Mesos

A Platform for Fine-Grained Resource Sharing in Data Centers

Benjamin Hindman, Andy Konwinski, Matei Zaharia,
Ali Ghodsi, Anthony D. Joseph, Randy Katz,
Scott Shenker, Ion Stoica
UC Berkeley

Challenges in Hadoop Cluster Management

Isolation (both fault and resource)

- » E.g. if co-locating production and experimental jobs

Testing and deploying upgrades

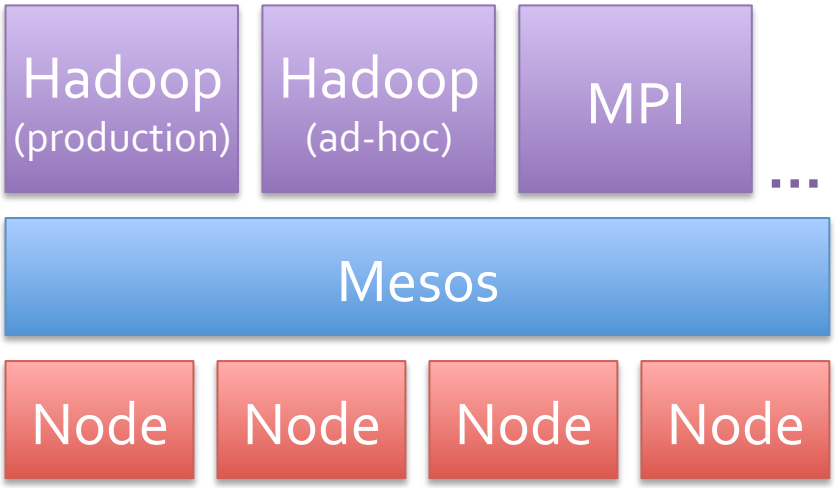
JobTracker scalability (in larger clusters)

Running non-Hadoop jobs

- » E.g. MPI, streaming MapReduce, etc

Mesos

Mesos is a cluster resource manager over which multiple instances of Hadoop and other distributed applications can run



What's Different About It?

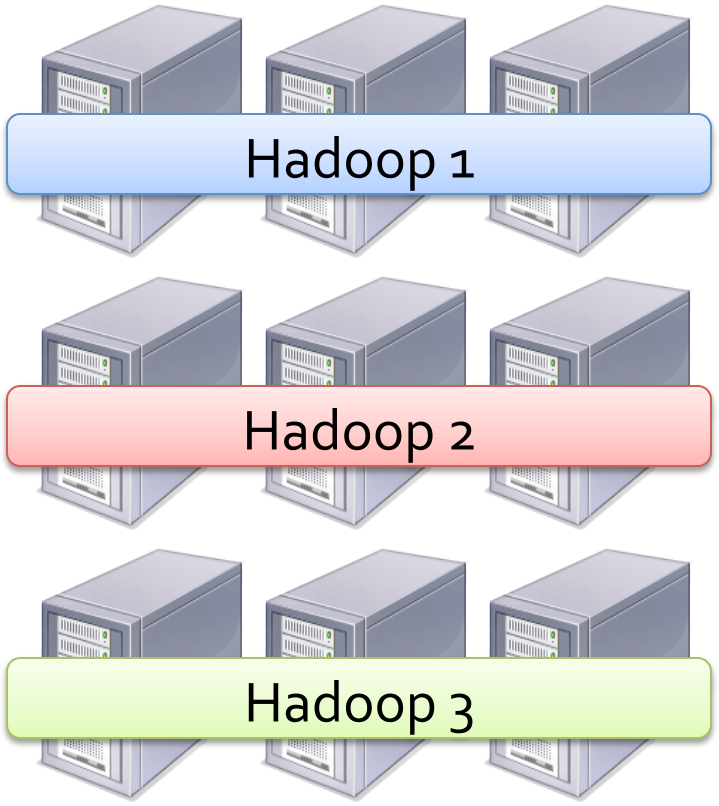
Other resource managers exist today, including
- » Hadoop on Demand
- » Batch schedulers (e.g. Torque)
- » VM schedulers (e.g. Eucalyptus)

However, have 2 problems with Hadoop-like workloads:
- » **Data locality** compromised due to static partitioning of nodes
- » **Utilization** hurt because jobs hold nodes for their full duration

Mesos addresses these through two features:
- » **Fine-grained sharing** at the level of tasks
- » **Two-level scheduling model** where jobs control placement

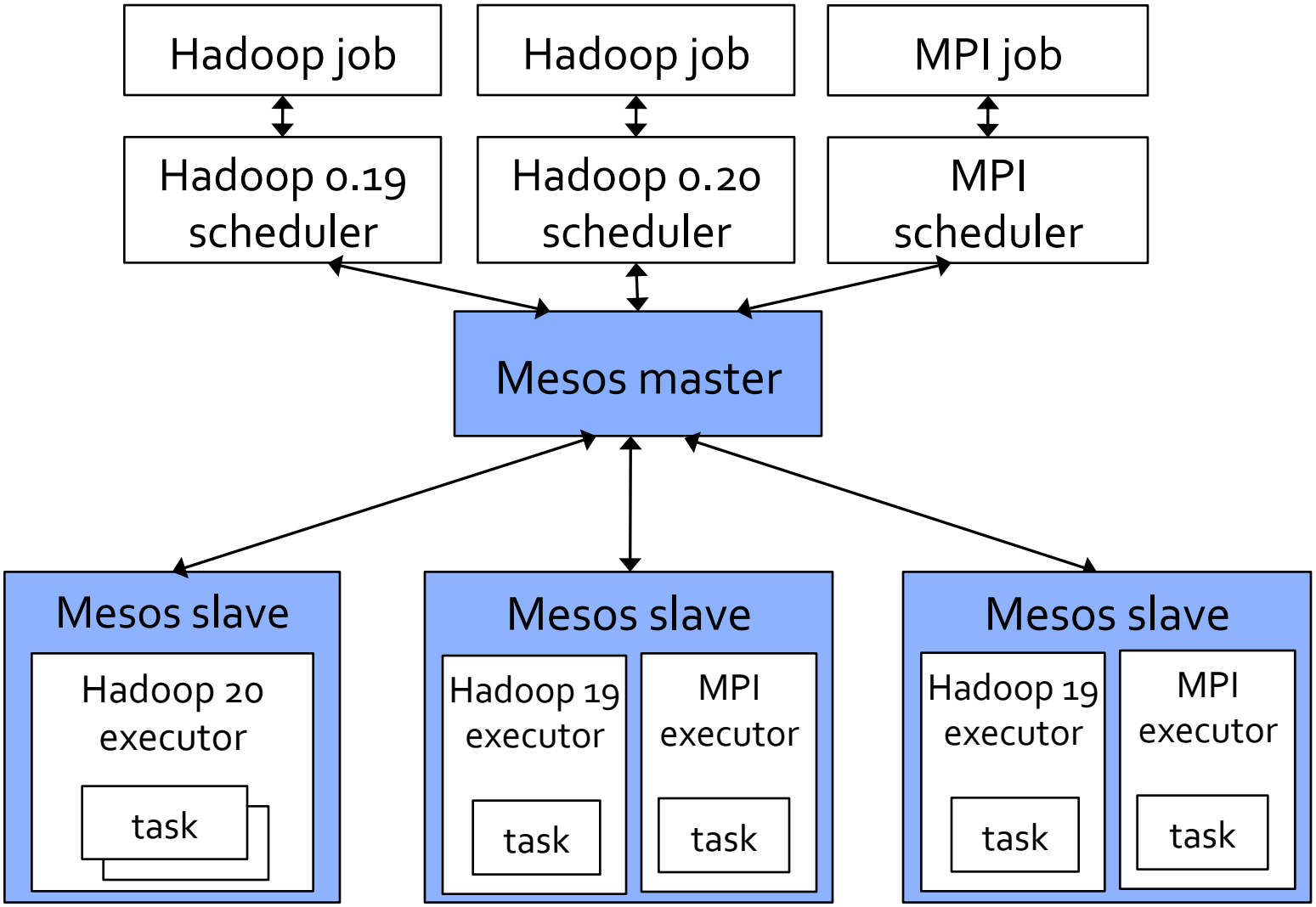
Fine-Grained Sharing

Coarse-Grained (HOD, etc)



Fine-Grained (Mesos)

Mesos Architecture



Design Goals

Scalability (to 10,000's of nodes)

Robustness (even to master failure)

Flexibility (support wide variety of frameworks)

Resulting design: simple, minimal core that pushes resource selection logic to frameworks

Other Features

Master fault tolerance using ZooKeeper

Resource isolation using Linux Containers

- » Isolate CPU and memory between tasks on each node
- » In newest kernels, can isolate network & disk IO too

Web UI for viewing cluster state

Deploy scripts for private clusters and EC2

Mesos Status

Prototype in 10000 lines of C++

Ported frameworks:

» Hadoop (0.20.2), MPI (MPICH2), Torque

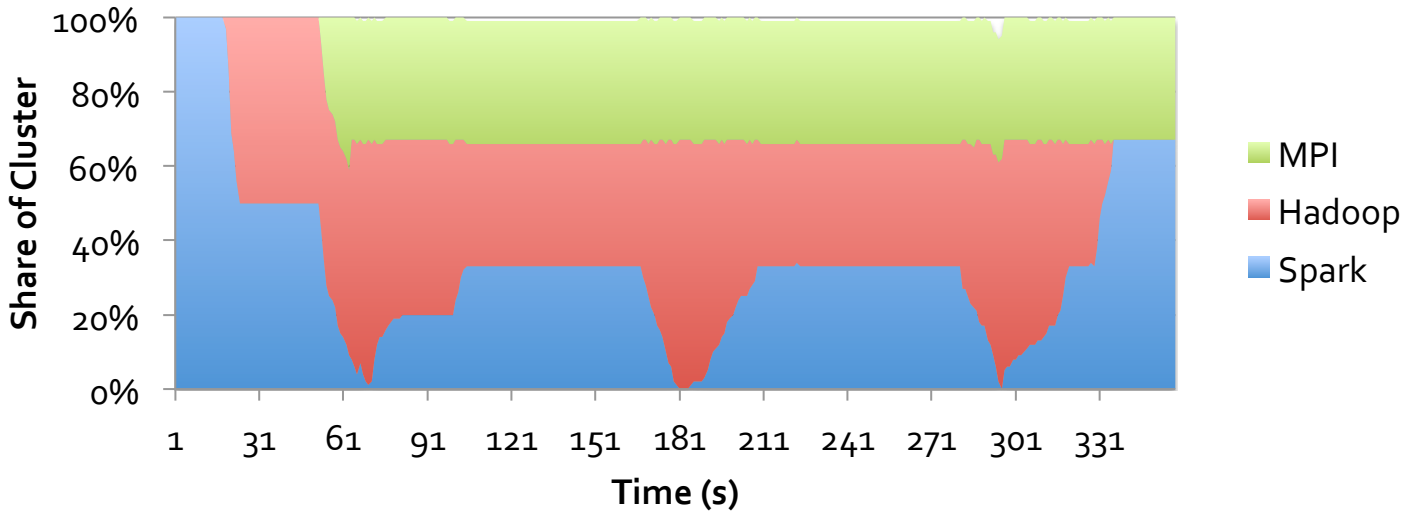
New frameworks:

» Spark, Scala framework for iterative & interactive jobs

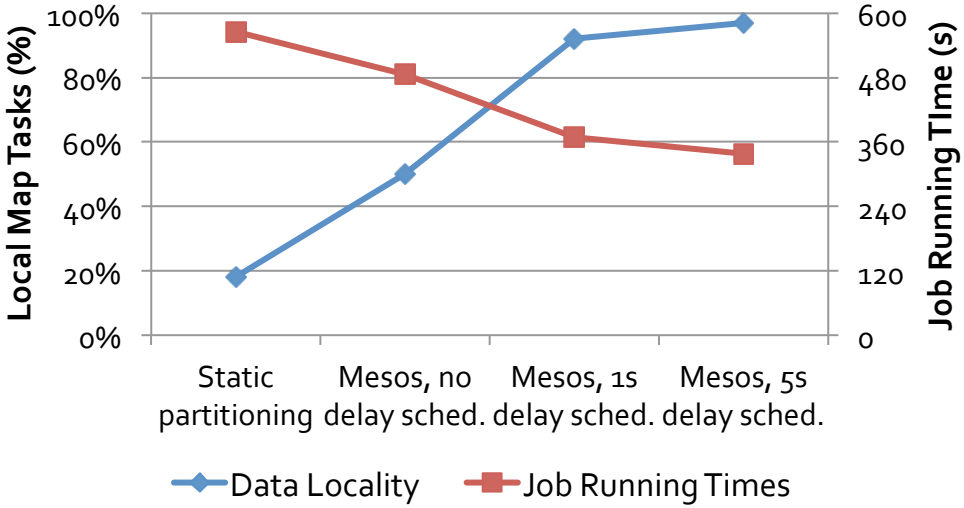
Test deployments at Twitter and Facebook

Results

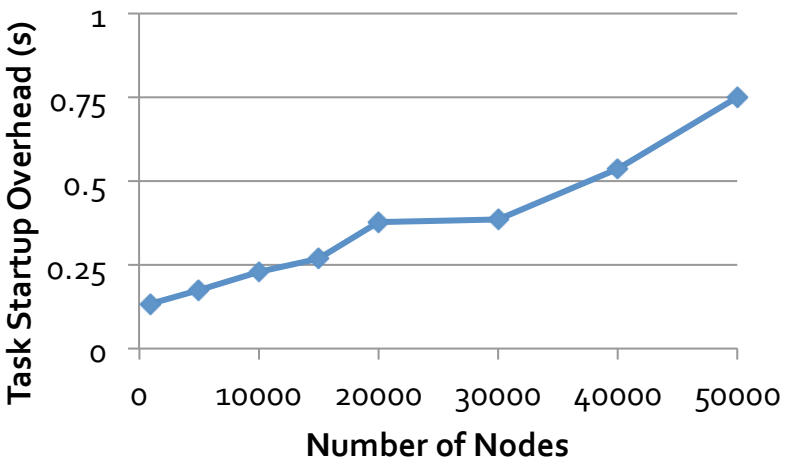
Dynamic Resource Sharing



Data Locality



Scalability



Spark

A framework for iterative and interactive cluster computing

Matei Zaharia, Mosharaf Chowdhury,
Michael Franklin, Scott Shenker, Ion Stoica

Spark Goals

Support iterative applications

- » Common in machine learning but problematic for MapReduce, Dryad, etc

Retain MapReduce's fault tolerance & scalability

Experiment with programmability

- » Integrate into Scala programming language
- » Support interactive use from Scala interpreter

Key Abstraction

Resilient Distributed Datasets (RDDs)

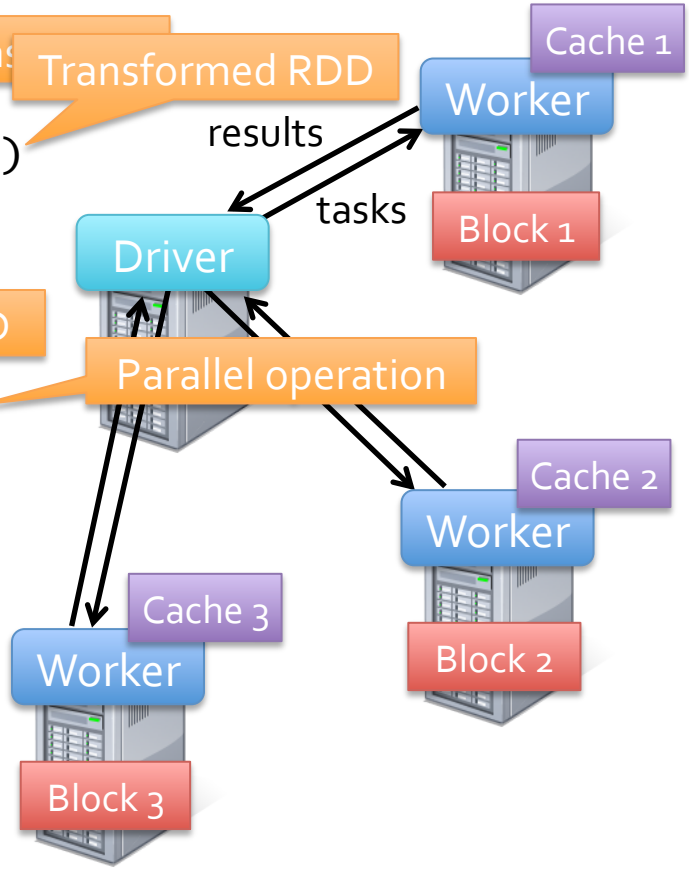
- » Collections of elements distributed across cluster that can persist across parallel operations
- » Can be stored in memory, on disk, etc
- » Can be transformed with map, filter, etc
- » Automatically rebuilt on failure

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

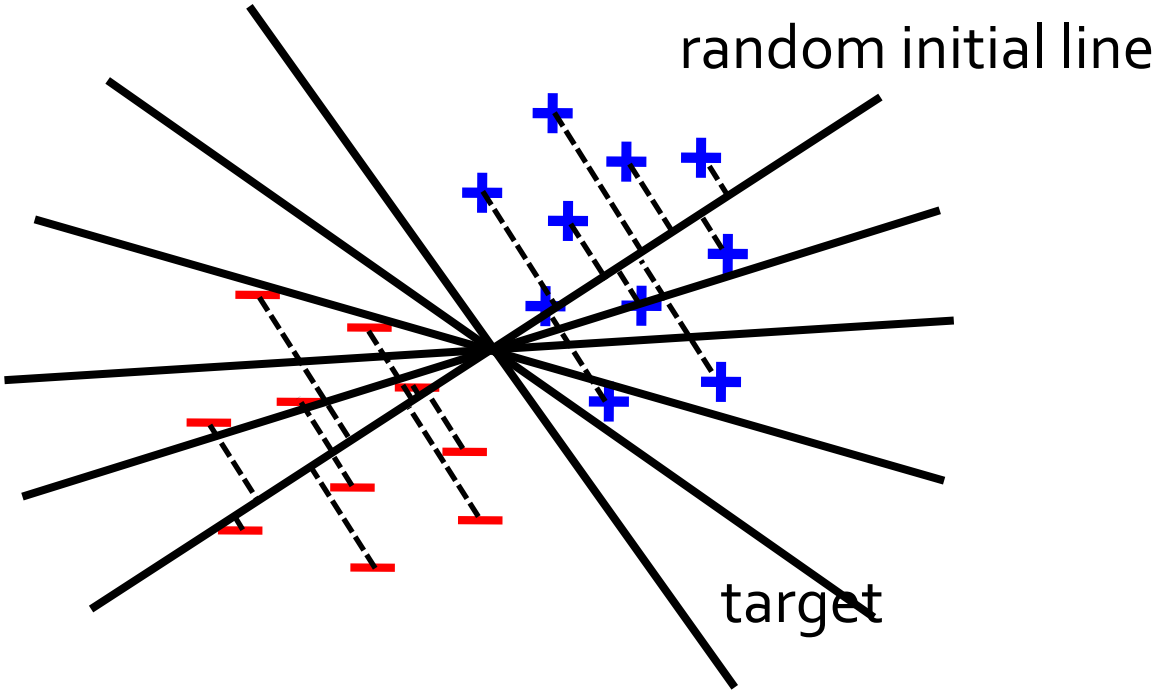
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(_.startsWith("ERROR"))
messages = errors.map(_.split('\t')(2))
cachedMsgs = messages.cache()

cachedMsgs.filter(_.contains("foo")).count
cachedMsgs.filter(_.contains("bar")).count
. . .
```



Example: Logistic Regression

Goal: find best line separating two sets of points



Logistic Regression Code

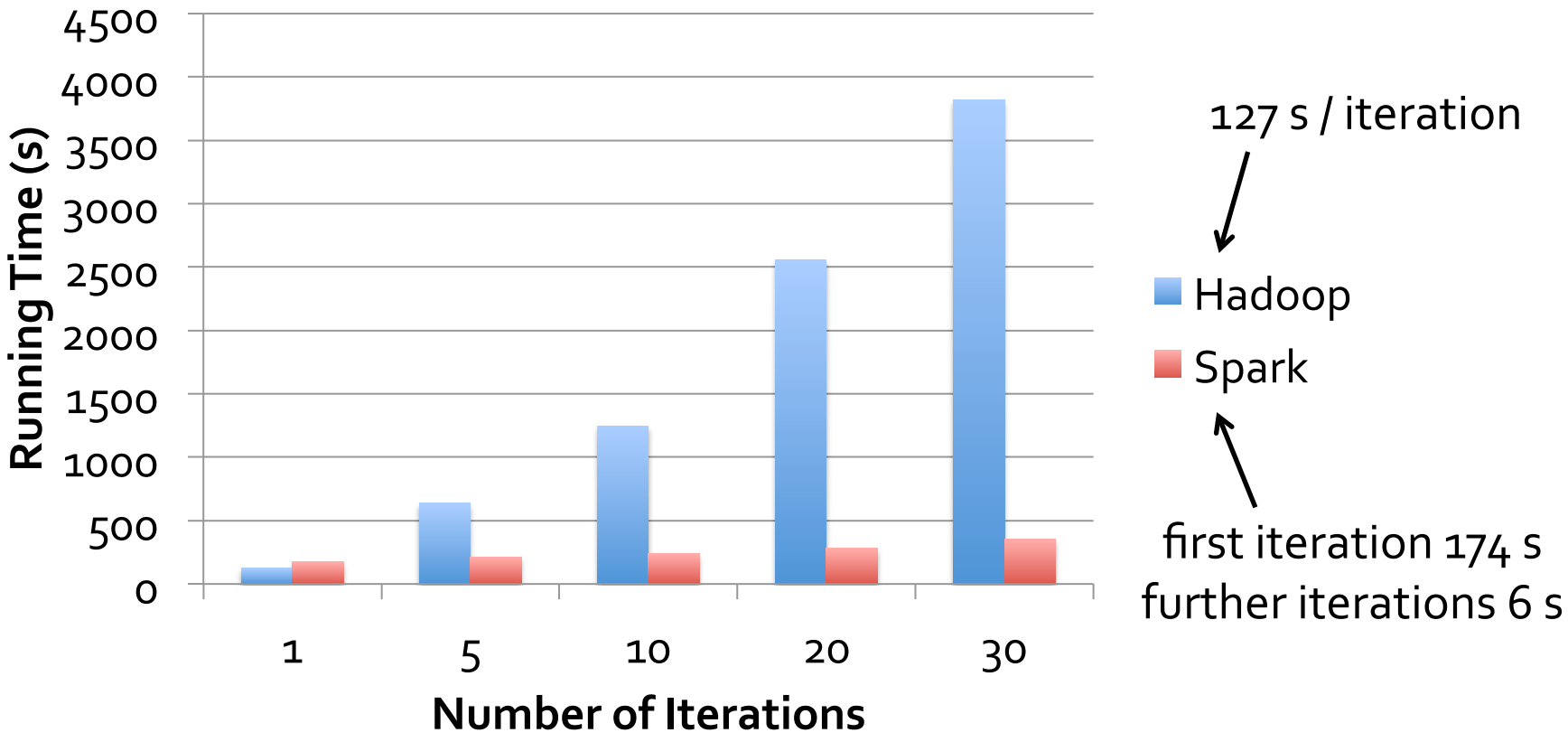
```
val data = spark.textFile(...).map(readPoint).cache()

var w = Vector.random(D)

for (i <- 1 to ITERATIONS) {
  val gradient = data.map(p => {
    val scale = (1/(1+exp(-p.y*(w dot p.x))) - 1) * p.y
    scale * p.x
  }).reduce(_ + _)
  w -= gradient
}

println("Final w: " + w)
```

Logistic Regression Performance



Spark Demo

Conclusion

Mesos provides a stable platform to multiplex resources among diverse cluster applications

Spark is a new cluster programming framework for iterative & interactive jobs enabled by Mesos

Both are open-source (but in very early alpha!)

http://github.com/mesos

http://mesos.berkeley.edu