confidence bounds are easy with the bootstrap." In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pp. 587-598. ACM, 2005.

[BlinkDB ‘13] Agarwal, Sameer, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. EuroSys, 2013.

[BlinkDB ‘14] Agarwal, Sameer, Henry Milner, Ariel Kleiner, Ameet Talwalkar, Michael Jordan, Samuel Madden, Barzan Mozafari, and Ion Stoica. "Knowing when you're wrong: building fast and reliable approximate query processing systems." SIGMOD, 2014.

[Quickr ‘16] Kandula, Srikanth, Anil Shanbhag, Aleksandar Vitorovic, Matthaios Olma, Robert Grandl, Surajit Chaudhuri, and Bolin Ding. "Quickr: Lazily approximating complex adhoc queries in bigdata clusters." SIGMOD, 2016.

Bibliography

[G-OLA ’15] Zeng, Kai, Sameer Agarwal, Ankur Dave, Michael Armbrust, and Ion Stoica. "G-ola: Generalized on-line aggregation for interactive analysis on big data." SIGMOD, 2015.

[ABM ‘14] Zeng, Kai, Shi Gao, Barzan Mozafari, and Carlo Zaniolo. "The analytical bootstrap: a new method for fast error estimation in approximate query processing." SIGMOD, 2014.

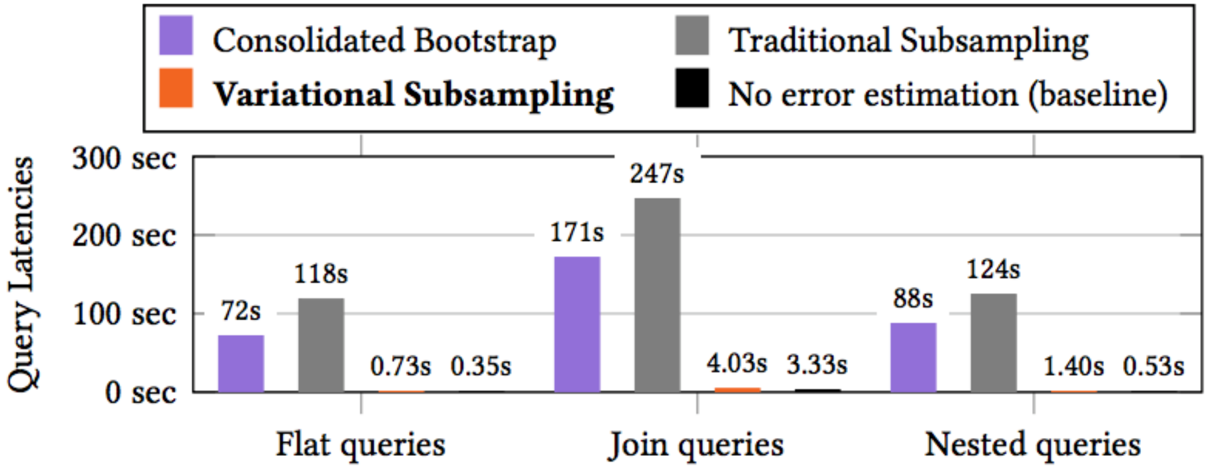
[Join Synopses ‘99] Acharya, Swarup, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. "Join synopses for approximate query answering." *SIGMOD Record*, 1999.

[Wander Join ‘16] Li, Feifei, Bin Wu, Ke Yi, and Zhuoyue Zhao. "Wander join: Online aggregation via random walks." SIGMOD, 2016.

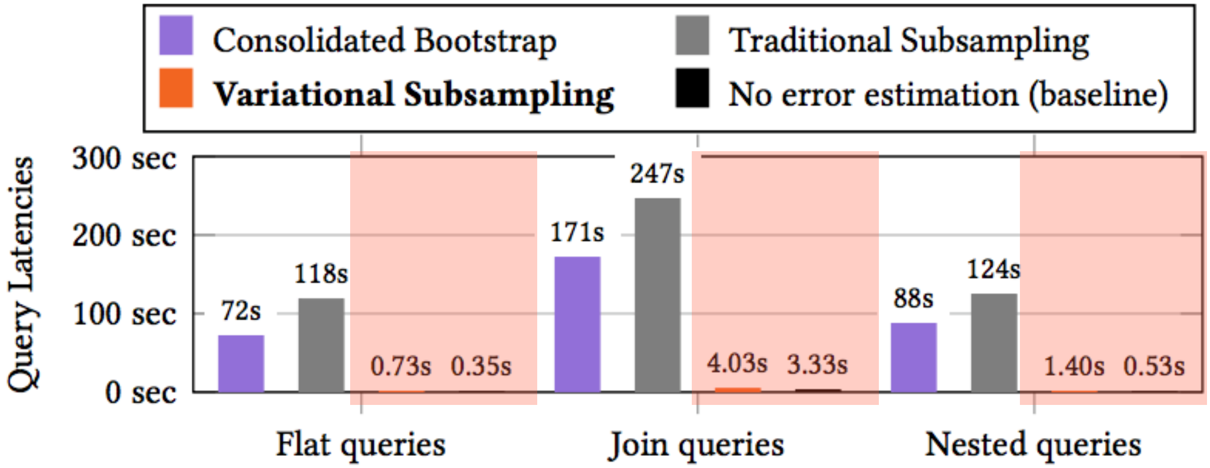
[Online ‘97] Hellerstein, Joseph M., Peter J. Haas, and Helen J. Wang. "Online aggregation." SIGMOD, 1997.

[Politis ‘94] Politis, Dimitris N., and Joseph P. Romano. "Large sample confidence regions based on subsamples under minimal assumptions." *The Annals of Statistics*, 1994

Variational subsampling: overhead

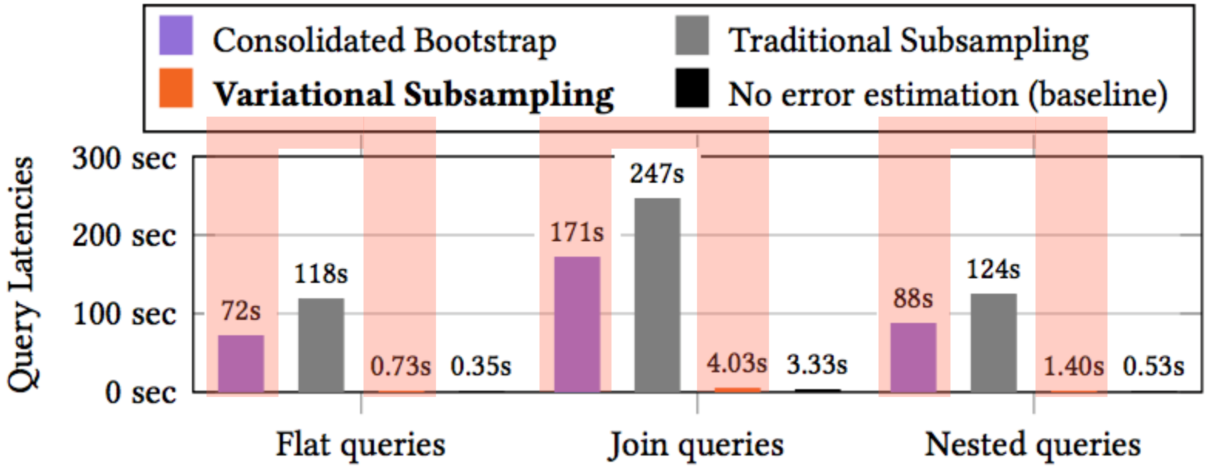


Variational subsampling: overhead



Overhead of *variational subsampling*: 0.38–0.87 *seconds*

Variational subsampling: overhead



Overhead of *variational subsampling*: 0.38–0.87 seconds

Variational subsampling was 100×–237× faster compared to Consolidated Bootstrap.