Apache Spark

Credit to Peerapon Vateekul

https://github.com/kaopanboonyuen/GISTDA2023

Content

- Introduction to Apache Spark
- How does Spark work?
- Spark MLlib & ML

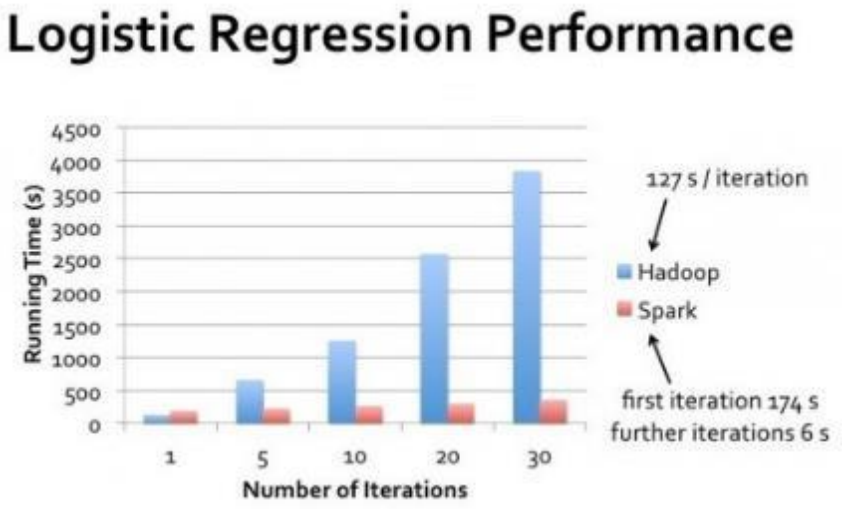
Introduction to Apache Spark

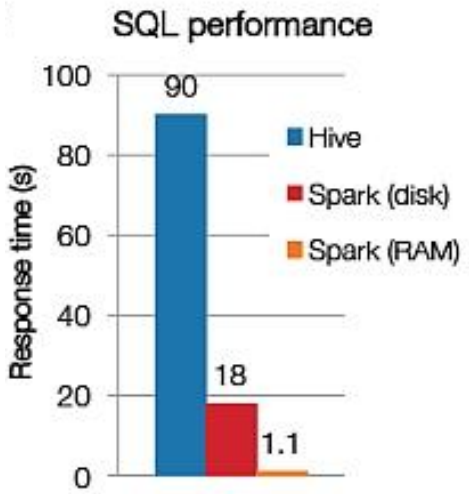
What is Apache Spark?

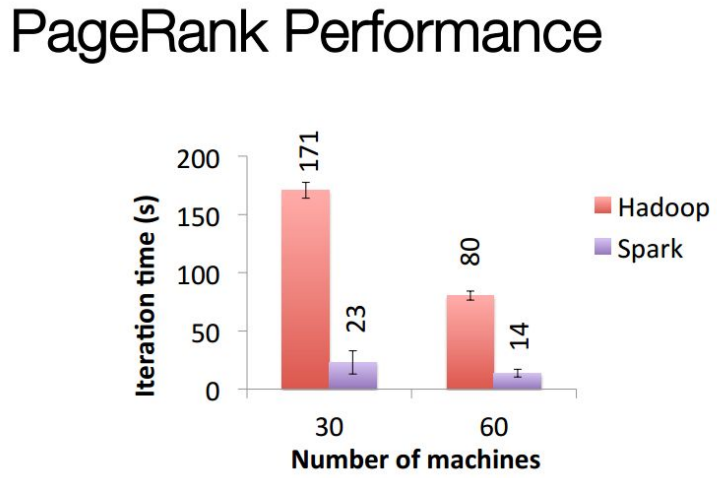
- Apache Spark is a fast and general-purpose cluster computing system
- In-memory processing on distributed dataset on distributed memory/disk
- Automatically rebuilt on failure
- Provides high-level APIs in Java, Scala, Python and R
- Rich set of higher-level tools
  - SparkSQL
  - MLlib
  - ML
  - Graphx
  - SparkStreaming
  - More

Speed

- Run up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk
- Apache Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing



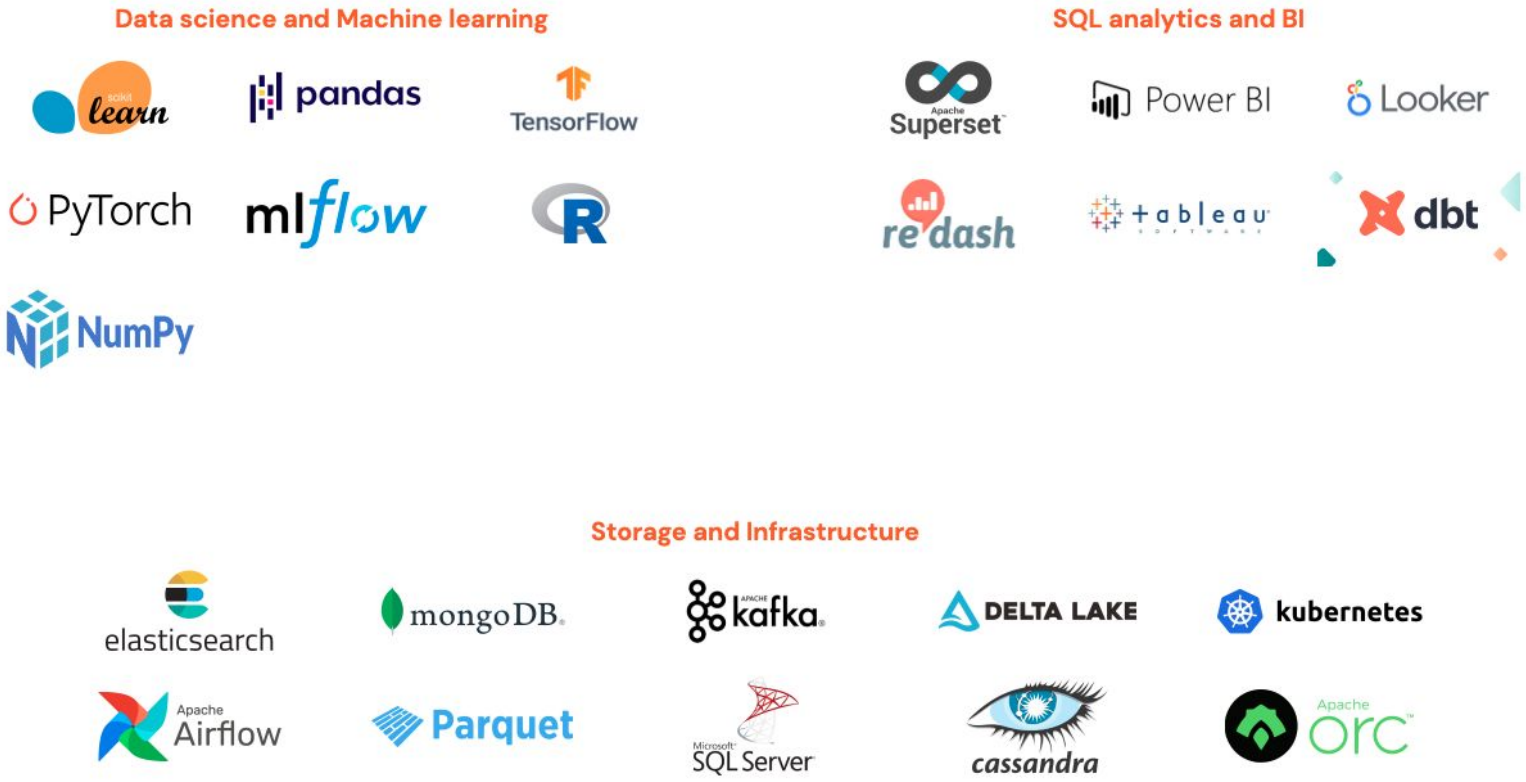




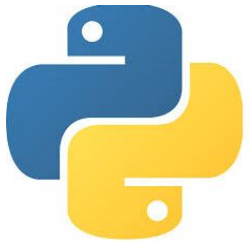
Reference: https://spark.apache.org/

Ecosystem

Apache Spark™ integrates with your favorite frameworks, helping to scale them to thousands of machines.



Ease of use




- Write applications quickly in Java, Scala, Python and R
- Spark offers over 80 high-level operators that make it easy to build parallel apps
- can use it *interactively* from the Scala, Python and R shells

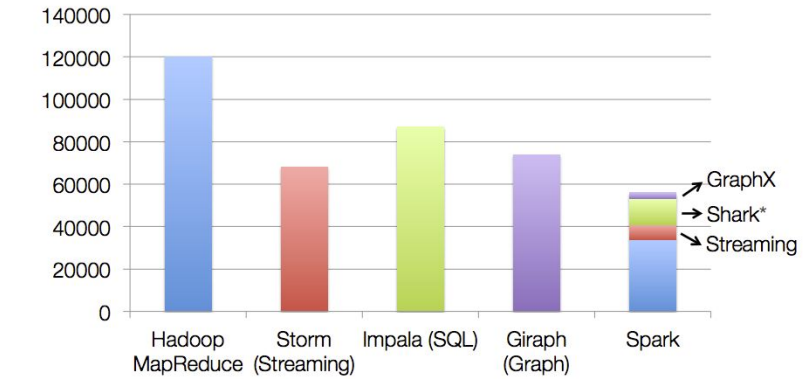
Python

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

```
val textFile = sc.textFile("hdfs://...")
val counts = textFile.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

Scala

Code Size

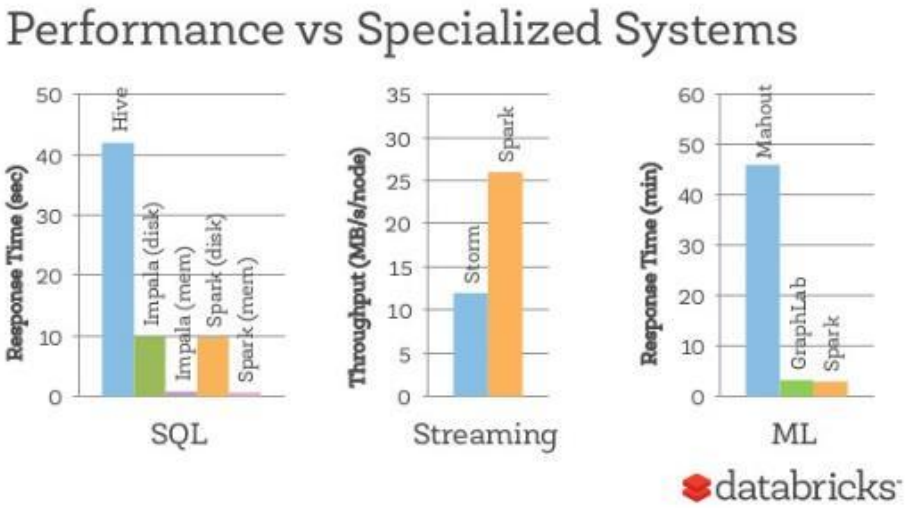


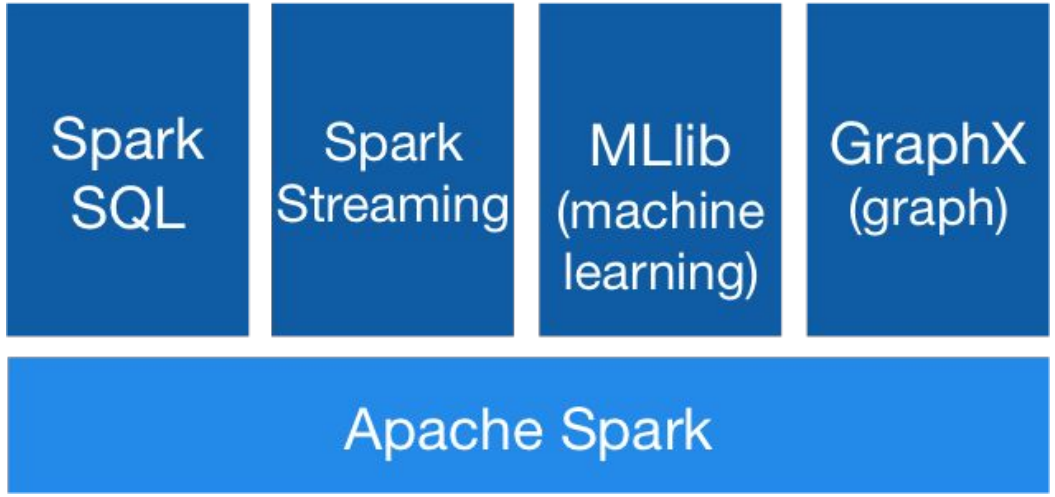
non-test, non-example source lines

* also calls into Hive

Generality

- Combine SQL, Streaming and complex analytics
- Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming
- You can combine these libraries seamlessly in the same application

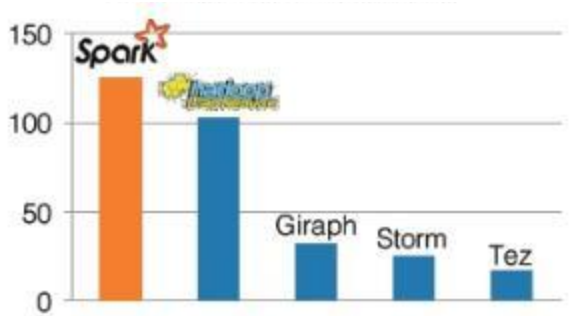




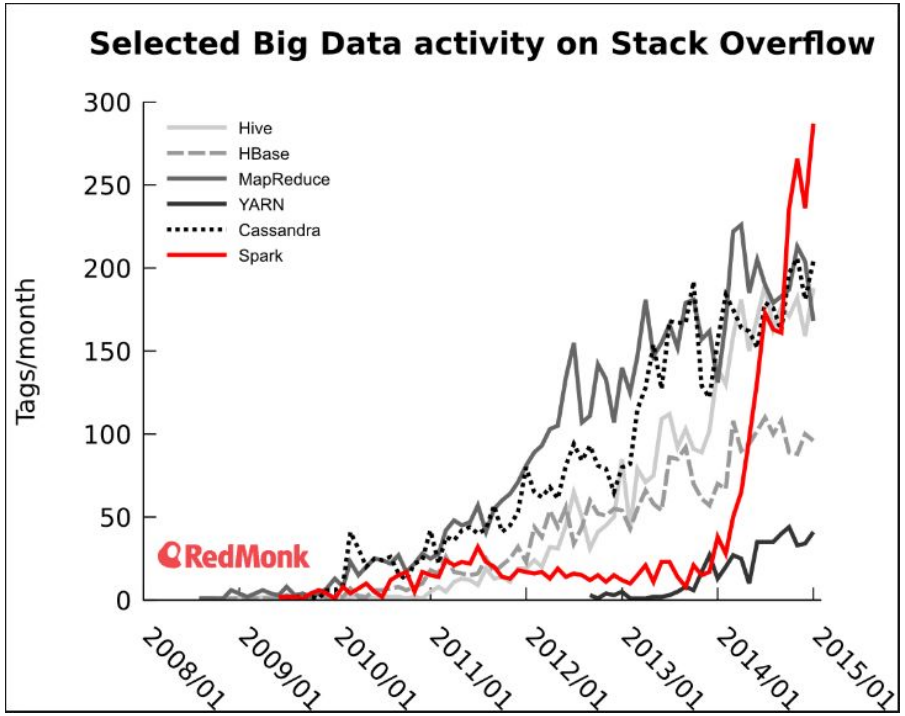
Run everywhere

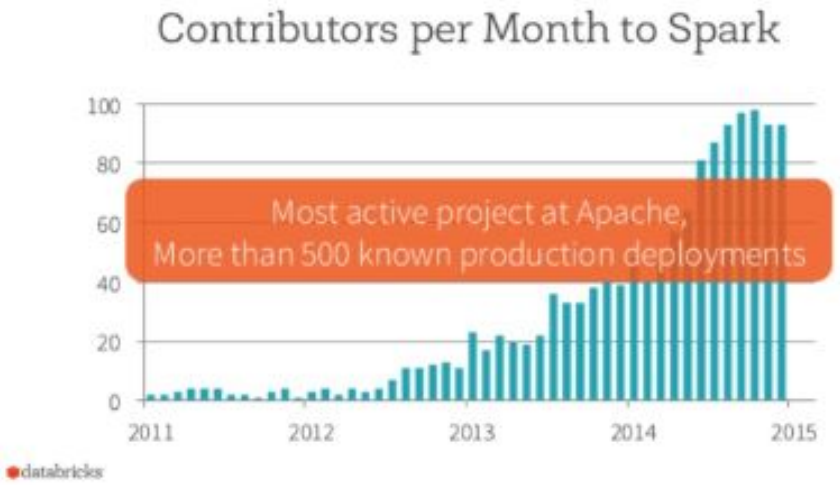
- Spark can run on
  - Hadoop(Yarn)
  - Mesos
  - Spark Standalone
  - In the cloud
- Spark can access data source from
  - HDFS
  - Cassandra
  - Hbase
  - S3
  - Hive
  - Any Hadoop data source

Community

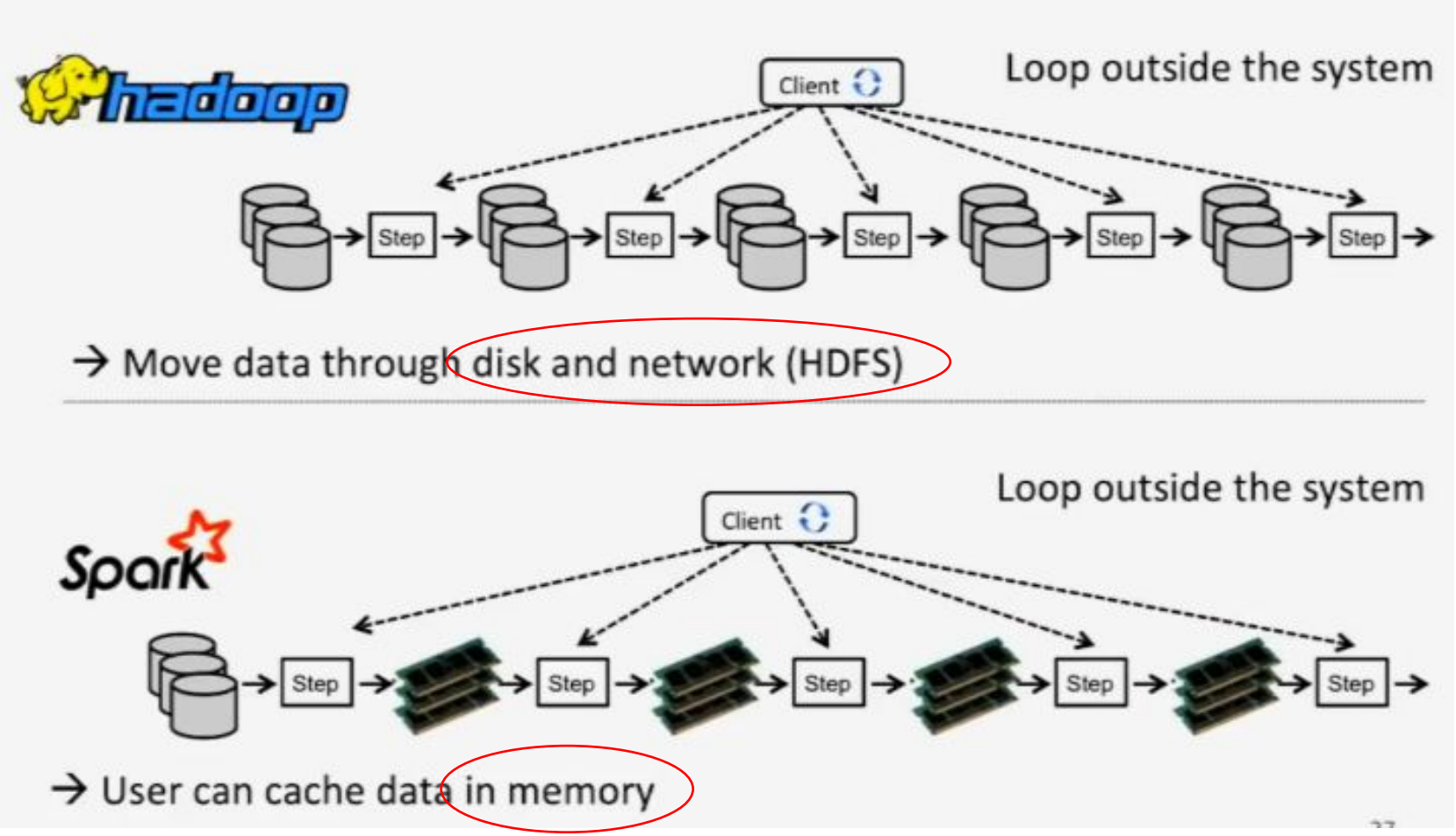


- Spark is fully open source
- Since 2009
  - built by a wide set of developers from over 200 companies
  - more than 1000 developers have contributed to Spark
- Since 2010
  - Spark has become one of the most active projects in Big Data
  - Spark has actually taken over Hadoop MapReduce and every other engine that we are aware of in terms of number of people contributing to it

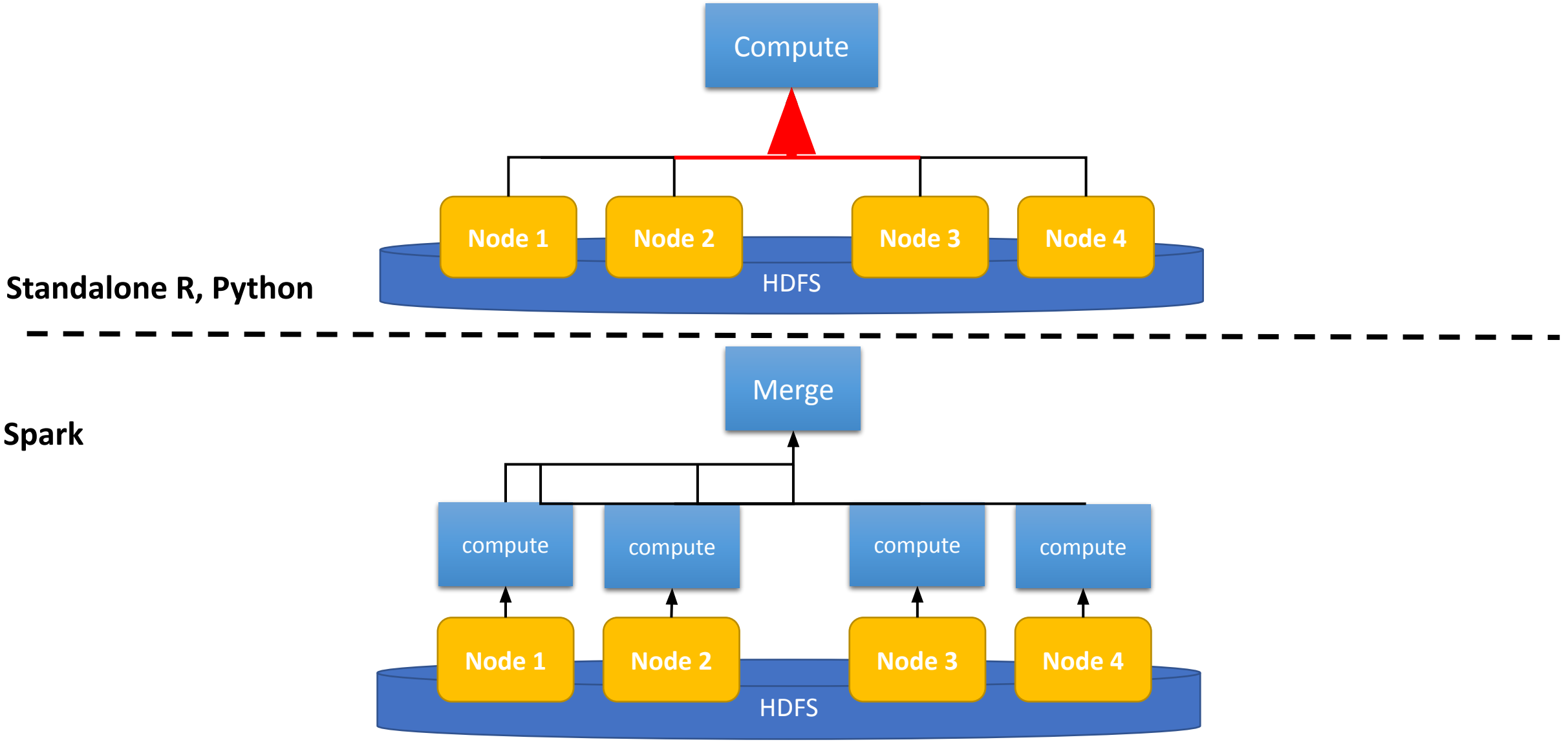




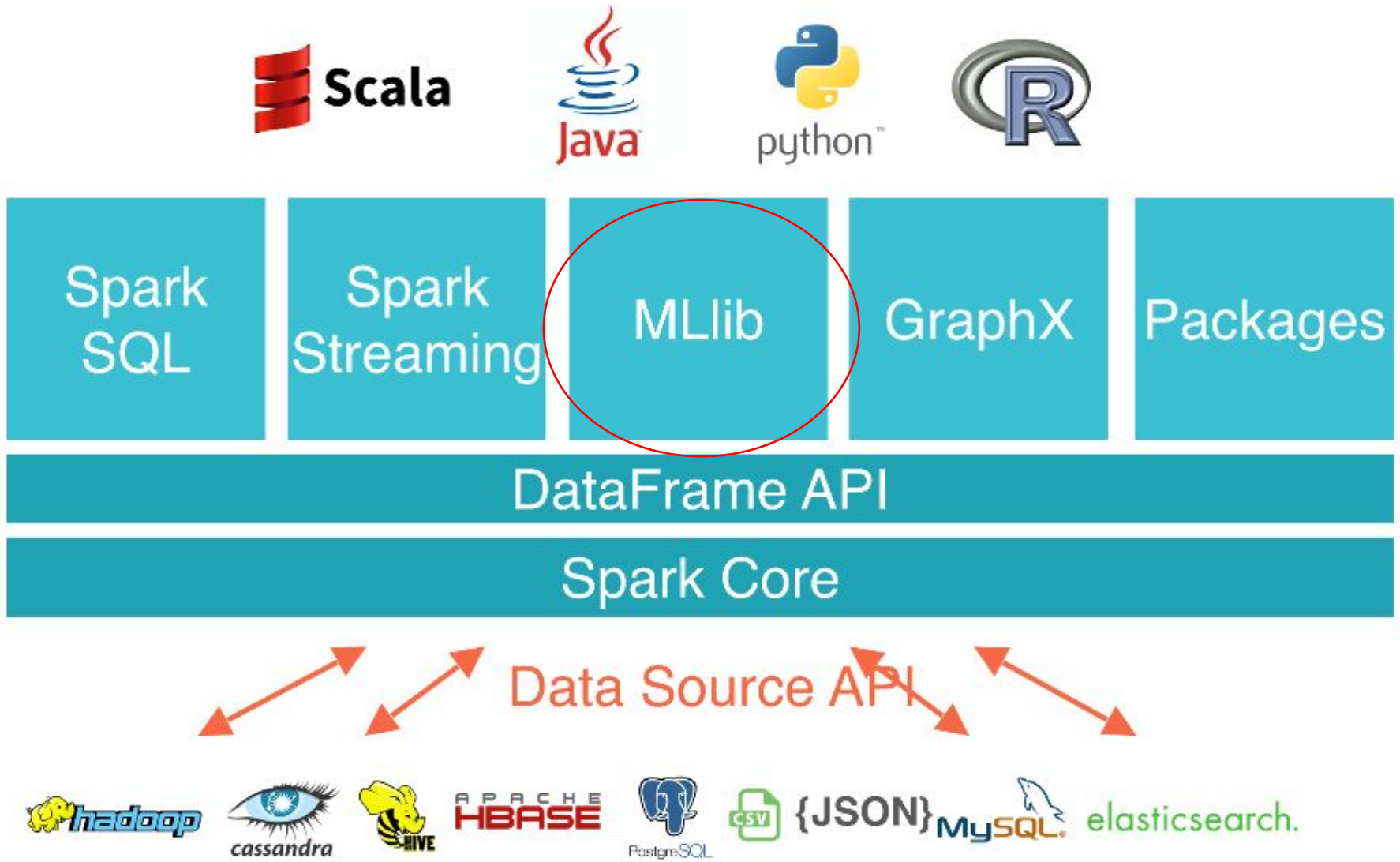
Compare to Hadoop



Compare to standalone R, Python



Conclusion


Scala
Java
python
Spark SQL
Spark Streaming
MLlib
GraphX
Packages
DataFrame API
Spark Core
Data Source API
hadoop
cassandra
HIVE
APACHE HBASE
PostgreSQL
CSV
{JSON}
MySQL
elasticsearch.

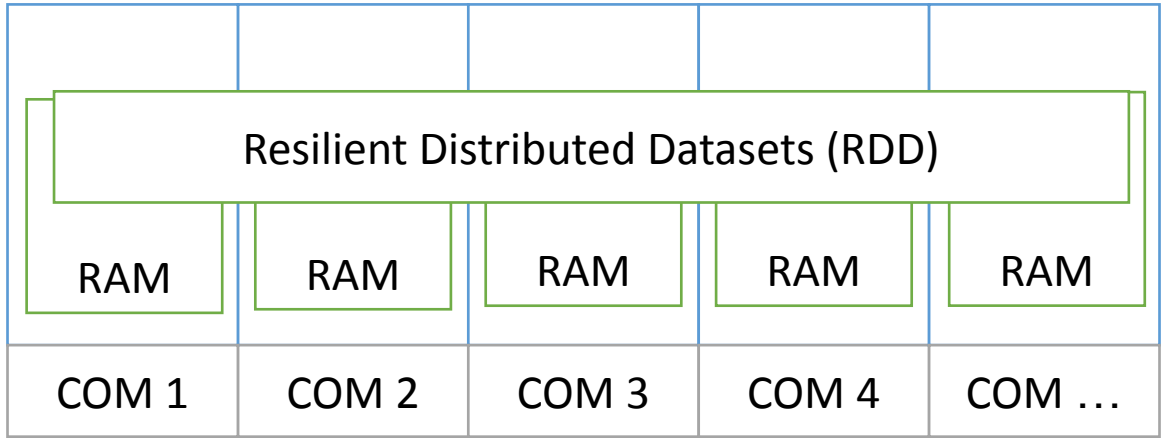
How does Spark work?

Apache Spark in this session

- Apache Spark Core (SparkContext)
  - **RDD**
- Apache Spark SQL (SparkSession)
  - **DataFrame**
- Apache Spark MLlib & ML
  - MLlib : machine learning for Spark RDD
    - will not add new features
    - May be **deprecated** in future release (Spark 2.2)
    - May be removed in future release (Spark 3.0)
  - ML : machine learning for Spark Dataframe
    - “Spark ML” is not an official name but occasionally used to refer to the MLlib DataFrame-based API
    - DataFrames provide a **more user-friendly** API than RDDs

Resilient Distributed Datasets (RDDs)

1. **Immutable** representation of data
2. Operations on one RDD **creates a new one**
3. Memory caching layer that stores data in a distributed, ***fault-tolerant*** **cache**
4. Created by parallel transformations on data in stable storage
5. **Lazy materialization**



Operations

- Spark has certain operations which can be performed on RDD
- **1) Transformation (from RDD to RDD)**:
  - Transformation refers to the operation applied on a **RDD to create new RDD**
  - **Lazy operations** to build RDDs from other RDDs
  - When perform transform operation, it will only store the step of transformation
  - filter, groupBy, map, flatmap
- **2) Action (from RDD to output)**:
  - Actions refer to an operation which also applies on RDD, that instructs Spark to perform computation on all steps of transformation and action then **send the result (output) back to driver**
  - Return a result or write it to storage
  - take, collect, reduce

- Example pyspark transformation and action operations :
  - https://www.analyticsvidhya.com/blog/2016/10/using-pyspark-to-perform-transformations-and-actions-on-rdd/
  - http://spark.apache.org/docs/latest/programming-guide.html#transformations
  - http://spark.apache.org/docs/latest/api/python/pyspark.html

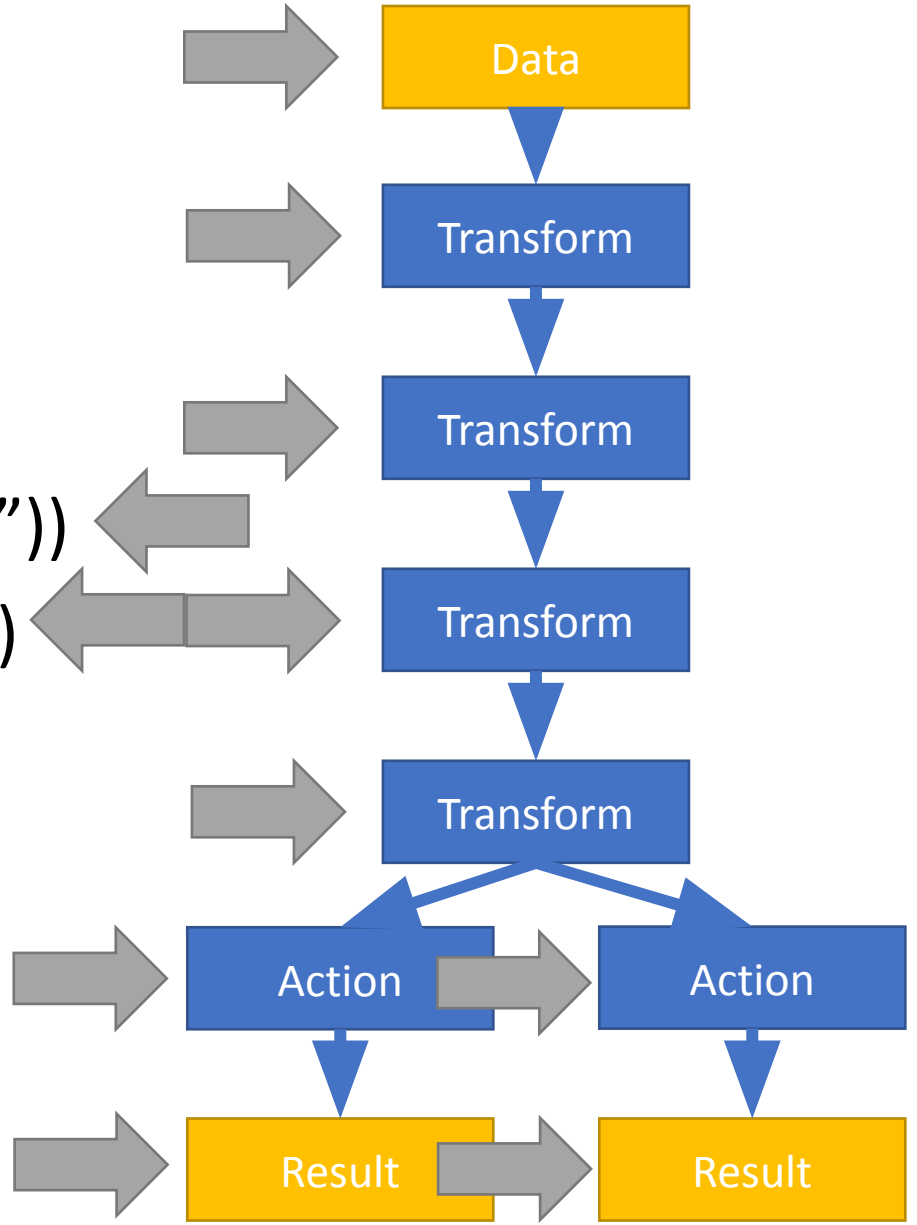
Operations

- Python :

```
data = sc.textFile("File Path")
rdd1 = data.map(lambda x : x.split(","))
rdd2 = rdd1.map(lambda x : tuple(x))
rdd3 = rdd2.groupByKey()
rdd4 = rdd3. mapValues(list)
count = rdd4.count()
result = rdd4.collect()
```

action

action



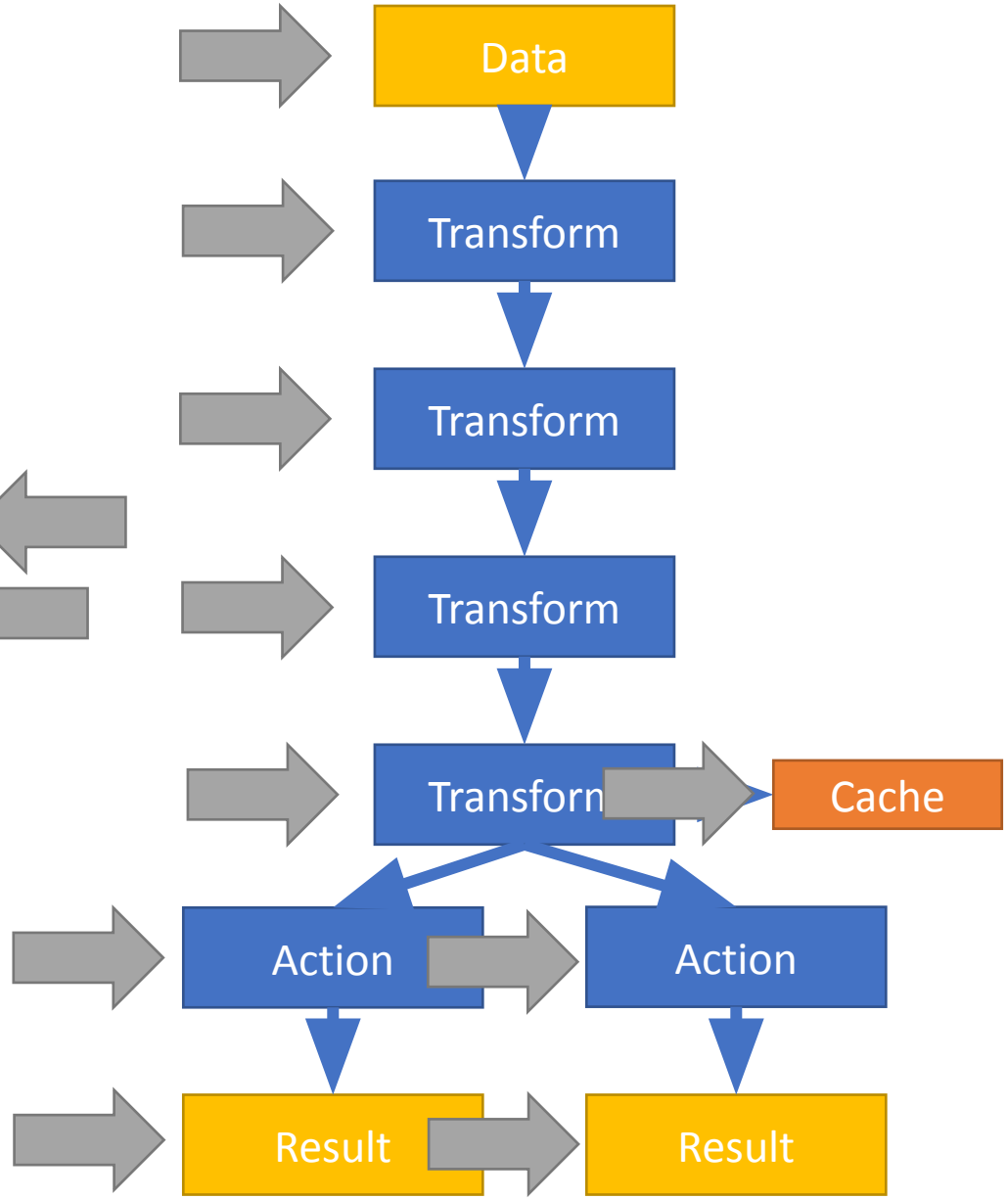
Operations

- Python :

```
data = sc.textFile("File Path")
rdd1 = data.map(lambda x : x.split(","))
rdd2 = rdd1.map(lambda x : tuple(x))
rdd3 = rdd2.groupByKey()
rdd4 = rdd3. mapValues(list)
rdd4.cache()
count = rdd4.count()
result = rdd4.collect()
rdd4.unpersist()
```

action

action

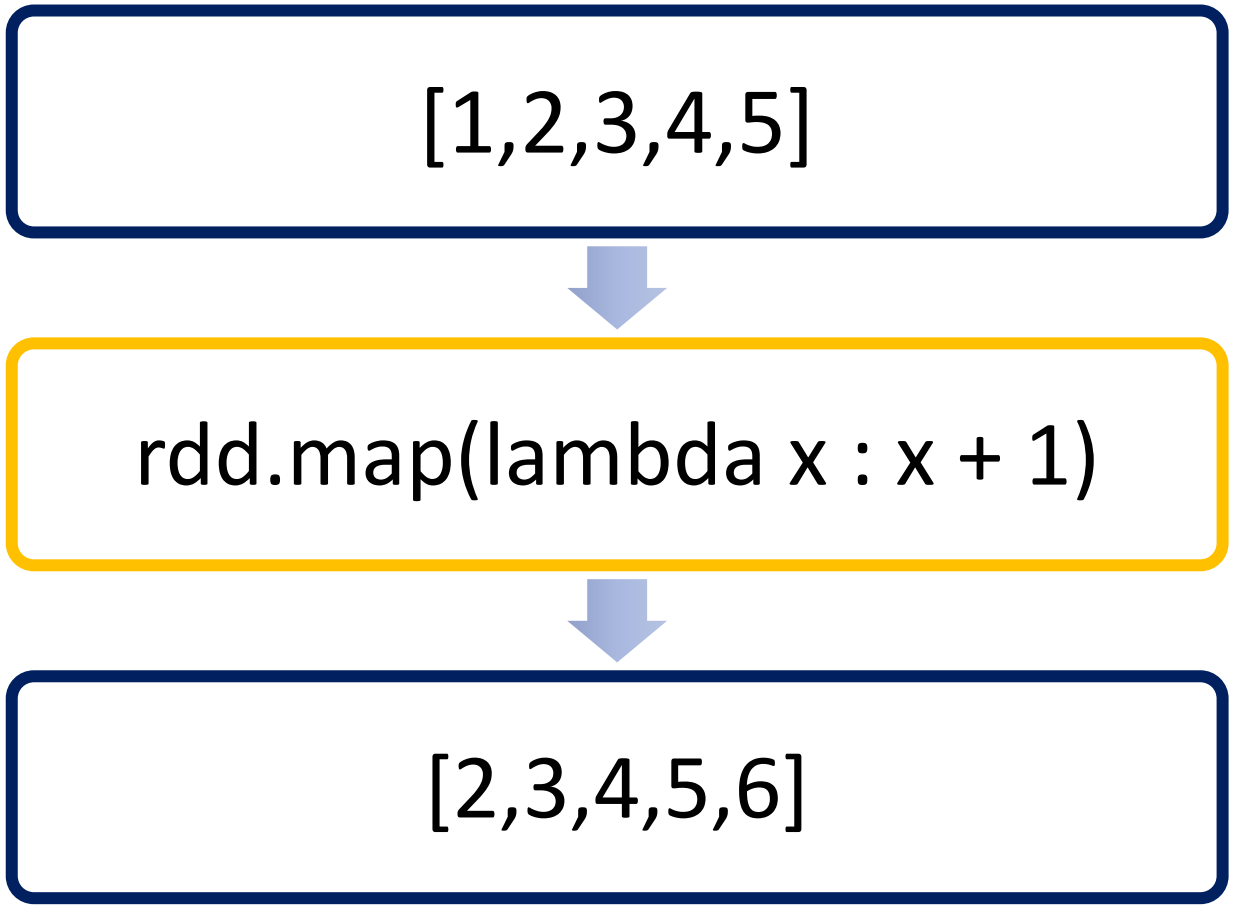


Basic Spark Operation
RDD - Transformation

Operation	Description
map(f, preservesPartitioning=False)	Return a new RDD by applying a function to each element of this RDD.
flatMap(f, preservesPartitioning=False)	Return a new RDD by first applying a function to all elements of this RDD, and then flattening the results.
filter(f)	Return a new RDD containing only the elements that satisfy a predicate.
sample(withReplacement, fraction, seed=None)	Return a sampled subset of this RDD.
union(other)	Return the union of this RDD and another one.
intersection(other)¶	Return the intersection of this RDD and another one. The output will not contain any duplicate elements, even if the input RDDs did.
distinct(numPartitions=None)	Return a new RDD containing the distinct elements in this RDD.
zip(other)	Zips this RDD with another one, returning key-value pairs with the first element in each RDD second element in each RDD, etc. Assumes that the two RDDs have the same number of partitions and the same number of elements in each partition
zipWithUniqueId()	Zips this RDD with generated unique Long ids.
reduceByKey(func, numPartitions=None)	Merge the values for each key using an associative and commutative reduce function. This will also perform the merging locally on each mapper before sending results to a reducer, similarly to a “combiner” in MapReduce.

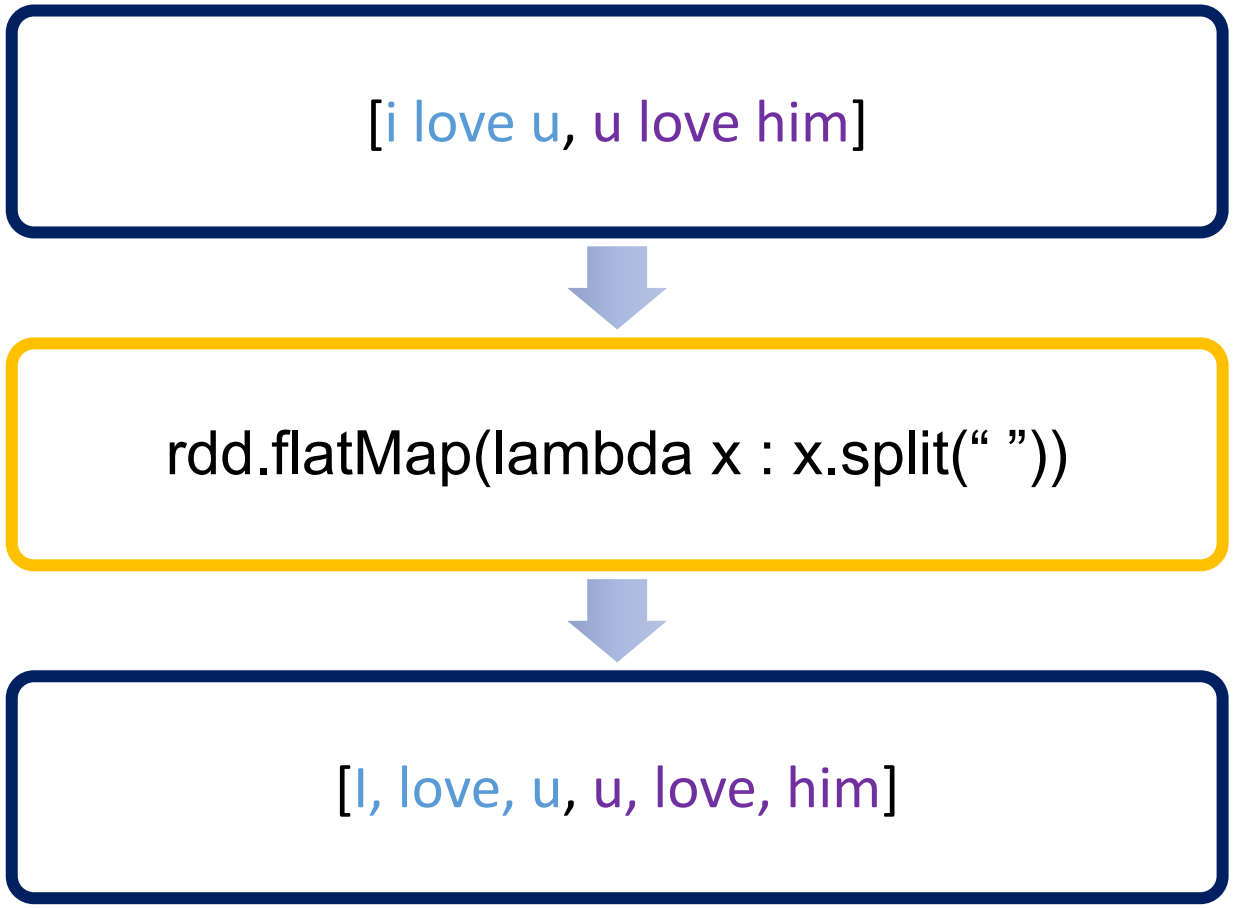
http://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

Basic Spark Operation
RDD - Transformation



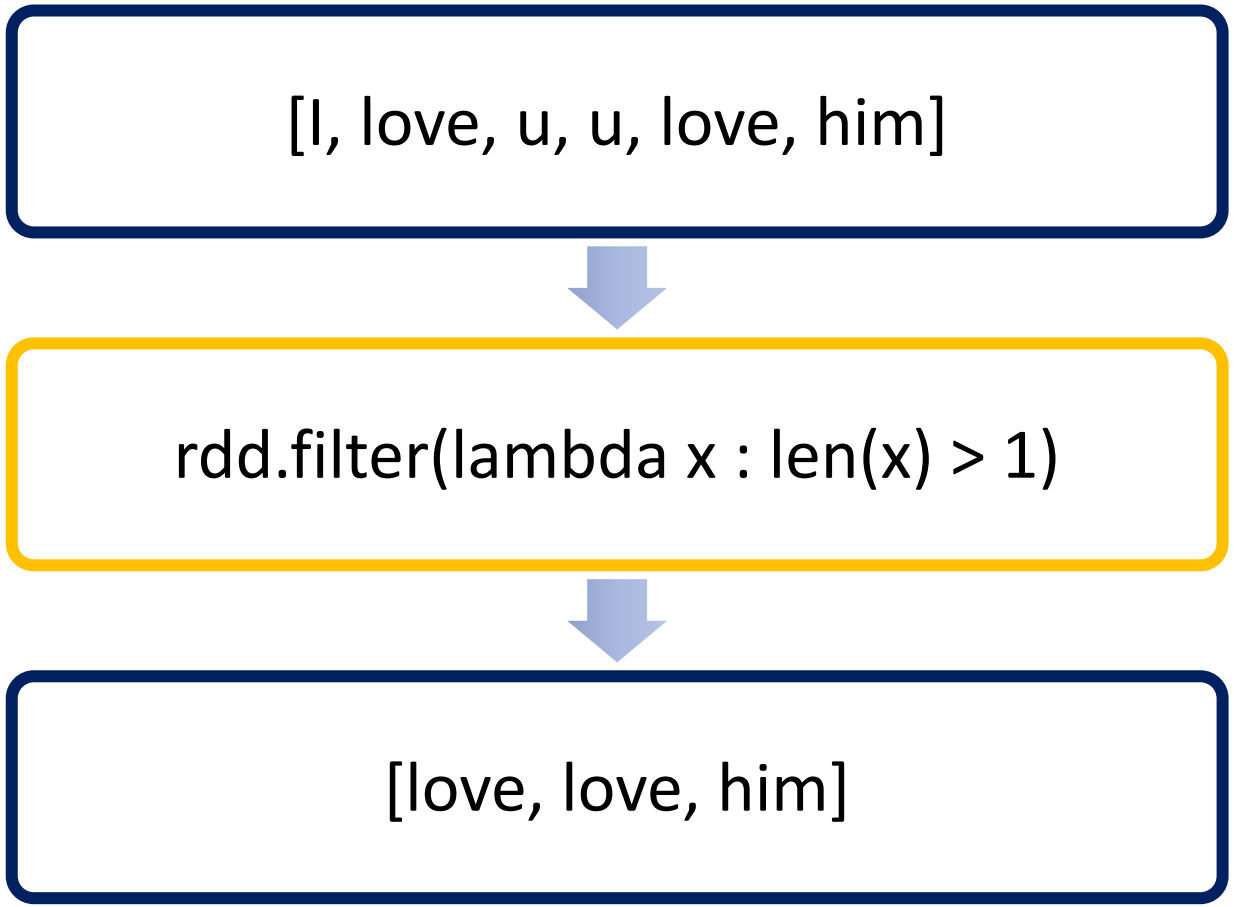
Map: input 1 □ output 1

Basic Spark Operation
RDD - Transformation

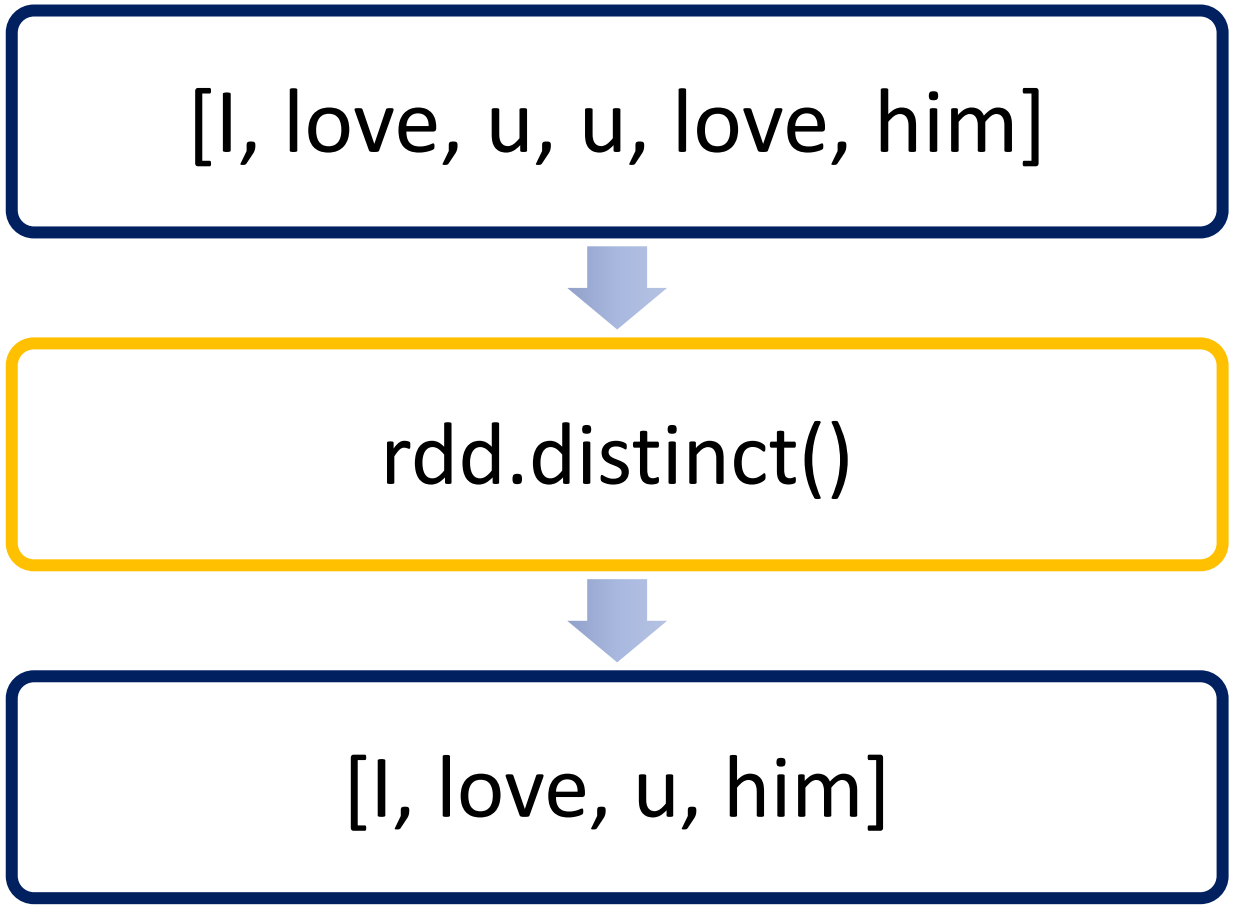


flatMap: input 1 ? output n (0, n)

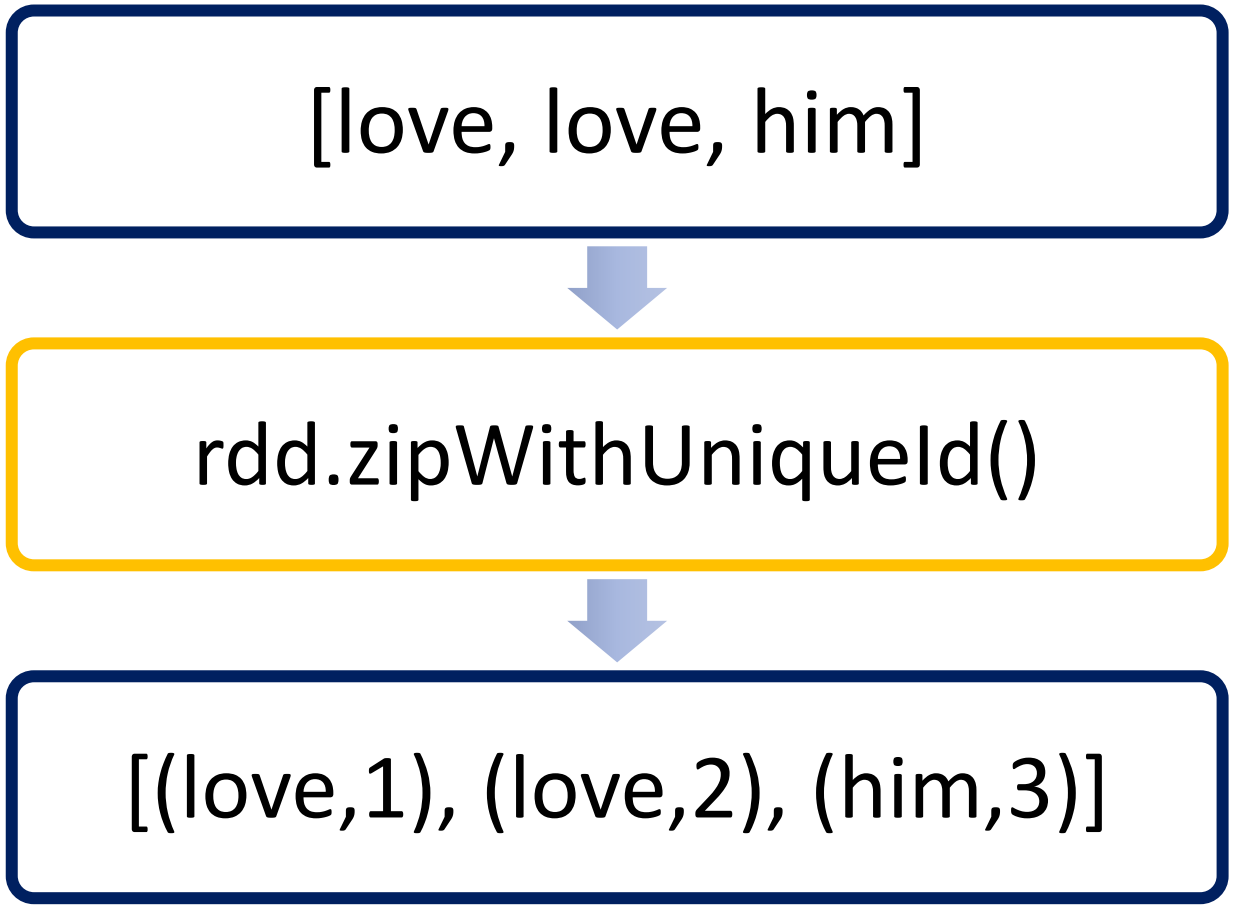
Basic Spark Operation
RDD - Transformation



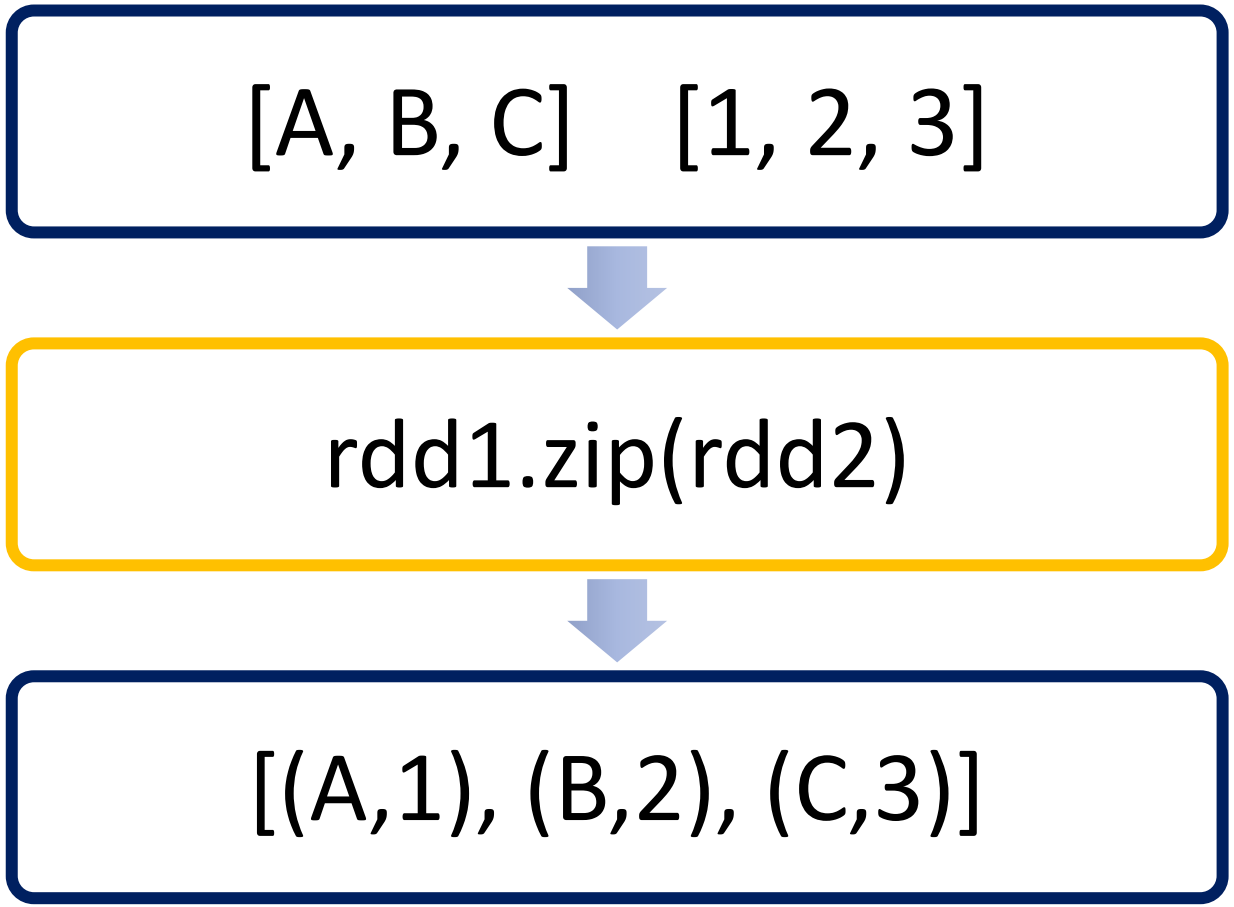
Basic Spark Operation
RDD - Transformation



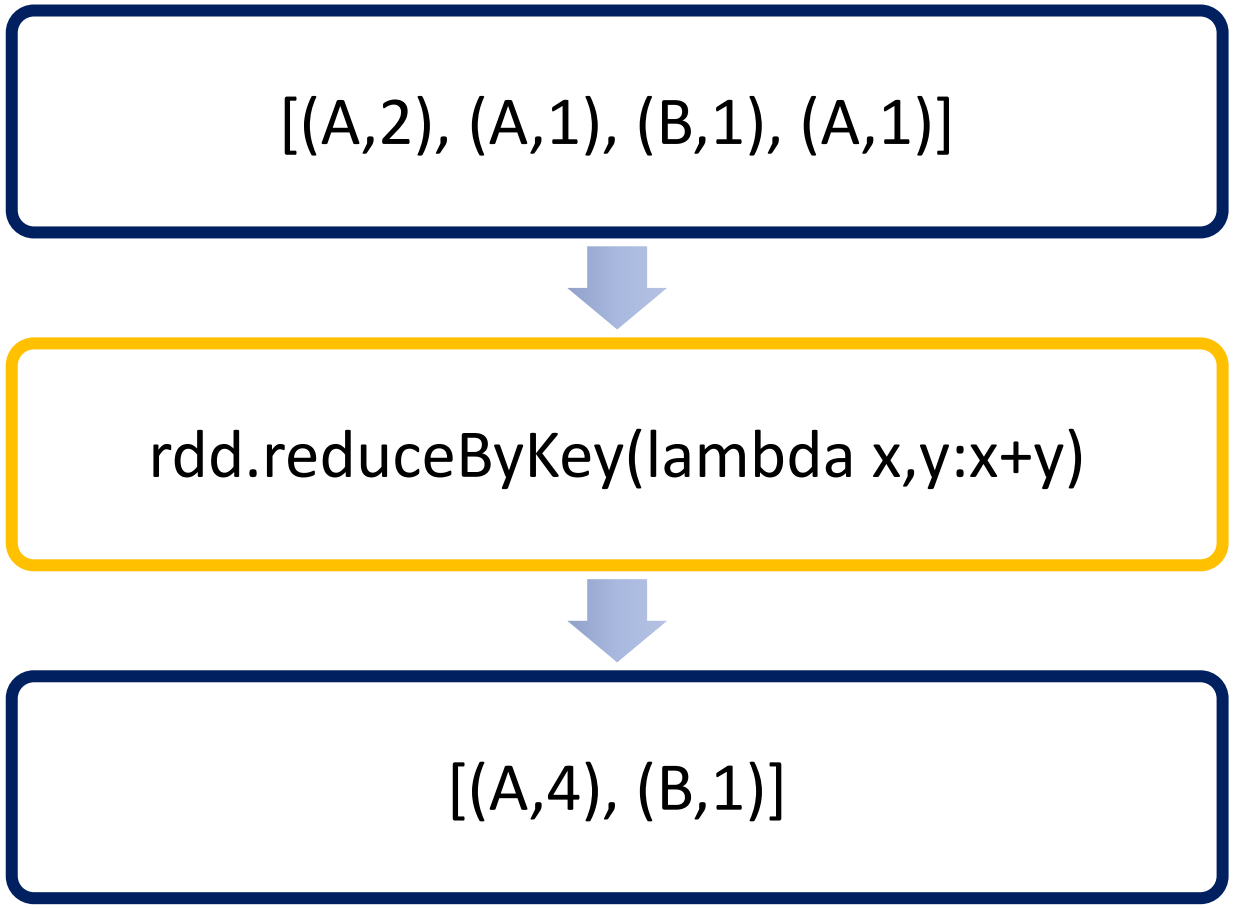
Basic Spark Operation
RDD - Transformation



Basic Spark Operation
RDD - Transformation



Basic Spark Operation
RDD - Transformation

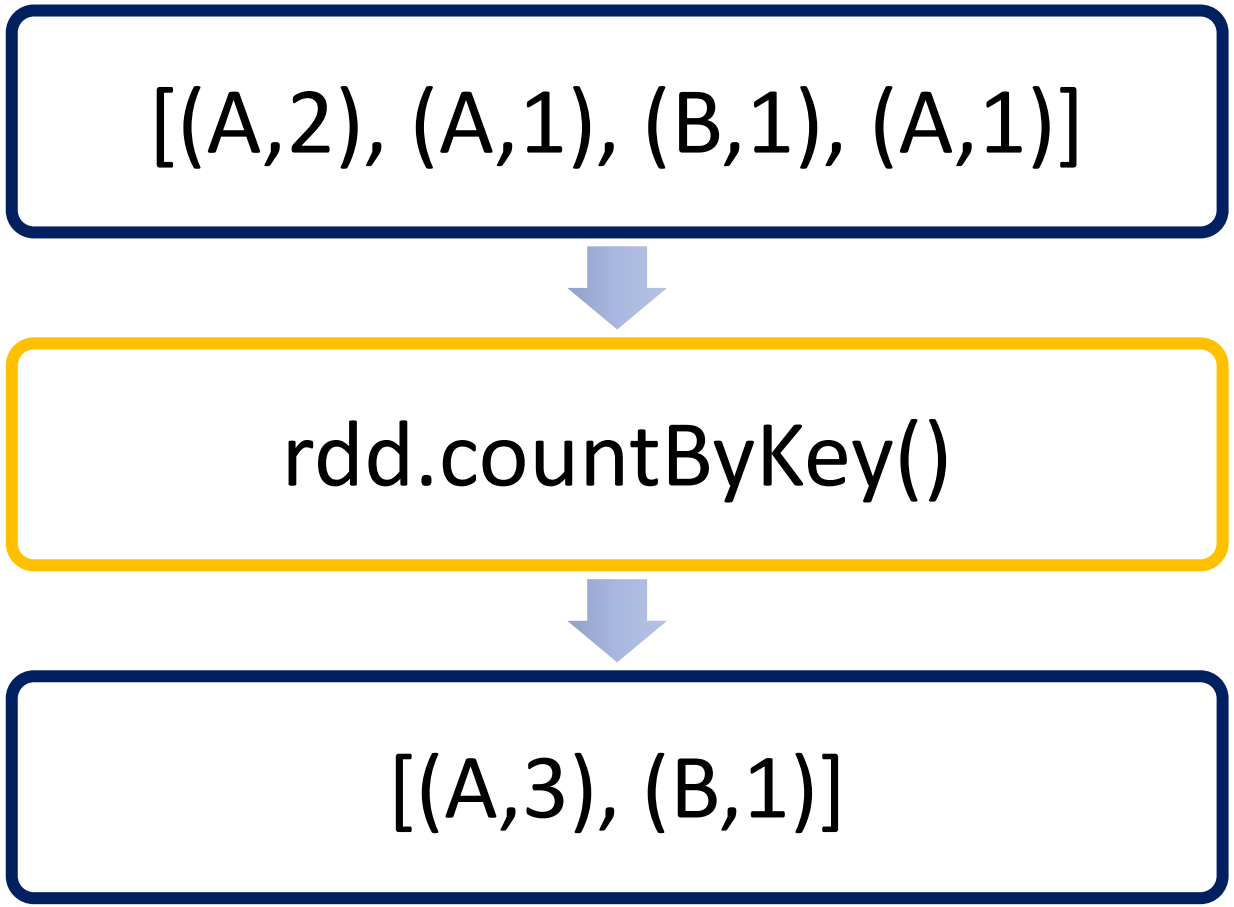


Basic Spark Operation
RDD - Action

Operation	Description
reduce(f)	Reduces the elements of this RDD using the specified commutative and associative binary operator. Currently reduces partitions locally.
collect()	Return a list that contains all of the elements in this RDD
take(num)	Take the first num elements of the RDD.
top(num, key=None)	Get the top N elements from an RDD.
count()	Return the number of elements in this RDD.
saveAsTextFile(path, compressionCodecClass=None)	Save this RDD as a text file, using string representations of elements.
countByKey()	Count the number of elements for each key, and return the result to the master as a dictionary.

http://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

Basic Spark Operation
RDD - Action



Basic Spark Operation: DataFrame

Operation	Description
printSchema()	Prints out the schema in the tree format.
show(n=20, truncate=True)	Prints the first n rows to the console.
selectExpr(*expr)	Projects a set of SQL expressions and returns a new DataFrame. This is a variant of select() that accepts SQL expressions.
withColumn(colName, col)	Returns a new DataFrame by adding a column or replacing the existing column that has the same name.
withColumnRenamed(existing, new)	Returns a new DataFrame by renaming an existing column. This is a no-op if schema doesn't contain the given column name
sample(withReplacement, fraction, seed=None)	Returns a sampled subset of this DataFrame.
union(other), intersect(other)	Return a new DataFrame containing union of rows in this frame and another frame. Return a new DataFrame containing rows only in both this frame and another frame.
groupBy(*cols)	Groups the DataFrame using the specified columns, so we can run aggregation on them. See GroupedData for all the available aggregate functions.
createOrReplaceTempView(name)	Creates or replaces a local temporary view with this DataFrame.
count()	Returns the number of rows in this DataFrame.
columns	Returns all column names as a list.

http://spark.apache.org/docs/latest/api/python/pyspark.sql.html

Basic Spark Operation (cont.): DataFrame

Operation	Description
select(*cols)	Projects a set of expressions and returns a new DataFrame.
drop(*cols)	Returns a new DataFrame that drops the specified column. This is a no-op if schema doesn't contain the given column name(s).
dropDuplicates(subset=None)	Return a new DataFrame with duplicate rows removed, optionally only considering certain columns.
dropna(how='any', thresh=None, subset=None)	Returns a new DataFrame omitting rows with null values. DataFrame.dropna() and DataFrameNaFunctions.drop() are aliases of each other.
fillna(value, subset=None)	Replace null values, alias for na.fill(). DataFrame.fillna() and DataFrameNaFunctions.fill() are aliases of each other.
filter(condition), where(condition)	Filters rows using the given condition. where() is an alias for filter().
collect()	Returns all the records as a list of Row.
rdd	Returns the content as an pyspark.RDD of Row.

http://spark.apache.org/docs/latest/api/python/pyspark.sql.html

Basic Spark Operation DataFrame

Name	Gender	Salary
A	M	24,000
B	M	25,000
C	F	36,000

Create new column

```
df.withColumn("SalaryK", df["Salary"] / 1000)
```

Name	Gender	Salary	SalaryK
A	M	24,000	24
B	M	25,000	25
C	F	36,000	36

Basic Spark Operation
DataFrame

Name	Gender	Salary
A	M	24,000
B	M	25,000
C	F	36,000

replace

```
df.withColumn("Salary", df["Salary"] / 1000)
```

Name	Gender	Salary
A	M	24
B	M	25
C	F	36

Basic Spark Operation DataFrame

Name	Gender	Salary
A	M	24
B	M	25
C	F	36

Just rename, no compute

df.withColumnRenamed(“Salary”, “SalaryK”)

Name	Gender	SalaryK
A	M	24
B	M	25
C	F	36

Basic Spark Operation DataFrame

Name	Gender	Salary
A	M	24
B	M	25
C	F	36

```
df.filter(df["SalaryK"] > 30)
df.where(df["SalaryK"] > 30)
```

Name	Gender	SalaryK
A	M	24
B	M	25
C	F	36

Basic Spark Operation DataFrame

Name	Gender	Salary
A	M	24
B	M	25
C	F	36

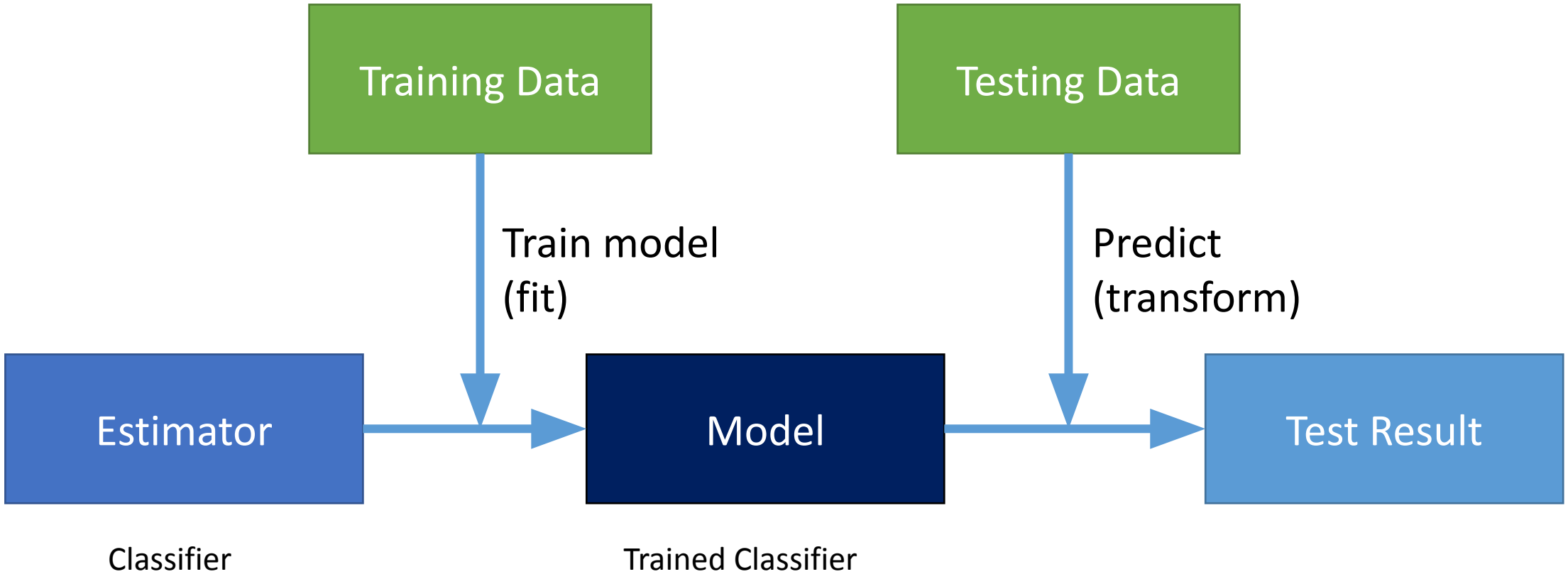
df.select(“Gender”,“SalaryK”)

Gender	SalaryK
M	24
M	25
F	36

Spark MLlib & ML

Basic Spark RDD Operation

Spark ML



Basic Spark Operation
Abstract Class **Estimator** (Spark ML)

Operation	Description
fit(dataset, params=None)	Fits a model to the input dataset with optional parameters. Parameters: dataset – input dataset, which is an instance of pyspark.sql.DataFrame params – an optional param map that overrides embedded params. If a list/tuple of param maps is given, this calls fit on each param map and returns a list of models. Returns: fitted model(s)

Other method in Specific Estimator use to assign parameter for specific algorithm

Example : **DecisionTreeClassifier**

- fit(dataset, params=None)
- setFeaturesCol(value)
- setLabelCol(value)
- setImpurity(value)
- setMaxBins(value)
- setMaxDepth(value)
- setMinInfoGain(value)
- setMinInstancesPerNode(value)
- setSeed(value)

http://spark.apache.org/docs/latest/api/python/pyspark.ml.html
http://spark.apache.org/docs/latest/api/python/pyspark.mllib.html

Basic Spark Operation
Abstract Class **Model** (Spark ML)

Operation	Description
transform(dataset, params=None)	Transforms the input dataset with optional parameters. Parameters: dataset – input dataset, which is an instance of pyspark.sql.DataFrame params – an optional param map that overrides embedded params.
save(path)	Save this ML instance to the given path, a shortcut of write().save(path).
Other method in Specific Model use to get some value and knowledge for specific algorithm	

http://spark.apache.org/docs/latest/api/python/pyspark.ml.html
http://spark.apache.org/docs/latest/api/python/pyspark.mllib.html

Spark ML API

- Feature Extractors
  - TF-IDF
  - Word2Vec
  - CountVectorizer
- Feature Transformers
  - Tokenizer
  - StopWordsRemover
  - n-gram
  - Binarizer
  - PCA
  - PolynomialExpansion
  - Discrete Cosine Transform (DCT)
  - StringIndexer
  - IndexToString
  - OneHotEncoder
  - VectorIndexer
  - Interaction
  - Normalizer
  - StandardScaler
  - MinMaxScaler
  - MaxAbsScaler
  - Bucketizer
  - ElementwiseProduct
  - SQLTransformer
  - VectorAssembler
  - QuantileDiscretizer
  - Imputer

- Feature Selectors
  - VectorSlicer
  - RFormula
  - ChiSqSelector
- Locality Sensitive Hashing
  - LSH Operations
    - Feature Transformation
    - Approximate Similarity Join
    - Approximate Nearest Neighbor Search
  - LSH Algorithms
    - Bucketed Random Projection for Euclidean Distance
    - MinHash for Jaccard Distance

- Linear methods
- Decision trees
  - Inputs and Outputs
    - Input Columns
    - Output Columns
- Tree Ensembles
  - Random Forests
    - Inputs and Outputs
      - Input Columns
      - Output Columns (Predictions)
  - Gradient-Boosted Trees (GBTs)
    - Inputs and Outputs
      - Input Columns
      - Output Columns (Predictions)

- Classification
  - Logistic regression
    - Binomial logistic regression
    - Multinomial logistic regression
  - Decision tree classifier
  - Random forest classifier
  - Gradient-boosted tree classifier
  - Multilayer perceptron classifier
  - Linear Support Vector Machine
  - One-vs-Rest classifier (a.k.a. One-vs-All)
  - Naive Bayes
- Regression
  - Linear regression
  - Generalized linear regression
    - Available families
  - Decision tree regression
  - Random forest regression
  - Gradient-boosted tree regression
  - Survival regression
  - Isotonic regression

- K-means
  - Input Columns
  - Output Columns
- Latent Dirichlet allocation (LDA)
- Bisecting k-means
- Gaussian Mixture Model (GMM)
  - Input Columns
  - Output Columns