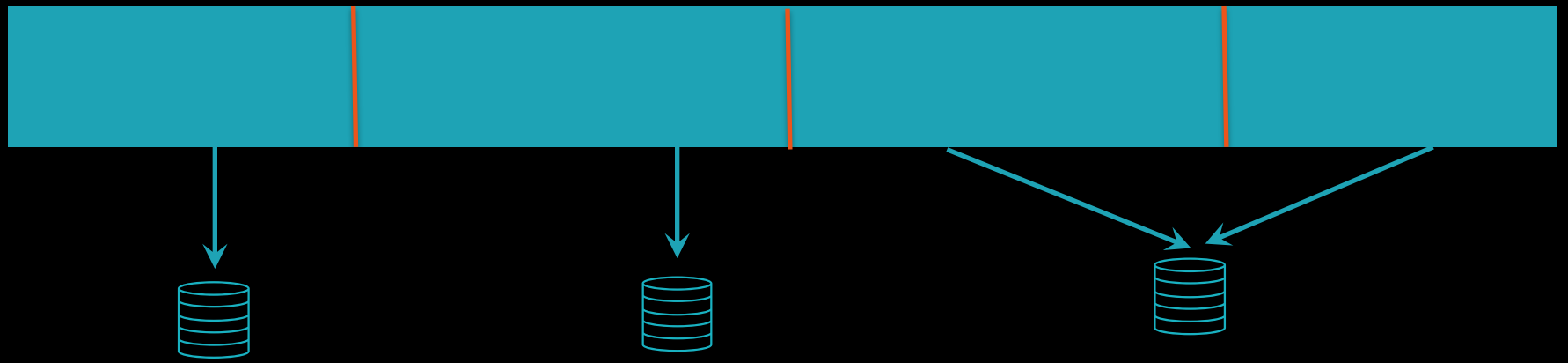
Resilient Distributed Dataset (RDD)

What are RDDs?

1. Distributed Data Abstraction

Logical Model Across Distributed Storage



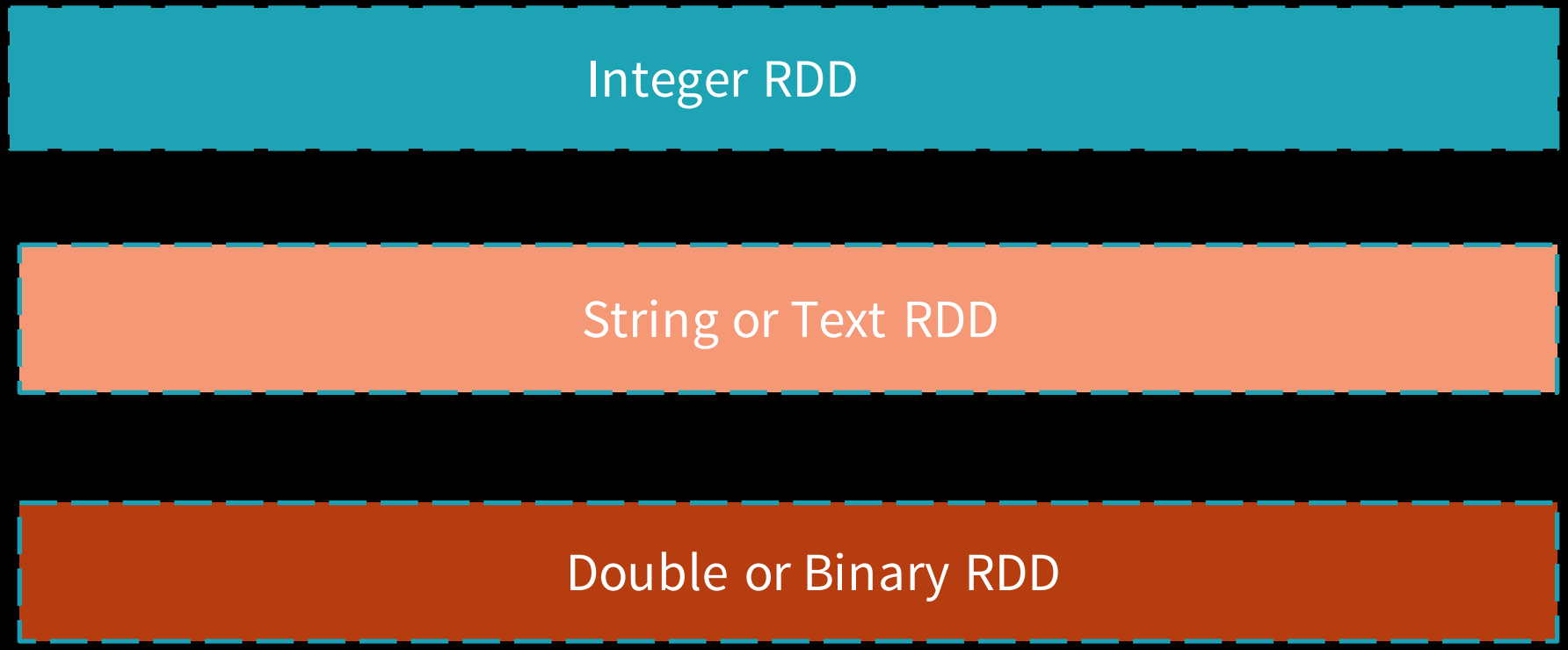
S3 or HDFS

2. Resilient & Immutable

RDD → T→ RDD -> RDD

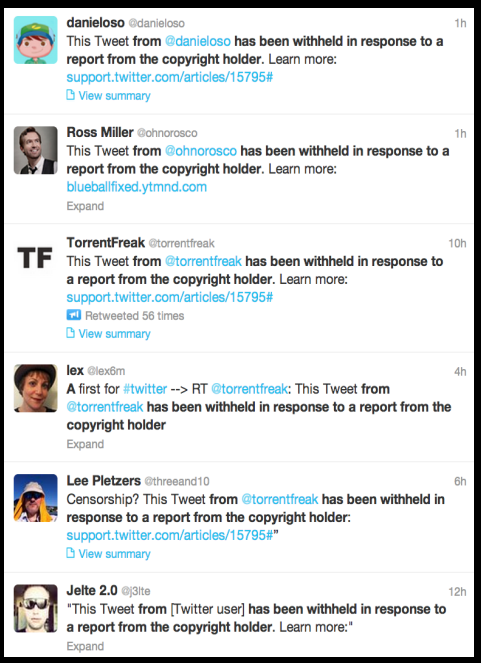
T = Transformation

3. Compile-time Type-safe



4. Unstructured/Structured Data: Text (logs, tweets, articles, social)

```
jkreps-mn:~ jkreps$ tail -f -n 20 /var/log/apache2/access_log
::1 - - [23/Mar/2014:15:07:00 -0700] "GET /images/apache_feather.gif HTTP/1.1" 200 4128
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/producer_consumer.png HTTP/1.1" 200 8
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_anatomy.png HTTP/1.1" 200 19579
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/consumer-groups.png HTTP/1.1" 200 268
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_compaction.png HTTP/1.1" 200 4141
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /documentation.html HTTP/1.1" 200 189893
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 200
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/kafka_log.png HTTP/1.1" 200 134321
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/mirror-maker.png HTTP/1.1" 200 17054
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /documentation.html HTTP/1.1" 200 189937
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /styles.css HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_logo.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/producer_consumer.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_anatomy.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/consumer-groups.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 304
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_compaction.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_log.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/mirror-maker.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:09:55 -0700] "GET /documentation.html HTTP/1.1" 200 195264
```



databricks

Structured Tabular data..

```
> %sql select ipaddress, datetime, method, endpoint, protocol, responsecode, agent from accesslog limit 10;
```

▸ (1) Spark Jobs

ipaddress	datetime	method	endpoint	protocol	responsecode	agent
10.223.144.123	04/Nov/2015:08:15:00 +0000	GET	/company/contact	HTTP/1.1	200	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_ AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36
10.223.144.123	04/Nov/2015:08:15:37 +0000	GET	/blog	HTTP/1.1	200	Mozilla/5.0 (Windows NT 6.3; WOW64; rv:35.0) Gecko/20100101 Firefox/35.0
10.223.96.51	04/Nov/2015:08:16:38 +0000	GET	/pantheon_healthcheck	HTTP/1.1	200	Pingdom.com_bot_version_1.4_(http://www.pi
10.223.144.123	04/Nov/2015:08:18:15 +0000	GET	/blog/2014/04/14/spark-with-java-8.html	HTTP/1.1	200	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.80 Safari/537.36

databricks

5. Lazy

RDD → T→ RDD → T→ RDD

T = Transformation

A = Action

Apache Spark Operations =

TRANSFORMATIONS

+

ACTIONS

 = easy  = medium

Essential Core & Intermediate Spark Operations

TRANSFORMATIONS

General

- map
- filter
- flatMap
- mapPartitions
- mapPartitionsWithIndex
- groupBy
- sortBy

Math / Statistical

- sample
- randomSplit

Set Theory / Relational

- union
- intersection
- subtract
- distinct
- cartesian
- zip

Data Structure / I/O

- keyBy
- zipWithIndex
- zipWithUniqueID
- zipPartitions
- coalesce
- repartition
- repartitionAndSortWithinPartitions
- pipe

ACTIONS

- reduce
- collect
- aggregate
- fold
- first
- take
- forEach
- top
- treeAggregate
- treeReduce
- forEachPartition
- collectAsMap

- count
- takeSample
- max
- min
- sum
- histogram
- mean
- variance
- stdev
- sampleVariance
- countApprox
- countApproxDistinct

- takeOrdered

- saveAsTextFile
- saveAsSequenceFile
- saveAsObjectFile
- saveAsHadoopDataset
- saveAsHadoopFile
- saveAsNewAPIHadoopDataset
- saveAsNewAPIHadoopFile

Why Use RDDs?

- … Offer Control & flexibility
- ... Low-level API
- ... Type-safe
- ... Encourage ***how-to***

Some code to read Wikipedia

```
val rdd = sc.textFile("/mnt/wikipediapagecounts.gz")

val parsedRDD = rdd.flatMap {

    line => line.split("""\s+""") match {

        case Array(project, page, numRequests, _) => Some((project, page, numRequests))

        case _ => None

    }

}

// filter only English pages ; count pages and requests to it.

parsedRDD.filter { case (project, page, numRequests) => project == "en" }.

        map { case (_, page, numRequests) => (page, numRequests) }.

        reduceByKey(_ + _).

        take(100). foreach { case (page, requests) => println(s"$page: $requests") }
```

When to Use RDDs?

- ... Low-level API & control of dataset
- ... Dealing with unstrucrured data (media streams or texts)
- ... Manipulate data with lambda functions than DSL
- ... Don’t care schema or structure of data
- ... Sacrifice optimization, performance & inefficiecies

What's the Problem?

What's the problem?

- ... Express ***how-to*** solution, not ***what-to***
- ... Not optimized by Spark
- ... Slow for non-JVM languages like Python
- ... ***Inadverdent inefficiecies*** ←

Inadvertent inefficiencies in RDDs

```
parsedRDD.filter { case (project, page, numRequests)  => project == "en" }.
        map { case (_, page, numRequests)  => (page, numRequests)  }.
        reduceByKey(_ + _).
        filter { case (page, _) => ! isSpecialPage(page) }.
        take(100). foreach { case (project, requests)  => println (s"project: $requests") }
```

Structured in Spark
DataFrames & Datasets APIs

databricks

Background: What is in an RDD?

- Dependencies
- Partitions (with optional locality info)
- Compute function: Partition => Iterator[T]

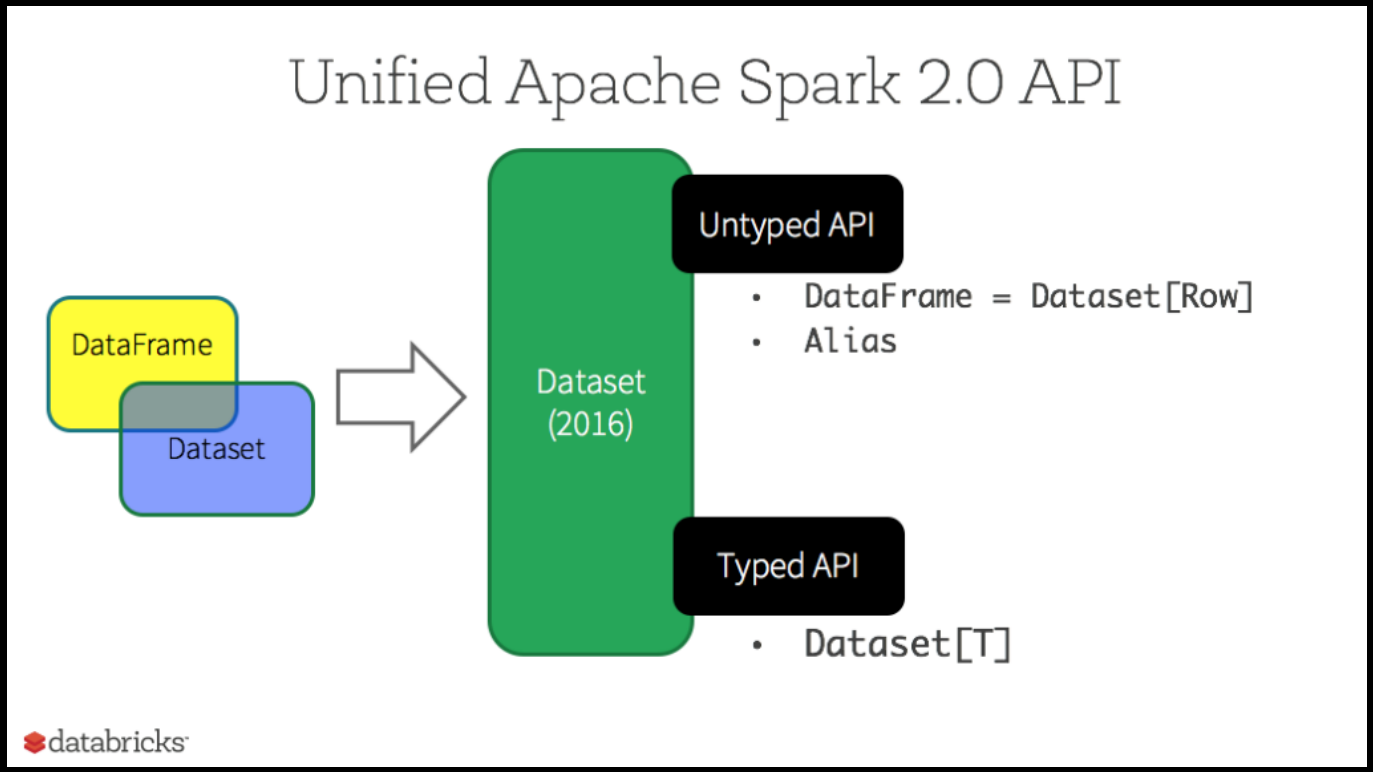
Opaque Computation
& Opaque Data

Structured APIs In Spark

	SQL	DataFrames	Datasets
Syntax Errors	Runtime	Compile Time	Compile Time
Analysis Errors	Runtime	Runtime	Compile Time

Analysis errors are reported before a distributed job starts

Unification of APIs in Spark 2.0



DataFrame API code.

```
// convert RDD -> DF with column names
val df = parsedRDD.toDF("project", "page", "numRequests")
//filter, groupBy, sum, and then agg()
df.filter($"project" === "en").
    groupBy($"page").
    agg(sum($"numRequests").as("count")).
    limit(100).
    show(100)
```

project	page	numRequests
en	23	45
en	24	200

Take DataFrame → SQL Table → Query

```
df. createOrReplaceTempView ("edits")

val results = spark.sql("""SELECT page, sum(numRequests)
AS count FROM edits WHERE project = 'en' GROUP BY page
LIMIT 100""")

results.show(100)
```

project	page	numRequests
en	23	45
en	24	200

Easy to write code... Believe it!

```
from pyspark.sql.functions import avg

dataRDD = sc.parallelize([("Jim", 20), ("Anne", 31), ("Jim", 30)])
dataDF = dataRDD.toDF(["name", "age"])

# Using RDD code to compute aggregate average
(dataRDD.map(lambda (x,y): (x, (y,1))) .reduceByKey(lambda x,y: (x[0] +y[0], x[1]
+y[1])) .map(lambda (x, (y, z)): (x, y / z)))

# Using DataFrame
dataDF.groupBy("name").agg(avg("age"))
```

name	age
Jim	20
Ann	31
Jim	30

Why structure APIs?

DataFrame

```
data.groupBy("dept").avg("age")
```

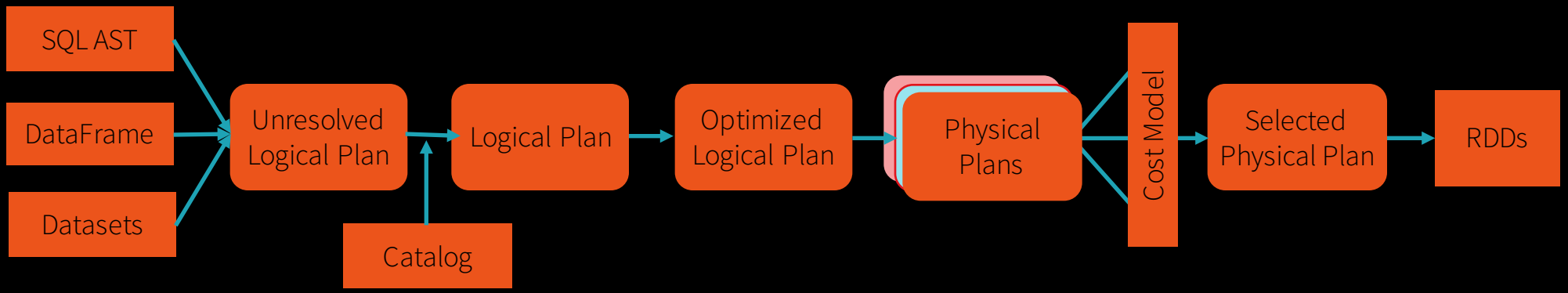
SQL

```
select dept, avg(age) from data group by 1
```

RDD

```
data.map { case (dept, age) => dept -> (age, 1) }
    .reduceByKey { case ((a1, c1), (a2, c2)) => (a1 + a2, c1 + c2)}
    .map { case (dept, (age, c)) => dept -> age / c }
```

Using Catalyst in Spark SQL



Analysis: analyzing a logical plan to resolve references

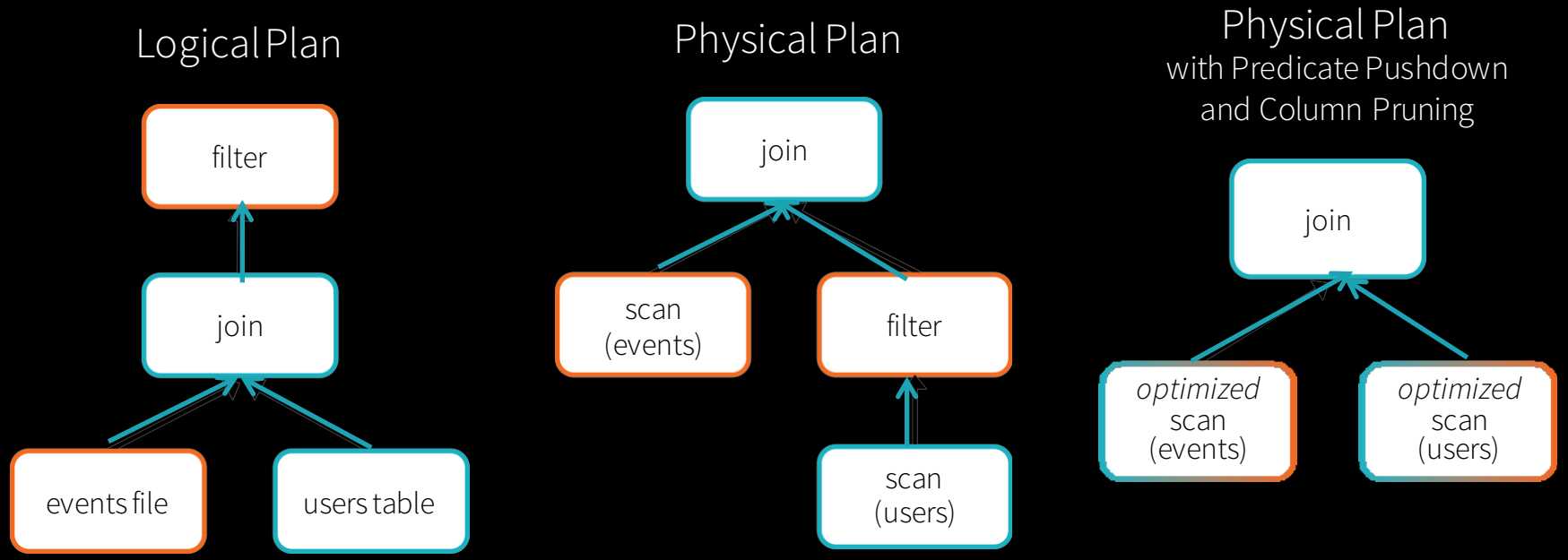
Logical Optimization: logical plan optimization

Physical Planning: Physical planning

Code Generation: Compile parts of the query to Java bytecode

DataFrame Optimization

```
users.join(events, users("id") === events("uid")) .
    filter(events("date") > "2015-01-01")
```

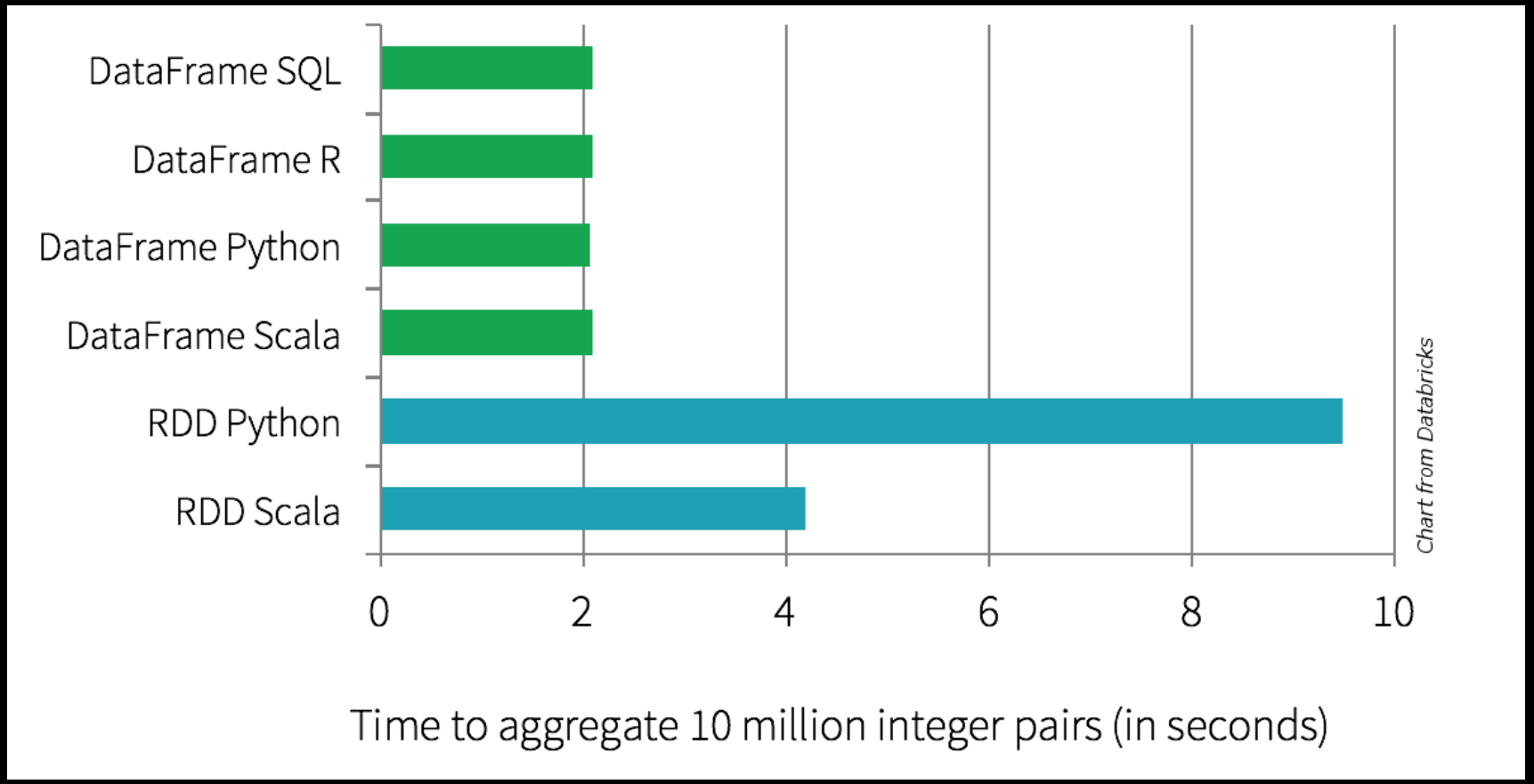


Dataset API in Spark 2.x

Type-safe: operate on domain objects with compiled lambda functions

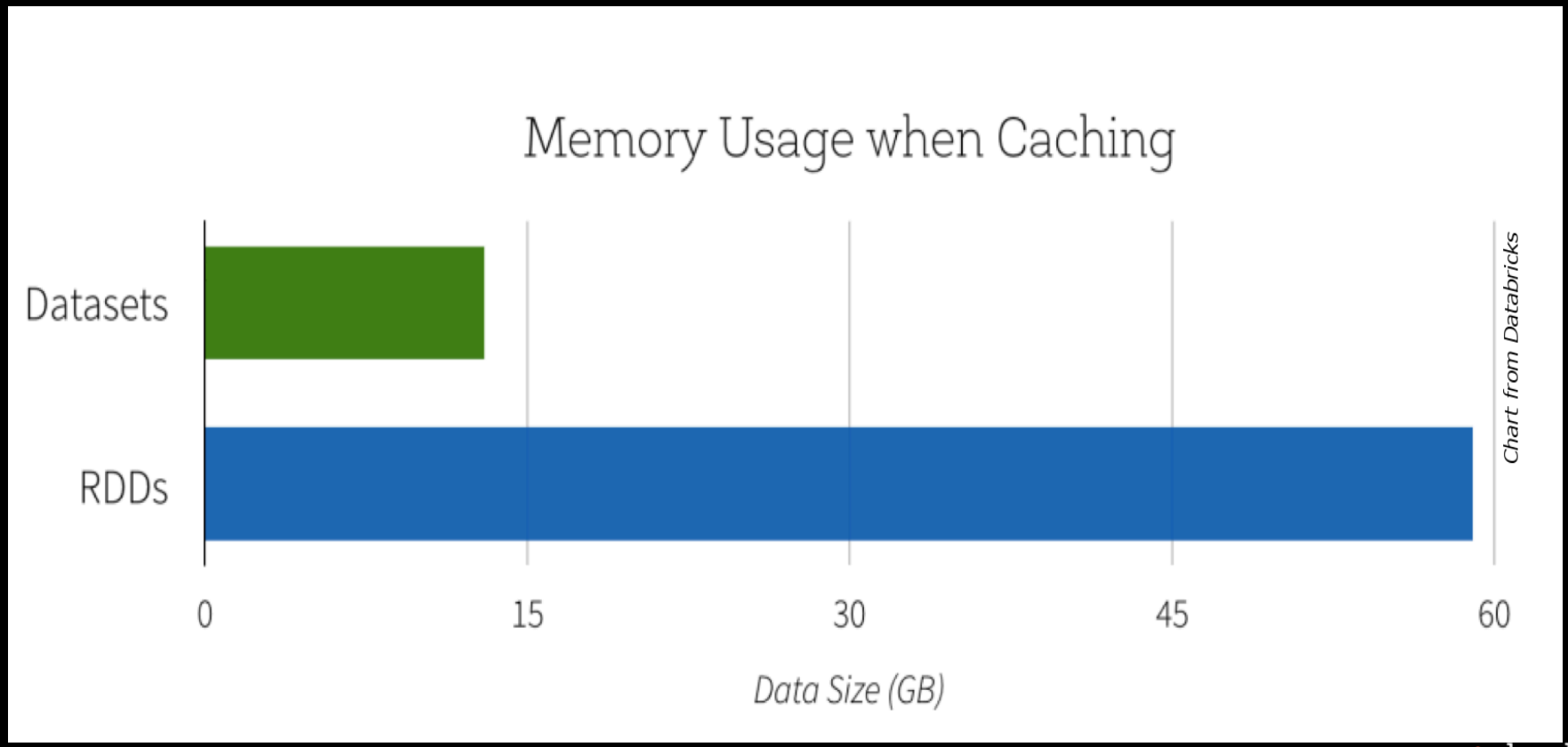
```
val df = spark.read.json("people.json")
// Convert data to domain objects.

case class Person(name: String, age: Int)
val ds: Dataset[Person] = df.as[Person]

val = filterDS = ds.filter(p=>p.age > 3)
```

DataFrames are Faster than RDDs



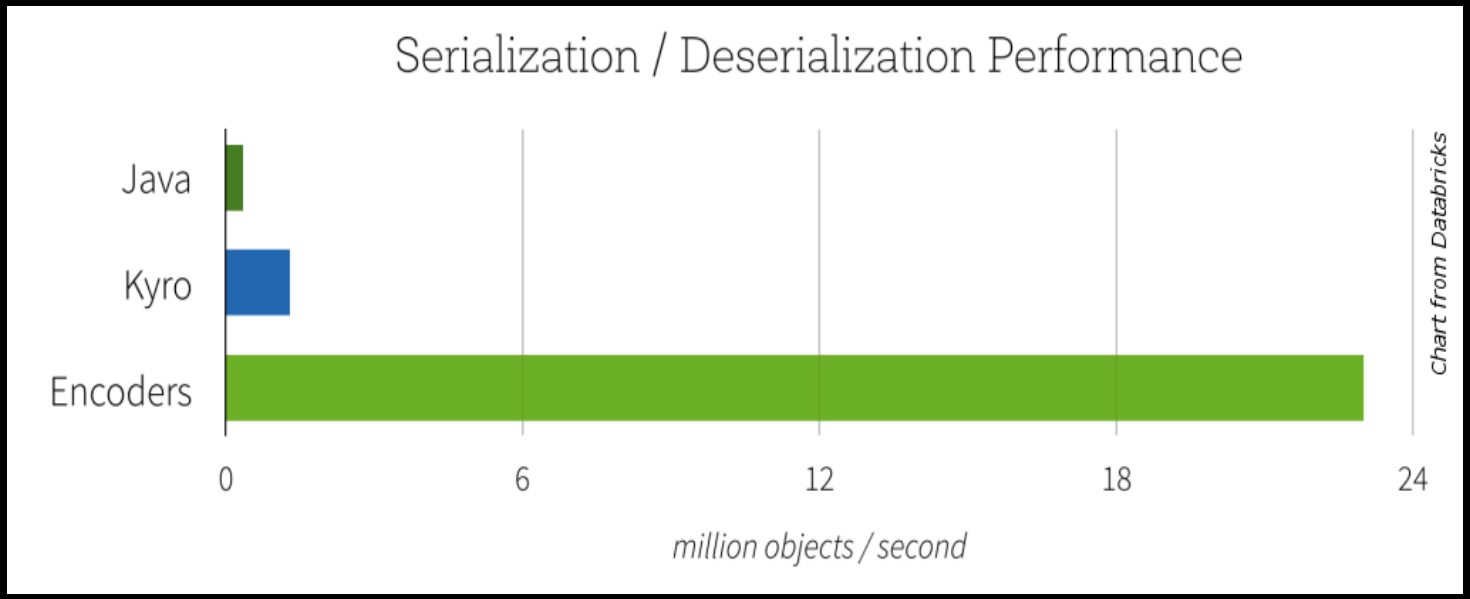
databricks

Datasets < Memory RDDs



databricks

Datasets Faster…



DataFrames & Datasets

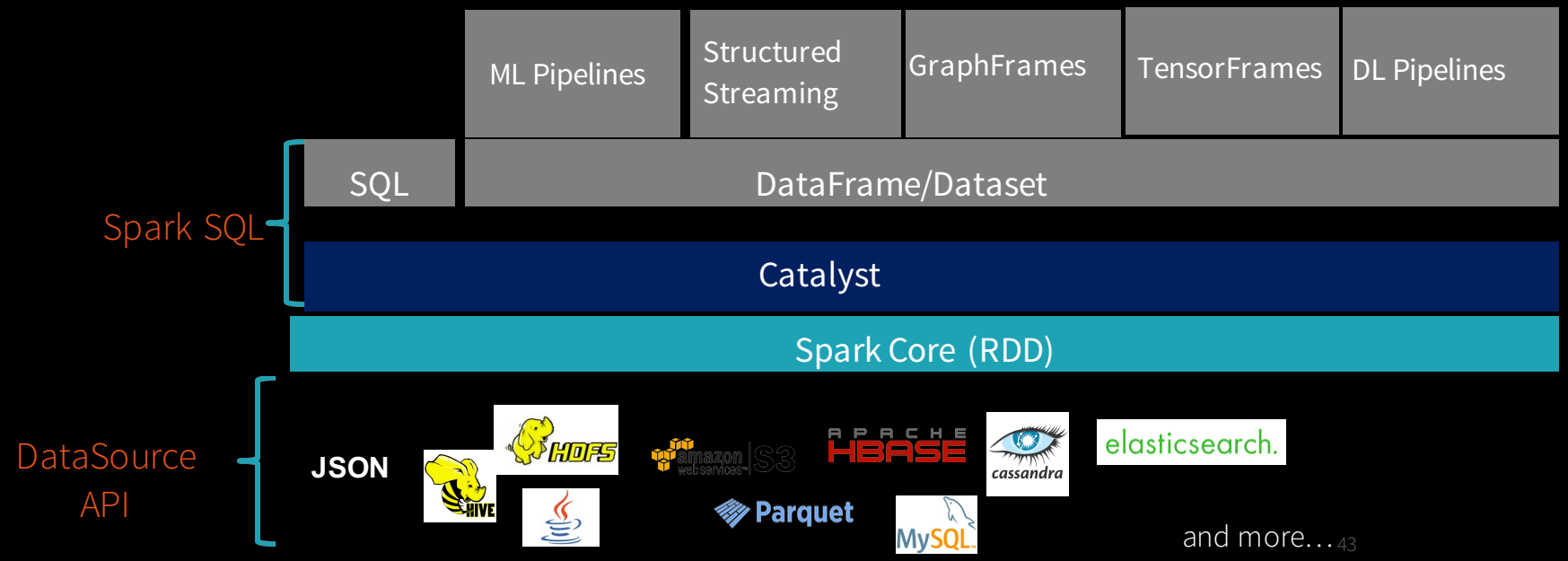
Why

- High-level APIs and DSL
- Strong Type-safety
- Ease-of-use & Readability
- ***What-to-do***

When

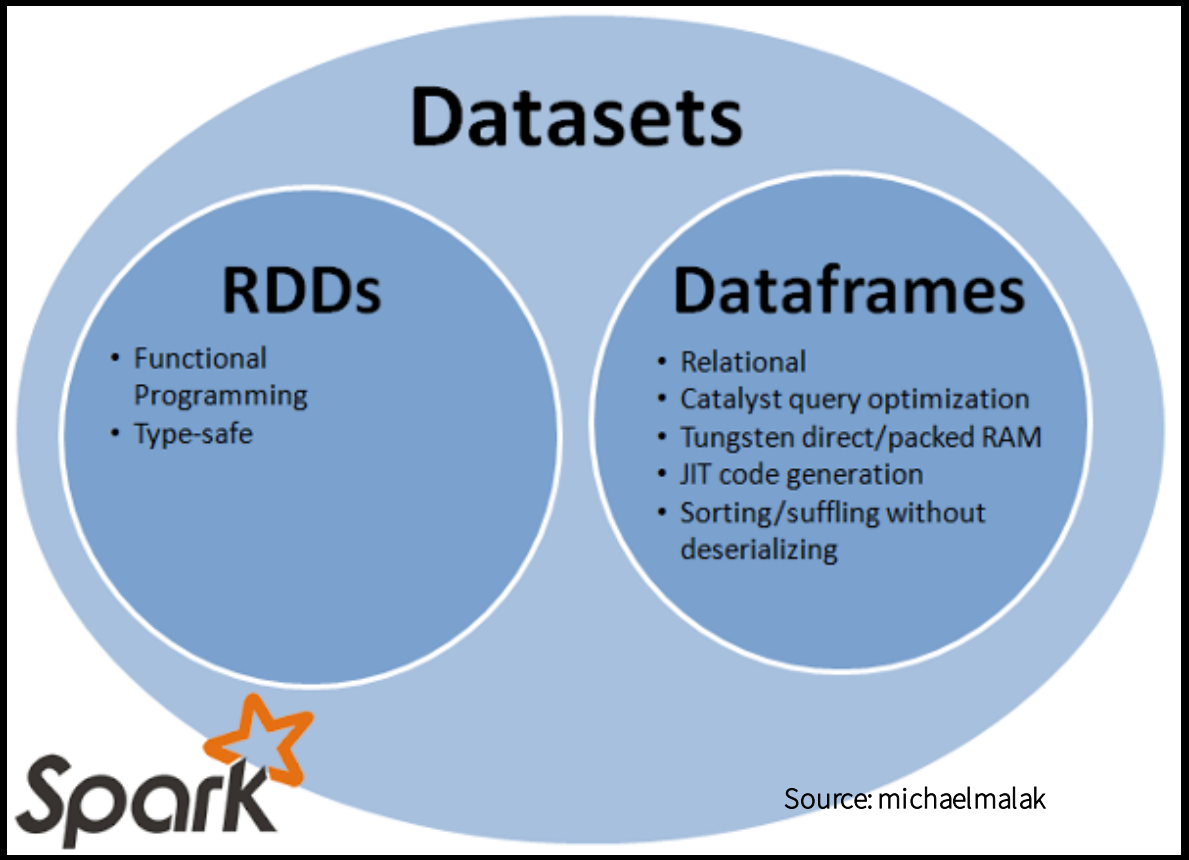
- Structured Data schema
- Code optimization & performance
- Space efficiency with Tungsten

Foundational Spark 2.x Components



databricks

Putting all Together: Conclusion



Resources

- Getting Started Guide with Apache Spark on Databricks
- docs.databricks.com
- Spark Programming Guide
- https://databricks.com/blog/2016/01/04/introducing-apache-spark-datasets.html
- https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html
- https://github.com/bmc/rdds-dataframes-datasets-presentation-2016
- Databricks Engineering Blogs