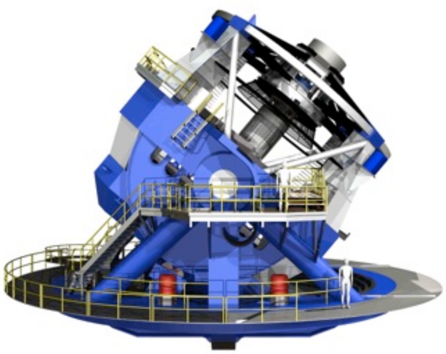
Big Data Analytics: The Apache Spark Approach

Michael Franklin
ATPESC
August 2017

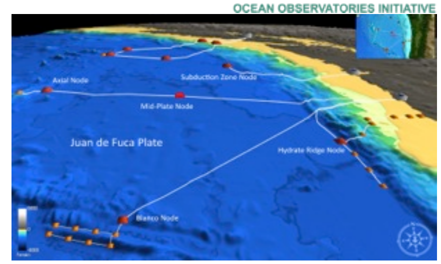
Nearly every field of endeavor is transitioning from “data poor” to “data rich”



Astronomy: LSST

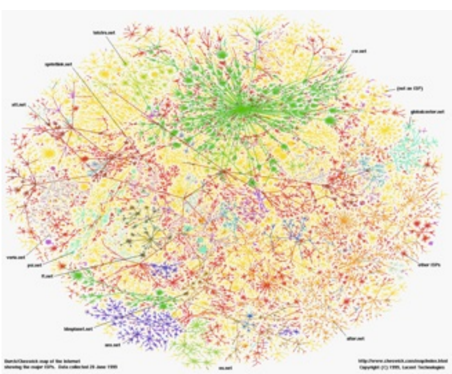
Physics: LHC

OCEAN OBSERVATORIES INITIATIVE



Oceanography

Neuroscience: EEG, fMRI



Sociology: The Web

Biology: Sequencing

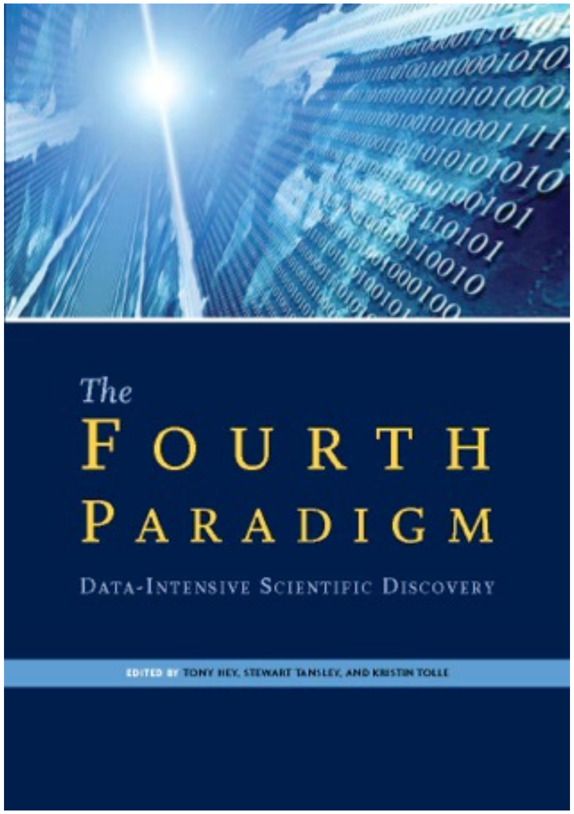
Economics: mobile, POS terminals

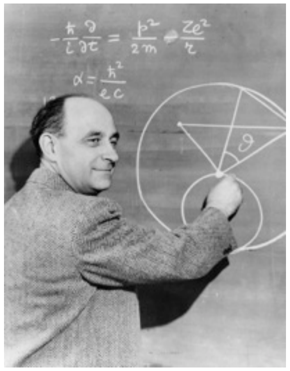
Data-Driven Medicine

Sports

The Fourth Paradigm of Science

1. Empirical + experimental
2. Theoretical
3. Computational
4. Data-Intensive





Open Source Ecosystem & Context

2006-2010
Autonomic Computing & Cloud

2011-2016
Big Data Analytics

Usenix HotCloud Workshop 2010

Spark: Cluster Computing with Working Sets

Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica
University of California, Berkeley

Abstract

MapReduce and its variants have been highly successful in implementing large-scale data-intensive applications on commodity clusters. However, most of these systems are built around an acyclic data flow model that is not suitable for other popular applications. This paper focuses on one such class of applications: those that reuse

MapReduce/Dryad job, each job must reload the data from disk, incurring a significant performance penalty.

- **Interactive analytics:** Hadoop is often used to run ad-hoc exploratory queries on large datasets, through SQL interfaces such as Pig [21] and Hive [1]. Ideally, a user would be able to load a dataset of interest into memory across a number of machines and query it re-

2006
- Core Hadoop
- HDFS, MR

2008
- Hbase
- ZooKeeper
- Core Hadoop

2009
- Hive
- Pig
- Mahout
- Hbase
- ZooKeeper
- Core Hadoop

2010
- Sqoop
- Whirr
- Avro
- Hive
- Pig
- Mahout
- Hbase
- Zookeeper
- Core Hadoop

2011
- Flume
- Bigtop
- Oozie
- MRUnit
- HCatalog
- Sqoop
- Whirr
- Avro
- Hive
- Pig
- Mahout
- Hbase
- Zookeeper
- Core Hadoop

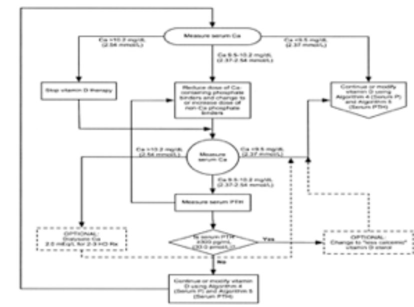
2012
- Spark
- Impala
- Solr
- Kafka
- Flume
- Bigtop
- Oozie
- MRUnit
- HCatalog
- Sqoop
- Whirr
- Avro
- Hive
- Pig
- Mahout
- Hbase
- Zookeeper
- Core Hadoop

Present
- Sentry
- Tez
- Parquet
- YARN
- Spark
- YARN
- Impala
- Solr
- Kafka
- Flume
- Bigtop
- Oozie
- MRUnit
- HCatalog
- Sqoop
- Whirr
- Avro
- Hive
- Pig
- Mahout
- Hbase
- Zookeeper
- Core Hadoop
- Knox

…

AMPLab Project Vision
"Making Sense of Data at Scale"

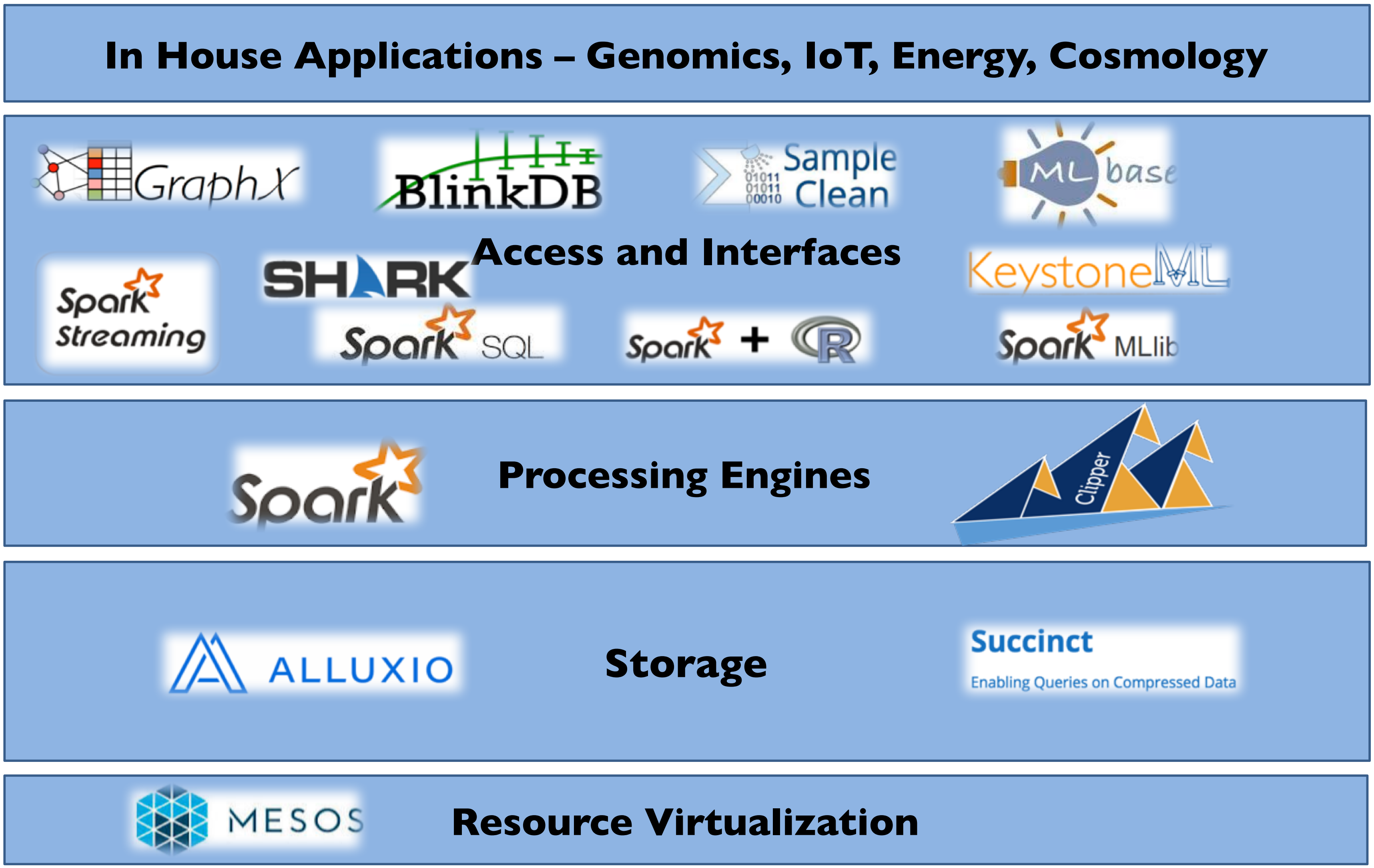
Algorithms
- Machine Learning, Statistical Methods
- Prediction, Business Intelligence



Machines
- Clusters and Clouds
- Warehouse Scale Computing

People
- Crowdsourcing, Human Computation
- Data Scientists, Analysts

Berkeley Data Analytics Stack



Some AMPLab numbers

- Funding – roughly 50/50 Govt/Industry Split
  - NSF CISE Expeditions, DARPA, DOE, DHS
  - Google, SAP, Amazon, IBM (Founding Sponsors) + dozens more
- Nearly 2M visits to amplab.cs.berkeley.edu
- 200+ Papers in Sys, ML, DB, … 3 ACM Dissertation Awards (1 + 2 HM); Numerous Best Paper and Best Demo Awards
- 40+ Ph.D.s granted (so far); Alumni on faculty at Berkeley, Harvey Mudd, Michigan, MIT, Stanford, Texas, Wisconsin,…
- 3 Spinout companies directly from AMPLab:
  - Databricks, Mesosphere, Alluxio
  - Nearly $250M raised to date
- Many industrial products & services based on or using Spark
- 3 Marriages (and numerous long-term relationships)

Apache Spark Meetups (August 2017)

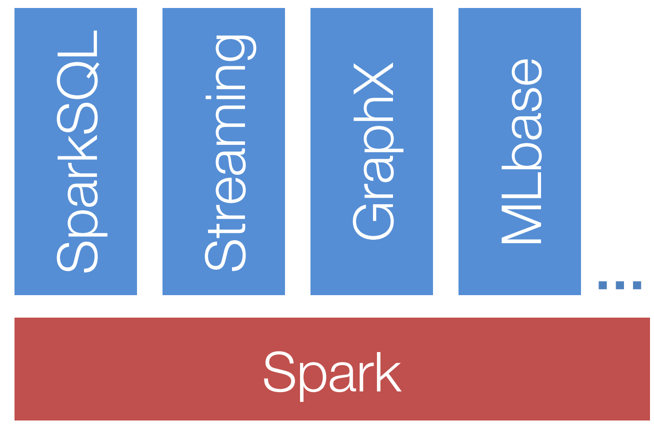
618 groups with 391,371 **members**
spark.meetup.com

We Hit A Data Management Inflection Point

- Massively scalable processing and storage
- Pay-as-you-go processing and storage (a.k.a. the cloud)
- Flexible schema on read vs. schema on write
- Integration of search, query and analysis
- Sophisticated machine learning/prediction
- Human-in-the-loop analytics
- Open source ecosystem driving innovation

BDAS Unification Strategy

- Specializing MapReduce leads to stovepiped systems
- Instead, **generalize** MapReduce:



1. Richer Programming Model
   ➔ Fewer Systems to Master
2. Data Sharing
   ➔ Less Data Movement leads to Better Performance

Spark showed 10x performance improvement on existing HDFS data with no migration.

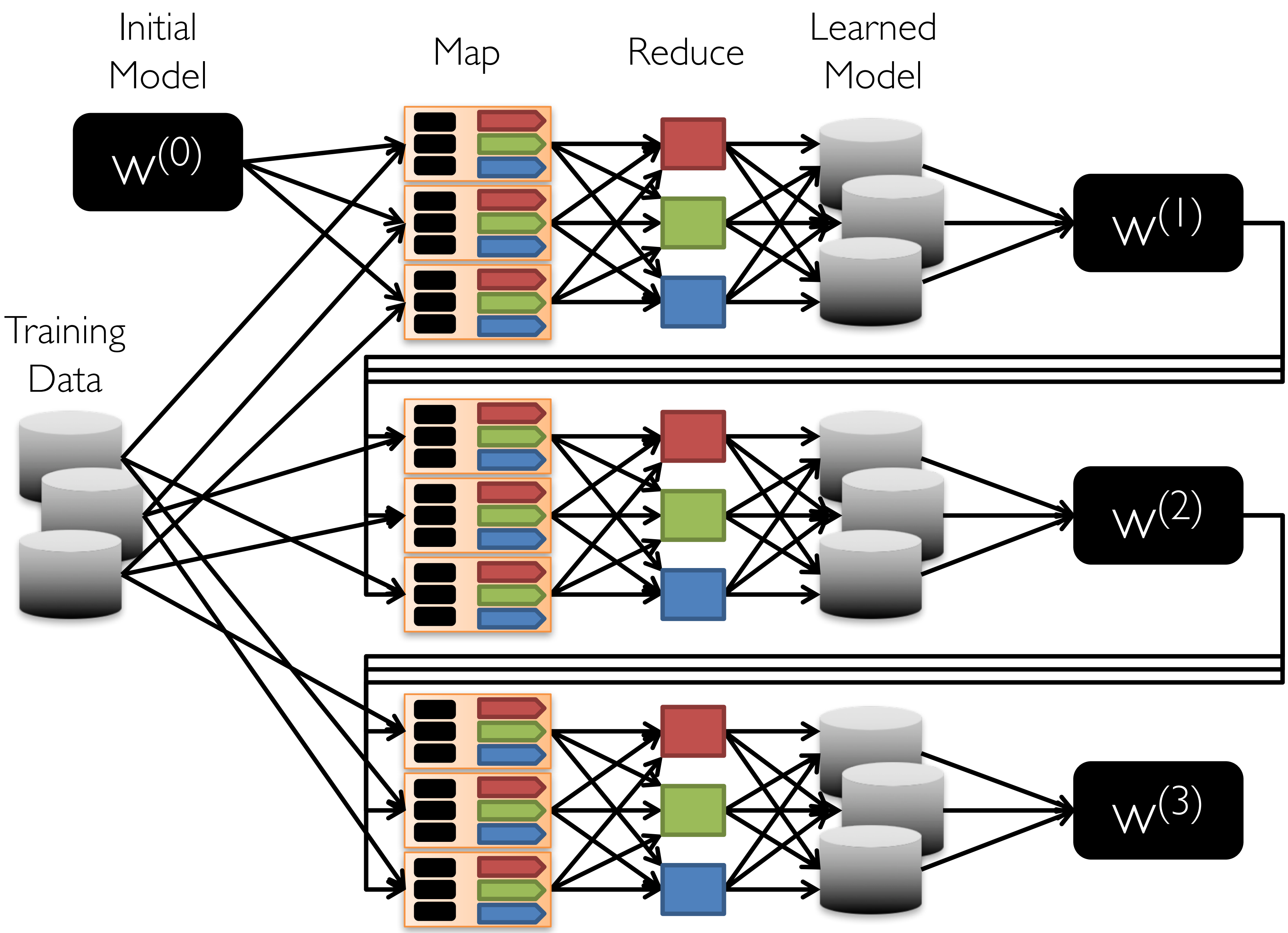
Abstraction: *Dataflow Operators*

- map
- filter
- groupBy
- sort
- union
- join
- leftOuterJoin
- rightOuterJoin

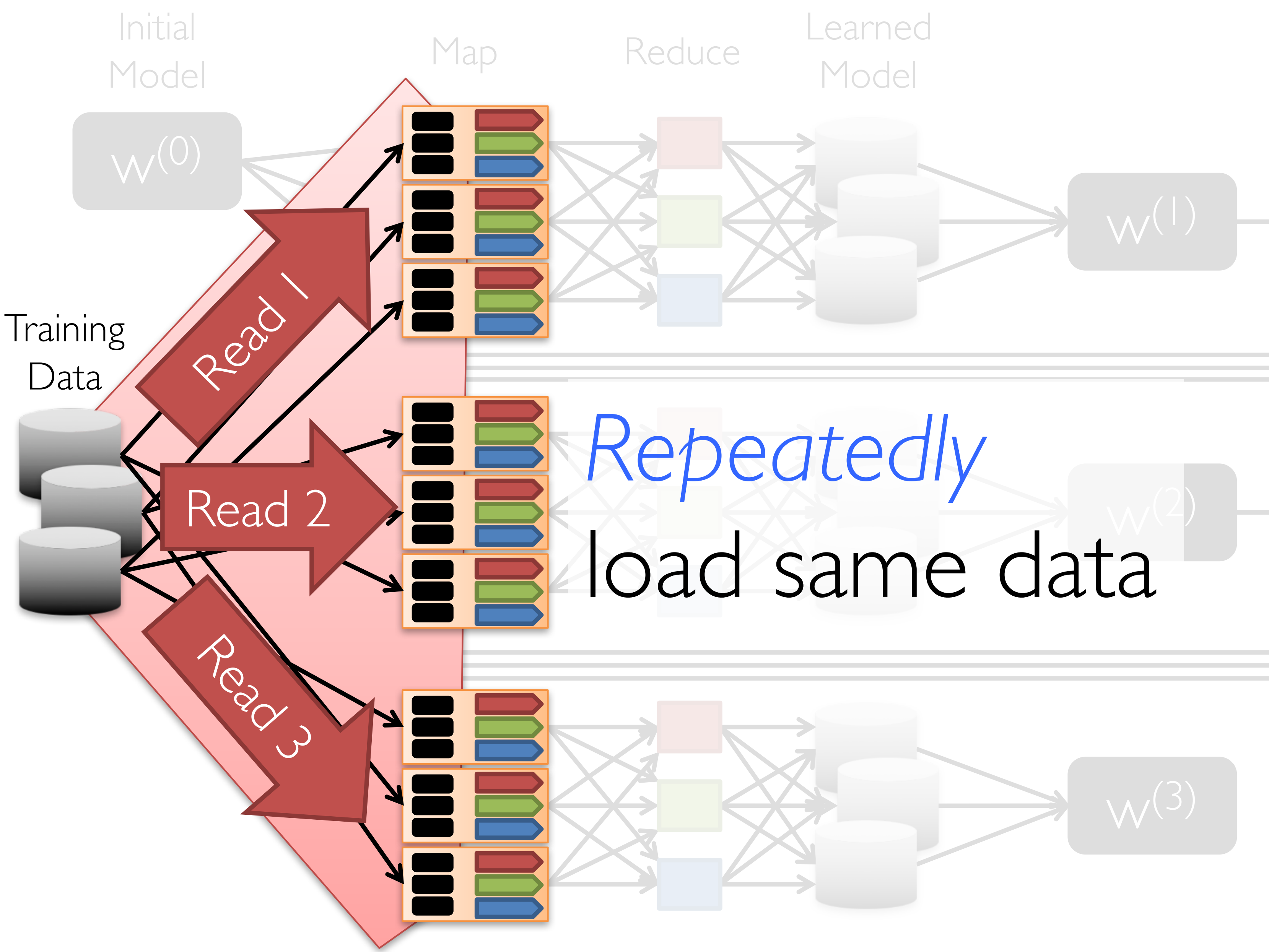
- reduce
- count
- fold
- reduceByKey
- groupByKey
- cogroup
- cross
- zip

sample
take
first
partitionBy
mapWith
pipe
save
...

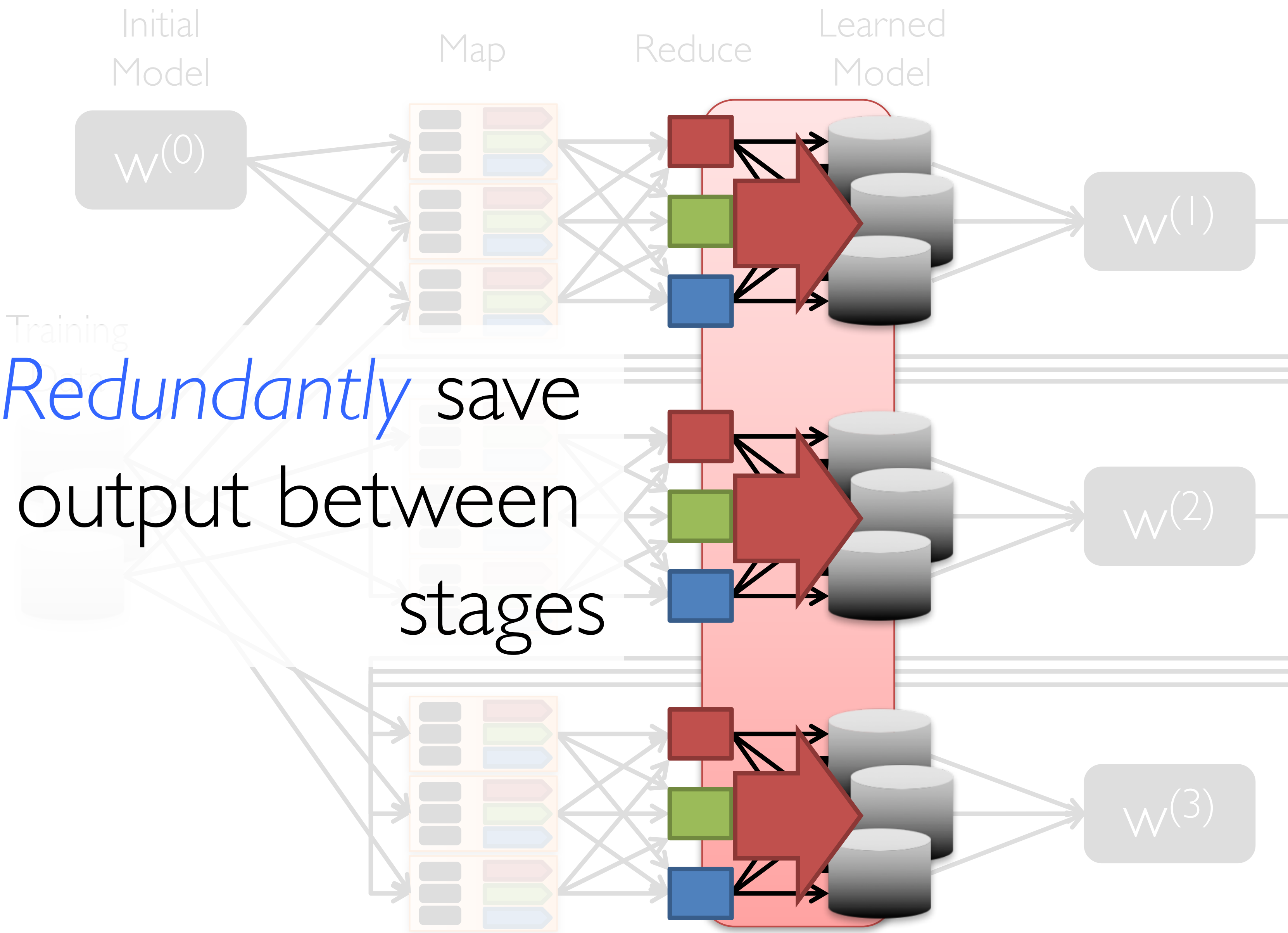
Iteration in Map-Reduce



Cost of Iteration in Map-Reduce

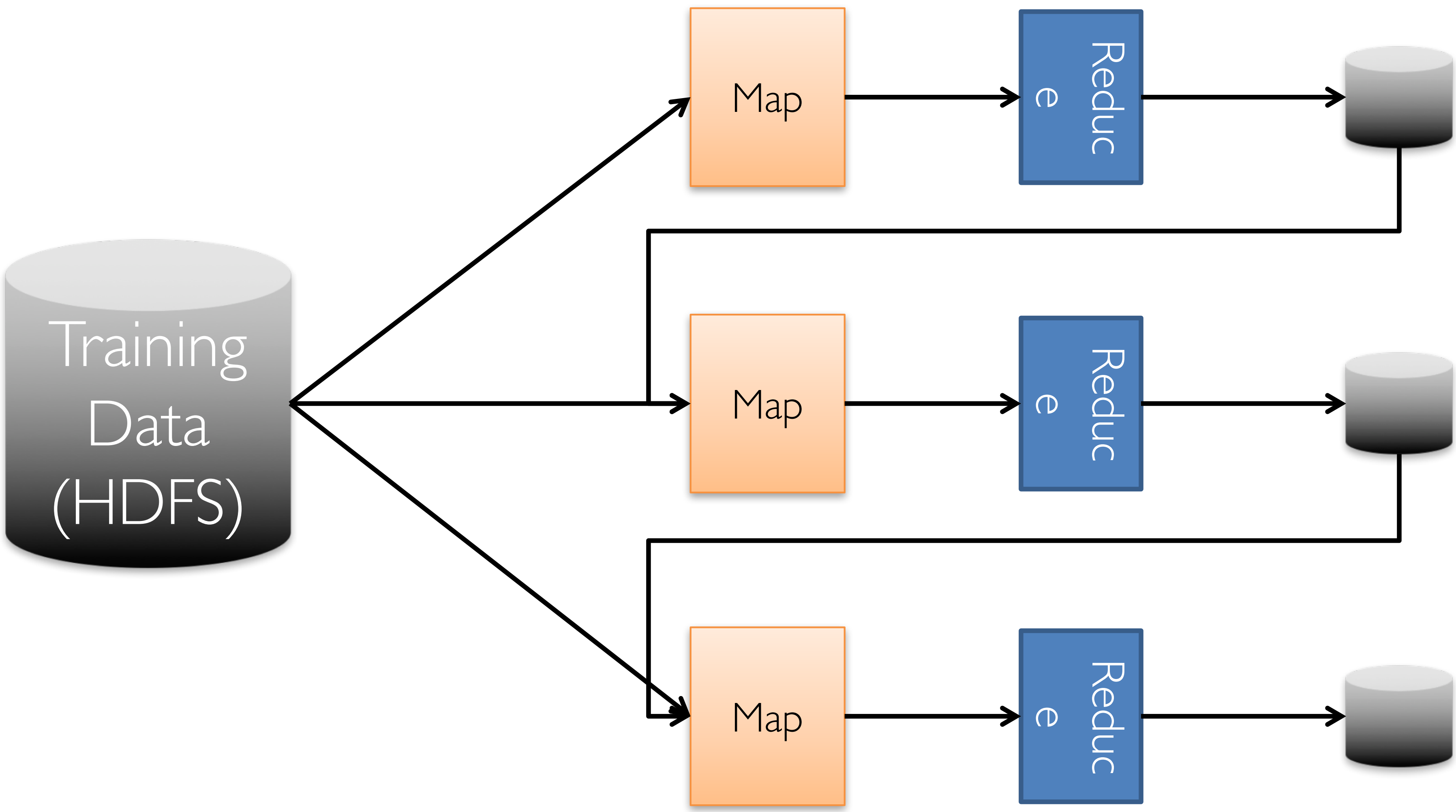


Cost of Iteration in Map-Reduce

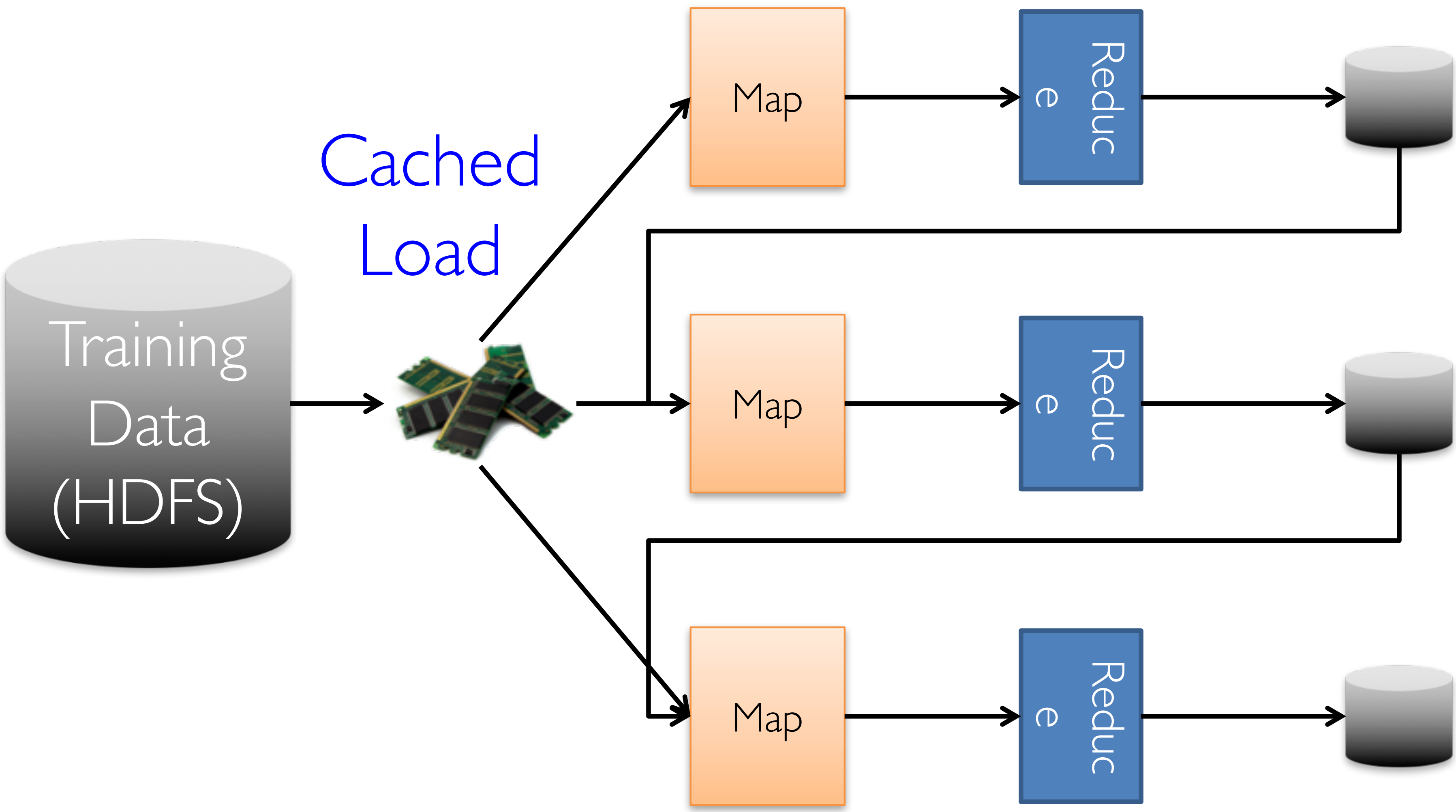


Redundantly save output between stages

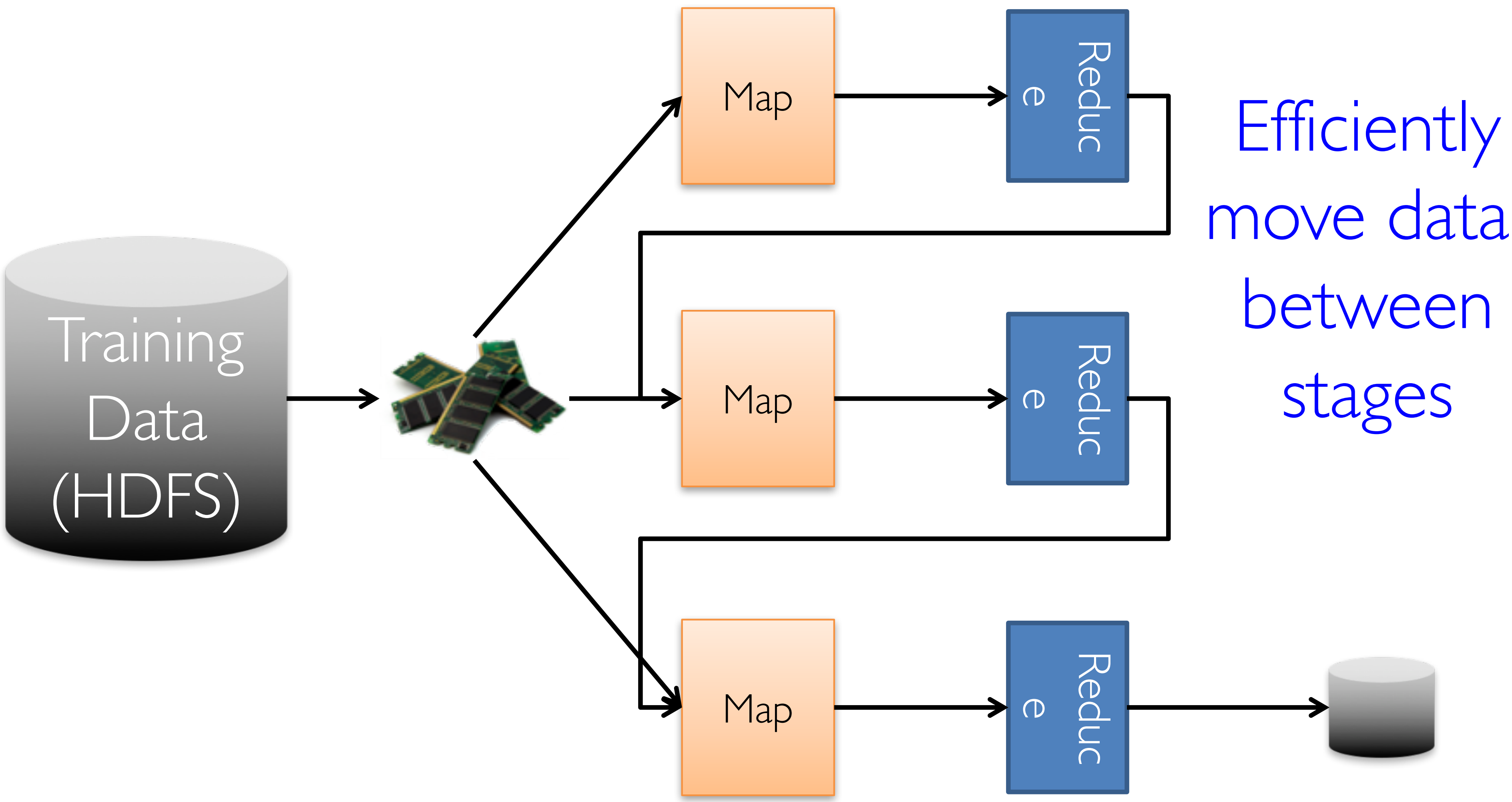
Dataflow View



Memory Opt. Dataflow



Memory Opt. Dataflow View

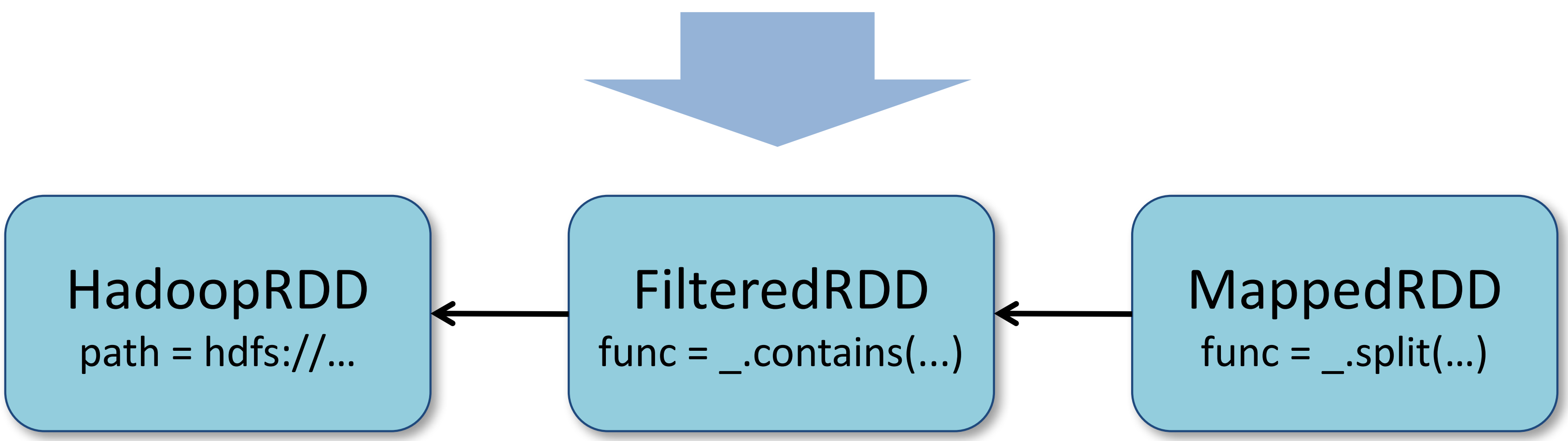


Spark: **10-100×** faster than Hadoop MapReduce

Spark Fault Tolerance

- **RDDs:** **Immutable** collections of objects that can be stored in memory or disk across a cluster
  - Built via parallel transformations (map, filter, …)
  - Automatically rebuilt on (partial) failure

```
messages = textFile(...).filter(_.contains(“error”))
                        .map(_.split(‘\t’)(2))
```



M. Zaharia, et al, Resilient Distributed Datasets: A fault-tolerant abstraction for in-memory cluster computing, NSDI 2012.

DataFrames
(main abstraction in Spark 2.0)

```
employees
.join(dept, employees("deptId") === dept("id"))
.where(employees("gender") === "female")
.groupBy(dept("id"), dept("name"))
.agg(count("name"))
```

Notes:
1) Some people think this is an improvement over SQL ☺
2) Dataframes can be typed

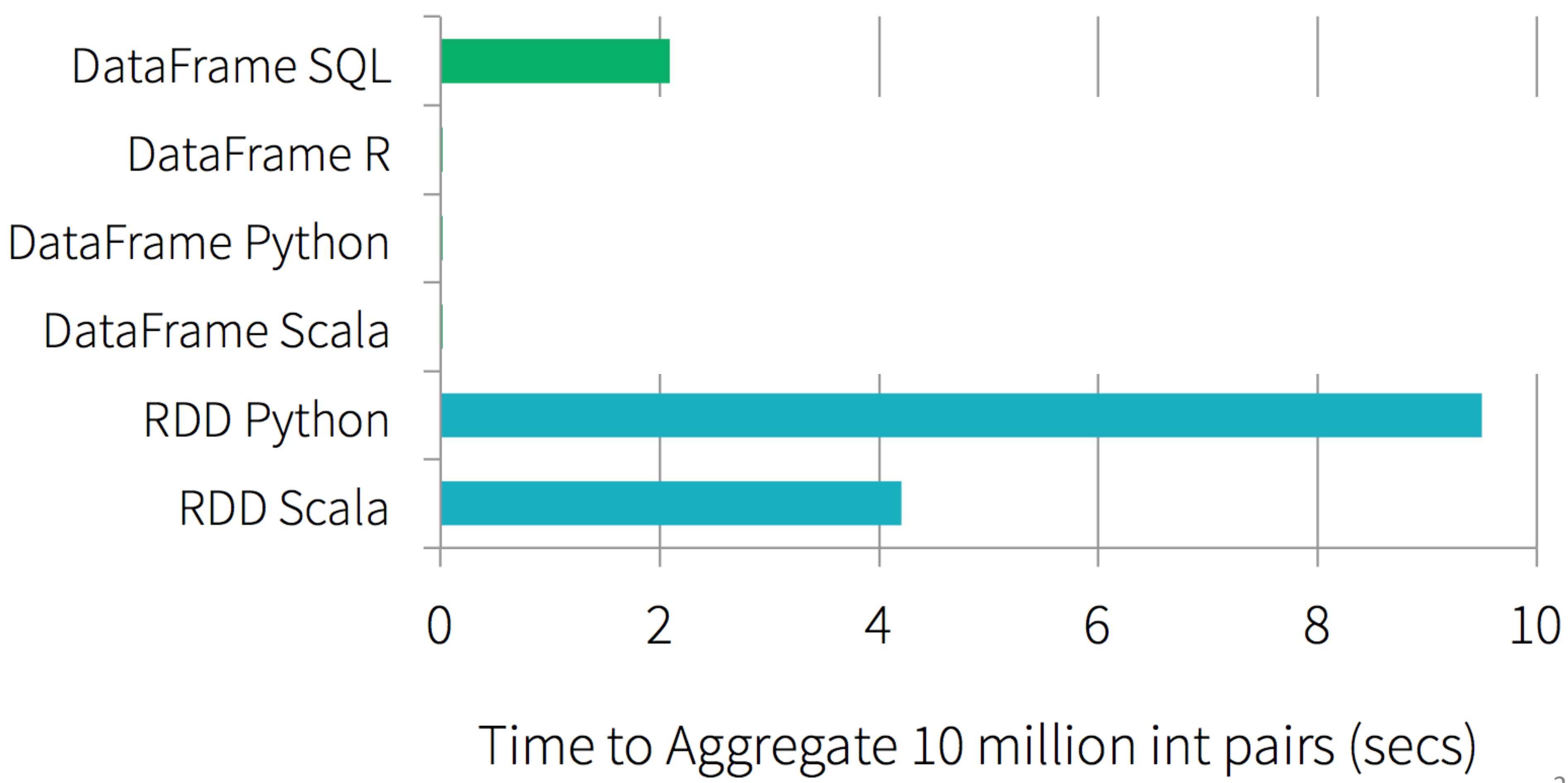
Catalyst Optimizer

- Typical DB optimizations across SQL and DF
  - Extensibility via Optimization Rules written in Scala
  - Open Source optimizer evolution!
- Code generation for inner-loops, iterator removal
- Extensible Data Sources: CSV, Avro, Parquet, JDBC, …

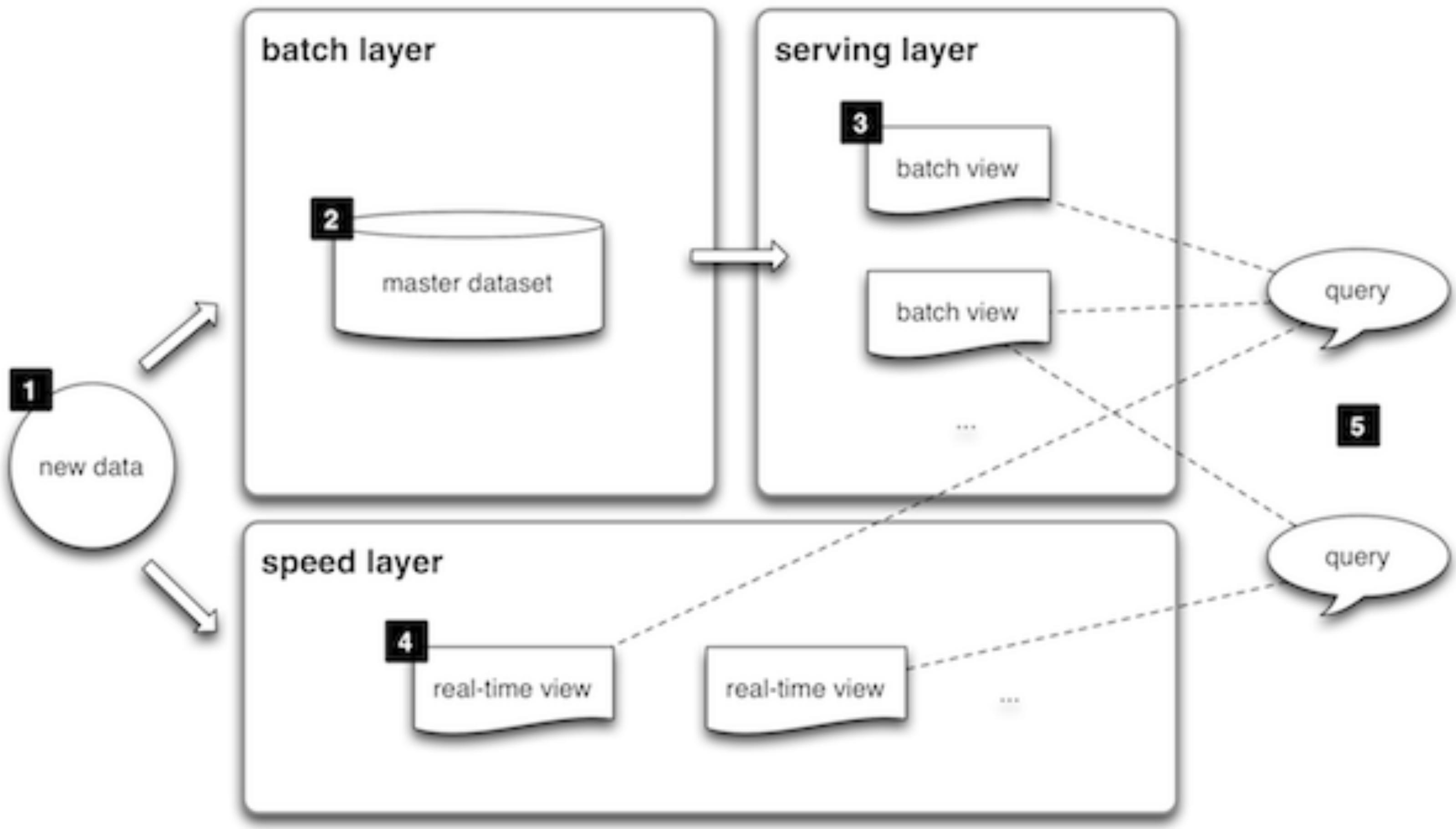
  via TableScan (all cols), PrunedScan (project),
  FilteredPrunedScan(push advisory selects and projects)
  CatalystScan (push advisory full Catalyst expression trees)
- Extensible (User Defined) Types

M. Armbrust, et al, Spark SQL: Relational Data Processing in Spark, SIGMOD 2015.

An interesting thing about SparkSQL Performance



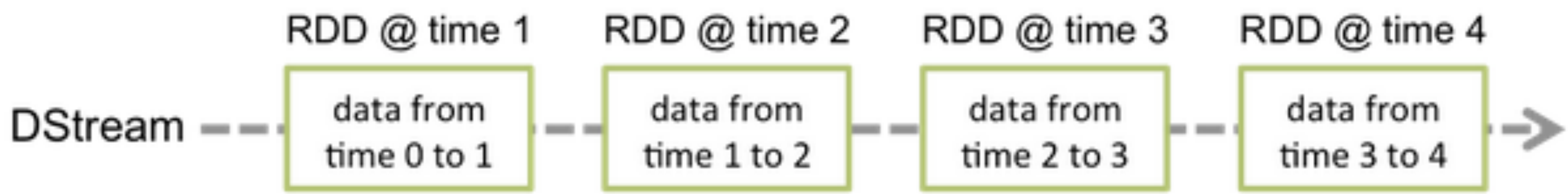
Lambda Architecture: one way to combine Real-Time + Batch



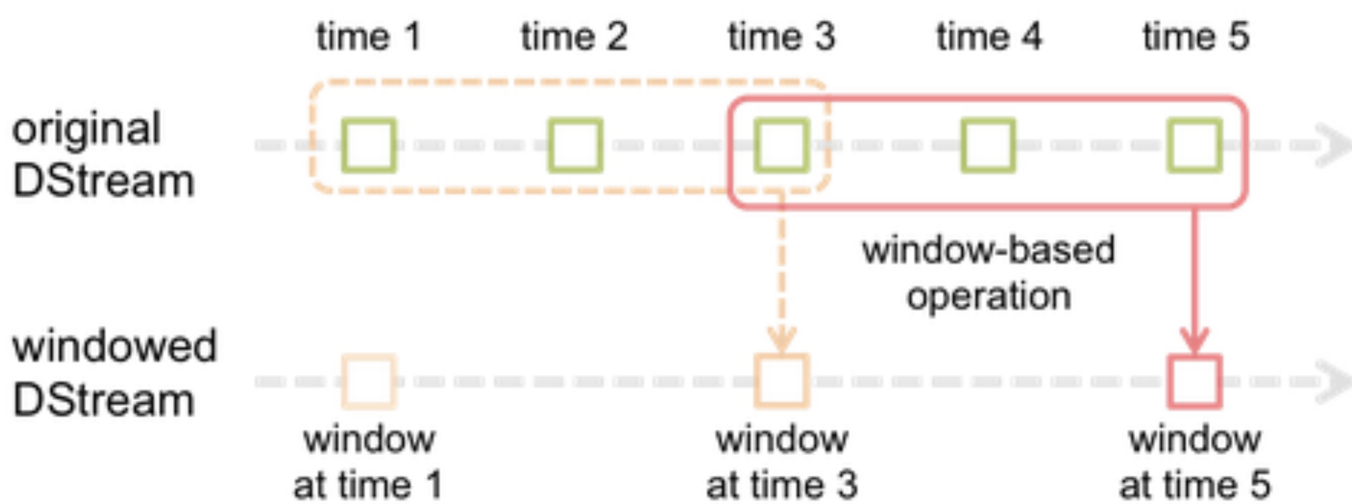
- lambda-architecture.net

Spark Streaming

- Microbatch approach provides low latency



Additional operators provide windowed operations



M. Zaharia, et al, Discretized Streams: Fault-Tollerant Streaming Computation at Scale, SOSP 2013
S. Venketaraman et al, Azkar: Fast and Adaptable Stream Processing at Scale, SOSP 2017

Spark Structured Streams (unified)

Batch Analytics

```
// Read data once from an S3 location
val inputDF = spark.read.json("s3://logs")

// Do operations using the standard DataFrame API and write to MySQL
inputDF.groupBy($"action", window($"time", "1 hour")).count()
        .write.format("jdbc")
        .save("jdbc:mysql//...")
```

Streaming Analytics

```
// Read data continuously from an S3 location
val inputDF = spark.readStream.json("s3://logs")

// Do operations using the standard DataFrame API and write to MySQL
inputDF.groupBy($"action", window($"time", "1 hour")).count()
       .writeStream.format("jdbc")
       .start("jdbc:mysql//...")
```

Putting it all Together: Multi-modal Analytics

SQL

```
// Load historical data as an RDD using Spark SQL
val trainingData = sql(
  "SELECT location, language FROM old_tweets")
```

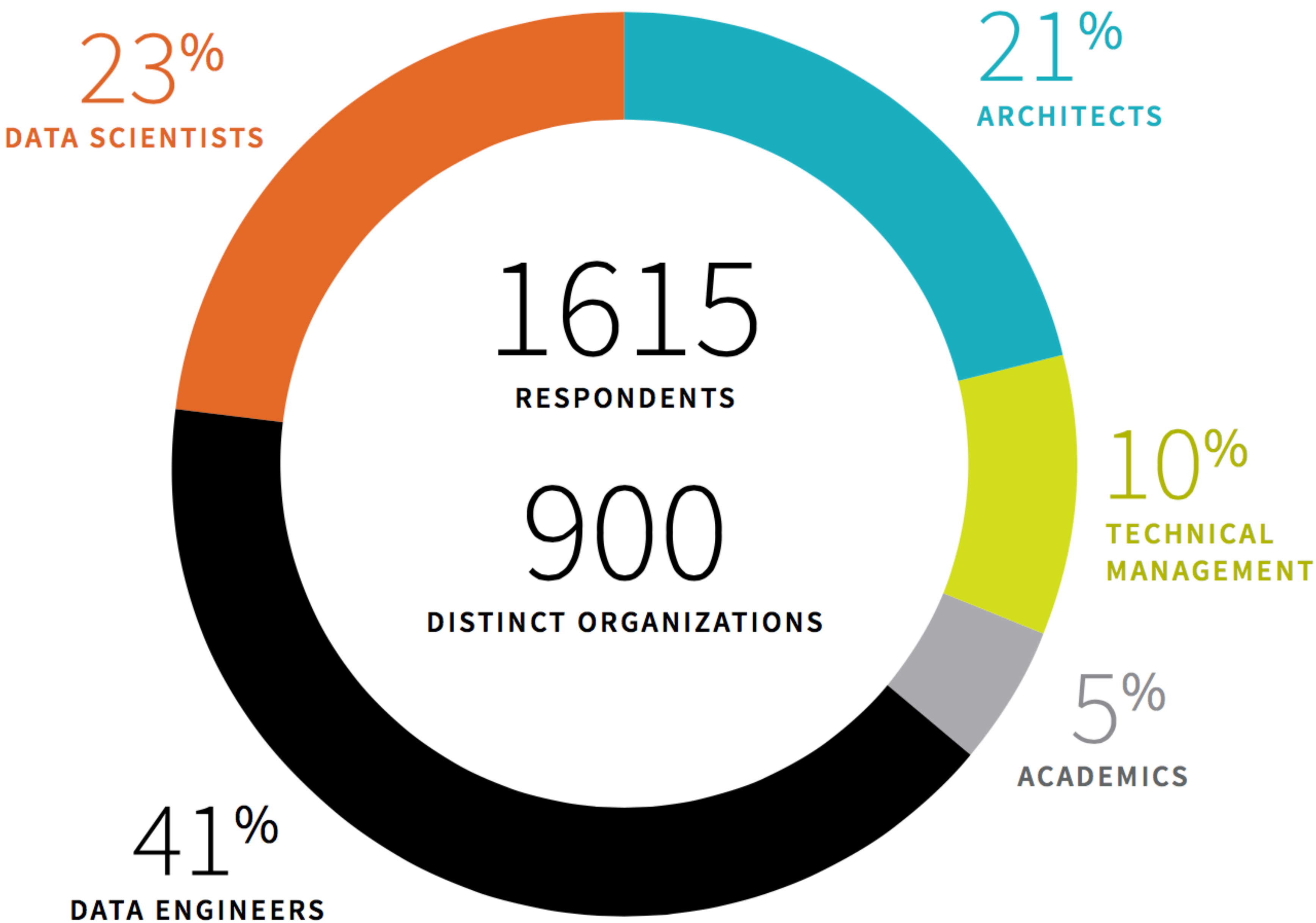
Machine Learning

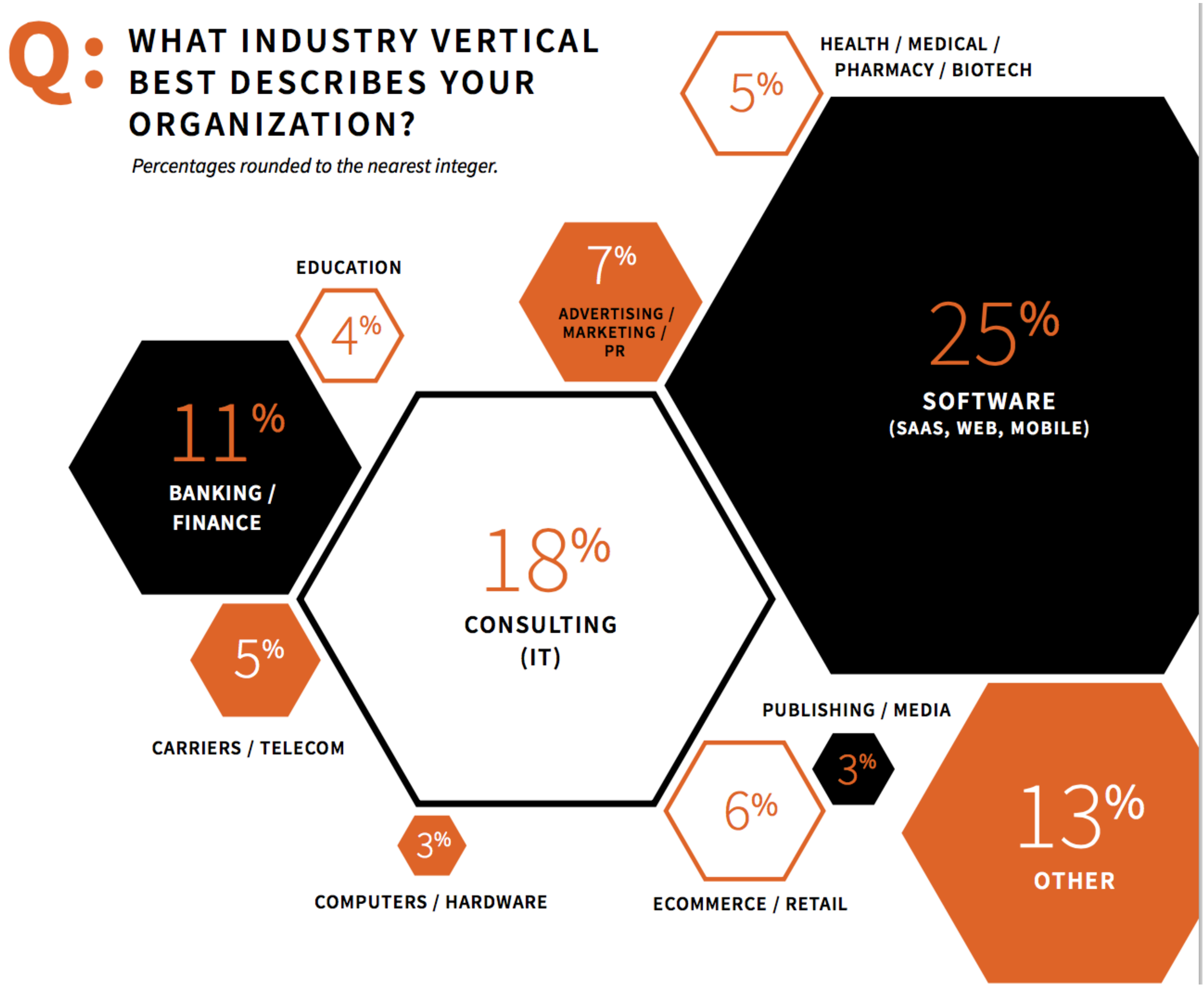
```
// Train a K-means model using MLlib
val model = new KMeans()
  .setFeaturesCol("location")
  .setPredictionCol("language")
  .fit(trainingData)
```

Streaming

```
// Apply the model to new tweets in a stream
TwitterUtils.createStream(...)
  .map(tweet => model.predict(tweet.location))
```

SPARK SURVEY 2016

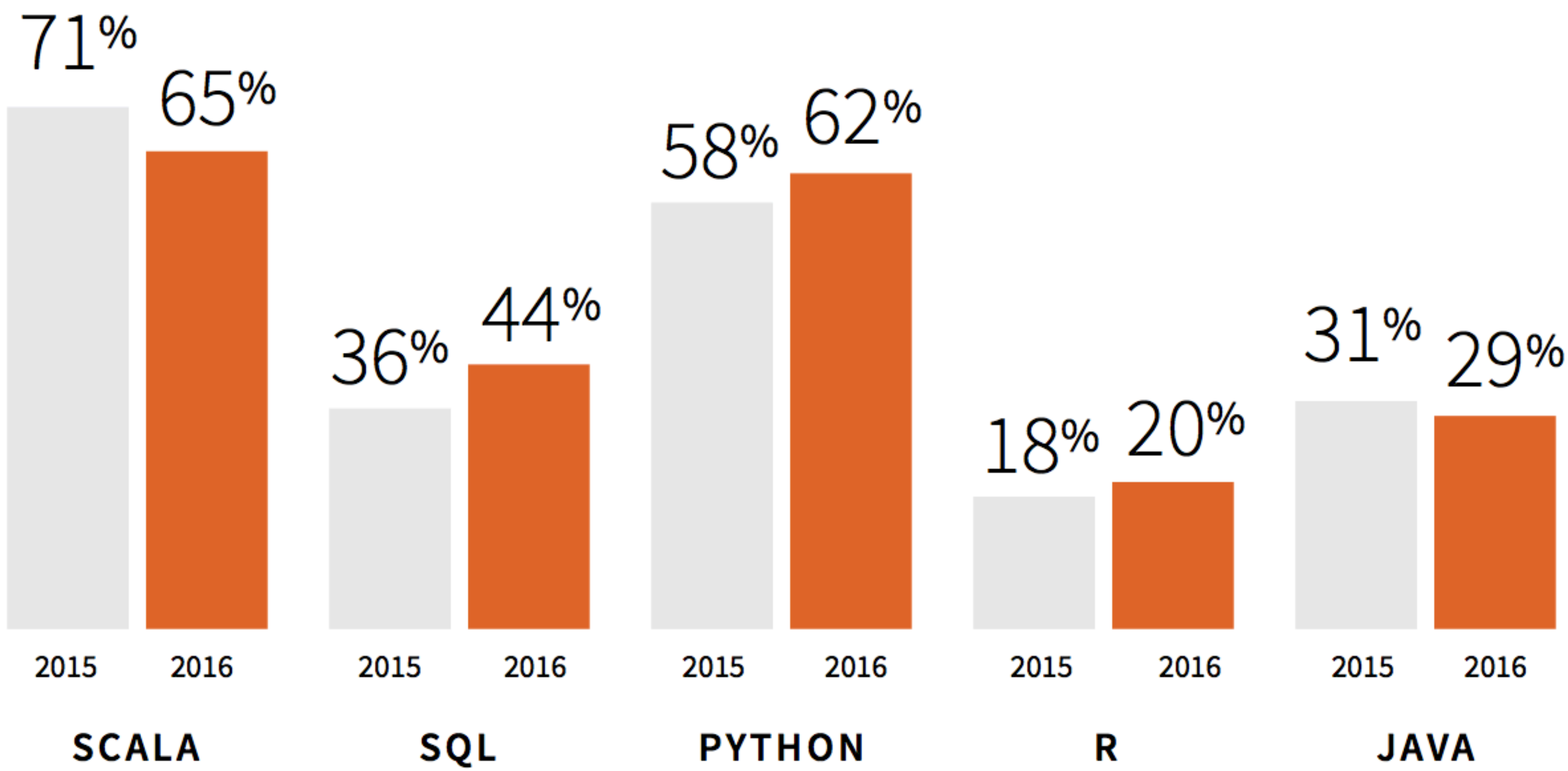




From: Spark User Survey 2016, 1615 respondents from 900 organizations
http://go.databricks.com/2016-spark-survey

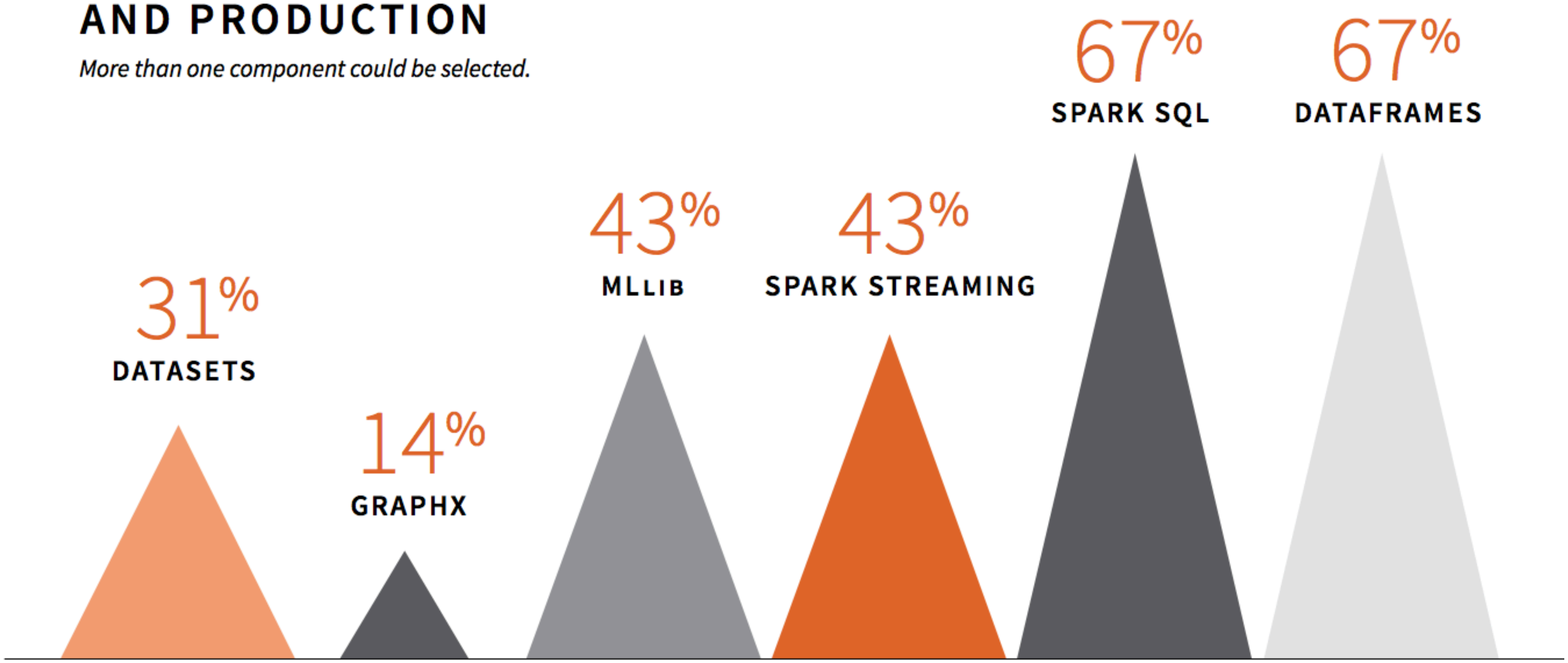
Q: WHICH LANGUAGES DO YOU USE SPARK IN?

% of respondents who use each language (more than one language could be selected)



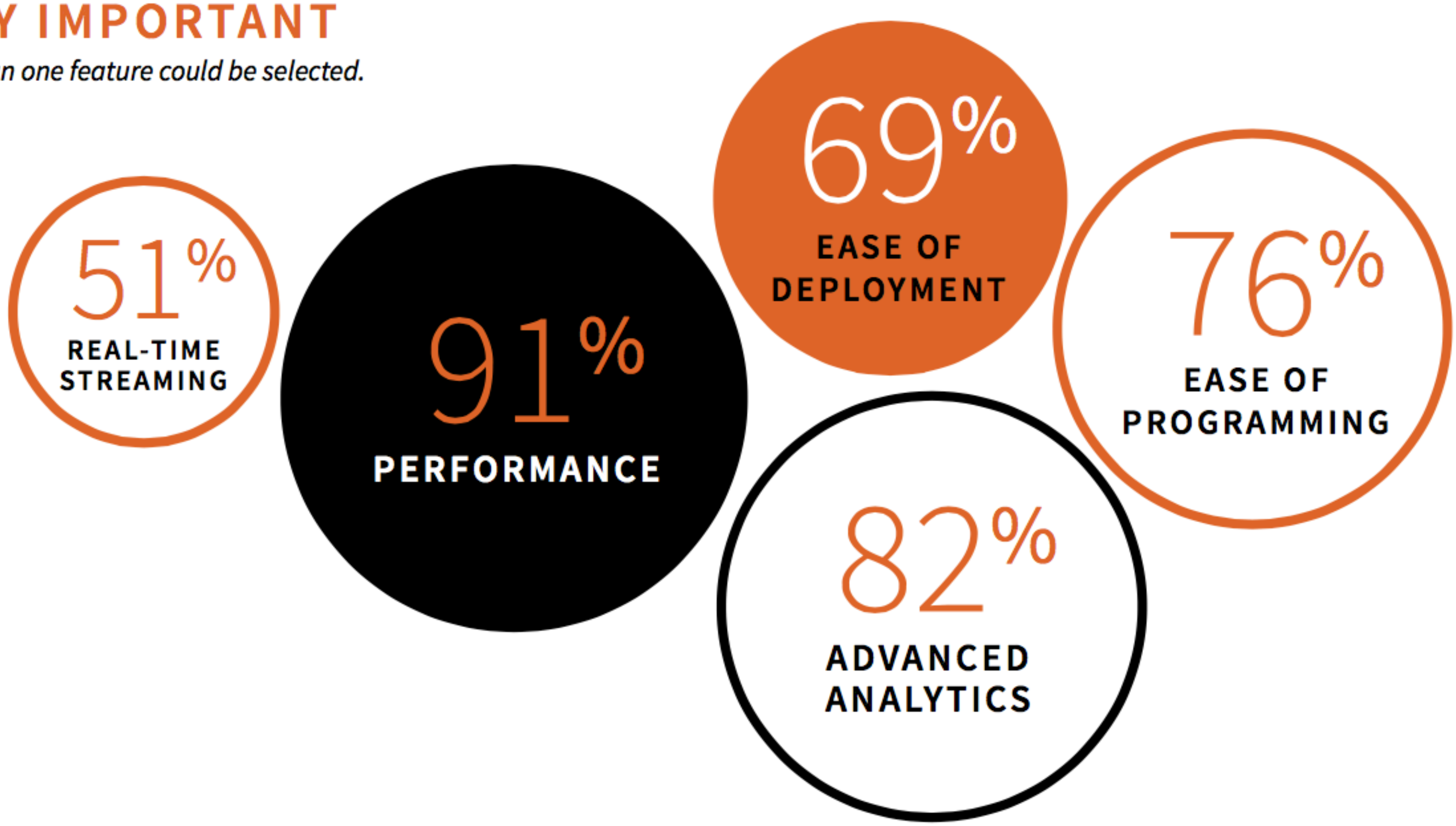
COMPONENTS USED IN PROTOTYPING AND PRODUCTION

More than one component could be selected.



% OF RESPONDENTS WHO CONSIDERED THE FEATURE VERY IMPORTANT

More than one feature could be selected.



Spark Ecosystem Attributes

- Spark focus was initially on
  - Performance + Scalability with Fault Tolerance
- Eventually, ease of development was a key feature
  - especially across multiple modalities: DB, Graph, Stream, etc.
- This was true of most Big Data software of that generation
- Low Latency (streaming) and Deep Learning are also garnering significant attention lately

What's Next?

Innovation in (open source) Big Data Software continues.

Performance, Scalability, and Fault Tolerance remain important, but we face new challenges, including:

Data Science Lifecycle

- Data Acquisition, Integration, Cleaning (i.e., wrangling)
- Data Integration remains a "wicked problem"
- Model Building
- Communicating results, Curation, "Translational Data Science"

Ease of Development and Deployment

- Can leverage database ideas (e.g., declarative query optimization)
- New components for "model serving" and "model management"

"Safe" Data Science

- end-to-end Bias Mitigation
- Security, Ethics and Data Privacy
- Explaining and influencing decisions
- Human-in-the-loop

Thanks and for More Info

Mike Franklin

mjfranklin@uchicago.edu

THE UNIVERSITY OF CHICAGO