O'REILLY®

Strata CONFERENCE

Making Data Work

Feb. 26 – 28, 2013

SANTA CLARA, CA

strataconf.com
#strataconf

Shark: SQL and Rich Analytics at Scale

Reynold Xin
UC Berkeley

amplab

Challenges in Modern Data Analysis

- Data volumes expanding.
- Faults and stragglers complicate parallel database design.
- Complexity of analysis: machine learning, graph algorithms, etc.
- Low-latency, interactivity.

MapReduce

- Apache Hive, Google Tenzing, Turn Cheetah...
- Enables fine-grained fault-tolerance, resource sharing, scalability.
- Expressive Machine Learning algorithms.
- High-latency, dismissed for interactive workloads.

MPP Databases

- Vertica, SAP HANA, Teradata, Google Dremel, Google PowerDrill, Cloudera Impala...
- Fast!
- Generally not fault-tolerant; challenging for long running queries as clusters scale up.
- Lack rich analytics such as machine learning and graph algorithms.

Apache Hive

- A data warehouse
  - initially developed by Facebook
  - puts structure/schema onto HDFS data (schema-on-read)
  - compiles HiveQL queries into MapReduce jobs
  - flexible and extensible: support UDFs, scripts, custom serializers, storage formats.

- Popular: 90+% of Facebook Hadoop jobs generated by Hive

- But slow: 30+ seconds even for simple queries

What is Shark?

- A data analysis (warehouse) system that
  - builds on Spark (MapReduce deterministic, idempotent tasks),
  - scales out and is fault-tolerant,
  - supports low-latency, interactive queries through in-memory computation,
  - supports both SQL and complex analytics such as machine learning,
  - is compatible with Apache Hive (storage, serdes, UDFs, types, metadata).

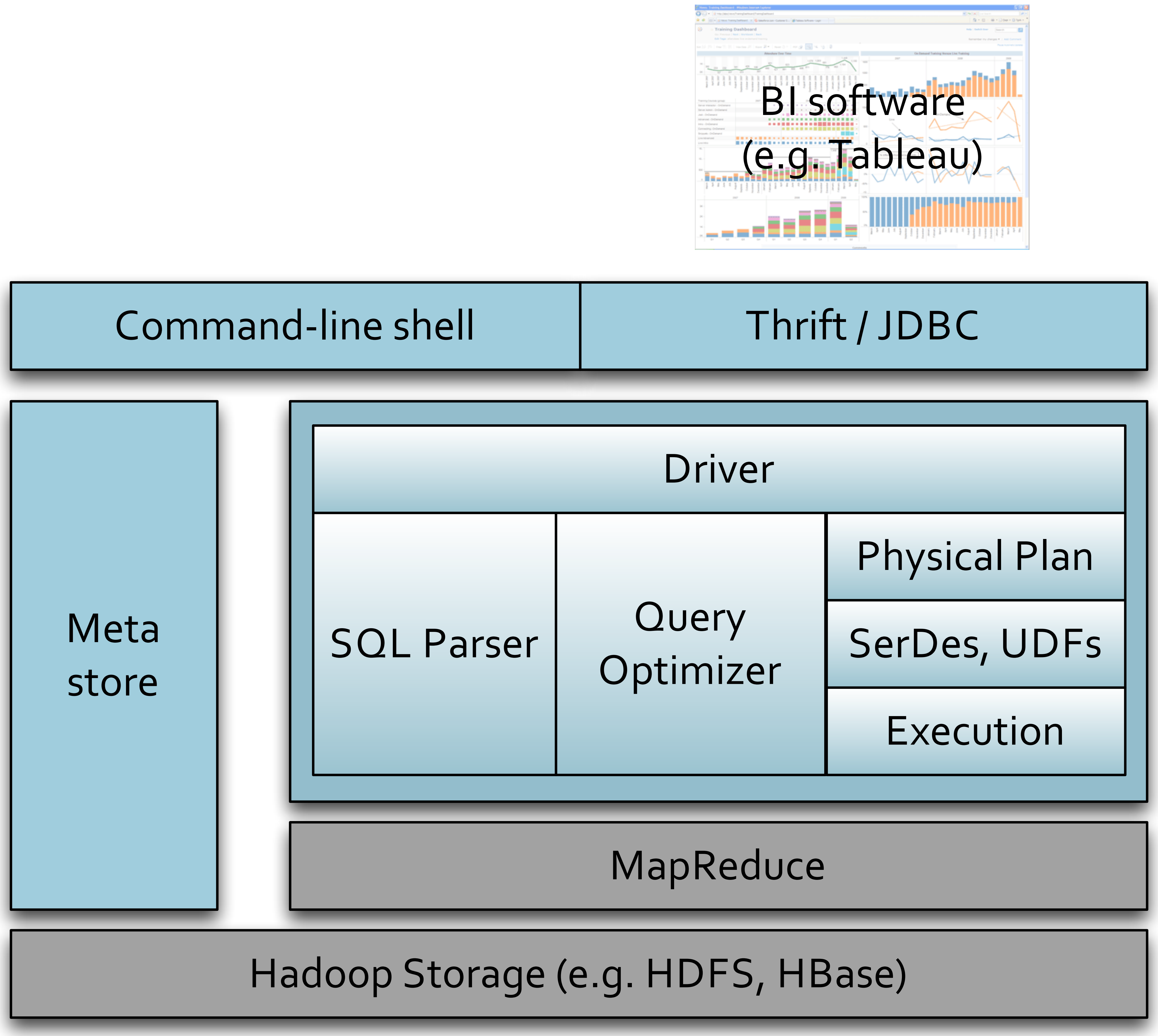
What is Shark?

- A data analysis (warehouse) system that

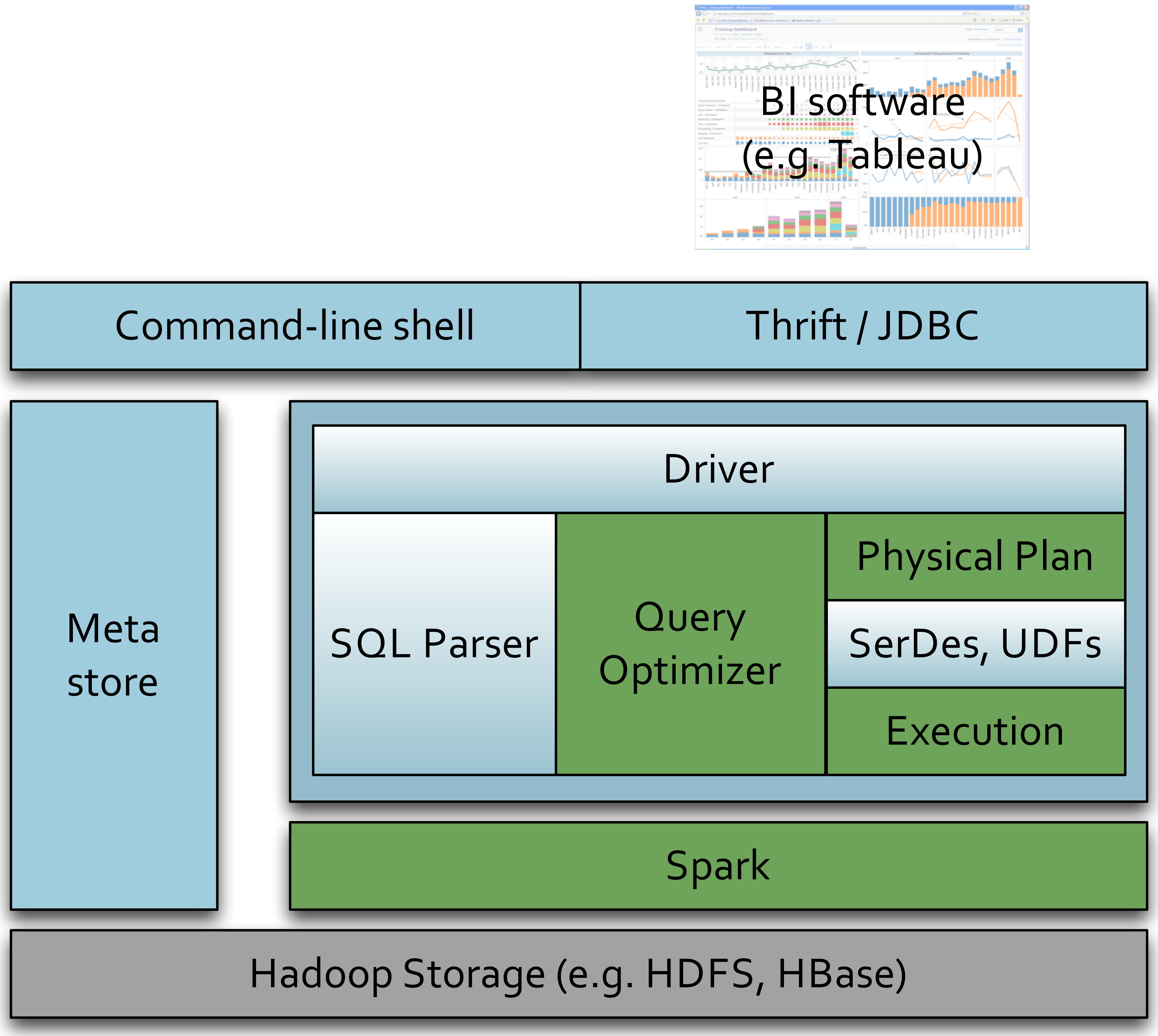
HOW DO I FIT PB OF DATA IN MEMORY???

  - builds on Spark (MapReduce deterministic, idempotent tasks),
  - scales out and is fault-tolerant,
  - supports low-latency, interactive queries through in-memory computation,
  - supports both SQL and complex analytics such as machine learning,
  - is compatible with Apache Hive (storage, serdes, UDFs, types, metadata).


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...
Reply
Retweeted
Favorite
Pocket
36
RETWEETS
23
FAVORITES
Lombard St
Russian Hill
Pacific Heights
101
Downtown
80
San Francisco
Google
Map data ©2012 Google
from SoMa
San Francisco, CA
4:33 PM - 9 Jul 12 via Twitter for iPhone · Embed this Tweet



Hive Architecture



Shark Architecture

Analyzing Data

- ```
  CREATE EXTERNAL TABLE wiki
  (id BIGINT, title STRING, last_modified STRING, xml STRING, text STRING)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
  LOCATION 's3n://spark-data/wikipedia-sample/';
  ```

- ```
  SELECT COUNT(*) FROM wiki_small WHERE TEXT LIKE '%Berkeley%';
  ```

Caching Data in Shark

- `CREATE TABLE wiki_small_in_mem TBLPROPERTIES ("shark.cache" = "true") AS SELECT * FROM wiki;`

- `CREATE TABLE wiki_cached AS SELECT * FROM wiki;`

- Creates a table that is stored in a cluster’s memory using RDD.cache().

Tuning the Degree of Parallelism

- 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.
  - `SET mapred.reduce.tasks=499;`
- Out of memory error on slaves if the number is too small.
- It is usually OK to set a higher value since the overhead of task launching is low in Spark.

Demo

18 months of Wikipedia traffic statistics

Engine Extensions and Features

- Partial DAG Execution (coming soon)
- Columnar Memory Store
- Machine Learning Integration
- Hash-based Shuffle vs Sort-based Shuffle
- Data Co-partitioning (coming soon)
- Partition Pruning based on Range Statistics
- Distributed Data Loading
- Distributed sorting
- Better push-down of limits
- ...

Partial DAG Execution (PDE)

- How to optimize the following query?

- `SELECT * FROM table1 a JOIN table2 b ON a.key=b.key`
  `WHERE my_crazy_udf(b.field1, b.field2) = true;`

Partial DAG Execution (PDE)

- How to optimize the following query?

- ```
  SELECT * FROM table1 a JOIN table2 b ON a.key=b.key
  WHERE my_crazy_udf(b.field1, b.field2) = true;
  ```

- Hard to estimate cardinality!
- Without cardinality estimation, cost-based optimizer breaks down.
  ```

Partial DAG Execution (PDE)

- PDE allows *dynamic alternation of query plans* based on statistics collected at run-time.

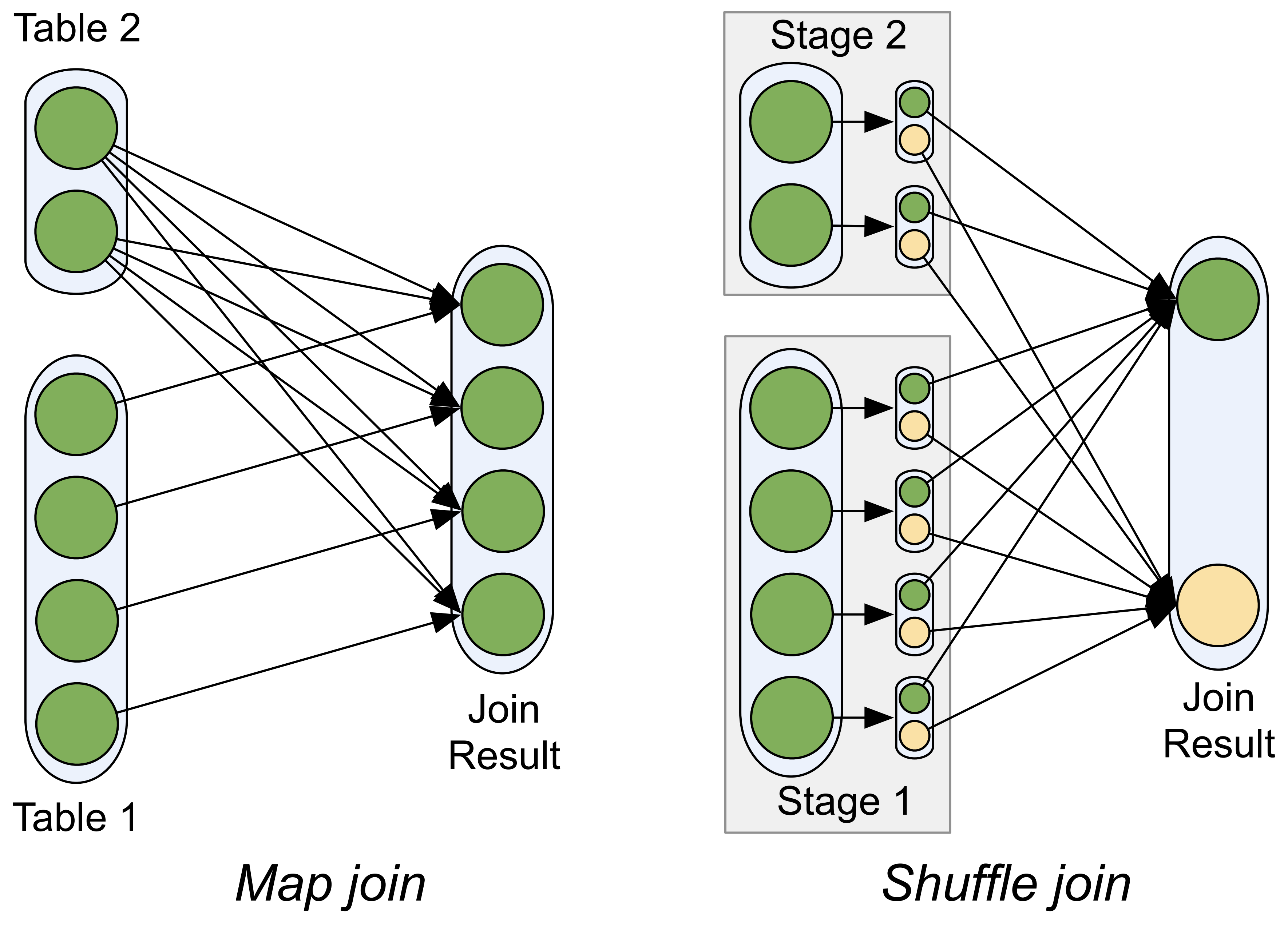
- Can gather customizable statistics at global and per-partition granularities while materializing map output.
  - partition sizes, record counts (skew detection)
  - “heavy hitters”
  - approximate histograms

Partial DAG Execution (PDE)

- PDE allows *dynamic alternation of query plans* based on statistics collected at run-time.

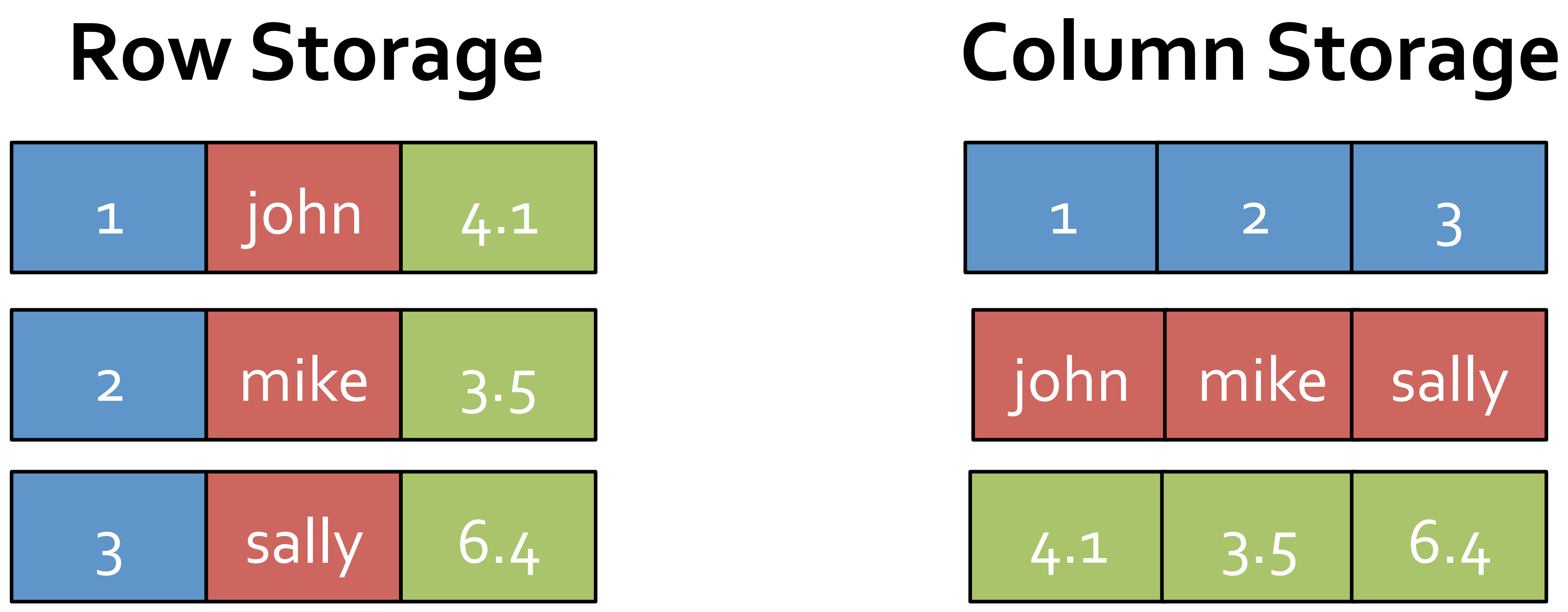
- Can gather customizable statistics at global and per-partition granularities while materializing map output.
  - partition sizes, record counts (skew detection)
  - “heavy hitters”
  - approximate histograms

- Alter query plan based on such statistics.
  - map join vs shuffle join
  - symmetric vs non-symmetric hash join



Columnar Memory Store

- Simply caching Hive records as JVM objects is inefficient.
- Shark employs column-oriented storage using *arrays of primitive* objects.



- Compact storage (as much as 5X less space footprint).
- JVM garbage collection friendly.
- CPU-efficient compression (e.g. dictionary encoding, run-length encoding, bit packing).

Machine Learning Integration

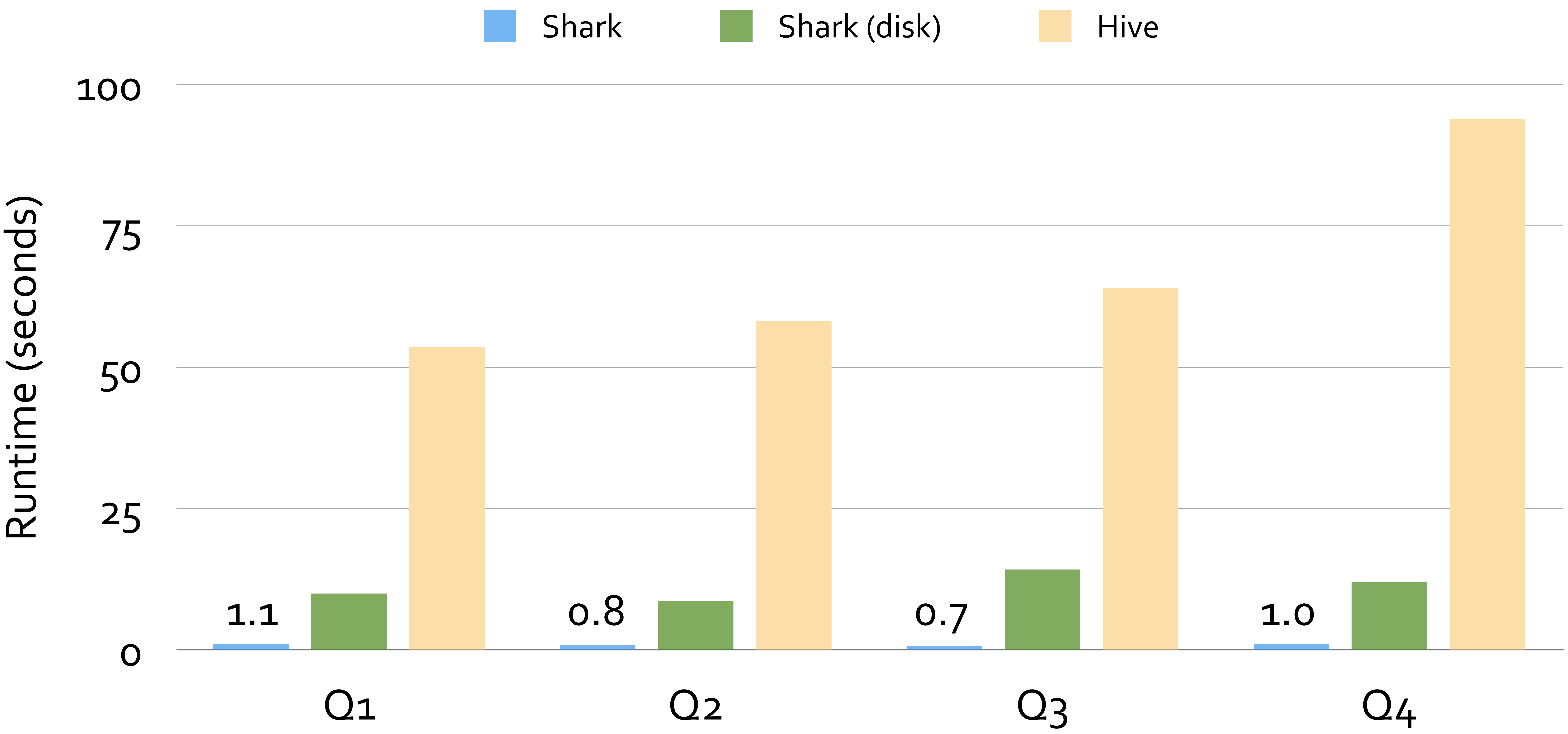
- Unified system for query processing and machine learning
- Write machine learning algorithms in Spark, optimized for iterative computations
- Query processing and ML share the same set of workers and caches

```
def logRegress(points: RDD[Point]): Vector {
  var w = Vector(D, _ => 2 * rand.nextDouble - 1)
  for (i <- 1 to ITERATIONS) {
    val gradient = points.map { p =>
      val denom = 1 + exp(-p.y * (w dot p.x))
      (1 / denom - 1) * p.y * p.x
    }.reduce(_ + _)
    w -= gradient
  }
  w
}

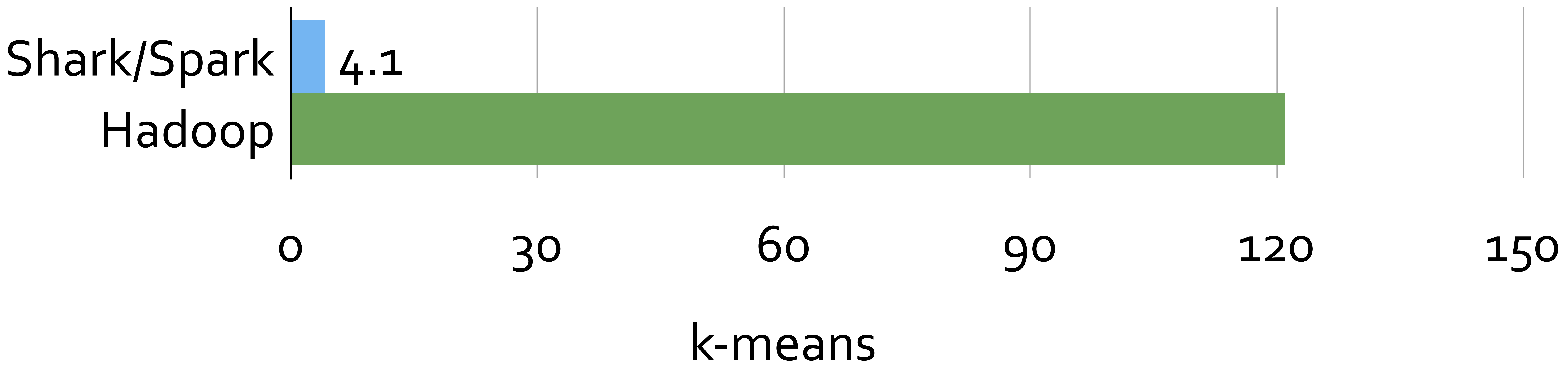
val users = sql2rdd("SELECT * FROM user u
   JOIN comment c ON c.uid=u.uid")

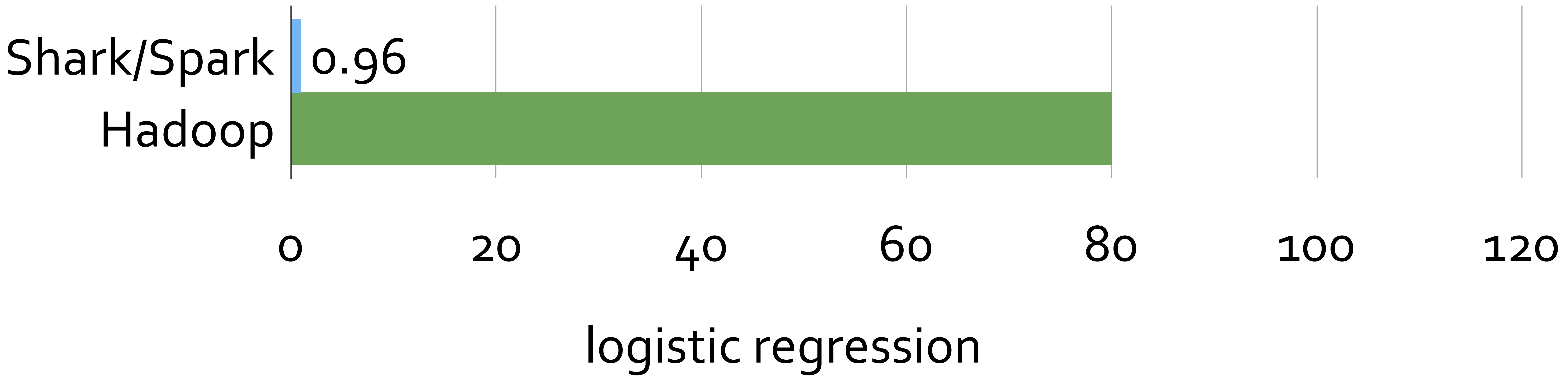
val features = users.mapRows { row =>
  new Vector(extractFeature1(row.getInt("age")),
             extractFeature2(row.getStr("country")),
             ...)}
val trainedVector = logRegress(features.cache())
```

Conviva Warehouse Queries (1.7 TB)



Machine Learning (1B records, 10 features/record)





Getting Started

- ~ 5 mins to install Shark locally
  - https://github.com/amplab/shark/wiki

- The Spark EC2 AMI comes with Shark installed (in /root)
  - ```
    spark-ec2 -k <keypair> -i <key-file> -s <num-slaves> launch <cluster-name>
    ```

- Also supports Amazon Elastic MapReduce (EMR)
  - http://tinyurl.com/spark-emr

- Use Apache Mesos or Spark standalone cluster mode for private cloud,
    ```

Open Source Development

- Spark/Shark is a very small code base.
  - Spark: 20K LOC
  - Shark: 7K LOC
- Easy to adapt and tailor to specific use cases.
- Already accepted major contributions from Yahoo!, ClearStory Data, Intel.
- Mailing list: shark-users @ googlegroups

Summary

- By using Spark as the execution engine and employing novel and traditional database techniques, Shark bridges the gap between MapReduce and MPP databases.
- It can answer queries up to 100X faster than Hive and machine learning 100X faster than Hadoop MapReduce.
- Try it out on EC2 (takes 10 mins to spin up a cluster): http://shark.cs.berkeley.edu

O'REILLY®

Strata CONFERENCE

Making Data Work

Feb. 26 – 28, 2013

SANTA CLARA, CA

strataconf.com

#strataconf

backup slides

	Shark	Impala
Focus	integrate SQL with complex analytics	data warehouse / OLAP
Execution	Spark (MapReduce like)	Parallel Databases
In-memory	in-memory tables	no (buffer cache)
Fault-tolerance	tolerate slave failures	no
Large (out-of-core) joins	yes	no
UDF	yes	no

Why are previous MR-based systems slow?

- Disk-based intermediate outputs.
- Inferior data format and layout (no control of data co-partitioning).
- Execution strategies (lack of optimization based on data statistics).
- Task scheduling and launch overhead!

Task Scheduling and Launch Overhead

- Hadoop uses heartbeat to communicate scheduling decisions.

- Hadoop task launch delay 5 - 10 seconds.

- Spark uses an event-driven architecture and can launch tasks in 5ms.
  - better parallelism
  - easier straggler mitigation
  - elasticity
  - multi-tenancy resource sharing

Task Scheduling and Launch Overhead