Beating State-of-the-art By -10000%

Reynold Xin, AMPLab, UC Berkeley

with help from
Joseph Gonzalez, Josh Rosen, Matei Zaharia,
Michael Franklin, Scott Shenker, Ion Stoica

NOT A TYPO

Beating State-of-the-art By -10000%

Reynold Xin, AMPLab, UC Berkeley

with help from
Joseph Gonzalez, Josh Rosen, Matei Zaharia,
Michael Franklin, Scott Shenker, Ion Stoica

MapReduce

deterministic, idempotent tasks

fault-tolerance
elasticity
resource sharing

“The bar for open source software
is at historical low.”

“The bar for open source software is at historical low.”

i.e. “This is the right time to do grad school.”

iterative machine learning

OLAP

strong temporal locality

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

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

YES

Spark

How to do in-memory computation efficiently in a fault-tolerant way?

Shark

How to do SQL query processing
efficiently in “MapReduce” style

SQL on top of Spark
Hive compatible
(UDF, Type, InputFormat, Metadata)

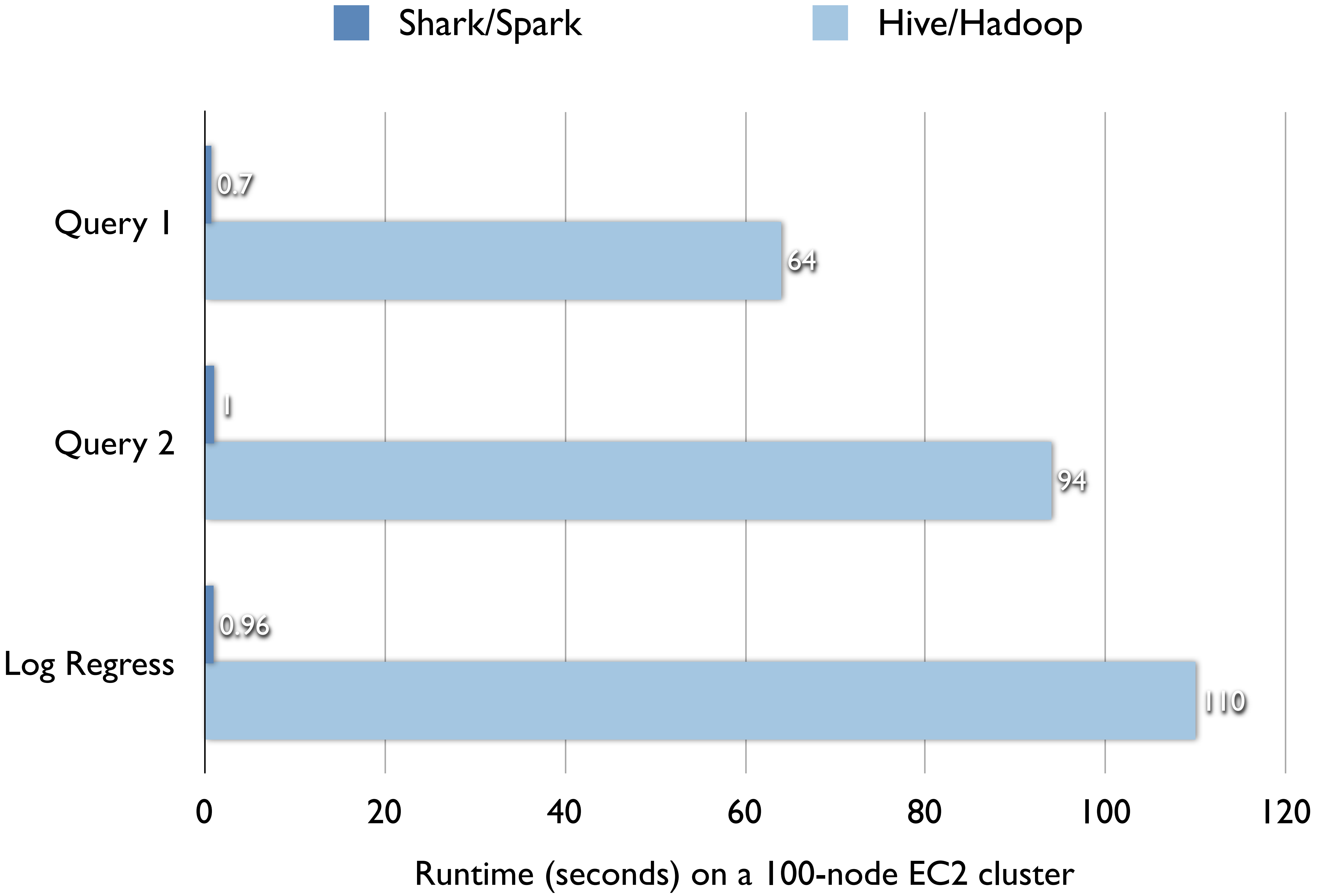
“You need to beat Hadoop by at least 100X to publish a paper in 2013.”

“You need to beat Hadoop by at least 100X to publish a paper in 2013.”

i.e. “You should’ve come to grad school 2 years earlier.”

Shark

in-memory columnar store
dynamic query re-optimization
and a lot of engineering...


Shark/Spark
Hive/Hadoop
Query 1
0.7
64
Query 2
1
94
Log Regress
0.96
110
0
20
40
60
80
100
120
Runtime (seconds) on a 100-node EC2 cluster

☑ iterative machine learning

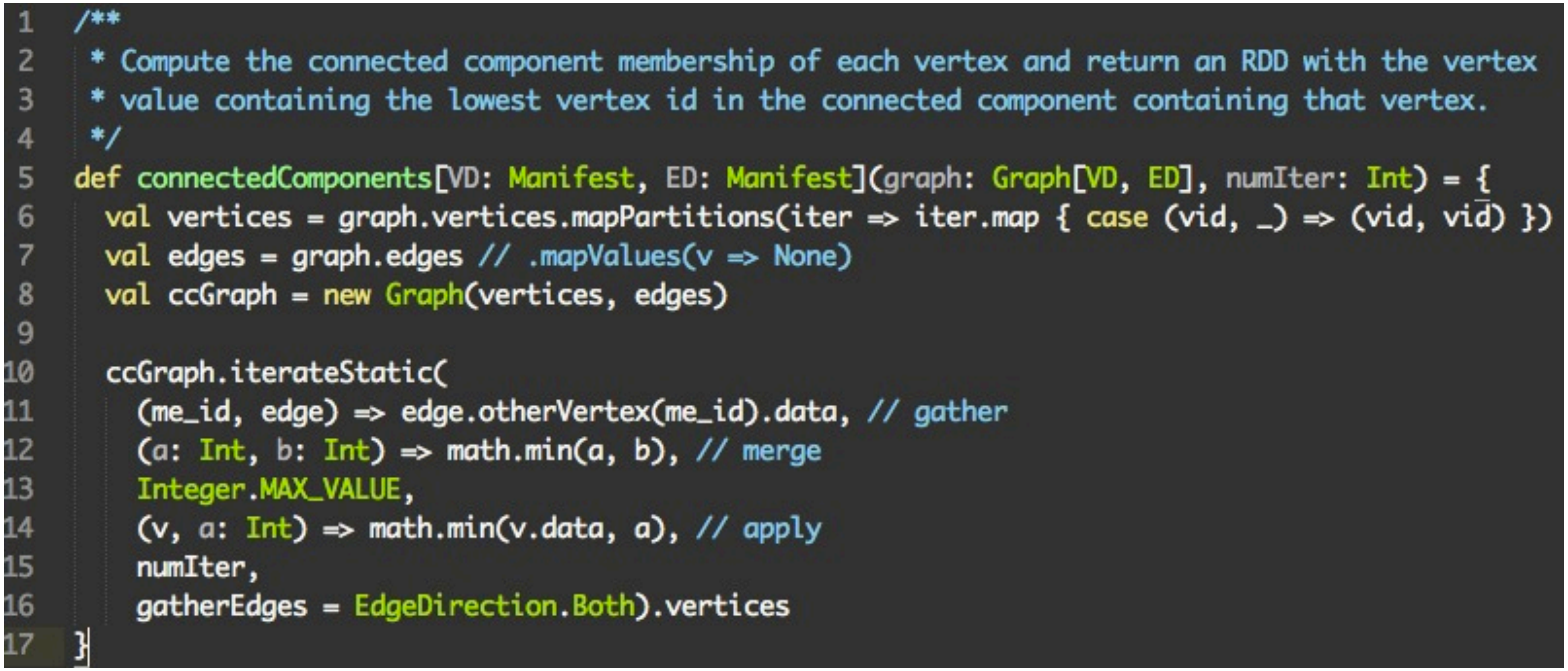
☑ SQL query processing

☑ iterative machine learning

☑ SQL query processing

graph computation

GraphLab on Spark



```
/**
 * Compute the connected component membership of each vertex and return an RDD with the vertex
 * value containing the lowest vertex id in the connected component containing that vertex.
 */
def connectedComponents[VD: Manifest, ED: Manifest](graph: Graph[VD, ED], numIter: Int) = {
  val vertices = graph.vertices.mapPartitions(iter => iter.map { case (vid, _) => (vid, vid) })
  val edges = graph.edges // .mapValues(v => None)
  val ccGraph = new Graph(vertices, edges)

  ccGraph.iterateStatic(
    (me_id, edge) => edge.otherVertex(me_id).data, // gather
    (a: Int, b: Int) => math.min(a, b), // merge
    Integer.MAX_VALUE,
    (v, a: Int) => math.min(v.data, a), // apply
    numIter,
    gatherEdges = EdgeDirection.Both).vertices
}
```

I spent a day pair-programming
with Joey Gonzalez
and improved performance by 10X.

Not bad for a day of work!

I spent a day pair-programming with Joey Gonzalez and improved performance by 10X.

but I later found out that it is still 10X slower than the latest version of GraphLab :(

A lot of open questions for fault-tolerant, distributed graph computation.

“MapReduce”?
Data partitioning?
Fault-tolerance?
Asynchrony?

iterative machine learning
www.spark-project.org

SQL query processing
shark.cs.berkeley.edu

graph computation
www.wait-another-year.com