Structural Bioinformatics Training Workshop & Hackathon 2018

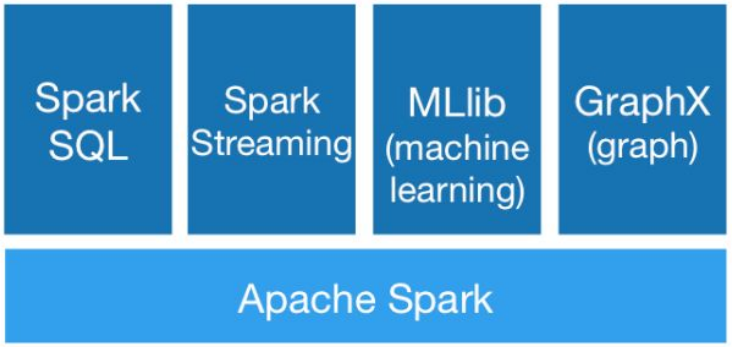
Apache Spark Introduction

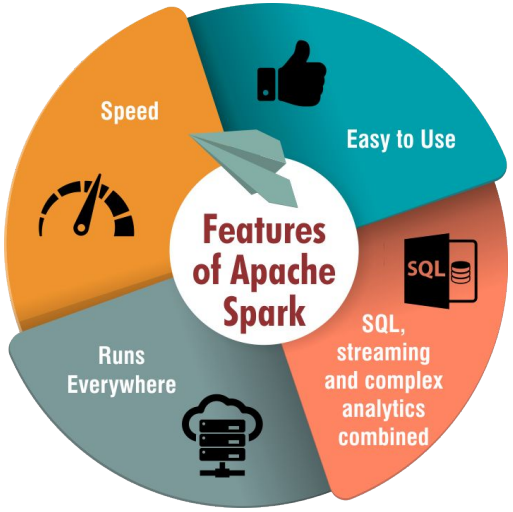
Mars Huang
Structural Bioinformatics Laboratory
San Diego Supercomputer Center
UC San Diego

Introduction to Apache Spark

Apache Spark is an open-source software framework that provides a distributed environment designed to store and process big data.

Apache Spark Ecosystem





Core API: Python, Java, R, Scala

Spark offers support for multiple languages and makes it easy to build parallel applications.

Initialize and close Spark

```
from pyspark.sql import SparkSession
```

SparkSession initializes the spark application

```
spark = SparkSession.builder.master("local[*]")
                    .appName("Tutorial_Example")
                    .getOrCreate()
```

SparkContext is the entry point for interacting with Spark RDD

```
sc = spark.sparkContext
```

Always remember to stop a SparkSession

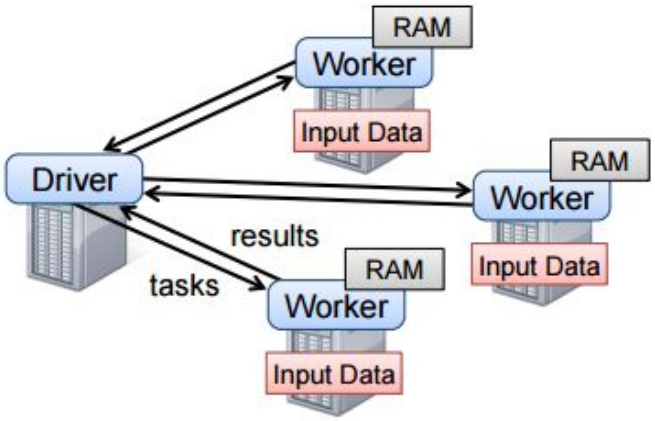
```
spark.stop()
```

More information:
https://spark.apache.org/docs/latest/configuration.html#available-properties

Distributed data structures

Spark revolves around the concept of a *resilient distributed dataset* (RDD):

- core Spark abstraction
- represents partitions across the cluster nodes
- enables parallel processing of datasets
- partitions can be in-memory or on-disk
- partitions can be recomputed on failure



There are two ways to create RDDs:

- parallelizing an existing collection in your driver program
- referencing a dataset in an external storage system

Create RDD

Parallelizing an existing collection:

```
data = [1, 2, 3, 4, 5]
distData = sc.parallelize(data, 4)
```

Collection *Number of partitions*

Referencing to a datasets on external storage:

```
distFile = sc.textFile("<path to your file>")
```

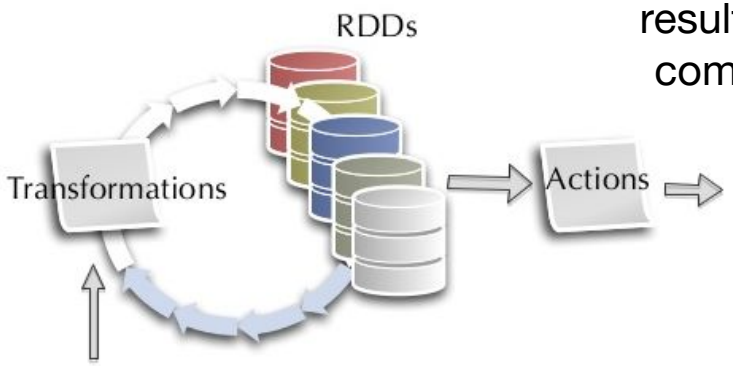
RDD operations

Spark provides a rich set of operators to manipulate RDDs

Transformations
map(func)
flatMap(func)
filter(func)
groupByKey()
reduceByKey(func)
mapValues(func)
…

Actions
take(N)
count()
collect()
reduce(func)
takeOrdered(N)
top(N)
…

Actions return final results of RDD computations



Transformations create new RDD from existing RDD

```
rdd1 = sc.parallelize(data)
rdd2 = rdd1.filter(func1)
rdd2.count()
rdd2.collect()
```

RDD actions: collect()

Spark RDD *collect* function returns all the elements of the dataset as a list

```
rdd = sc.parallelize([1, 2, 3, 4, 5])
output_list = rdd.collect()
```

Note:
Collect function is used in most of the problems in the exercise section.

RDD transformations: filter()

Filter :	filter(f : T ⇒ Bool)	Return a new dataset formed by selecting those elements of the source on which *func* returns true.

Passing lambda functions:

```
rdd2 = rdd1.filter(lambda x: x % 2 == 0)
```

RDD transformations: map() and flatMap()

Map :	map(f : T ⇒ U)	
FlatMap :	flatMap(f : T ⇒ List<U>)	

```
rdd1 = sc.parallelize(data,2)
rdd2 = rdd1.map(lambda x: x*x)

rdd4 = rdd1.flatMap(lambda x: [x**2, x**3])
```

Working with Key-Value Pairs: PairRDD

Calling a function that returns a tuple or list of size 2 will automatically generate a Key-Value Pair

```
emails = sc.textFile(textFilePath)
emails.map(lambda x: x.split('@')).collect()
```

SHOW EXAMPLE OF KEY-VALUE PAIR

Some methods on *SparkContext* produce pair RDD by default for reading files in certain Hadoop formats

```
sc.sequenceFile(path).collect()
```

Transformations on PairRDDs

When called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function *func: (V,V) => V*

```
pairRDD.reduceByKey(Function f)
```

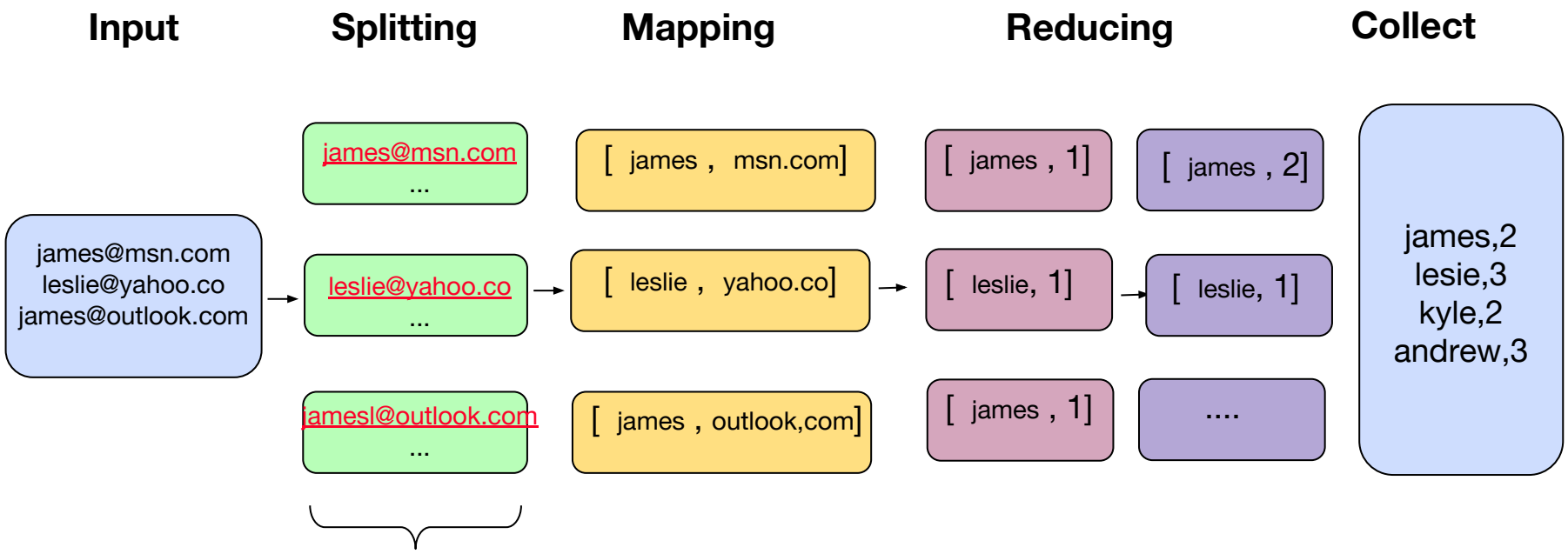
Actions on PairRDDs

collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
count()	Return the number of elements in the dataset.
first()	Return the first element of the dataset (similar to take(1)).
take(*n*)	Return an array with the first *n* elements of the dataset.

```
pairRDD.count()
```

saveAsTextFile(*path*)	Write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem
saveAsSequenceFile(*path*)	Write the elements of the dataset as a Hadoop SequenceFile in a given path in the local filesystem

Count number of domains with the same username with Spark

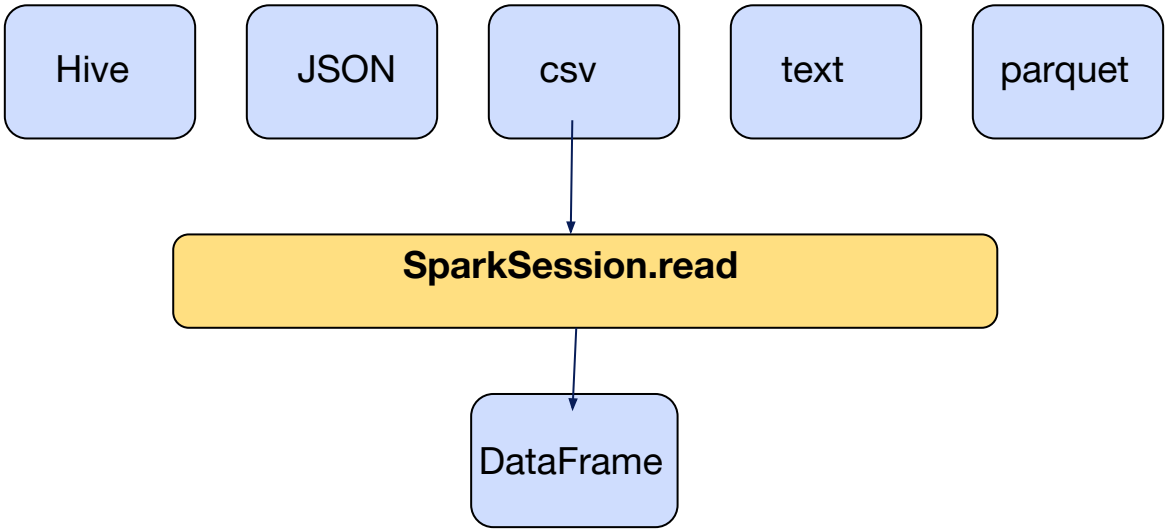



```
emails = sc.textFile("emails.txt")
```

```
pairRdd.reduceByKey(lambda x,y: x+y)
```

Spark DataFrame

A DataFrame distributed collection of data organized into named columns. It is conceptually equivalent to a table in relational databases or a dataframe in R/ Python




```
df = spark.read.csv("./cars.csv", header = True, inferSchema = True)
```

Spark DataFrame Operations

show(n)	Prints the first n rows to the console
filter()	Filter dataframe using given condition
select(cols)	Selects a few columns of a dataframe and returns a new dataframe
groupBy()	Group by values of a specific column, usually follows by an action (count)
sort(n)	Sort dataframe by a specific column
toPandas()	Transform a Spark Dataframe to a pandas DataFrame

```
df_2015 = df.filter(df['YEAR'] == 2015)

df_sub = df.select(df['Make'], df['Model'], df['Size'])

df_manufacturer = df.groupBy('Make').count()

df_sorted = df_manufacturer.sort('count', ascending=False)
```

Resources

- **Apache Spark Programming Guide**
  - https://spark.apache.org/docs/latest/programming-guide.html
- **Pyspark cheatsheet:**
  - https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PySpark_Cheat_Sheet_Python.pdf
- **UC Berkeley Ampcamp**
  - http://ampcamp.berkeley.edu/
- **Coursera**
  - https://www.coursera.org/learn/hadoop/lecture/9cq0R/introduction-to-apache-spark
  - https://www.coursera.org/learn/hadoop/lecture/v7hd5/architecture-of-spark
- **edX**
  - https://www.edx.org/course/introduction-apache-spark-uc-berkeleyx-cs105x

Funding

This workshop was supported by the National Cancer Institute of the National Institutes of Health under Award Number U01CA198942. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.