MSAN 694 : Distributed Computing

Diane Woodbridge, Ph.D.
MSAN, University of San Francisco

Reviews

Class Overview

Motivation - Why Distributed Computing?

What is Distributed Computing?

Spark

RDD Creation

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Spark Interview Questions

~~What is Apache Spark?~~

~~Explain the key features of Spark.~~

~~What is RDD?~~

~~How to create RDD.~~

What is ”partitions”?

Types or RDD operations?

What is “transformation”?

What is “action”?

~~Functions of “spark core”?~~

~~What is “spark context”?~~

What is an “RDD lineage”?

Which file systems does Spark support?

~~List the various types of “Cluster Managers” in Spark.~~

What is “YARN”?

What is “Mesos”?

What is a “worker node”?

What is an “accumulator”?

~~What is “Spark SQL” (Shark)?~~

~~What is “SparkStreaming”?~~

~~What is “GraphX”?~~

~~What is “MLlib”?~~

https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Spark Interview Questions

~~What are the advantages of using Apache Spark over Hadoop MapReduce for big data processing?~~

~~What are the languages supported by Apache Spark for developing big data applications?~~

~~Can you use Spark to access and analyze data stored in Cassandra databases?~~

Is it possible to run Apache Spark on Apache Mesos?

How can you minimize data transfers when working with Spark?

Why is there a need for broadcast variables?

~~Name a few companies that use Apache Spark in production.~~

What are the various data sources available in SparkSQL?

What is the advantage of a Parquet file?

What do you understand by Pair RDD?

Is Apache Spark a good fit for Reinforcement learning?

https://www.dezyre.com/article/top-50-spark-interview-questions-and-answers-for-2016/208

Contents

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Contents

RDD Creation

Two ways of creating RDDs.

1. Loading an external data.
   `lines = sc.textFile("README.md")`
2. Takes a collection such as `Seq` (`Array` or `List`) and creates RDD from its element and distribute to Spark executors in the process.

```
lines = sc.parallelize(["spark",
"spark is fun!"])
```

- Check the number of partitions.

```
lines.getNumPartitions()
```

https://spark.apache.org/docs/1.2.0/configuration.html

Note :
Python Lambda Expression

Python Lambda Expression

- A shortened way to define functions inline.
  - Create an anonymous function using the "lambda" keyword at runtime.

```
def f(x):
    return x+2
f(2)

g = lambda x : x+2
g(2)
```

➔ Can be used as a function parameter for RDD operations.

https://docs.python.org/3/tutorial/controlflow.html
http://www.secnetix.de/olli/Python/lambda_functions.hawk

Contents

RDD Operations

Two types

1. Transformation
   - Produce a new RDD by performing data manipulation on another RDD.
   - Ex. map, filter, flatmap, mapPartitions, sample, union, intersection, distinct, groupByKey, reduceByKey, aggregateByKey, sortByKey, join, cogroup, cartesian, pipe, coalesce, repartition, repartitionAndSortWithinPartitions.
2. Actions
   - Trigger a computation to return the result to the calling program or to perform some actions on an RDD's elements.
   - Ex. reduce, collect, count, first, take, takeSample, takeOrdered, saveAsTextFile, saveAsSequenceFile, saveAsObjectFile, countByValue, foreach.

https://spark.apache.org/docs/1.2.0/programming-guide.html

Contents

RDD Operations - Transformation

Construct a new RDD from an existing RDD.

```
line_with_spark =
lines.filter(lambda lines :
"spark" in lines)
```

Lazy Evaluation. (cf. Hadoop MapReduce)

- Computation doesn't take place until an action is triggered.

Return RDDs.

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

RDD Operation - Transformation

Transformation Operation Types

map(*func*)	Return a new distributed dataset formed by passing each element of the source through a function *func*.
filter(*func*)	Return a new dataset formed by selecting those elements of the source on which *func* returns true.
flatMap(*func*)	Similar to map, but each input item can be mapped to 0 or more output items (so *func* should return a Seq rather than a single item).
mapPartitions(*func*)	Similar to map, but runs separately on each partition (block) of the RDD, so *func* must be of type Iterator<T> => Iterator<U> when running on an RDD of type T.
mapPartitionsWithIndex(*func*)	Similar to mapPartitions, but also provides *func* with an integer value representing the index of the partition, so *func* must be of type (Int, Iterator<T>) => Iterator<U> when running on an RDD of type T.
sample(withReplacement, fraction, seed)	Sample a fraction *fraction* of the data, with or without replacement, using a given random number generator seed.
distinct([*numTasks*]))	Return a new dataset that contains the distinct elements of the source dataset.
union(*otherDataset*)	Return a new dataset that contains the union of the elements in the source dataset and the argument.
intersection(*otherDataset*)	Return a new RDD that contains the intersection of elements in the source dataset and the argument.

RDD Operation - Transformation

Element-wise Transformation

- map(func)
  - Apply a function to each element in the RDD.
- flatMap(func)
  - Call each element in RDD individually.
  - Concatenates multiple arrays into a collections that has one level structure.
- filter(func)
  - Return an RDD that passes the filtering requirement.

Example 1-1

Load a text file(“ignatian_pedagogy”) and split each line by space.

Example 1-1

Load a text file("ignatian_pedagogy") and split each line by space.

```
lines = sc.textFile("../Data/ignatian_pedagogy")

lines.collect()

words = lines.map(lambda line : line.split())

words.collect()
```

Create am RDD representing the lines of text in a file.

Example 1-2

Generate a list of words within one level structure.

Example 1-3

Find words including “USF”.

RDD Operation - Transformation

Partition-wise Transformation

- mapPartitions(func)
  - Return a new RDD by applying a function to each partition of the RDD.
- mapPartitionsWithIndex(func)
  - Return a new RDD by applying a function to each partition of the RDD, while tracking the index of the original partition.

Example 2

Parallelize numbers between 1 and 16.

Calculate the count and sum in each partition.

RDD Operation - Transformation

Set Operation

- Format : rdd1.operator(rdd2)
- distinct()
  - Return only one of each element.
- union()
  - If there are duplicated elements, it returns all duplicates.
- intersection()
  - Return common elements.
- subtract()
  - Return elements that are in rdd1 only.
- cartesian()
  - Return cartesian product (all pairs between rdd1 and rdd2)

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Example 3-1

Find distinct words in “ignatian_pedagogy".

Example 3-2

Create a flatmap of distinct words from “README.md”

Example 3-3

What is union, intersection, subtract and cartesian product of the sets from Example 3-1 and Example 3-2?

Contents

RDD Operation - Action

Compute a result based on an RDD.

Return the result to the driver program or save it to external storage system.

```
line_with_spark.count()
```

Return non-RDDs.

RDD Operation - Action

Action Operation Types

reduce(func)	Combine the elements of the RDD together in parallel. (eg. Sum)
fold(zero)(func)	Same as reduce(), but with the provided zero value.
aggregate(zero)(SeqOp, combOp)	Similar to reduce() but used to return a different type.
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.
countByValue()	Return the number of times each elemnt occurs in the RDD.
take(n)	Return an array with the first n elements of the dataset.
top(n)	Return the top n elements of the RDD.
first()	Return the first element of the dataset (similar to take(1)).
takeSample(withReplacement, num, [seed])	Return an array with a random sample of num elements of the dataset, with or without replacement, optionally pre-specifying a random number generator seed.
foreach(func)	Run a function func on each element of the dataset. This is usually done for side effects such as updating an Accumulator or interacting with external storage systems. Note: modifying variables other than Accumulators outside of the foreach() may result in undefined behavior. See Understanding closures for more details.

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

RDD Operation - Action

- reduce(func)
  - Take a function that operates on two elements of the type in your RDD.
  - Returns a new element of the same type.

    Ex. sum = rdd.reduce(lambda x, y = x+y)
- fold(zeroValue)(func)
  - Take a function with the same signature as needed for reduce().
  - Take a “zero value” to be used for the initial call on each partition.
  - Returns a new element of the same type.
- aggregate(zeroValue)(seqOp, combOp)
  - Supply an initial zero value of the type we want to return.
  - seqOp : Function to combine the elements from the RDD with the accumulator. Runs once in a partition.
  - combOp : Function to merge two accumulators, given that each nodes accumulates its own results locally.
  - Can return an element of a different type.

Example 4-1

For the numbers between 1 and 9, calculate sum of the odd numbers.

Example 4-2

For the numbers between 1 and 9, calculate sum of the odd numbers using fold().

Example 4-3

Using aggregate(), return (sum, # of elements) of odd numbers.

RDD Operation - Action

- collect()
  - Return the entire RDD's contents.
- count()
  - Return the count of elements.
- countByValue()
  - Return the number of times each element occurs in the RDD.
- top(n)
  - Return top elements from an RDD, using the default ordering on the data.
- take(n)
  - Return n elements.
- first()
  - Return the first element of the data.
- takeSample(withReplacement, num, seed)
  - Return a fixed-size sample subset of an RDD.
- foreach()
  - Used for performing computations on each element in the RDD.

RDD Operation - Action

sample() vs. takeSample()

- sample(withReplacement, fraction, seed) : **Transformation**
  - Creates a new RDD with random elements from the calling RDD.
    - withReplacement : Allow sample multiple times.
    - fraction :
      - Expected number of times each element is going to be sampled (positive double), when replacement is used.
      - Expected probability that each element is going to be sampled (between 0 and 1), when replacement is not used.
    - seed : Random number generation. (Same seeds generates the same numbers.)

RDD Operation - Action

sample() vs. takeSample()

- takeSample(withReplacement, num, seed) : **Action**
  - Return a fixed-size sample subset of an RDD as an array (not RDD).
    - withReplacement : Allow sample multiple times.
    - num : The exact number of sampled element. (Integer)
    - seed : Random number generation. (Same seeds generates the same numbers.)

Example 5-1

```
x = sc.parallelize([3,4,1,2])
y = sc.parallelize(range(2,6))
z = x.union(y)
```

Try collect(), count(), countByValue(), top(n), take(n), first(), takeSample() operations on z.

RDD Operation - Action

Numeric RDD Action Types

count()	Return the number of elements in the RDD.
mean()	Return the mean of the RDD's elements.
sum()	Add up the elements in the RDD.
max()	Return the maximum item in the RDD.
min()	Return the minimum item in the RDD.
variance()	Return the variance of the RDD's elements.
stdev()	Return the standard deviation of the RDD's

https://spark.apache.org/docs/1.1.1/api/python/pyspark.rdd.RDD-class.html

Contents

RDD Creation

RDD Operations - Transformations

RDD Operations - Actions

Run python script in Spark

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Run python script in Spark

```
from pyspark import SparkConf, SparkContext
```

Create SparkConf object to configure the application.

```
conf = SparkConf().setMaster("local[*]").setAppName("AppName")
```

Initializing a SparkContext (SC).

```
sc = SparkContext(conf = conf)
```

For closing Spark, call `sc.stop()`

Unset your environment variables.

Run your standalone program.

```
spark-submit yourscript
```

Run python script in Spark

```
from pyspark import SparkConf, SparkContext
```

Create SparkConf object to configure the application.

```
conf = SparkConf().setMaster("local[*]").setAppName("AppName")
```

Initializing a SparkContext (SC).

```
sc = SparkContext(conf = cor
```

For closing Spark, call `sc.stop()`

Unset your environment variables.

Run your standalone program.

```
spark-submit yourscript
```

Cluster URL
"local[*]" : indicate all thread on the local machine.

Run python script in Spark

```
from pyspark import SparkConf, SparkContext
```

Create SparkConf object to configure the application.

```
conf = SparkConf().setMaster("local[*]").setAppName("AppName")
```

Initializing a SparkContext (SC).

```
sc = SparkContext(conf = conf)
```

For closing Spark, call `sc.stop()`

Unset your environment variables.

Run your standalone program.

```
spark-submit yourscript
```

Identify the application if you're connecting to a cluster.

Example 6

Write a python script (.py) for printing the number of lines in ”README.md” and the first line and run on spark.

Example 6

```
from pyspark import SparkConf, SparkContext

#Create SparkContext
conf =
SparkConf().setMaster("local[*]).setAppName("Ap
pName")

sc = SparkContext(conf = conf)

#Load Data.
lines=sc.textFile("../Data/README.md")
print(lines.count())
print(lines.first())

sc.stop()
```

```
unset PYSPARK_DRIVER_PYTHON
unset PYSPARK_DRIVER_PYTHON_OPTS
spark-submit ex6.py > output_file.txt
```

References

Distributed Computing with Spark, Reza Zadeh, http://stanford.edu/~rezab/slides/bayacm_spark.pdf

Spark Online Documentation : http://spark.apache.org/docs/latest/

Spark RDD Class: https://spark.apache.org/docs/1.1.1/api/python/pyspark.rdd.RDD-class.html

Karau, Holden, et al. Learning spark: lightning-fast big data analysis. O'Reilly Media, Inc., 2015.

Zecevic, Petar, et al. Spark in Action, Manning, 2016.

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO