Conquering Big Data with BDAS (Berkeley Data Analytics)

Ion Stoica

UC Berkeley / Databricks / Conviva

Extracting Value from Big Data

Insights, diagnosis, e.g.,

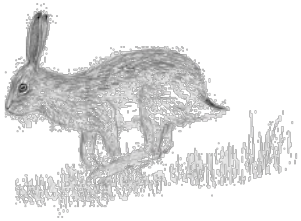
- » Why is user engagement dropping?
- » Why is the system slow?
- » Detect spam, DDoS attacks

Decisions, e.g.,

- » Decide what features to add to a product
- » Personalized medical treatment
- » Decide when to change an aircraft engine part
- » Decide what ads to show

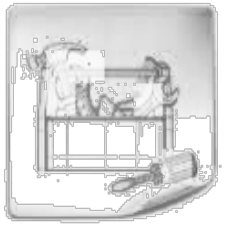
Data only as useful as the decisions it enables

What do We Need?



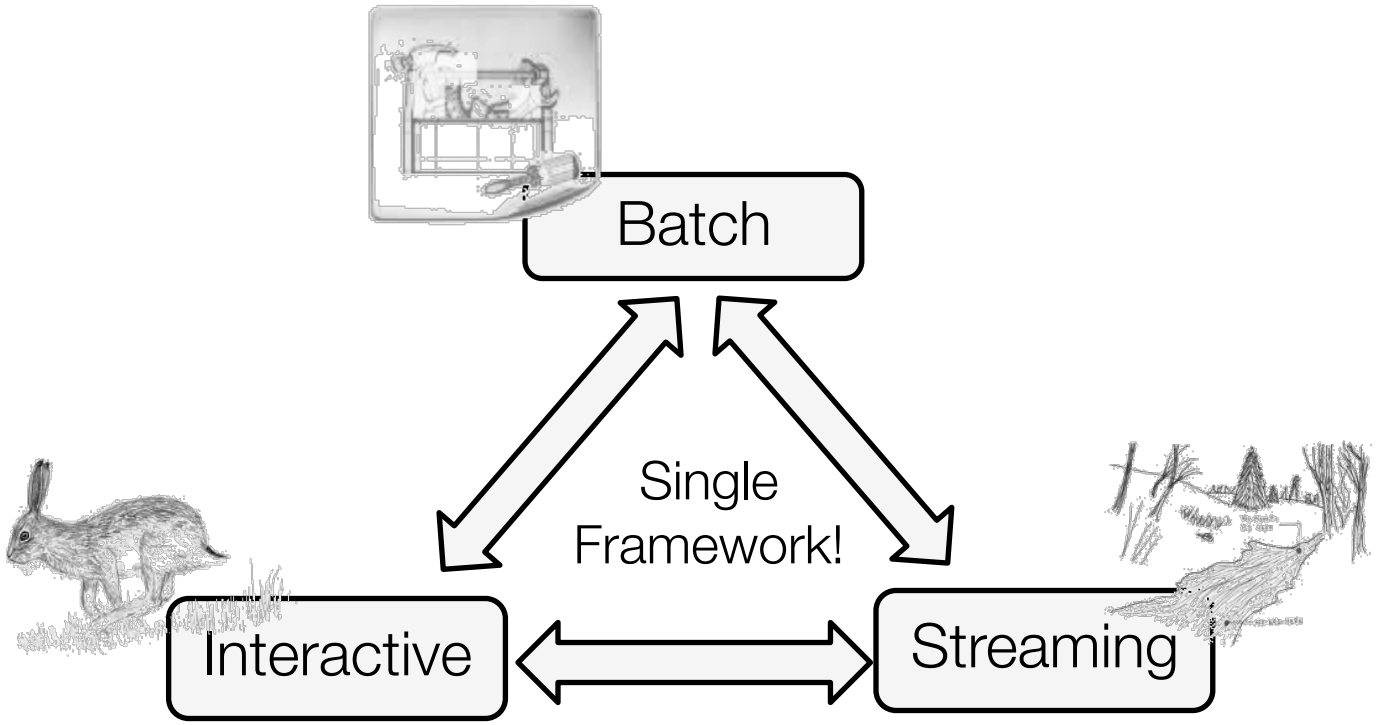
Interactive queries: enable faster decisions
- » E.g., identify why a site is slow and fix it

Queries on streaming data: enable decisions on real-time data
- » E.g., fraud detection, detect DDoS attacks



Sophisticated data processing: enable "better" decisions
- » E.g., anomaly detection, trend analysis

Our Goal



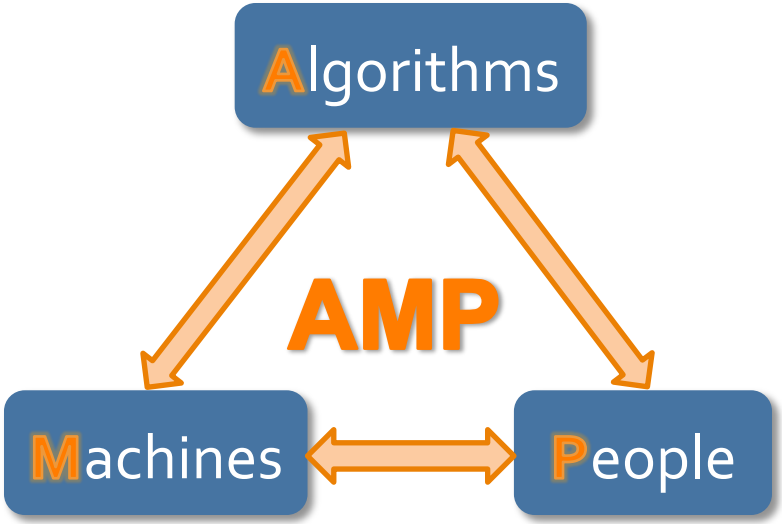
Support *batch*, *streaming*, and *interactive* computations…

… in a unified framework

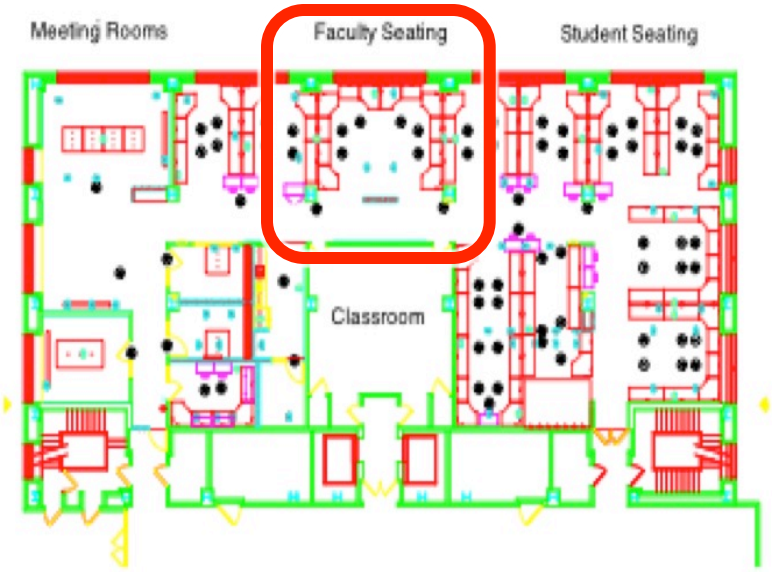
Easy to develop *sophisticated* algorithms (e.g., graph, ML algos)

The Berkeley AMPLab

January 2011 – 2017
- » 8 faculty
- » > 40 students
- » 3 software engineer team



Organized for collaboration



3 day retreats
(twice a year)

AMPCamp3
(August, 2013)

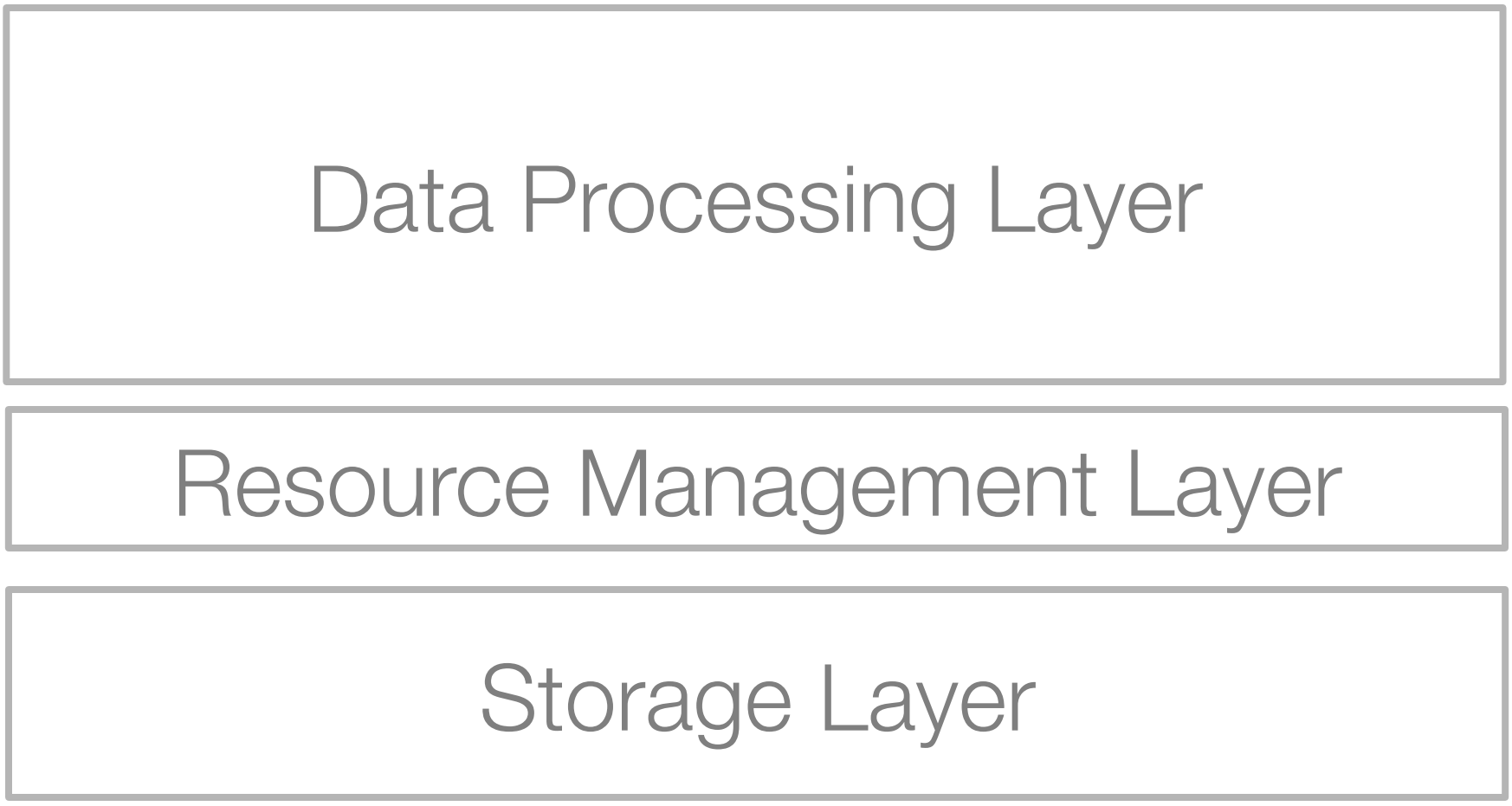
220 campers
(100+ companies)

The Berkeley AMPLab

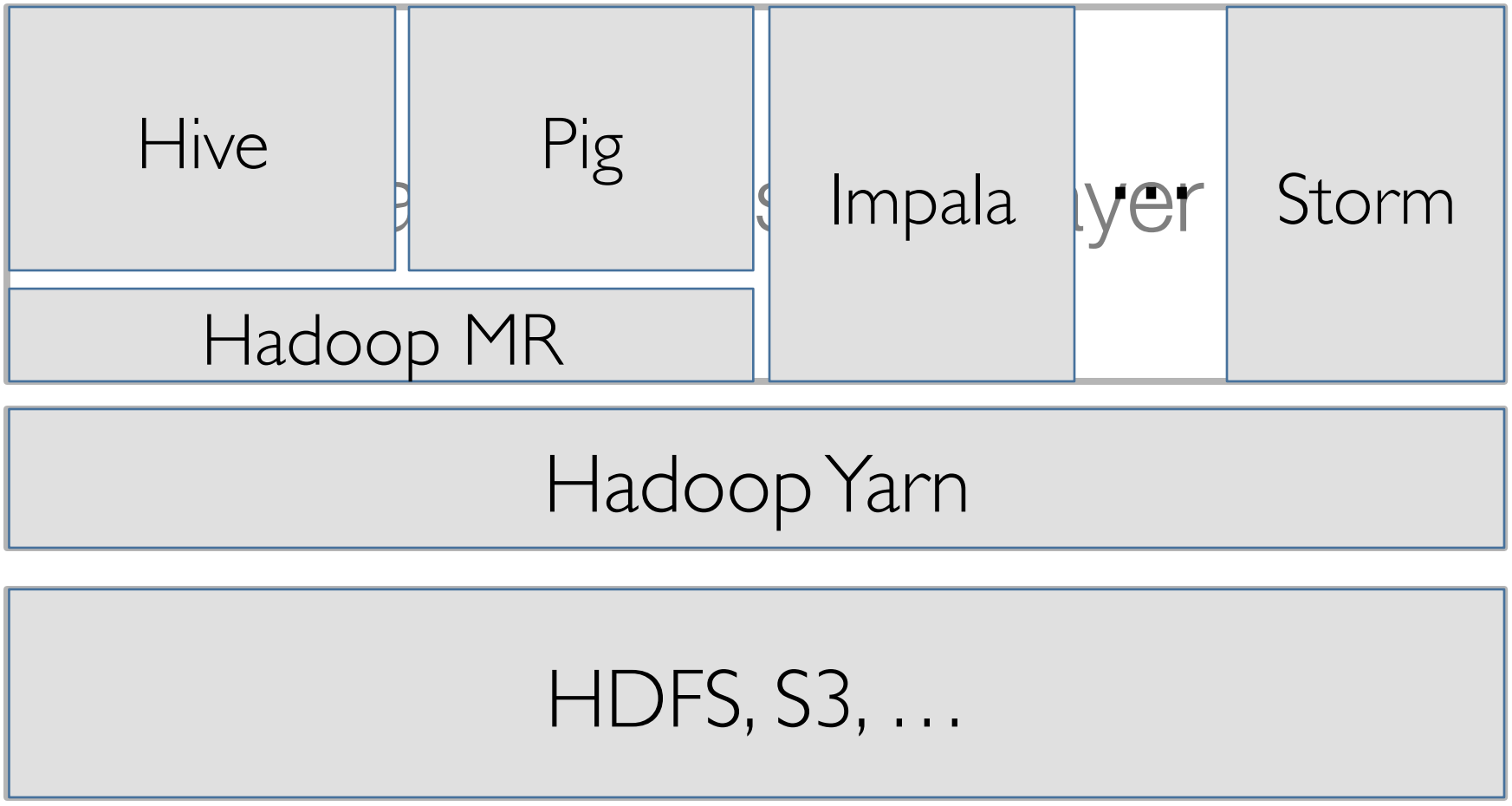
Governmental and industrial funding:

Goal: Next generation of open source data analytics stack for industry & academia:
Berkeley Data Analytics Stack (BDAS)

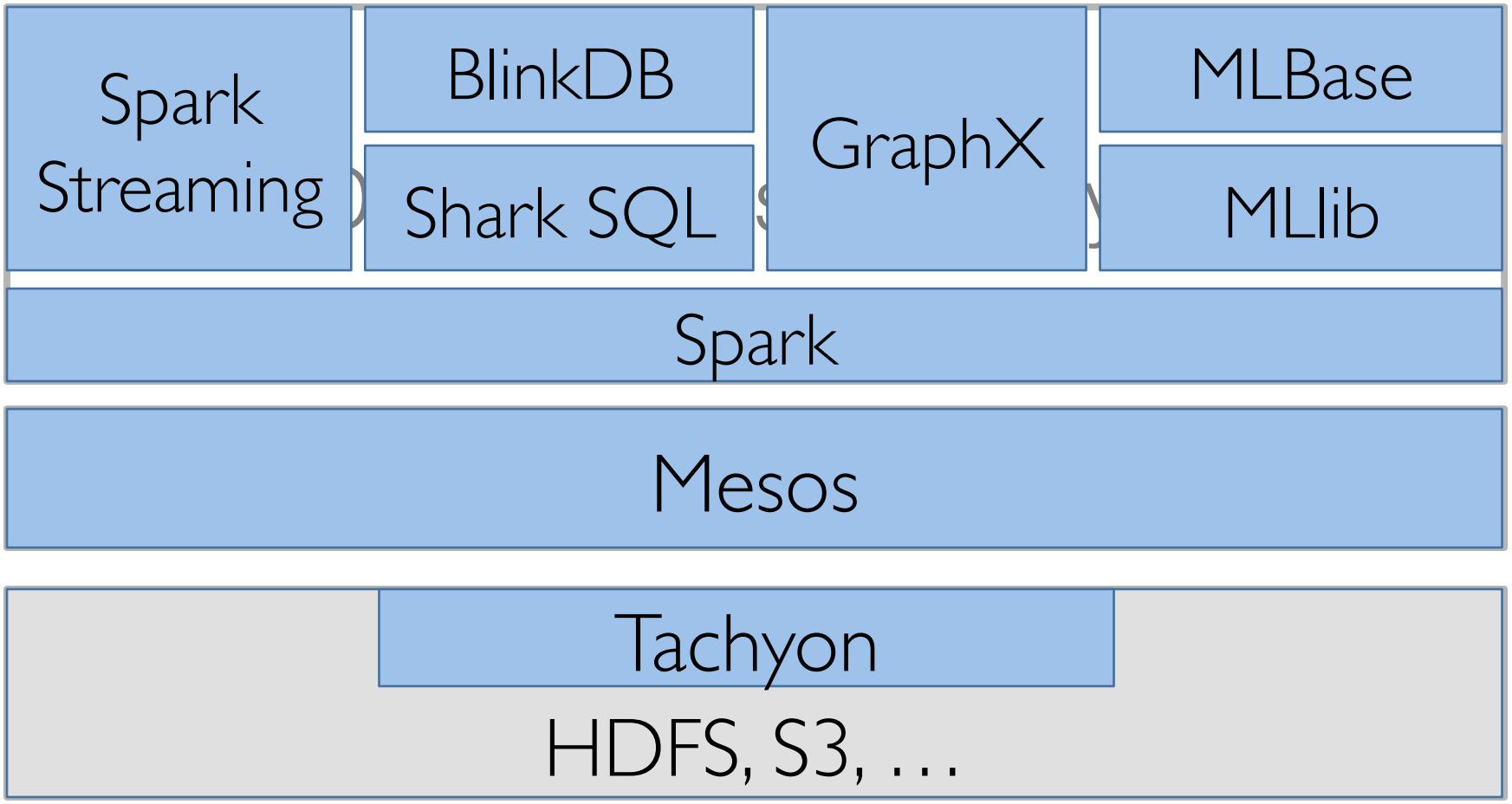
Data Processing Stack



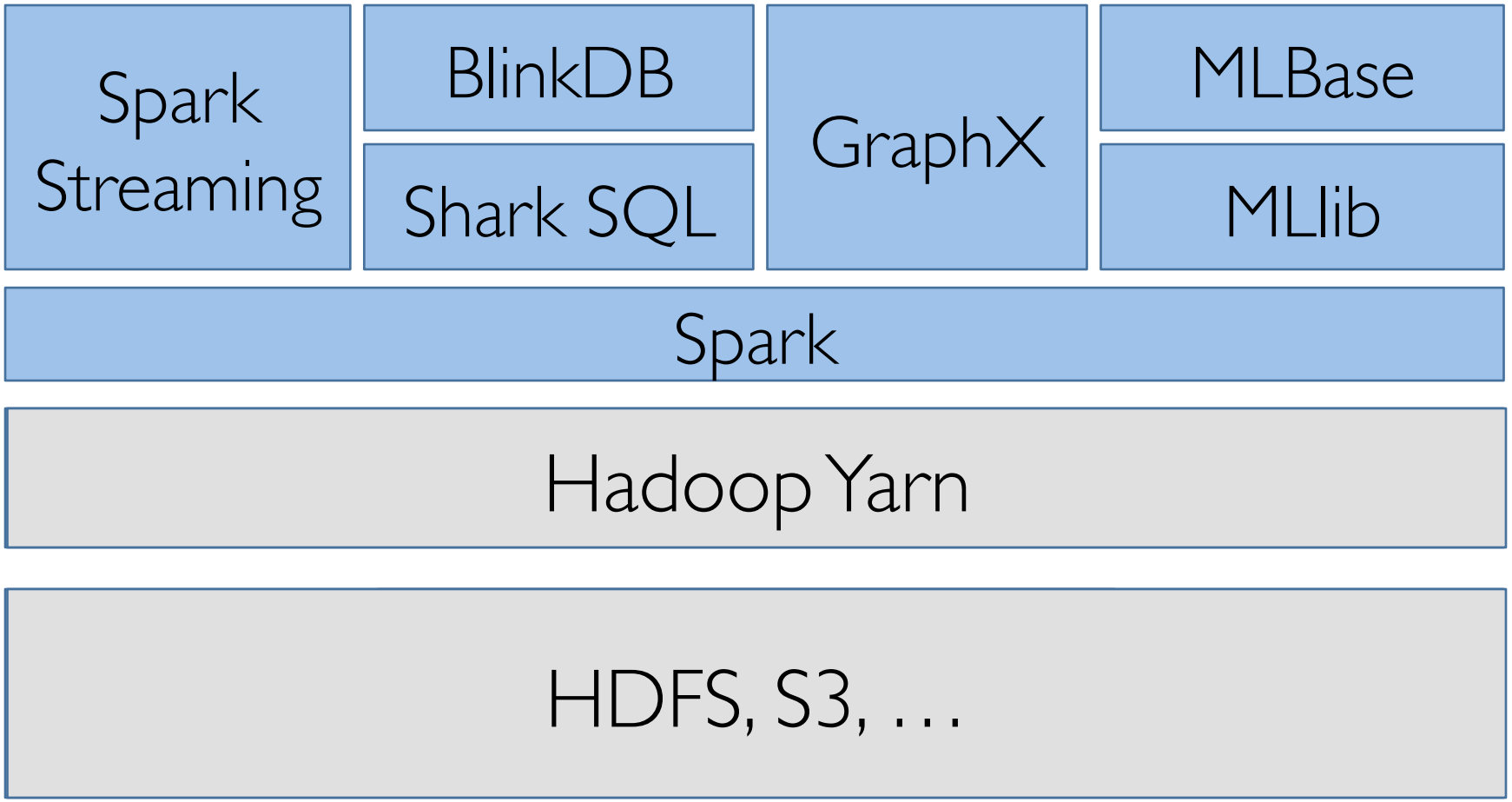
Hadoop Stack



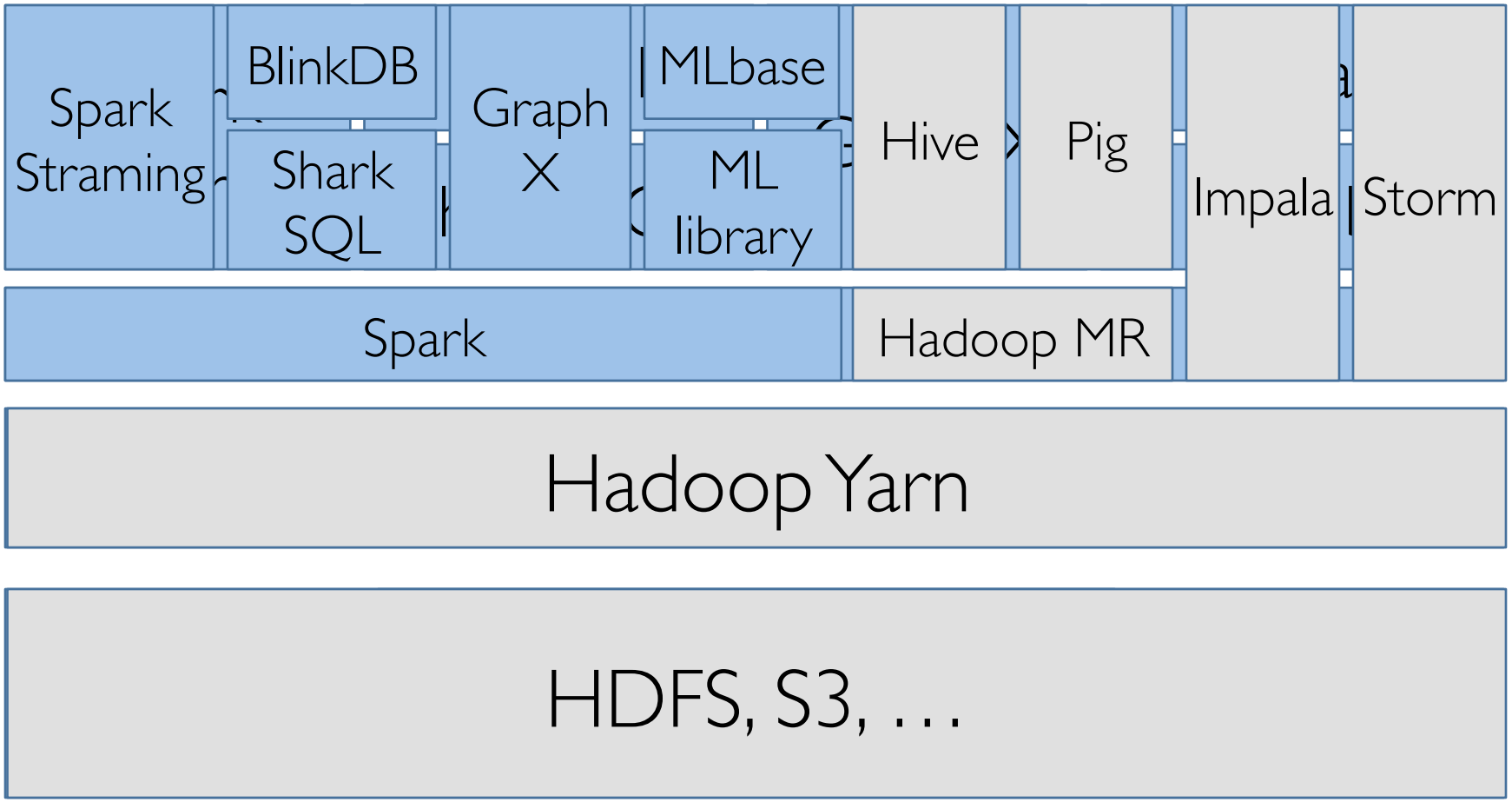
BDAS Stack


Spark Streaming
BlinkDB
Shark SQL
GraphX
MLBase
MLlib
Spark
Mesos
Tachyon
HDFS, S3, …

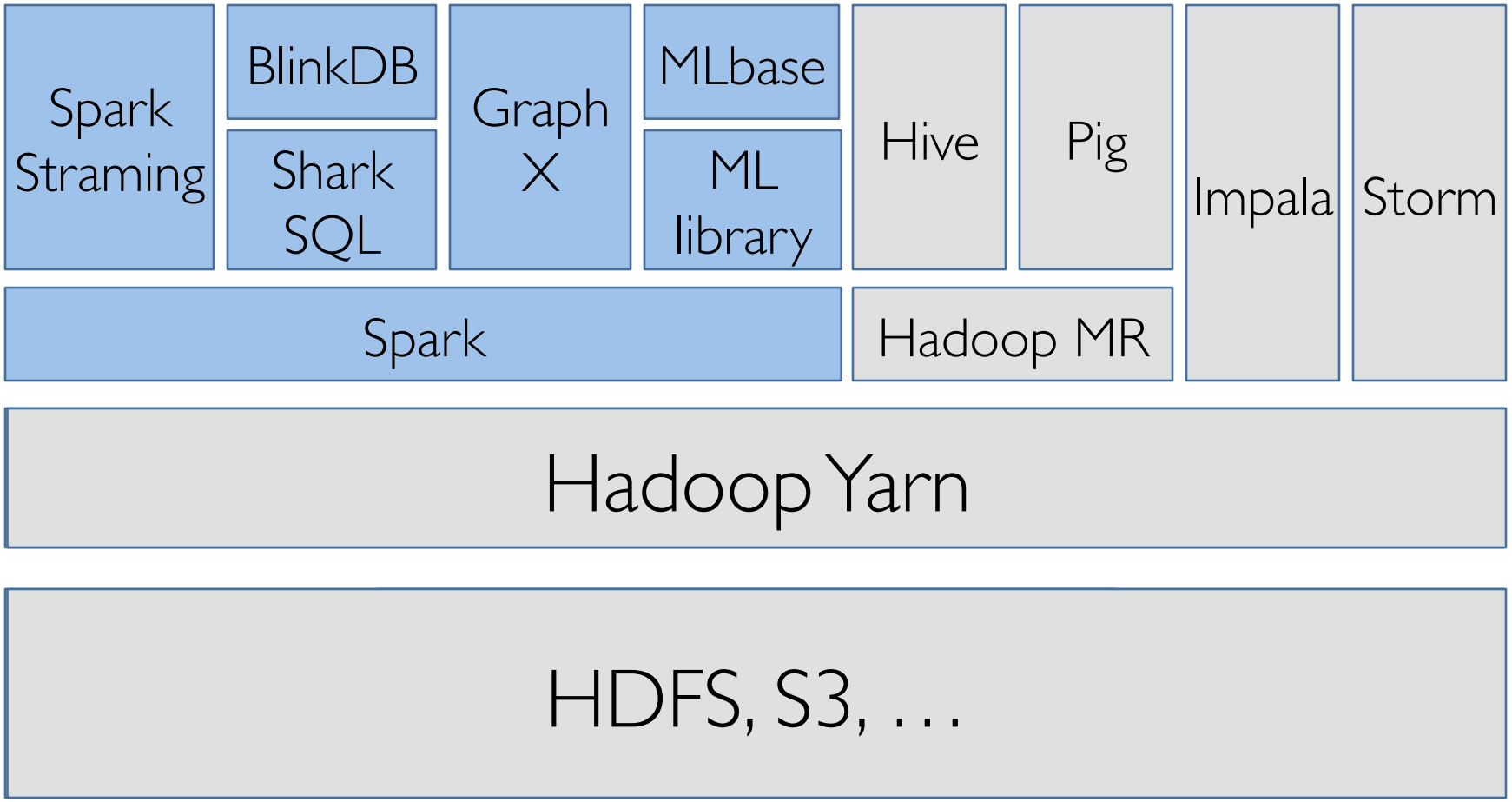
How do BDAS & Hadoop fit together?



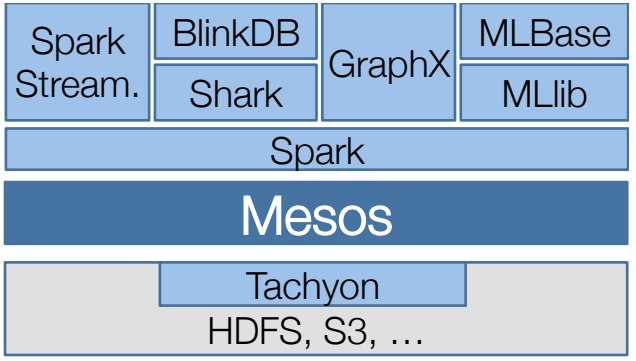
How do BDAS & Hadoop fit together?



How do BDAS & Hadoop fit together?



Apache Mesos



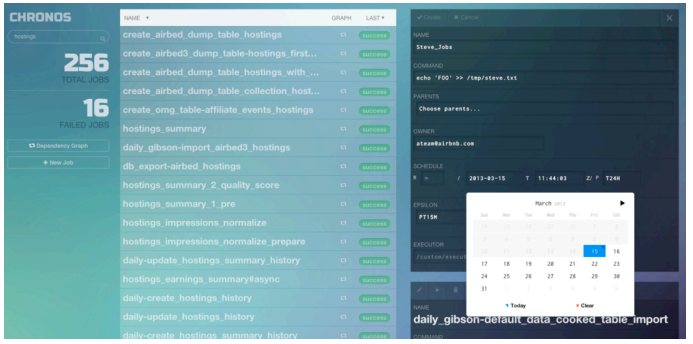
Enable multiple frameworks to share same cluster resources (e.g., Hadoop, Storm, Spark)

Twitter's large scale deployment

- » 10,000+ servers,
- » 500+ engineers running jobs on Mesos

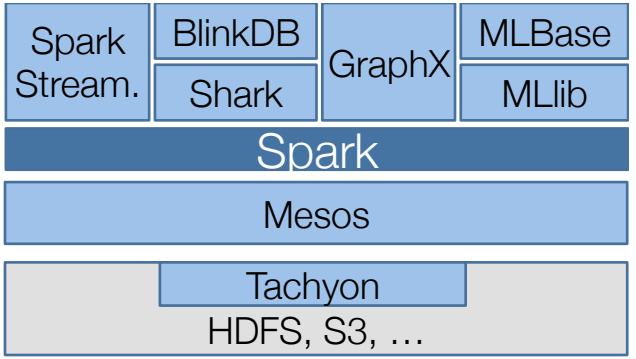
Third party Mesos schedulers

- » AirBnB's Chronos
- » Twitter's Aurora



Mesospehere: startup to commercialize Mesos

Apache Spark



Distributed Execution Engine

- » Fault-tolerant, efficient in-memory storage
- » Low-latency large-scale task scheduler
- » Powerful prog. model and APIs: Python, Java, Scala

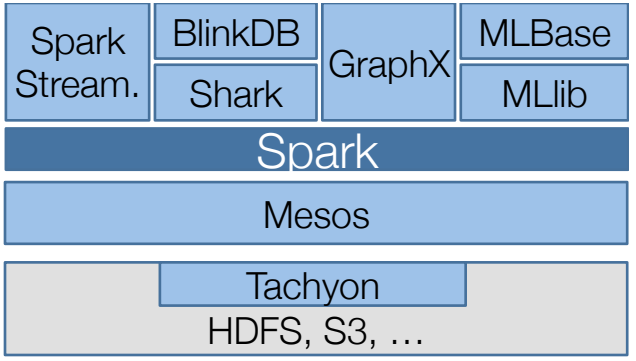
Fast: up to 100x faster than Hadoop MR

- » Can run sub-second jobs on hundreds of nodes

Easy to use: 2-5x less code than Hadoop MR

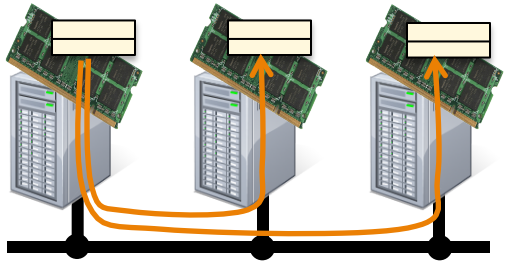
General: support interactive & iterative apps

Fault Tolerance



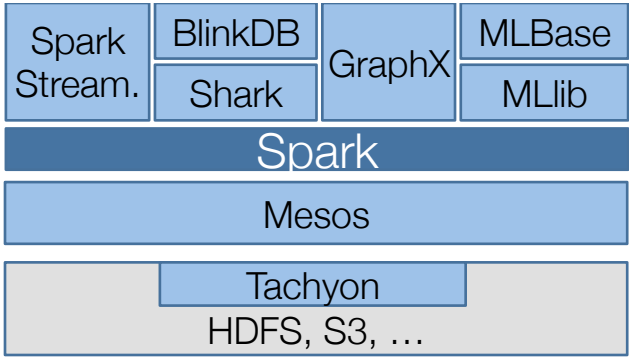
Need to achieve
- » High throughput reads and **writes**
- » Efficient memory usage

Replication
- » Writes bottlenecked by network
- » Inefficient: store multiple replicas



Persist update logs
- » Big data processing can generate massive logs

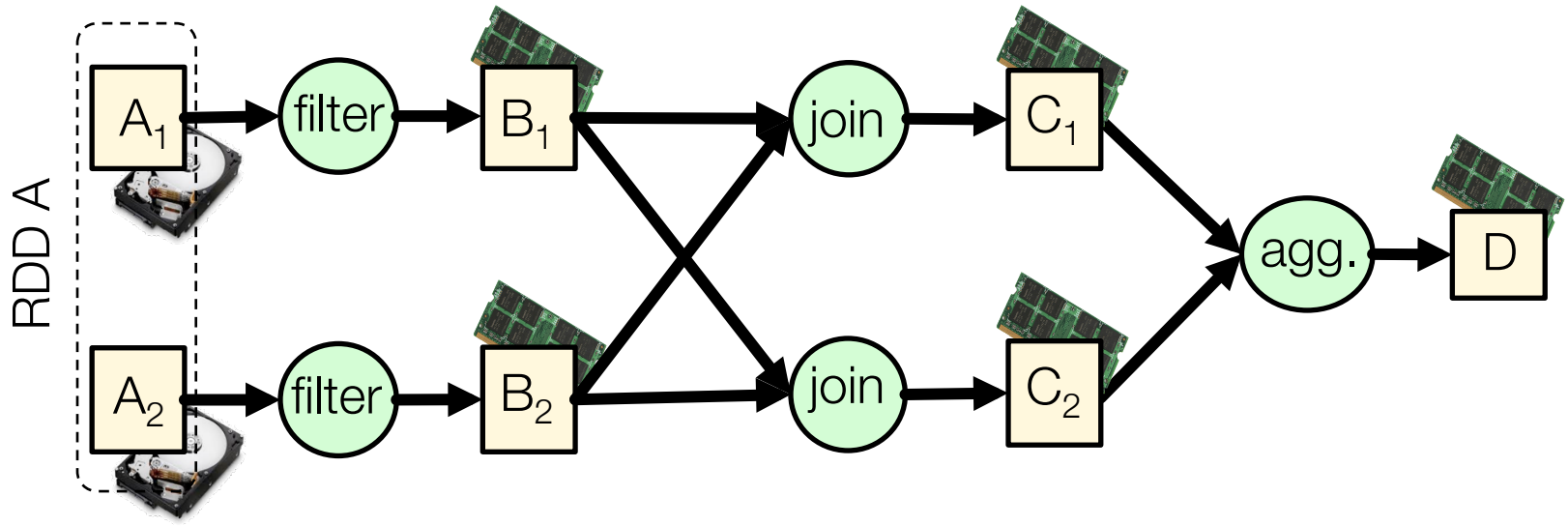
Our solution: Resilient Distributed Datasets (RDDs)
- » Partitioned collection of immutable records
- » Use lineage to reconstruct lost data



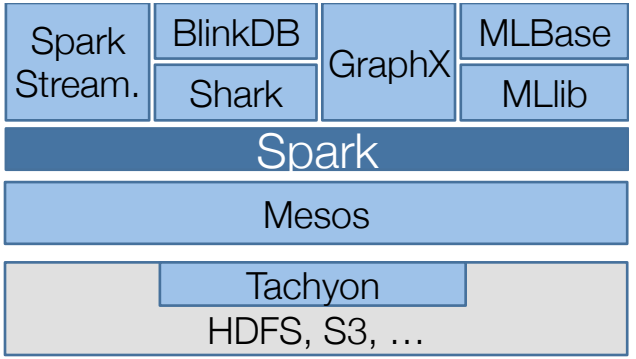
RDD Example

Two-partition RDD A={A_1, A_2} stored on disk

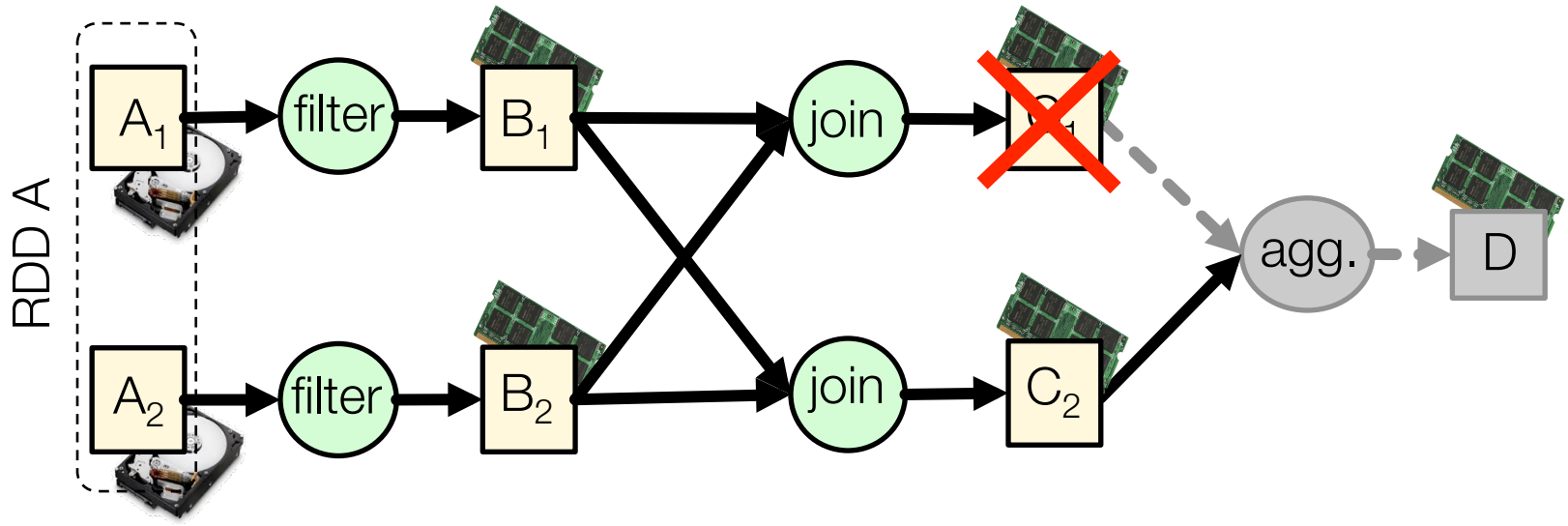
1) filter and cache → RDD B
2) join→ RDD C
3) aggregate → RDD D



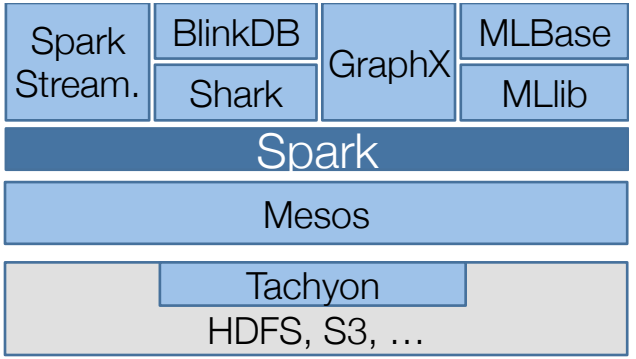
RDD Example



C_1 lost due to node failure before reduce finishes

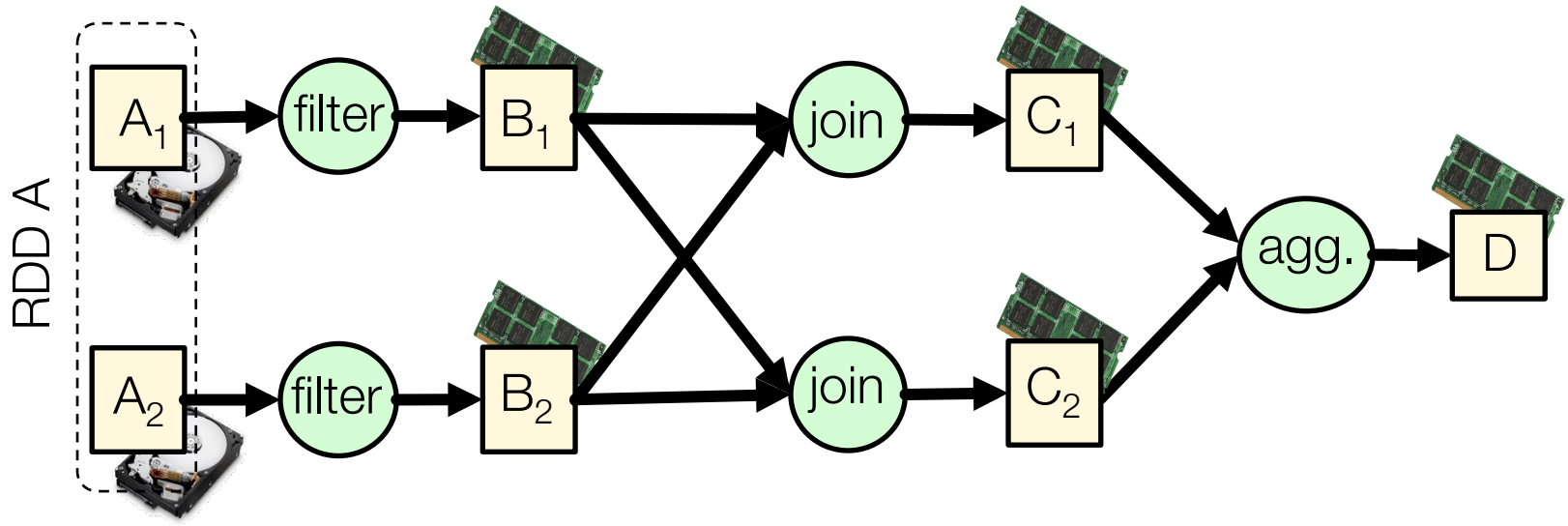


RDD Example

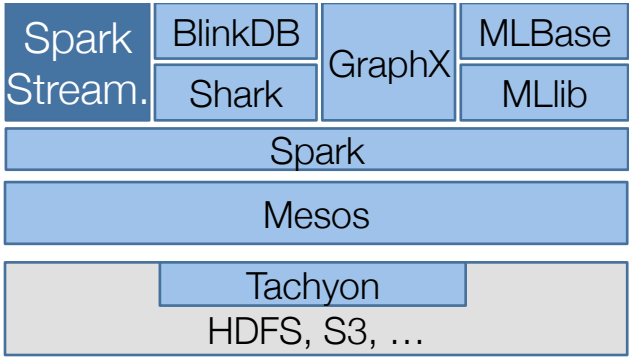


C_1 lost due to node failure before reduce finishes

Reconstruct C_1, eventually, on different node



Spark Streaming

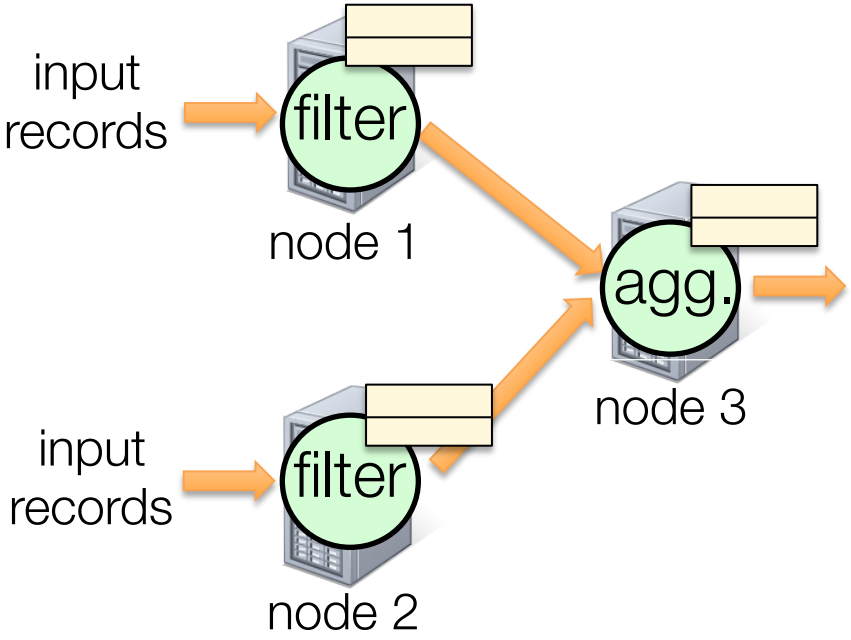


Existing solutions: record-by-record processing

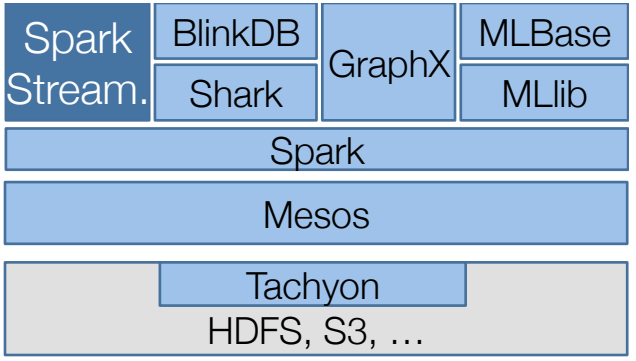
Low latency

Hard to

- » Provide fault tolerance
- » Mitigate straggler

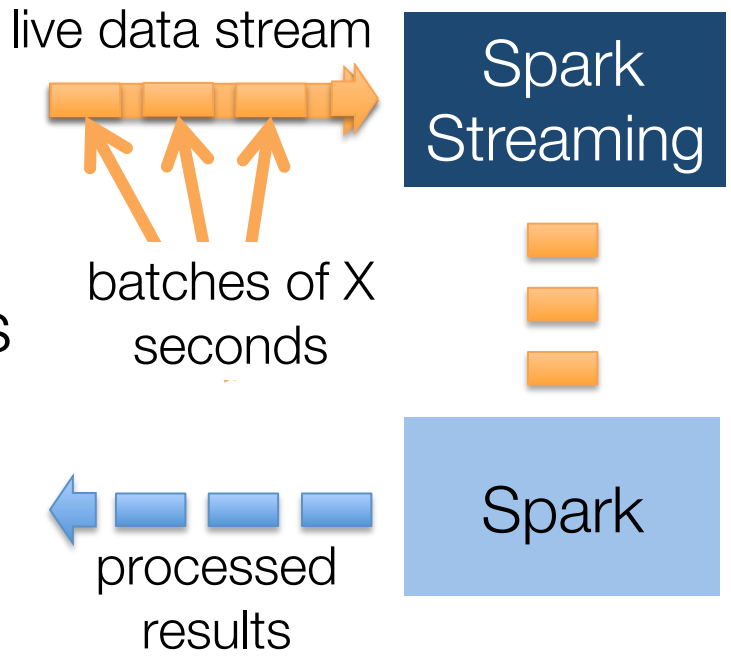


Spark Streaming



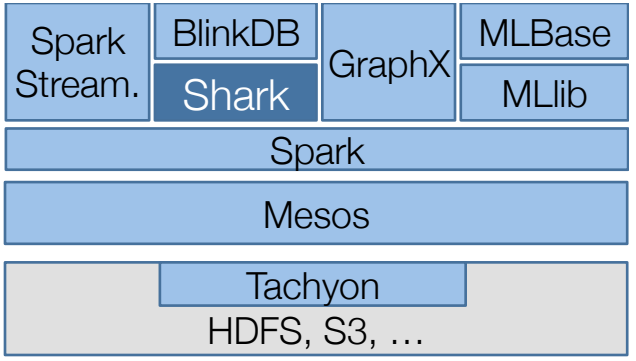
Implemented as sequence of **micro-jobs** (<1s)

- » Fault tolerant
- » Mitigate stragglers
- » Ensure exactly one semantics



Spark & SparkStreaming: batch, interactive, and streaming computations

Shark



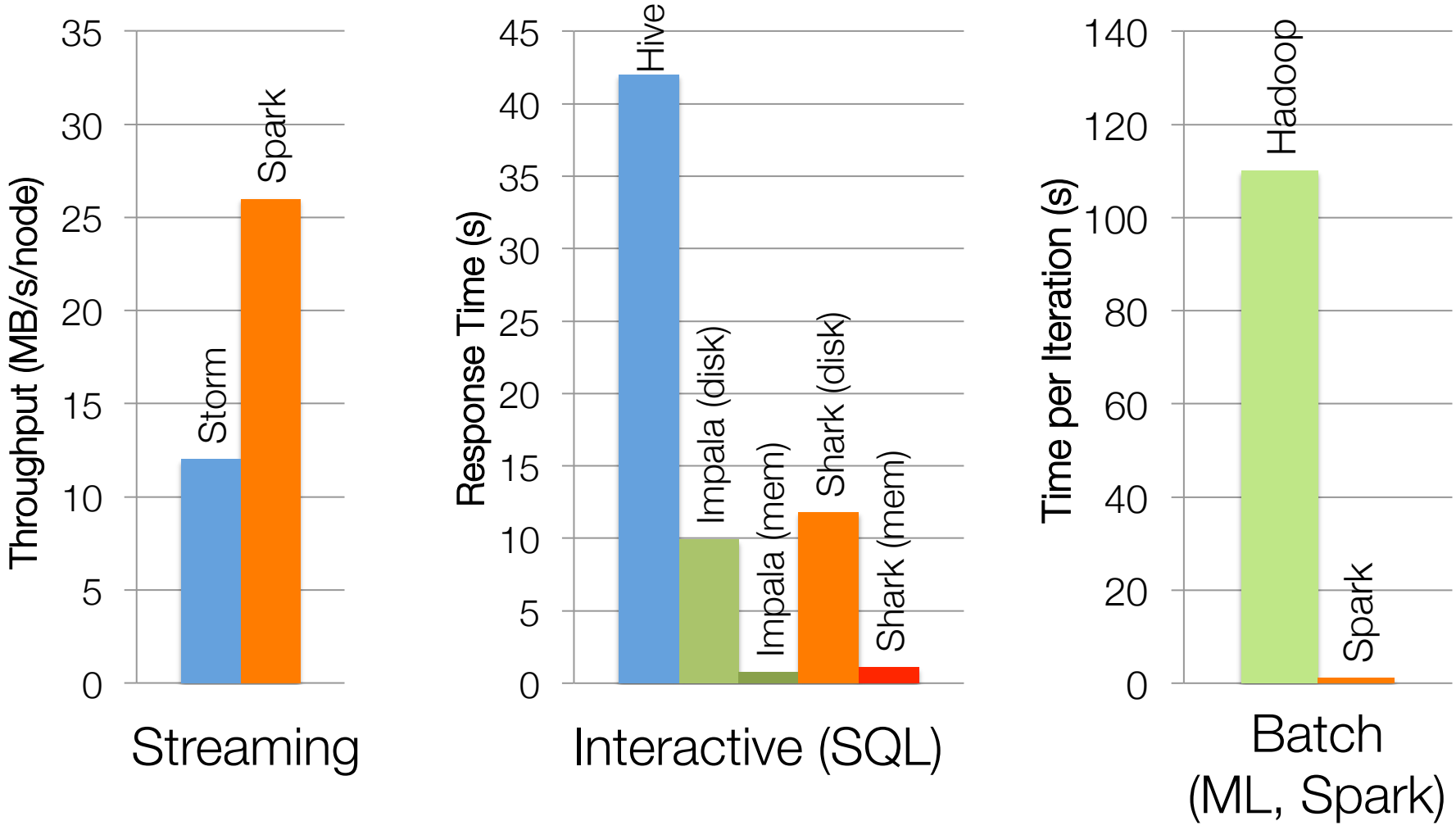
Hive over Spark: full support for HQL and UDFs

Up to 100x when input is in memory

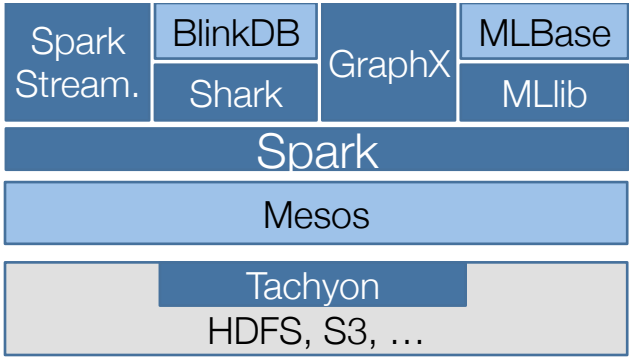
Up to 5-10x when input is on disk

Running on hundreds of nodes at Yahoo!

Not Only General, but Fast



Spark Distribution



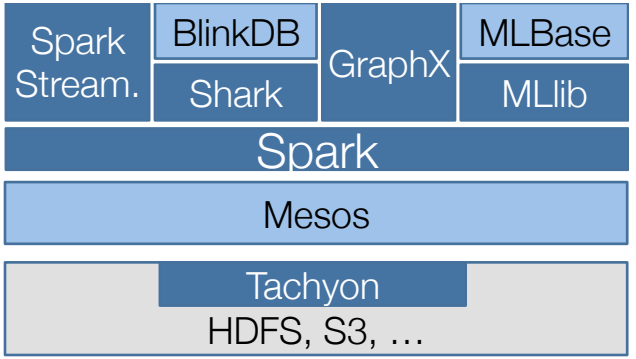
Includes

- Spark (core)
- Spark Streaming
- GraphX (alpha release)
- MLlib

In the future:

- Shark
- Tachyon

Explosive Growth



2,500+ Spark meetup users

180+ contributors from 30+ companies

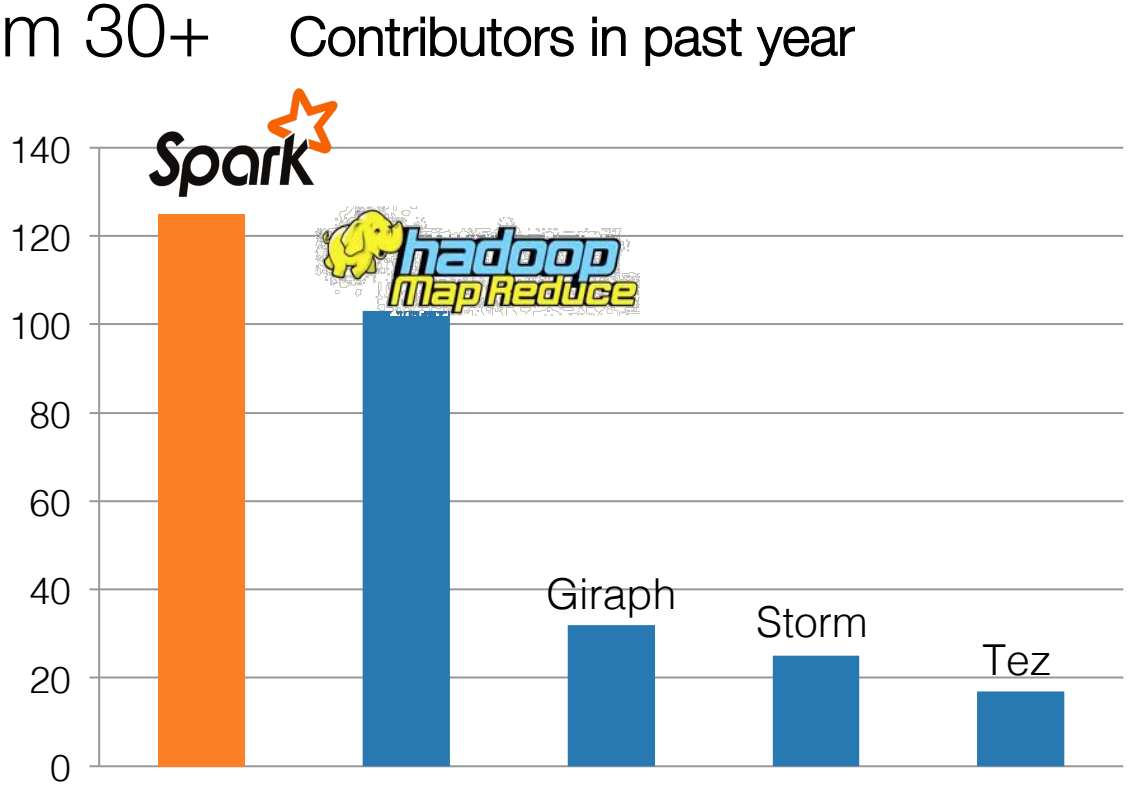
1st Spark Summit

» 450+ attendees

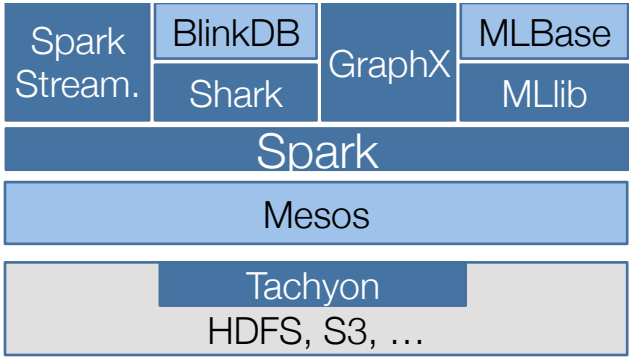
» 140+ companies

2nd Spark Summit

» June 30 – July 2



Explosive Growth



Databricks: founded in 2013 to commercialize Spark Platform

Included in all major Hadoop Distributions

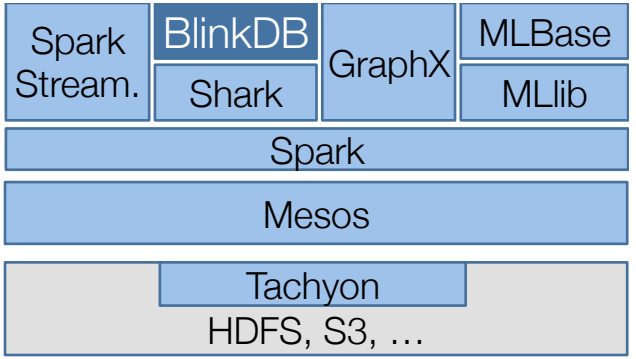
- » Cloudera
- » MapR
- » Hortonworks (technical preview)

Enterprise support: Cloudera, MapR, Datastax

Spark and Shark available on Amazon's EMR

DATABRICKS cloudera Ask Bigger Questions MAPR TECHNOLOGIES Hortonworks amazon web services

BlinkDB

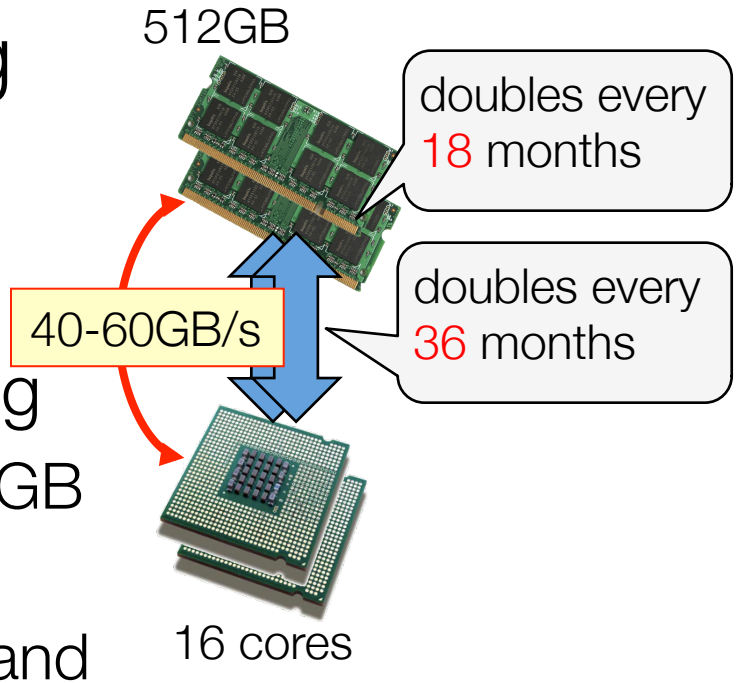


Trade between query performance and accuracy using sampling

Why?

» In-memory processing doesn't guarantee interactive processing

- E.g., ~10's sec just to scan 512 GB RAM!
- Gap between memory capacity and transfer rate increasing

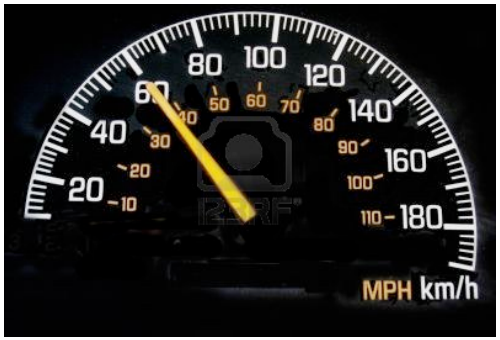


Key Insight

Computations don't always need ***exact*** answers

- Input often *noisy*: exact computations do *not* guarantee exact answers
- *Error* often acceptable if *small* and *bounded*

Best scale
± 200g error



Speedometers
± 2.5 % error
(edmunds.com)

OmniPod Insulin Pump
± 0.96 % error
(www.ncbi.nlm.nih.gov/pubmed/22226273

Approach: Sampling

Compute results on samples instead of full data

» Typically, error depends on sample size (n) ***not*** on original data size, i.e., **error** $\alpha\ 1/\sqrt{n}$

Can trade between answer's ***latency*** and ***accuracy*** and ***cost***

BlinkDB Interface

SELECT avg(sessionTime)

FROM Table

WHERE city='San Francisco' AND 'dt=2012-9-2'

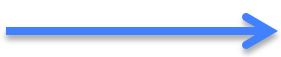
WITHIN 1 SECONDS  234.23 ± 15.32

BlinkDB Interface

SELECT avg(sessionTime)

FROM Table

WHERE city='San Francisco' AND 'dt=2012-9-2'

WITHIN 2 SECONDS  ~~234.23 ± 15.32~~

239.46 ± 4.96

SELECT avg(sessionTime)

FROM Table

WHERE city='San Francisco' AND 'dt=2012-9-2'

ERROR 0.1 **CONFIDENCE** 95.0%

Quick Results

Dataset

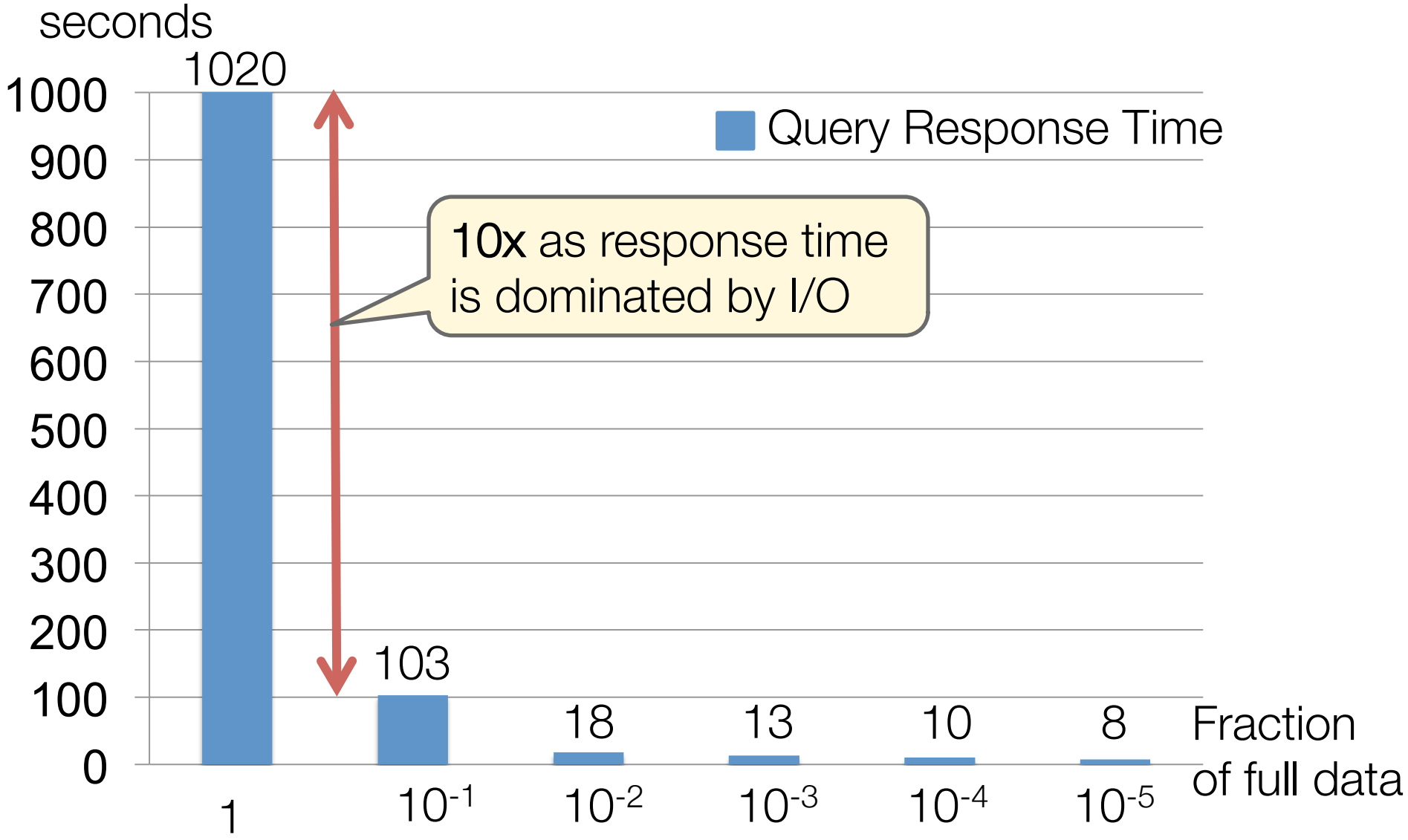
- 365 mil rows, 204GB on disk
- 600+ GB in memory (deserialized format)

Query: query computing 95-th percentile

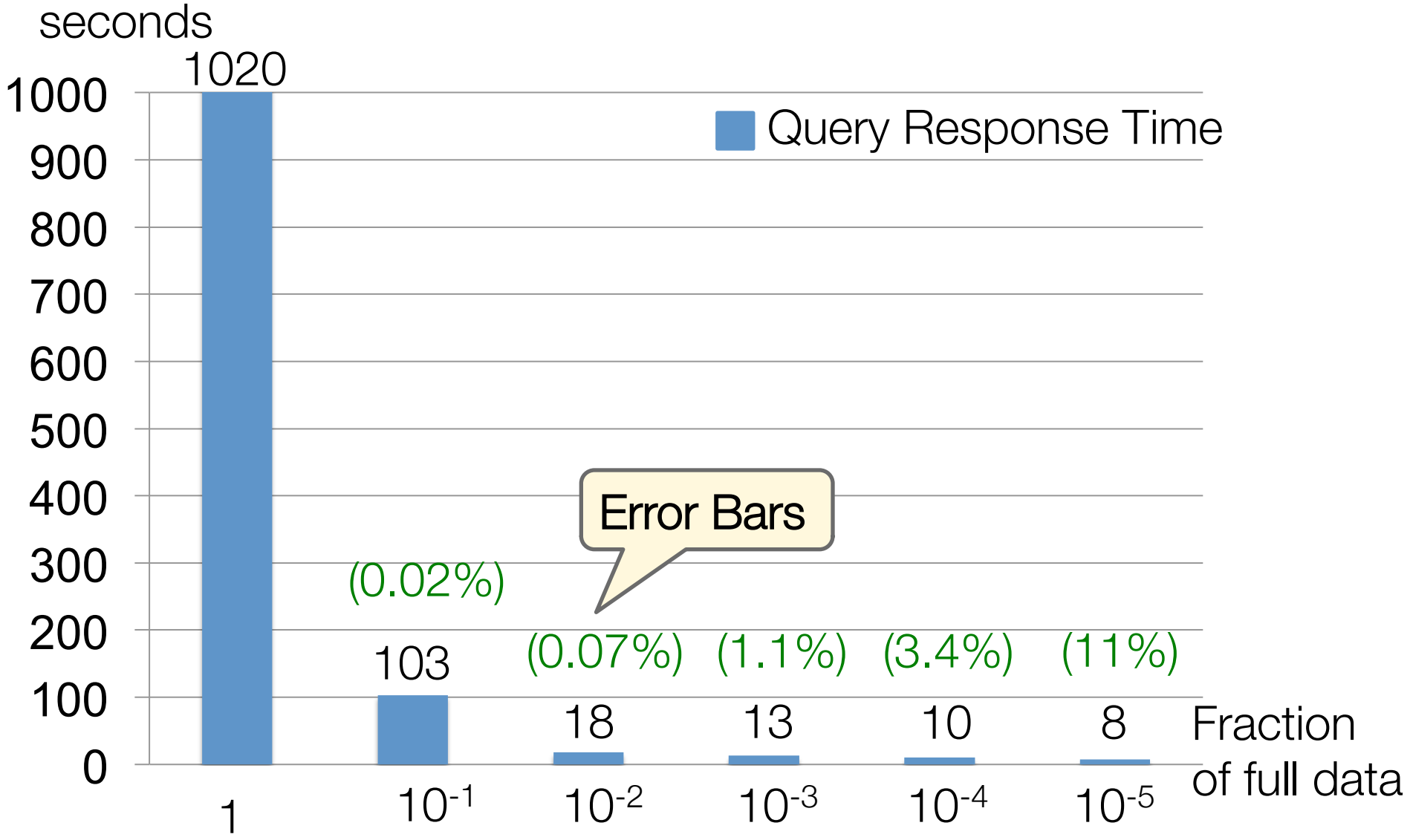
25 EC2 instances with

- 4 cores
- 15GB RAM

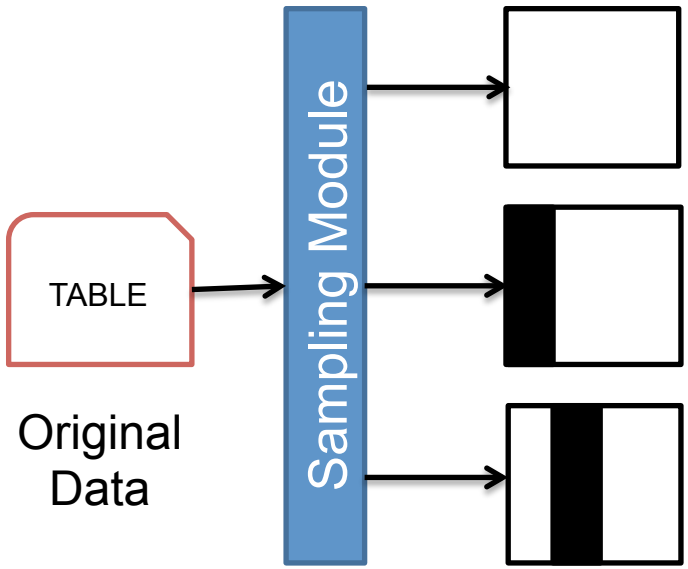
Query Response Time



Query Response Time

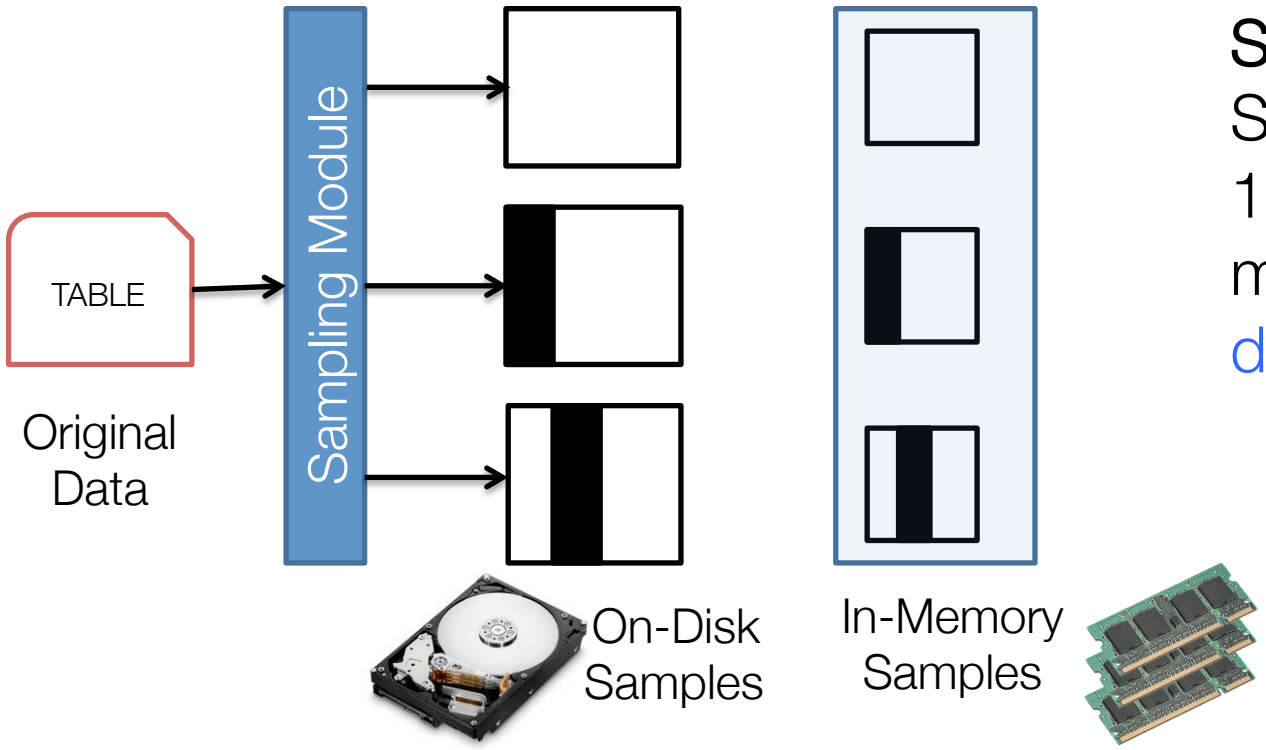


BlinkDB Overview



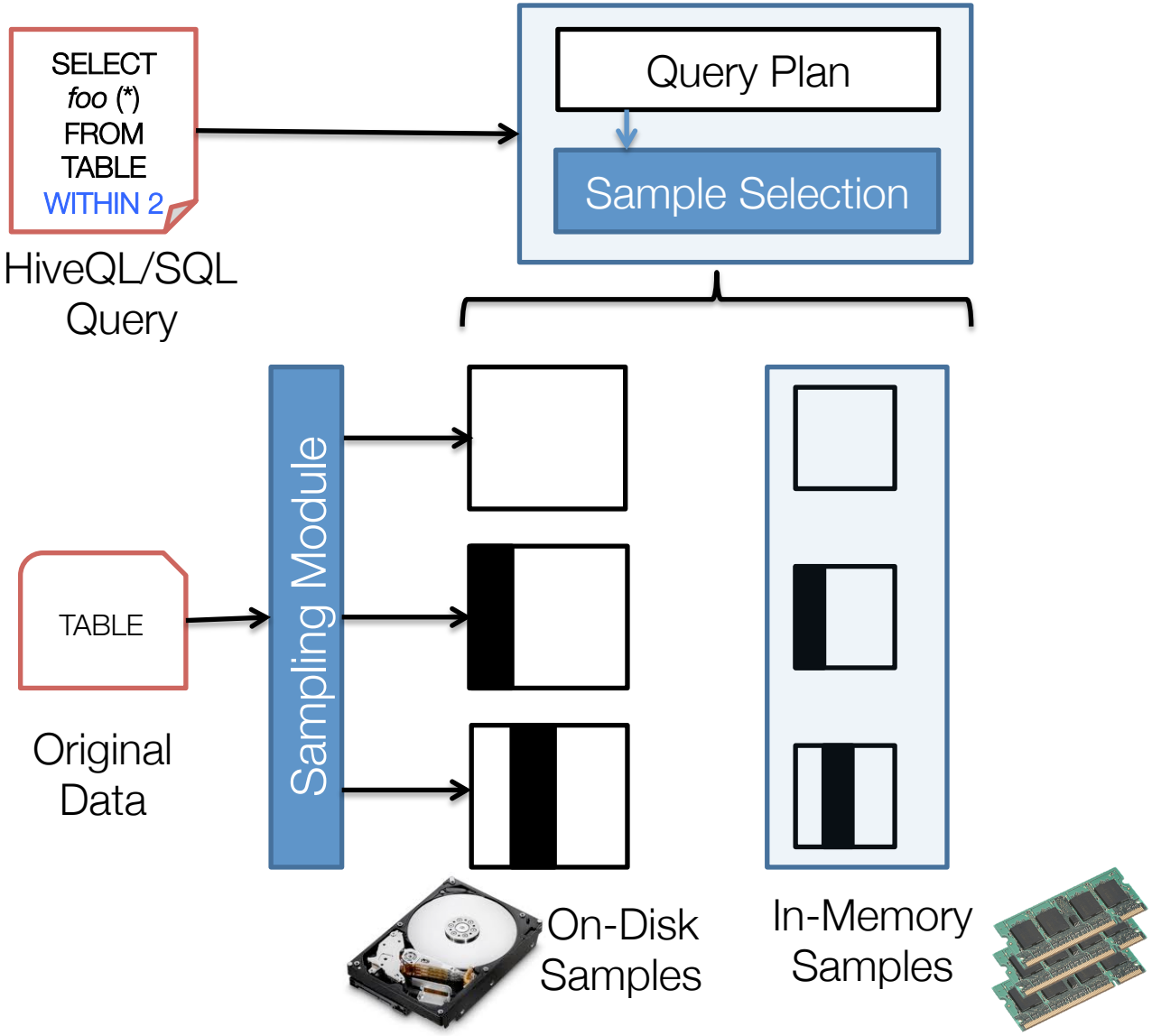
Offline-sampling: Optimal set of samples across different dimensions (*columns or sets of columns*) to support ad-hoc exploratory queries

BlinkDB Overview

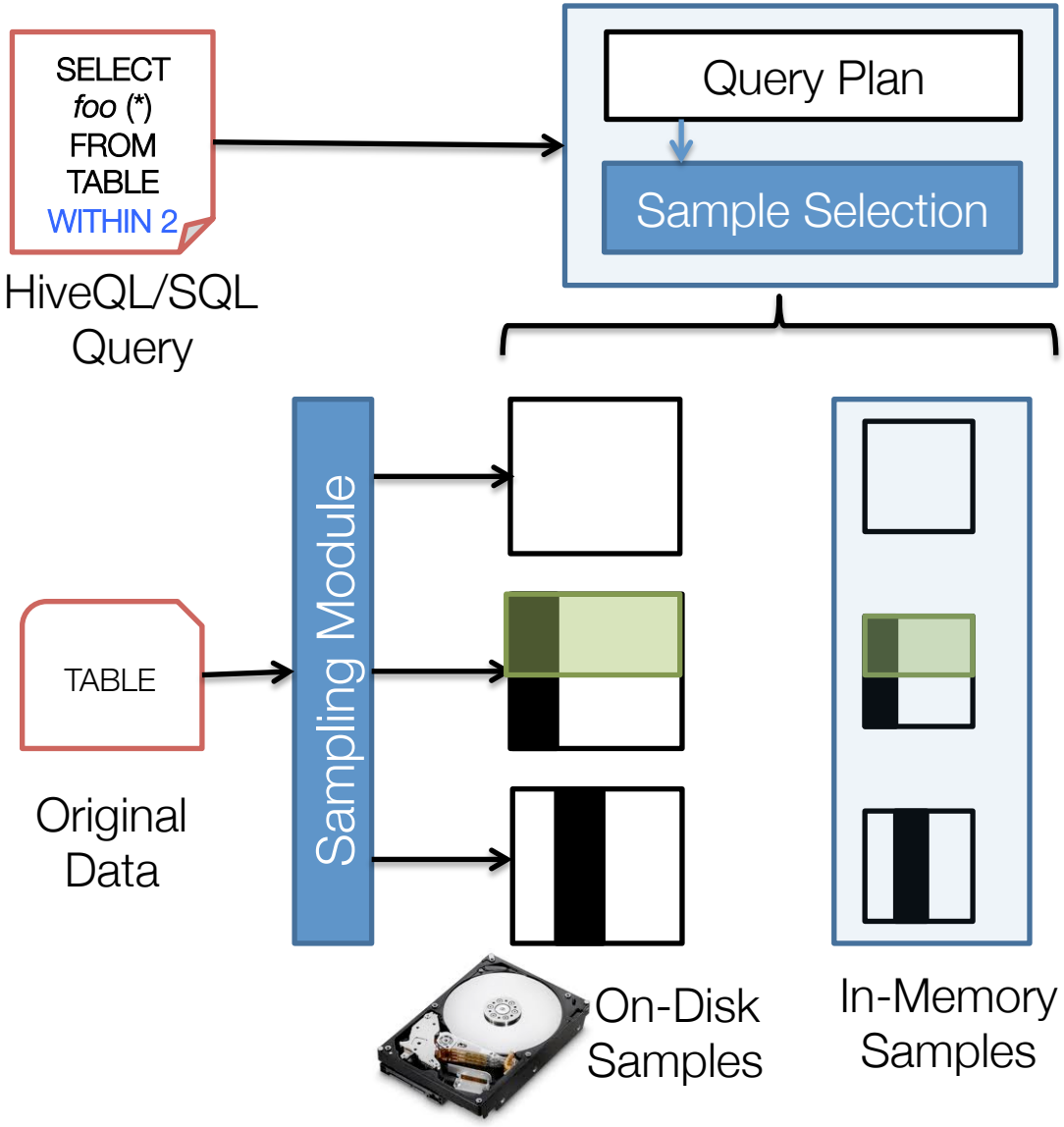


Sample Placement: Samples striped over 100s or 1,000s of machines both on disks and in-memory.

BlinkDB Overview

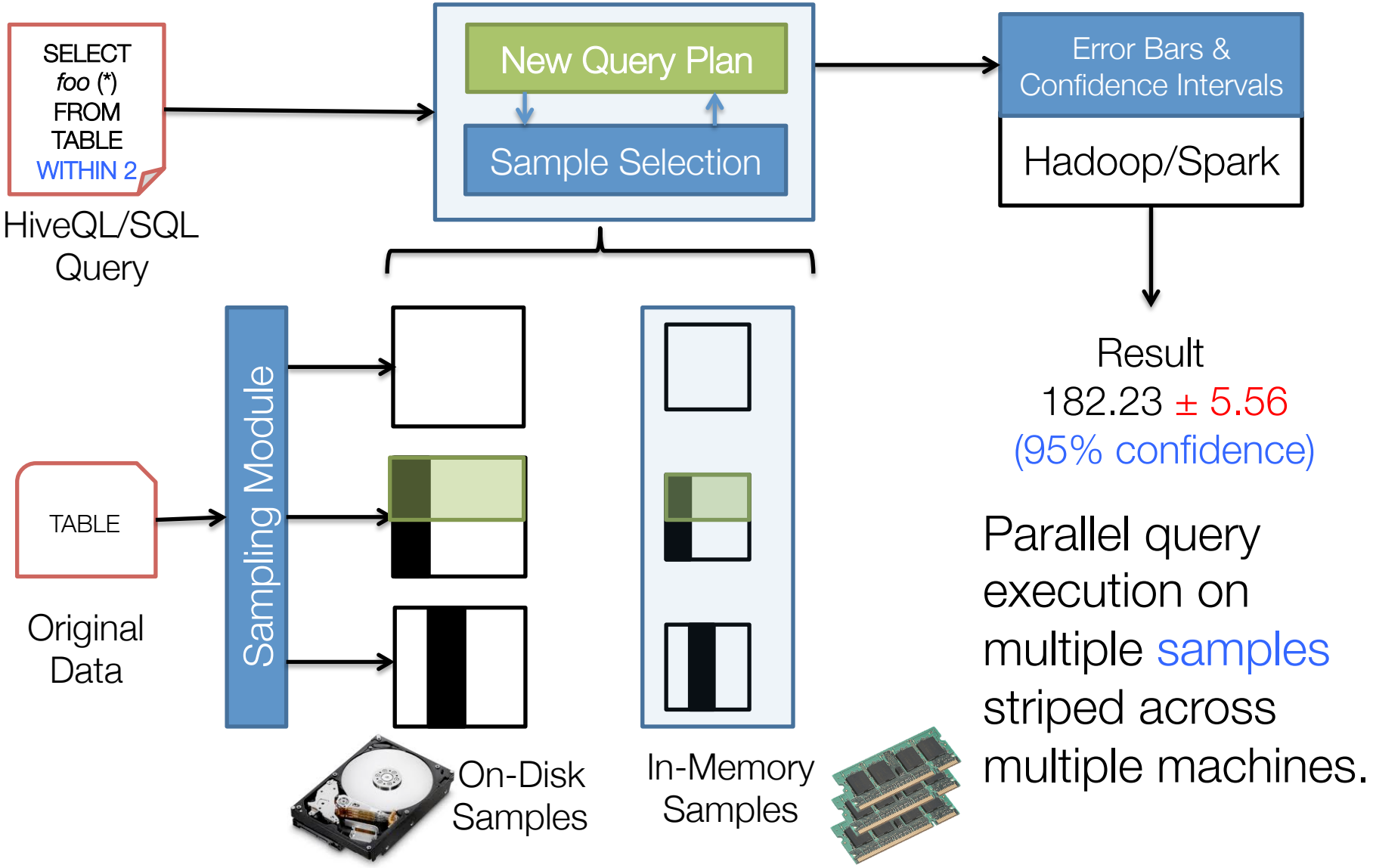


BlinkDB Overview



Online sample selection to pick best sample(s) based on query latency and accuracy requirements

BlinkDB Overview



BlinkDB Challenges

Which set of samples to build given a storage budget?

Which sample to run the query on?

How to accurately estimate the error?

BlinkDB Challenges

Which set of samples to build given a storage budget?

Which sample to run the query on?

How to accurately estimate the error?

How to Accurately Estimate Error?

Close formulas for limited number of operators

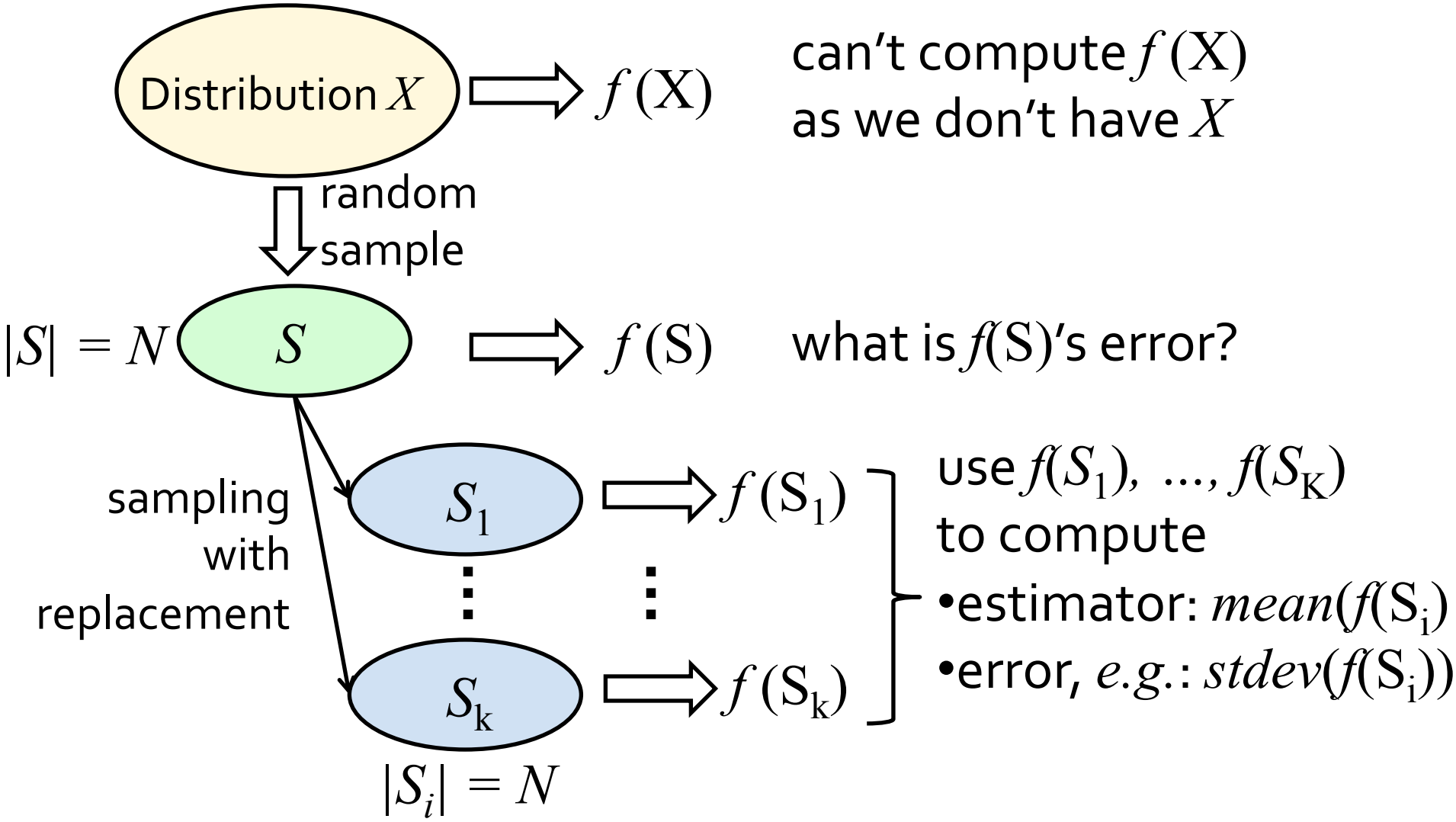
- E.g., count, mean, percentiles

What about user defined functions (UDFs)?

- Use bootsrap technique

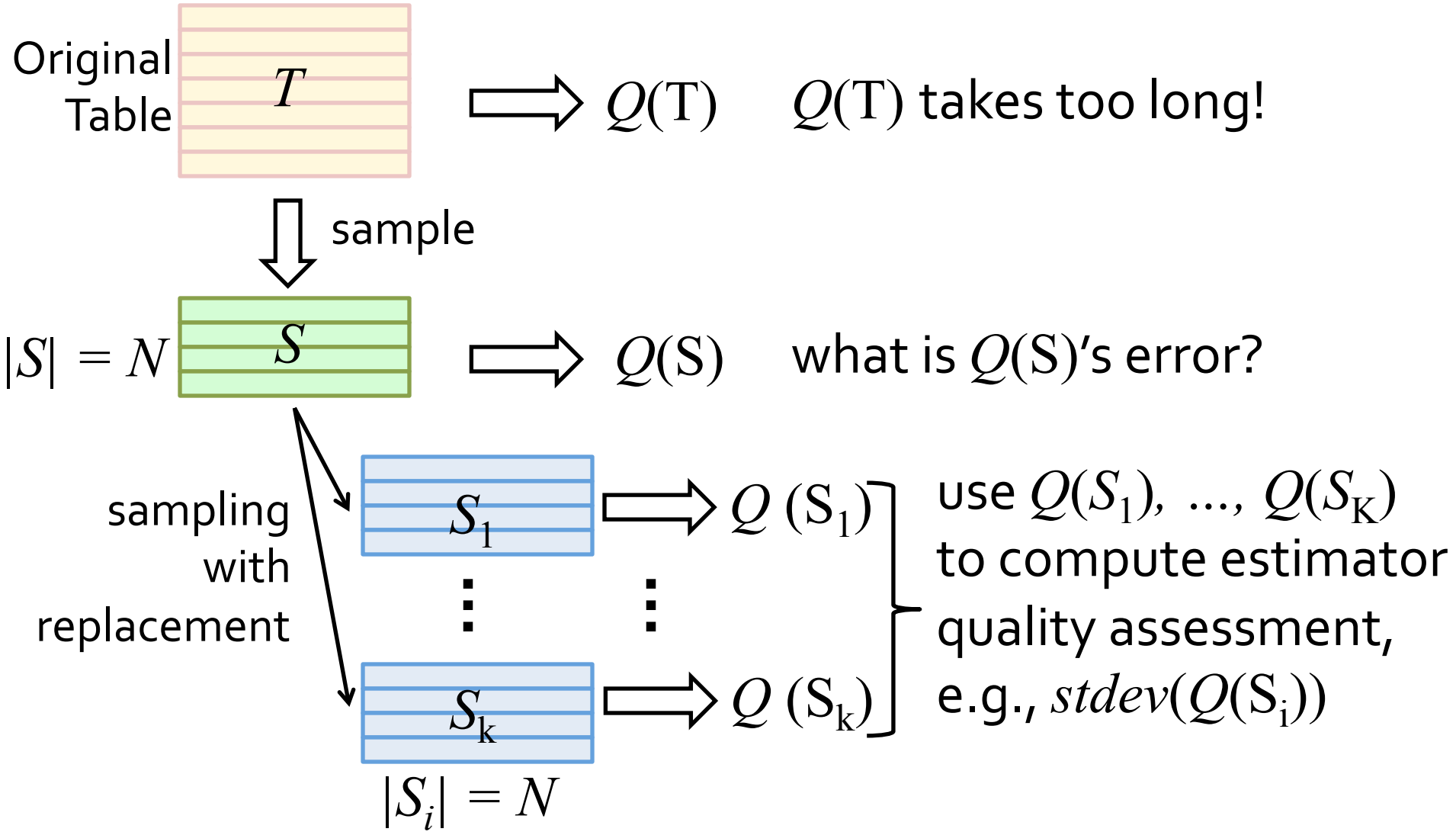
Bootstrap

Quantify accuracy of a sample estimator, $f()$



Bootstrap for BlinkDB

Quantify accuracy of a query on a sample table



How Do You Know Error Estimation is Correct?

Assumption: $f()$ is Hadamard differentiable

- » How do you know an UDF is Hadamard differentiable?
- » Sufficient, not necessary condition

Only **approximations** of true error distribution (true for closed formula as well)

Previous work doesn't address error estimation correctness

How Bad it Is?

Workloads

- » Conviva: **268** real-world **113** had custom User-Defined Functions
- » Facebook

Closed Forms/Bootstrap fails for

- » **3 in 10** Conviva Queries
- » **4 in 10** Facebook Queries

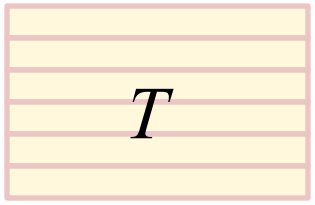
Need runtime diagnosis!

Error Diagnosis

Compare bootstrapping with **ground truth** for small samples

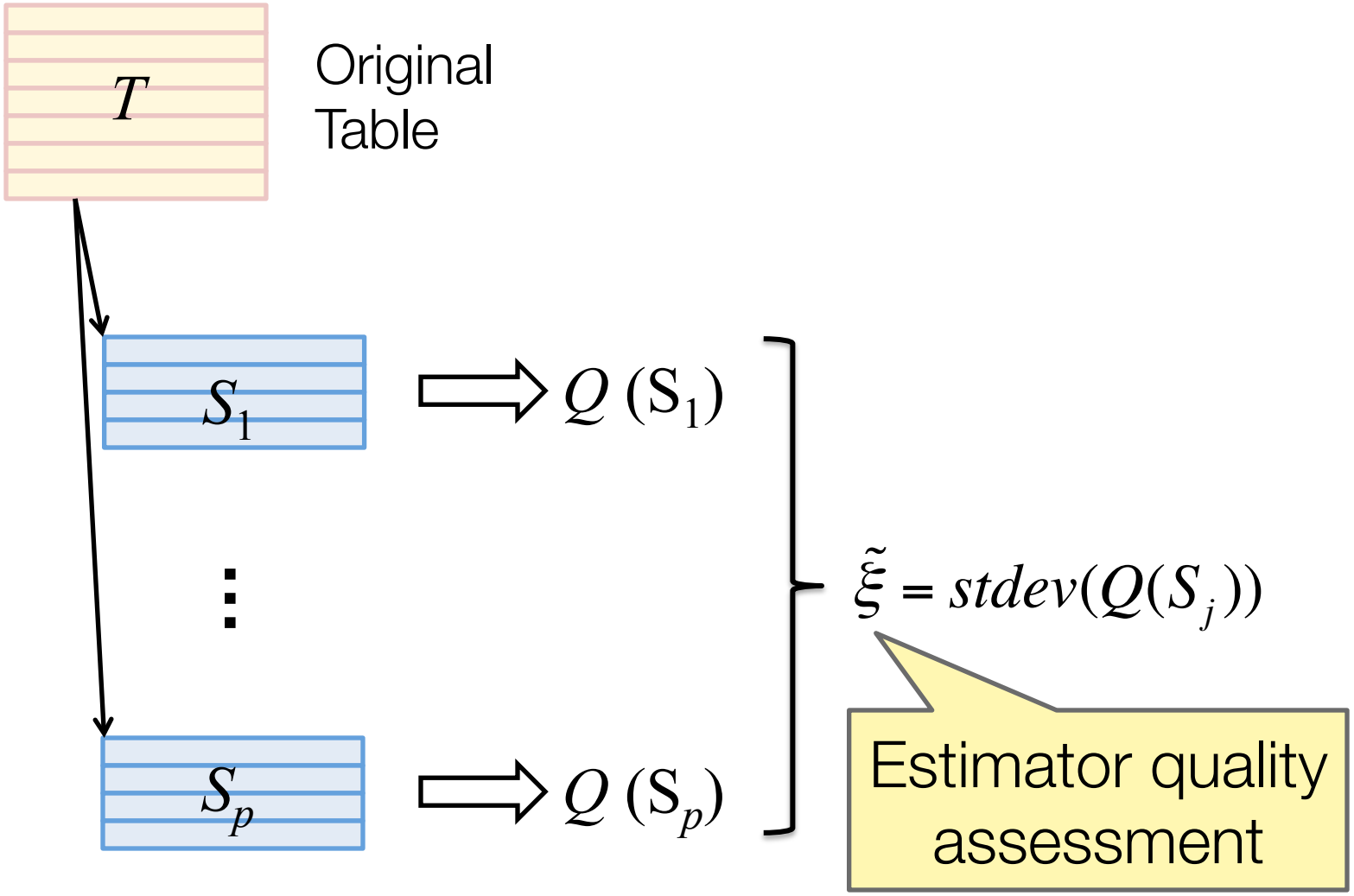
Check whether error improves as sample size increases

Ground Truth (Approximation)

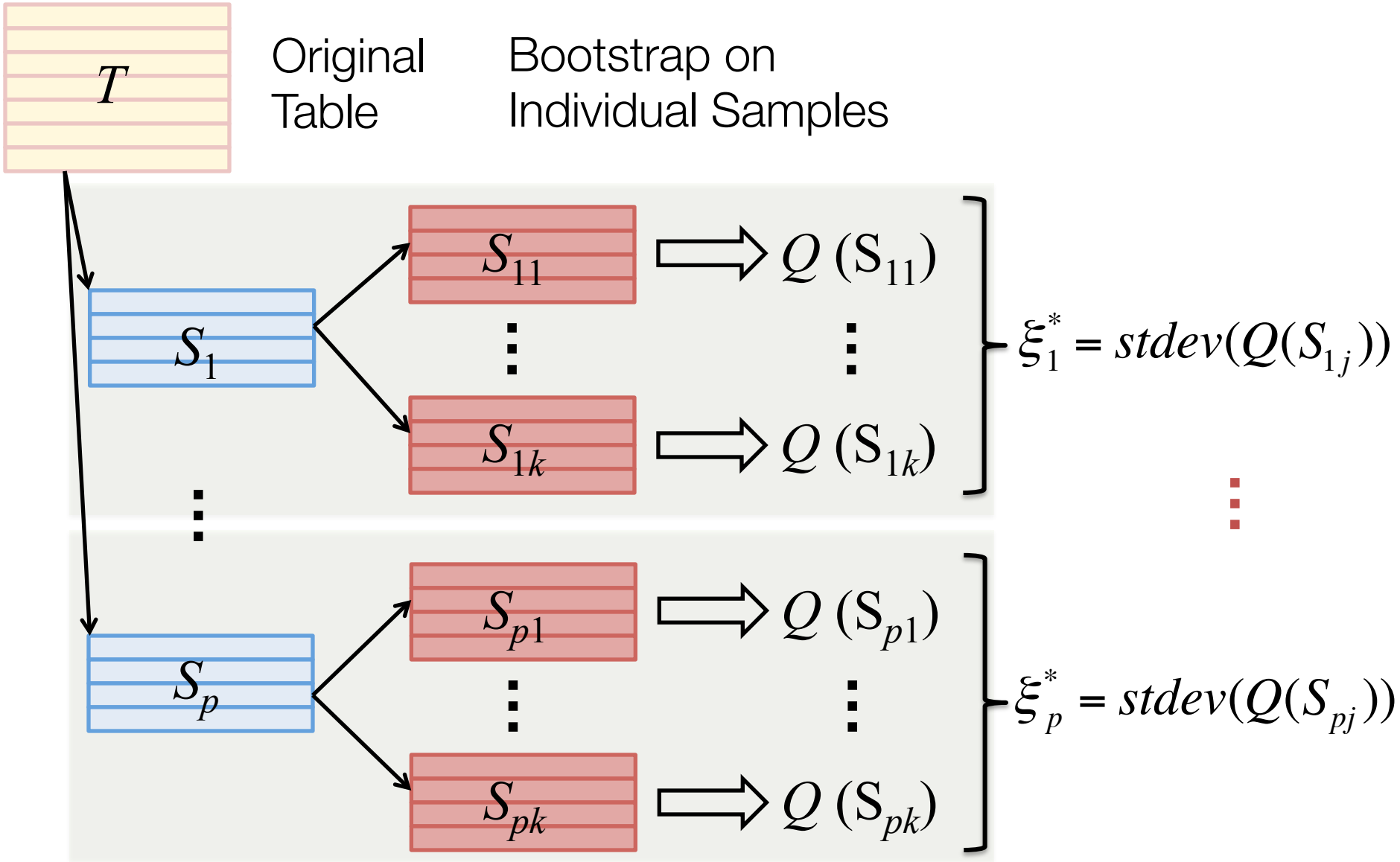


Original
Table

Ground Truth (Approximation)



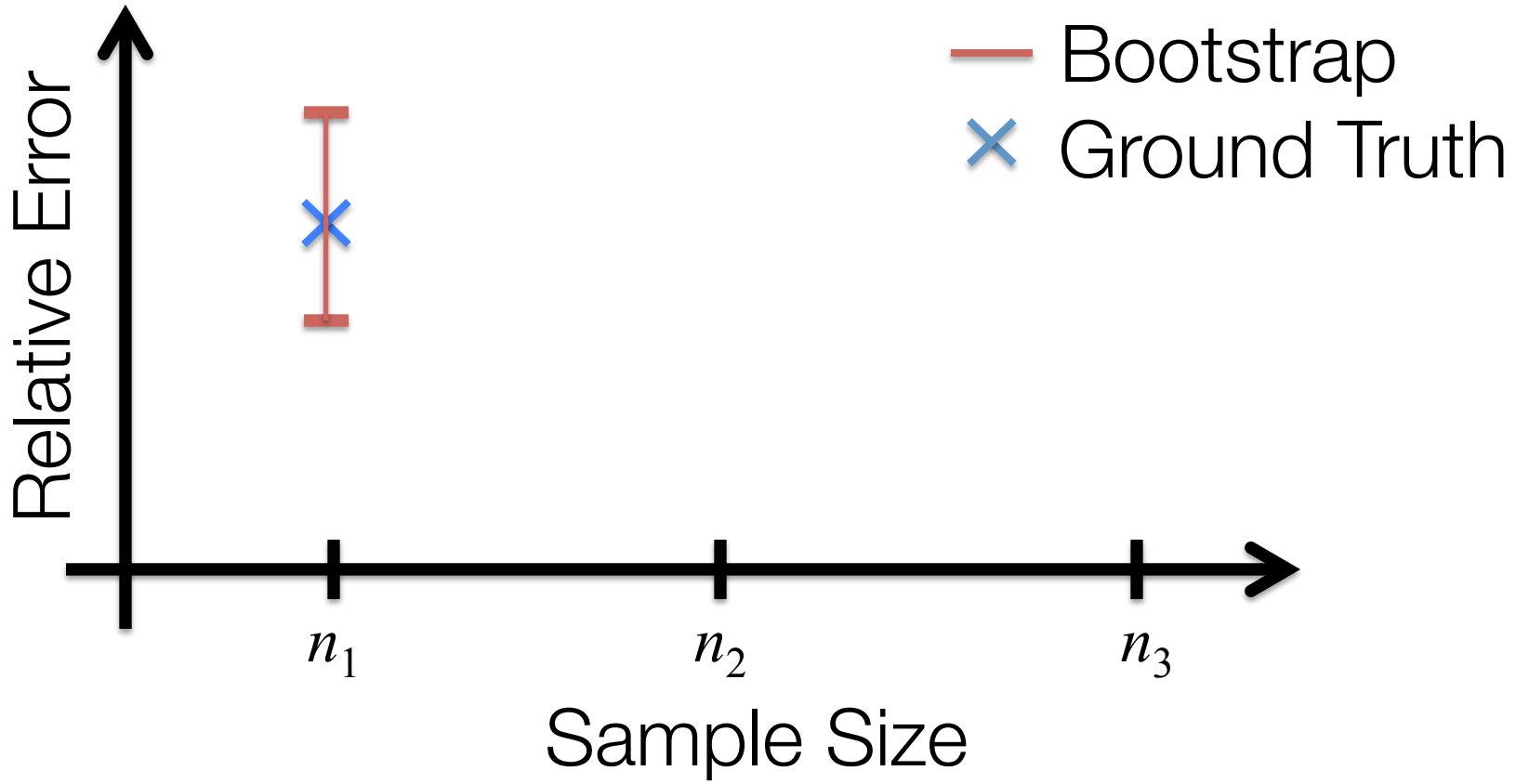
Ground Truth and Bootstrap



Ground Truth vs. Bootstrap

$$\tilde{\xi}_i = mean(\xi_{ij})$$

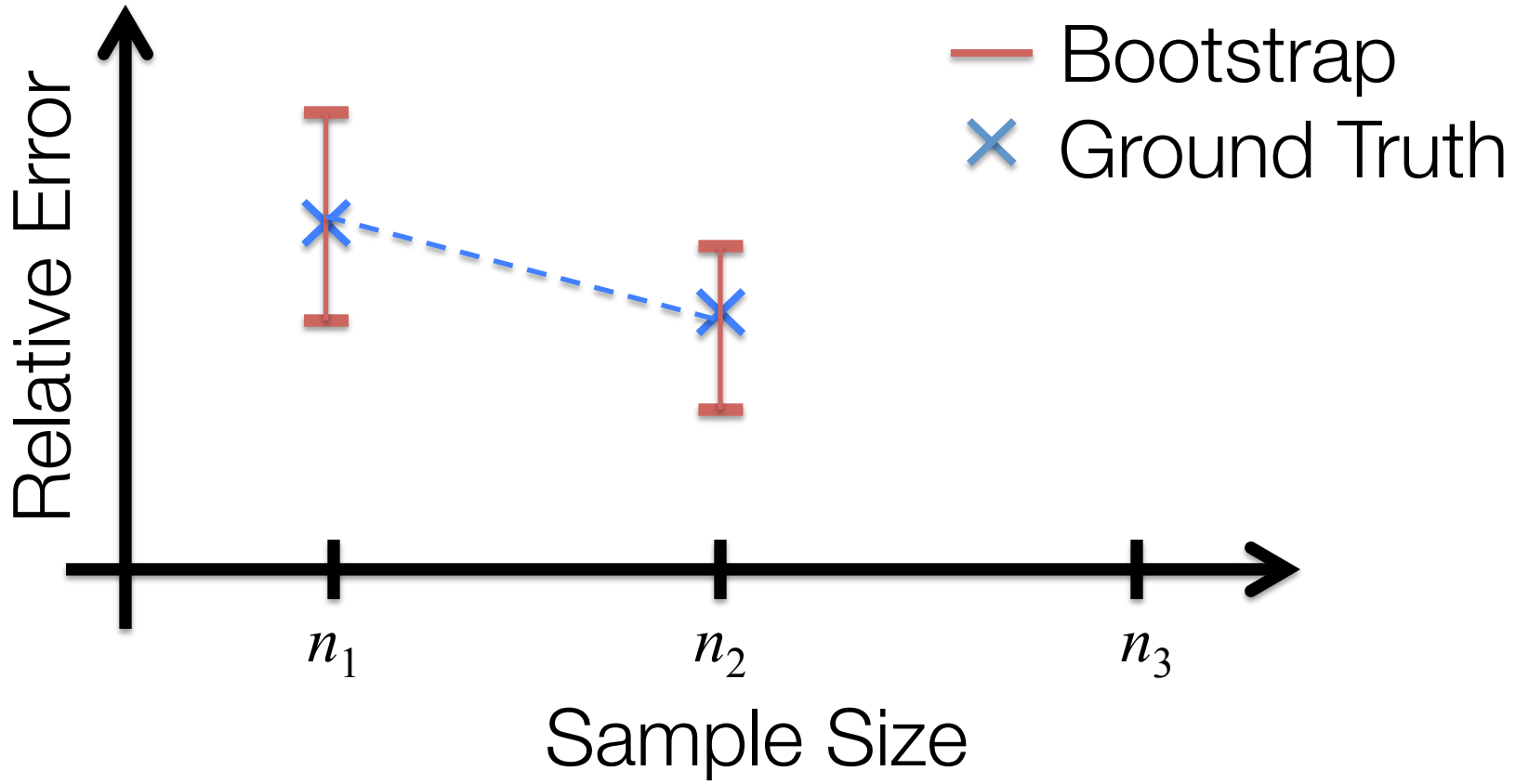
$$\xi^*_{i1} = stdev(Q(S_{i1j})) \quad \cdots \quad \xi^*_{ip} = stdev(Q(S_{ipj}))$$



Ground Truth vs. Bootstrap

$$\tilde{\xi}_i = mean(\xi_{ij})$$

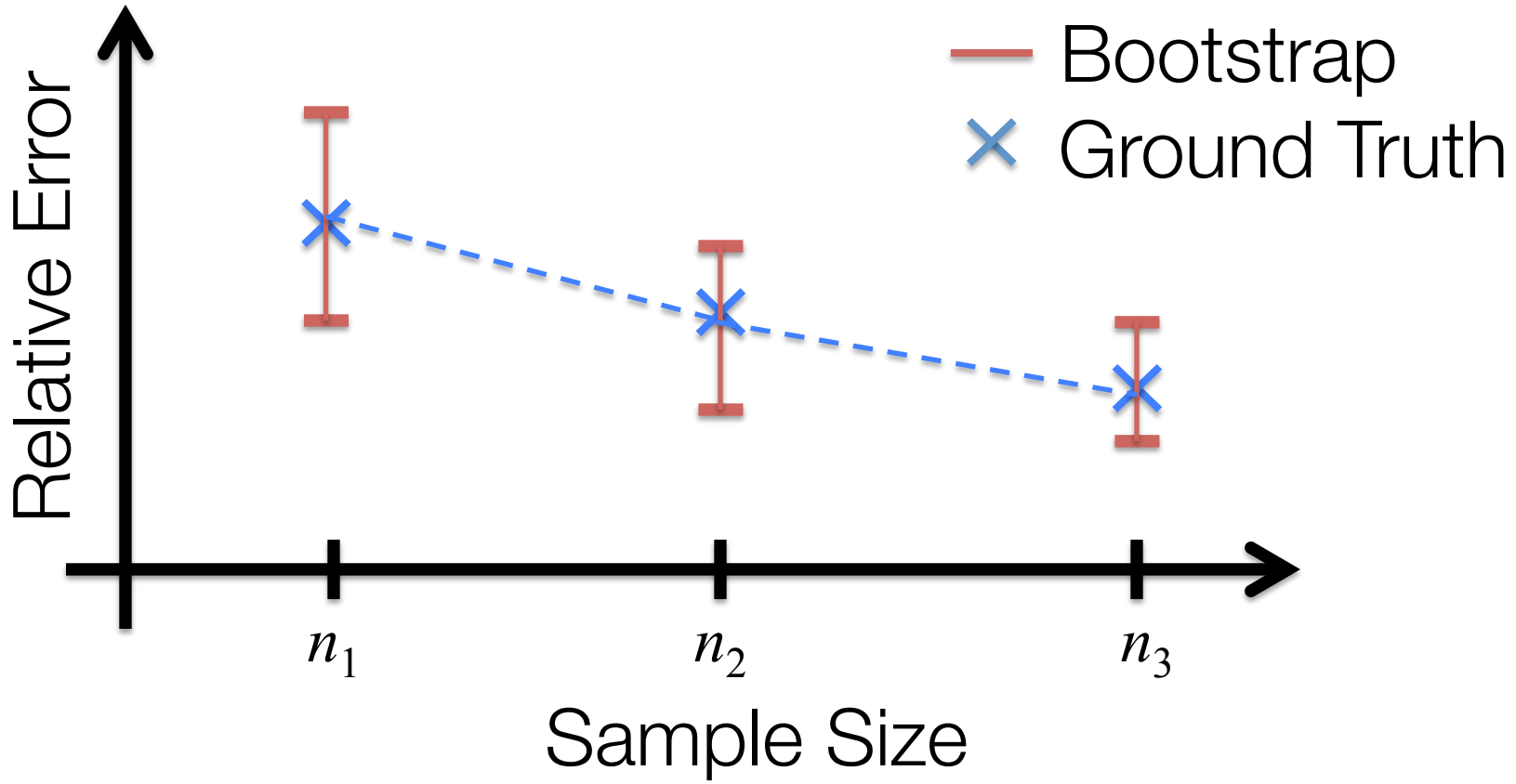
$$\xi^*_{i1} = stdev(Q(S_{i1j})) \quad \cdots \quad \xi^*_{ip} = stdev(Q(S_{ipj}))$$



Ground Truth vs. Bootstrap

$$\tilde{\xi}_i = mean(\xi_{ij})$$

$$\xi^*_{i1} = stdev(Q(S_{i1j})) \quad \cdots \quad \xi^*_{ip} = stdev(Q(S_{ipj}))$$



How Well Does it Work in Practice?

Evaluated on Conviva Query Workload

Diagnostic predicted that 207 (77%) queries can be approximated

- False Negatives: 18
- False Positives: 3 (conditional UDFs)

Overhead

Boostrap and Diagnostic overheads can be very large

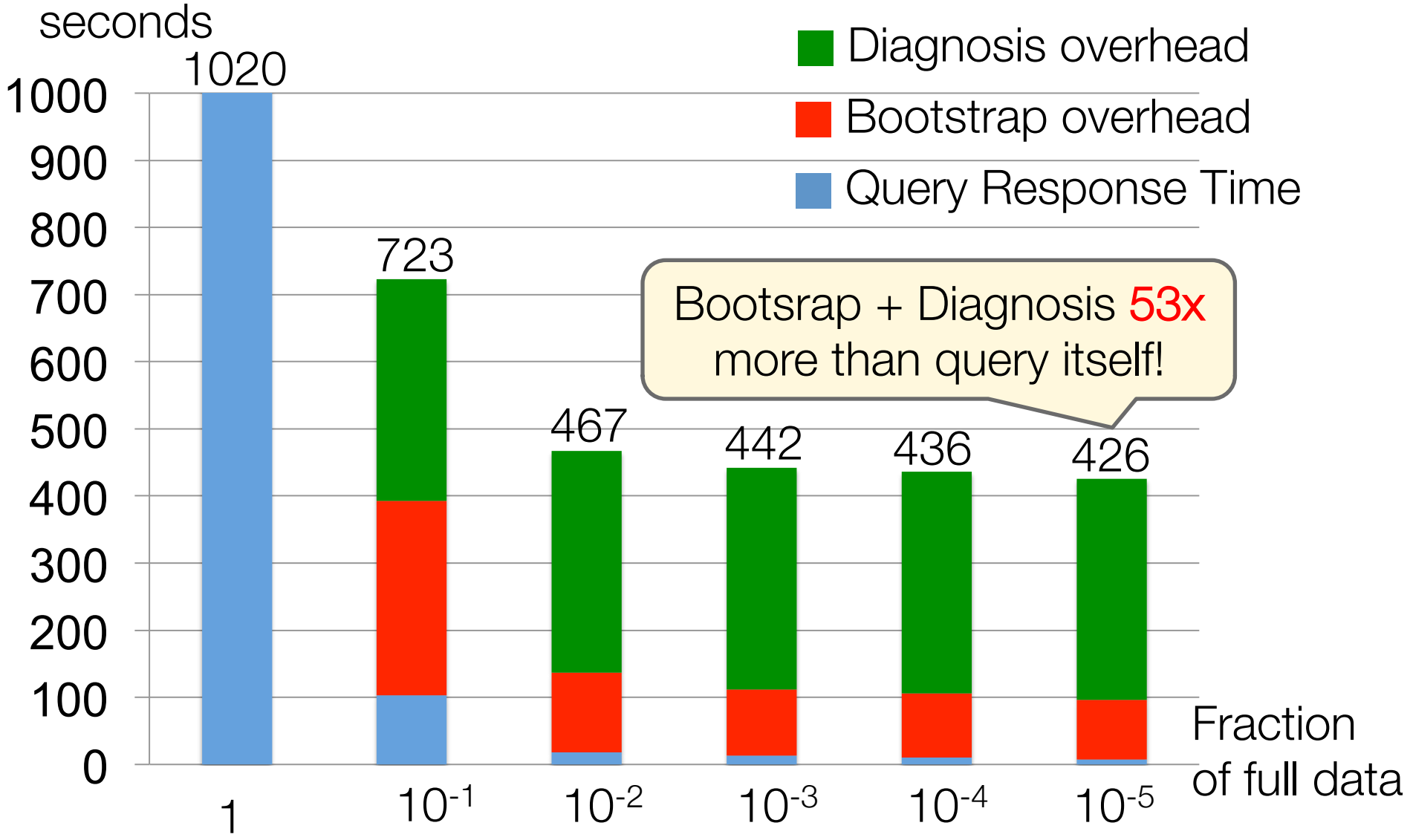
- » Diagnostics requires to run 30,000 small queries!

Optimization

- » Pushdown filter
- » One pass execution

Can reduce overhead by orders of magnitude

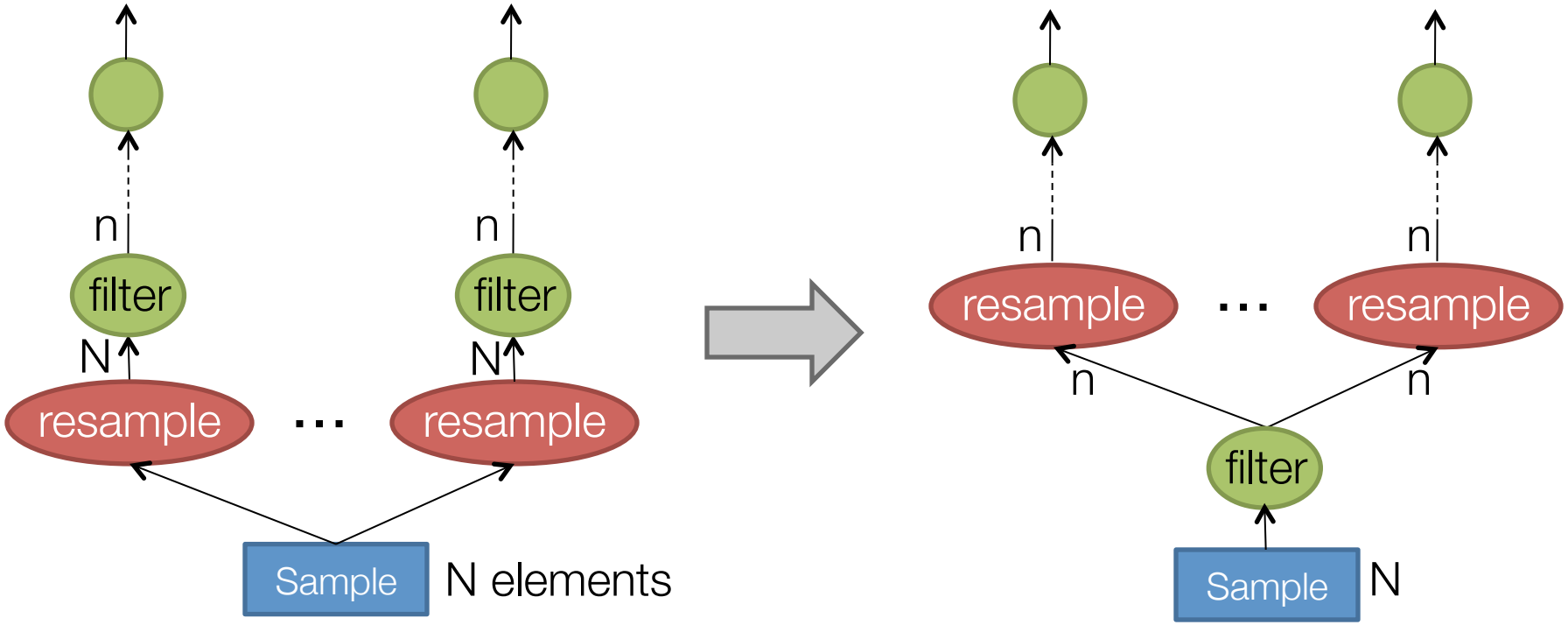
Query + Bootstrap + Diagnosis



Optimization: Filter Pushdown

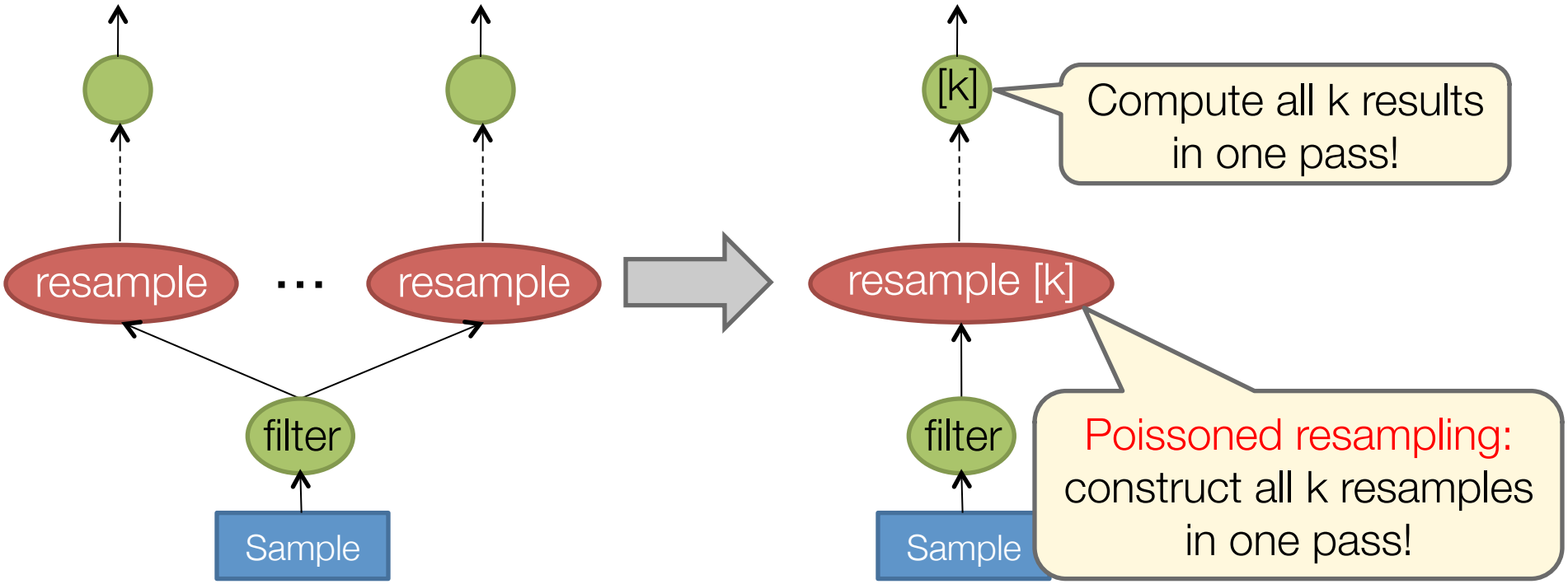
Perform filtering before resampling

» Can dramatically reduce I/O



Assume n << N

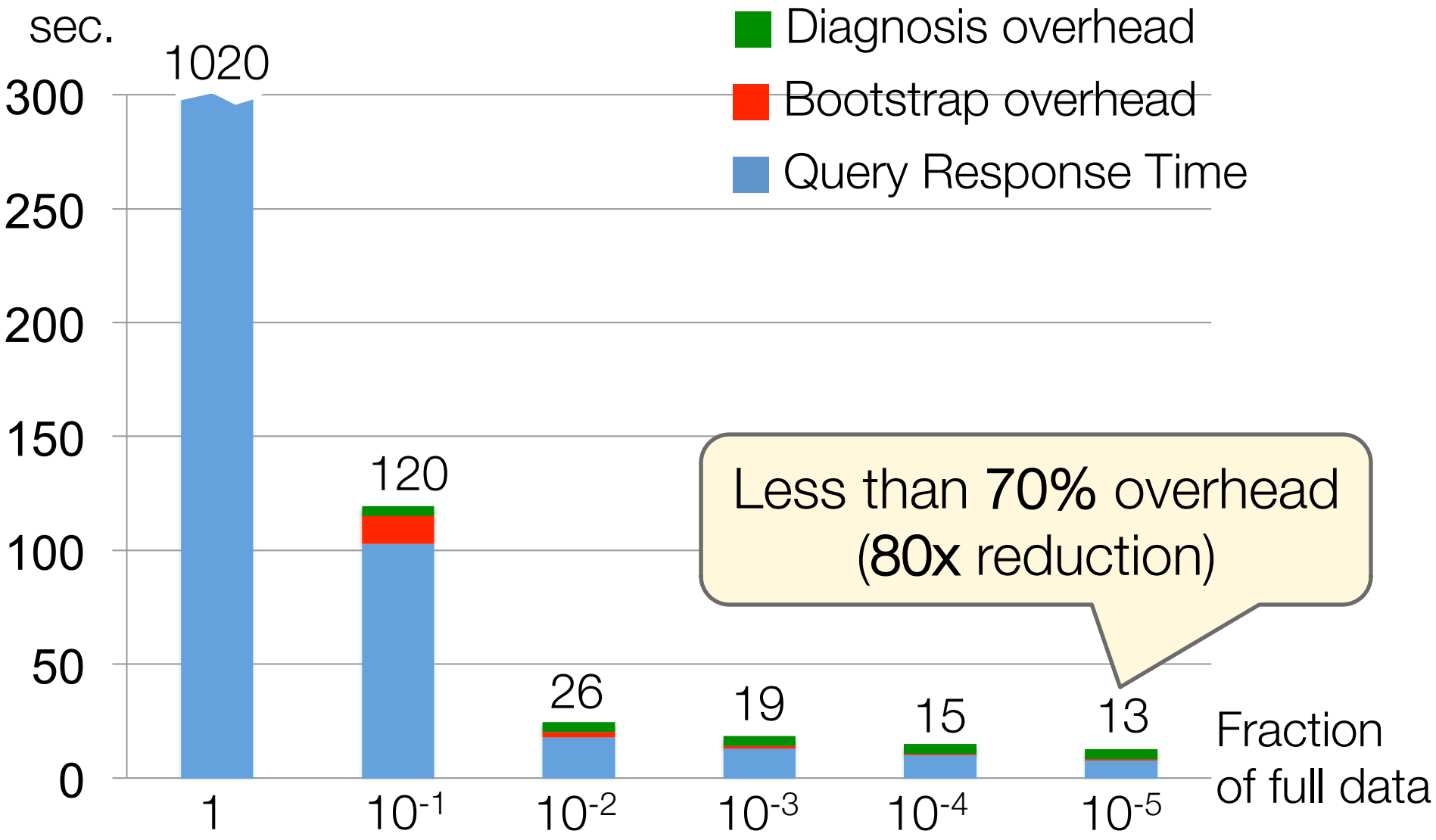
Optimization: One Pass Exec.



For each resample add new column

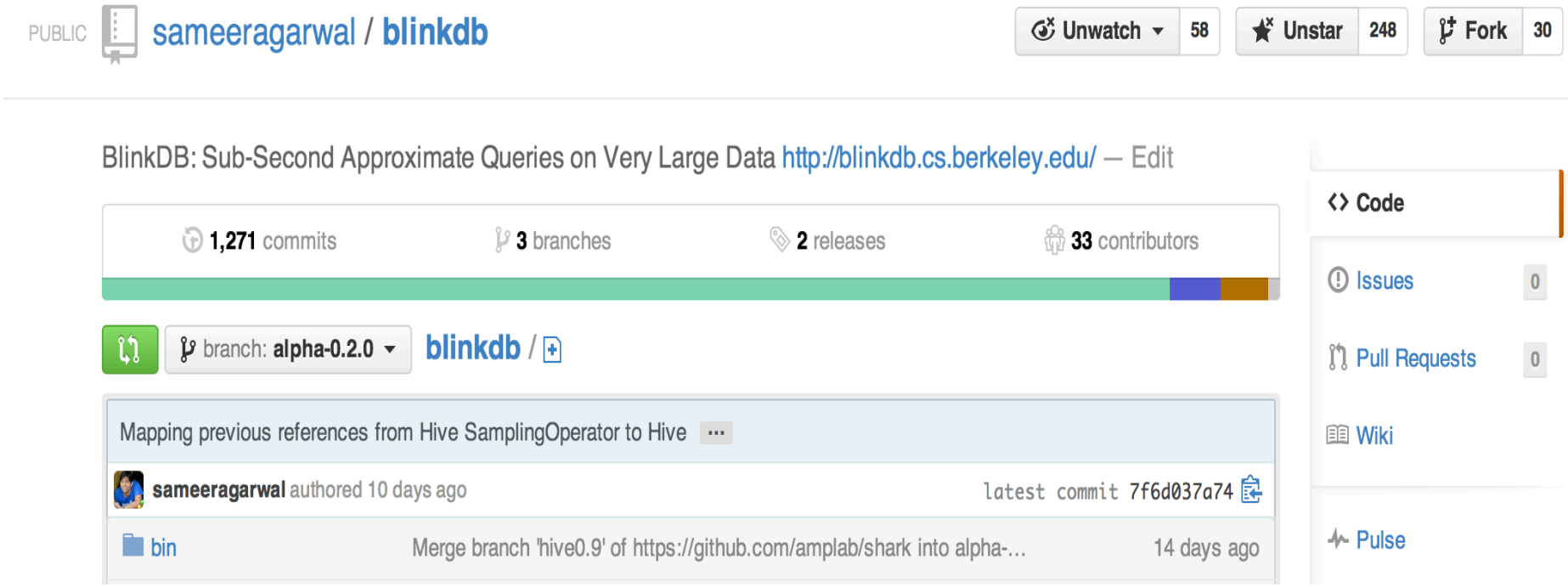
- » Specify how many times each row has been selected
- » Query generate results for each resamples in one pass

Query + Bootstrap + Diagnosis
(with Filter Pushdown and Single Pass)



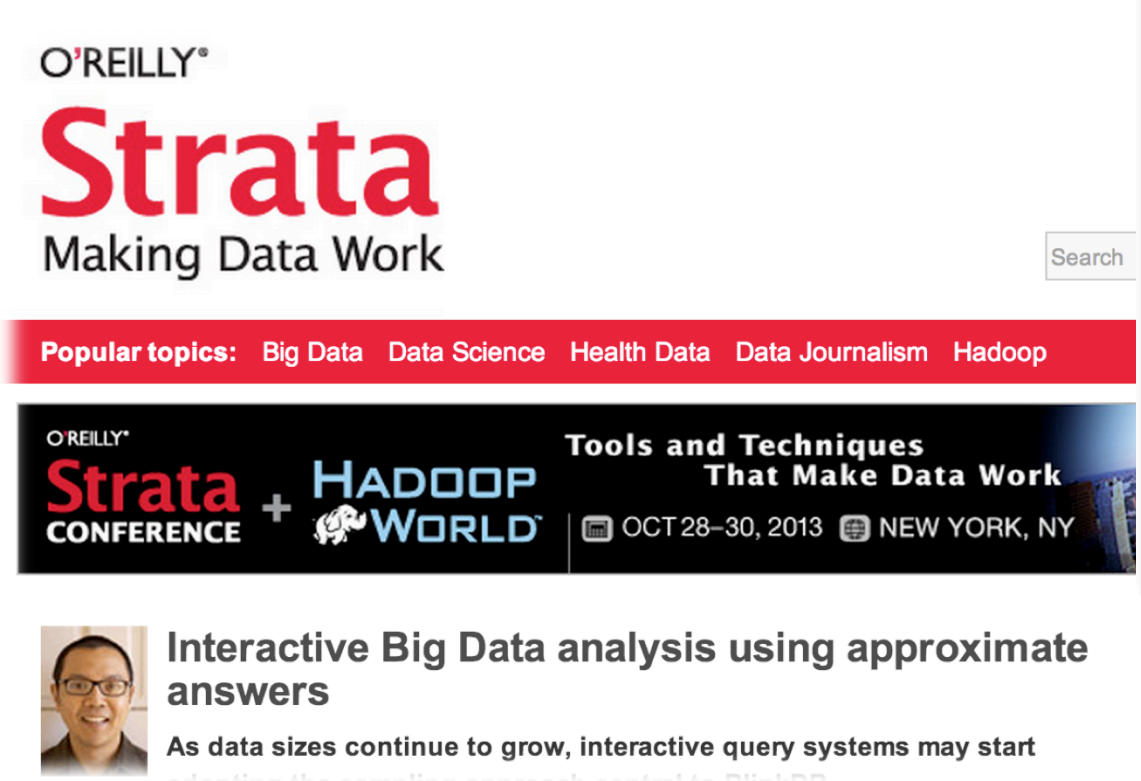
Open Source

BlinkDB is open-sourced and released at http://blinkdb.org



Open Source

Used regularly by 100+ engineers at Facebook Inc.



Lesson Learned

Focus on novel usage scenarios

Be paranoid about simplicity

- » Very hard to build real complex systems in academia

Basic functionality first, performance opt. next

Example: Mesos

Focus on novel usage scenarios

- » Let multiple frameworks share same cluster

Be paranoid about simplicity

- » Just enforce allocation policy across frameworks, e.g., fair sharing
- » Let frameworks decide which slots they accept, which tasks to run, and when to run them
- » First release: 10K code

Basic functionality first, performance opt. next

- » First, support arbitrary scheduling, but inneficient
- » Latter added filters to improve performance

Example: Spark

Focus on novel usage scenarios

- » Interactive queries and iterative (ML) algorithms

Be paranoid about simplicity

- » Immutable data; avoid complex consistency protocols
- » First release: 2K code

Basic functionality first, performance opt. next

- » First, no automatic check-pointing
- » Latter to add automatic checkpoint

Example: BlinkDB

Focus on novel usage scenarios

- » Approximate computations; error diagnosis

Be paranoid about simplicity

- » Use bootstrap as generic technique; no support for close formula

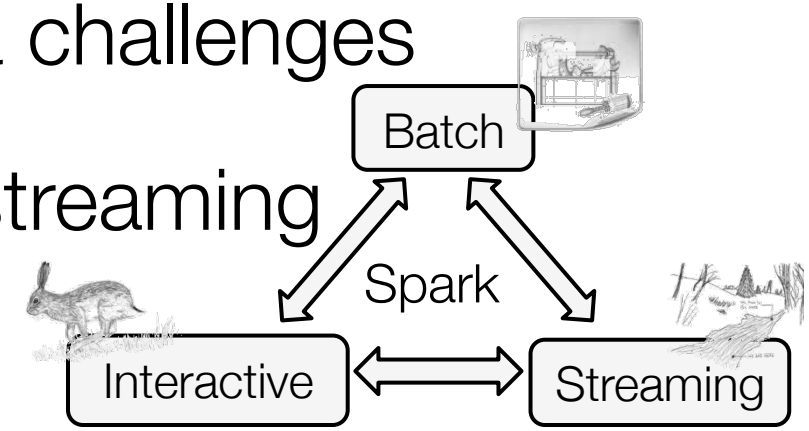
Basic functionality first, performance opt. next

- » First, straightforward error estimation, very expensive
- » Latter, optimizations to reduce overhead
- » First, manual sample generation
- » Later, automatic sample generation (to do)

Summary

BDAS: address next Big Data challenges

Unify batch, interactive, and streaming



Enable users to trade between

- Response time, accuracy, and cost

Explosive adoption

- 30+ companies, 180+ individual contributors (Spark)
- Spark platform included in all major Hadoop distros
- Enterprise grade support and professional services for both Mesos and Spark platform