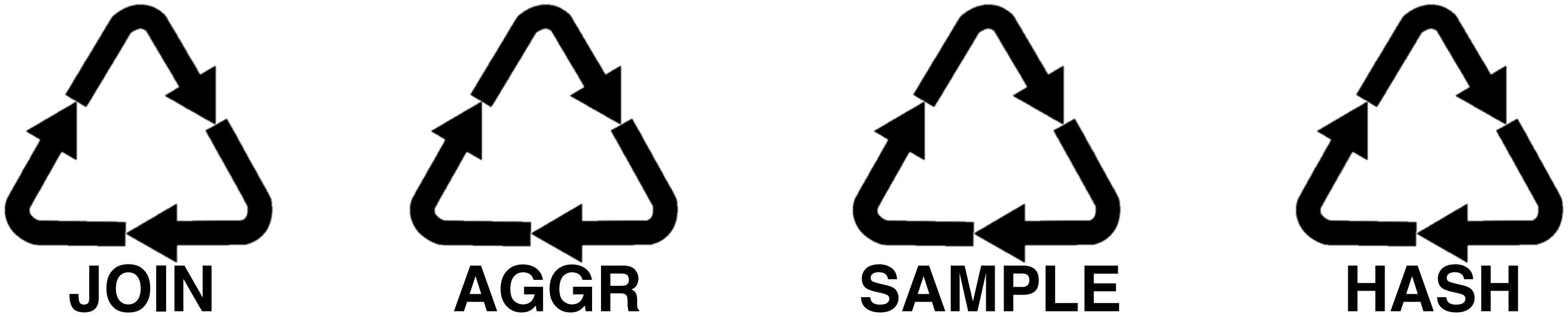


Taster: *Self-Tuning*, *Elastic* and *Online* Approximate Query Processing

Matthaios Olma Odysseas Papapetrou Raja Appuswamy Anastasia Ailamaki

AiAS Data-Intensive Applications and Systems · EURECOM Sophia Antipolis · TU/e · EPFL

Challenges of interactive data exploration

Exploratory Applications
- Dynamic & data-driven

Scientific exploration

"Internet of Things" analytics

Interactive response time

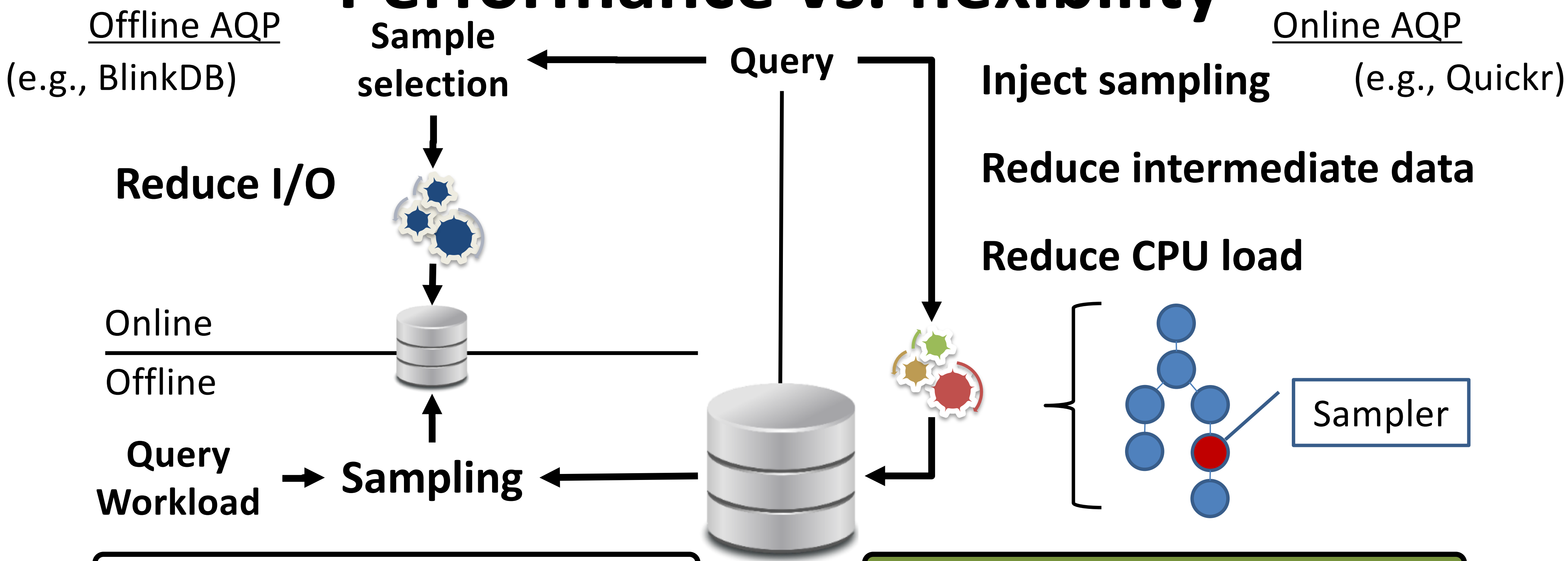
Reduce result precision – use AQP

Instant access to data

Building data summaries is expensive

Enable AQP with minimal pre-processing

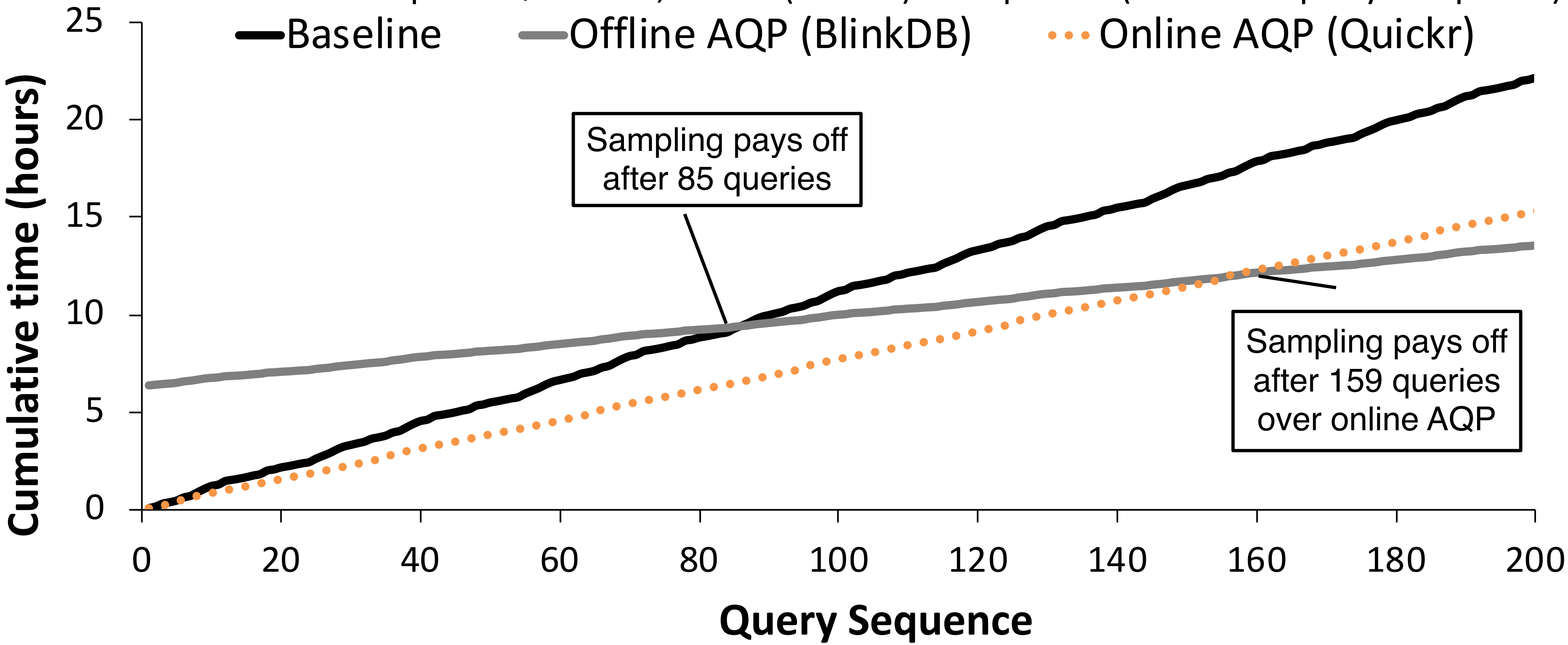
Performance vs. flexibility



Pre-sampling	No Preprocessing
Workload knowledge required	No workload knowledge
0.5-2x storage overhead	No storage overhead
~10x performance	~2x performance

Reducing pre-processing time

11 node SparkSQL cluster, TPC-H (300GB) 200 queries (18 TPC-H query templates)



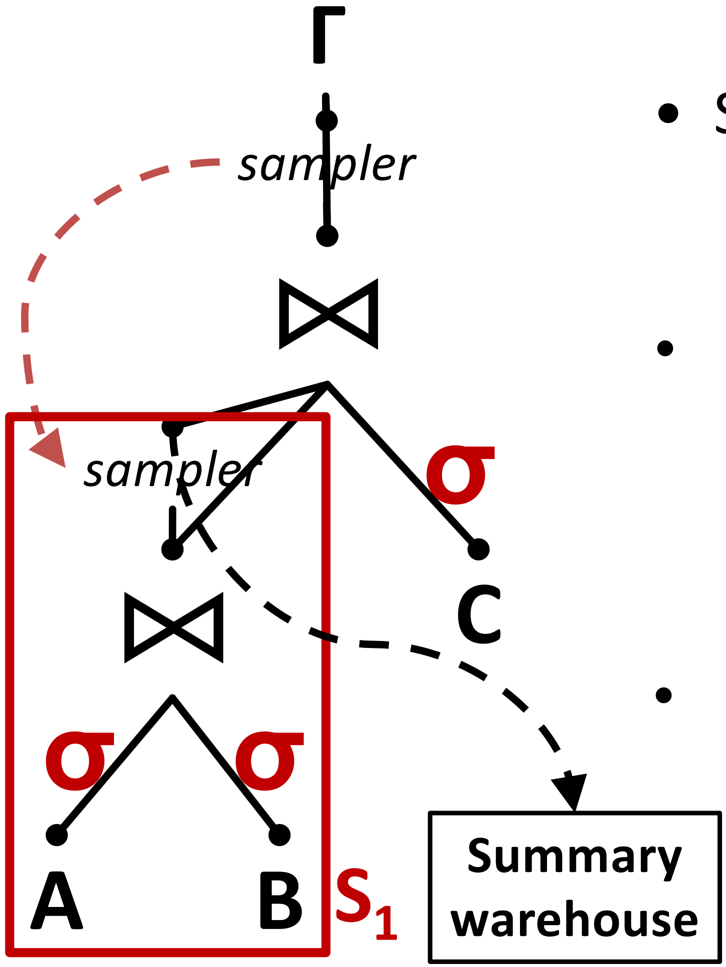
Ideal: No sampling preparation cost & interactive access

Enhancing Online Approx. Query Processing

- Reduce the amount of data accessed
  - Materialize and Re-use intermediate generated summaries
- Adapt materialized summaries to workload and storage budget
- Use a variety of summaries other than samples

What to materialize | **If/When to materialize** | **If/When to evict**

Materialize and re-use synopses

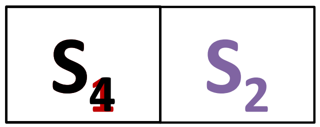


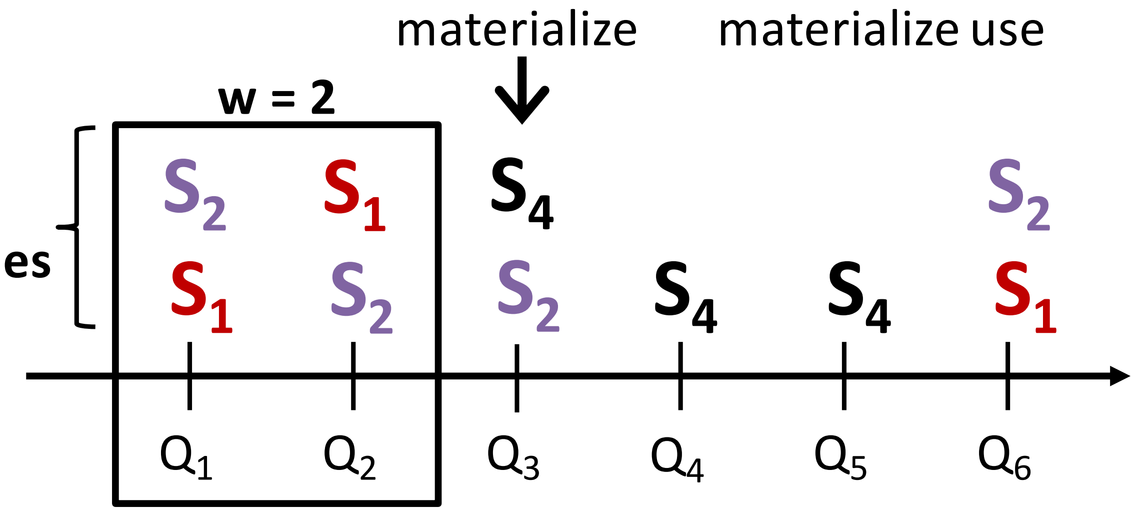
- Store all subplans and statistics in hitmap
  - Update when subplans re-appear
- Calculate prospective gains (cost:benefit)
  - Performance gains over future workload
  - Storage cost
  - Maximize benefit – Knapsack constraint problem
- Decide to materialize
  - Inject materializer operator
  - Store intermediate result in-memory and flush offline

Adapting materialized summaries

- Window-based prediction

Summary warehouse





- Ideal window size depends on: user, task, data
  - Keep statistics for (1-a)w, w, (1+a)w
  - Adapt window size based on quality of predictions

Abide to storage requirements despite workload shifts

Online tuning of window size improves forecast efficiency

Combining different data summaries

Sampling

All queries on subset of data

- Keep schema of original table
- Precision depends on query
- Uniform/Stratified sampling

Answer large subset of queries

Large size ~ 10% of input

I/O cost depending on size

Sketches

Some queries on all data

- Count/Sum/Avg
- Aggregations
- Single grouping attribute

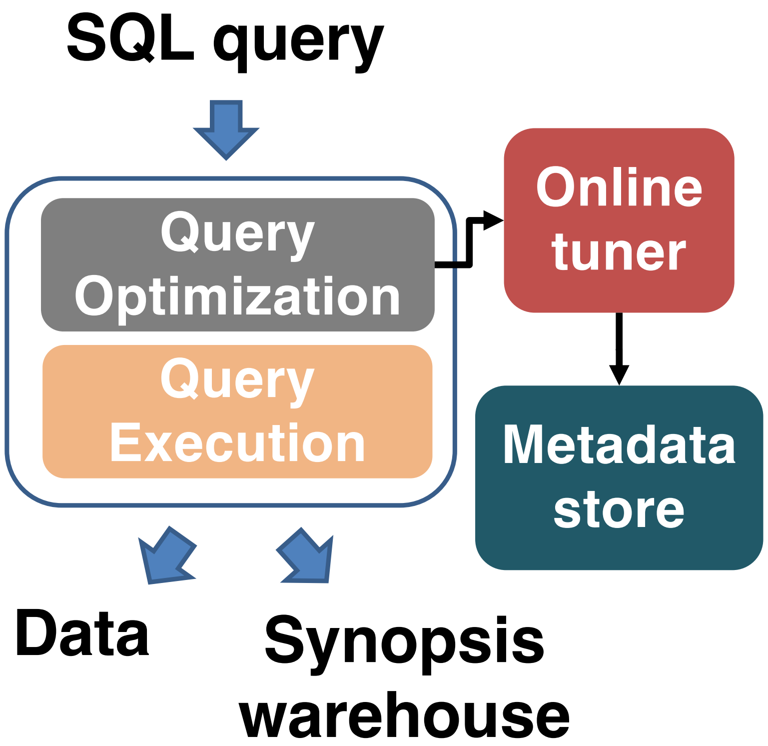
Answer specific queries

Compact ~KB

Constant access time

Utilize each summary where useful

Taster Architecture



- Inject approximation operators into plans
- Re-use existing materialized synopses
- Choose which synopsis to generate
- Store statistics about the historical plans
- Store the synopsis over HDFS

Experimental Setup

Datasets

- TPC-H: sf300 (300GB), 18 query templates

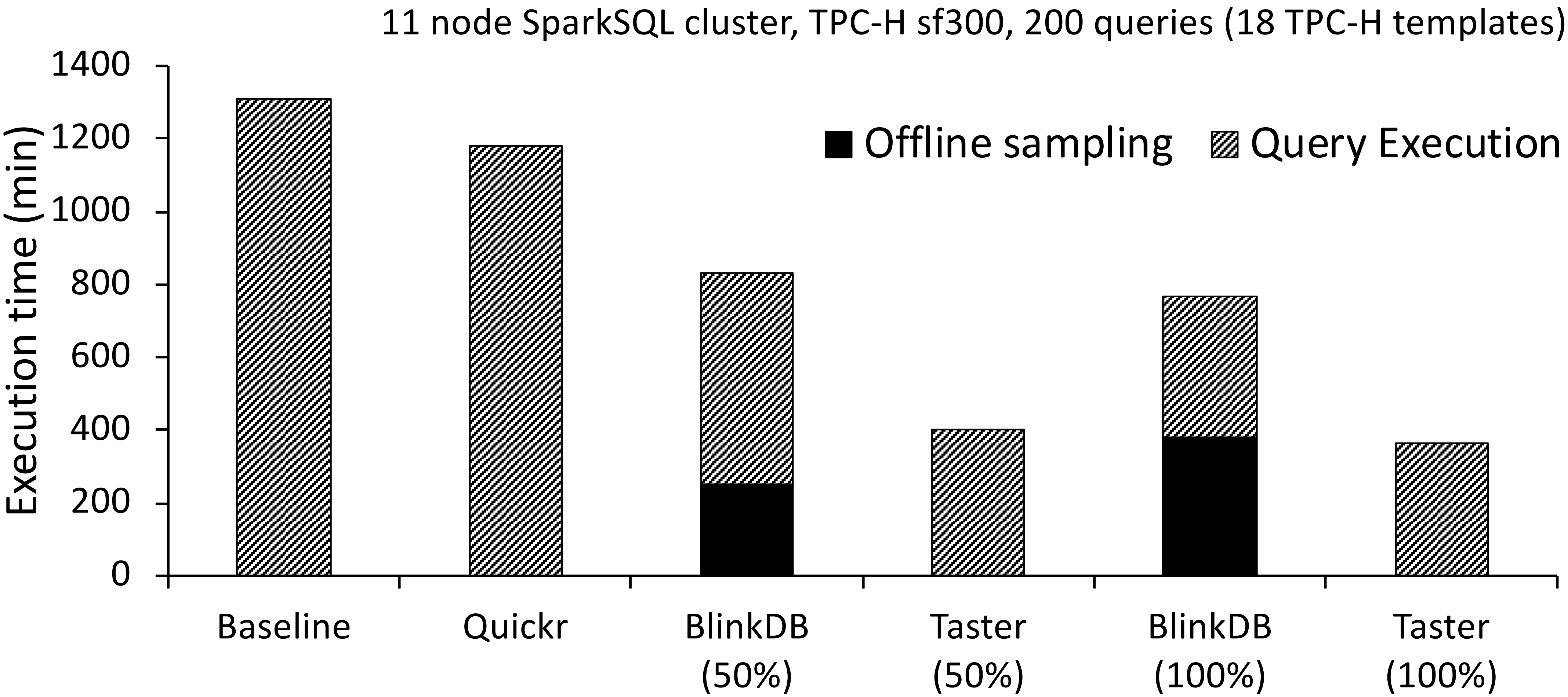
Systems

- SparkSQL (2.1.0)
- BlinkDB, Quickr, Taster over SparkSQL (2.1.0)

Hardware

- 11 nodes x 2 x Intel Xeon X5660 CPU @ 2.80GHz, 48GB RAM, 10GbE (fix for each)

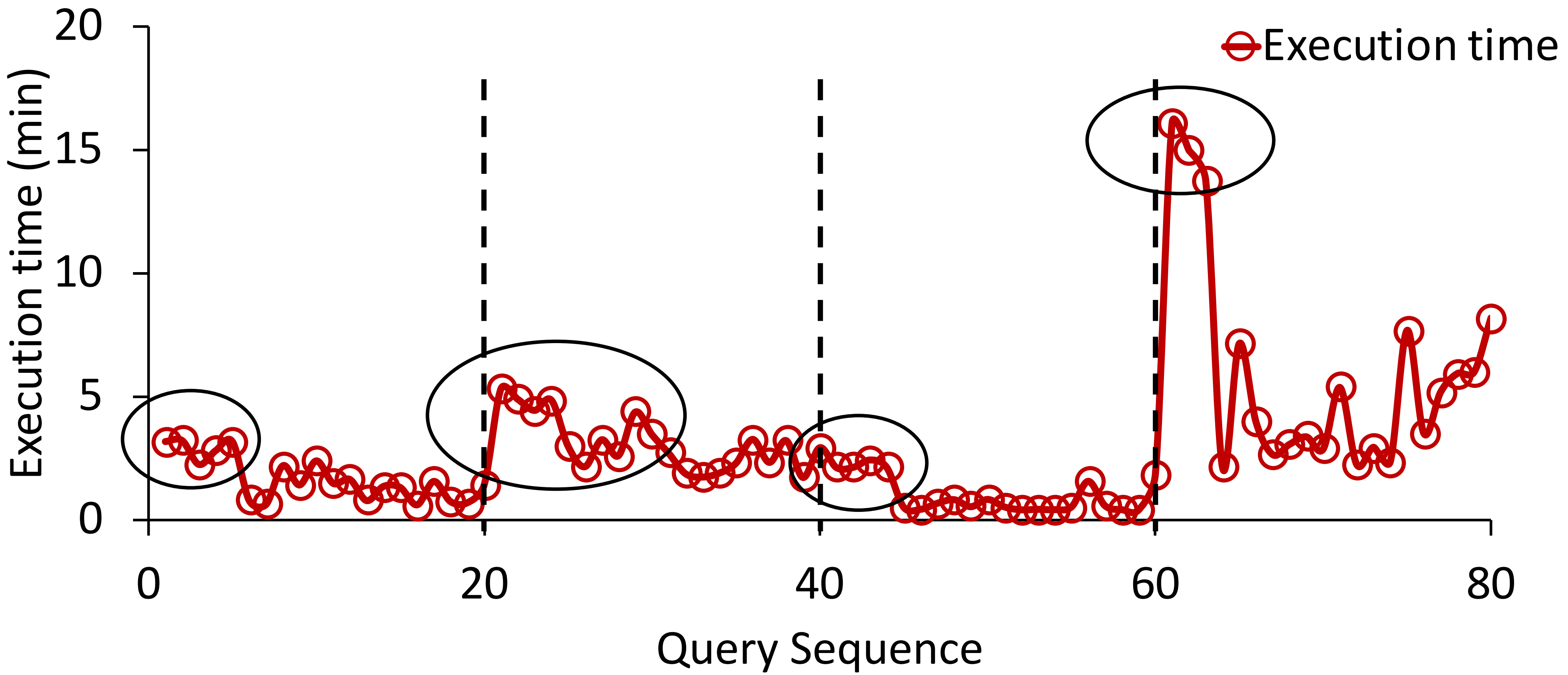
End-to-End execution time



Taster offers comparable execution time to state-of-the-art

Adapting to shifting workload

11 node SparkSQL cluster, TPC-H sf300, 80 queries (18 TPC-H templates)



Taster adapts efficiently to changes in workload

Take home message

- Piggy-back the creation of summaries over the query execution
  - In the context of distributed approximate query processing
- Adapt data summaries to workload shifts and reduce storage budget
- Provide query performance comparable to offline AQP approaches
  - With reduced building and storage cost

Thank you!