# Shark:
# Hive (SQL) on Spark

Reynold Xin

UC Berkeley AMP Camp

Aug 21, 2012

amplab UC BERKELEY

```sql
SELECT page_name, SUM(page_views) views
FROM wikistats GROUP BY page_name
ORDER BY views DESC LIMIT 10;
```

```
Stage 0: Map-Shuffle-Reduce

Mapper(row) {
  fields = row.split("\t")
  emit(fields[0], fields[1]);
}

Reducer(key, values) {
  page_views = 0;
  for (page_views in values) {
    sum += value;
  }
  emit(key, page_views);
}
```

```
Stage 1: Map-Shuffle

Mapper(row) {
  emit(page_views, page_name);
}

... shuffle sorts the data

Stage 2: Local

data = open("stage1.out")
for (i in 0 to 10) {
  print(data.getNext())
}
```

# Outline

Hive and Shark

Data Model

Shark Demo

Beyond SQL Basics

# Apache Hive

Puts structure/schema onto HDFS data

Compiles HiveQL queries into MapReduce jobs

Very popular: 90+% of Facebook Hadoop jobs generated by Hive

Initially developed by Facebook

# OLAP vs OLTP

Hive is NOT for online transaction processing (OTLP)

Focuses on **scalability** and **extensibility** for data warehouses / online analytical processing (OLAP)

# Scalability

Massive scale out and fault tolerance capabilities on commodity hardware

Can handle petabytes of data

Easy to provision (because of scale-out)

# Extensibility

Data types: primitive types and complex types

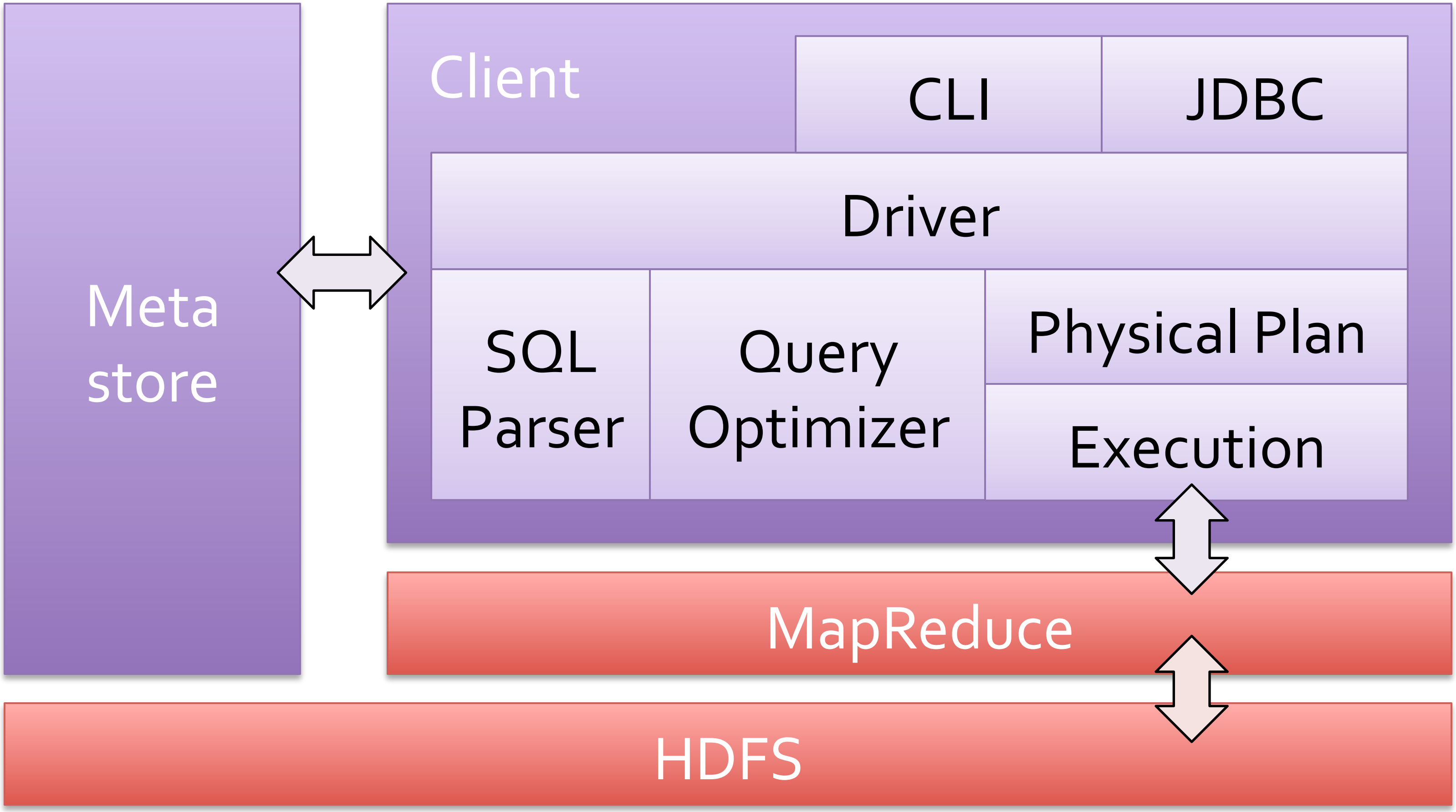
User-defined functions

Scripts

Serializer/Deserializer: text, binary, JSON…

Storage: HDFS, Hbase, S3…

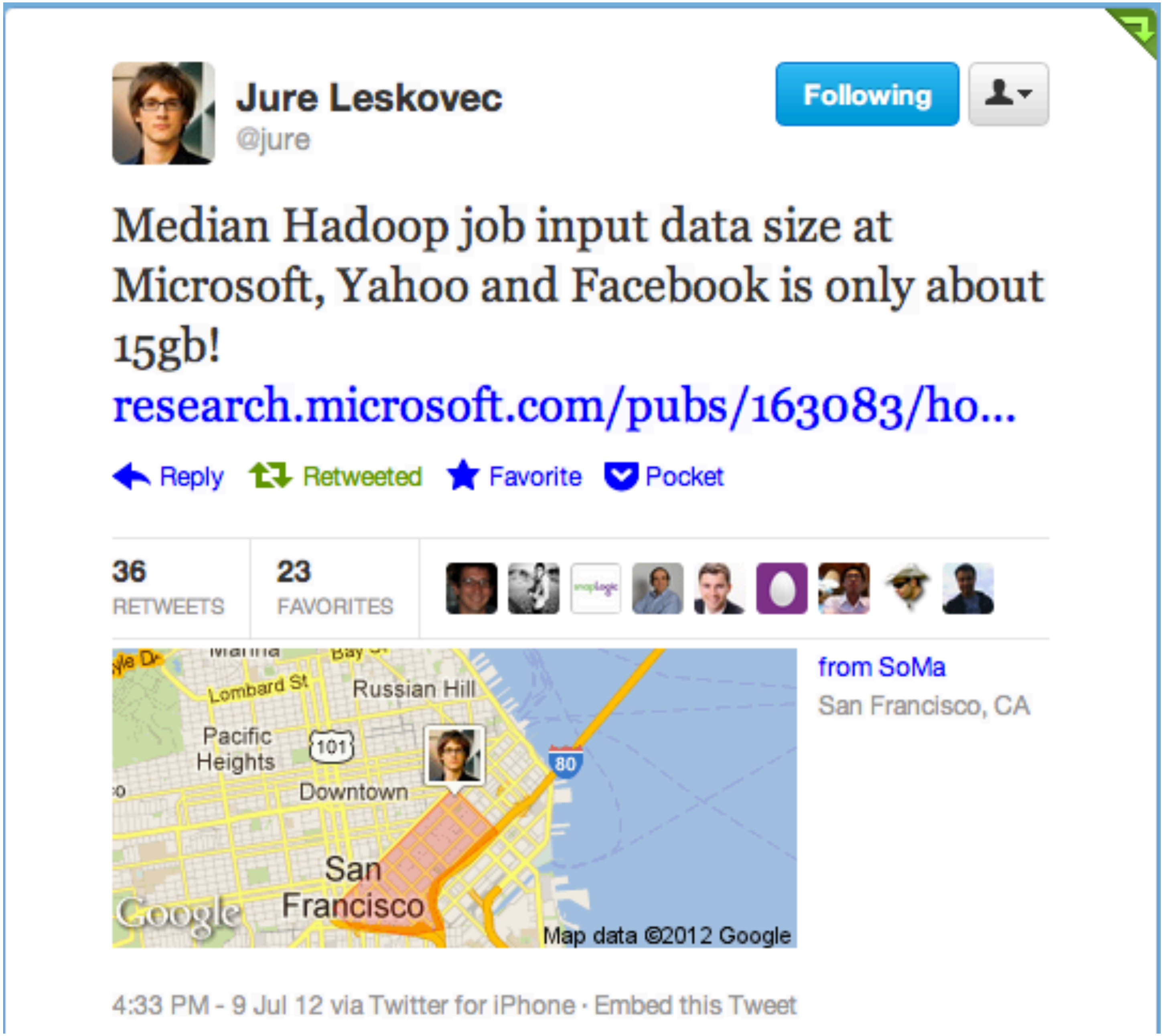
# Hive Architecture



# I/O Bound

Hive query processing is I/O bound

Can in-memory computation help in petabyte-scale warehouses?



Jure Leskovec
@jure

Following

Median Hadoop job input data size at Microsoft, Yahoo and Facebook is only about 15gb! research.microsoft.com/pubs/163083/ho...

36 RETWEETS

23 FAVORITES

from SoMa
San Francisco, CA

4:33 PM - 9 Jul 12 via Twitter for iPhone · Embed this Tweet

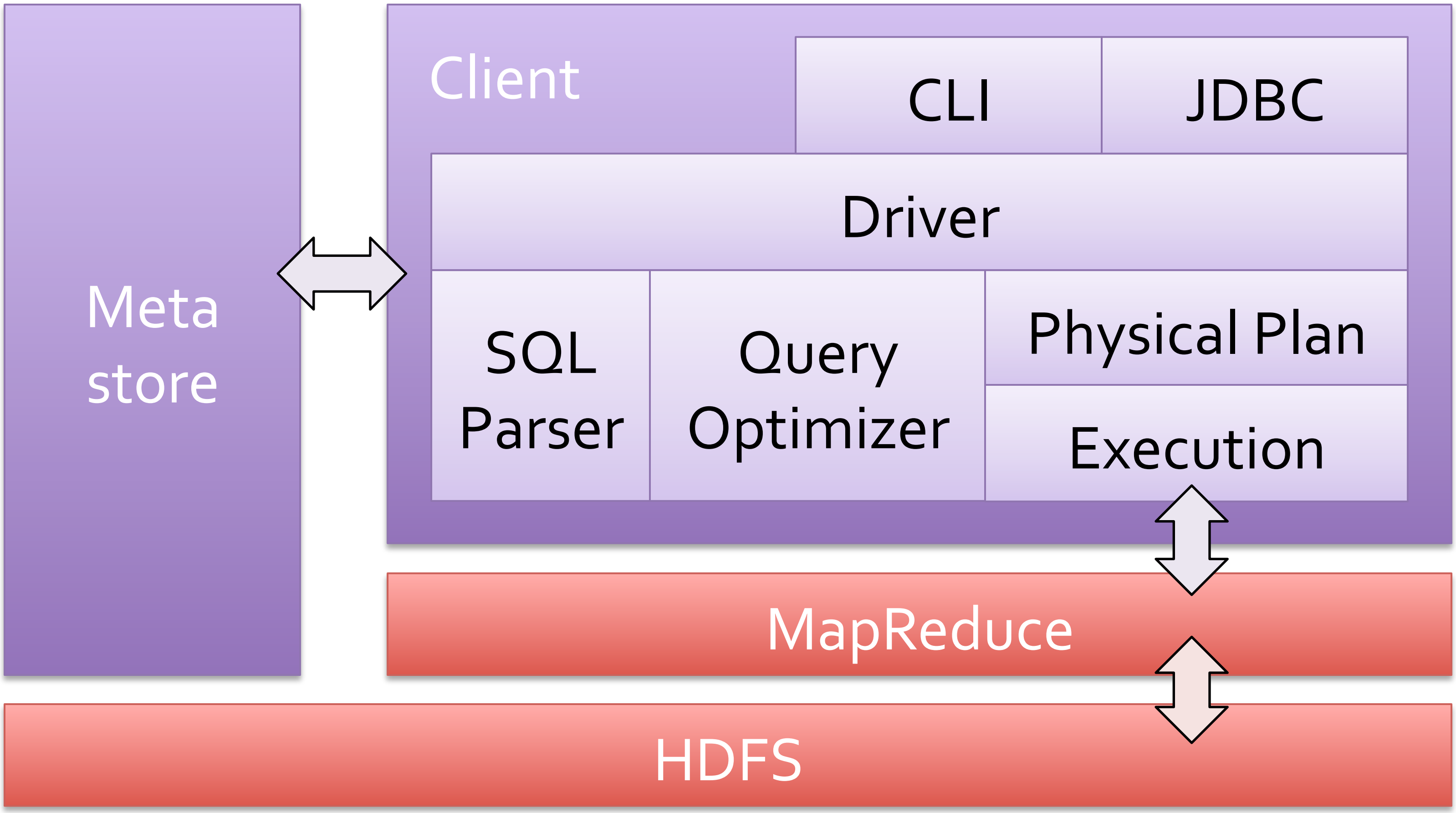
# Shark Motivations

Data warehouses exhibit a huge amount of temporal locality

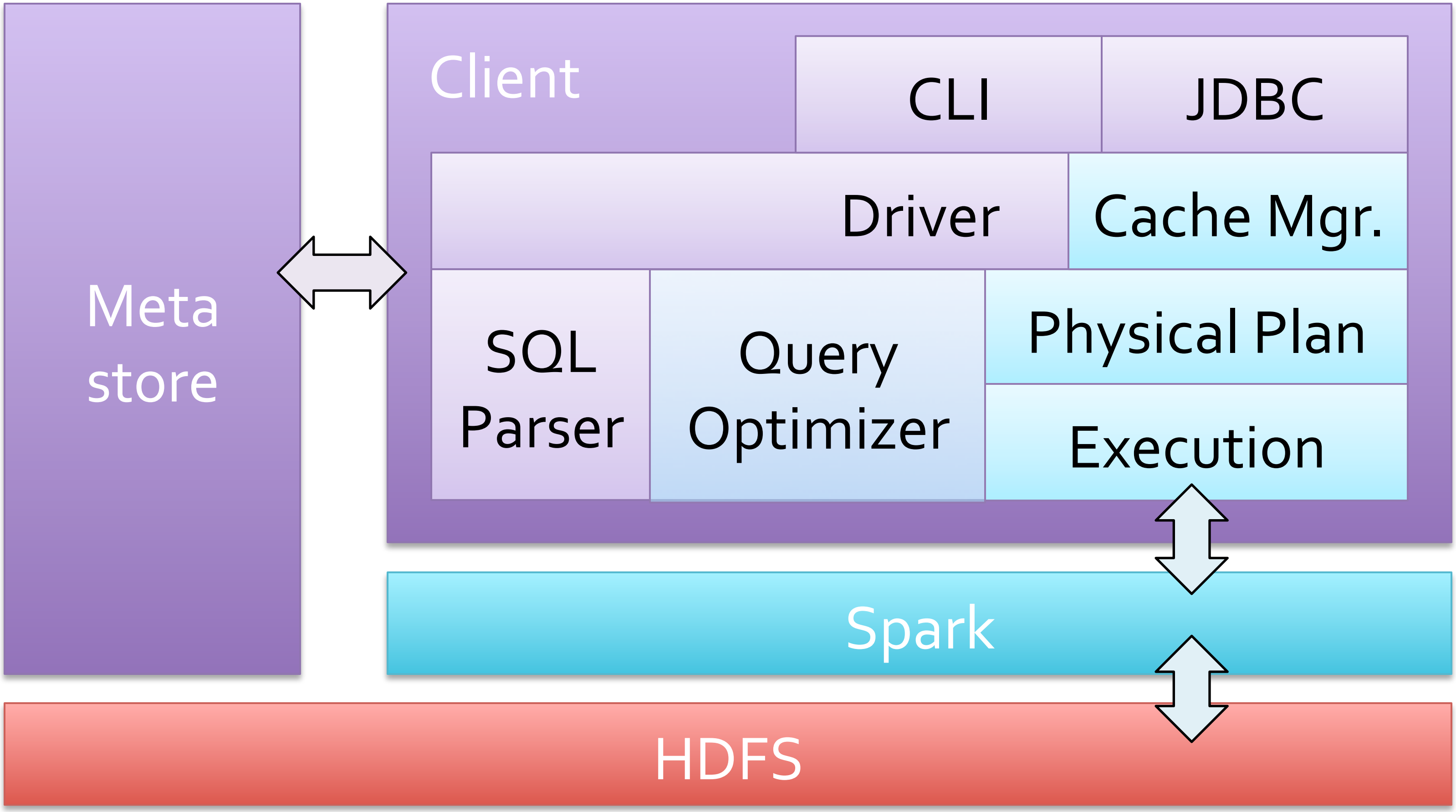
» 90% of Facebook queries could be served in RAM

Can we keep all the benefits of Hive (scalability and extensibility) and exploit the temporal locality?

# Hive


Client
CLI
JDBC
Driver
Meta store
SQL Parser
Query Optimizer
Physical Plan
Execution
MapReduce
HDFS

# Shark


Client
CLI
JDBC
Driver
Cache Mgr.
Meta store
SQL Parser
Query Optimizer
Physical Plan
Execution
Spark
HDFS

# Shark

Data warehouse system compatible with Hive

- » Supports Hive QL, UDFs, SerDes, scripts, types
- » A few esoteric features not yet supported

Makes Hive queries run faster

- » Allows caching data in a cluster's memory
- » Various other performance optimizations

Integrates with Spark for machine learning ops

# Caching Data in Shark

```
CREATE TABLE mytable_cached AS SELECT * FROM
mytable WHERE count > 10;
```

Creates a table cached in a cluster’s memory using RDD.cache()

# Outline

Hive and Shark

Data Model

Shark Demo

Beyond SQL Basics

# Data Model

Tables: unit of data with the same schema

Partitions: e.g. range-partition tables by date

Buckets: hash-partitions within partitions
(not yet supported in Shark)

# Data Types

Primitive types

- » TINYINT, SMALLINT, INT, BIGINT
- » BOOLEAN
- » FLOAT, DOUBLE
- » STRING

Complex types

- » Structs: STRUCT {a INT; b INT}
- » Arrays: ['a', 'b', 'c’]
- » Maps (key-value pairs): M['key’]

# Hive QL

Subset of SQL

- » Projection, selection
- » Group-by and aggregations
- » Sort by and order by
- » Joins
- » Sub-queries, unions

Hive-specific

- » Supports custom map/reduce scripts (TRANSFORM)
- » Hints for performance optimizations

# Outline

Hive and Shark

Data Model

Shark Demo

Beyond SQL Basics

# Outline

Hive and Shark

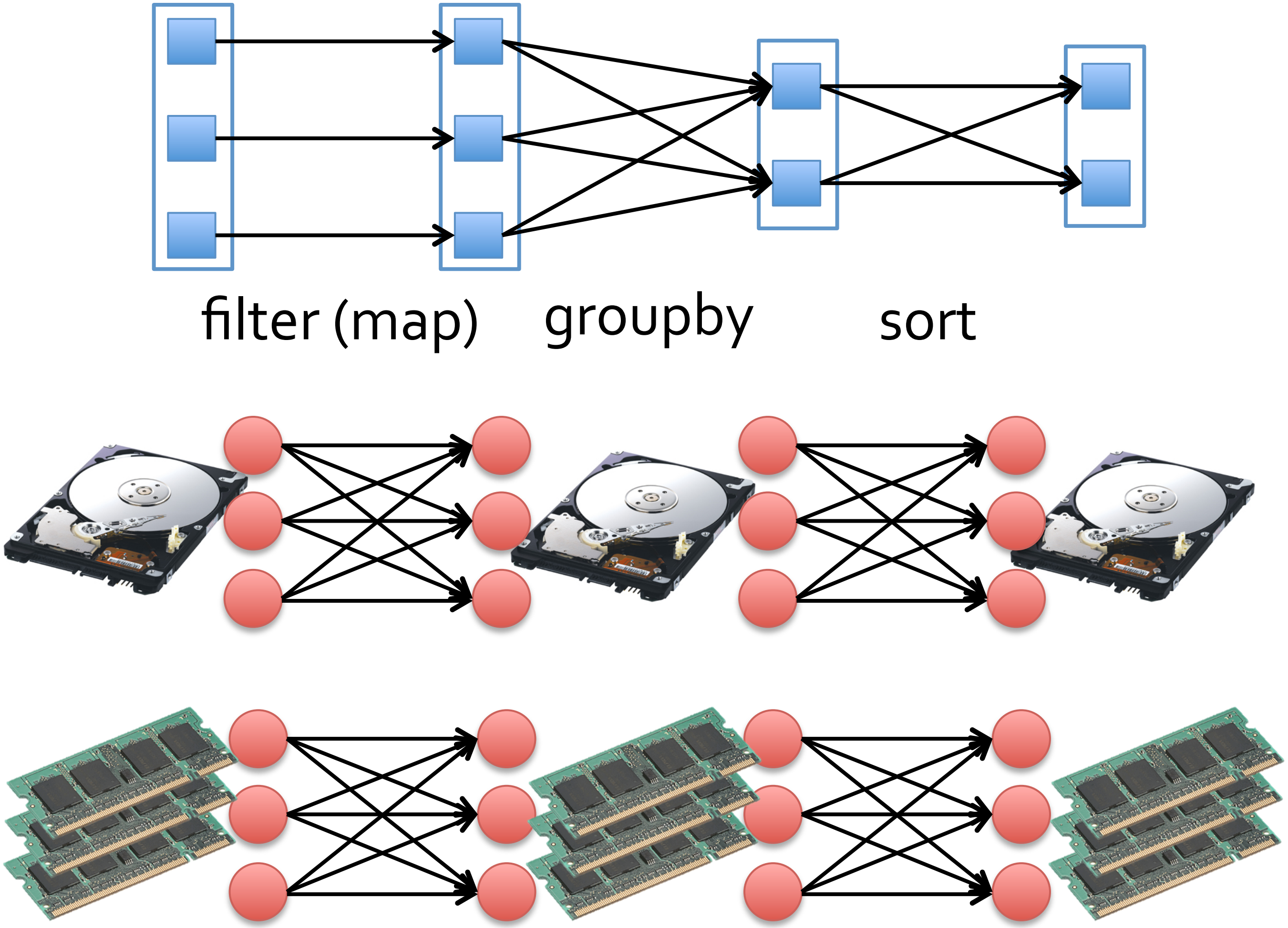
Data Model

Shark Demo

Beyond SQL Basics

```
select
  page_name,
  sum(page_views) hits
from wikistats_cached
where
  page_name like "%berkeley%”
group by page_name
order by hits;
```

```
select page_name, sum(page_views) hits
from wikistats_cached
where page_name like "%berkeley%”
group by page_name order by hits;
```



# Additional Improvements

Caching data in-memory

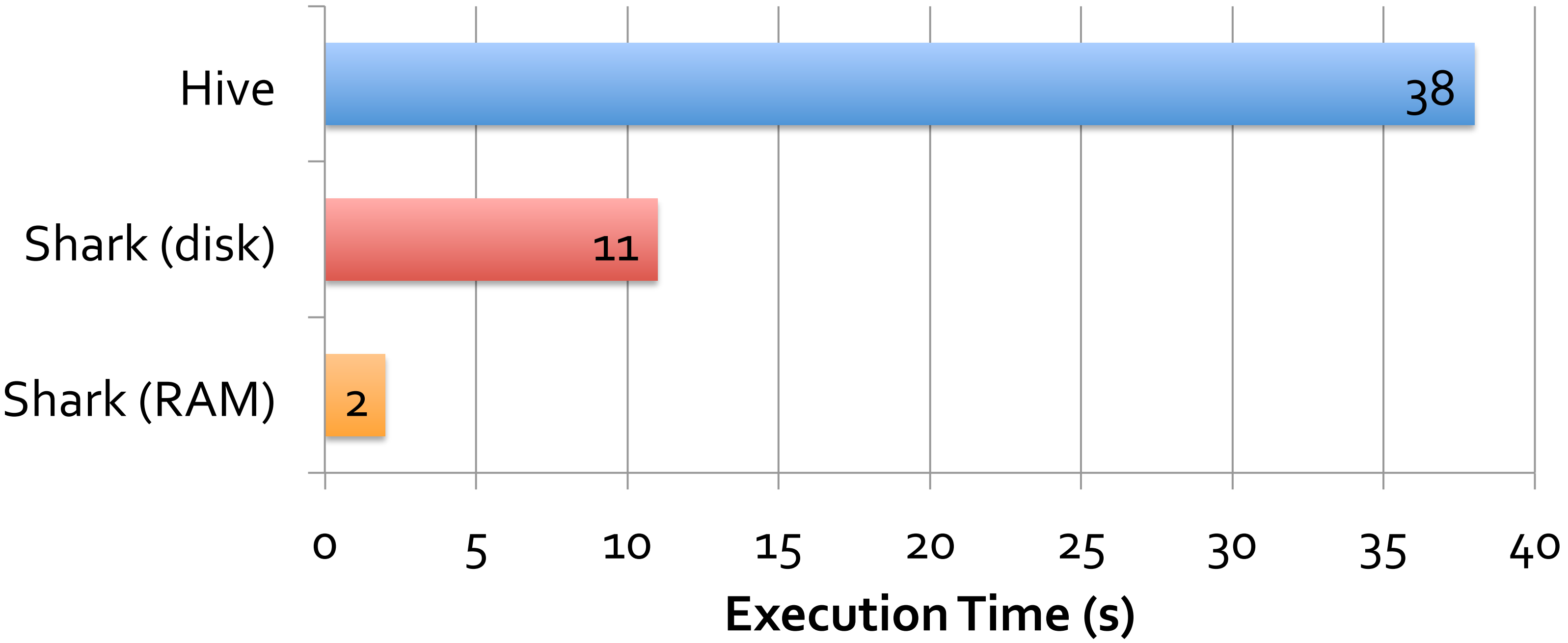
Hash-based shuffles for group-by

Distributed sort

Better push-down of limits

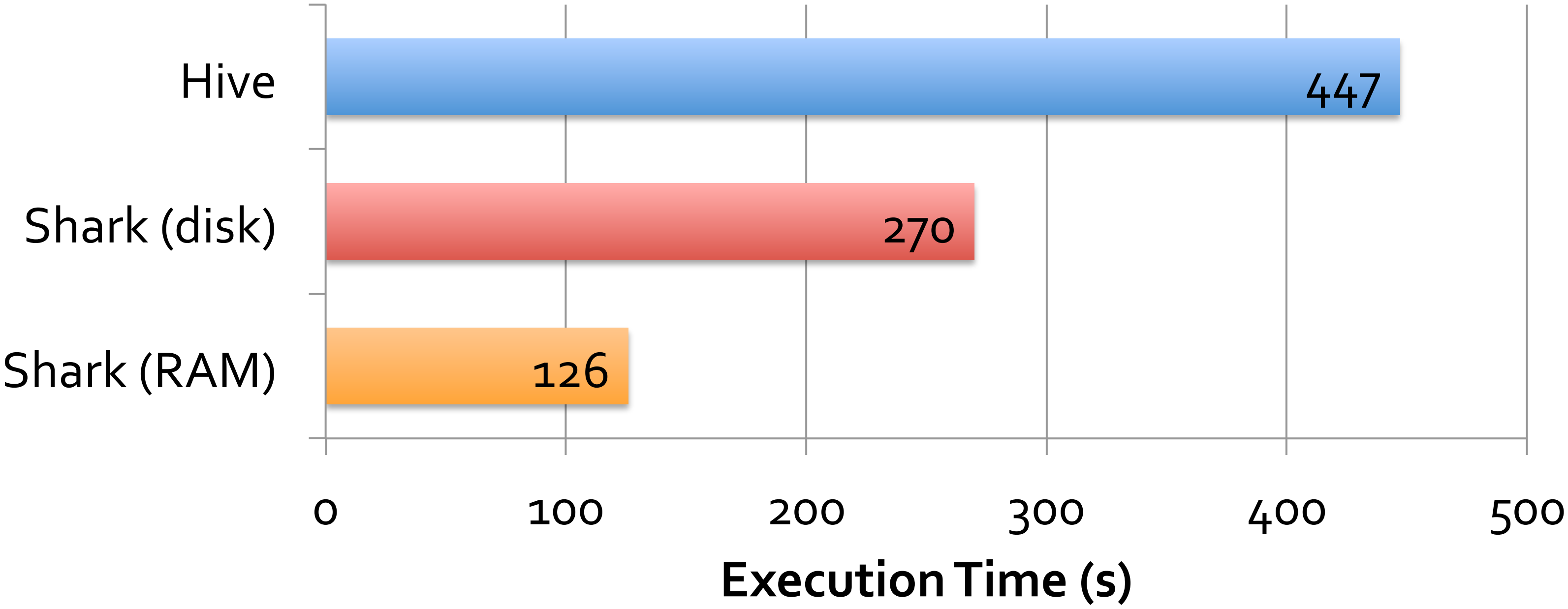
# Caching

```
SELECT * FROM pages WHERE body LIKE ‘%XYZ%’
```



# Sort, limit, hash shuffle

```
SELECT sourceIP, AVG(pageRank), SUM(adRevenue) AS earnings
FROM rankings AS R, userVisits AS V ON R.pageURL = V.destURL
WHERE V.visitDate BETWEEN ‘1999-01-01’ AND ‘2000-01-01’
GROUP BY V.sourceIP
ORDER BY earnings DESC LIMIT 1
```



# Performance Tuning

Two parameters that can significantly affect performance:

Setting the number of reducers

Map-side aggregation

# Num of Reducers

```
SET mapred.reduce.tasks=50;
```

Shark relies on Spark to infer the number of map tasks (automatically based on input size)

Number of reduce tasks needs to be specified by the user

Out of memory error on slaves if num too small

# Map-side Aggregation

```
SET hive.map.aggr=TRUE;
```

Aggregation functions are algebraic and can be applied on mappers to reduce shuffle data

Each mapper builds a hash-table to do the first-level aggregation

# Map-side Aggregation

Use: small number of distinct keys

```
SELECT dt, count(page_views)
FROM wikistats GROUP BY dt;
```

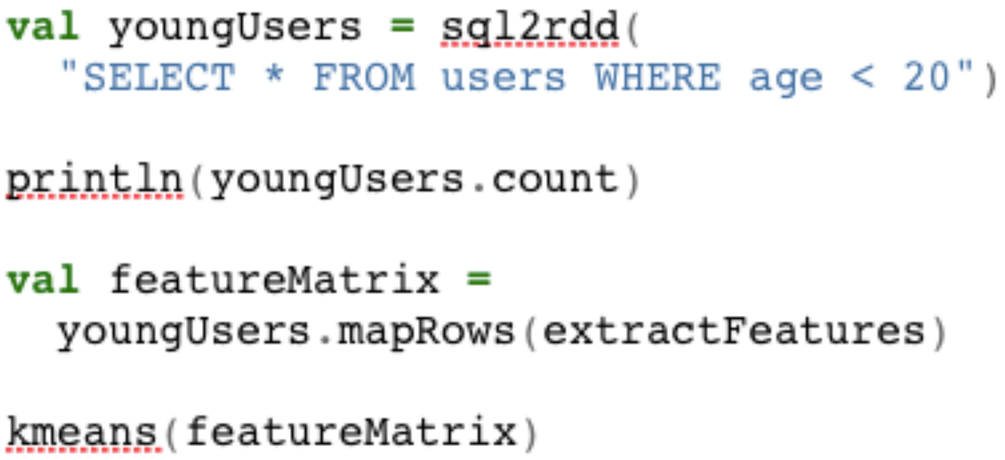
Do NOT use: large number of distinct keys

```
SELECT page_name, count(page_views)
FROM wikistats GROUP BY page_name;
```

# SQL/Spark Integration

Write simple SQL queries for extracting data, and express complex analytics in Spark

Query processing and machine learning share the same set of workers and caches



```
val youngUsers = sql2rdd(
  "SELECT * FROM users WHERE age < 20")

println(youngUsers.count)

val featureMatrix =
  youngUsers.mapRows(extractFeatures)

kmeans(featureMatrix)
```

# Getting Started

The Spark EC2 AMI comes with Shark installed (in /root/shark)

Requires Mesos to deploy in private cloud

Hadoop YARN deployment coming soon

# Exercises

Each on-site audience gets a 4-node EC2 cluster preloaded with Wikipedia traffic statistics data

Streaming audiences get an AMI preloaded with all software (Mesos, Spark, Shark)

Use Spark and Shark to analyze the data

# More Information

Hive resources:

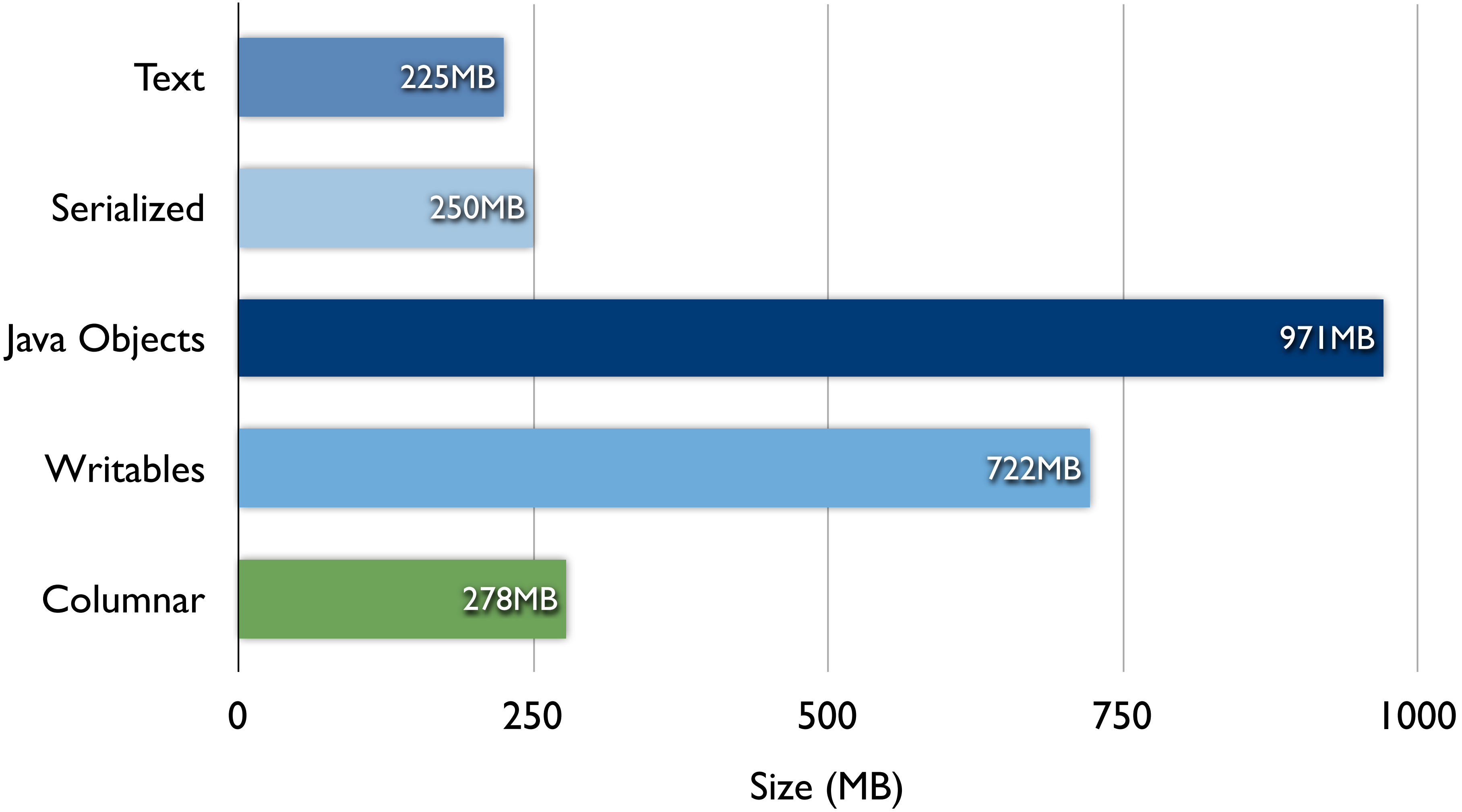
- https://cwiki.apache.org/confluence/display/Hive/GettingStarted
- http://hive.apache.org/docs/

Shark resources:

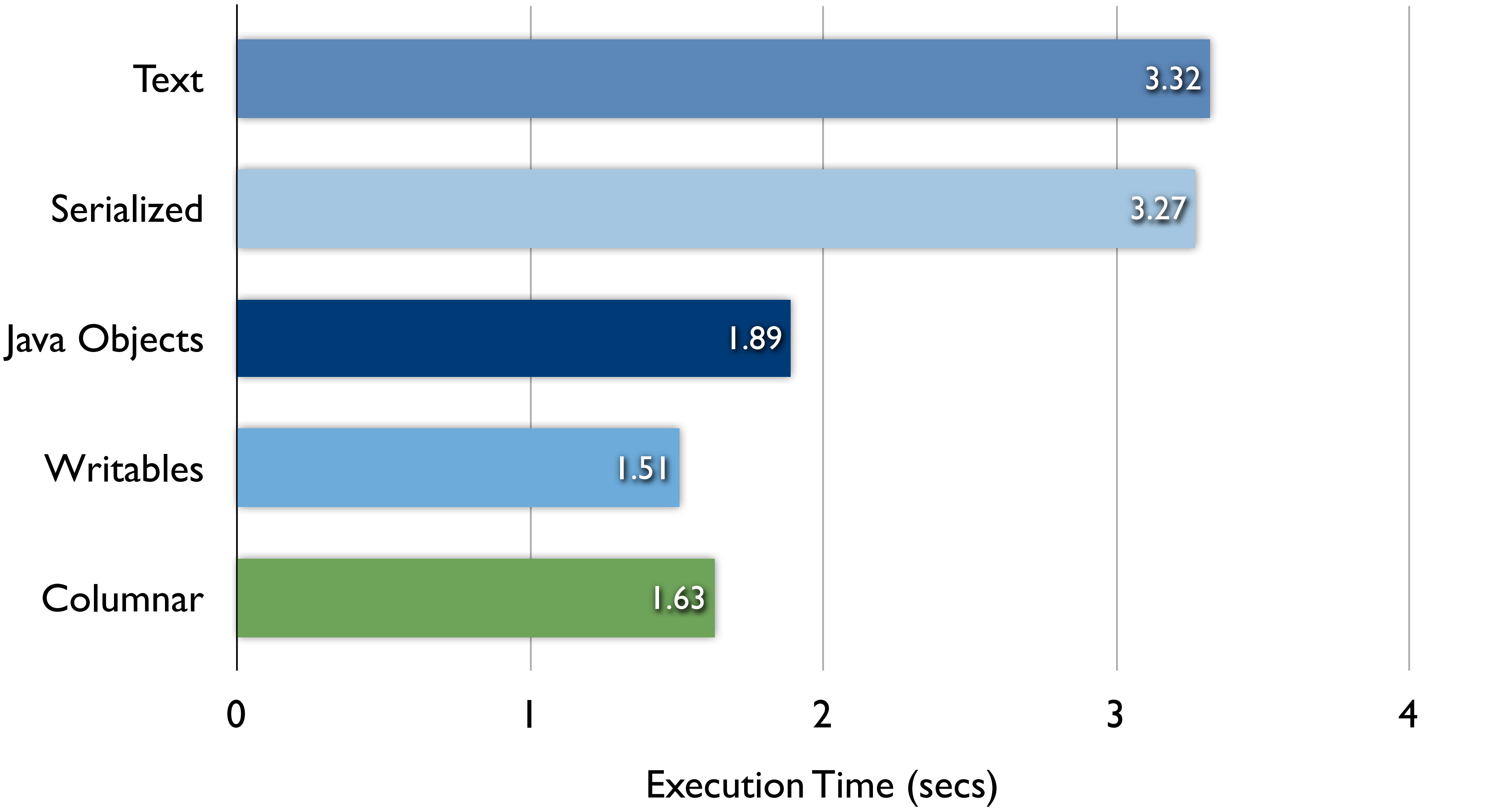
- http://shark.cs.berkeley.edu
- https://github.com/amplab/shark

# Backup slides

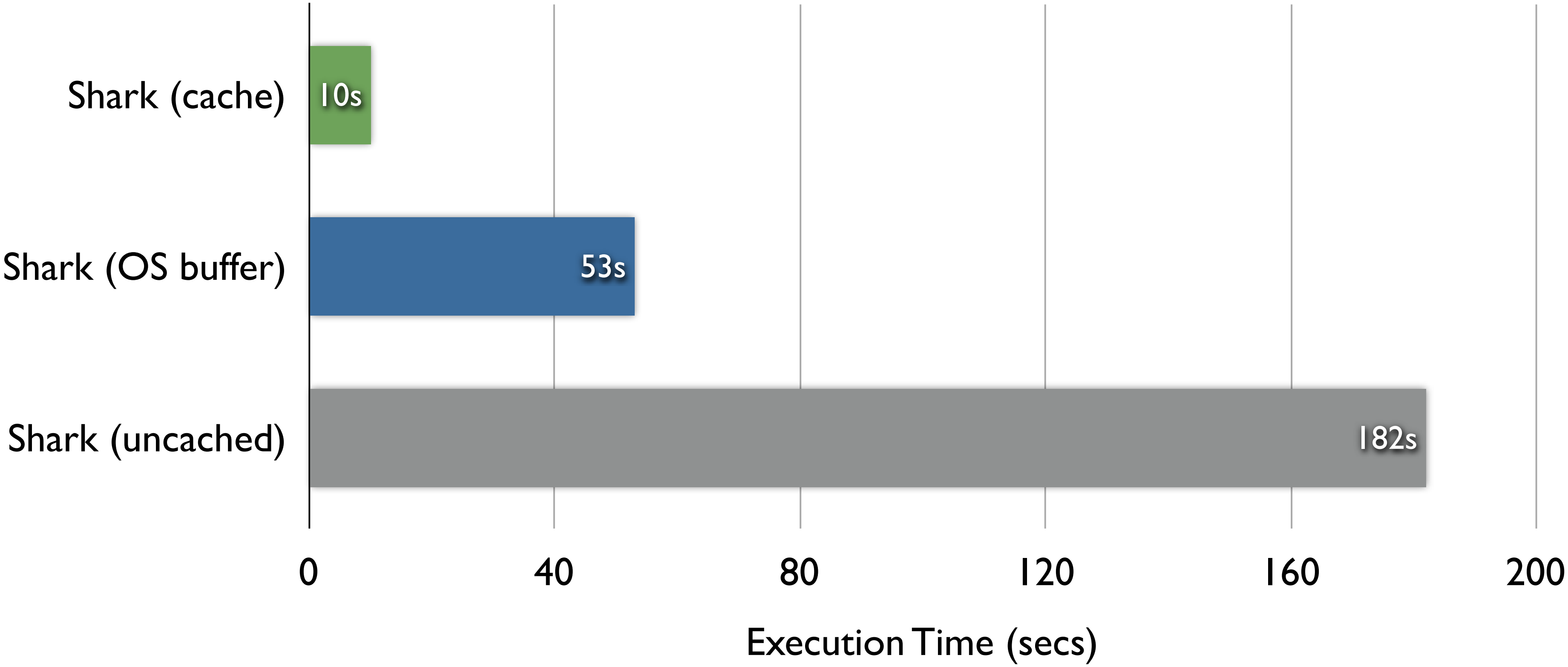
## Memory Footprint of Caching TPC-H lineitem Table (1.8 million rows)



Performance Comparison of Caching (selecting 725 rows out of 1.8 million rows)



## Comparison with OS Buffer Cache (Query 1. Large Sequential Scan & Grep)


Shark (cache)
10s
Shark (OS buffer)
53s
Shark (uncached)
182s
Execution Time (secs)