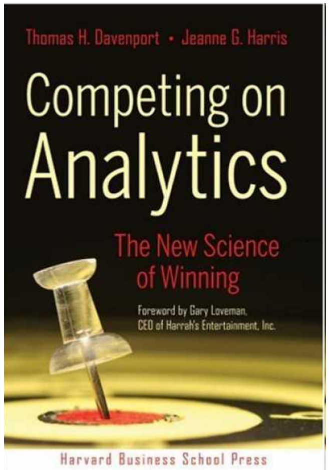
Low-Latency Analytics on Massive Data



Ion Stoica

(and many, many others)

UC Berkeley

amplab UC BERKELEY

Intel Science & Technology Center for Cloud Computing

What is Big Data used For?

Reports, e.g.,

» Track business processes, transactions

Diagnosis, e.g.,

» Why is user engagement dropping?
» Why is the system slow?
» Is this spam?

Decisions, e.g.,

» Personalized treatment
» Decide what feature to add to a product
» Decide what ad to show

What is Big Data used For?

Reports, e.g.,

» Track business processes, transactions

Diagnosis, e.g.,

» Why is user engagement dropping?

Data is as useful as the decisions it enables

Decisions, e.g.,

» Personalized treatment

» Decide what feature to add to a product

» Decide what ad to show

Data Processing Goals

Low latency on historical data

» E.g., diagnosis, root cause analysis

Low latency on live data (streaming)

» E.g., real-time dashboard

Sophisticated data processing: "better" decisions

» E.g., anomaly detection, trend analysis

Data Processing Goals

Low latency on historical data

» E.g., diagnosis, root cause analysis

Goal: Low latency computations on **massive** datasets for both **historical** and **live** data

Sophisticated data processing: "better" decisions

» E.g., anomaly detection, trend analysis

Today's Open Analytics Stack…

..mostly focused on large on-disk datasets

» Sophisticated processing on massive data, but slow

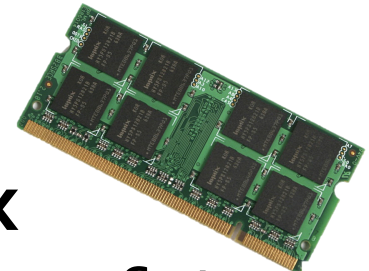
Application

Data Processing

Storage

Infrastructure

Key Ideas



Add RAM (and SSDs) to the mix

» Surprising # of real-world working sets fit in memory

- Inputs of 90% of MapReduce jobs at Facebook and Yahoo! can fit in cluster memory (85% at Microsoft)

» Provide interactive queries and data streaming

Allow users to **trade** between query's (computation's)

» **Response time**

» **Accuracy**

» **Cost**

Our Stack

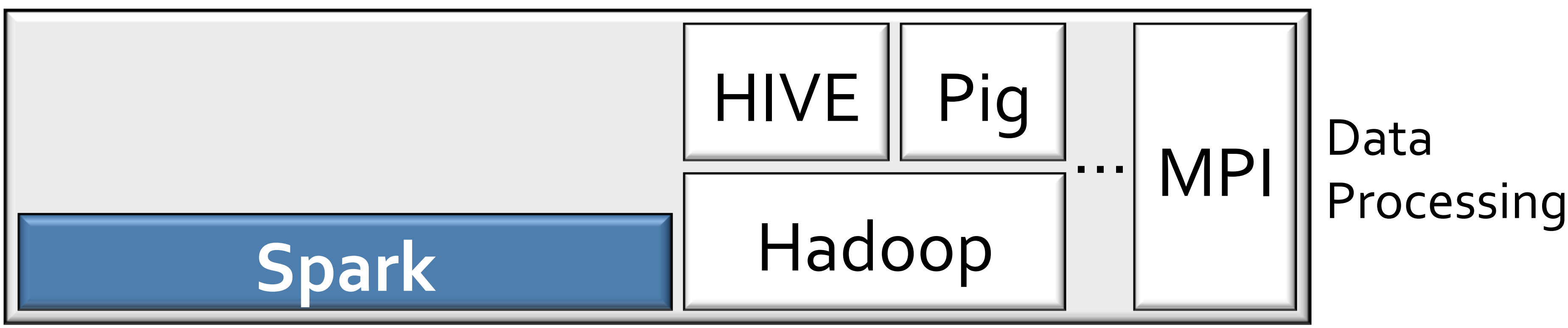
Application	New applications: Cancer Genomics, Carat
Data Processing	In-memory processing & allow users to trade between resp. time and quality
Data Management	Data sharing across frameworks, new data abstractions (includes storage)
Resource Management	Share infrastructure across frameworks (i.e., multi-programing for datacenters)

Frameworks: Spark

In-memory framework for

» **low-latency** computations on **historical data**

» iterative computations



Spark

SCALA interface

x10 – x100 faster than Hadoop

Challenge: Need a distributed memory abstraction that is both **fault-tolerant** and **efficient**!

Possible Solutions

Replicate data in memory

» Slow: network throughput much lower than memory throughput

» Inefficient: use at least twice as much memory

Log the updates

» Inefficient: logs for data intensive applications typically very large → writing on the disk slow

Our Solution

Resilient Distributed Data Sets (RDD)

» Partitioned collection of records

» Immutable

» Can be created only through deterministic operations from other RDDs

Handle of each RDD stores its **lineage**:

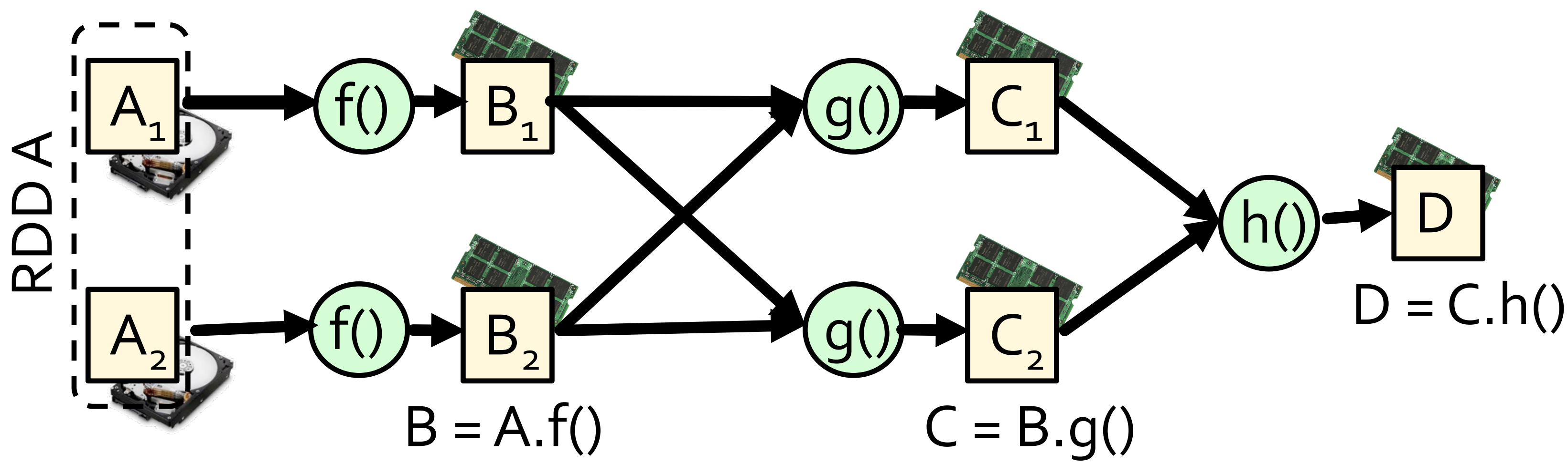
» Lineage: sequence of operations that created the RDD

Recovery: use lineage information to rebuild RDD

RDD Example

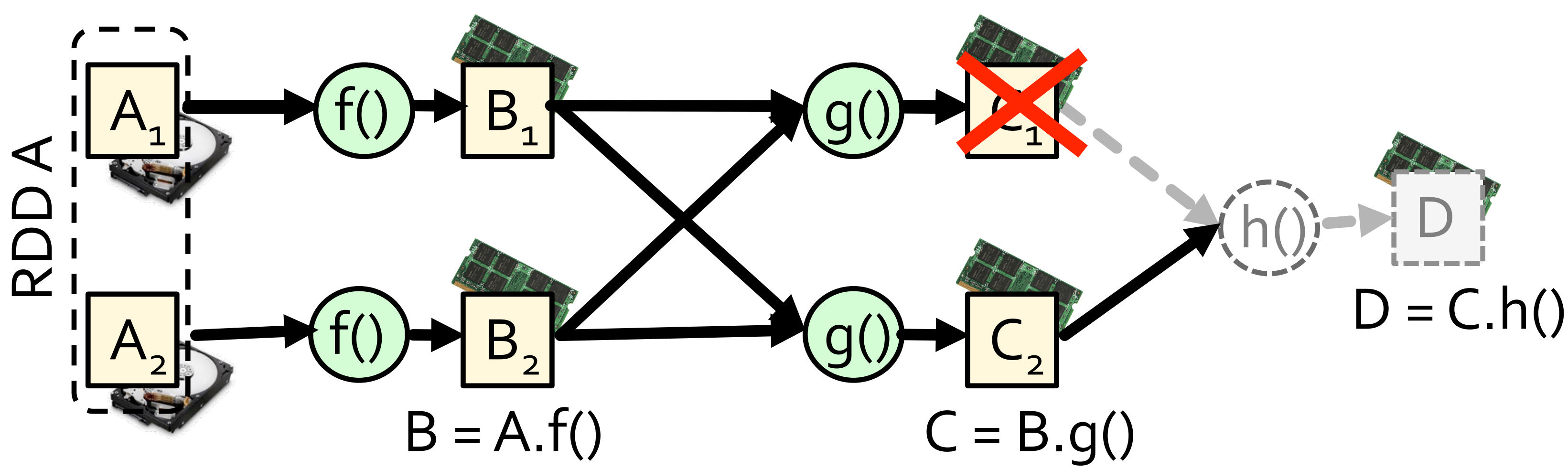
Two-partition RDD $A=\{A_1, A_2\}$ stored on disk

1) Read and cache after applying f() → RDD B
2) Shuffle, and apply g() → RDD C
3) Aggregate using h() → D



RDD Example

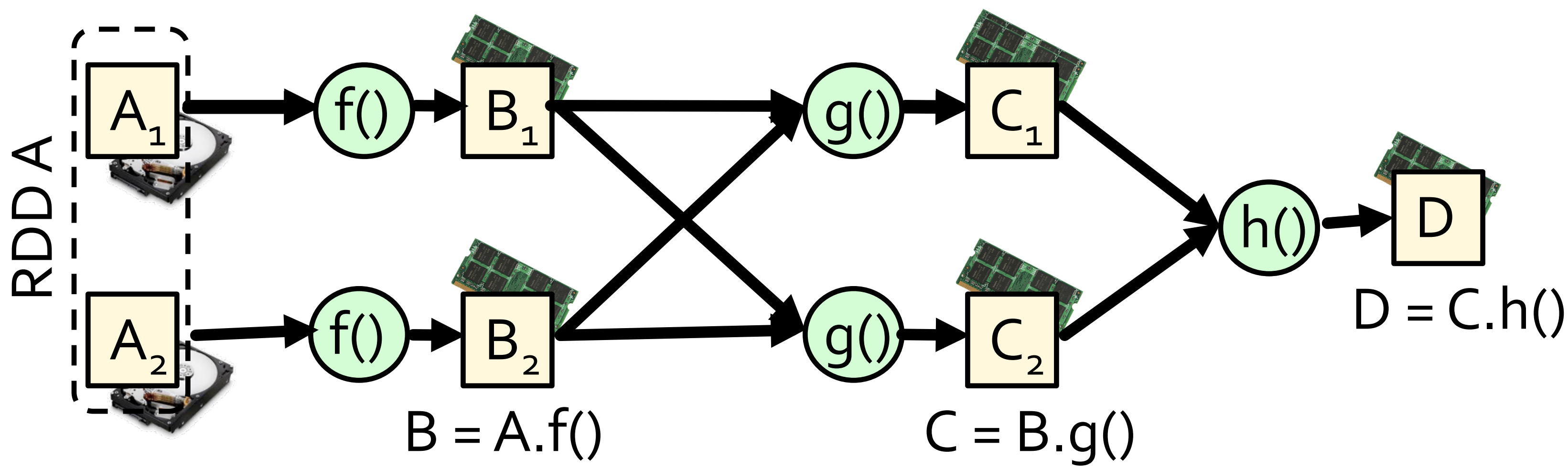
C_1 lost due to node failure before h() is computed



RDD Example

C_1 lost due to node failure before h() is computed

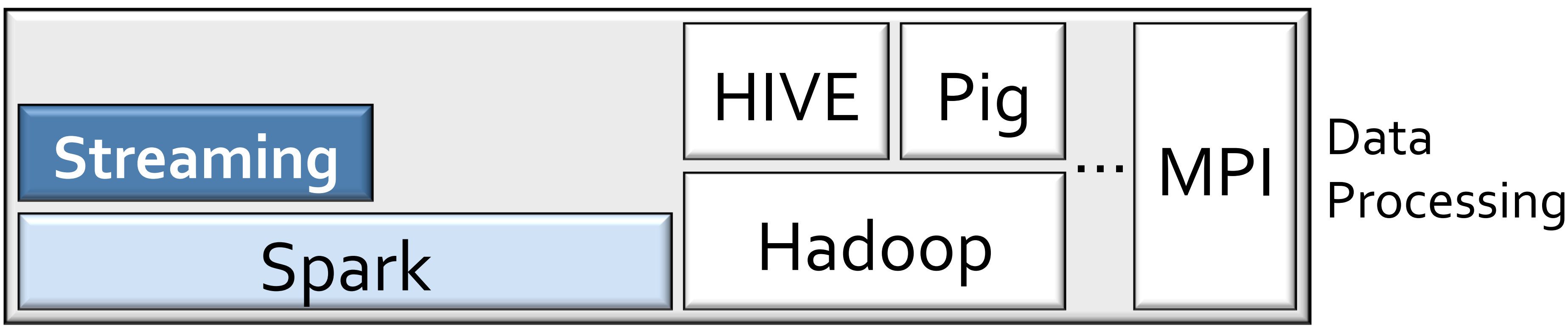
Reconstruct C_1, eventually, on a different node



Frameworks: Streaming

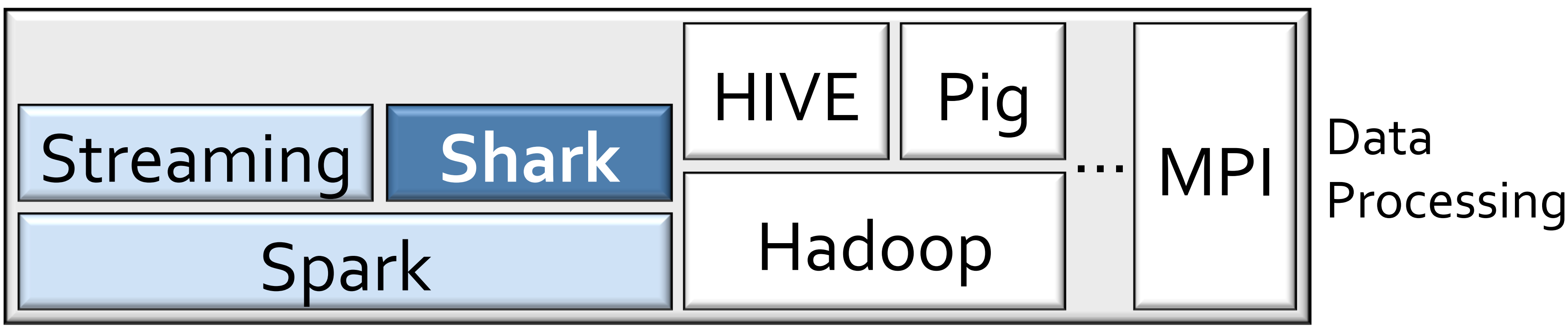
Add streaming functionality to Spark

» **Low-latency** computations on **live data**



Frameworks: Shark

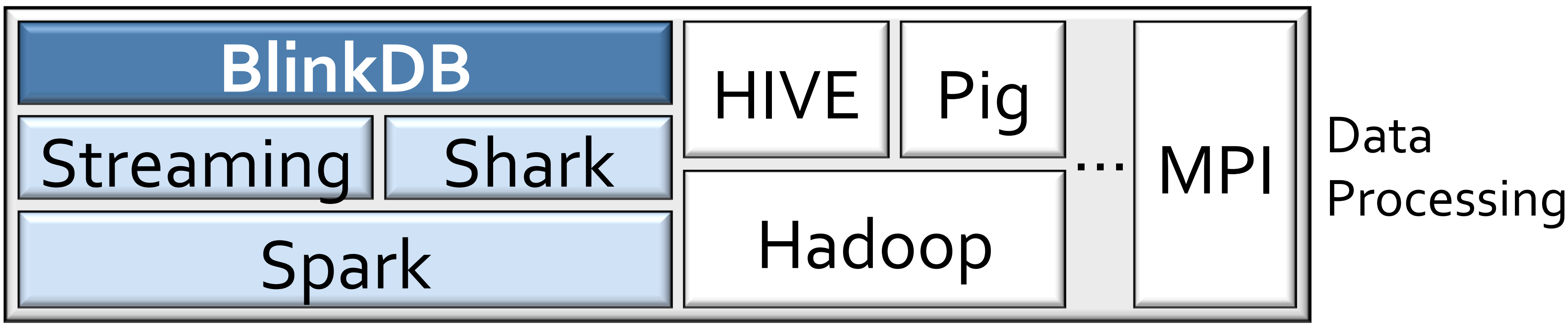
HIVE over Spark: Interactive SQL-like queries for data fitting into memory



Frameworks: BlinkDB

Allow users to **trade** between computation's

- » **accuracy**
- » **time**
- » **cost**



Why BlinkDB?

Even if all data in memory, query may take 10's sec

» Just scanning 200-300GB RAM may take 10 sec

Too slow for…

» real-time (e.g., sub-second) decisions, and…

» … even for interactive queries

Exact results not always necessary, e.g.,

» Does blue background increase user engagement?

» Has the service slowed down?

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

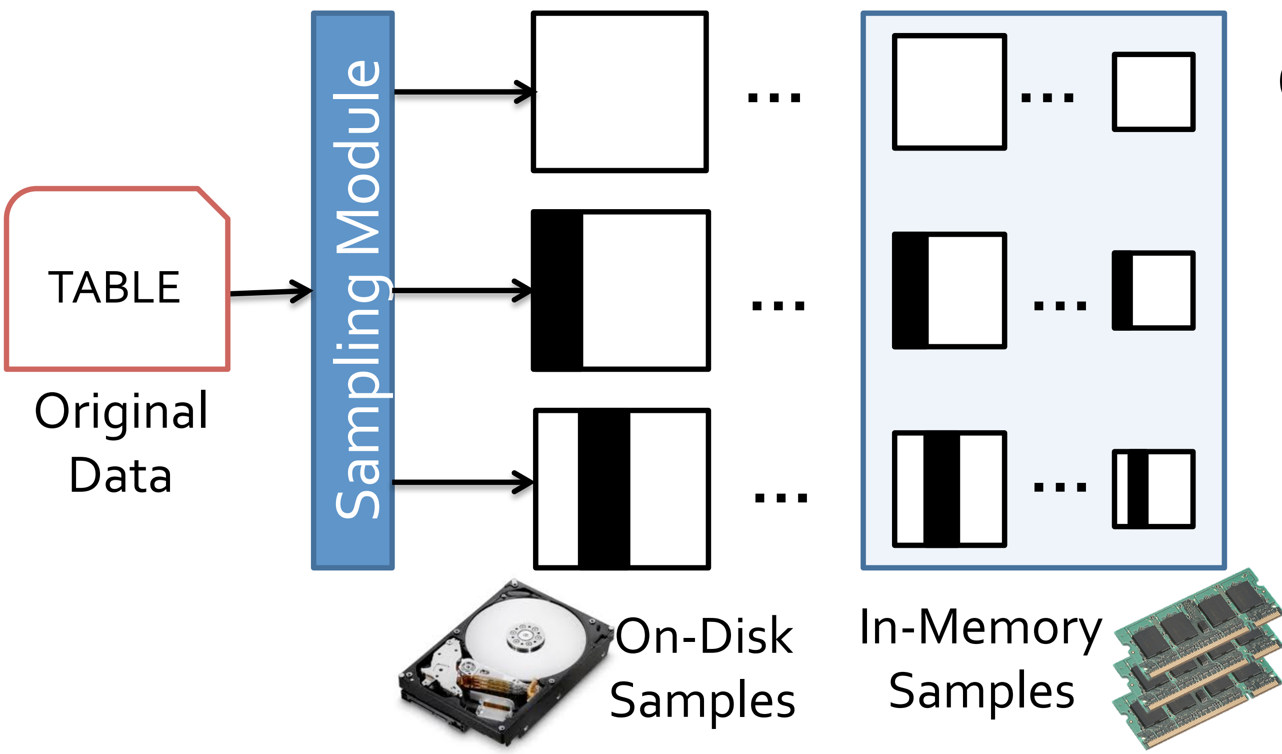
SELECT avg(sessionTime)

FROM Table

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

ERROR 0.1 **CONFIDENCE** 95.0%

System Architecture



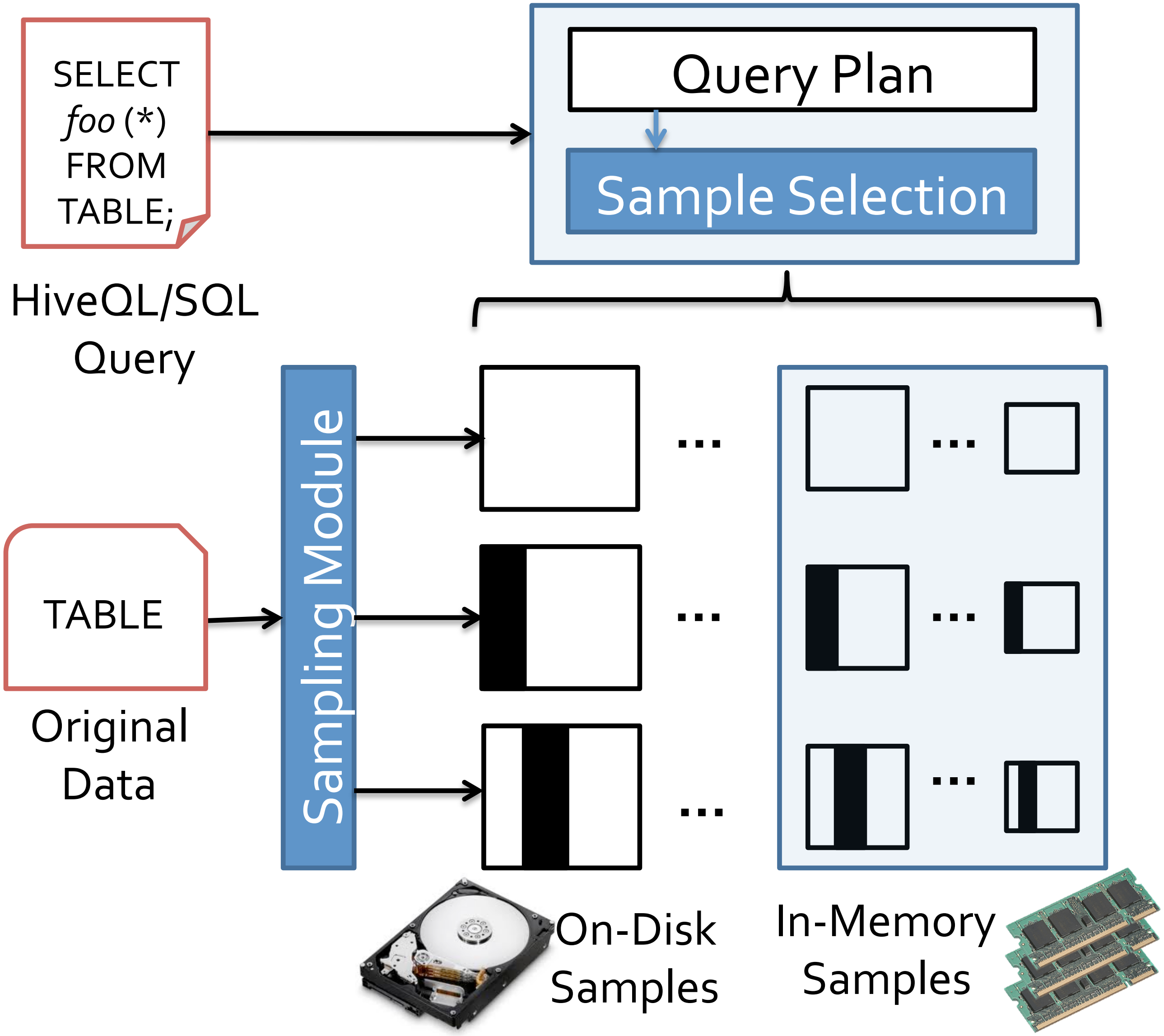
Offline-sampling:

» **Uniform random**

» **Stratified** on diff. set of columns

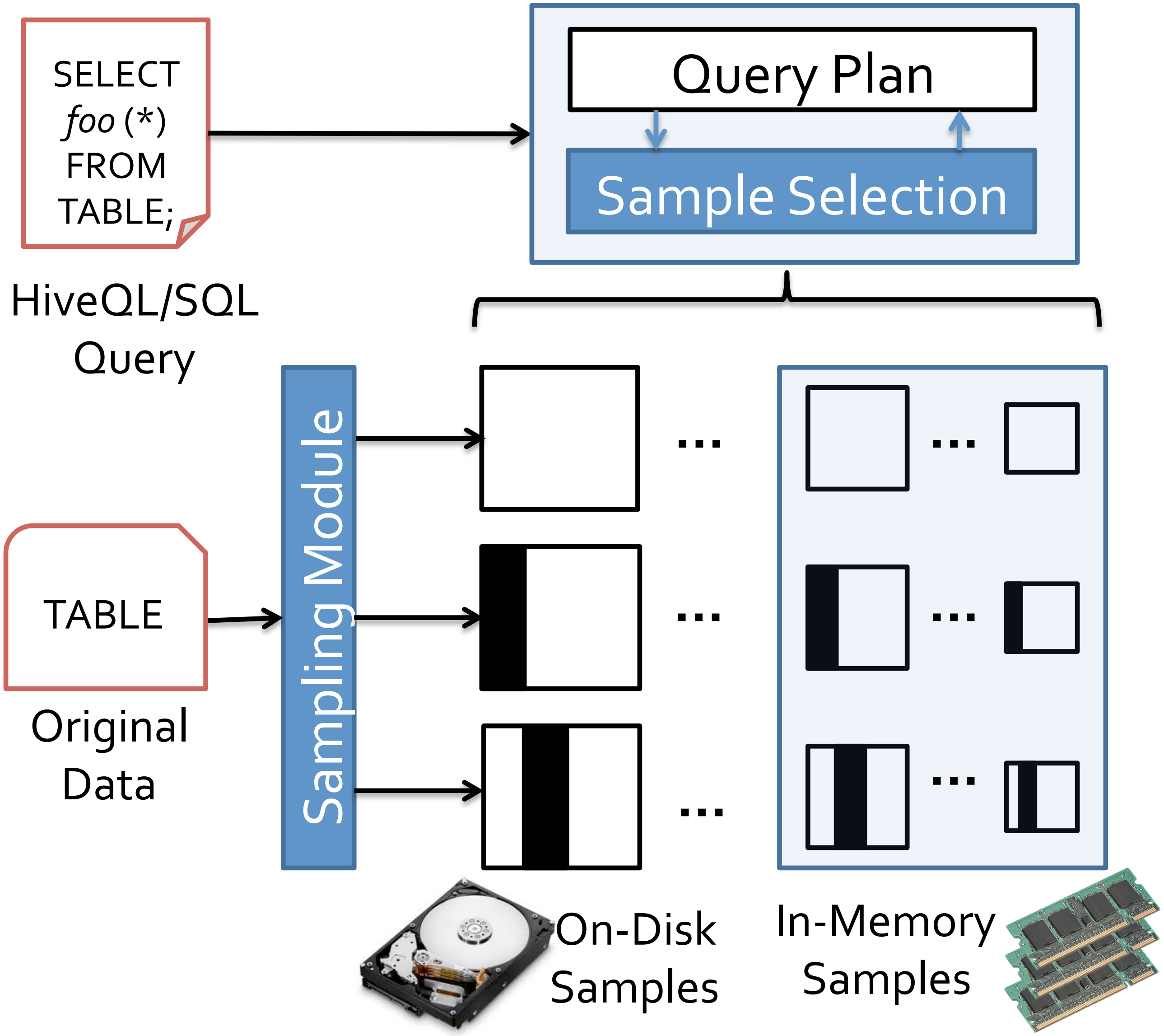
» Diff. **granularities**

System Architecture



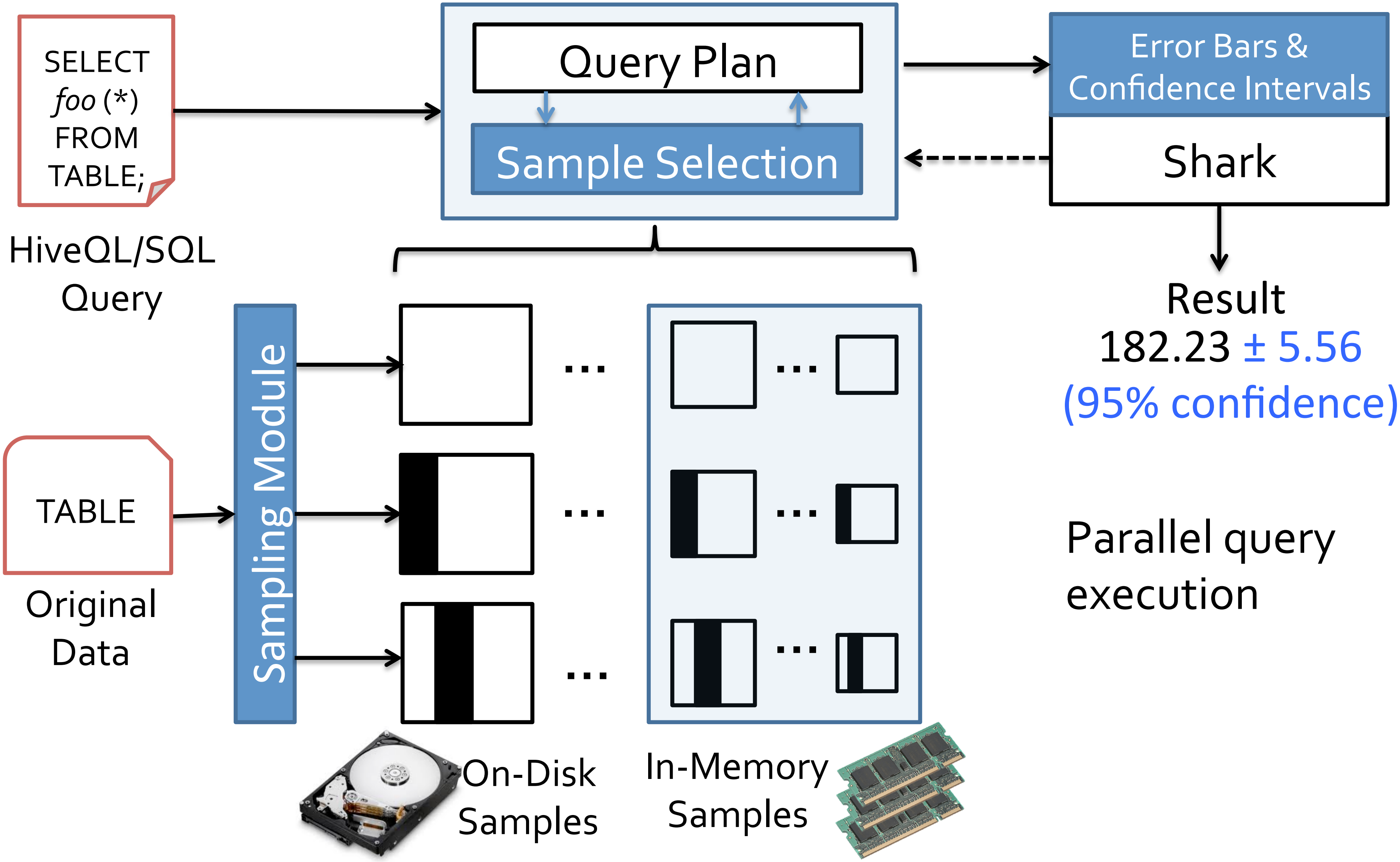
Build **Error-Latency Profile (ELP)**: Predict **time** and **error** of the query running on different samples

System Architecture



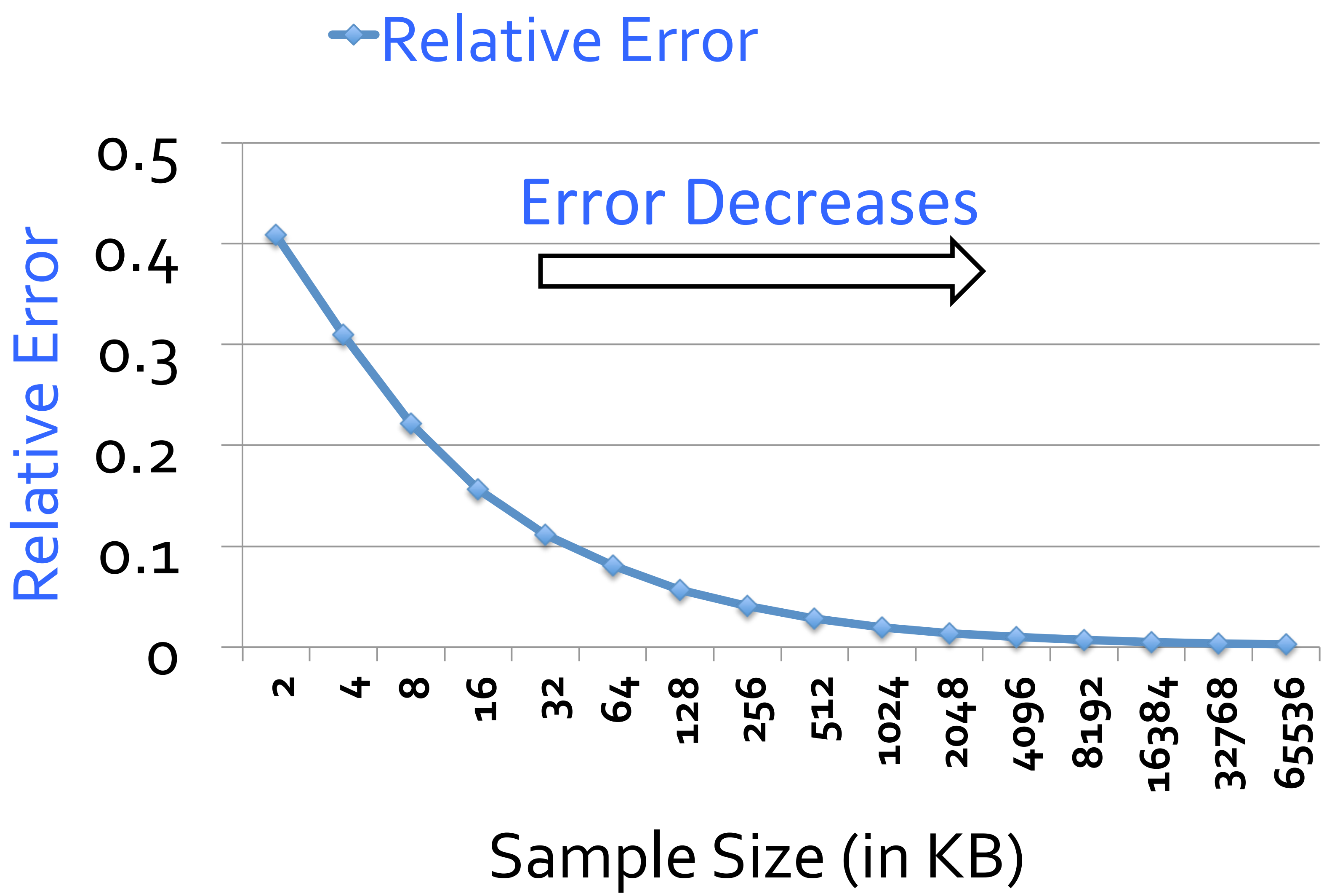
Online sample selection: pick best sample(s) based on **ELP**

System Architecture

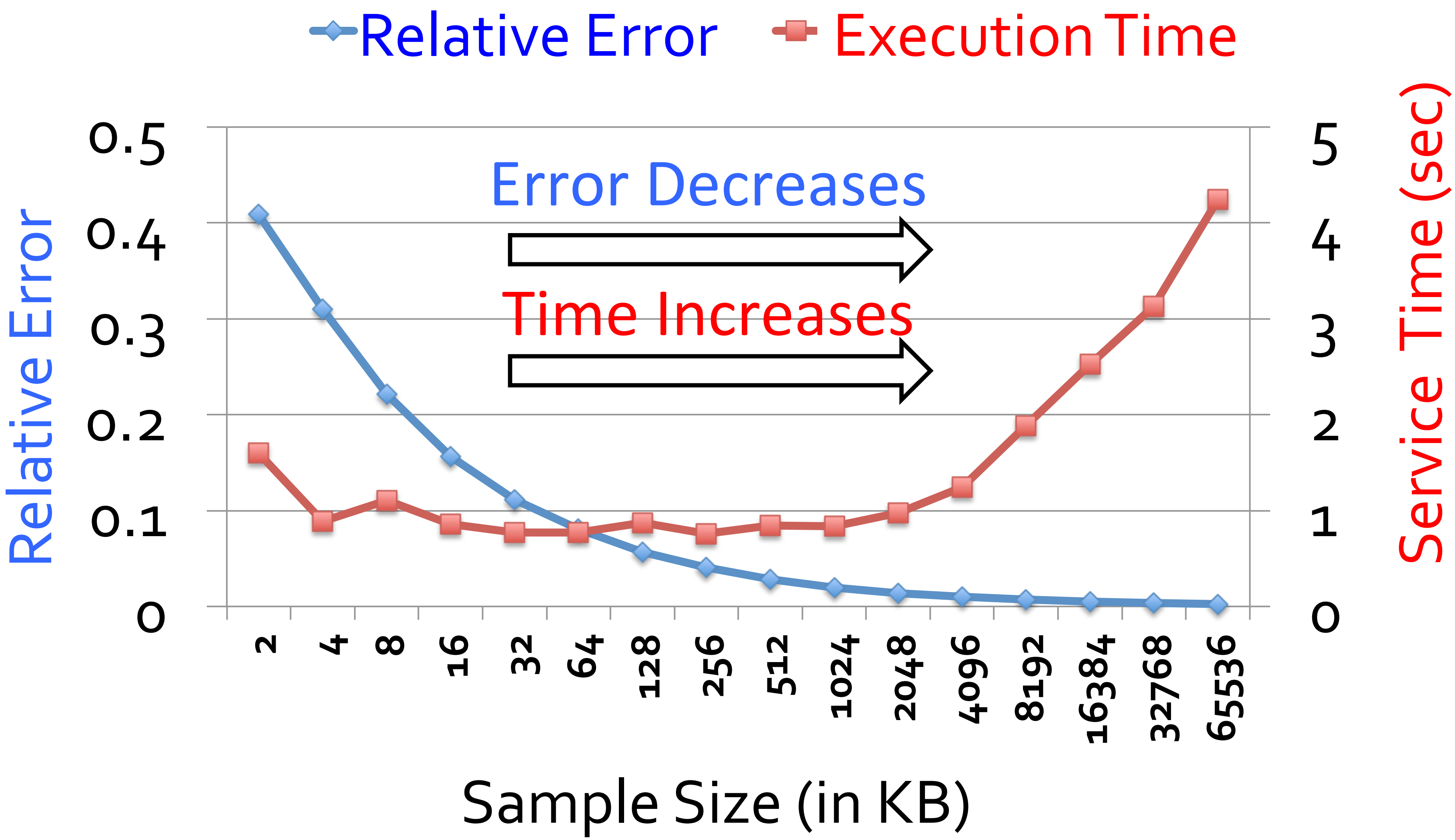


Parallel query
execution

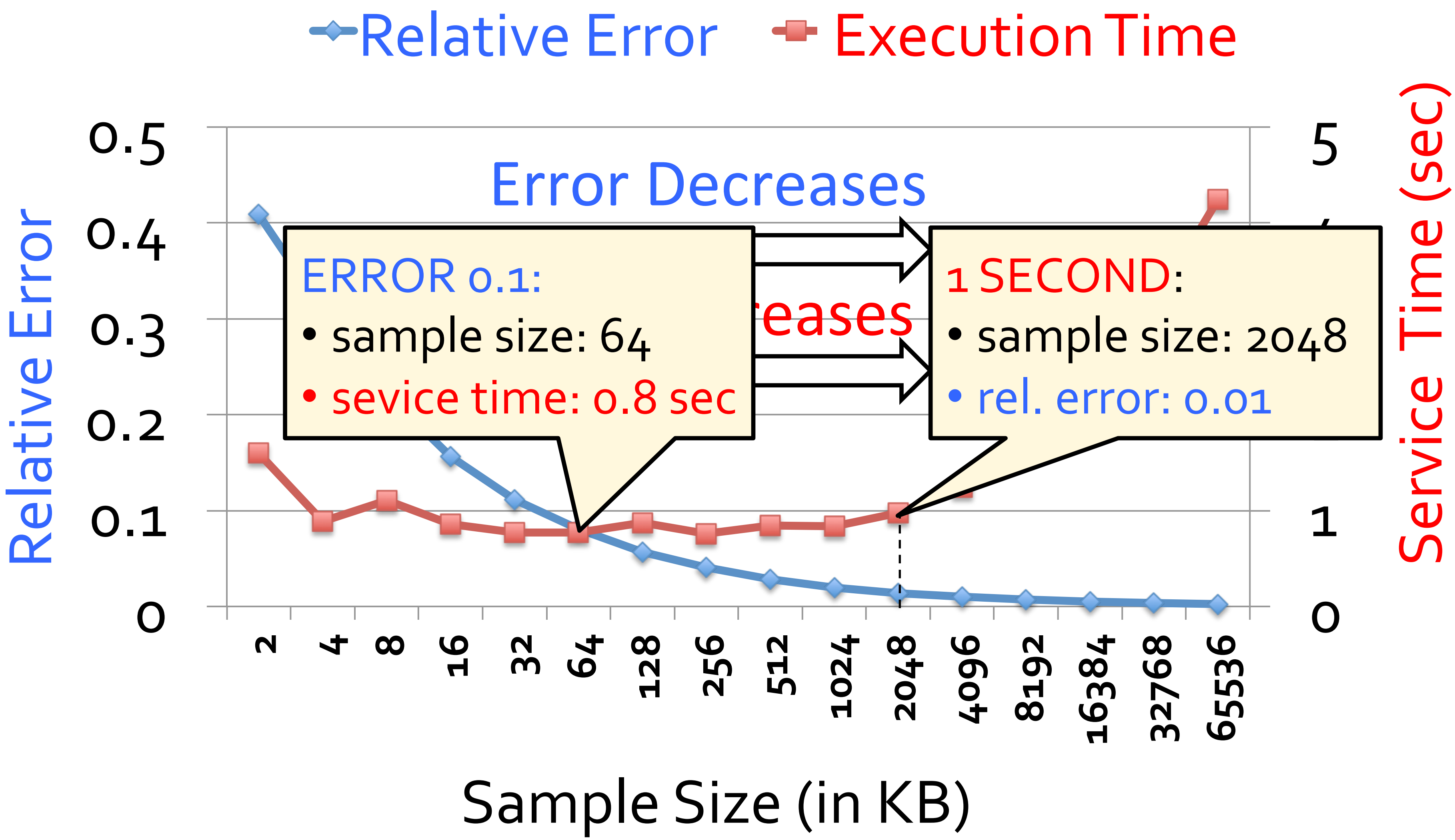
Error-Latency Profile (ELP)



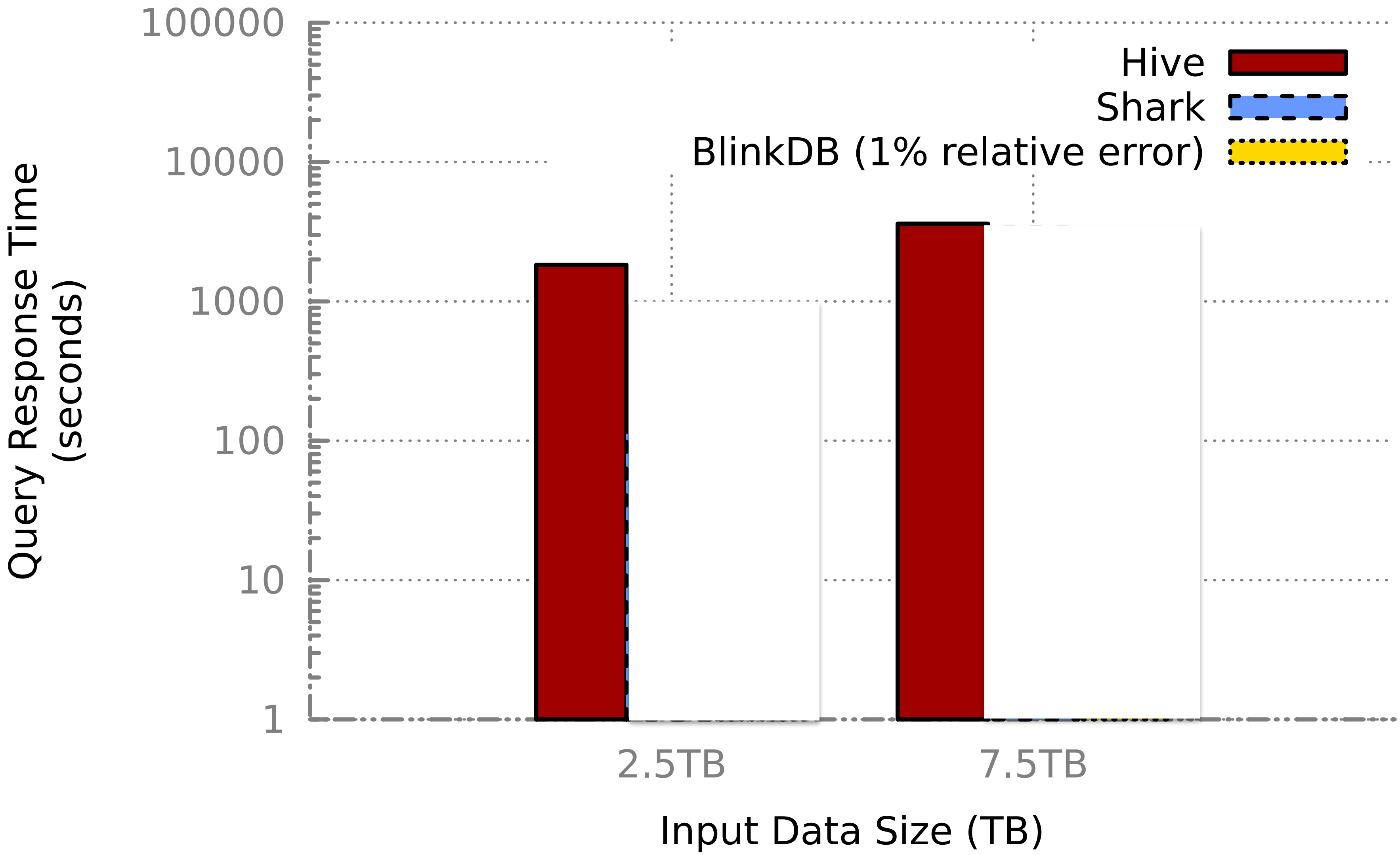
Error-Latency Profile (ELP)



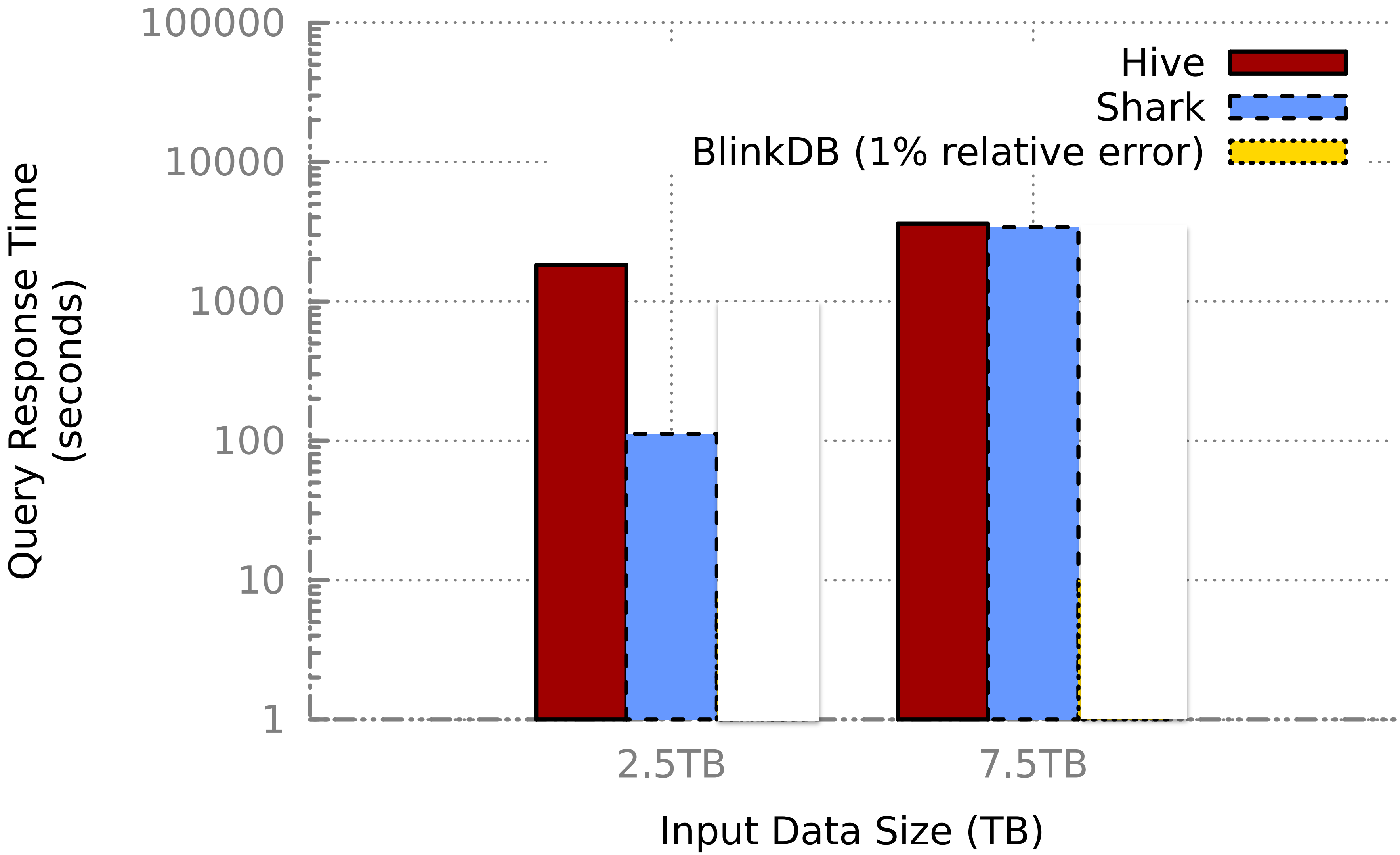
Error-Latency Profile (ELP)



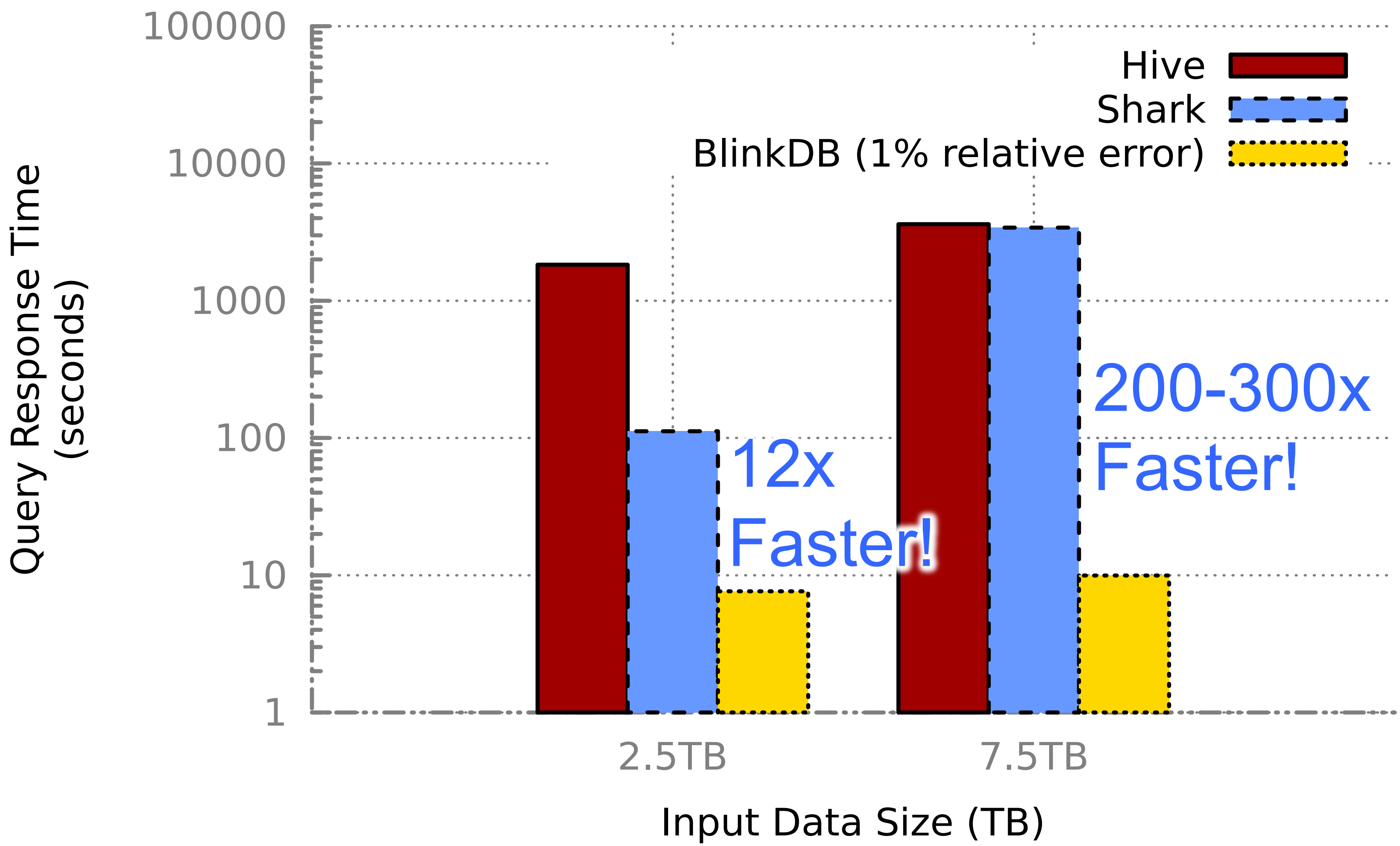
BlinkDB: Evaluation



BlinkDB: Evaluation



BlinkDB: Evaluation



BlinkDB Challenges

Which set of samples to build given a storage budget?

How do we accurately estimate the service time?

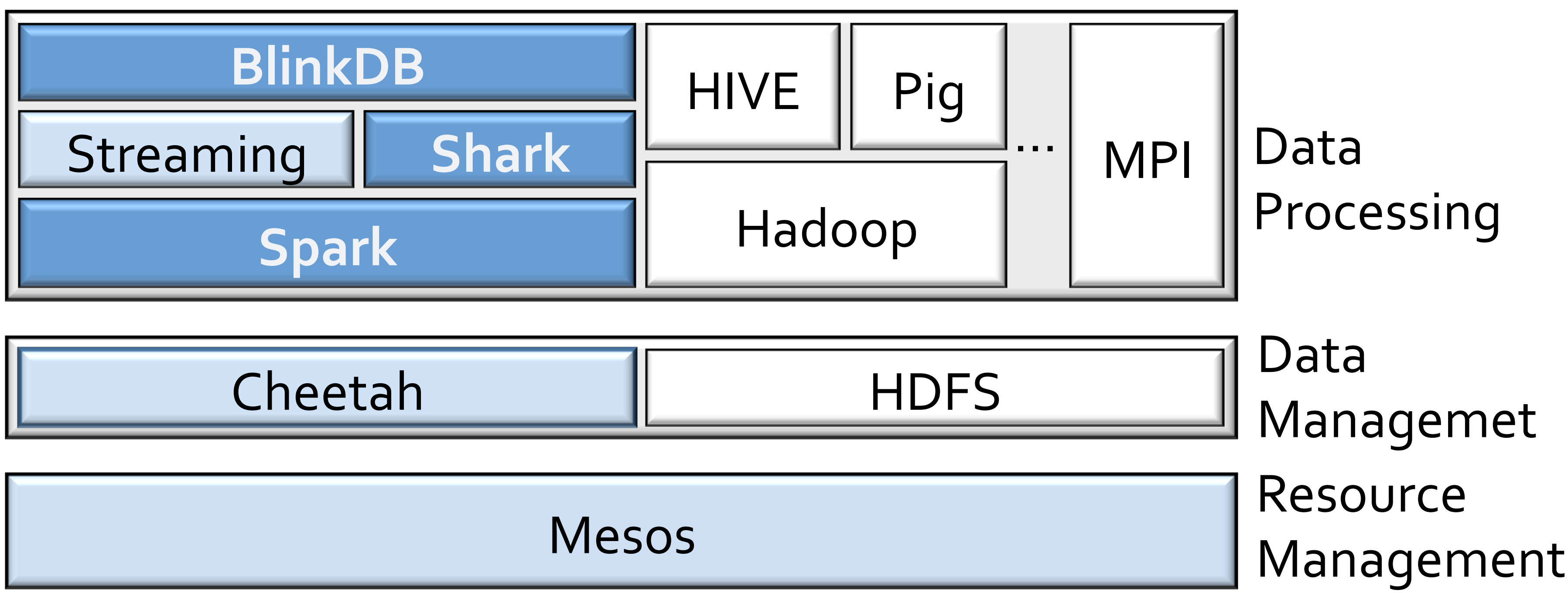
How do we accurately estimate the error?

- » What about user defined functions (UDFs)?

Summary

Build full Data Analytics Stack compatible with existing open source stack

Low latency computations on massive historical and live data



Status

Several components have already been released

» **Mesos:** deployed on +2,500 servers at Twitter

» **Spark:** used by dozen companies

» **Shark:** just released in October

» **Carat:** ~400K downloads on AppStore

Future work: highly scalable Machine Learning algorithms

https://amplab.cs.berkeley.edu

Students Involved in Projects

Spark: Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley

Shark: Reynold Shin, Matei Zaharia, Josh Rosen

BlinkDB: Sameer Agrawal, Aurojit Panda, Henry Milner, Barzan Mozafari (PostDoc, MIT)

Thank you!