How Apache Spark fits into the Big Data landscape

databricks

Spark

What is Spark?

What is Spark?

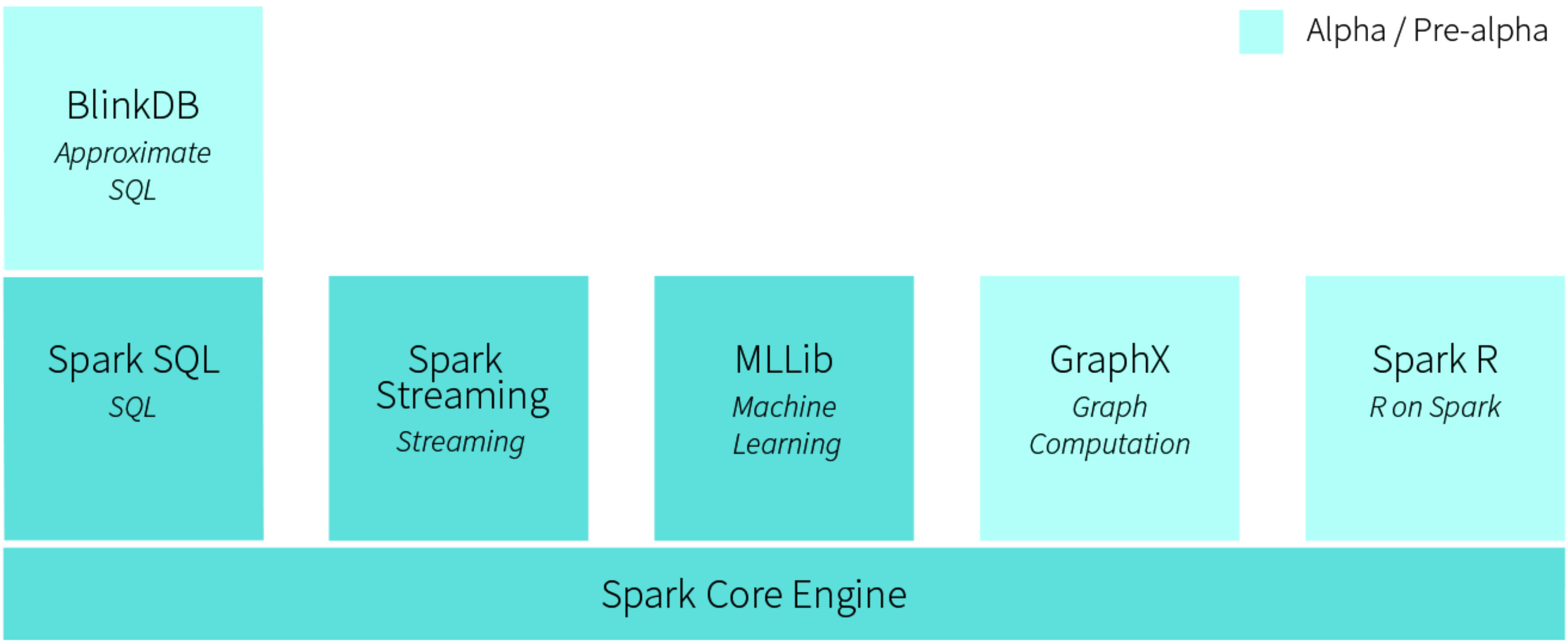
Developed in 2009 at UC Berkeley AMPLab, then open sourced in 2010, Spark has since become one of the largest OSS communities in big data, with over 200 contributors in 50+ organizations

"Organizations that are looking at big data challenges – including collection, ETL, storage, exploration and analytics – should consider Spark for its in-memory performance and the breadth of its model. It supports advanced analytics solutions on Hadoop clusters, including the iterative model required for machine learning and graph analysis."

Gartner, *Advanced Analytics and Data Science* (2014)

spark.apache.org

What is Spark?



What is Spark?

Spark Core is the general execution engine for the Spark platform that other functionality is built atop:

- *in-memory computing* capabilities deliver speed
- *general execution model* supports wide variety of use cases
- *ease of development* – native APIs in Java, Scala, Python (+ SQL, Clojure, R)

What is Spark?

```java
public class WordCount {
  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

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

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length < 2) {
      System.err.println("Usage: wordcount <in> [<in>...] <out>");
      System.exit(2);
    }
    Job job = new Job(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    for (int i = 0; i < otherArgs.length - 1; ++i) {
      FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
    }
    FileOutputFormat.setOutputPath(job,
      new Path(otherArgs[otherArgs.length - 1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

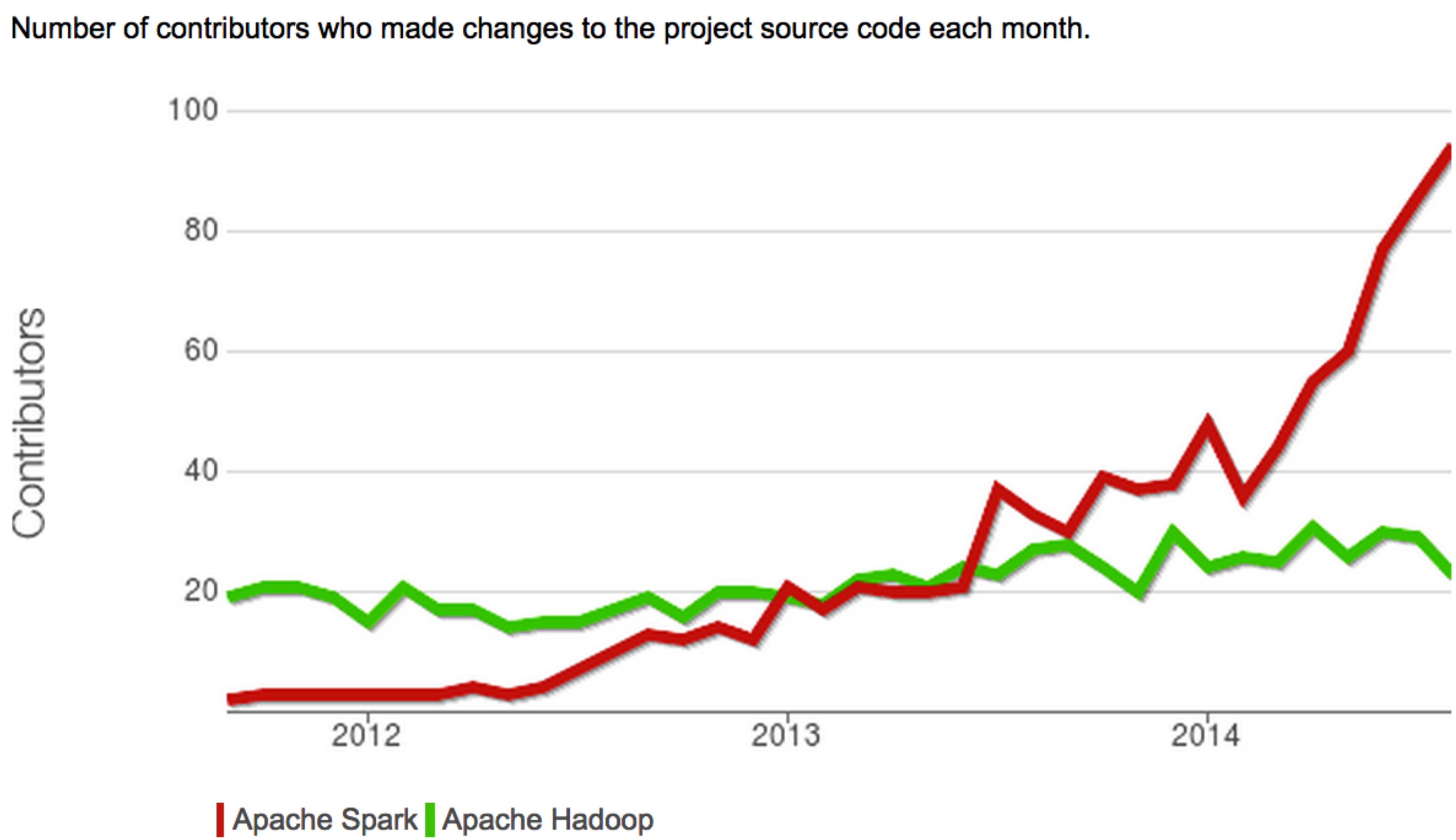
```scala
val f = sc.textFile(inputPath)
val w = f.flatMap(l => l.split(" ")).map(word => (word, 1)).cache()
w.reduceByKey(_ + _).saveAsText(outputPath)
```

WordCount in 3 lines of Spark

WordCount in 50+ lines of Java MR

What is Spark?

Sustained exponential growth, as one of the most active Apache projects **ohloh.net/orgs/apache**



DATASTAX
IBM
MAPR
SAP
Pivotal
Hortonworks
STRATIO
ORACLE
guavus
cloudera
bluedata
MicroStrategy
Typesafe
elasticsearch.
pentaho
ADATAO
DATA INTELLIGENCE FOR ALL
tresata
Atigeo
NUBE
ZOOMDATA
DATA in Motion
Qlik
APERVI
Alpine
DIYOTTA
talend

A Brief History

A Brief History: *Functional Programming for Big Data*

Theory, Eight Decades Ago:
what can be computed?

Alonso Church
wikipedia.org

Haskell Curry
haskell.org

Praxis, Four Decades Ago:
algebra for applicative systems

John Backus
acm.org

David Turner
wikipedia.org

Reality, Two Decades Ago:
machine data from web apps

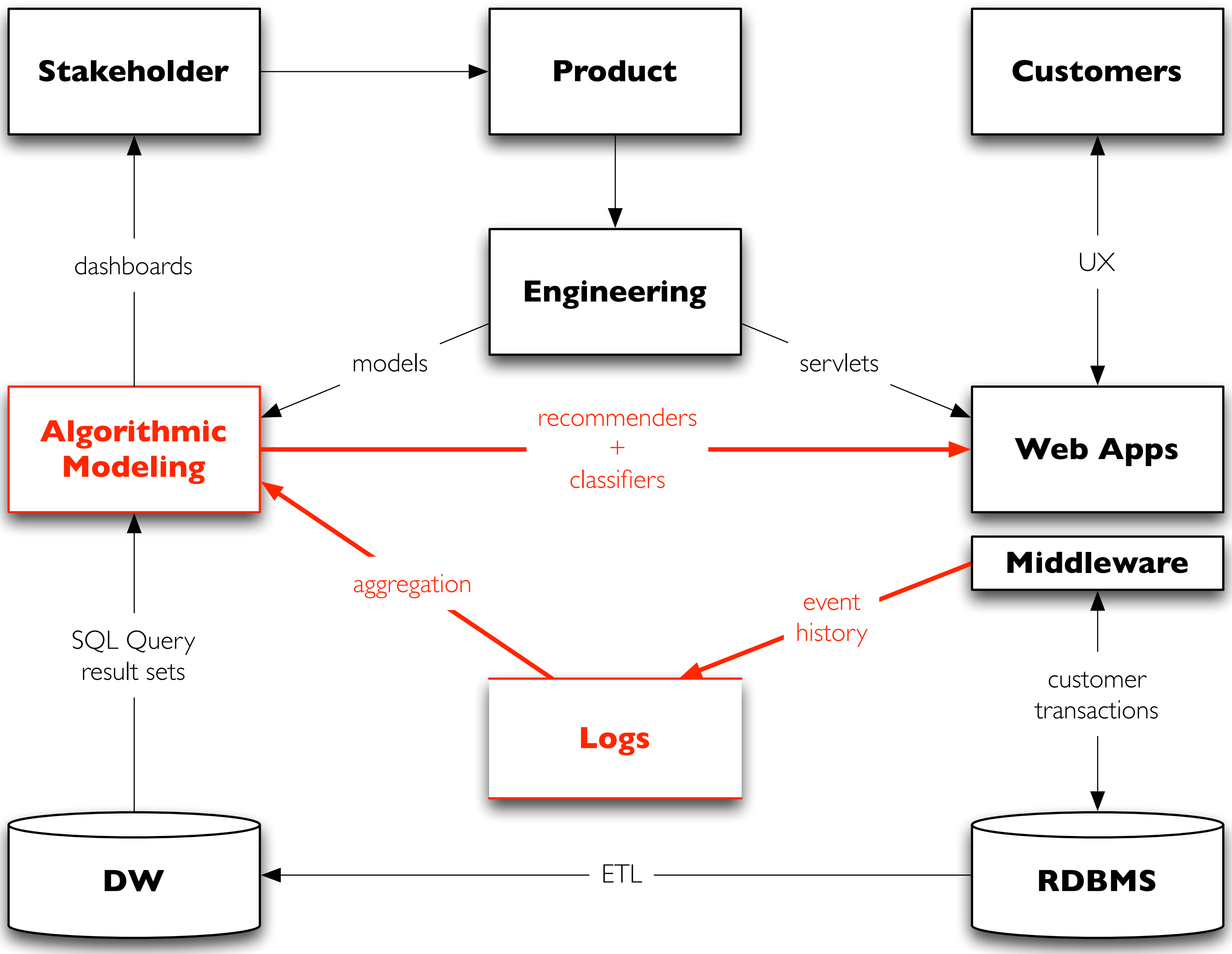
Pattie Maes
MIT Media Lab

amazon.com
ebay
YAHOO!
Google

A Brief History: *Functional Programming for Big Data*

circa late 1990s:
explosive growth e-commerce and machine data implied that workloads could not fit on a single computer anymore…

notable firms led the shift to *horizontal scale-out* on clusters of commodity hardware, especially for machine learning use cases at scale


Stakeholder
Product
Customers
dashboards
Engineering
UX
models
servlets
Algorithmic Modeling
recommenders
+
classifiers
Web Apps
Middleware
aggregation
event
history
SQL Query
result sets
customer
transactions
Logs
DW
ETL
RDBMS

Amazon
"Early Amazon: Splitting the website" – Greg Linden
glinden.blogspot.com/2006/02/early-amazon-splitting-website.html

eBay
"The eBay Architecture" – Randy Shoup, Dan Pritchett
addsimplicity.com/adding_simplicity_an_engi/2006/11/you_scaled_your.html
addsimplicity.com.nyud.net:8080/downloads/eBaySDForum2006-11-29.pdf

Inktomi (**YHOO** Search)
"Inktomi's Wild Ride" – Erik Brewer (0:05:31 ff)
youtu.be/E91oEn1bnXM

Google
"Underneath the Covers at Google" – Jeff Dean (0:06:54 ff)
youtu.be/qsan-GQaeyk
perspectives.mvdirona.com/2008/06/11/JeffDeanOnGoogleInfrastructure.aspx

MIT Media Lab
"Social Information Filtering for Music Recommendation" – Pattie Maes
pubs.media.mit.edu/pubs/papers/32paper.ps
ted.com/speakers/pattie_maes.html

A Brief History: *Functional Programming for Big Data*

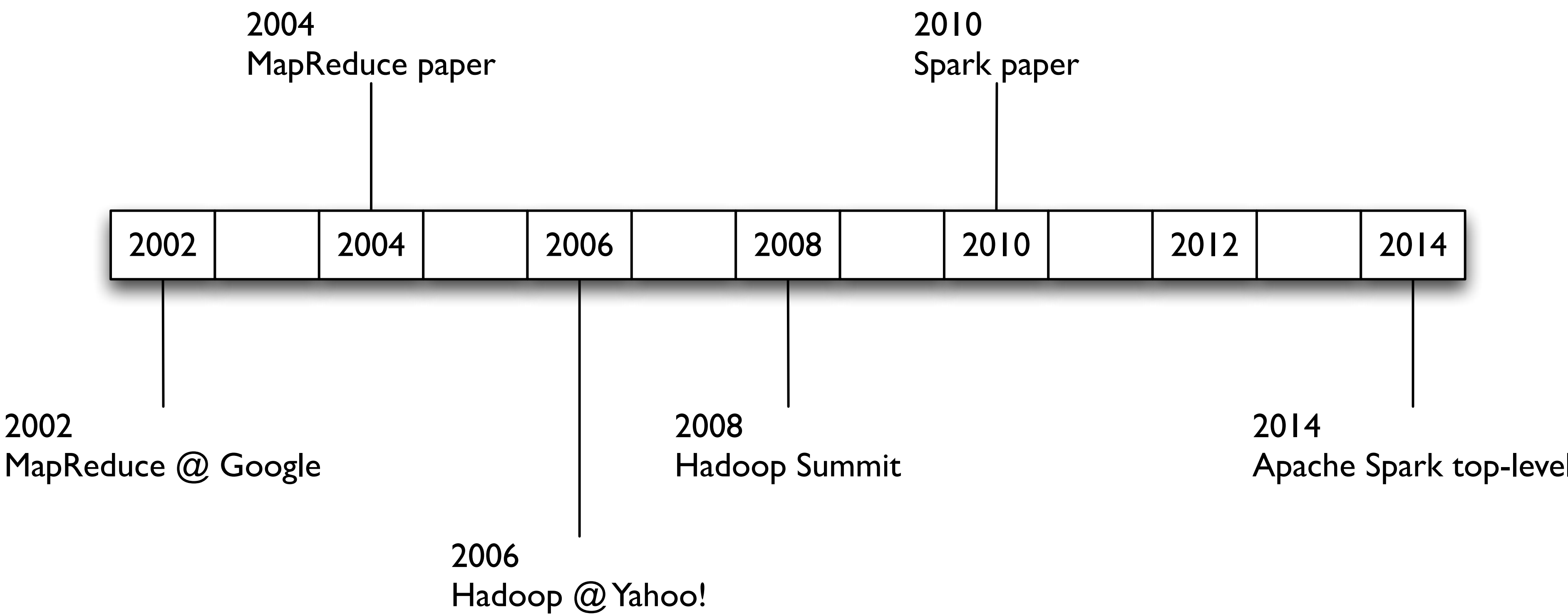
circa 2002:
mitigate risk of large distributed workloads lost due to disk failures on commodity hardware…

Google

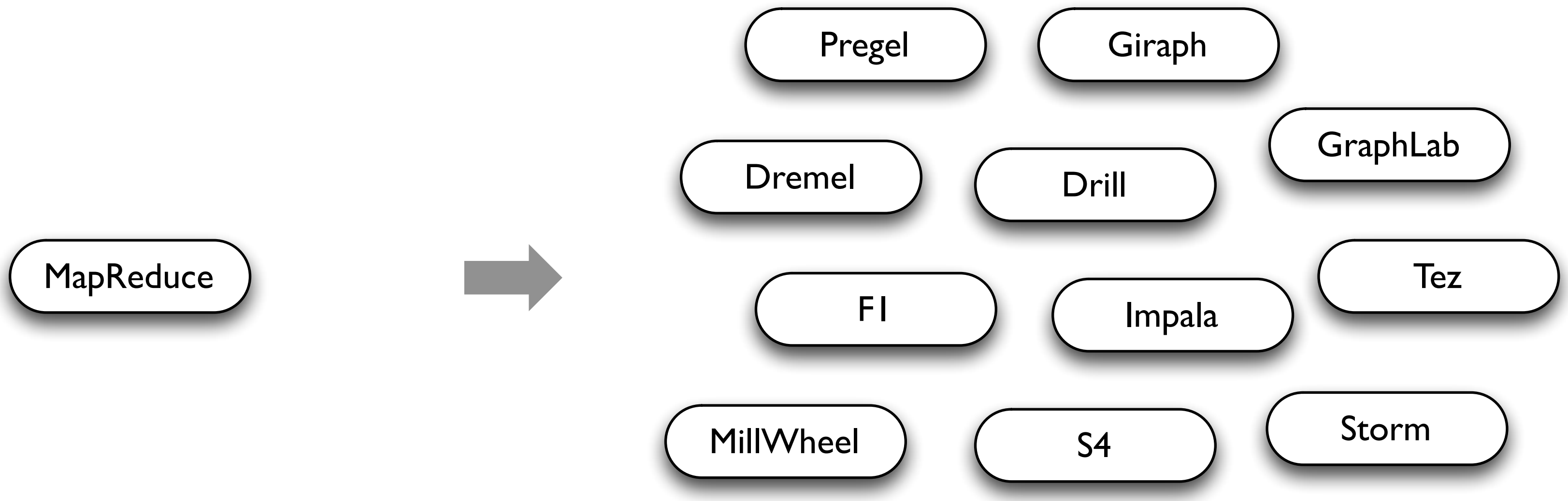
Google File System
Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung
research.google.com/archive/gfs.html

MapReduce: Simplified Data Processing on Large Clusters
Jeffrey Dean, Sanjay Ghemawat
research.google.com/archive/mapreduce.html

A Brief History: *Functional Programming for Big Data*



A Brief History: *Functional Programming for Big Data*



General Batch Processing

Specialized Systems:
iterative, interactive, streaming, graph, etc.

MR doesn't compose well for large applications, and so *specialized systems* emerged as workarounds

A Brief History: *Functional Programming for Big Data*

circa 2010:
a unified engine for enterprise data workflows, based on commodity hardware a decade later…

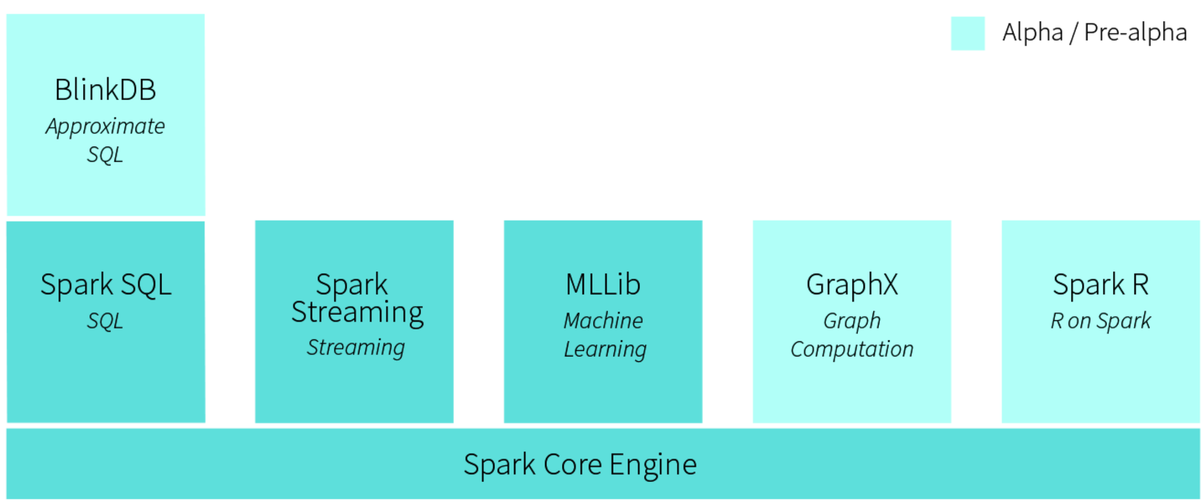
Spark: Cluster Computing with Working Sets
Matei Zaharia, Mosharaf Chowdhury,
Michael Franklin, Scott Shenker, Ion Stoica
people.csail.mit.edu/matei/papers/2010/hotcloud_spark.pdf

Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing
Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave,
Justin Ma, Murphy McCauley, Michael Franklin, Scott Shenker, Ion Stoica
usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf

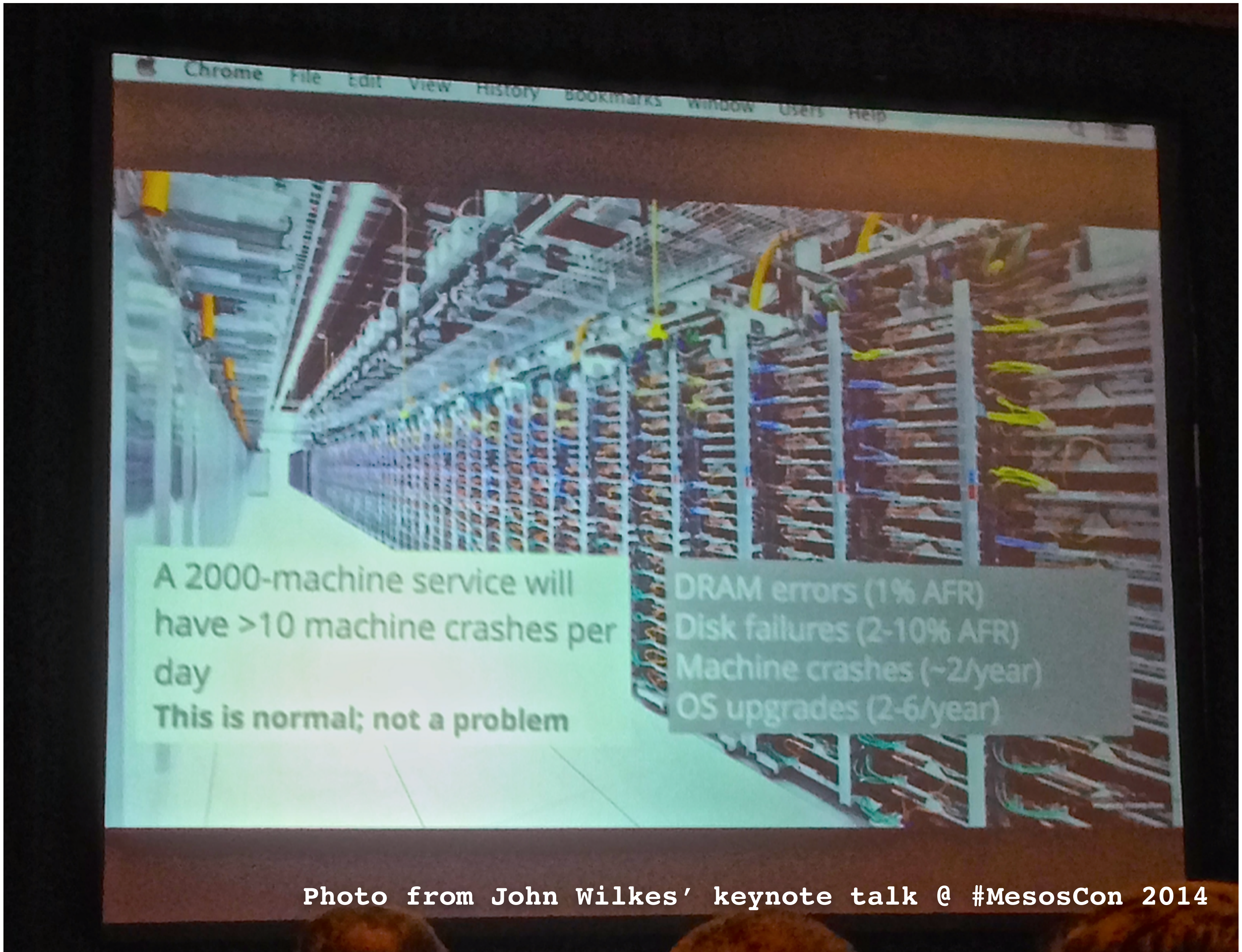
A Brief History: *Functional Programming for Big Data*

In addition to simple *map* and *reduce* operations, Spark supports SQL queries, streaming data, and complex analytics such as machine learning and graph algorithms out-of-the-box.

Better yet, combine these capabilities seamlessly into one integrated workflow…



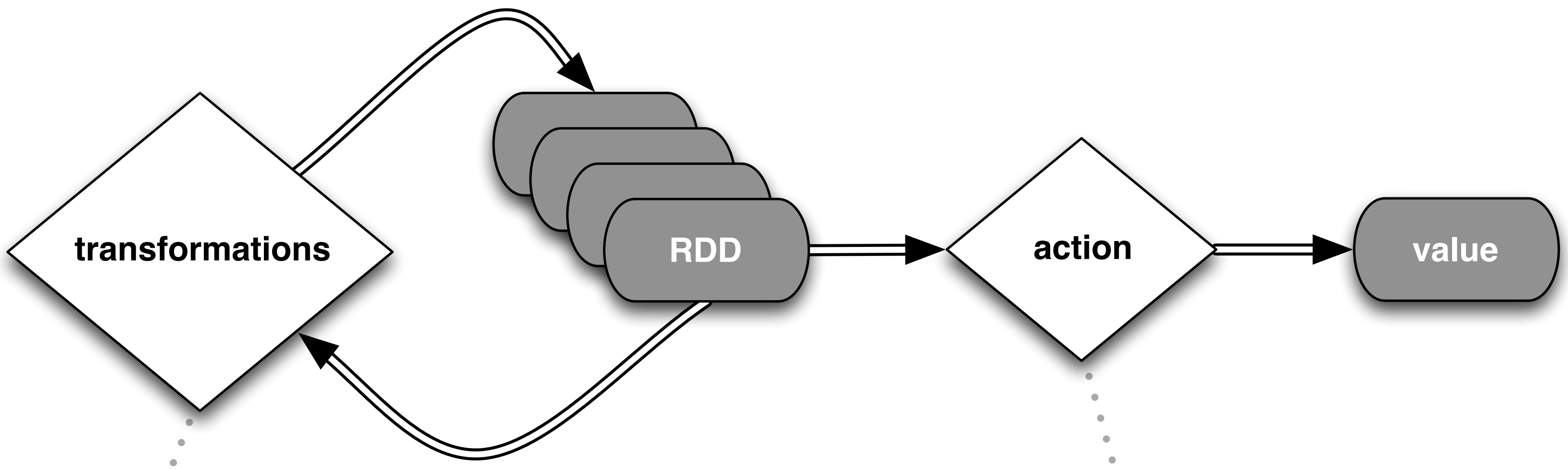
TL;DR: *Generational trade-offs for handling Big Compute*



TL;DR: *Generational trade-offs for handling Big Compute*

Cheap Memory	⇨	**recompute**	(RDD)
Cheap Storage	⇨	**replicate**	(DFS)
Cheap Network	⇨	**reference**	(URI)

TL;DR: *Applicative Systems and Functional Programming – RDDs*




```
// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()
```

```
// action 1
messages.filter(_.contains("mysql")).count()
```

TL;DR: *Big Compute in Applicative Systems, by the numbers…*

1. Express business logic in a preferred native language (*Scala*, *Java*, *Python*, *Clojure*, *SQL*, *R*, etc.) leveraging FP/closures
2. Build a graph of what must be computed
3. Rewrite graph into stages using *graph reduction* to determine how to move/combine predicates, where synchronization barriers are required, what can be computed in parallel, etc. (Wadsworth, Henderson, Turner, et al.)
4. Handle synchronization using **Akka and reactive programming**, with an LRU to manage in-memory working sets (RDDs)
5. Profit

TL;DR: *Big Compute…Implications*

Of course, if you *can* define the structure of workloads in terms of abstract algebra, this all becomes much more interesting – having vast implications on machine learning at scale, IoT, industrial applications, optimization in general, etc., as we retool the industrial plant

However, we'll leave that for another talk…

http://justenoughmath.com/

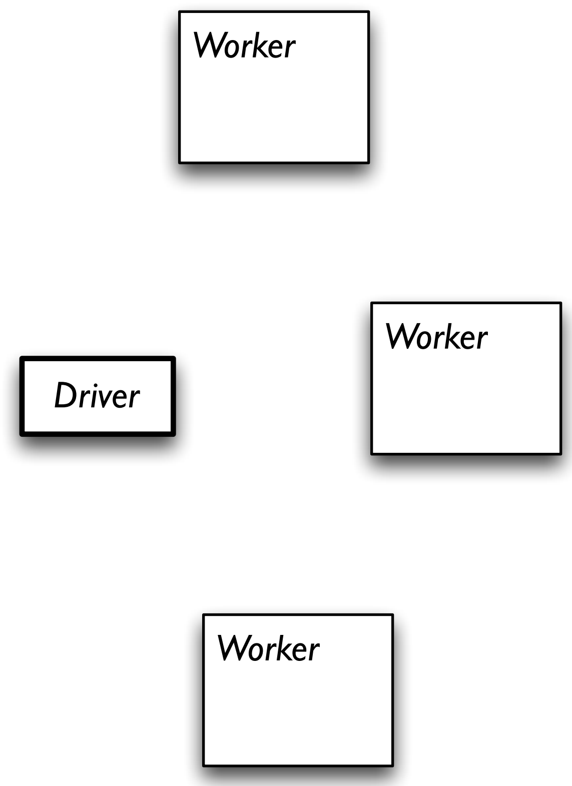
Spark Deconstructed

Spark Deconstructed: *Log Mining Example*

```
// load error messages from a log into memory
// then interactively search for various patterns
// https://gist.github.com/ceteri/8ae5b9509a08c08a1132

// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

Spark Deconstructed: *Log Mining Example*

We start with Spark running on a cluster…
submitting code to be evaluated on it:



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

discussing the other part

Spark Deconstructed: *Log Mining Example*

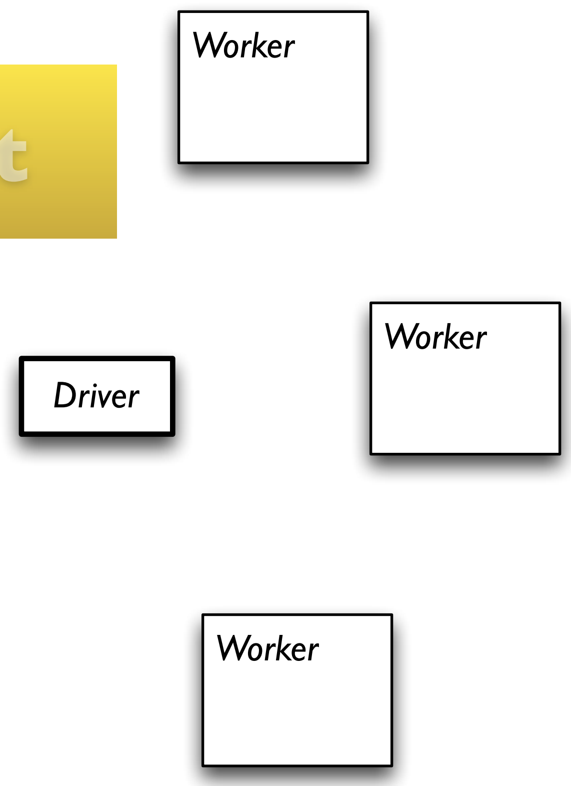
At this point, take a look at the transformed RDD *operator graph*:

```
scala> messages.toDebugString
res5: String =
MappedRDD[4] at map at <console>:16 (3 partitions)
  MappedRDD[3] at map at <console>:16 (3 partitions)
    FilteredRDD[2] at filter at <console>:14 (3 partitions)
      MappedRDD[1] at textFile at <console>:12 (3 partitions)
        HadoopRDD[0] at textFile at <console>:12 (3 partitions)
```

Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

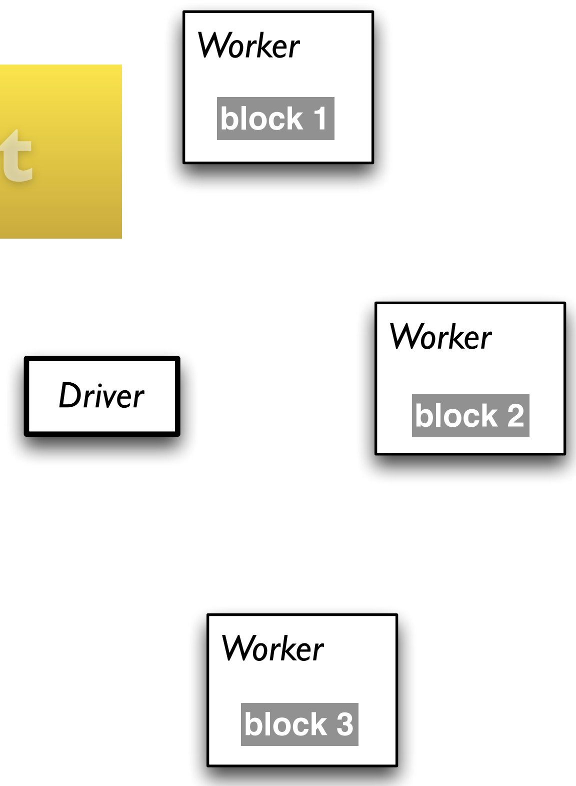
discussing the other part



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

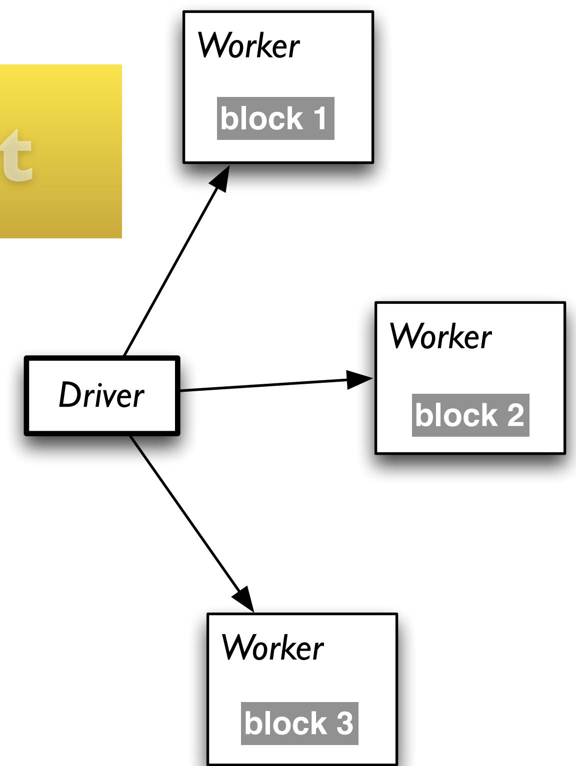
discussing the other part



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

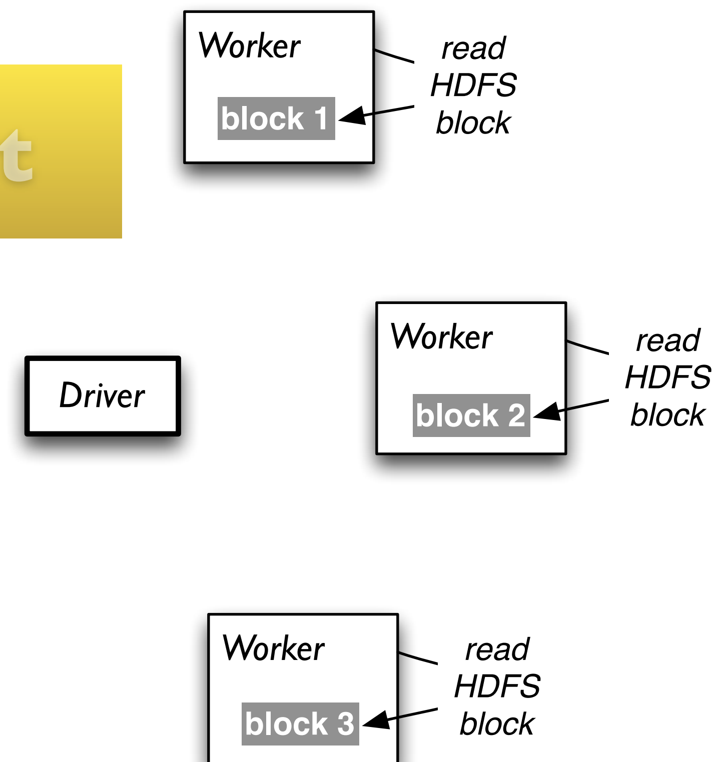
discussing the other part



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

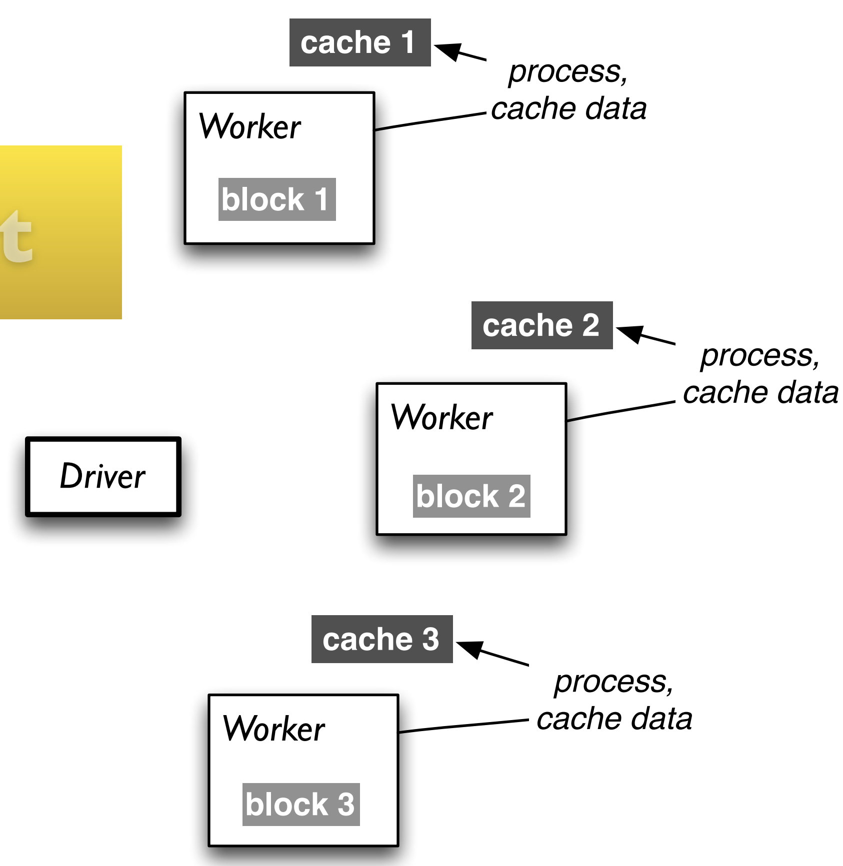
discussing the other part



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

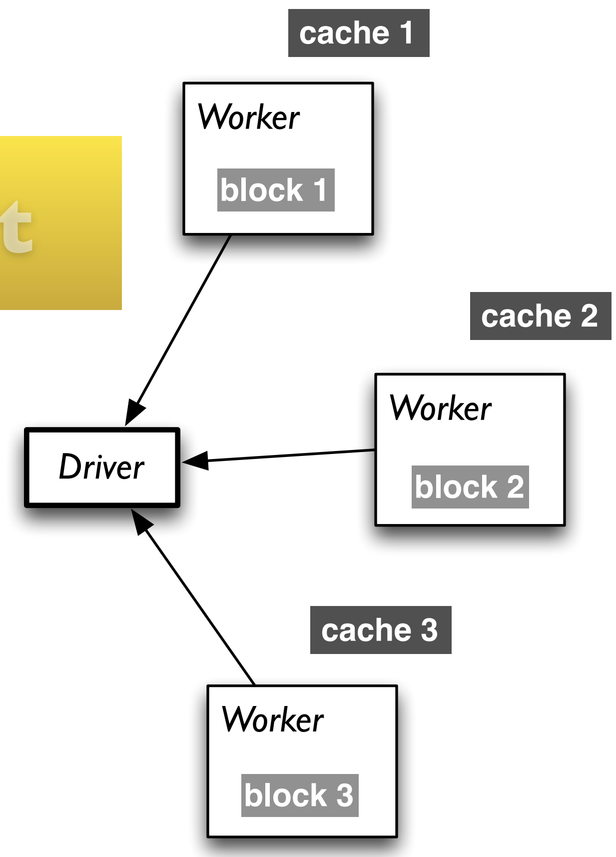
discussing the other part



Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

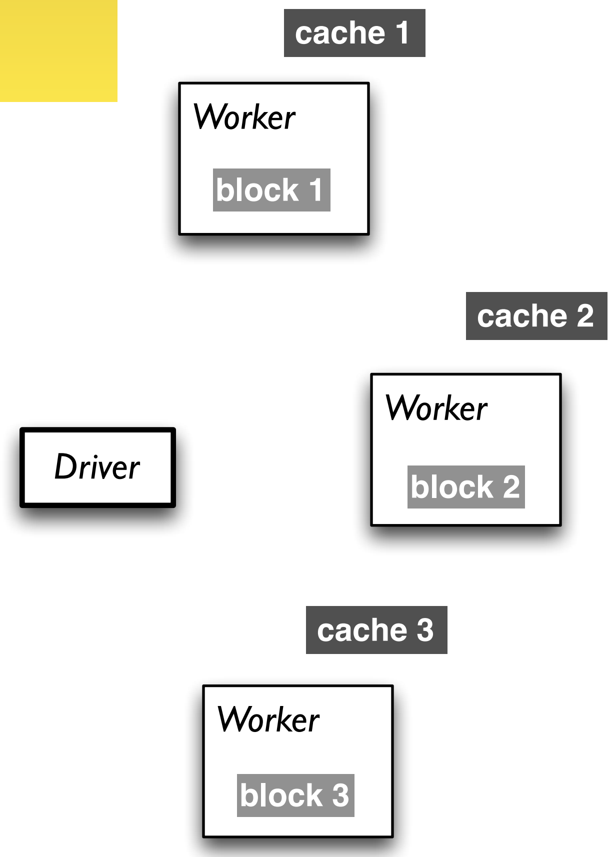
discussing the other part



Spark Deconstructed: *Log Mining Example*

discussing the other part

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

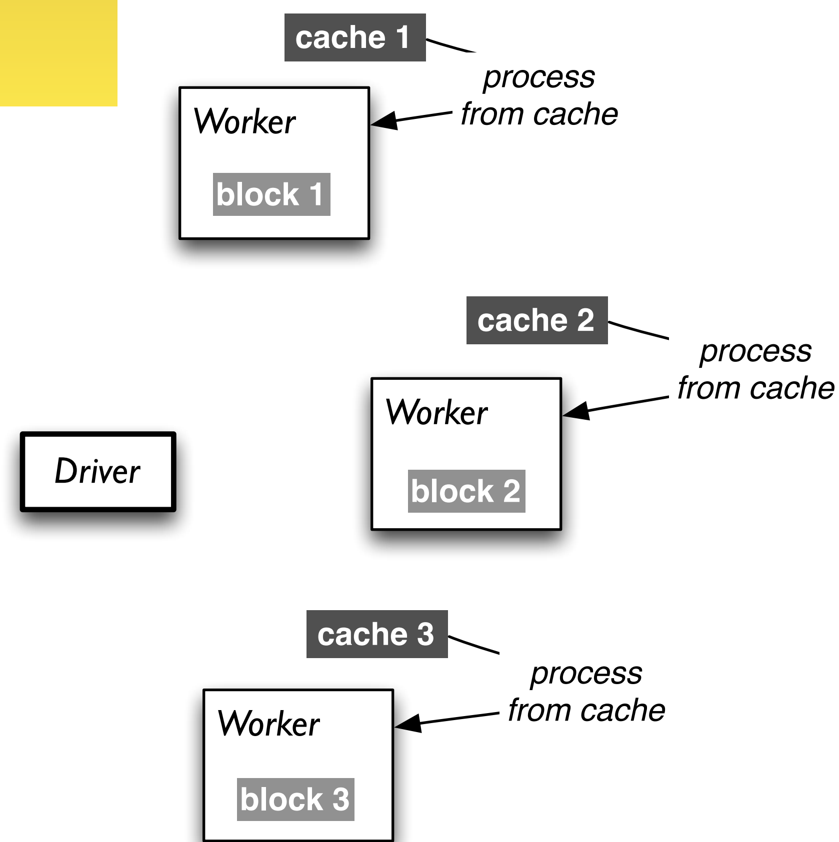


Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()
```

discussing the other part

```
// action 2
messages.filter(_.contains("php")).count()
```

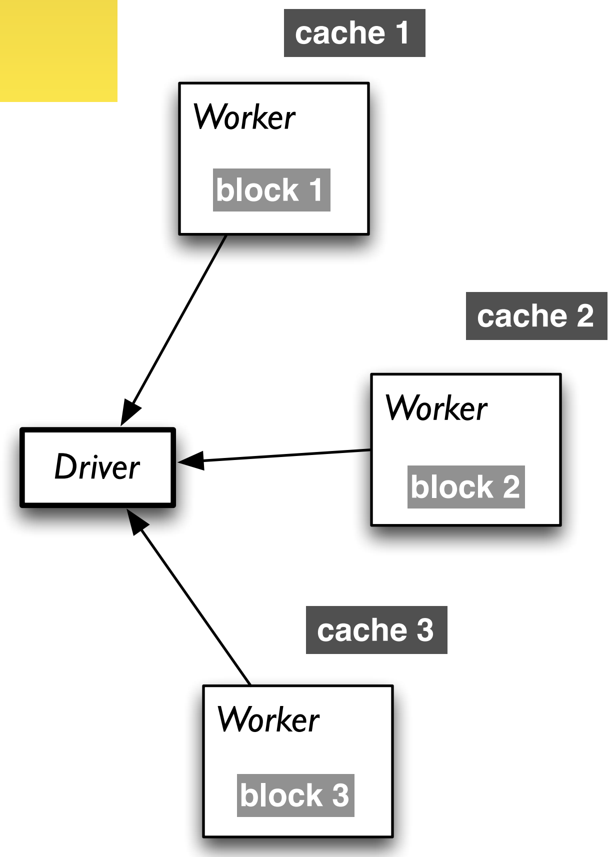


Spark Deconstructed: *Log Mining Example*

```
// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()
```

discussing the other part

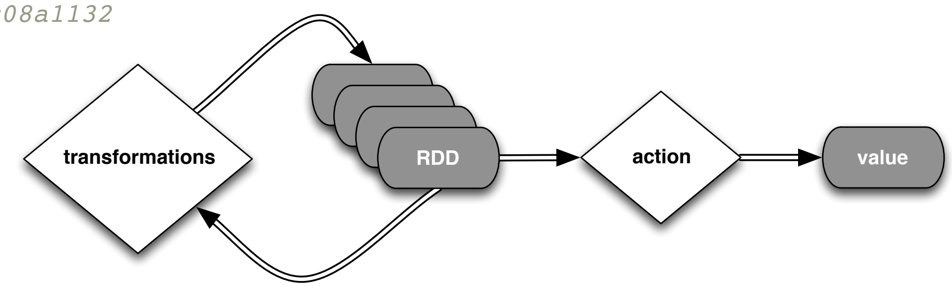
```
// action 2
messages.filter(_.contains("php")).count()
```



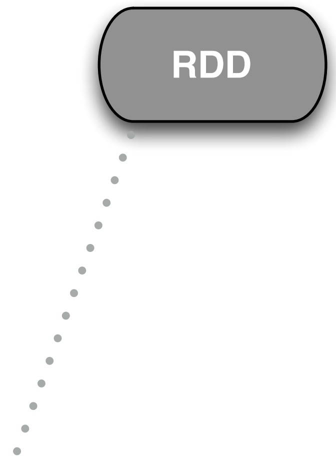
Spark Deconstructed:

Looking at the RDD transformations and actions from another perspective…

```
// load error messages from a log into memory
// then interactively search for various patterns
// https://gist.github.com/ceteri/8ae5b9509a08c08a1132

// base RDD
val lines = sc.textFile("hdfs://...")

// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()

// action 1
messages.filter(_.contains("mysql")).count()

// action 2
messages.filter(_.contains("php")).count()
```

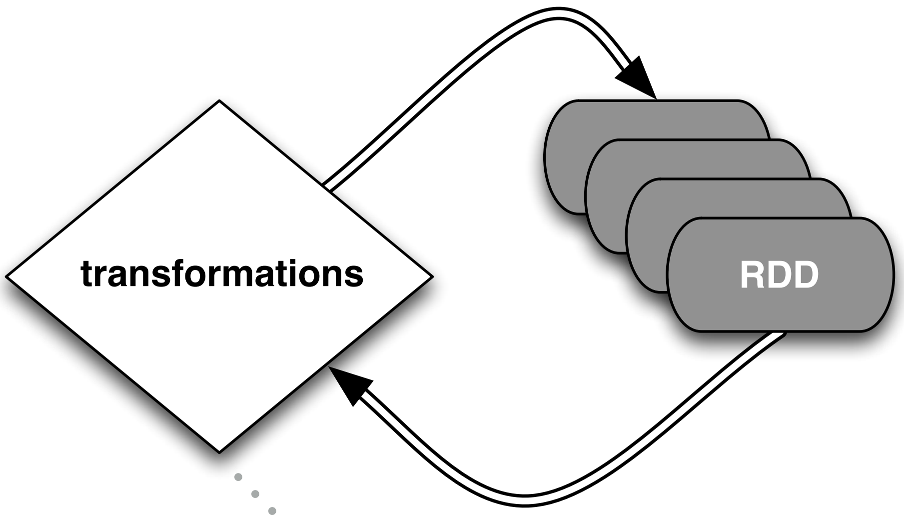


Spark Deconstructed:



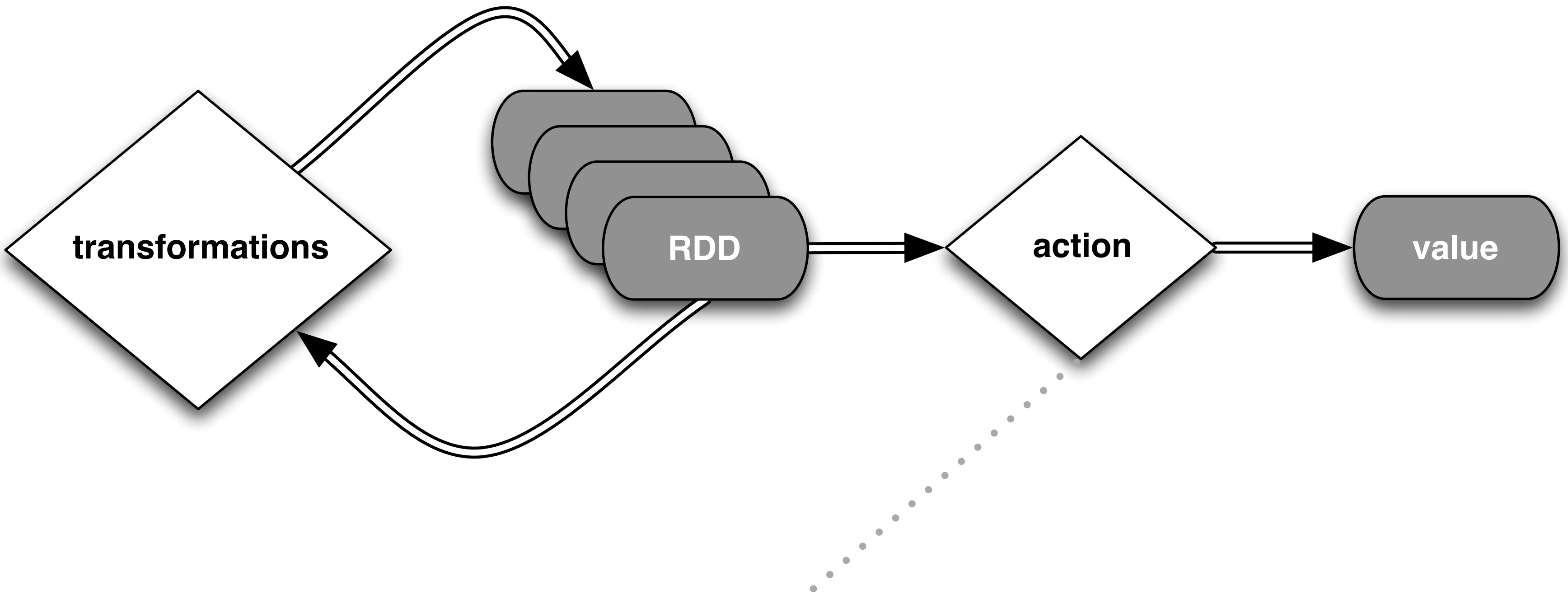

```scala
// base RDD
val lines = sc.textFile("hdfs://...")
```

Spark Deconstructed:




```
// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()
```

Spark Deconstructed:




```
// action 1
messages.filter(_.contains("mysql")).count()
```

Unifying the Pieces

Unifying the Pieces: *Spark SQL*

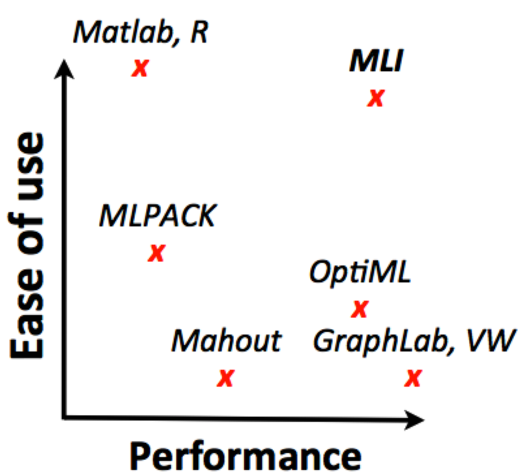
```
// http://spark.apache.org/docs/latest/sql-programming-guide.html

val sqlContext = new org.apache.spark.sql.SQLContext(sc)
import sqlContext._

// define the schema using a case class
case class Person(name: String, age: Int)

// create an RDD of Person objects and register it as a table
val people = sc.textFile("examples/src/main/resources/
people.txt").map(_.split(",")).map(p => Person(p(0), p(1).trim.toInt))

people.registerAsTable("people")

// SQL statements can be run using the SQL methods provided by sqlContext
val teenagers = sql("SELECT name FROM people WHERE age >= 13 AND age <= 19")

// results of SQL queries are SchemaRDDs and support all the
// normal RDD operations…
// columns of a row in the result can be accessed by ordinal
teenagers.map(t => "Name: " + t(0)).collect().foreach(println)
```

Unifying the Pieces: *Spark Streaming*

```
// http://spark.apache.org/docs/latest/streaming-programming-guide.html

import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._

// create a StreamingContext with a SparkConf configuration
val ssc = new StreamingContext(sparkConf, Seconds(10))

// create a DStream that will connect to serverIP:serverPort
val lines = ssc.socketTextStream(serverIP, serverPort)

// split each line into words
val words = lines.flatMap(_.split(" "))

// count each word in each batch
val pairs = words.map(word => (word, 1))
val wordCounts = pairs.reduceByKey(_ + _)

// print a few of the counts to the console
wordCounts.print()

ssc.start()              // start the computation
ssc.awaitTermination()   // wait for the computation to terminate
```

Unifying the Pieces: *MLlib*

```
// http://spark.apache.org/docs/latest/mllib-guide.html

val train_data = // RDD of Vector
val model = KMeans.train(train_data, k=10)

// evaluate the model
val test_data = // RDD of Vector
test_data.map(t => model.predict(t)).collect().foreach(println)
```



MLI: An API for Distributed Machine Learning
Evan Sparks, **Ameet Talwalkar**, et al.
International Conference on Data Mining (2013)
http://arxiv.org/abs/1310.5426

Unifying the Pieces: *GraphX*

```
// http://spark.apache.org/docs/latest/graphx-programming-guide.html

import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD

case class Peep(name: String, age: Int)

val vertexArray = Array(
  (1L, Peep("Kim", 23)), (2L, Peep("Pat", 31)),
  (3L, Peep("Chris", 52)), (4L, Peep("Kelly", 39)),
  (5L, Peep("Leslie", 45))
  )
val edgeArray = Array(
  Edge(2L, 1L, 7), Edge(2L, 4L, 2),
  Edge(3L, 2L, 4), Edge(3L, 5L, 3),
  Edge(4L, 1L, 1), Edge(5L, 3L, 9)
  )

val vertexRDD: RDD[(Long, Peep)] = sc.parallelize(vertexArray)
val edgeRDD: RDD[Edge[Int]] = sc.parallelize(edgeArray)
val g: Graph[Peep, Int] = Graph(vertexRDD, edgeRDD)

val results = g.triplets.filter(t => t.attr > 7)

for (triplet <- results.collect) {
  println(s"${triplet.srcAttr.name} loves ${triplet.dstAttr.name}")
}
```

Unifying the Pieces: *Summary*

Demo, if time permits (perhaps in the hallway):

Twitter Streaming Language Classifier
databricks.gitbooks.io/databricks-spark-reference-applications/twitter_classifier/README.html

For many more Spark resources online, check:
databricks.com/spark-training-resources

TL;DR: *Engineering is about costs*

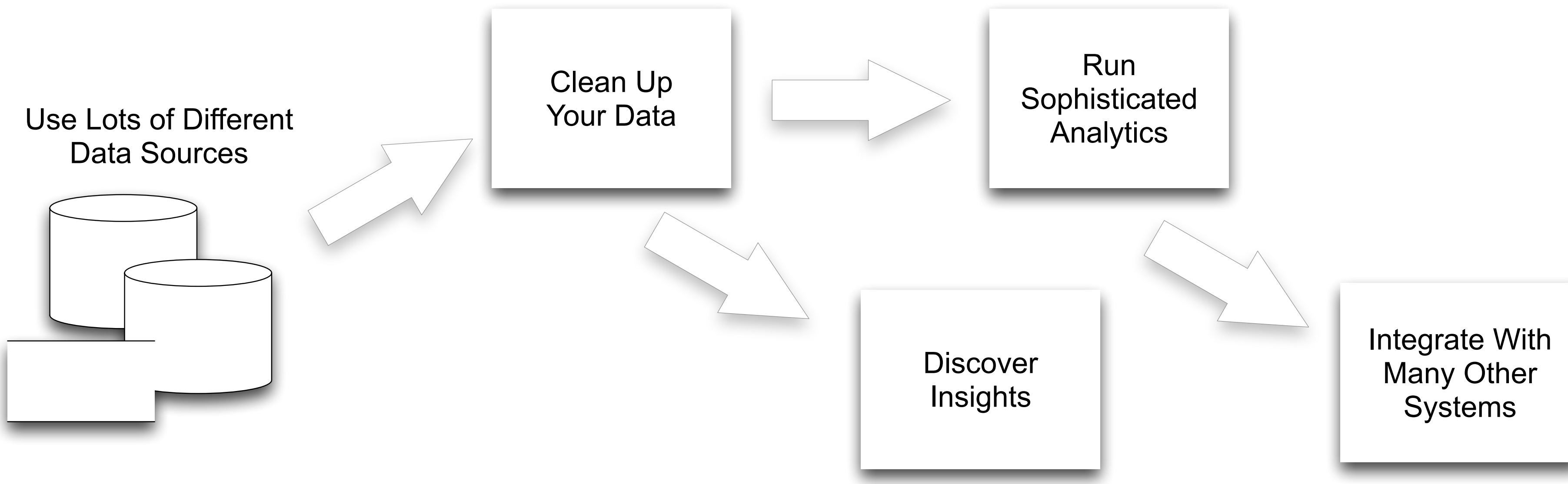
Sure, maybe you'll squeeze slightly better performance by using many specialized systems…

However, putting on an Eng Director hat, would you be also prepared to pay the corresponding costs of:

- learning curves for your developers across several different frameworks
- ops for several different kinds of clusters
- maintenance + troubleshooting mission-critical apps across several systems
- tech-debt for OSS that ignores the math (80 yrs!) plus the fundamental h/w trade-offs

Integrations

Spark Integrations:



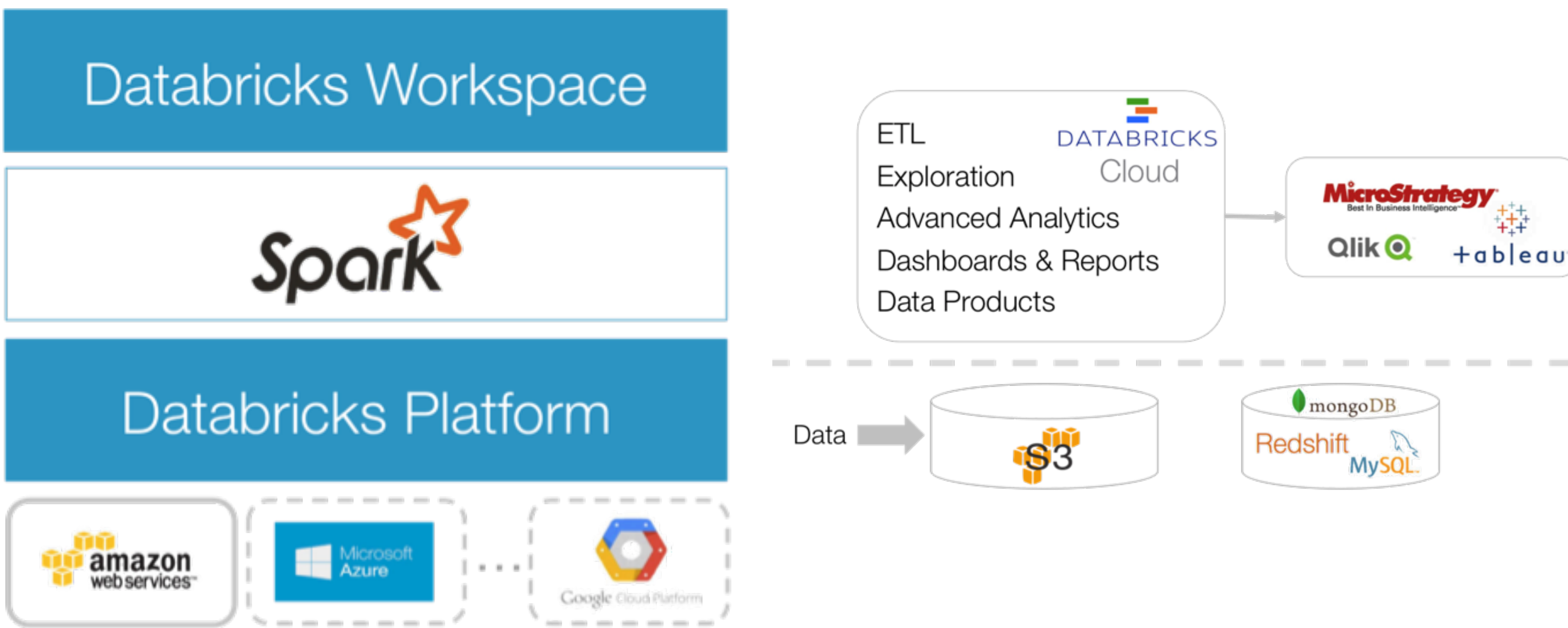
cloud-based notebooks… ETL… the Hadoop ecosystem…
widespread use of PyData… advanced analytics in streaming…
rich custom search… web apps for data APIs…
low-latency + multi-tenancy…

Spark Integrations: *Unified platform for building Big Data pipelines*

Databricks Cloud

databricks.com/blog/2014/07/14/databricks-cloud-making-big-data-easy.html

youtube.com/watch?v=dJQ5lV5Tldw#t=883



Spark Integrations: *The proverbial Hadoop ecosystem*

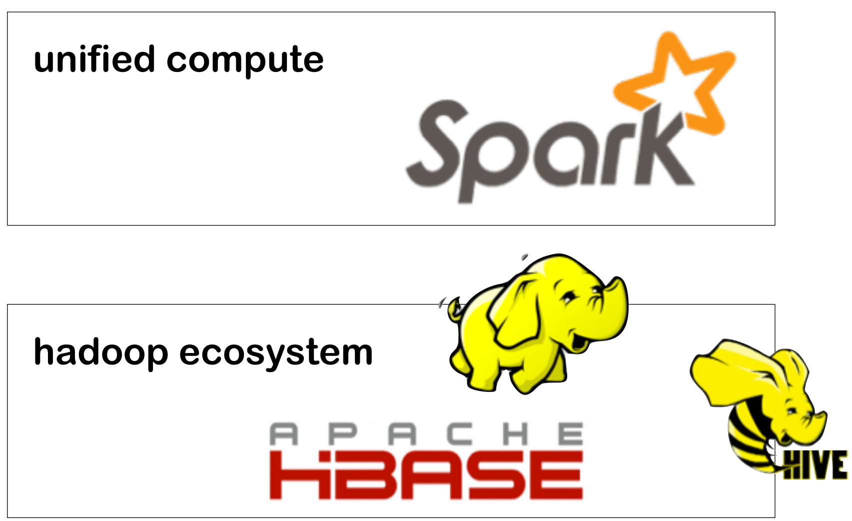
Spark + Hadoop + HBase + etc.

mapr.com/products/apache-spark

vision.cloudera.com/apache-spark-in-the-apache-hadoop-ecosystem/

hortonworks.com/hadoop/spark/

databricks.com/blog/2014/05/23/pivotal-hadoop-integrates-the-full-apache-spark-stack.html



Spark Integrations: *Leverage widespread use of Python*

Spark + PyData

spark-summit.org/2014/talk/A-platform-for-large-scale-neuroscience

cwiki.apache.org/confluence/display/SPARK/PySpark+Internals

unified compute

Spark

Py Data

Anaconda

pandas

matplotlib

IP[y]:

scikits learn

NumPy

Spark Integrations: *Advanced analytics for streaming use cases*

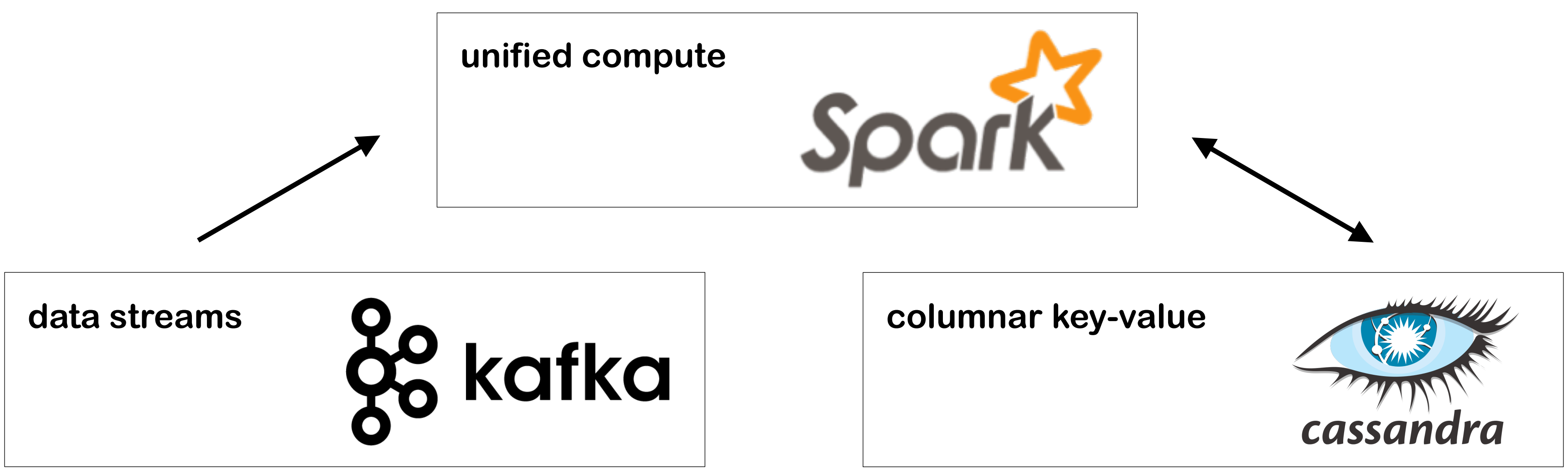
Kafka + Spark + Cassandra

datastax.com/documentation/datastax_enterprise/4.5/datastax_enterprise/spark/sparkIntro.html

http://helenaedelson.com/?p=991

github.com/datastax/spark-cassandra-connector

github.com/dibbhatt/kafka-spark-consumer



Spark Integrations: *Rich search, immediate insights*

Spark + ElasticSearch

databricks.com/blog/2014/06/27/application-spotlight-elasticsearch.html

elasticsearch.org/guide/en/elasticsearch/hadoop/current/spark.html

spark-summit.org/2014/talk/streamlining-search-indexing-using-elastic-search-and-spark

unified compute

document search

Spark Integrations: *Building data APIs with web apps*

Spark + Play

typesafe.com/blog/apache-spark-and-the-typesafe-reactive-platform-a-match-made-in-heaven

unified compute

web apps

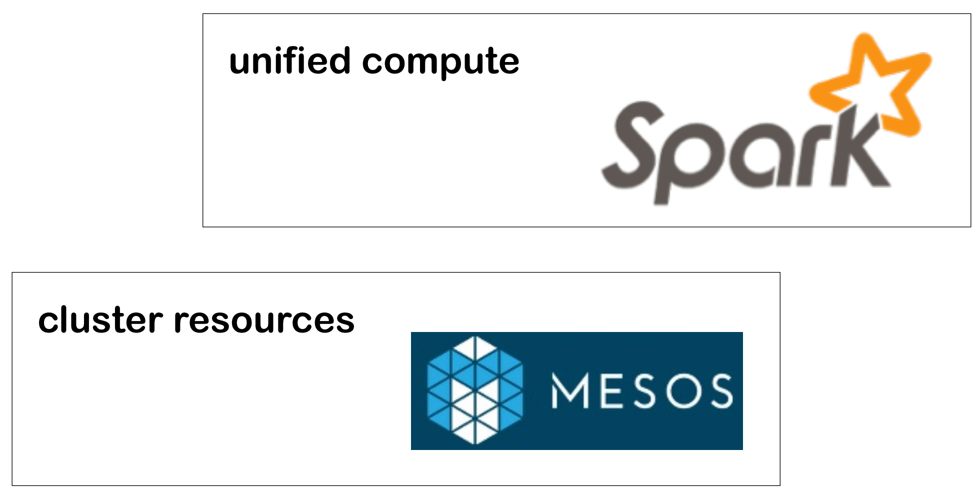
Spark Integrations: *The case for multi-tenancy*

Spark + Mesos

spark.apache.org/docs/latest/running-on-mesos.html

+ Mesosphere + Google Cloud Platform

ceteri.blogspot.com/2014/09/spark-atop-mesos-on-google-cloud.html

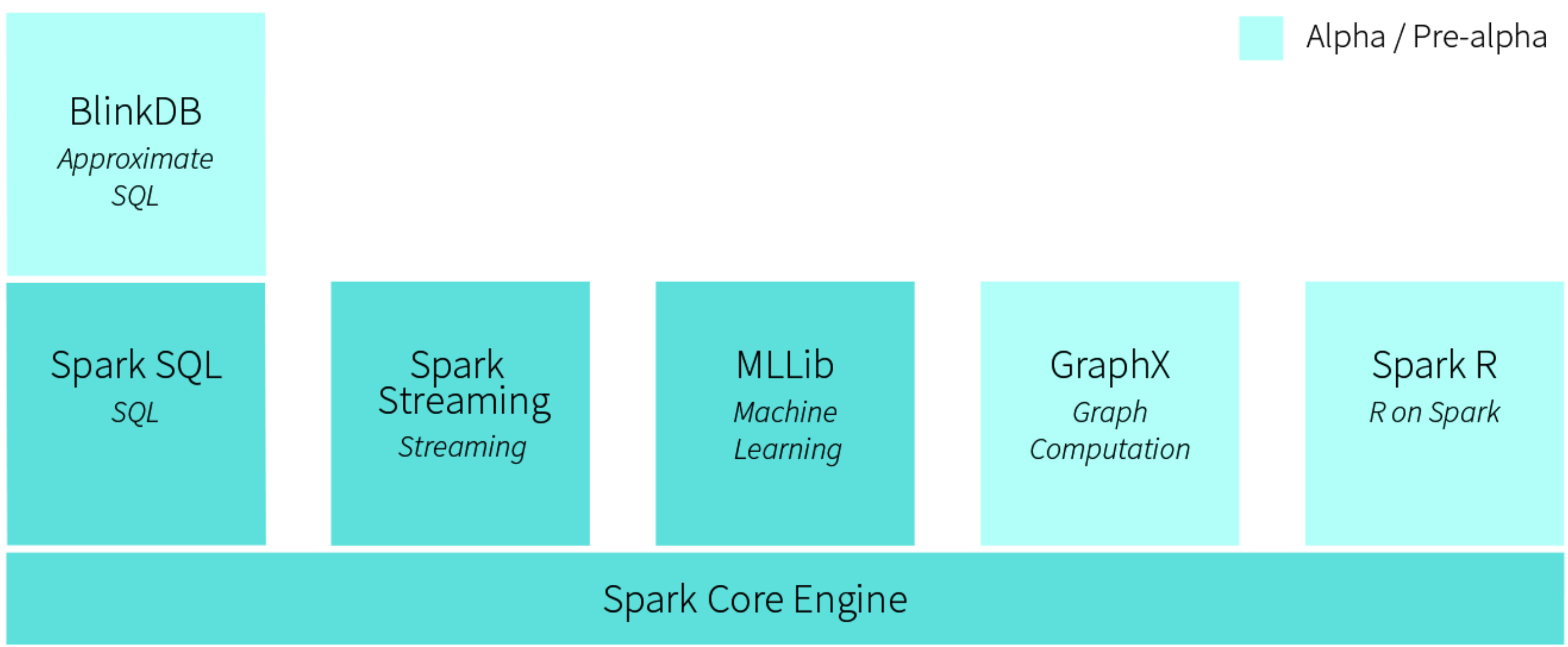


Advanced Topics

Advanced Topics:

Other **BDAS** projects running atop Spark for graphs, sampling, and memory sharing:

- **BlinkDB**
- **Tachyon**



BlinkDB **blinkdb.org/**

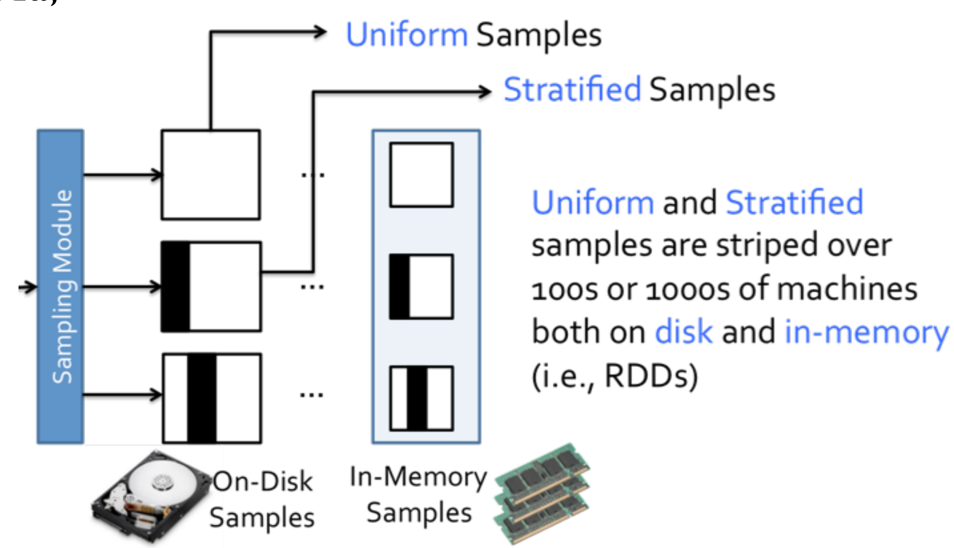
massively parallel, approximate query engine for running interactive SQL queries on large volumes of data

- allows users to trade-off query accuracy for response time
- enables interactive queries over massive data by running queries on data samples
- presents results annotated with meaningful error bars

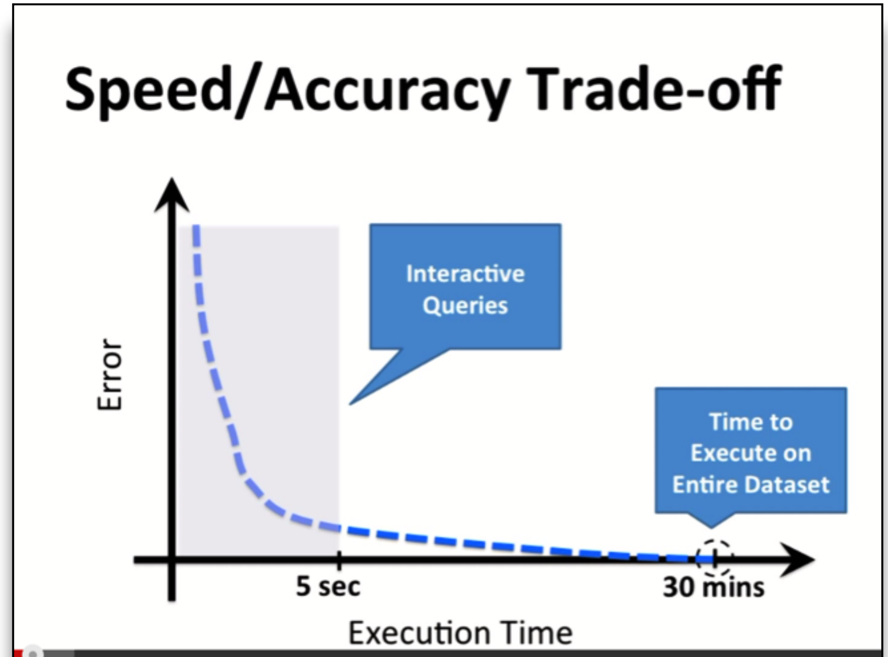
Advanced Topics: *BlinkDB*

"Our experiments on a 100 node cluster show that BlinkDB can answer queries on up to 17 TBs of data in less than 2 seconds (over 200 x faster than Hive), within an error of 2-10%."

BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data
Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, Ion Stoica
EuroSys (2013)
dl.acm.org/citation.cfm?id=2465355

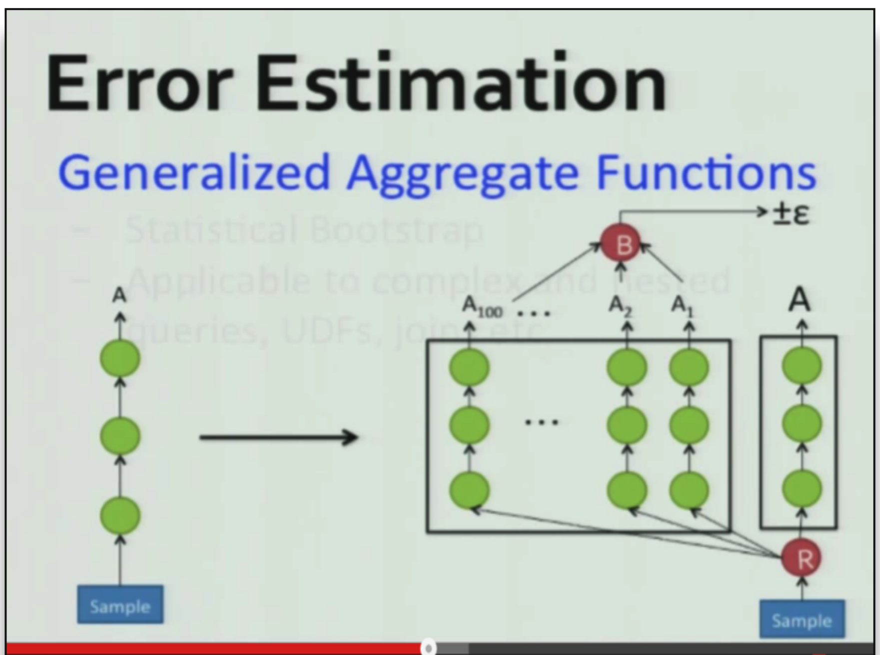


Advanced Topics: *BlinkDB*



Introduction to using BlinkDB
Sameer Agarwal
youtu.be/Pc8_EM9PKqY

Deep Dive into BlinkDB
Sameer Agarwal
youtu.be/WoTTbdk0kCA



Advanced Topics: *Tachyon*

Tachyon **tachyon-project.org/**

- fault tolerant distributed file system enabling reliable file sharing at memory-speed across cluster frameworks
- achieves high performance by leveraging lineage information and using memory aggressively
- caches working set files in memory thereby avoiding going to disk to load datasets that are frequently read
- enables different jobs/queries and frameworks to access cached files at memory speed

Advanced Topics: *Tachyon*

More details:

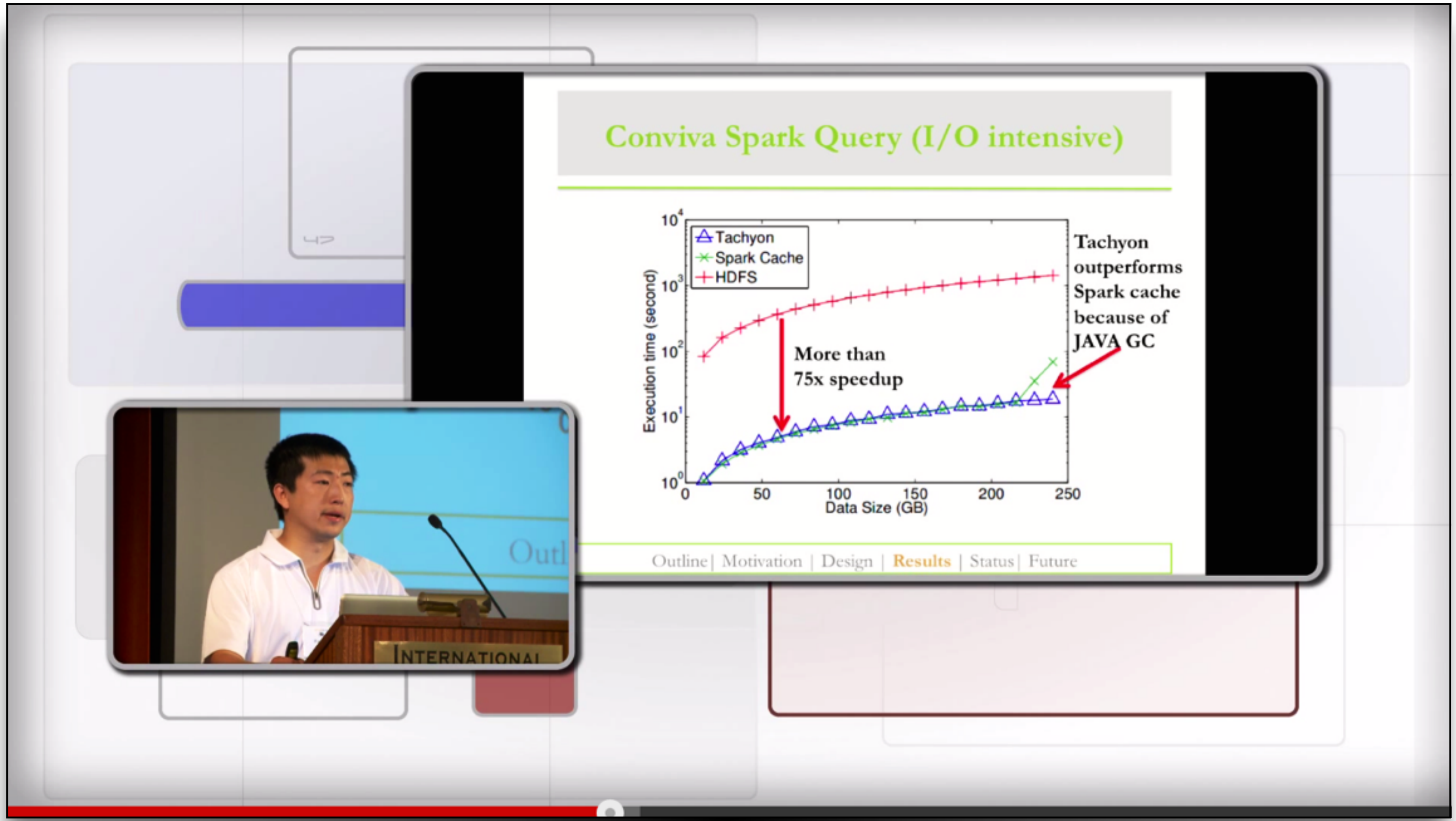
tachyon-project.org/Command-Line-Interface.html

ampcamp.berkeley.edu/big-data-mini-course/tachyon.html

timothysc.github.io/blog/2014/02/17/bdas-tachyon/

Advanced Topics: *Tachyon*

Introduction to Tachyon
Haoyuan Li
youtu.be/4lMAsd2LNEE



Case Studies

Summary: *Case Studies*

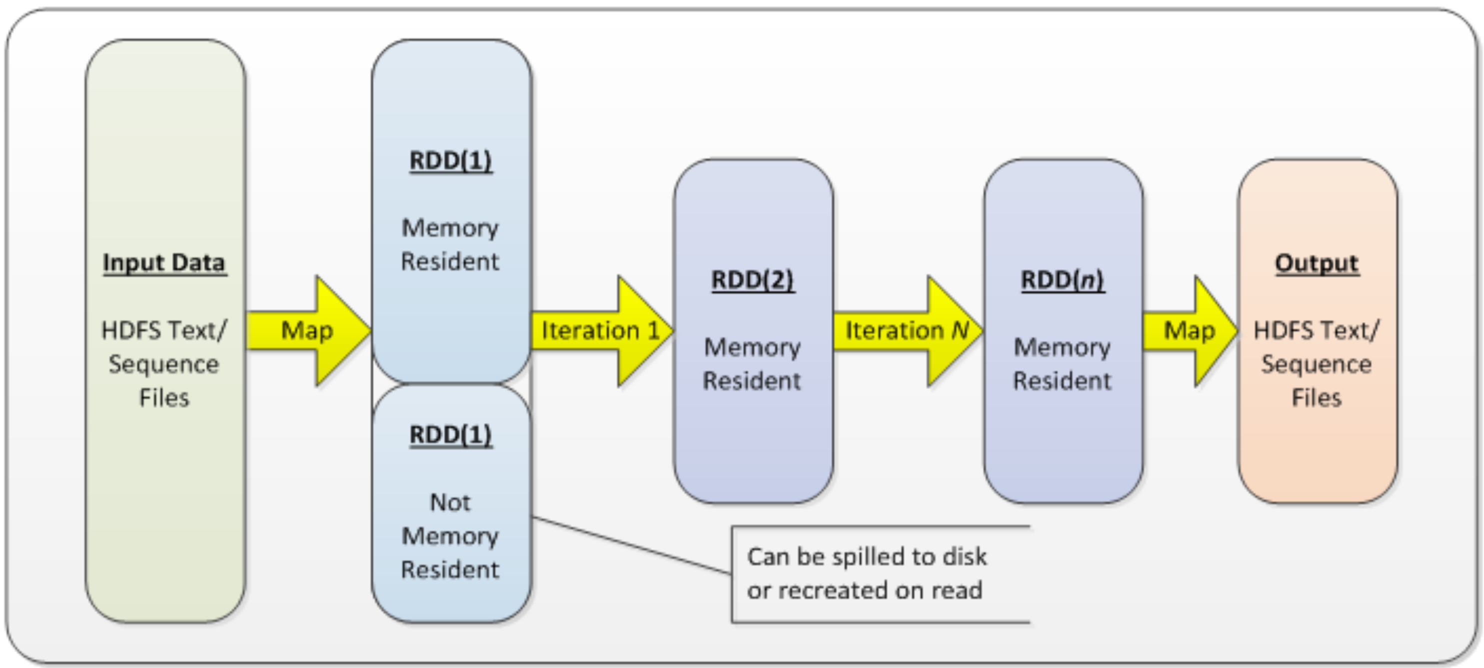
Spark at Twitter: Evaluation & Lessons Learnt
Sriram Krishnan
slideshare.net/krishflix/seattle-spark-meetup-spark-at-twitter

- Spark can be more interactive, efficient than MR
  - *Support for iterative algorithms and caching*
  - *More generic than traditional MapReduce*
- Why is Spark faster than Hadoop MapReduce?
  - *Fewer I/O synchronization barriers*
  - *Less expensive shuffle*
  - *More complex the DAG, greater the performance improvement*

Summary: *Case Studies*

Using Spark to Ignite Data Analytics

ebaytechblog.com/2014/05/28/using-spark-to-ignite-data-analytics/

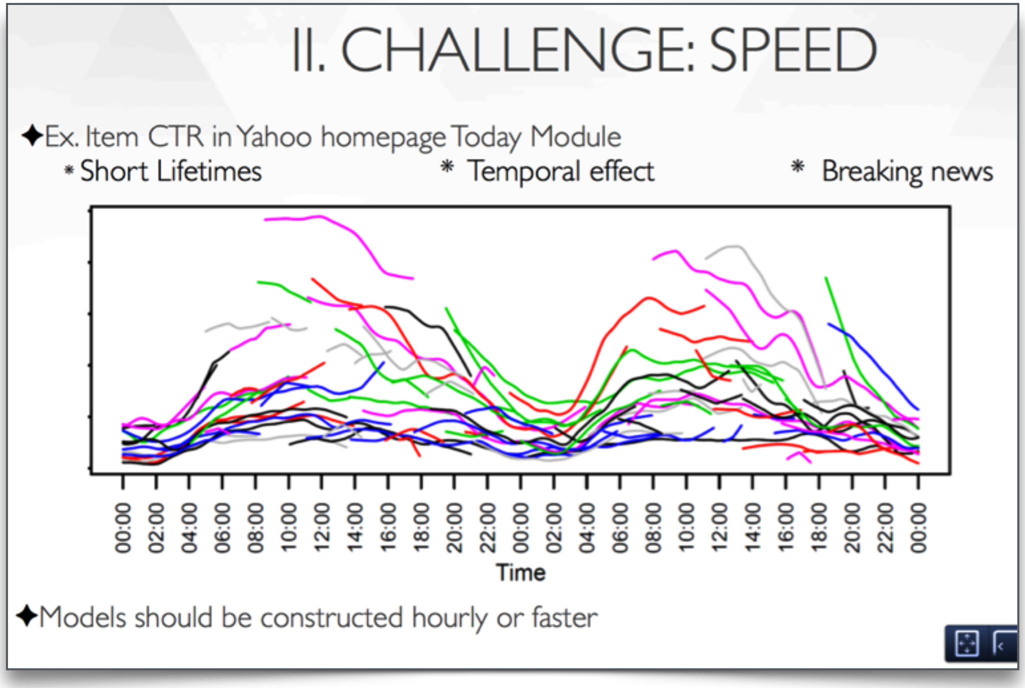


Summary: *Case Studies*

Hadoop and Spark Join Forces in Yahoo

Andy Feng

spark-summit.org/talk/feng-hadoop-and-spark-join-forces-at-yahoo/



Collaborative Filtering with Spark

Chris Johnson

slideshare.net/MrChrisJohnson/collaborative-filtering-with-spark

- collab filter (ALS) for music recommendation
- Hadoop suffers from I/O overhead
- show a progression of code rewrites, converting a Hadoop-based app into efficient use of Spark

Summary: *Case Studies*

Why Spark is the Next Top (Compute) Model
Dean Wampler
slideshare.net/deanwampler/spark-the-next-top-compute-model

- Hadoop: most algorithms are much harder to implement in this restrictive map-then-reduce model
- Spark: fine-grained "combinators" for composing algorithms
- slide #67, any questions?

Summary: *Case Studies*

Open Sourcing Our Spark Job Server

Evan Chan

engineering.ooyala.com/blog/open-sourcing-our-spark-job-server

- **github.com/ooyala/spark-jobserver**
- REST server for submitting, running, managing Spark jobs and contexts
- company vision for Spark is as a multi-team big data service
- shares Spark RDDs in one SparkContext among multiple jobs

Summary: *Case Studies*

Beyond Word Count:
Productionalizing Spark Streaming

Ryan Weald

spark-summit.org/talk/weald-beyond-word-count-productionalizing-spark-streaming/

blog.cloudera.com/blog/2014/03/letting-it-flow-with-spark-streaming/

- overcoming 3 major challenges encountered while developing production streaming jobs
- write streaming applications the same way you write batch jobs, reusing code
- stateful, exactly-once semantics out of the box
- integration of **Algebird**

Installing the Cassandra / Spark OSS Stack

Al Tobey

tobert.github.io/post/2014-07-15-installing-cassandra-spark-stack.html

- install+config for Cassandra and Spark together
- *spark-cassandra-connector* integration
- examples show a Spark shell that can access tables in Cassandra as RDDs with types pre-mapped and ready to go

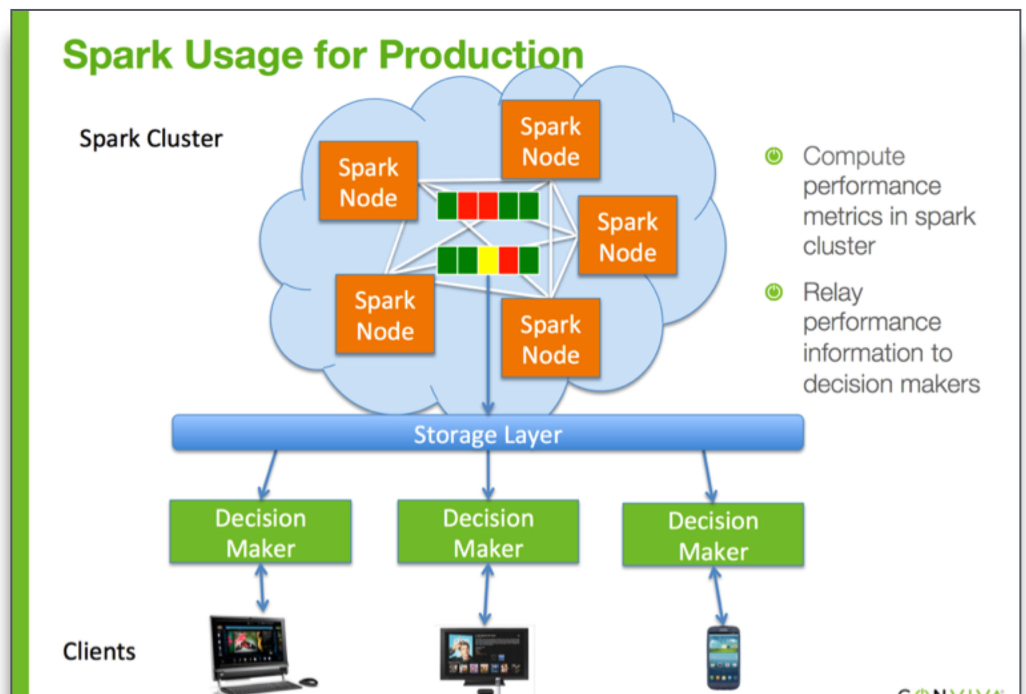
Summary: *Case Studies*

CONVIVA

One platform for all: real-time, near-real-time, and offline video analytics on Spark

Davis Shepherd, Xi Liu

spark-summit.org/talk/one-platform-for-all-real-time-near-real-time-and-offline-video-analytics-on-spark



Resources

certification:

Apache Spark developer certificate program

- http://oreilly.com/go/sparkcert
- defined by Spark experts @Databricks
- assessed by O'Reilly Media
- preview @Strata NY

community:

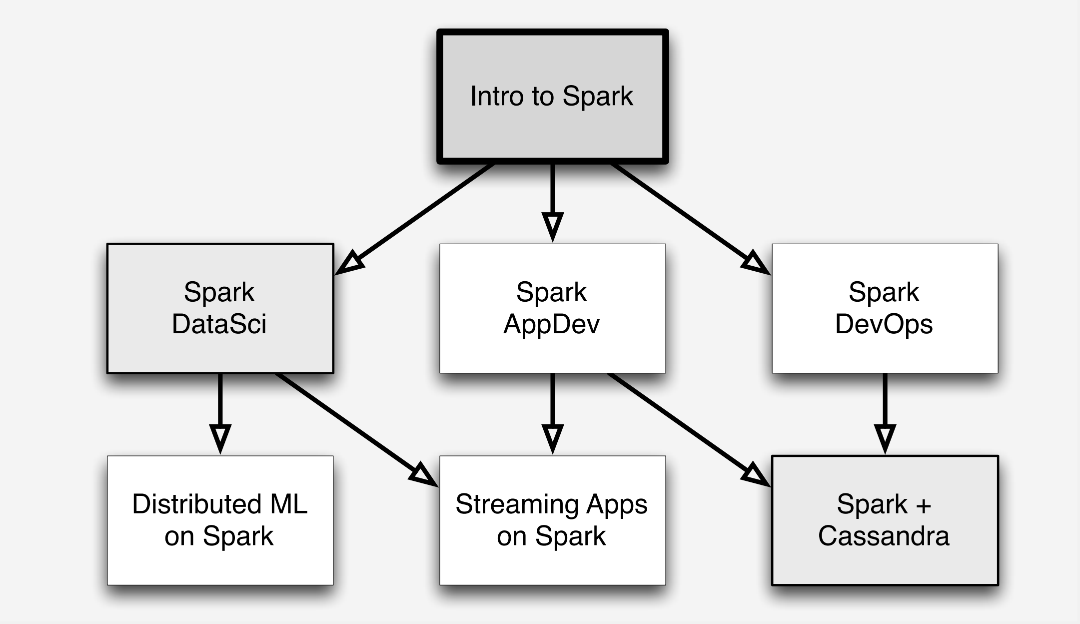
spark.apache.org/community.html

video+slide archives: spark-summit.org

local events: Spark Meetups Worldwide

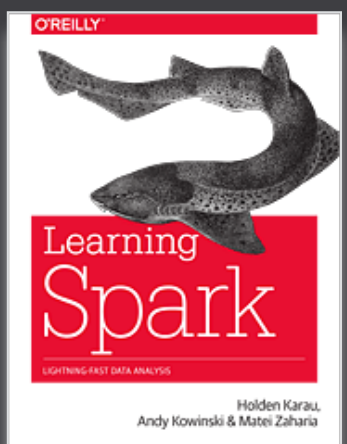
resources: databricks.com/spark-training-resources

workshops: databricks.com/spark-training



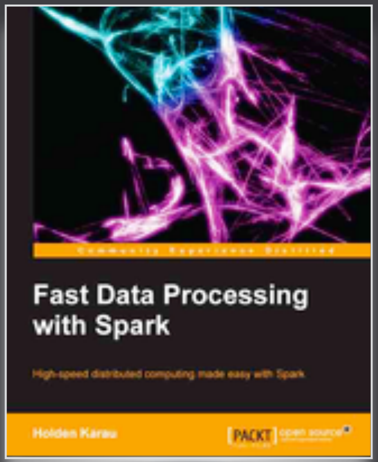
books:

Learning Spark
Holden Karau,
Andy Konwinski,
Matei Zaharia
O'Reilly (2015*)
shop.oreilly.com/product/0636920028512.do



Spark in Action
Chris Fregly
Manning (2015*)
sparkinaction.com/

Fast Data Processing with Spark
Holden Karau
Packt (2013)
shop.oreilly.com/product/9781782167068.do



events:

Strata NY + Hadoop World
NYC, Oct 15-17
strataconf.com/stratany2014

Big Data TechCon
SF, Oct 27
bigdatatechcon.com

Strata EU
Barcelona, Nov 19-21
strataconf.com/strataeu2014

Data Day Texas
Austin, Jan 10
datadaytexas.com

Strata CA
San Jose, Feb 18-20
strataconf.com/strata2015

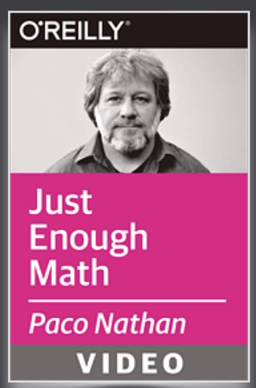
Spark Summit East
NYC, Mar 18-19
spark-summit.org/east

Spark Summit 2015
SF, Jun 15-17
spark-summit.org

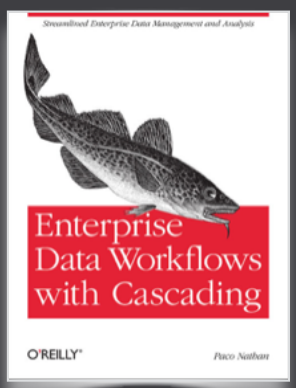
presenter:

monthly newsletter for updates, events, conf summaries, etc.:

liber118.com/pxn/



Just Enough Math
O'Reilly, 2014
justenoughmath.com
preview: youtu.be/TQ58cWgdCpA



Enterprise Data Workflows with Cascading
O'Reilly, 2013
shop.oreilly.com/product/0636920028536.do