MAPR

Apache Spark

Keys Botzum

Senior Principal Technologist, MapR Technologies

June 2014

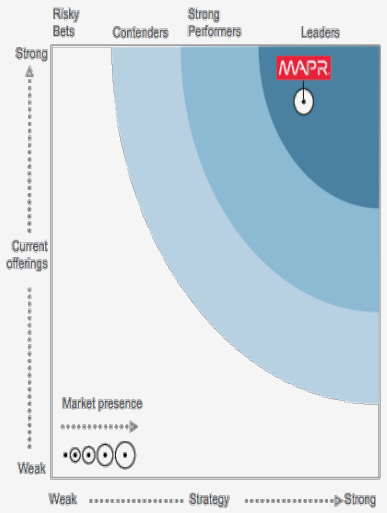
Agenda

- MapReduce
- Apache Spark
- How Spark Works
- Fault Tolerance and Performance
- Examples
- Spark and More

MapR: Best Product, Best Business & Best Customers

Top Ranked

FORRESTER®



Exponential Growth

3X bookings Q1 '13 – Q1 '14

90% software licenses

80% of accounts expand 3X

<1% lifetime churn

>$1B in incremental revenue generated by 1 customer

500+ Customers

LIVE NATION ENTERTAINMENT

hp

beatsMUSIC

SAMSUNG

AADHAAR

CISCO

rubicon PROJECT

comScore

Cloud Leaders

Google

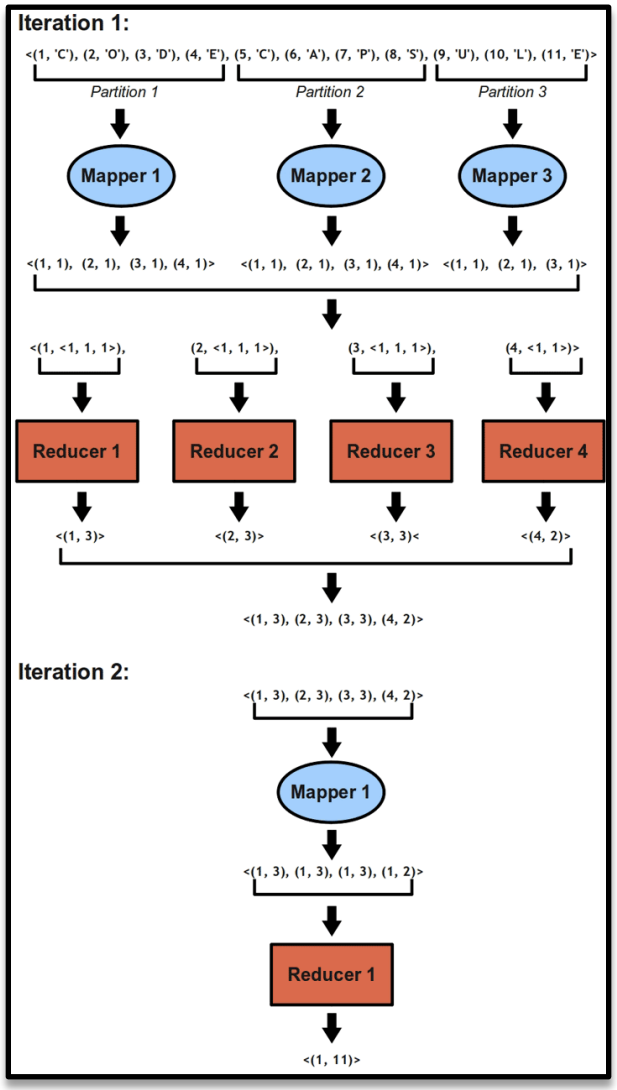
amazon web services™

Review: MapReduce

MapReduce: A Programming Model

- **MapReduce: Simplified Data Processing on Large Clusters** *(published 2004)*

- **Parallel and Distributed Algorithm:**
- Data Locality
- Fault Tolerance
- Linear Scalability

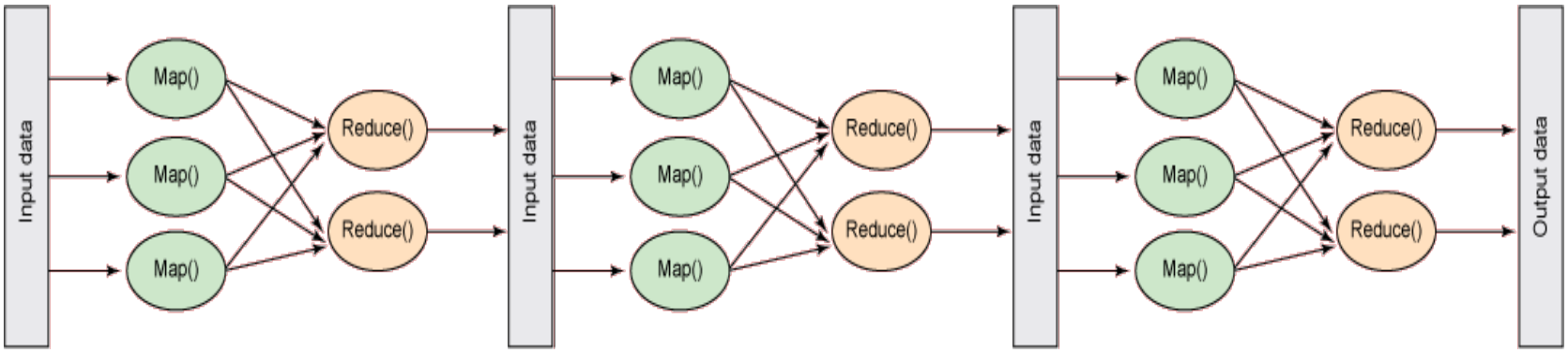


MapReduce Basics

- Assumes scalable distributed file system that shards data
- Map
  - Loading of the data and defining a set of keys
- Reduce
  - Collects the organized key-based data to process and output
- Performance can be tweaked based on known details of your source files and cluster shape (size, total number)

MapReduce Processing Model

- Define mappers
- Shuffling is automatic
- Define reducers
- For complex work, chain jobs together



MapReduce: The Good

- Built in fault tolerance
- Optimized IO path
- Scalable
- Developer focuses on Map/Reduce, not infrastructure
- simple? API

MapReduce: The Bad

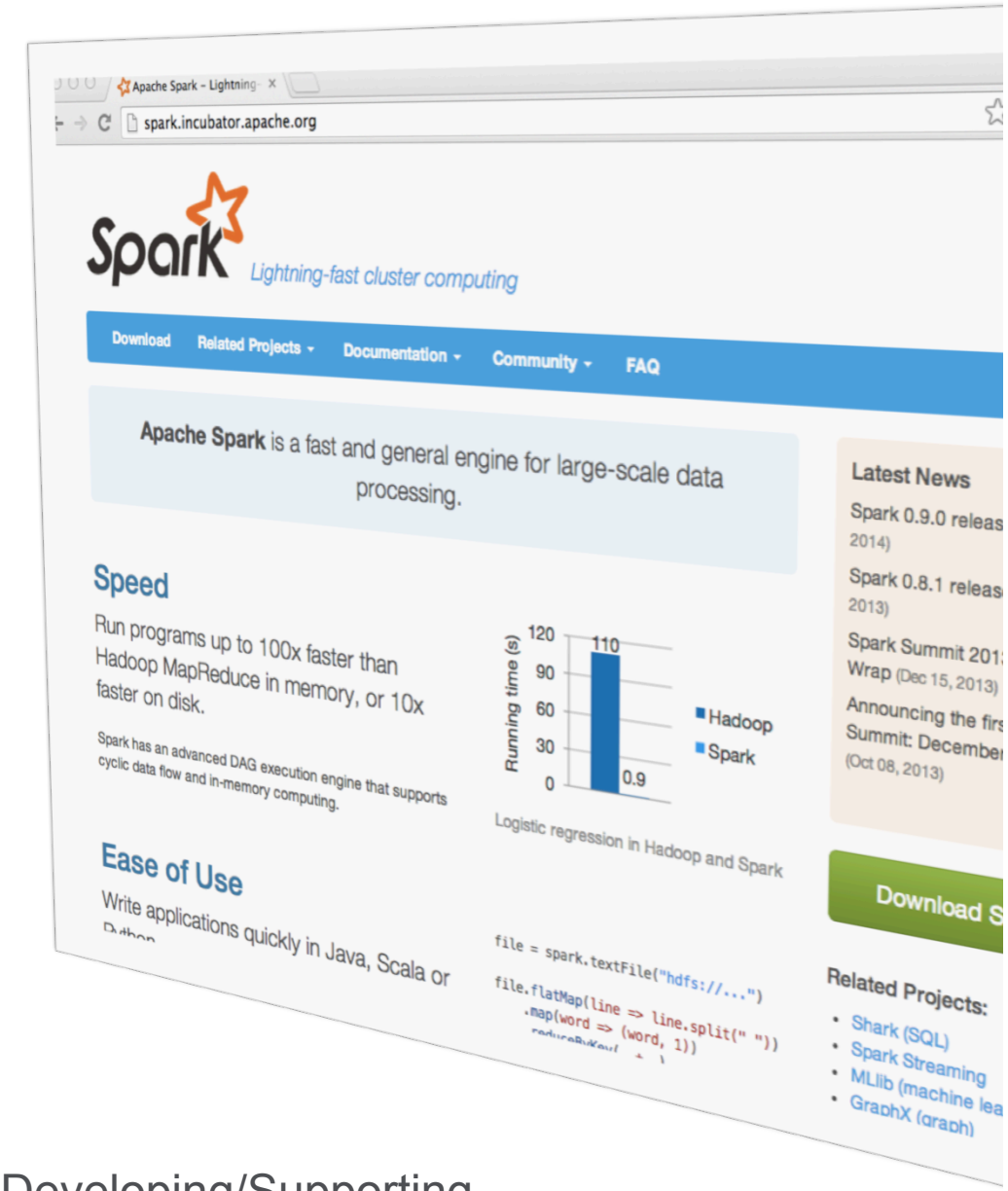
- Optimized for disk IO
  - Doesn't leverage memory well
  - Iterative algorithms go through disk IO path again and again
- Primitive API
  - Developer's have to build on very simple abstraction
  - Key/Value in/out
  - Even basic things like join require extensive code
- Result often many files that need to be combined appropriately

Apache Spark

Apache Spark

- Originally developed in 2009 in UC Berkeley's AMP Lab
- Fully open sourced in 2010 – now at Apache Software Foundation

- **spark.apache.org**
- **github.com/apache/spark**
- **user@spark.apache.org**



DATABRICKS - Commercial Vendor Developing/Supporting

Spark: Easy and Fast Big Data

- **Easy to Develop**
  - Rich APIs in Java, Scala, Python
  - Interactive shell

  2-5× less code

- **Fast to Run**
  - General execution graphs
  - In-memory storage

Resilient Distributed Datasets (RDD)

- Spark revolves around RDDs
- Fault-tolerant read only collection of elements that can be operated on in parallel
- Cached in memory or on disk

http://www.cs.berkeley.edu/~matei/papers/2012/nsdi_spark.pdf

RDD Operations - Expressive

- Transformations
  - Creation of a new RDD dataset from an existing
    - map, filter, distinct, union, sample, groupByKey, join, reduce, etc…
- Actions
  - Return a value after running a computation
    - collect, count, first, takeSample, foreach, etc…

Check the documentation for a complete list

http://spark.apache.org/docs/latest/scala-programming-guide.html#rdd-operations

Easy: Clean API

Write programs in terms of **transformations** on **distributed datasets**

- **Resilient Distributed Datasets**
- Collections of objects spread across a cluster, stored in RAM or on Disk
- Built through parallel transformations
- Automatically rebuilt on failure

- **Operations**
- Transformations (e.g. map, filter, groupBy)
- Actions (e.g. count, collect, save)

Easy: Expressive API

- `map`
- `reduce`

Easy: Expressive API

- map
- filter
- groupBy
- sort
- union
- join
- leftOuterJoin
- rightOuterJoin
- reduce
- count
- fold
- reduceByKey
- groupByKey
- cogroup
- cross
- zip

sample

take

first

partitionBy

mapWith

pipe

save ...

Easy: Example – Word Count

- **Hadoop MapReduce**

```
public static class WordCountMapClass extends MapReduceBase
  implements Mapper<LongWritable, Text, Text, IntWritable> {

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(LongWritable key, Text value,
                  OutputCollector<Text, IntWritable> output,
                  Reporter reporter) throws IOException {
    String line = value.toString();
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      output.collect(word, one);
    }
  }
}

public static class WorkdCountReduce extends MapReduceBase
  implements Reducer<Text, IntWritable, Text, IntWritable> {

  public void reduce(Text key, Iterator<IntWritable> values,
                     OutputCollector<Text, IntWritable> output,
                     Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
  }
}
```

- **Spark**

```
val spark = new SparkContext(master, appName, [sparkHome], [jars])
val file = spark.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

Easy: Example – Word Count

- **Hadoop MapReduce**

```
public static class WordCountMapClass extends MapReduceBase
  implements Mapper<LongWritable, Text, Text, IntWritable> {

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(LongWritable key, Text value,
                  OutputCollector<Text, IntWritable> output,
                  Reporter reporter) throws IOException {
    String line = value.toString();
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      output.collect(word, one);
    }
  }
}

public static class WorkdCountReduce extends MapReduceBase
  implements Reducer<Text, IntWritable, Text, IntWritable> {

  public void reduce(Text key, Iterator<IntWritable> values,
                     OutputCollector<Text, IntWritable> output,
                     Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
  }
}
```

- **Spark**

```
val spark = new SparkContext(master, appName, [sparkHome], [jars])
val file = spark.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

Easy: Works Well With Hadoop

- **Data Compatibility**
- Access your existing Hadoop Data
- Use the same data formats
- Adheres to data locality for efficient processing

- **Deployment Models**
- "Standalone" deployment
- YARN-based deployment
- Mesos-based deployment
- Deploy on existing Hadoop cluster or side-by-side

Easy: User-Driven Roadmap

- **Language support**
  - Improved Python support
  - SparkR
  - Java 8
  - Integrated Schema and SQL support in Spark's APIs
- **Better ML**
  - Sparse Data Support
  - Model Evaluation Framework
  - Performance Testing

Example: Logistic Regression

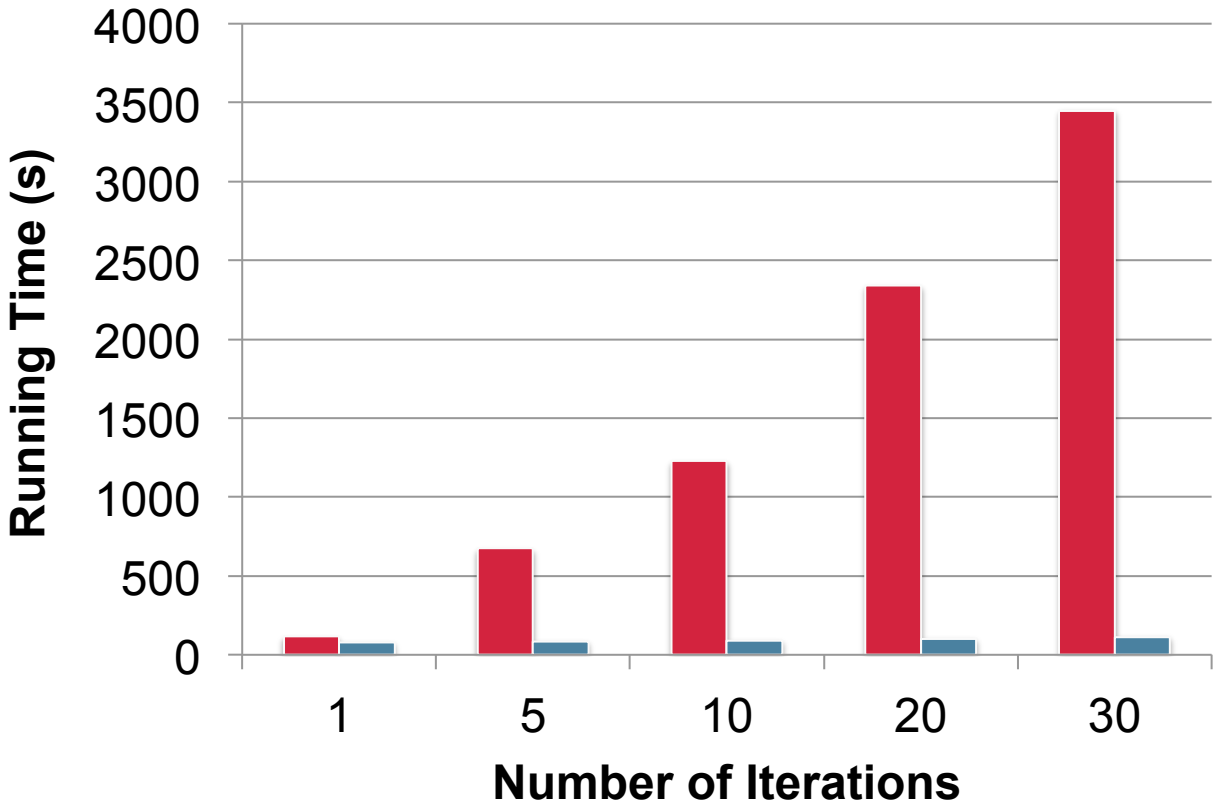
```
data = spark.textFile(...).map(readPoint).cache()

w = numpy.random.rand(D)

for i in range(iterations):
    gradient = data
     .map(lambda p: (1 / (1 + exp(-p.y * w.dot(p.x))))
           * p.y * p.x)
      .reduce(lambda x, y: x + y)
    w -= gradient

print “Final w: %s” % w
```

Fast: Logistic Regression Performance



110 s / iteration

Hadoop

Spark

first iteration 80 s
further iterations 1 s

Easy: Multi-language Support

Python

```
lines = sc.textFile(...)
lines.filter(lambda s: “ERROR” in s).count()
```

Scala

```
val lines = sc.textFile(...)
lines.filter(x => x.contains(“ERROR”)).count()
```

Java

```
JavaRDD<String> lines = sc.textFile(...);
lines.filter(new Function<String, Boolean>() {
  Boolean call(String s) {
    return s.contains(“error”);
  }
}).count();
```

Easy: Interactive Shell

Scala based shell

```
% /opt/mapr/spark/spark-0.9.1/bin/spark-shell

scala> val logs = sc.textFile("hdfs:///user/keys/logdata")
scala> logs.count()
…
res0: Long = 232681

scala> logs.filter(l => l.contains("ERROR")).count()
….
res1: Long = 205
```

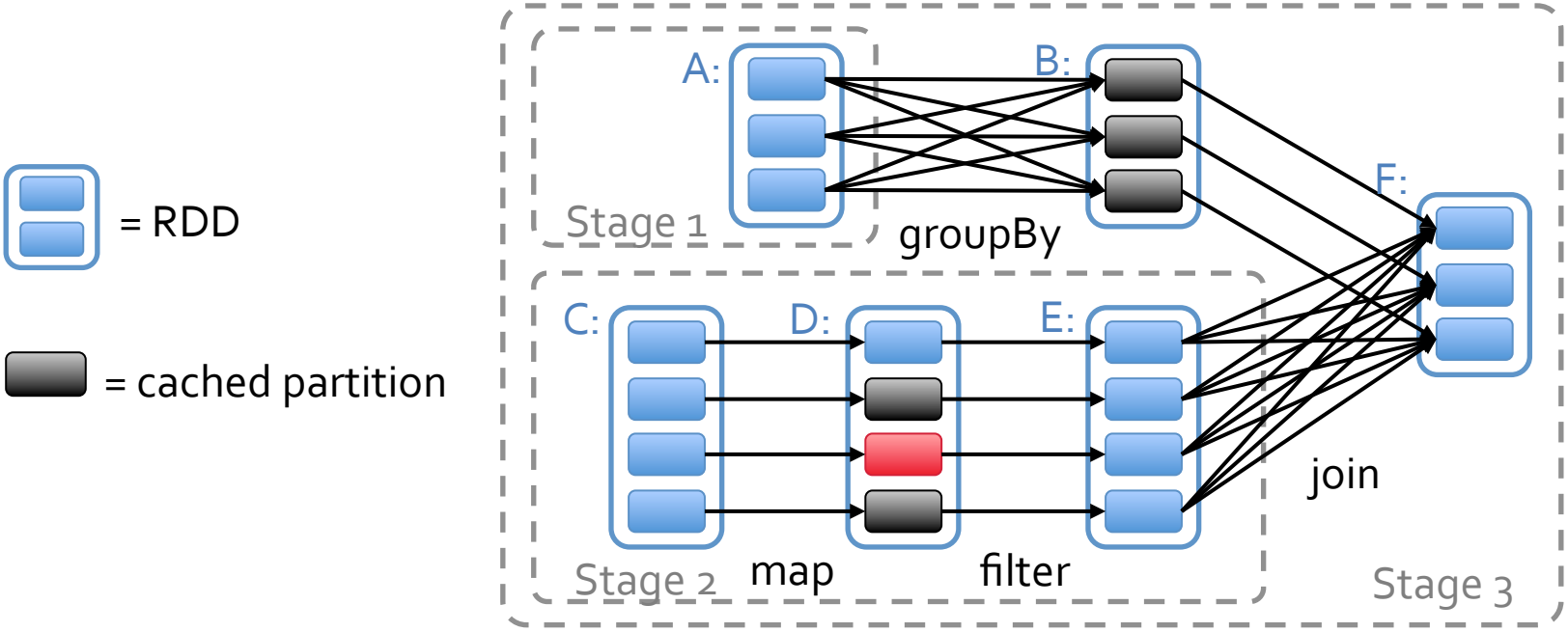
Python based shell as well - pyspark

Fault Tolerance and Performance

Fast: Using RAM, Operator Graphs

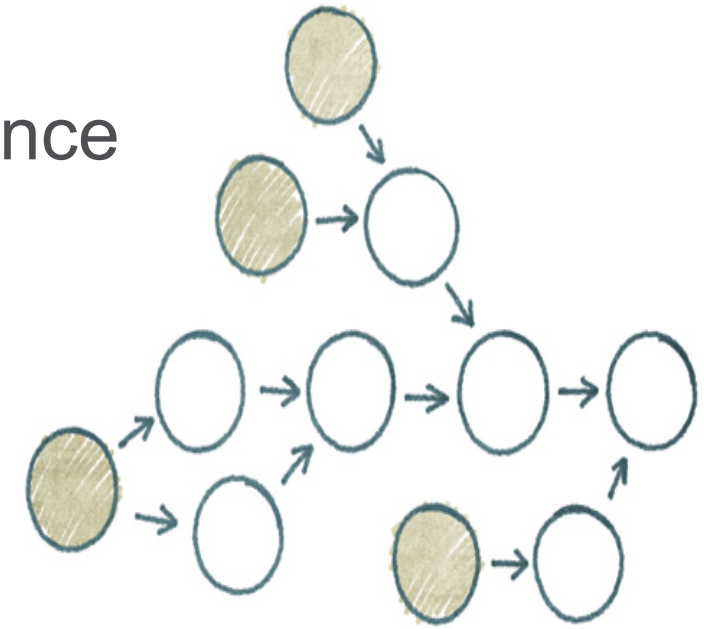
- **In-memory Caching**
- Data Partitions read from RAM instead of disk

- **Operator Graphs**
- Scheduling Optimizations
- Fault Tolerance



Directed Acylic Graph (DAG)

- Directed
  - Only in a single direction
- Acyclic
  - No looping
- This supports fault-tolerance



Easy: Fault Recovery

RDDs track *lineage* information that can be used to efficiently recompute lost data

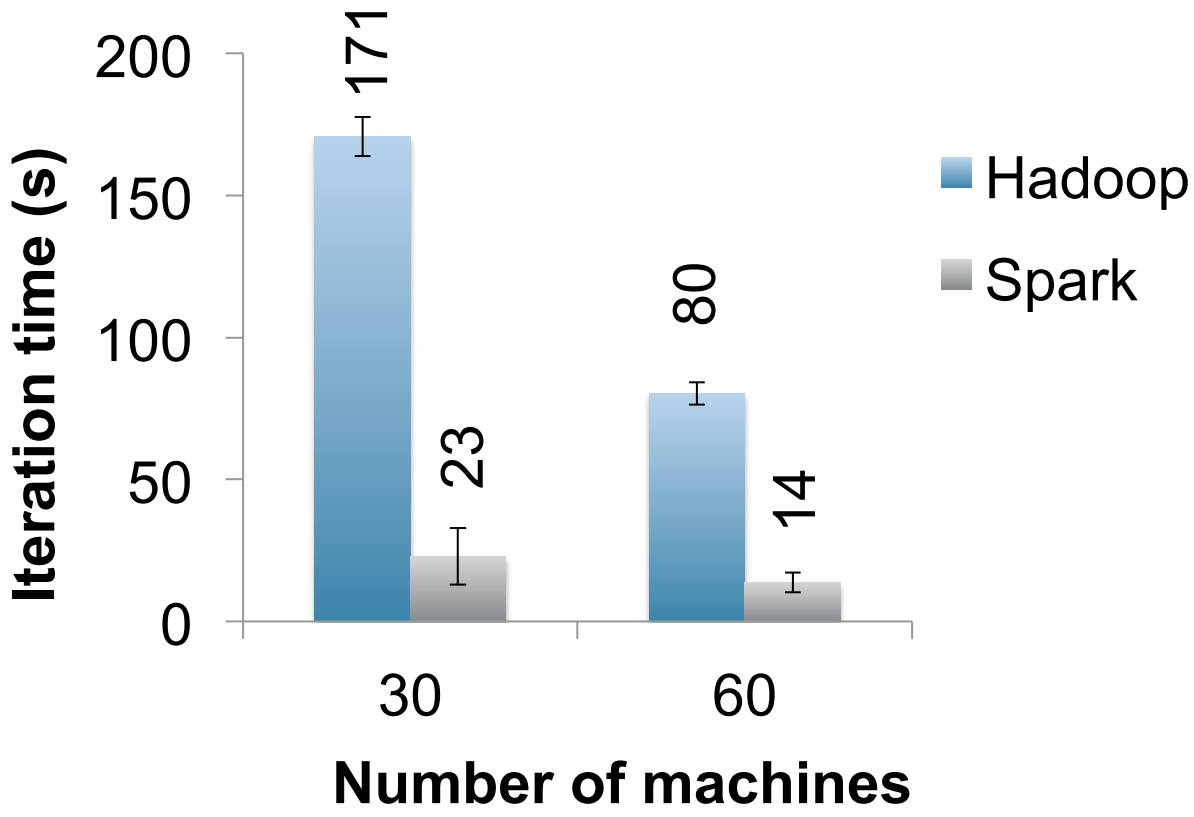
```
msgs = textFile.filter(lambda s: s.startsWith("ERROR"))
               .map(lambda s: s.split("\t")[2])
```

RDD Persistence / Caching

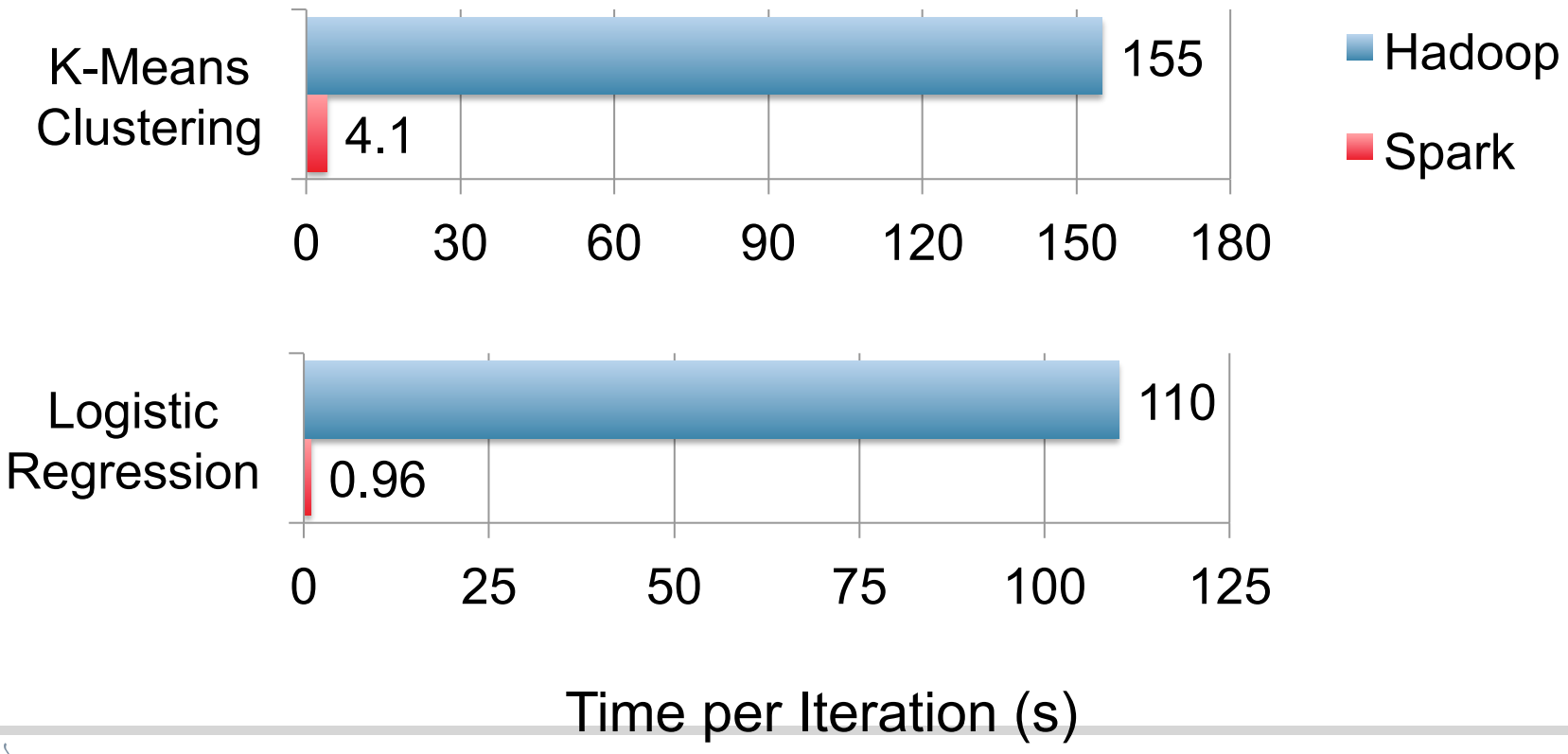
- Variety of storage levels
  - memory_only (default), memory_and_disk, etc…
- API Calls
  - persist(StorageLevel)
  - cache() – shorthand for persist(StorageLevel.MEMORY_ONLY)
- Considerations
  - Read from disk vs. recompute (memory_and_disk)
  - Total memory storage size (memory_only_ser)
  - Replicate to second node for faster fault recovery (memory_only_2)
    - Think about this option if supporting a time sensitive client

http://spark.apache.org/docs/latest/scala-programming-guide.html#rdd-persistence

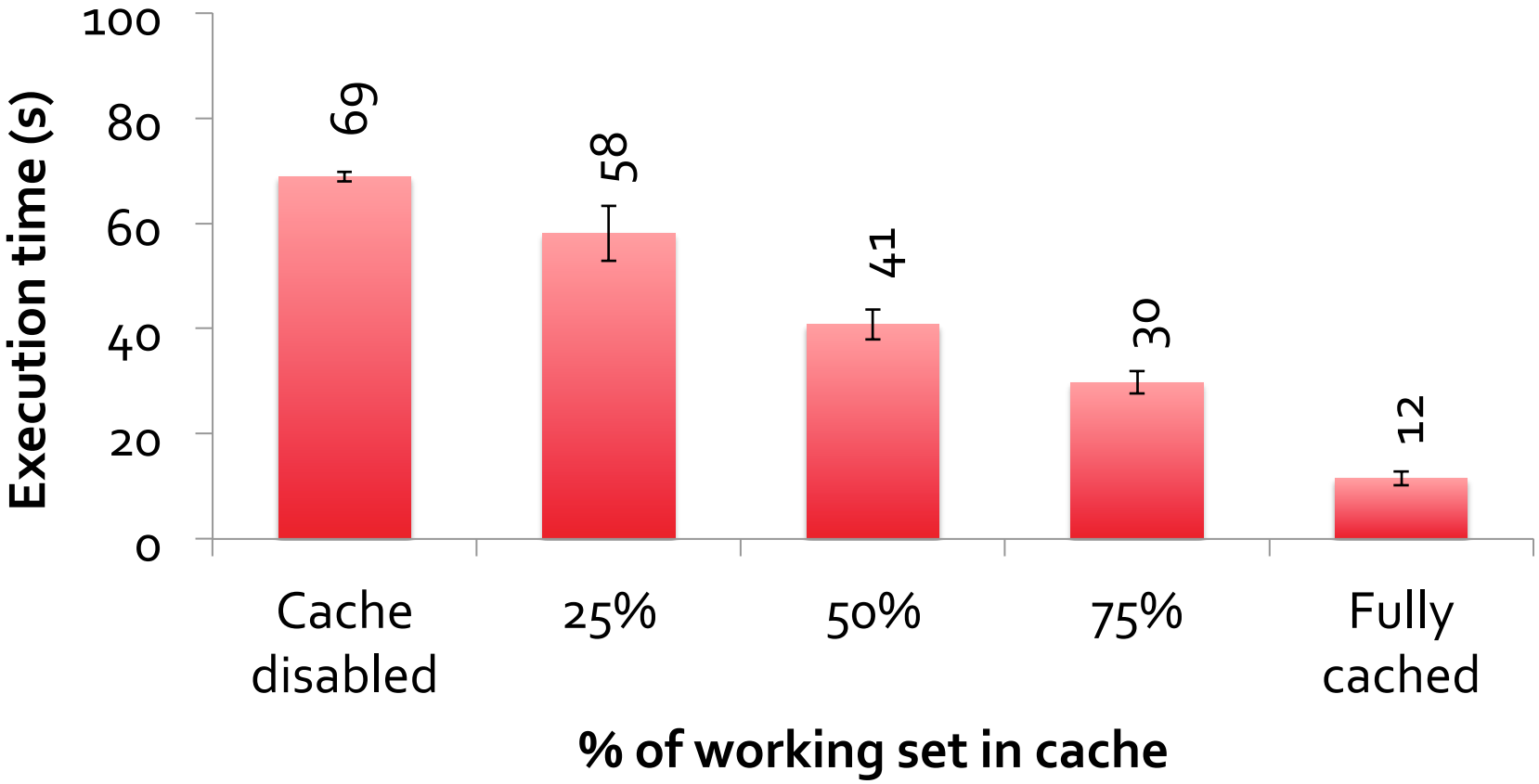
PageRank Performance



Other Iterative Algorithms

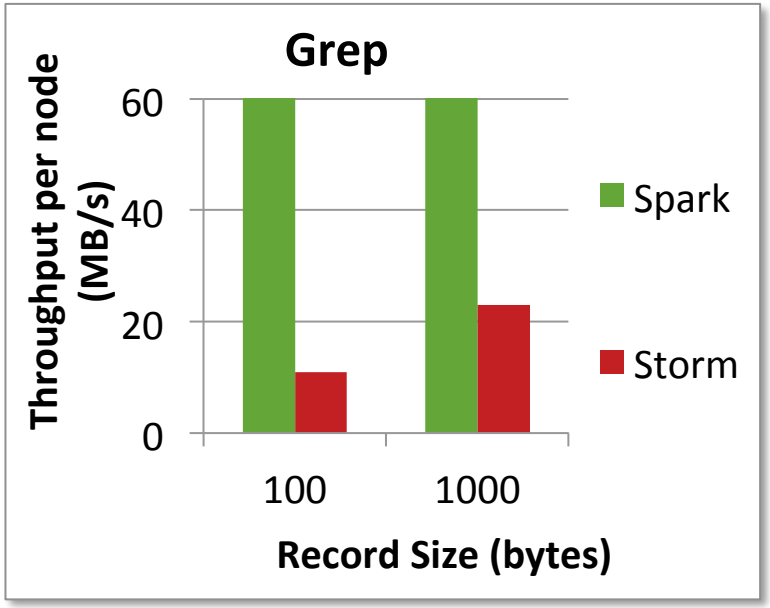


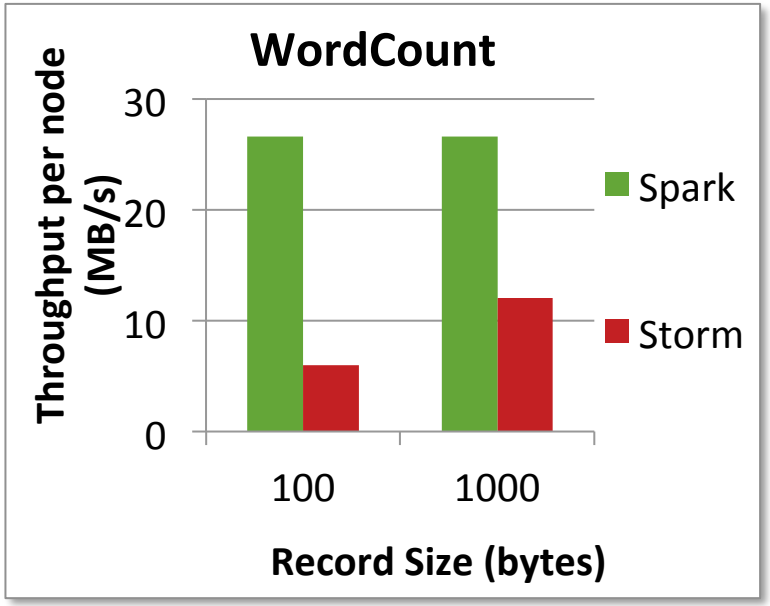
Fast: Scaling Down



Comparison to Storm

- Higher throughput than Storm
  - Spark Streaming: **670k** records/sec/node
  - Storm: **115k** records/sec/node
  - Commercial systems: **100-500k** records/sec/node

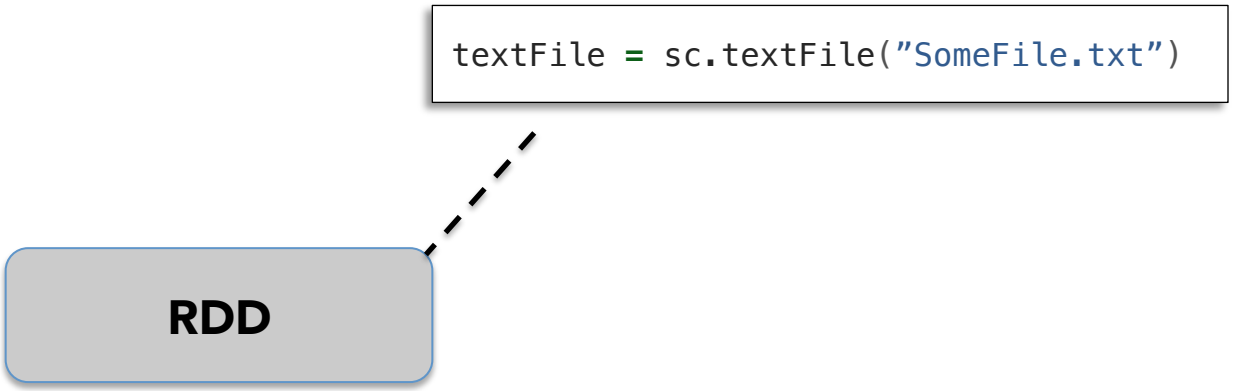




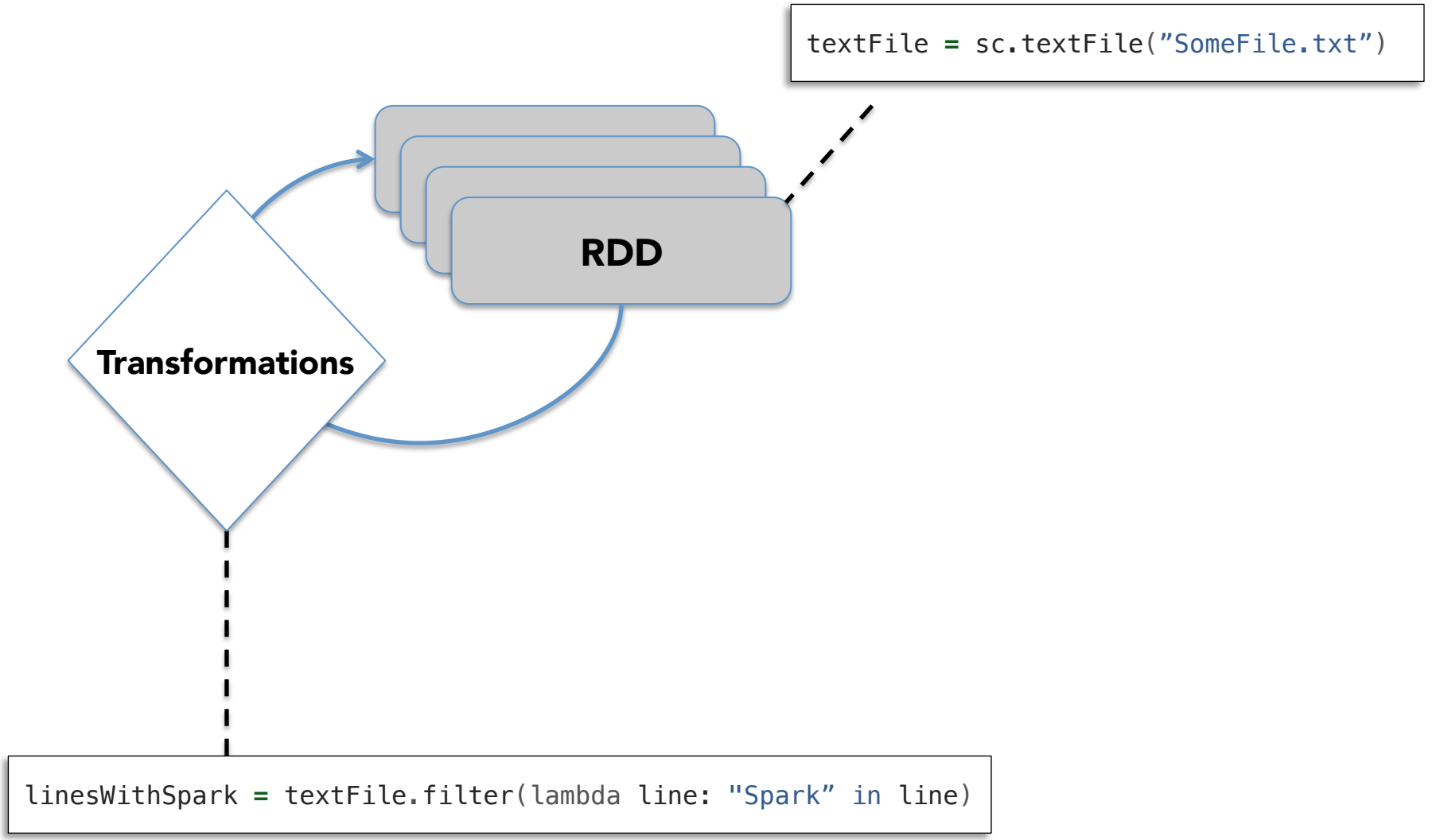
How Spark Works

Working With RDDs

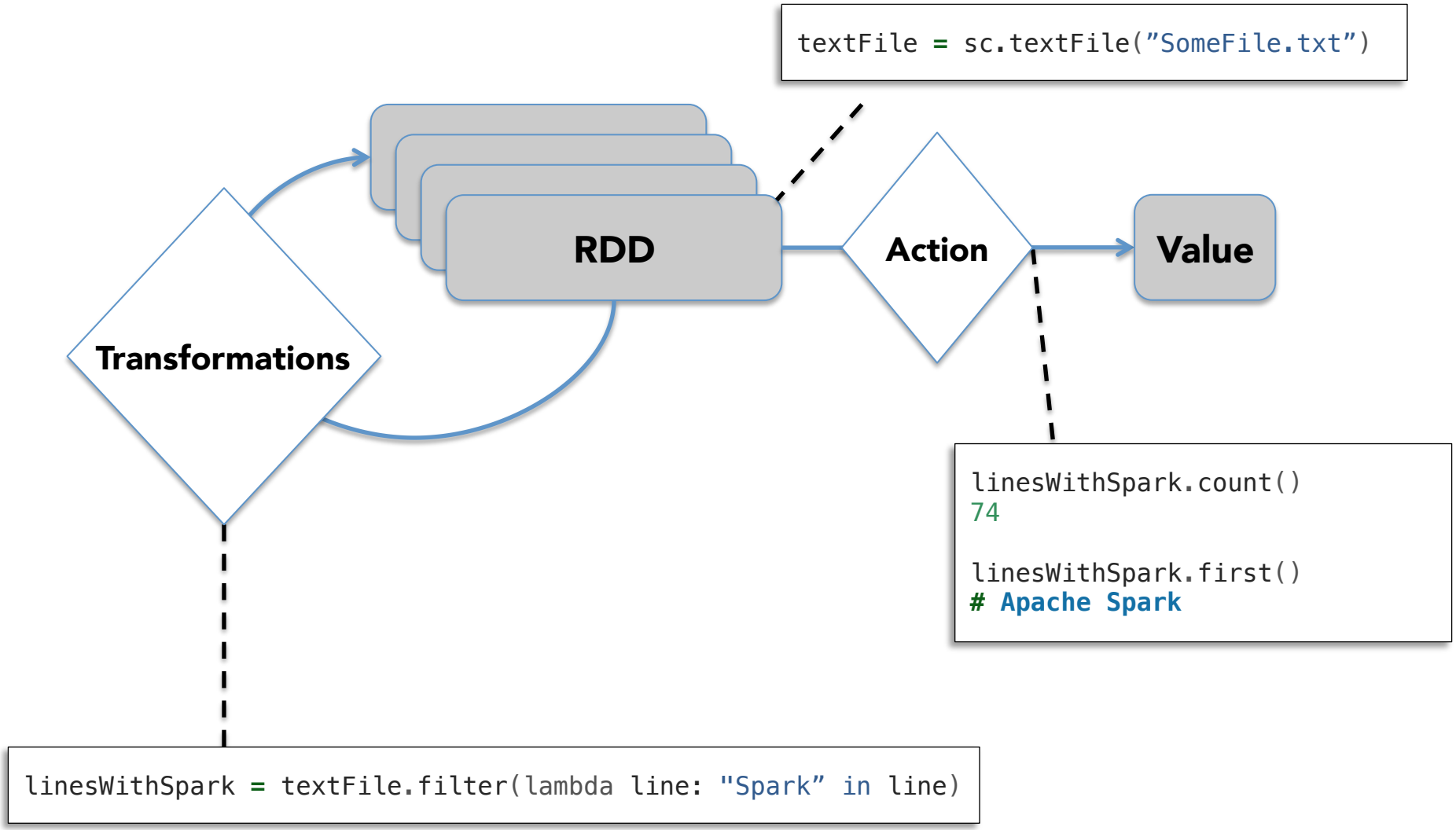
Working With RDDs



Working With RDDs



Working With RDDs



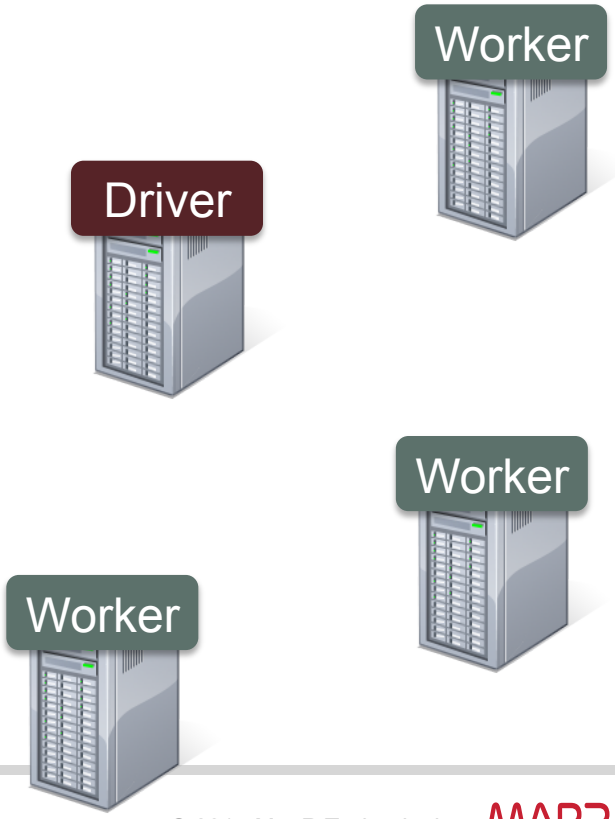
Example: Log Mining

Example: Log Mining

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

Example: Log Mining

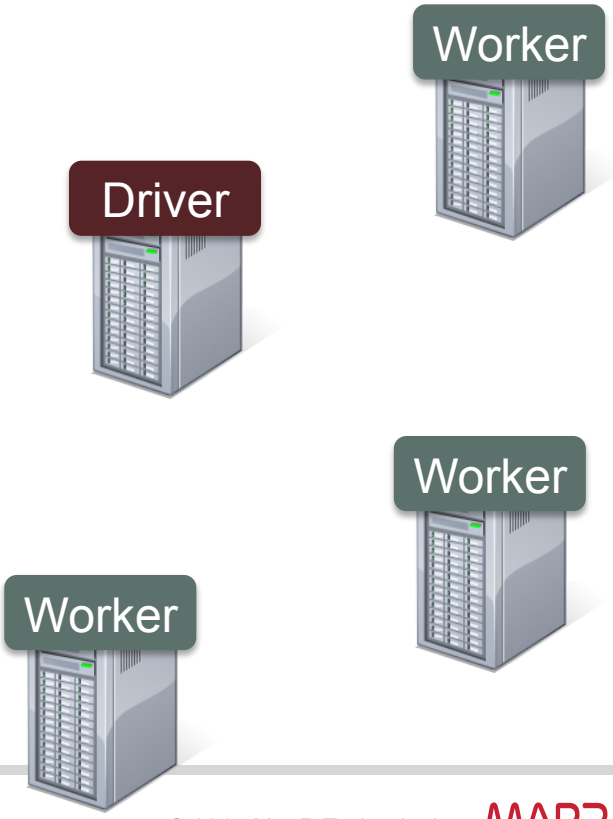
Load error messages from a log into memory, then interactively search for various patterns



Example: Log Mining

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

```
lines = spark.textFile(“hdfs://...”)
```

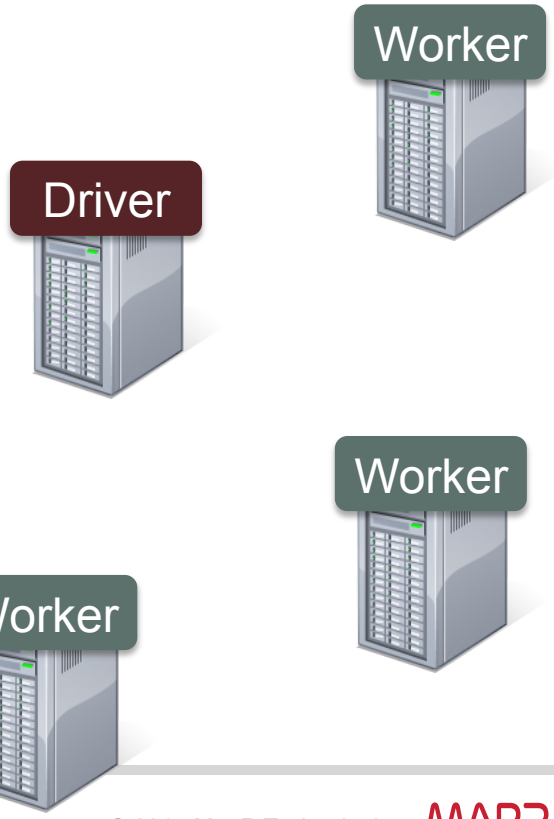


Example: Log Mining

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

Base RDD

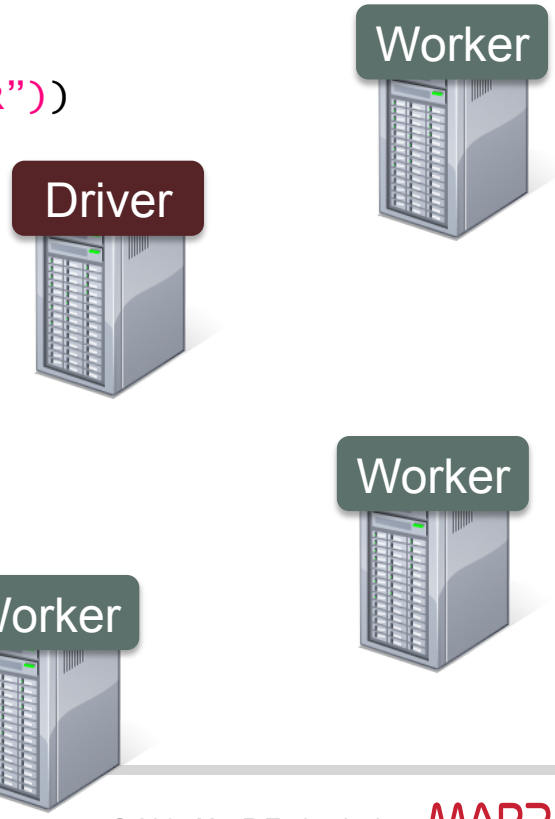
```
lines = spark.textFile("hdfs://...")
```



Example: Log Mining

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

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
```

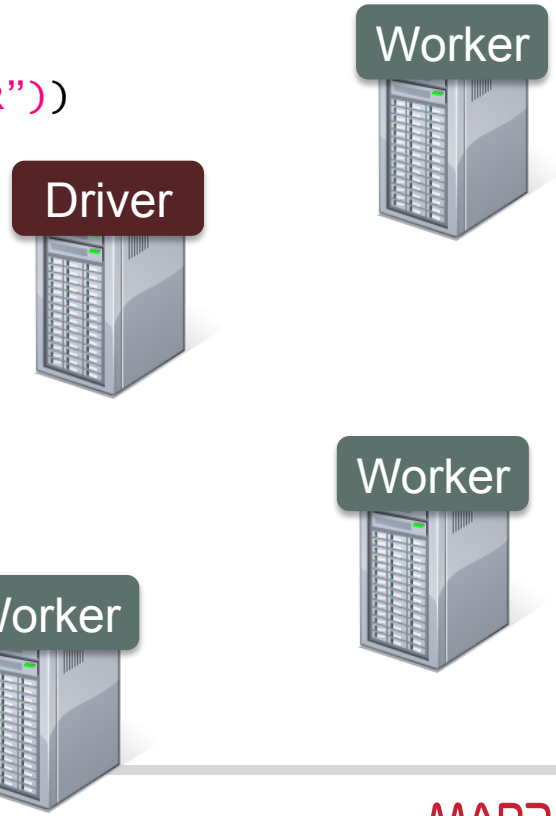


Example: Log Mining

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

Transformed RDD

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
```

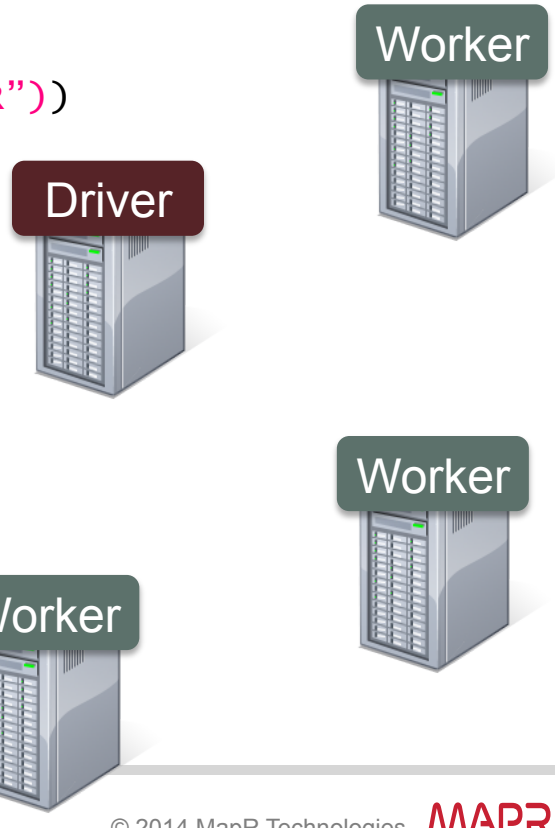


Example: Log Mining

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

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()
```

```
messages.filter(lambda s: “mysql” in s).count()
```

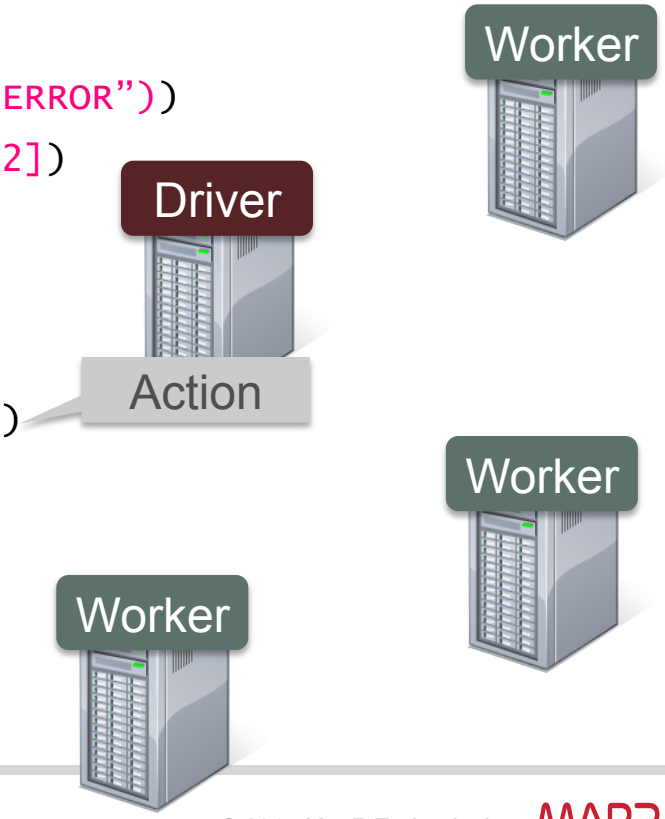


Example: Log Mining

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

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
```

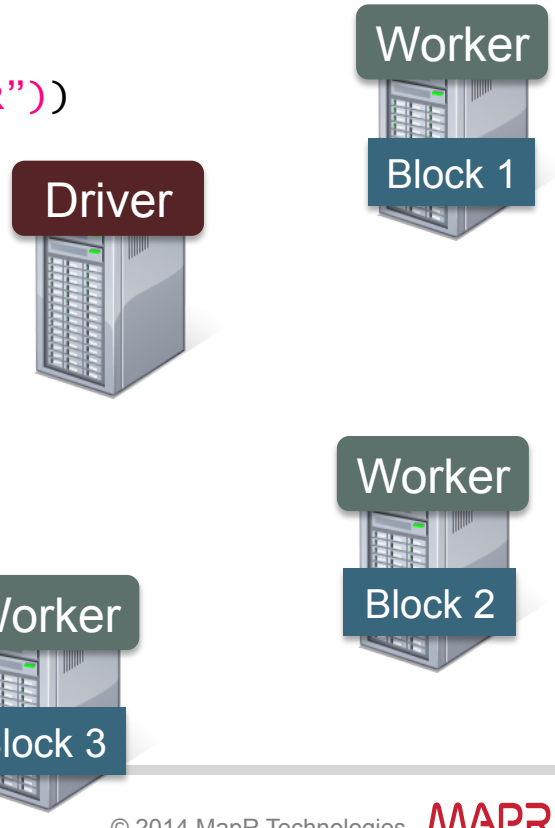


Example: Log Mining

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

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()
```

```
messages.filter(lambda s: “mysql” in s).count()
```

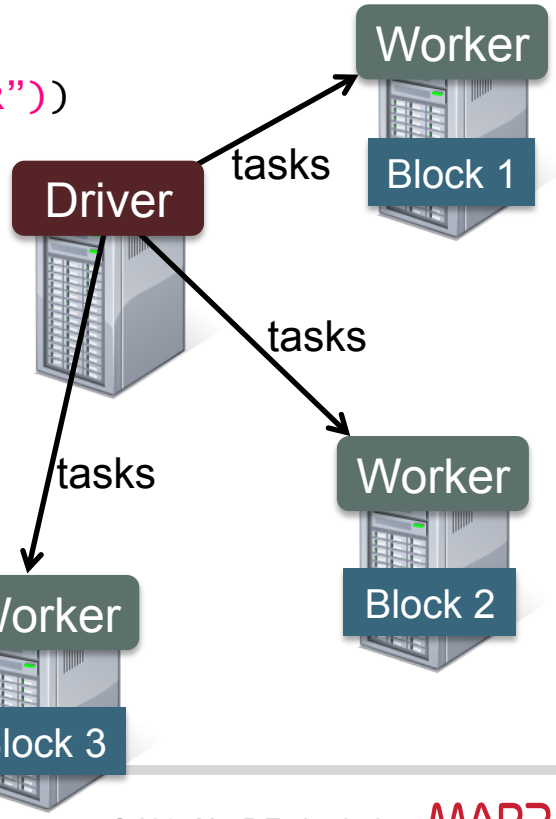


Example: Log Mining

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

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
```

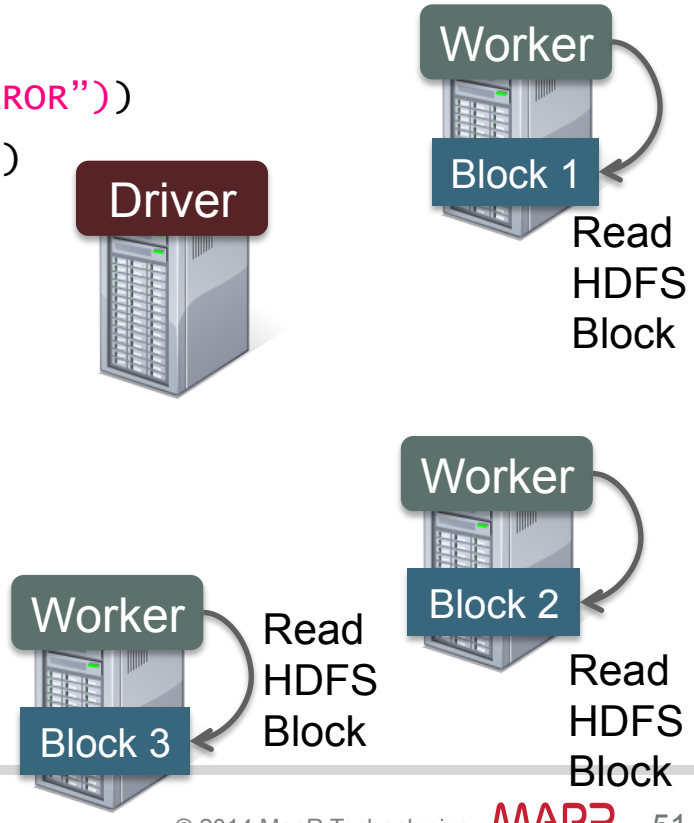


Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
```

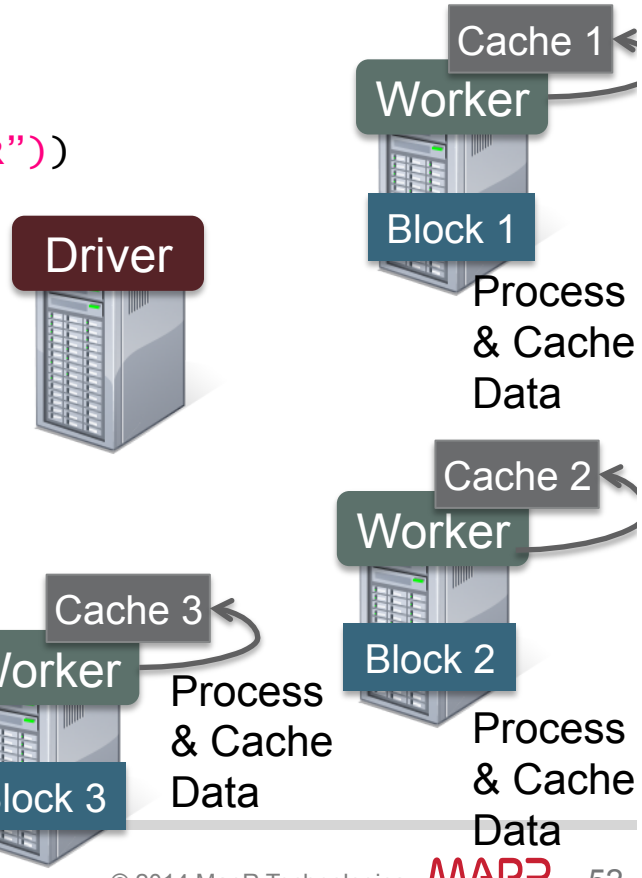


Example: Log Mining

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

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
```

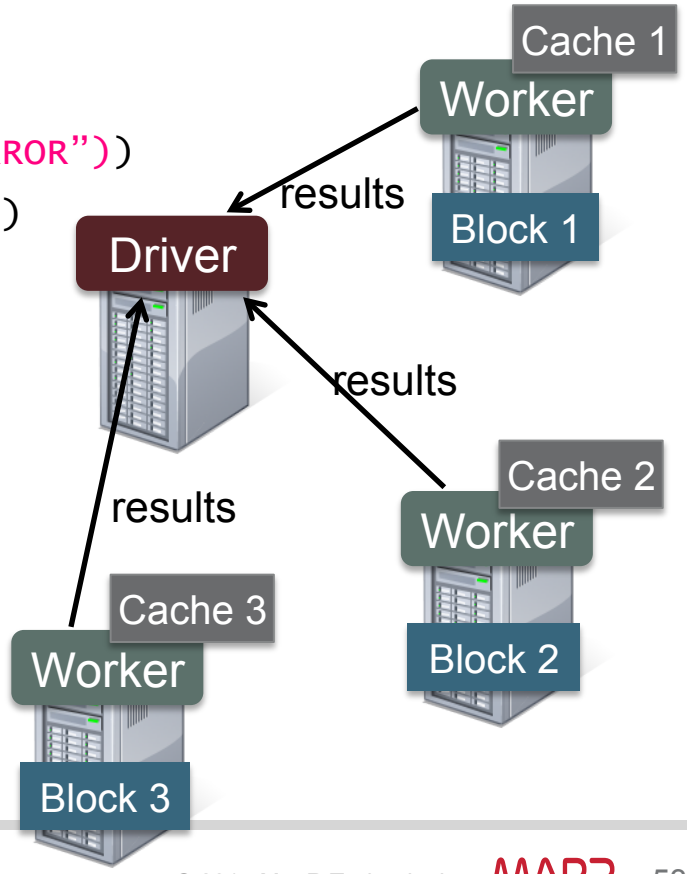


Example: Log Mining

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

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
```

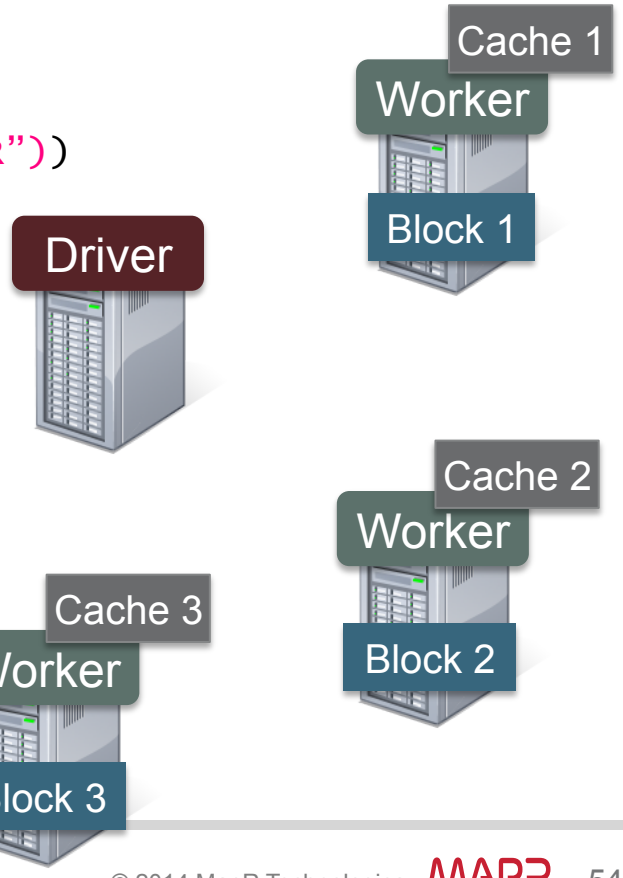


Example: Log Mining

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

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()
```

```
messages.filter(lambda s: “mysql” in s).count()
messages.filter(lambda s: “php” in s).count()
```

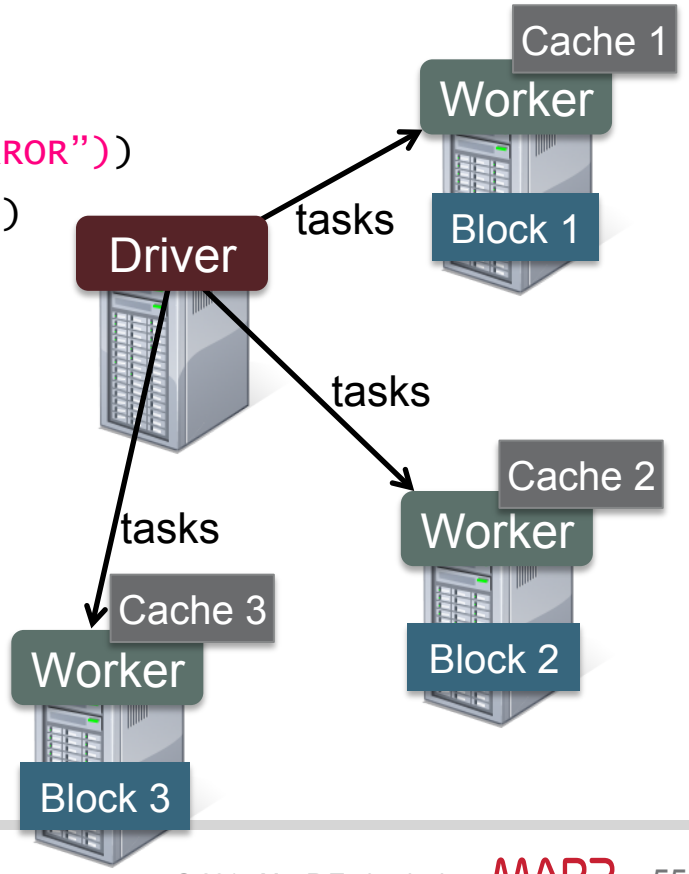


Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```

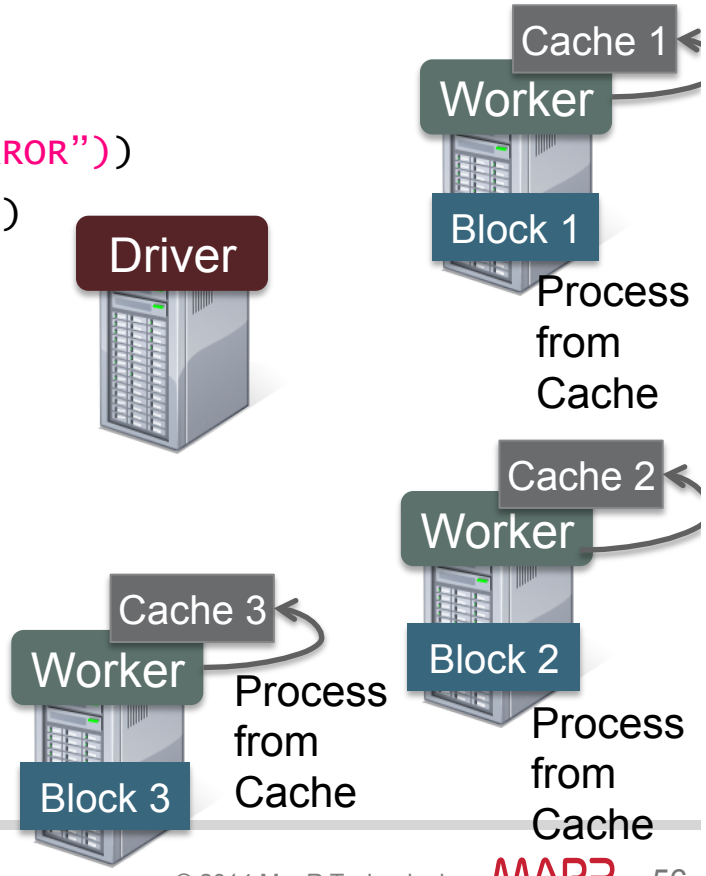


Example: Log Mining

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

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()
```

```
messages.filter(lambda s: “mysql” in s).count()
messages.filter(lambda s: “php” in s).count()
```

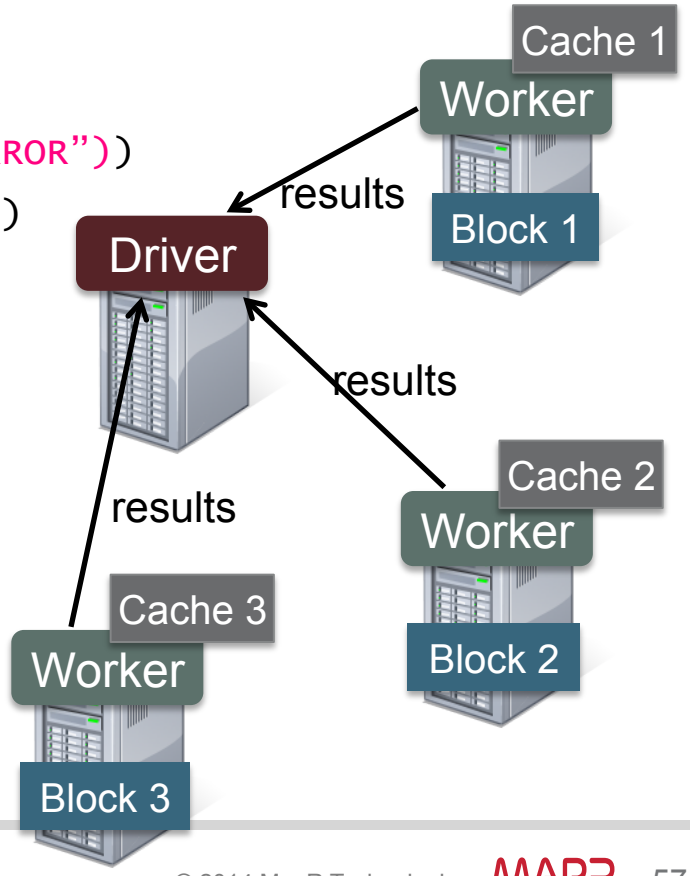


Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```

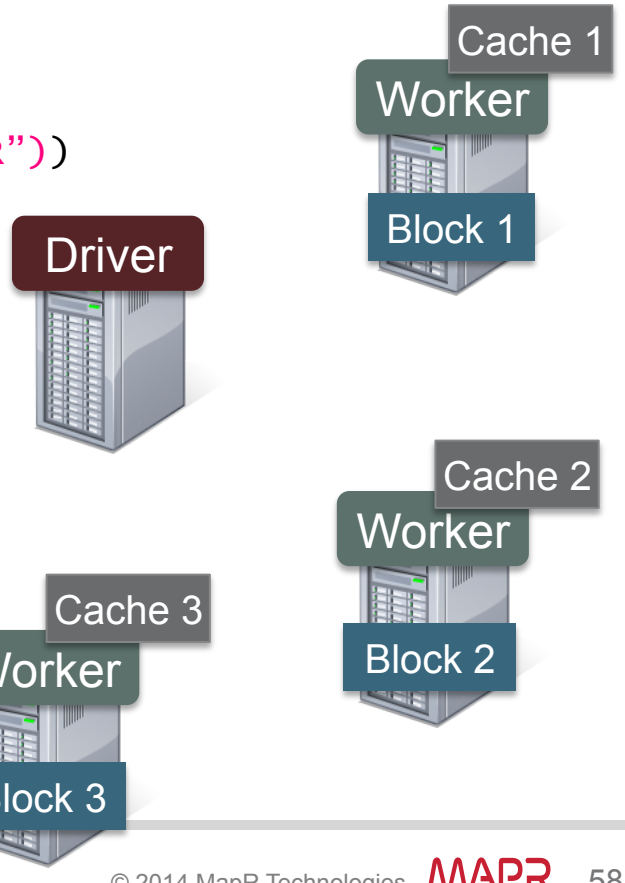


Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()
```

```
messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```



Cache your data → Faster Results

Full-text search of Wikipedia

- 60GB on 20 EC2 machines
- 0.5 sec from cache vs. 20s for on-disk

Example: Page Rank

Example: PageRank

- Good example of a more complex algorithm
  - Multiple stages of map & reduce
- Benefits from Spark's in-memory caching
  - Multiple iterations over the same data

Basic Idea

Give pages ranks (scores) based on links to them

- Links from many pages ➔ high rank
- Link from a high-rank page ➔ high rank

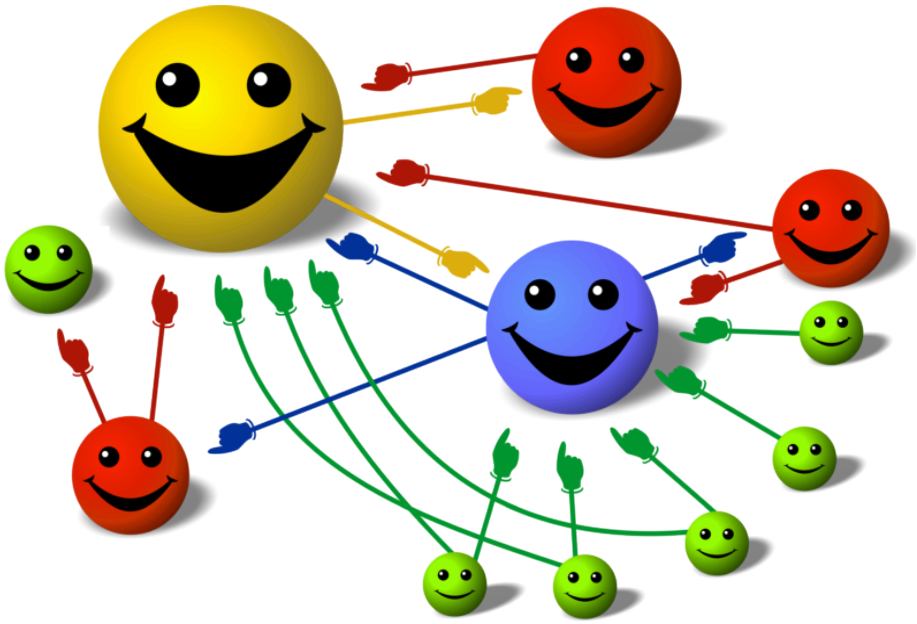
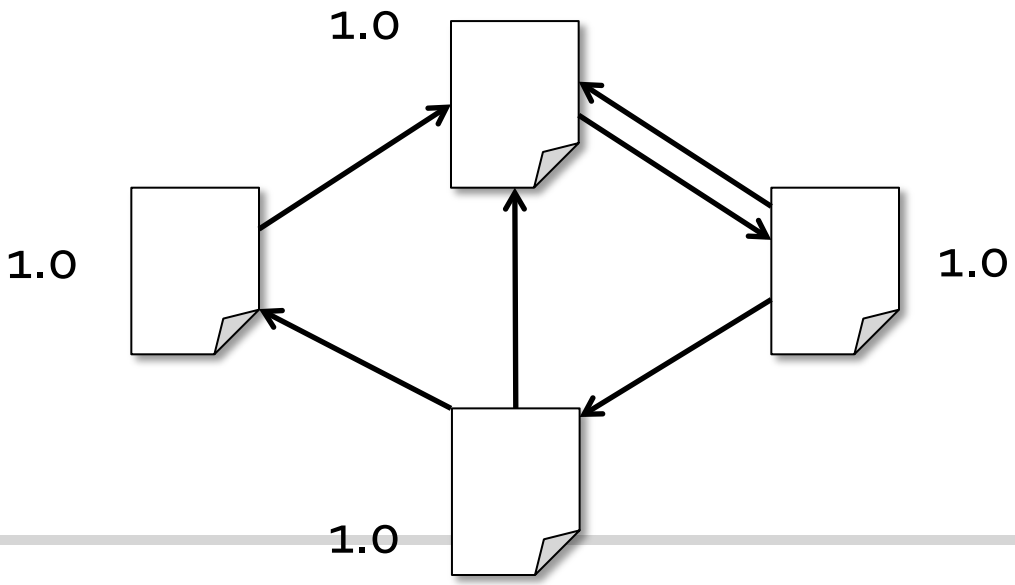


Image: en.wikipedia.org/wiki/File:PageRank-hi-res-2.png

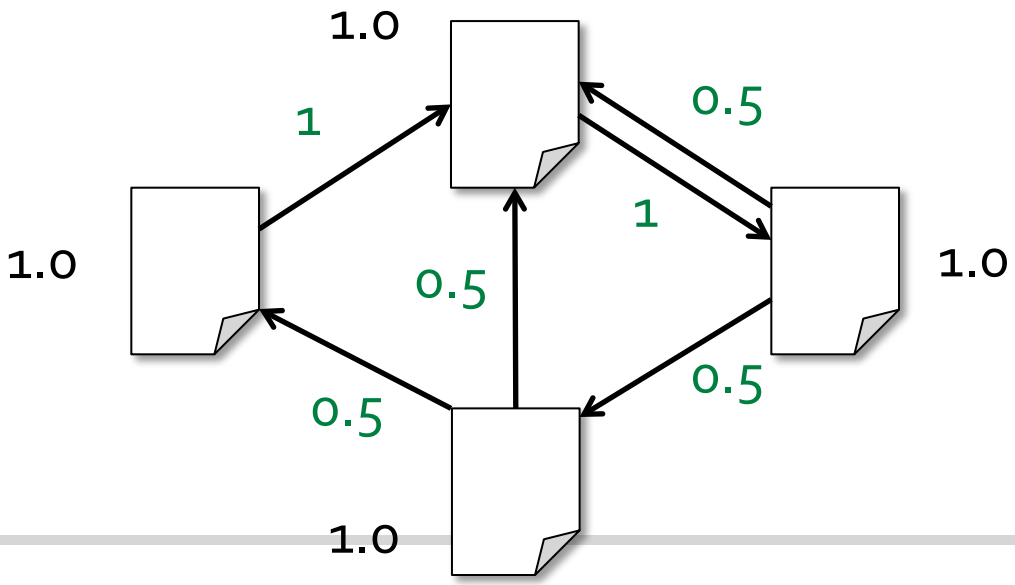
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



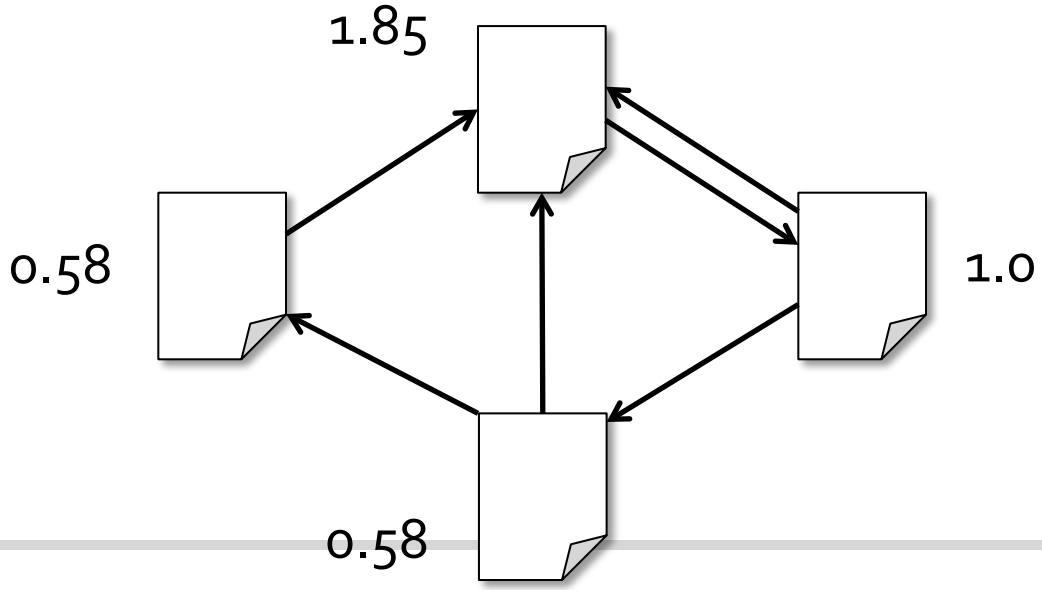
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



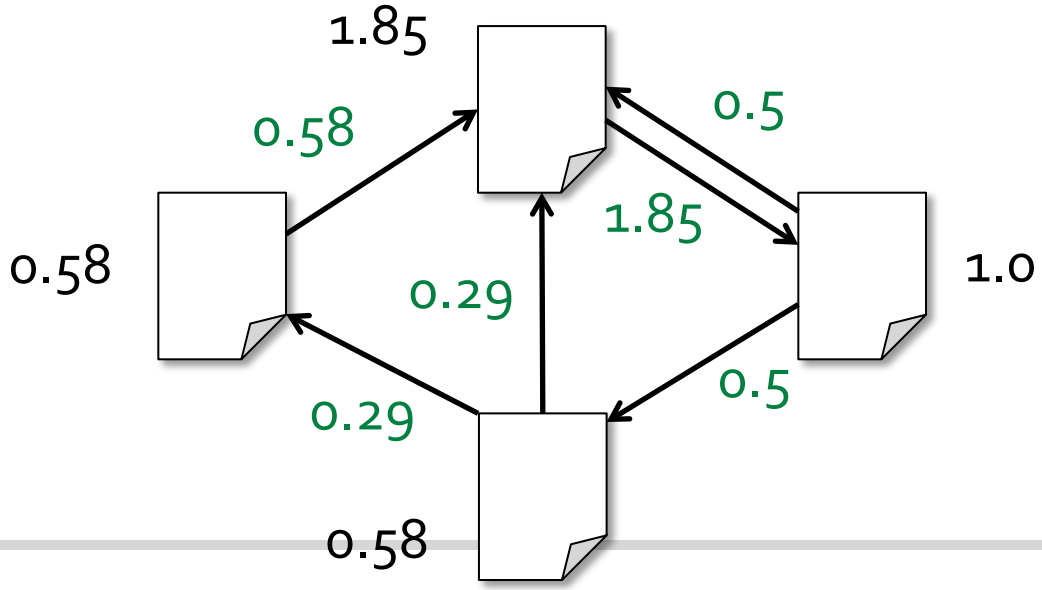
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



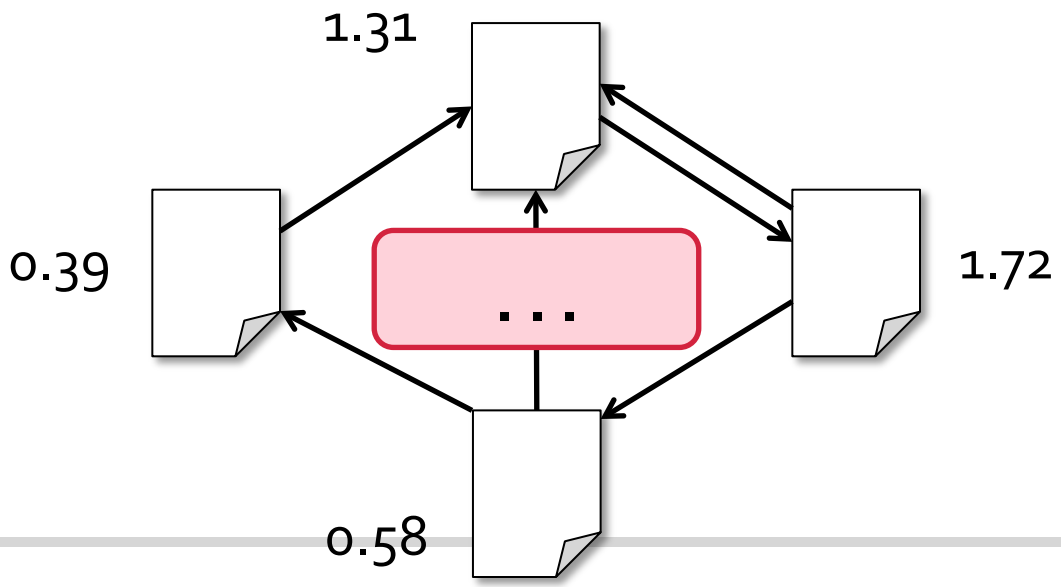
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



Algorithm

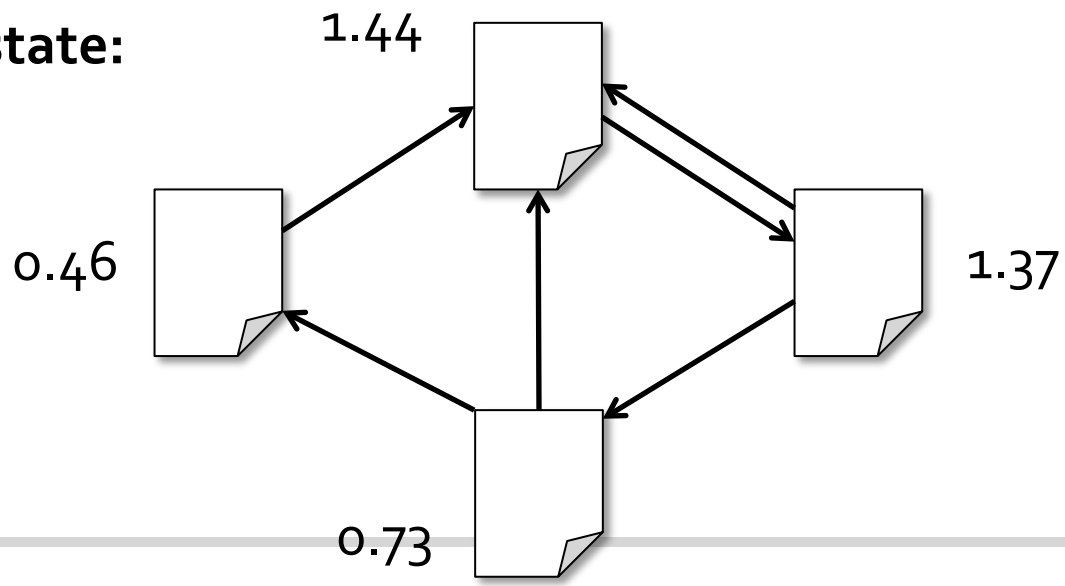
1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs

Final state:



Scala Implementation

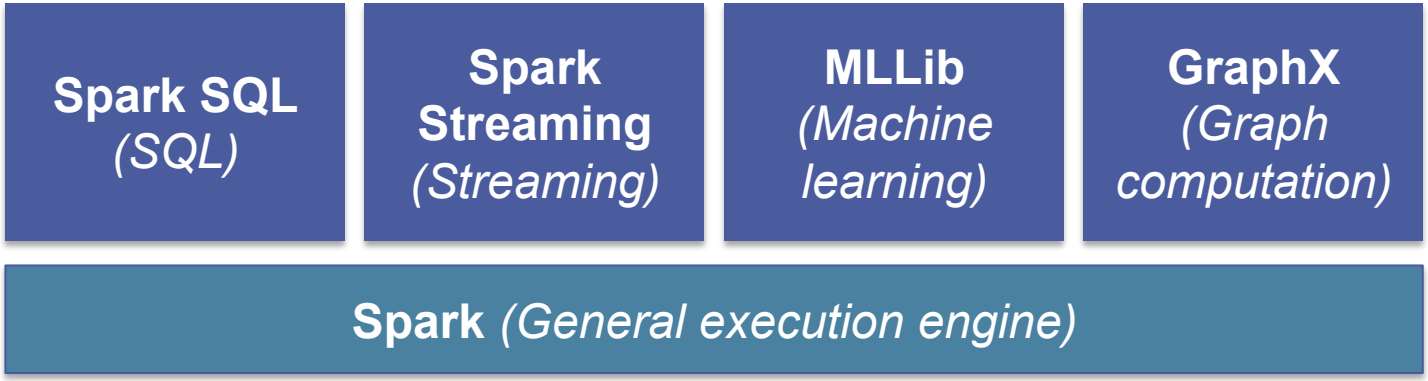
```
val links = // load RDD of (url, neighbors) pairs
var ranks = // give each url rank of 1.0

for (i <- 1 to ITERATIONS) {
  val contribs = links.join(ranks).values.flatMap {
    case (urls, rank)) =>
      urls.map(dest => (dest, rank/urls.size))
  }
  ranks = contribs.reduceByKey(_ + _)
                  .mapValues(0.15 + 0.85 * _)
}
ranks.saveAsTextFile(...)
```

https://github.com/apache/spark/blob/master/examples/src/main/scala/org/apache/spark/examples/SparkPageRank.scala

Spark and More

Easy: Unified Platform



Continued innovation bringing new functionality, e.g.,:

- **BlinkDB** (Approximate Queries)
- **SparkR** (R wrapper for Spark)
- **Tachyon** (off-heap RDD caching)

Spark on MapR

- Certified Spark Distribution
- Fully supported and packaged by MapR in partnership with Databricks
  - mapr-spark package with Spark, Shark, Spark Streaming today
  - Spark-python, GraphX and MLLib soon
- YARN integration
  - Spark can then allocate resources from cluster when needed

References

- Based on slides from Pat McDonough at

- Spark web site: http://spark.apache.org/
- Spark on MapR:
  - http://www.mapr.com/products/apache-spark
  - http://doc.mapr.com/display/MapR/Installing+Spark+and+Shark

Q&A

Engage with us!

@mapr   maprtech

mapr-technologies   MapR

kbotzum@mapr.com   maprtech