SPARK: FAULT-TOLERANT IN-MEMORY CLUSTER COMPUTING

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael Franklin, Scott Shenker, Ion Stoica (UC Berkeley)

MOTIVATION

- Cluster computing frameworks like MapReduce and Dryad provide a wide range of computational operators, but lack an abstraction for memory
- This makes them inefficient for apps that *reuse* datasets:
  - **Iterative** algorithms (machine learning, graphs, …)
  - **Interactive** data mining (e.g. Matlab, Python, SQL)

CHALLENGE

- How do we design a distributed memory abstraction that is both *general*, *fault-tolerant* and *efficient*?
- Traditional in-memory storage systems (key-value stores, databases, etc) replicate data or logs for fault tolerance, which would greatly slow down in-memory computation

RESILIENT DISTRIBUTED DATASETS (RDDs)

- Achieve fault tolerance efficiently by restricting the programming interface to *coarse-grained operations*
- Can then recover using *lineage* (log one operation to apply to many records, rather than logging the data)
- Still general enough to express many parallel algorithms, because these algorithms are data-parallel to start with
  - Can express MapReduce, Dryad, SQL, Pregel, iterative MR (Haloop), and new apps that these don't capture
  - Unify these specialized models for the first time

CURRENT PROJECTS

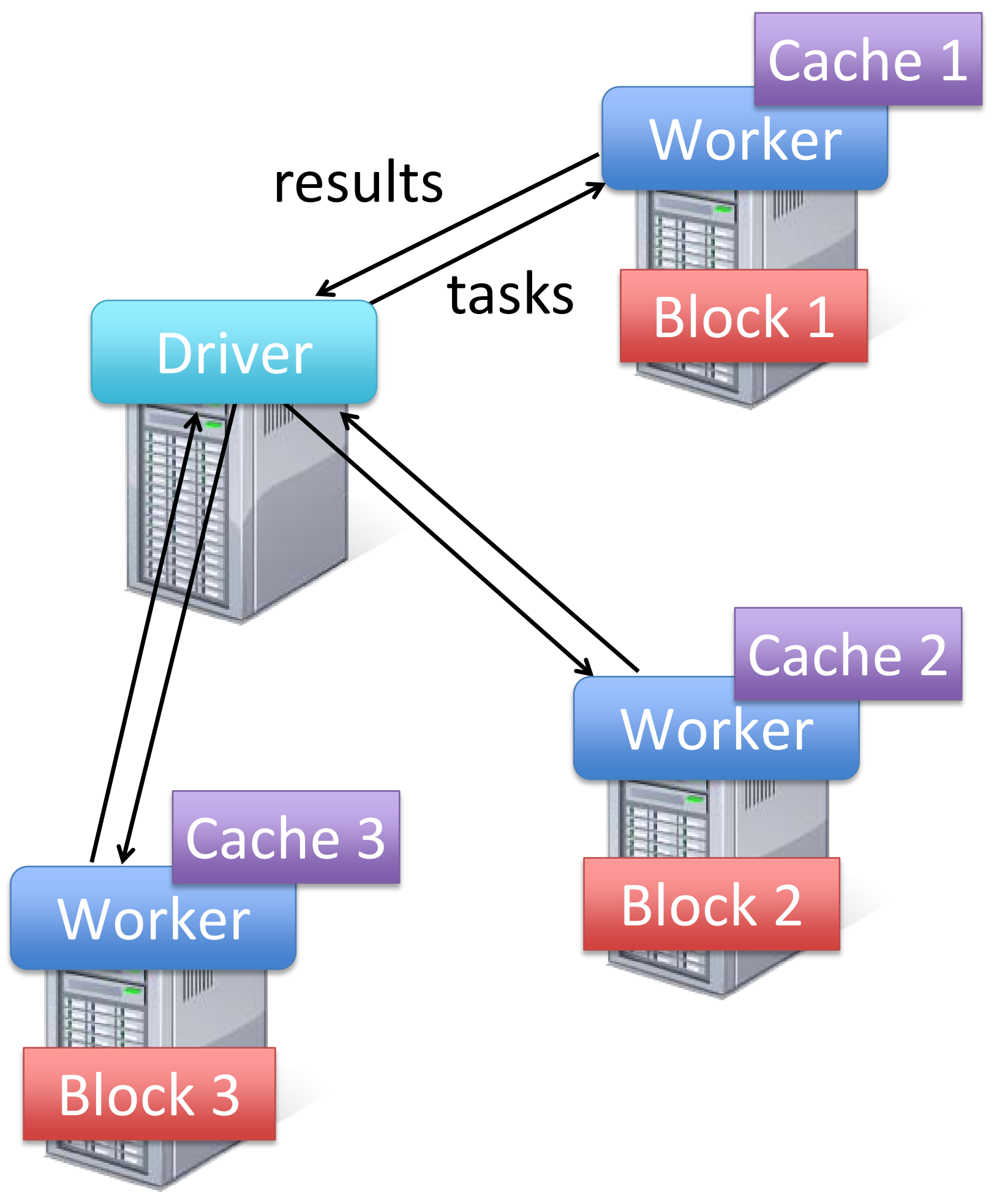
- Hive on Spark (Shark): interactive SQL queries on big data at 20x the speed of Apache Hive
- Lineage-based replay debugger:
  - Rebuild RDDs created during a Spark program and query them interactively
  - Re-run any task in a Java debugger (recreating its data)
- Streaming Spark: extend RDDs for low-latency processing

OPEN SOURCE: www.spark-project.org

Intel Science & Technology Center for Cloud Computing

ARCHITECTURE

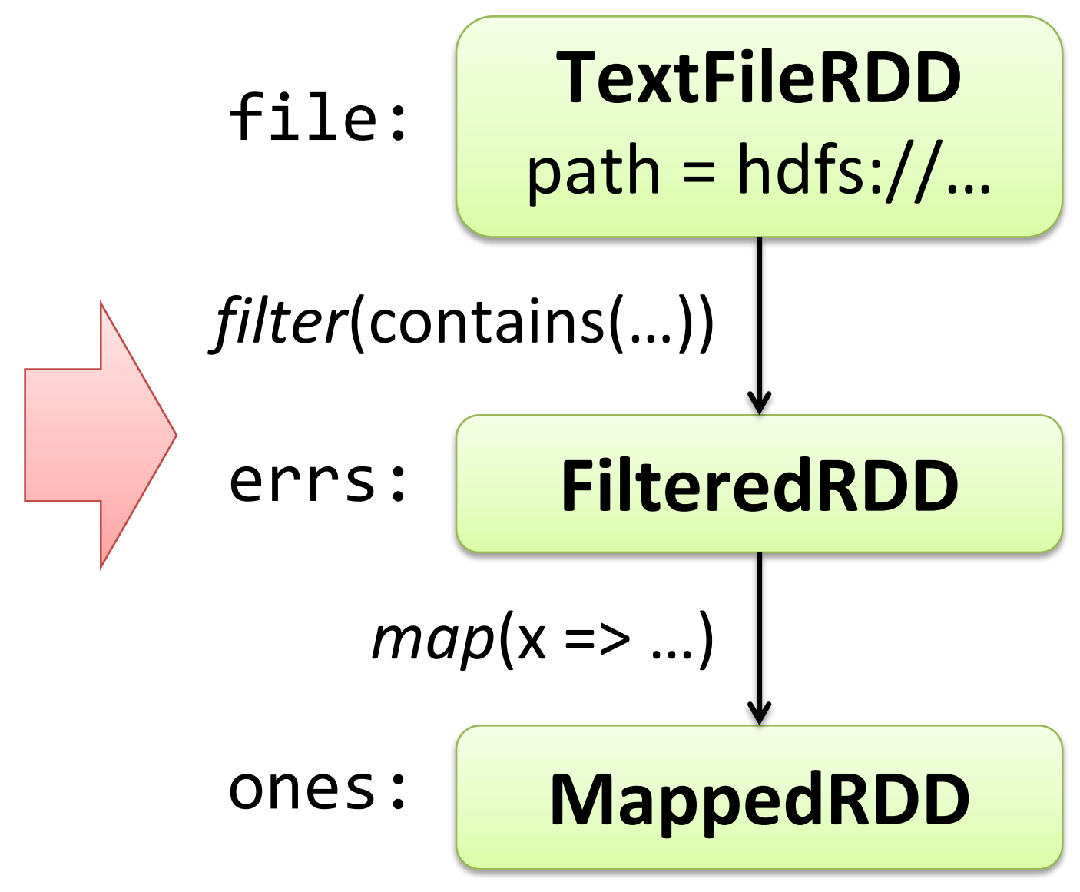
- Nodes keep partitions of RDDs in RAM as requested by user
- Fault tolerance through *lineage*
  - RDDs remember series of transformations needed to rebuild each partition
- Language-integrated Scala API
- Runs on Mesos resource mgr.
- Can share data with Hadoop



LINEAGE EXAMPLE

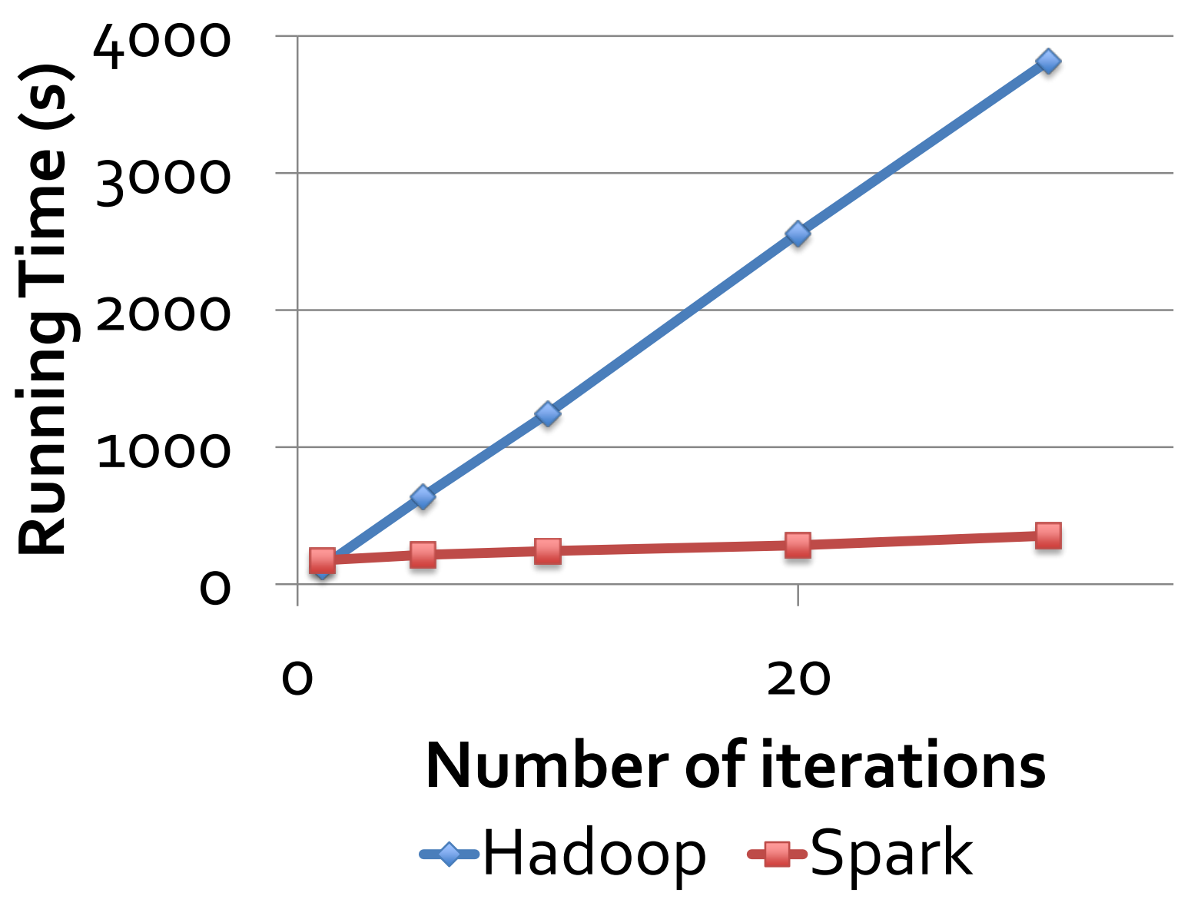
```
// Build an RDD containing all the
// lines with “ERROR” in a log file
file = spark.textFile("hdfs://...”)
errs = file.filter(_.contains("ERROR"))
errs.persist()

// Count errors in the in-memory RDD
ones = errs.map(_ => 1)
count = ones.reduce(_ + _)
```

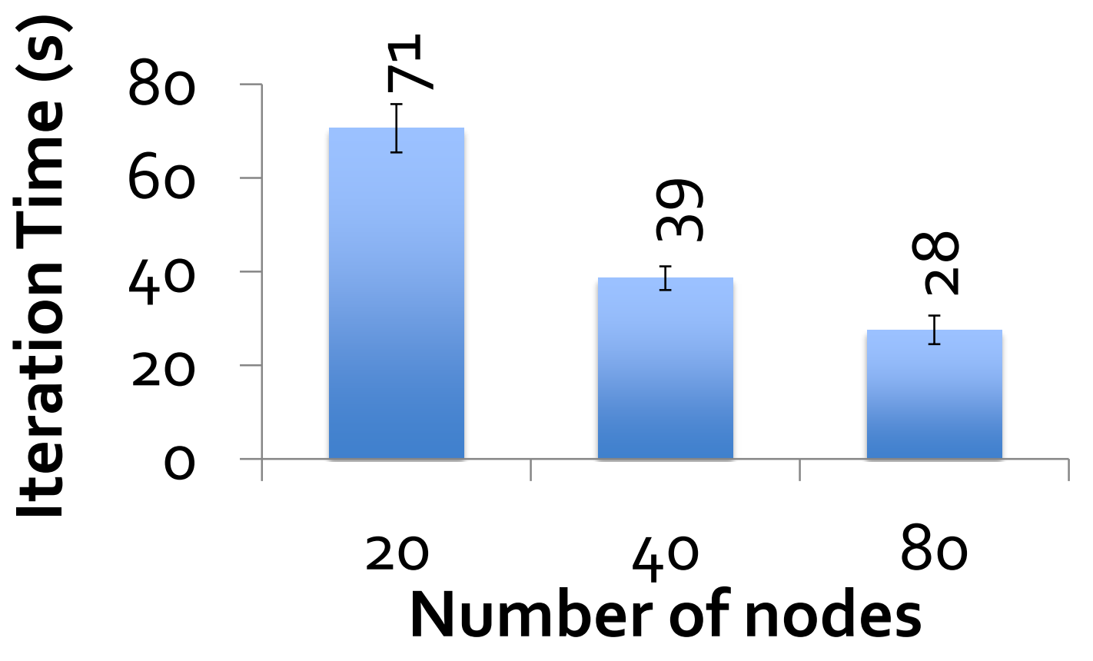


RESULTS

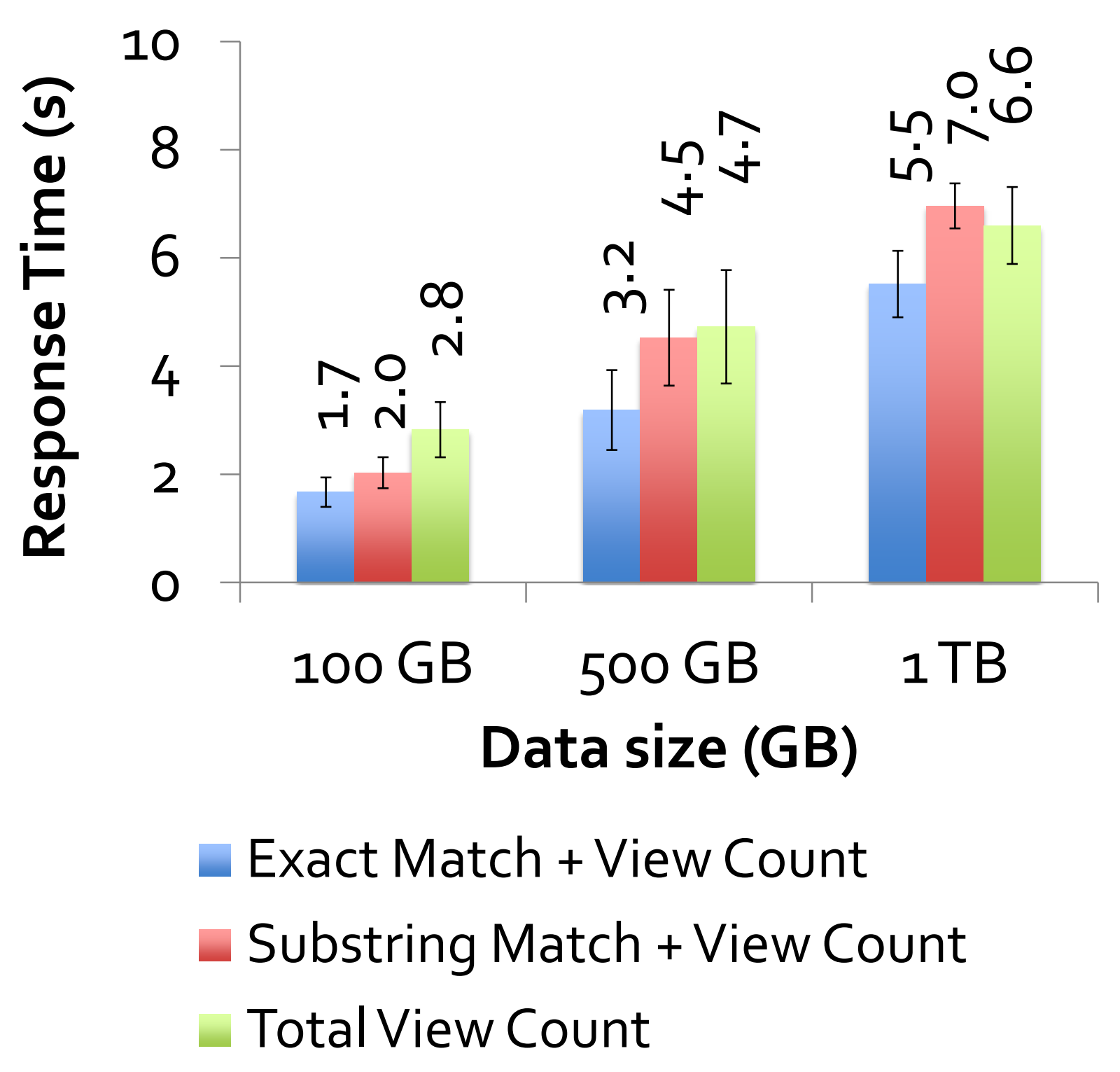
Logistic Regression



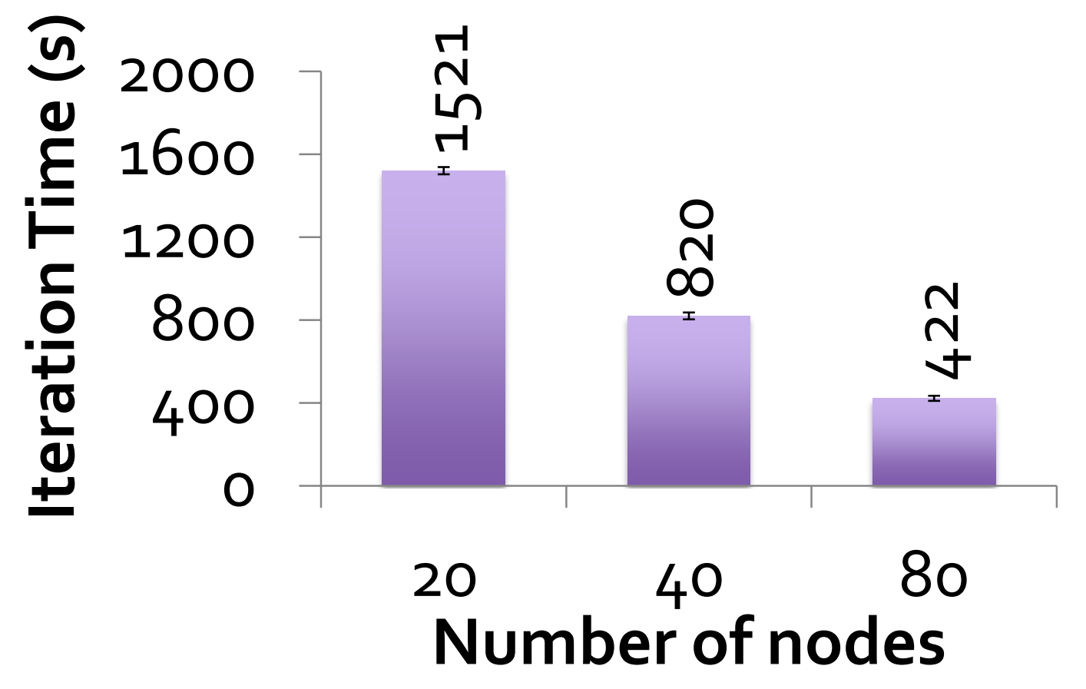
Twitter Spam Classifier



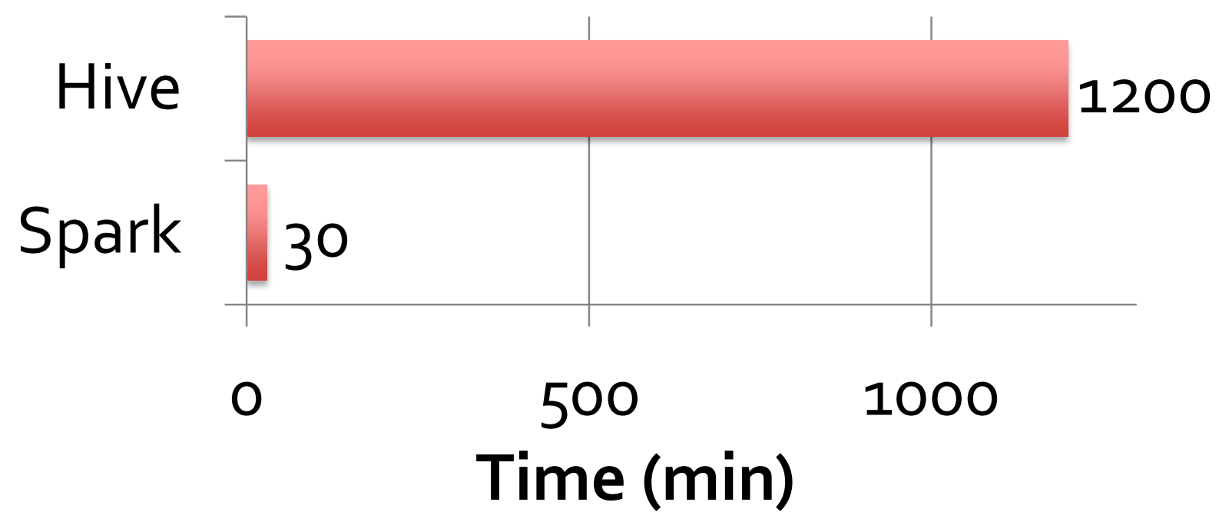
Interactive Queries



City Traffic Estimation



Conviva GeoReport


Hive
Spark
Time (min)