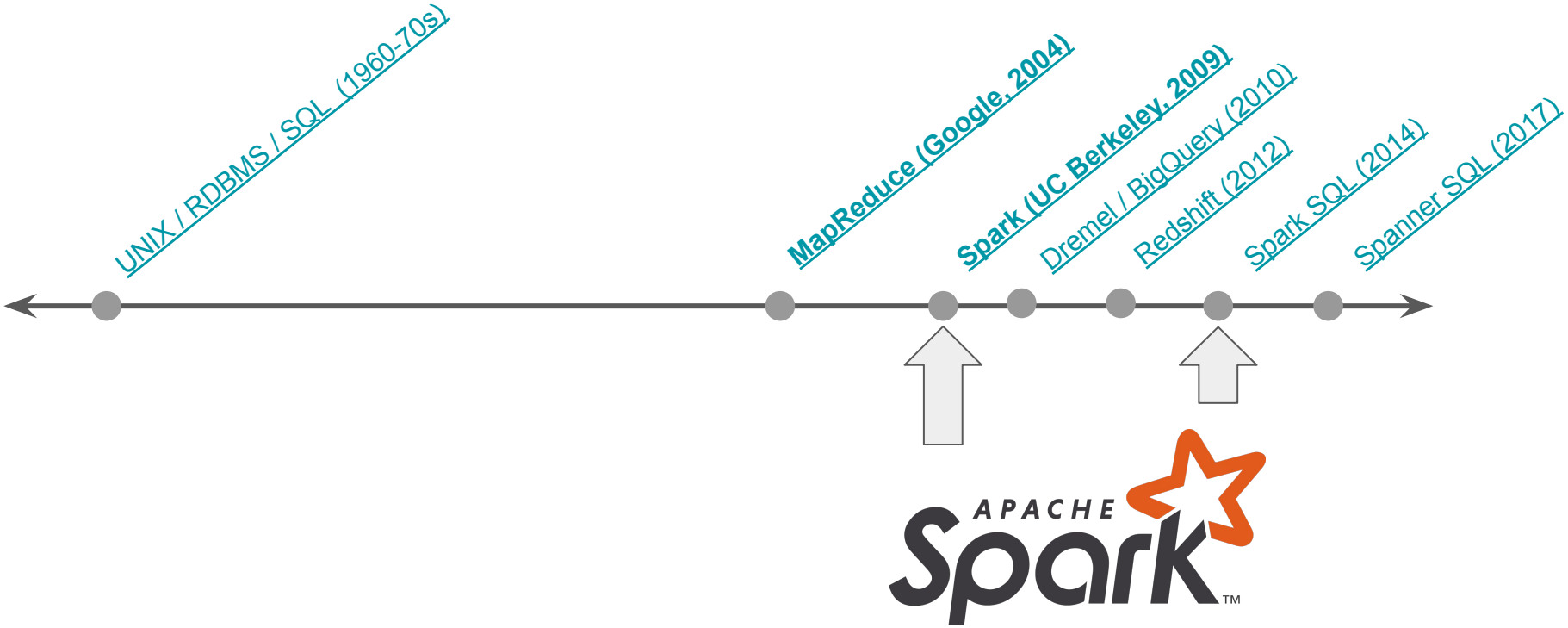
CSC 369

Introduction to Distributed Computing

[Distributed] Data Processing Timeline



Timeline

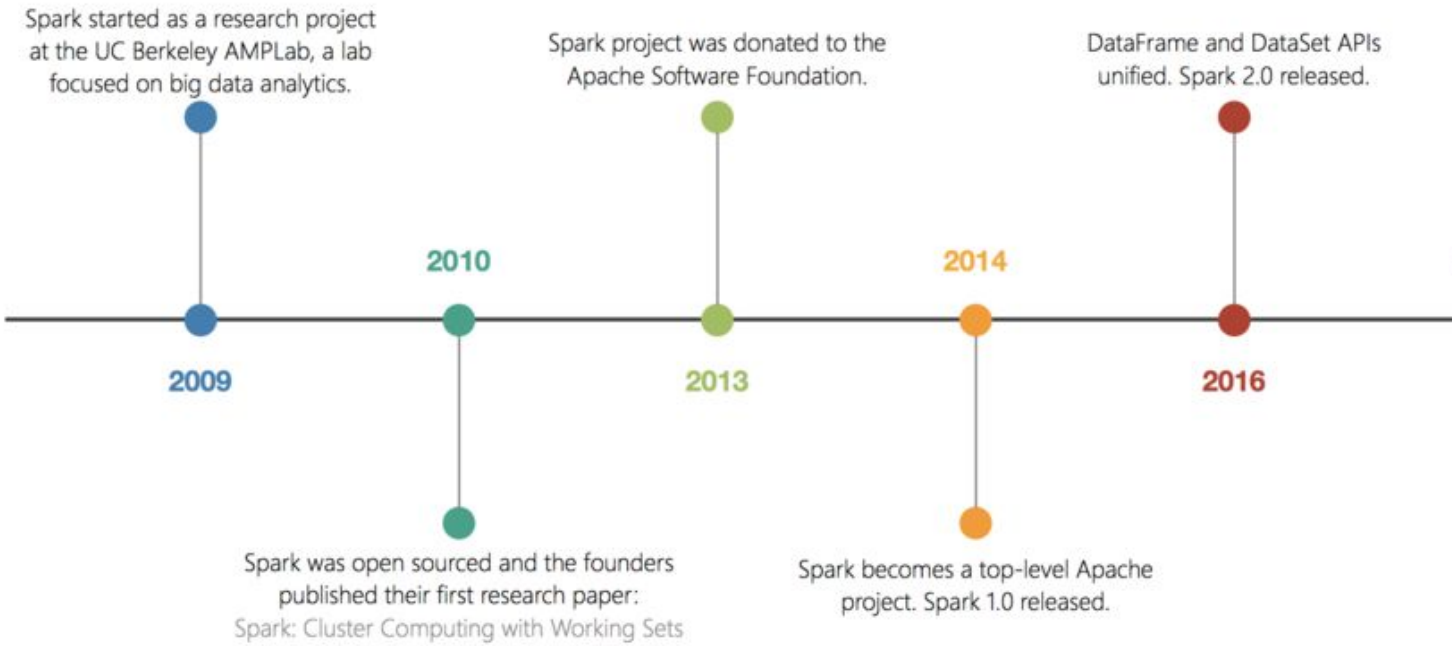


Image Source

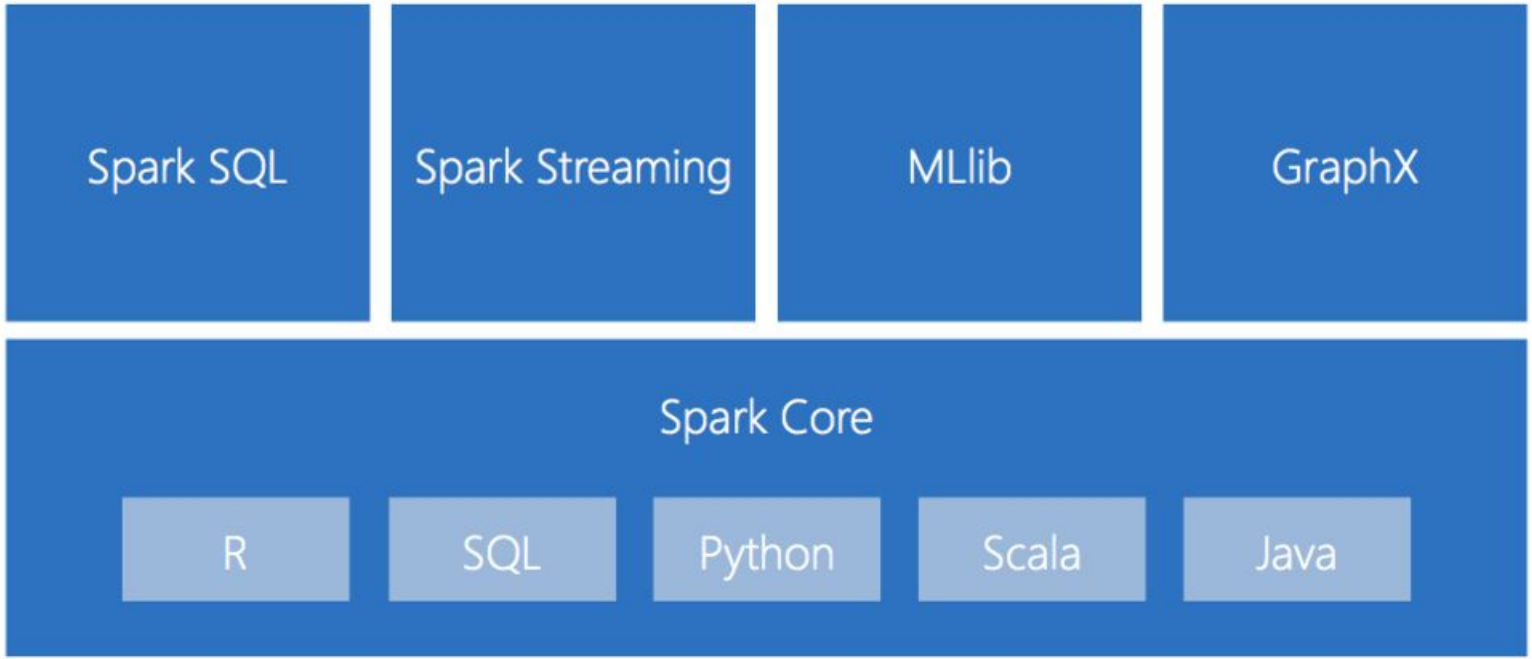


Image Source

Spark / RDD Example: WordCount

```
text_file = sc.textFile("sample.txt")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.collect()
```

Input and Output

INPUT

Apache Spark is a fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.

sample.txt

OUTPUT

```
[(u'and', 5), (u'is', 1), (u'graphs.', 1), (u'including', 1),
(u'tools', 1), (u'processing,', 2), (u'for', 3), (u'Python', 1),
(u'Scala,', 1), (u'fast', 1), (u'machine', 1), (u'rich', 1),
(u'Apache', 1), (u'Spark', 3), (u'supports', 2), (u'engine', 1),
(u'provides', 1), (u'higher-level', 1), (u'Streaming.', 1),
(u'general-purpose', 1), (u'optimized', 1), (u'R,', 1), (u'data', 1),
(u'a', 2), (u'system.', 1), (u'GraphX', 1), (u'also', 1),
(u'execution', 1), (u'set', 1), (u'general', 1), (u'computing', 1),
(u'SQL', 2), (u'graph', 1), (u'APIs', 1), (u'MLlib', 1), (u'an', 1),
(u'structured', 1), (u'cluster', 1), (u'that', 1), (u'of', 1), (u'in',
1), (u'It', 2), (u'high-level', 1), (u'Java,', 1), (u'learning,', 1)]
```

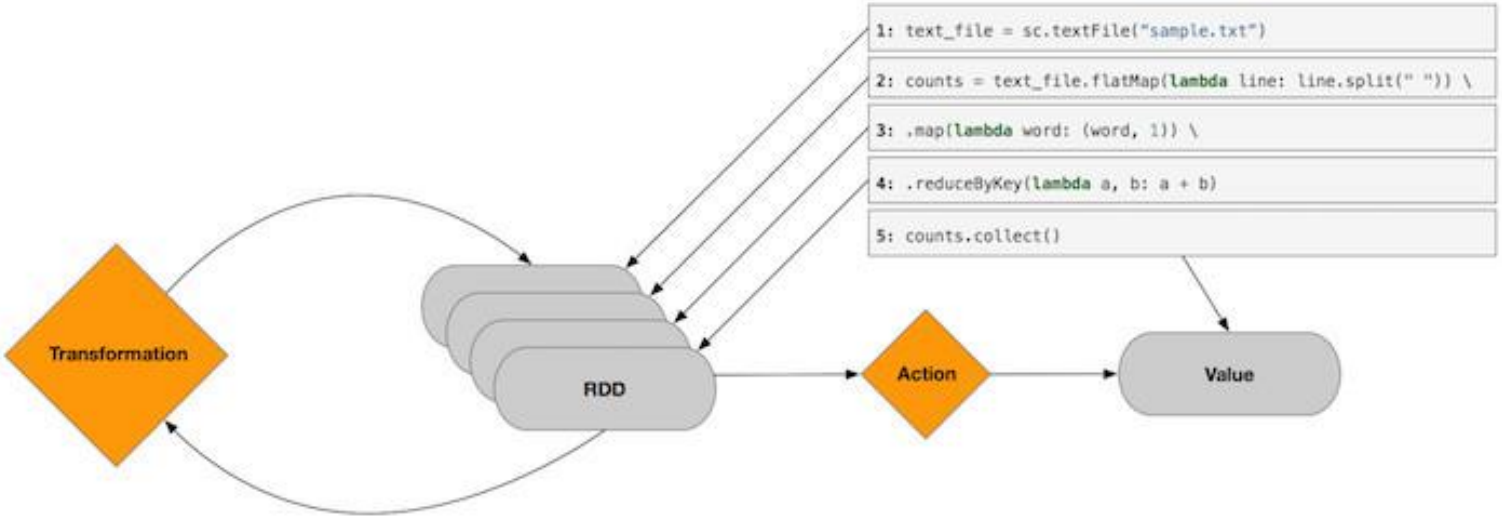


Image Source

RDD Transformations and Actions

Filters (a subset of data passes through)

Transformations	Actions
`filter()`	`collect()`
`distinct()`	`first()`
`sample()`	`take()`
`sampleByKey()`	`top()`
	`takeOrdered()`
	`takeSample()`
	`lookup()`

RDD Transformations and Actions

Projections: transformations of data

Transformations	Actions
`map()`	`collectAsMap()`
`flatMap()`	
`mapValues()`	
`flatMapValues()`	
`forEach()`	
`keys()`	
`values()`	
`keyBy()`	
`zipWithIndex()`	
`zipWithUniqueId()`	

RDD Transformations and Actions

Grouping: combining the data

Aggregation: computations

Transformations

```
groupBy()
groupByKey()
histogram()
```

Actions

```
reduce()
count()
max()
min()
mean()
sum()
stdev()
variance()
stats()
```

```
sampleStdDev()
sampleVariance()
countByKey()
countByValue()
sampleStdDev()
sampleVariance()
countApprox()
meanApprox()
sumApprox()
```

RDD Transformations and Actions

Joins and Pairings: combining RDDs

Transformations

```
join()
fullOuterJoin()
leftOuterJoin()
rightOuterJoin()
cartesian()
```

```
coGroup()
zip()
```

```
union()
intersection()
subtract()
subtractByKey()
```

RDD Transformations and Actions

Partitioning: creating/managing splits in data

Transformations

```
partitionBy()
repartition()
randomSplit()
coalesce()
glom()
```

RDD Transformations and Actions

Other (sorting)

Transformations

`sortBy()`
`sortByKey()`

RDD Transformations and Actions

Saving Results

Actions

```
saveAsNewAPIHadoopDataset()
saveAsNewAPIHadoopFile()
saveAsPickleFile()
saveAsTextFile()
```

[Google Colab Notebook - PySpark Introduction]

Find Top Visitors: UNIX / SQL

```
awk '{print $1}' access.log | # Split by whitespace, extract client address (field #1)
  sort     | # Occurrences of the same client address will appear consecutively
  uniq -c  | # Collapse repeated client address into a count
  sort -nr | # Sort (numeric), in reverse (descending) order
  head -n 10 # Show top 10 client addresses (preceded by count of log lines)
```

```
SELECT COUNT(*) AS c, client_address
FROM access_log
GROUP BY client_address
ORDER BY c DESC
LIMIT 10
```