MSAN 694 : Distributed Computing

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

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Reviews

Pair RDD

Pair RDD Operation - Transformation

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/

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

Contents

Pair RDDs - Operations

Actions on Two Pair RDDs

Function name	Purpose
subtractByKey(otherDataset)	Remove elements with a key present in the other RDD.
join(otherDataset)	Perform an inner join between two RDDs.
rightOuterJoin(otherDataset)	Perform a join between two RDDs where the key must be present in the first RDD.
leftOuterJoin(otherDataset)	Perform a join between two RDDs where the key must be present in the other RDD.
cogroup(otherDataset)	Group data from both RDDs sharing the same key.

http://spark.apache.org/docs/latest/programming-guide.html
https://spark.apache.org/docs/0.6.2/api/core/spark/PairRDDFunctions.html

Pair RDDs - Operations

Actions on Two Pair RDDs

- subtractByKey(otherDataset)
  - Return an RDD with the pairs whose keys are not in otherDataset.

http://spark.apache.org/docs/latest/programming-guide.html
https://spark.apache.org/docs/0.6.2/api/core/spark/PairRDDFunctions.html

Example 1

Using “filtered_registered_business_sf.csv” and “supervisor_sf.csv”, list business names without a district supervisor.

Pair RDDs - Operations

Actions on Two Pair RDDs

- join(otherDataset) - Perform an inner join between two RDDs.
- rightOuterJoin(otherDataset) - Perform a join between two RDDs where the key must be present in the first RDD.
- leftOuterJoin(otherDataset) - Perform a join between two RDDs where the key must be present in the other RDD.

http://spark.apache.org/docs/latest/programming-guide.html
https://spark.apache.org/docs/0.6.2/api/core/spark/PairRDDFunctions.htmlz

Pair RDDs - Operations

Actions on Two Pair RDDs

- cogroup(otherDataset)
  - Group data from both RDDs sharing the same key.
  - Go over two RDDs sharing the same key.
  - Return the key and the respective lists from two RDD values.
    - Pairs of (Key, (Resulterable, ResultIteratable)).
  - Can work on more than two RDDs at once.

http://spark.apache.org/docs/latest/programming-guide.html
https://spark.apache.org/docs/0.6.2/api/core/spark/PairRDDFunctions.html

Example 2

first_num_pairs = sc.parallelize({(2,3),(1,2),(1,3),(2,4),(3,6)})

second_num_pairs = sc.parallelize({(1,3),(2,2)})

- What are the results of …

1. first_num_pairs.subtract(second_num_pairs)
2. first_num_pairs.subtractByKey(second_num_pairs)
3. first_num_pairs.join(second_num_pairs)
4. first_num_pairs.rightOuterJoin(second_num_pairs)
5. first_num_pairs.leftOuterJoin(second_num_pairs)
6. first_num_pairs.cogroup(second_num_pairs)

Example 3

Using Example 1, list (zip,(business_name, supervisor_id)) pairs **ordered by supervisor_id.**

- Only if both business and supervisor exist.
- If a business exists.
- If a supervisor exists.

Example 4

Genearte Key(zip) and value pair RDDs from “filtered_registered_business_sf.csv” and “supervisor_sf.csv” and cogroup() the RDDs.

Pair RDDs

Additional Actions

- As pair RDDs are a subset of RDDs, it support the same functions as RDDs.
  - Ex. filter, map, etc.
  - Additional actions available on Pair RDDs.

Function name	Purpose
countByKey()	Count the number of elements for each key.
lookup(key)	Return all values associated with the provided key.

http://spark.apache.org/docs/latest/programming-guide.html
https://spark.apache.org/docs/0.6.2/api/core/spark/PairRDDFunctions.html

Example 5

From ”filtered_registered_business_sf.csv”, create a pair RDD of (zip, (store name, city))

- Count pairs which do not have a key.
- Filter pairs that do not include “San Francisco” in the city value.

Contents

Tuning Spark – Persist in Memory/Disk

RDDs are by default recomputed each time.

However, if you want to reuse an RDD for multiple actions, you can ask Spark to store the content in memory/disk and query repeatedly.

```
line_with_spark.persist(StorageLevel.persistency_level)
```

https://spark.apache.org/docs/latest/tuning.html

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

Tuning Spark – Persist in Memory/Disk

Persistency Level

Storage Level	Meaning
MEMORY_ONLY	Store RDDs in memory, If an RDD does not fit in memory, some partitions will not be cached and will be recomputed on the fly each time they're needed. This is the default level.
MEMORY_AND_DISK	If an RDD does not fit in memory, store the partitions that don't fit on disk, and read them from there when they're needed.
MEMORY_ONLY_SER (Java and Scala)	Store RDDs as *serialized* Java objects (one byte array per partition). This is generally more space-efficient than deserialized objects, especially when using a fast serializer, but more CPU-intensive to read.
MEMORY_AND_DISK_SER (Java and Scala)	Similar to MEMORY_ONLY_SER, but spill partitions that don't fit in memory to disk instead of recomputing them on the fly each time they're needed.
DISK_ONLY	Store RDD partitions only on disk.
MEMORY_ONLY_2, MEMORY_AND_DISK_2, etc.	Same as the levels above, but replicate each partition on two cluster nodes.
OFF_HEAP (experimental)	Similar to MEMORY_ONLY_SER, but store the data in off-heap memory. This requires off-heap memory to be enabled.

Tuning Spark – Persist in Memory/Disk

There are replicated storage options available with each of the storage level.

- Replicated storage levels provide much faster fault recovery than RDD lineage in the event of a task or node failure.

https://spark.apache.org/docs/latest/tuning.html

Tuning Spark – Persist in Memory/Disk

`cache()` is the same as calling `persist(MEMORY_ONLY)`.

- Once persisted, RDDs can be reused multiple times without requiring reevaluation (recalculation).
- If there is not enough memory available to cache the RDD, it will be reevaluated for each lineage triggered by an action.

`unpersist()` lets you manually remove them from the cache.

https://spark.apache.org/docs/latest/tuning.html

Tuning Spark – Persist in Memory/Disk

`persist()`

- Many options (See Table) - memory/disk and replication.
  - When using disk options, the persisted data on disk can be used to reconstitute partitions lost due to executor or memory failure.

`unpersist()` lets you unpersist the RDD.

https://spark.apache.org/docs/latest/tuning.html

Example 6

Load a big file and perform an action.

```
#without persisting
start = time.time()
lines.count()
print "first ",
print time.time() - start

start = time.time()
lines.count()
print "second ",
print time.time() - start
```

```
first  59.7045681477
second  56.7281107903
```

```
#with memory and disk
lines.persist(StorageLevel.MEMORY_AND_DISK)
start = time.time()
lines.count()
print "first ",
print time.time() - start #trigger reevaluation

start = time.time()
lines.count()
print "second ",
print time.time() - start #doesn't trigger reevaluation

start = time.time()
lines.count()
print "third ",
print time.time() - start #doesn't trigger reevaluation
```

```
first  79.0769069195
second  20.8628258705
third  20.4466919899
```

➔ Computation triggered by an action.

Tuning Spark – Persist in Memory/Disk

Check the persistency level

- getSotrageLevel() – returns different storage option flags set for an RDD.
  - StorageLevel(useDisk, useMemory, useOffHeap, deserialized, replication = 1)
    - useDisk : If set, partitions that do not fit in memory will be written to disk.
    - useMemory : If set, the RDDs will be stored in-memory.
    - useOffHeap : If set, the RDD will be stored outside of the Spark executor in an external system such as Tachyon.
    - deserialization : If set, the RDD will be stored as deserialized Java objects.
    - replication : An integer that controls the number of copies of the persisted data to be stored.

DATA INSTITUTE
UNIVERSITY OF SAN FRANCISCO

https://spark.apache.org/docs/latest/tuning.html

Tuning Spark – Persist in Memory/Disk

Check the persistency level

- getSotrageLevel() – returns different storage option flags set for an RDD.
  - StorageLevel(useDisk, useMemory, useOffHeap, deserialized, replication = 1)

```
file_name = "../Data/README.md"

lines = sc.textFile(file_name)

lines_with_Spark = lines.filter(lambda x : "Spark" in x)

lines_with_Spark.getStorageLevel()
```

```
StorageLevel(False, False, False, False, 1)
```

https://spark.apache.org/docs/latest/tuning.html

Tuning Spark – Persist in Memory/Disk

Check the persistency level

- getSotrageLevel() – returns different storage option flags set for an RDD.
  - StorageLevel(useDisk, useMemory, useOffHeap, deserialized, replication = 1)

```
lines_with_Spark.cache()
```

```
PythonRDD[2] at RDD at PythonRDD.scala:48
```

```
lines_with_Spark.getStorageLevel()
```

```
StorageLevel(False, True, False, False, 1)
```

https://spark.apache.org/docs/latest/tuning.html

Tuning Spark – Persist in Memory/Disk

Check the persistency level

- getSotrageLevel() – returns different storage option flags set for an RDD.
  - StorageLevel(useDisk, useMemory, useOffHeap, deserialized, replication = 1)

```
lines_with_Spark.persist(StorageLevel(False,True,False,False,3))
```

```
PythonRDD[2] at RDD at PythonRDD.scala:48
```

```
lines_with_Spark.getStorageLevel()
```

```
StorageLevel(False, True, False, False, 3)
```

https://spark.apache.org/docs/latest/tuning.html

Example 7

Try different persistency levels on RDDs which include “Spark”.

Example 7

In “USF_Mission.txt”, find lines with “USF” using lambda and a user defined function.

```
file_name = "../Data/README.md"

lines = sc.textFile(file_name)

lines_with_Spark = lines.filter(lambda x : "Spark" in x)

lines_with_Spark.is_cached

lines_with_Spark.persist()

lines_with_Spark.unpersist()

lines_with_Spark.cache()

lines_with_Spark.is_cached
```

Example 7

Try different persistency levels on RDDs which include “Spark”.

```
from pyspark.storagelevel import StorageLevel
```

```
lines_with_Spark.persist(StorageLevel.DISK_ONLY)
```

```
lines_with_Spark.persist(StorageLevel.MEMORY_AND_DISK_SER )
```

```
lines_with_Spark.persist(StorageLevel.MEMORY_ONLY_2)
```

What happens?
Why?

Tuning Spark – Persist in Memory/Disk

Persisted RDDs in the Spark Application UI.

Spark 2.2.0 | Jobs | Stages | Storage | Environment | Executors | SQL | PySparkShell application UI

Storage

RDDs

RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size on Disk
PythonRDD	Disk Serialized 1x Replicated	1	50%	0.0 B	812.0 B

When does this appear?

Tuning Spark – Persist in Memory/Disk

- If your RDDs fit comfortably with MEMORY_ONLY, leave them that way. This is the most CPU-efficient option, allowing operations on the RDDs to run as fast as possible. Iterative algorithms are often good candidates for caching.
- Don't spill to disk unless the functions that computed your datasets are expensive, or they use a large amount of the data. Otherwise, recomputing a partition may be as fast as reading it from disk.
- Use the replicated storage levels if you want fast fault recovery (e.g. if using Spark to serve requests from a web application). All the storage levels provide full fault tolerance by recomputing lost data, but the replicated ones let you continue running tasks on the RDD without waiting to recompute a lost partition.

Contents

Tips - Passing functions

Passing functions to Spark.

- Most of Spark's transformations and actions depend on passing in functions that are used by Spark to compute data.

1. Pass in lambda functions : for shorter functions.
2. Pass in top-level functions or locally defined functions.

Example 8

In “USF_Mission.txt”, find lines with “USF” using lambda and a user defined function.

Example 8

In “USF_Mission.txt”, find lines with “USF” using lambda and a user defined function.

```
file_name = "../Data/USF_Mission.txt"
input_RDD = sc.textFile(file_name)
```

```
USF_RDD_1 = input_RDD.filter(lambda x : "USF" in x)
```

```
def containsRDD (x):
    return "USF" in x
```

```
USF_RDD_2 = input_RDD.filter(containsRDD)
```

```
USF_RDD_1.collect() == USF_RDD_2.collect()
```

References

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

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

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.