Distributed Computing

DIANE WOODBRIDGE, PH.D

Distributed Computing

Course Objectives

Understand needs and concepts of distributed computing.

Understand the Spark and its stack.

Being competent to work with Spark on a distributed computing environment.

- Programming with RDDs.
- Work with key/value pairs.
- Work with DataFrame.
- Work with SparkSQL, MLlib, ML, Spark Streaming, (and GarphX).
- Work on Amazon AWS.

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 analyse 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

Pair RDDs

Definition

Creation

Transformation on Pair RDDs

Actions on Pair RDDs

Pair RDDs

Definition

- Key-value pairs. – commonly used for many operations including aggregations, ETL (extract, transform, and load) in Spark.
- Allow operations on each key in parallel or regroup data across the network such as reduceByKey(), join(), etc.

Pair RDDs

Creation

- Map Function with a lambda or user function to have a pair (a,b).
- Load data from
  - Files - Ex. text files, JSON, CSV, TSV, etc.
  - Filesystems - Ex. Local, Amazon S3, HDFS, etc.
  - Structured Data – Ex. Apache Hive, JSON, etc.
  - Databases – Ex. JDBC, Cassandra, Hbase, Elasticsearch, etc.

Example 1

From the “README.md” file,

- Extract all the words. (space separated)
- Generate key-value pairs of (Word, 1).

Example 1

From the “README.md” file,

- Extract all the words. (space separated)
- Generate key-value pairs of (Word, 1).

```
rdd = sc.textFile("README.md")
```

```
word = rdd.flatMap(lambda x: x.split(" "))
```

```
word_map = word.map(lambda x : (x,1))
```

```
word_map.collect()
```

```
[(u'#', 1),
 (u'Apache', 1),
 (u'Spark', 1),
 (u'', 1),
 (u'Spark', 1),
 (u'is', 1),
 (u'a', 1),
 (u'fast', 1),
 (u'and', 1),
 (u'general', 1),
 (u'cluster', 1),
```

Pair RDDs - Operations

- Transformation on one Pair RDD

Function name	Purpose
reduceByKey(func)	Combine values with the same key.
groupByKey()	Group values with the same key.
combineByKey(createCombiner, mergeValue, mergeCombiners)	Combine values with the same key using a different result type.
mapValues(func)	Apply a function to each value of a pair RDD without changing the key.
flatMapValues(func)	Apply a function that returns an iterator to each value of a pair RDD, and for each element returned, produce a key/value entry with the old key. Often used for tokenization.
keys()	Return an RDD of just the keys.
values()	Return an RDD of just the values.
sortByKey()	Return an RDD sorted by the key.

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

Example 2

For num_pairs = sc.parallelize({(2,3),(1,2),(1,3),(2,4)})

- What are the results of …
  1. reduced_num_pairs = num_pairs.reduceByKey(lambda x,y : x+y)
  2. grouped_num_pairs = num_pairs.groupByKey()
  3. num_pairs.mapValues(lambda x : x+1)
  4. num_pairs.flatMapValues(lambda x : range(x,5))
  5. num_pairs.keys()
  6. num_pairs.values()
  7. num_pairs.sortByKey()
  8. sum_count = num_pairs.combineByKey((lambda x:(x,1)),(lambda x,y: (x[0]+y, x[1]+1)), (lambda x,y:(x[0]+y[0], x[1]+y[1])))

Example 2

reduceByKey()

- Similar to reduce().
- Run several parallel reduce operations, one for each key in the data set.
- Transformation (Not an action) : Returns a new RDD consisting of each key and the reduced value for that key.
- foldByKey() is also similar to fold().

```
num_pairs = sc.parallelize({(2,3),(1,2),(1,3),(2,4)})
```

```
reduced_num_pairs = num_pairs.reduceByKey(lambda x,y : x+y)
```

```
reduced_num_pairs.collect()
```

```
[(1, 5), (2, 7)]
```

Example 2

groupByKey()

- Group data using the key.
- Return an RDD of (Key, Resulterable) pairs.

```
grouped_num_pairs = num_pairs.groupByKey()
```

```
grouped_num_pairs.map(lambda x : (x[0], list(x[1]))).collect() #iterate over the results
```

```
[(1, [2, 3]), (2, [3, 4])]
```

Example 2

combineByKey(createCombiner, mergeValue, mergeCombiners)

- Similar to aggregate().
  - Allow to return values that are not the same type as the input data.
  - Go through the elements in a partition and merge data in partitions.

Sum and occurrence of the each key.
Output : (Key, (Sum, Occurrence))

```
sum_count = num_pairs.combineByKey((lambda x:(x,1)),(lambda x,y: (x[0]+y, x[1]+1)),(lambda x,y:(x[0]+y[0], x[1]+y[1])))
```

```
sum_count.collect()
```

```
[(1, (5, 2)), (2, (7, 2))]
```

Example 2

TODO:

Using the output of the comineByKey() example, calculate a pair of a number and its average value.

Example 2

```
num_pairs.mapValues(lambda x : x+1).collect()
```

```
[(1, 3), (1, 4), (2, 4), (2, 5)]
```

```
num_pairs.flatMapValues(lambda x : range(x,5)).collect()
```

```
[(1, 2), (1, 3), (1, 4), (1, 3), (1, 4), (2, 3), (2, 4), (2, 4)]
```

```
num_pairs.keys().collect()
```

```
[1, 1, 2, 2]
```

```
num_pairs.values().collect()
```

```
[2, 3, 3, 4]
```

```
num_pairs.sortByKey().collect()
```

```
[(1, 2), (1, 3), (2, 3), (2, 4)]
```

Example - WordCount

From the “README.md” file,

- Extract all the words. (space separated)
- Generate key-value pairs of (Word, Count).
- Sort the pairs by Count (Descending/Ascending Order).

Example - WordCount

```
file_name = "/Users/dwoodbridge/Class/MSAN694/Example_Data/README.md"
```

```
lines = sc.textFile(file_name)
```

```
word = lines.flatMap(lambda line : line.split())
```

```
word_map = word.map(lambda word : (word,1))
```

What is next?

Example 3

Load “filtered_registered_business_sf.csv”.

Generate Pair RDDs with key = zipcode, value = entire line using map().

What is the size of the RDDs?

Print all the pairs.

Example 3

```
lines = sc.textFile("filtered_registered_business_sf.csv")
```

```
pairs = lines.map(lambda x: (x.split(",")[0],x))
```

```
len(pairs.collect())
```

```
198566
```

```
for line in pairs.collect():
    print line
```

```
(u'94123', u'94123,Tournahu George L,3301 Broderick St,San Francisco,CA', 1)
(u'94124', u'94124,Stephens Institute Inc,2225 Jerrold Ave,San Francisco,CA', 1)
(u'94105', u'94105,Stephens Institute Inc,180 New Montgomery St,San Francisco,CA', 1)
(u'94108', u'94108,Stephens Institute Inc,540 Powell St,San Francisco,CA', 1)
(u'94107', u'94107,Stephens Institute Inc,460 Townsend St,San Francisco,CA', 1)
(u'94109', u'94109,Stephens Institute Inc,1835-49 Van Ness Ave,San Francisco,CA', 1)
(u'94102', u'94102,Stephens Institute Inc,620 Sutter St,San Francisco,CA', 1)
(u'94102', u'94102,Stephens Institute Inc,655 Sutter St,San Francisco,CA', 1)
(u'94109', u'94109,Stephens Institute Inc,1055 Pine St,San Francisco,CA', 1)
(u'94107', u'94107,Stephens Institute Inc,121 Wisconsin St,San Francisco,CA', 1)
(u'94102', u'94102,Stephens Institute Inc,150 Hayes St,San Francisco,CA', 1)
(u'94133', u'94133,Stephens Institute Inc,2300 Stockton St,San Francisco,CA', 1)
(u'94133', u'94133,Stephens Institute Inc,2801 Leavenworth St,San Francisco,CA', 1)
```

Example 4

From “filtered_registered_business_sf.csv”, group the values by key and print all the associated values.

Example 4

```
lines = sc.textFile("filtered_registered_business_sf.csv")
sf_business = lines.map(lambda x: (x.split(",")[0],x))
```

```
#for key in sf_business.keys().collect():
    #print key
```

```
#groupByKey
sf_business_group = sf_business.groupByKey()
for business in sf_business_group.collect():
    print "---------"
    print business[0]
    for business_value in business[1]:
       print business_value
```

```
---------

,Hartmann Studios Incorporated,100 W Ohio Ave,,
,Cardno Entrix,5252 Westchester St 250,,
,Bond Blacktop Inc,27607 Industrial Blvd,,
,Moonka Nishi,105 Galisteo Ct,,
,Cooper Jim B,278 1st Ave,,
,Ascencion Flores Ismael O,10412 Arcata Ct,,
,Opower Inc,1515 N Courthouse Rd 610,,
,Htut Chris,218 Warwick St,,
,Red Oxygen Inc,445 Upper Edward St 39,,
,Miniclip America Inc,3rd Floor Diamond House 36-38 Hatton Garden,"London,+uk",
,Vip Plumbing And Drain Cleanin,4020 Payne Rd,,
,Intelex Technologies Inc,905 King St 600,"Toronto,+canada",
,East & West Alum Craft Inc,7465 Conway Ave,Burnaby,
,Act Fuels Inc,Herengracht 469,"Amsterdam,+nl",
```

Example 5

From “filtered_registered_business_sf.csv” in Example 4, group and print all the associated values sorted by keys.

Example 5

```
: lines = sc.textFile("filtered_registered_business_sf.csv")
  sf_business = lines.map(lambda x: (x.split(",")[0],x))
```

```
: #groupByKey
  sf_business_group = sf_business.groupByKey()
  #print values sorted by keys
  for business in sf_business_group.sortByKey().collect():
      print "---------"
      print business[0]
      for business_value in business[1]:
         print business_value
```

```
---------

,Hartmann Studios Incorporated,100 W Ohio Ave,,
,Cardno Entrix,5252 Westchester St 250,,
,Bond Blacktop Inc,27607 Industrial Blvd,,
,Moonka Nishi,105 Galisteo Ct,,
,Cooper Jim B,278 1st Ave,,
,Ascencion Flores Ismael O,10412 Arcata Ct,,
,Opower Inc,1515 N Courthouse Rd 610,,
,Htut Chris,218 Warwick St,,
,Red Oxygen Inc,445 Upper Edward St 39,,
,Miniclip America Inc,3rd Floor Diamond House 36-38 Hatton Garden,"London,+uk",
,Vip Plumbing And Drain Cleanin,4020 Payne Rd,,
,Intelex Technologies Inc,905 King St 600,"Toronto,+canada",
```

Example 6

From “filtered_registered_business_sf.csv”,

- Using reduceByKey(), generate pair RDDs with zip code and frequency.
- Using combineByKey(), generate pair RDDs with zip code and frequency.
  - combineByKey(createCombiner, mergeValue, mergeCombiners)
    - Similar to aggregate operation on RDDs, but works on pairRDDs.

http://spark.apache.org/docs/latest/api/python/pyspark.html
http://abshinn.github.io/python/apache-spark/2014/10/11/using-combinebykey-in-apache-spark/

Example 6

```
lines = sc.textFile("filtered_registered_business_sf.csv")
```

```
sf_business = lines.map(lambda x: (x.split(",")[0],1))
```

```
#reduceByKey
sf_business_group = sf_business.reduceByKey(lambda x , y : x+y)
```

```
#foldByKey
from operator import add
sf_business_group = sf_business.foldByKey(0, (lambda x, y: x+y))
```

```
#combineByKey
sf_business = lines.map(lambda x: (x.split(",")[0], None))
sf_business_group = sf_business.combineByKey((lambda x : 1), (lambda x,y : x+1), (lambda x,y : x+y)  )
```

```
print sf_business_group.sortByKey().collect()
```

```
[(u'', 92), (u'0', 377), (u'10001', 24), (u'10002', 1), (u'10003', 12), (u'10004', 10), (u'10005', 6), (u'10006', 6),
 (u'10007', 3), (u'10010', 15), (u'10011', 10), (u'10012', 10), (u'10013', 14), (u'10014', 7), (u'10016', 9), (u'1001
7', 24), (u'10018', 18), (u'10019', 14), (u'10020', 7), (u'10021', 1), (u'10022', 41), (u'10023', 74), (u'10024', 1),
 (u'10025', 1), (u'10036', 18), (u'10037', 1), (u'10038', 9), (u'10040', 1), (u'10041', 1), (u'10055', 2), (u'10065',
 1), (u'10081', 1), (u'10104', 2), (u'10110', 1), (u'10111', 1), (u'10112', 1), (u'10118', 1), (u'10119', 1), (u'1012
8', 1), (u'10153', 1), (u'10154', 4), (u'10158', 1), (u'10170', 3), (u'10174', 1), (u'10175', 1), (u'10176', 1), (u'1
0178', 1), (u'10271', 1), (u'10272', 1), (u'10279', 1), (u'10281', 2), (u'10282', 3), (u'10285', 1), (u'10314', 2), (
u'10454', 1), (u'10471', 1), (u'10502', 1), (u'10504', 5), (u'10510', 1), (u'10516', 1), (u'10522', 2), (u'10524', 1)
, (u'10528', 1), (u'10549', 1), (u'10567', 1), (u'10573', 1), (u'10577', 8), (u'10580', 1), (u'10583', 1), (u'10591',
 4), (u'10595', 1), (u'10601', 1), (u'10603', 2), (u'10604', 1), (u'10705', 1), (u'10801', 2), (u'10960', 1), (u'1096
```

Pair RDDs - Operations

- Transformation 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

Example 7

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

Example 7

```
lines = sc.textFile("filtered_registered_business_sf.csv")
sf_business = lines.map(lambda x: (x.split(",")[0],x))

lines = sc.textFile("supervisor_sf.csv") #include supervisor id
sf_supervisor = lines.map(lambda x: (x.split(",")[0],x))
```

```
sf_supervisor_business = sf_supervisor.cogroup(sf_business)
```

```
for element in sf_supervisor_business.collect():
    for grouped_element in element [1]:
        print str(element[0]) + "::"
        print list(grouped_element)
```

Example 8

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 8

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

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

```
first_num_pairs.subtract(second_num_pairs).collect()
```

```
[(2, 3), (3, 6), (1, 2), (2, 4)]
```

```
first_num_pairs.subtractByKey(second_num_pairs).collect()
```

```
[(3, 6)]
```

Example 8

```
first_num_pairs.join(second_num_pairs).collect()
```

```
[(1, (2, 3)), (1, (3, 3)), (2, (3, 2)), (2, (4, 2))]
```

```
first_num_pairs.rightOuterJoin(second_num_pairs).collect()
```

```
[(1, (2, 3)), (1, (3, 3)), (2, (3, 2)), (2, (4, 2))]
```

```
first_num_pairs.leftOuterJoin(second_num_pairs).collect()
```

```
[(1, (2, 3)), (1, (3, 3)), (2, (3, 2)), (2, (4, 2)), (3, (6, None))]
```

Example 8

cogroup()

- 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.

```
cogroup = first_num_pairs.cogroup(second_num_pairs)
```

```
cogroup.map(lambda x : (x[0], (list(x[1][0]), list(x[1][1])))).collect() #iterate over the results
```

```
[(1, ([2, 3], [3])), (2, ([3, 4], [2])), (3, ([6], []))]
```

Example 9

From “filtered_registered_business_sf.csv” and “supervisor_sf.csv”,

- Generate pair RDDs where key is a zip code.
- Apply inner join operation on business and supervisor information.
- Apply left and right outer join operations on business and supervisor information.

Example 9

```
lines = sc.textFile("filtered_registered_business_sf.csv")
sf_business = lines.map(lambda x: (x.split(",")[0],x))

lines = sc.textFile("supervisor_sf.csv") #include supervisor id
sf_supervisor = lines.map(lambda x: (x.split(",")[0],x))
```

```
sf_business_supervisor = sf_business.join(sf_supervisor)
```

```
sf_business_supervisor = sf_business.leftOuterJoin(sf_supervisor)
```

```
sf_business_supervisor = sf_business.rightOuterJoin(sf_supervisor)
```

Pair RDDs

Additional Functions.

- 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.
collectAsMap()	Collect the result as a map (**data structure**!) to provide easy lookup.
lookup(key)	Return all values associated with the provided key.

Example 10

Using filter : Filter the pair using the defined function. Ex. Lambda function

From the example 4 (“filtered_registered_business_sf.csv”),

- Count by key.
- Compare the output with collect() and collectAsMap().
- Find pairs without a key.
- Filter pairs that do not include “San Francisco” in the value. (Count and print them.)

Example 10

```
lines = sc.textFile("filtered_registered_business_sf.csv")
```

```
pairs = lines.map(lambda x: (x.split(",")[0],x))
```

```
pairs.countByKey()
```

```
pairs.collect()
```

```
pairs.collectAsMap()
```

```
pairs.lookup('')
```

```
len(pairs.lookup(''))
```

```
pairs.filter(lambda x: "San Francisco" not in x[1]).collect()
```

Tuning the level of Parallelism on Distributed Systems

Partitioning
- For parallel collections, users can specify the number of partitions to cut the RDD into.
- Definition
  - your_rdd = sc.parallelize(data, **partitionSize**)
  - sc.parallelize(data).transformation(…,**partitionSize**)
  - **Check** :
    - your_rdd.getNumPartitions() : getNumPartitions() -Returns the number of partitions in RDD
    - your_rdd.glom().collect() : glom() – return an RDD created by coalescing all elements within each partition into a list.

Problem
- Sending data back and forth between executors on parallelized distributed system causes network traffic.

Solution
- Partitioner
  - Defines how the elements in **a key-value pair RDD** are partitioned by key.
    - Manage data commonly accessible together on the same node.
    - Maps each key to a partition ID, from 0 to numPartitions - 1.
  - Types : HashPartitioner, RangePartitioner, Custom Partitioner.

https://spark.apache.org/docs/1.4.0/api/java/org/apache/spark/Partitioner.html

Example 11

Define data between 1 and 100 and load them into random number of partitions (between 1 and 5).
Check the number of partitions.

Example 11

With data = [('a',3),('b',4),('a',5)],

- Parallelize data into 3 partitioners.
- Perform reduceByKey(lambda x,y : x+y) and check the number of partitioner.
- Perform reduceByKey(lambda x,y : x+y, **NUM**) and check the number of partitioner.

Example 11

```
data = [('a',3),('b',4),('a',5)]
```

```
rdd = sc.parallelize(data,3)
```

```
rdd.collect()
```

```
[('a', 3), ('b', 4), ('a', 5)]
```

```
rdd.reduceByKey(lambda x,y : x+y).getNumPartitions()
```

```
3
```

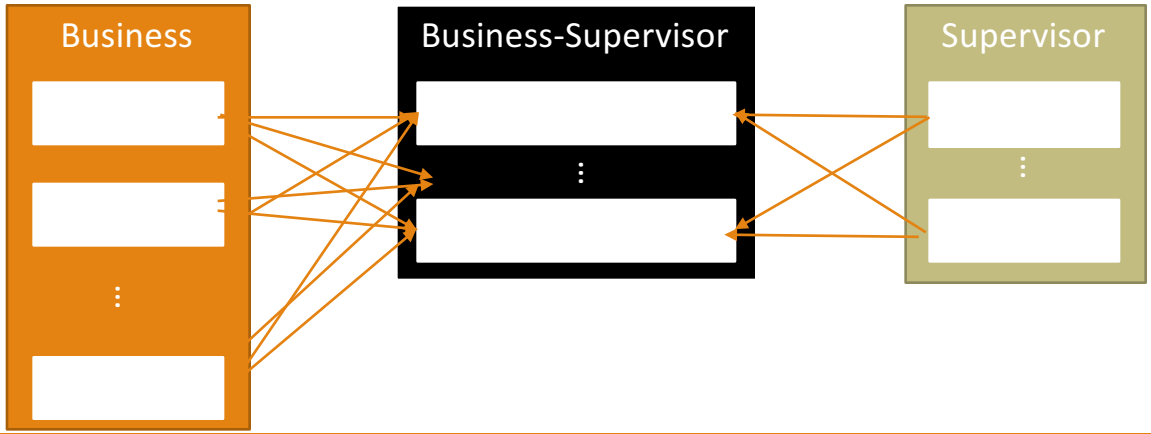
```
rdd.reduceByKey(lambda x,y : x+y,2).getNumPartitions()
```

```
2
```

Tuning the Level of Parallelism on Distributed Systems

Data Partitioning

- Organize data for minimizing network traffic/communication to improve performance.
- Useful when a dataset is reused multiple times in key-oriented operations such as joins.
- Partitioning is available on pair RDDs to group element based on each key's hash value (default : hash).
  - Ensure a set of keys appears together on some node.



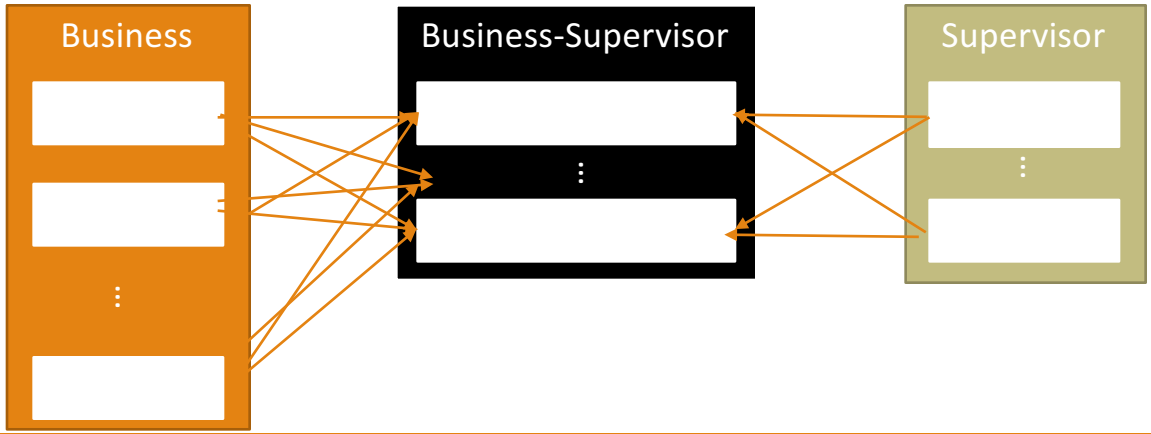
Tuning the Level of Parallelism on Distributed Systems

```
lines = sc.textFile("filtered_registered_business_sf.csv")
sf_business = lines.map(lambda x: (x.split(",")[0],x)).persist()
```

Joining is going to be inefficient

It not know anything about how the keys are partitioned in the datasets.

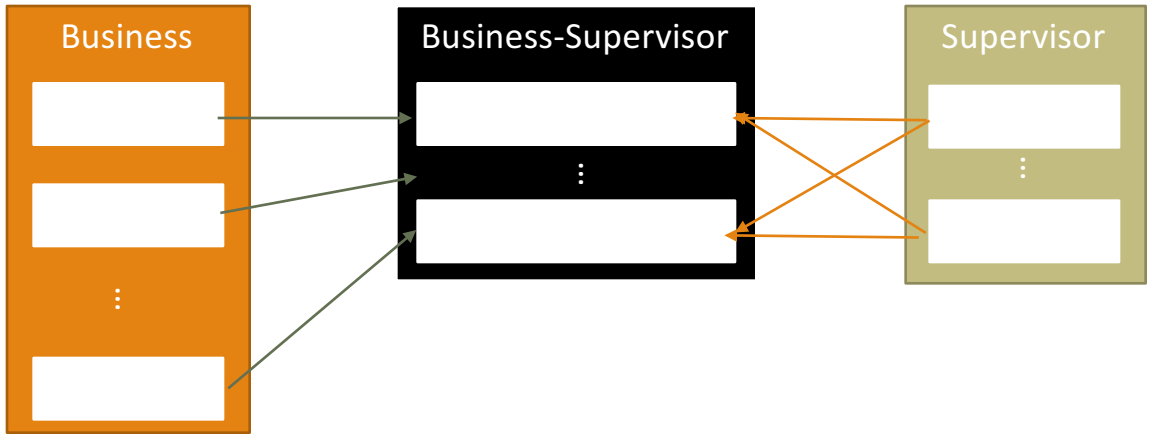
Hash all the keys, sending elements with the same key across the network (shuffling) and then join elements with the same key.



Tuning the Level of Parallelism on Distributed Systems

Data Partitioning and **Reducing Data Shuffling**

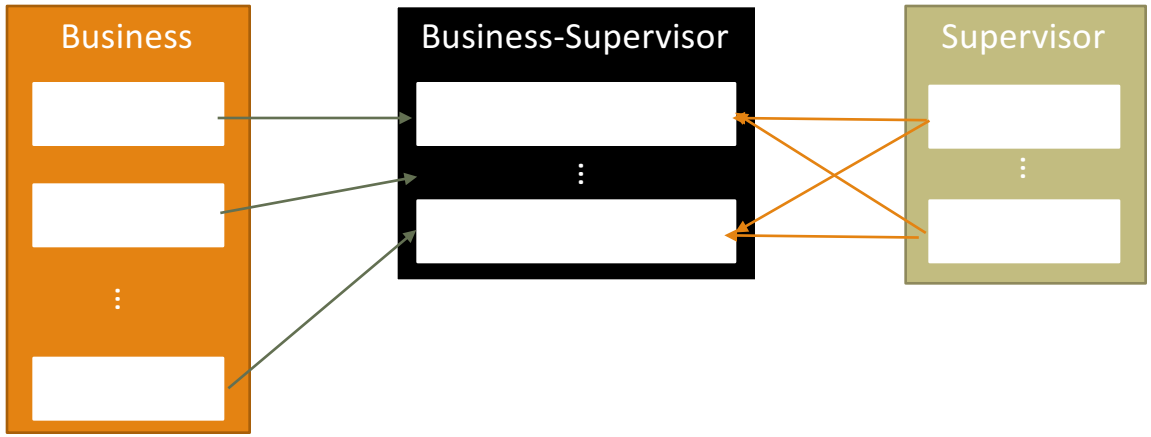
- By using the partitionBy() transformation, Spark minimizes shuffling data as it knows where the data is.
  - partitionBy() uses HashPartitioner by default.



http://spark.apache.org/docs/1.3.1/api/java/index.html?org/apache/spark/HashPartitioner.html

Example 12

```
lines = sc.textFile("filtered_registered_business_sf.csv")
partition_size = 5
sf_business = lines.map(lambda x:
(x.split(",")[0],x)).partitionBy(partition_size).persist()
sf_business.join(supervisor)
```



http://spark.apache.org/docs/1.3.1/api/java/index.html?org/apache/spark/HashPartitioner.html

Example 12

```
lines = sc.textFile("filtered_registered_business_sf.csv")
partition_size = 5
sf_business = lines.map(lambda x:
(x.split(",")[0],x)).partitionBy(partition_size).persist()
sf_business.join(supervisor)
```

```
lines = sc.textFile("filtered_registered_business_sf.csv")
sf_business = lines.map(lambda x: (x.split(",")[0],x)).persist()
```

```
sf_business.glom().first()
```

```
lines = sc.textFile("filtered_registered_business_sf.csv")
partition_size = 5
sf_business = lines.map(lambda x: (x.split(",")[0],x)).partitionBy(partition_size).persist()
```

```
sf_business.glom().first()
```

http://spark.apache.org/docs/1.3.1/api/java/index.html?org/apache/spark/HashPartitioner.html

Example 12

Data Partitioning

- Improve the previous example..
- Use partitionBy() transformation on sf_business to hash-partition it with a user-defined partition size.

```
lines = sc.textFile("filtered_registered_business_sf.csv")
partition_size = 5
sf_business = lines.map(lambda x: (x.split(",")[0],x)).partitionBy(partition_size).persist()
```

partitionBy() : Spark know that it is hash-partitioned.
persist() : Without persisting an RDD after partitioning, it will cause subsequent uses of the RDD to repeat the partitioning of the data.
➔Much Faster : Only shuffle supervisor data, sending supervisor data with each particular key to the machine that has the corresponding key of business.

Example 13

- From "filtered_registered_business_sf.csv", generate key-value pair RDDs with different partition sizes, see the key values on each partition using glom().
  - glom(): Return an RDD created by coalescing all elements within each partition into a list.
- Do same keys locate in the same partition?

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

Example 13

```
lines = sc.textFile("filtered_registered_business_sf.csv")
partition_size = 10
sf_business = lines.map(lambda x: (x.split(",")[0],x)).partitionBy(partition_size).persist()
sf_business.glom().collect()
#if partitionFunc's default is None, not "None" is hash.
#https://spark.apache.org/docs/1.1.1/api/python/pyspark.rdd.RDD-class.html
```

partitionBy(self, numPartitions, partitionFunc=portable_hash)

Return a copy of the RDD partitioned using the specified partitioner.

```
>>> pairs = sc.parallelize([1, 2, 3, 4, 2, 4, 1]).map(lambda x: (x, x))
>>> sets = pairs.partitionBy(2).glom().collect()
>>> set(sets[0]).intersection(set(sets[1]))
set([])
```

Tuning the Level of Parallelism on Distributed Systems

Operations benefiting from partitioning

- Operations involving shuffling data by key across the network.
  - Ex.cogroup(). groupWith(), join(), leftOuterJoin(), rightOuterJoin(), groupByKey(), reduceByKey(), combineByKey(), lookUp()

Tuning the Level of Parallelism on Distributed Systems

Repartitioning RDDs.

- In case, we want to change the partitioning of an RDD outside of the context of grouping and aggregation operations.
- repartition()
  - Shuffle data across the network to create a new set of partitions.
  - Expensive. ☹
- coalesce()
  - Optimized version of repartition() – avoid data movement and reduce the number of RDD partitions.

```
rdd_2 = rdd.reduceByKey(lambda x,y : x+y)
rdd_2.glom().collect()
```

```
[[], [('b', 4)], [('a', 8)]]
```

```
rdd_2.repartition(2).glom().collect()
```

```
[[('b', 4), ('a', 8)], []]
```

```
rdd_2.coalesce(2).glom().collect()
```

```
[[('b', 4)], [('a', 8)]]
```

Example - PageRank

Determine a measure of importance (a "rank") to each document based on how many documents have links to it.

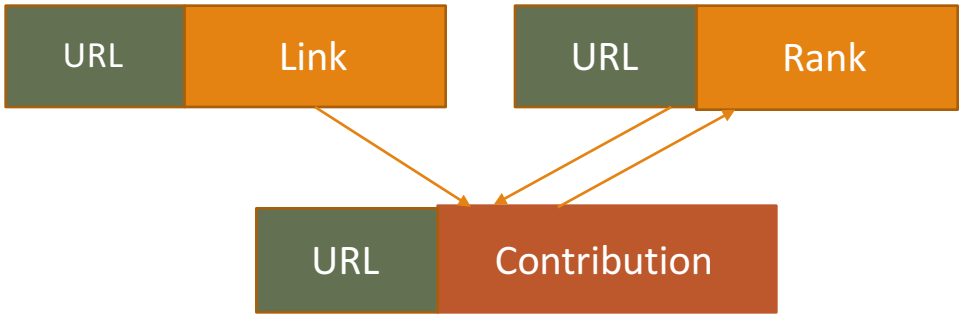
Applications : Rank web pages, influential users in a social network, etc.

Also could be implemented using GraphX.

Algorithm
Read each page and its links information.

Initialize each page's rank to 1.

Iterate :
contribution = rank/numNeighbors
rank = 0.15 + 0.18 * contribution.



https://github.com/apache/spark/blob/master/examples/src/main/python/pagerank.py
https://www.safaribooksonline.com/library/view/learning-spark/9781449359034/ch04.html#idp10672624

Example - PageRank

```
# Loads in input file. It should be in format of:
#     URL         neighbor URL
#     URL         neighbor URL
#     URL         neighbor URL
#     ...
lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])

# Loads all URLs from input file and initialize their neighbors.
links = lines.map(lambda urls: parseNeighbors(urls)).distinct().groupByKey().cache()

# Loads all URLs with other URL(s) link to from input file and initialize ranks of them to one.
ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0))

# Calculates and updates URL ranks continuously using PageRank algorithm.
for iteration in range(int(sys.argv[2])):
    # Calculates URL contributions to the rank of other URLs.
    contribs = links.join(ranks).flatMap(
        lambda url_urls_rank: computeContribs(url_urls_rank[1][0], url_urls_rank[1][1]))

    # Re-calculates URL ranks based on neighbor contributions.
    ranks = contribs.reduceByKey(add).mapValues(lambda rank: rank * 0.85 + 0.15)

# Collects all URL ranks and dump them to console.
for (link, rank) in ranks.collect():
    print("%s has rank: %s." % (link, rank))
```

Consider.. partitionBy().persist()

Algorithm

Read each page and its links information.

Initialize each page's rank to 1.

Iterate :

contribution = rank/numNeighbors

rank = 0.15 + 0.18 * contribution.

Since links is a static dataset, we partition it at the start with partitionBy(), so that it does not need to be shuffled across the network.
We call cache() (or persist()) on links to keep it in RAM across iterations.

https://github.com/apache/spark/blob/master/examples/src/main/python/pagerank.py
https://www.safaribooksonline.com/library/view/learning-spark/9781449359034/ch04.html#idp10672624

Loading and Saving Data

Spark supports unstructured/semi-structured/structured types of data.

File Formats include the followings (and many more).

Format	Structured	Comments
Text File	No	One line is one element for a single file.
JSON	Semi	Most libraries require one record per line.
CSV/TSV	Yes	Contains fixed number of fields per line.
Sequence File	Yes	Hadoop file format used for key-value data.
Object File	Yes	Used for saving data from a Spark job to be consumed by shared code.

Loading and Saving Data

File Formats

- Text Format
  - Load
    - One line is one element in a single file.
      - Use textFile(file_name) method.
    - Can load multiple text files as **pair RDDs**, with the key being the file name and the value being the file content.
      - Use wholeTextFiles(dir _name).
      - Useful when each file represent a certain time period's data and need to use it for statistics. Ex. sales stat.
  - Save
    - Use saveAsTextFile(new_subdir_name) method.
    - Return multiple output files underneath the new_subdir_name, as Spark writes the output from multiple nodes.

Example 14

Load all the .csv files under a directory as Pair RDDs with key being a file name and values being the content of the file.

Example 14

```
lines = sc.wholeTextFiles("/Users/dwoodbridge")
```

```
lines.count()
```

```
7
```

```
for item in lines.take(10):
    print item
```

```
(u'file:/Users/dwoodbridge/get-pip.py', u'#!/usr/bin/env python\n#\
nt blob of binary data here is, you might\n# even be worried that w
being\n# paranoid!). This is a base85 encoding of a zip file, this
Pip is a thing that installs packages, pip itself is a package that
hey\'re looking to run this get-pip.py\n# script. Pip has a lot of
ckages, various edge cases on various platforms, and other such sor
n its code base. Because of this\n# we basically include an entire
e the alternatives are attempt to implement a "minipip" that probab
dge cases, or compress pip itself\n# down into a single file.\n#\n#
```

Example 15

From "supervisor_sf.csv” in your class example directory (not spark dir), filter data of “94103” and save under ”supervisor_94103.”.

Example 15

```
lines = sc.textFile("/Users/dwoodbridge/DistributedComputing/Day2/Example/supervisor_sf.csv")
```

```
filtered_lines = lines.filter(lambda x: "94103," in x) #.repartition(1) generates one file.
```

```
filtered_lines.saveAsTextFile("/Users/dwoodbridge/DistributedComputing/Day2/Example/supervisor_94103")
```

Loading and Saving Data

File Formats

- JSON
  - Use python's json package.
  - Load
    - Use loads() method.
      - One JSON record per row.

```
import json
data = input.map(lamdbda x: json.loads(x))
```

      - If loading multiple files, load the whole file and then parse each file.
    - Use SparkSQL.
  - Save
    - Use json.dumps() and then saveAsTextFile().

Example 16

Load "walmart_search_san_francisco.json”

Filter items that is available online and save as “walmart_search_online_items_sanfrancisco” in JSON format.

Example 16

```
import json
```

```
#Load Data
json_input = sc.textFile("walmart_search_san_francisco.json")
json_data = json_input.map(lambda x:json.loads(x))
```

```
#Filter Data
json_online_items = json_data.filter(lambda x : x['items'][0]['availableOnline'] == False)
```

```
json_dumps = json_online_items.map(lambda x : json.dumps(x))
json_dumps.saveAsTextFile("walmart_search_online_items_san_francisco")
```

Loading and Saving Data

File Formats

- Comma Separated Value (CSV) and Tab separated Value (TSV)
  - Use python's csv package.
  - Load
    - Use textFile()/wholeTextFiles() and parse the csv to load the data.
    - And use StringIO.StringIO() to read a string buffer and csv.DictReader(input, *fieldnames*) to read information into a dict whose keys are given by the optional *fieldnames* parameter.
  - Save
    - Write a function to write each row using stringIO(),DictWriter(), WriteRow(), etc.
    - Save the file using saveAsTextFile().

Example 17

Read data from supervisor_sf_2.csv.

Example 17

```
import csv
import StringIO
```

```
input = sc.textFile("supervisor_sf_2.csv")
```

```
def csvLoader(line):
    print type(line)
    input = StringIO.StringIO(line)
    reader = csv.DictReader(input, fieldnames=["Supervisor","ZIP_Code"])
    return reader.next()
```

```
csv_data = input.map(csvLoader)
```

```
csv_data.collect()
```

```
[{'Supervisor': 'Supervisor_District', 'ZIP_Code': 'ZIP_Code'},
 {'Supervisor': '8', 'ZIP_Code': '94102'},
 {'Supervisor': '6', 'ZIP_Code': '94102'},
 {'Supervisor': '3', 'ZIP_Code': '94102'},
 {'Supervisor': '5', 'ZIP_Code': '94102'},
 {'Supervisor': '8', 'ZIP_Code': '94103'},
```

Group Project – Todo1

Form a group with common interests (practicum, data mining project, etc.)

- 3 or 4 members.
- Choose a data set for a group project that can benefit from Spark. List reasons.
- Determine the size of the data.
- Define data analytics goal for the project and see what kinds of data mining algorithm libraries could be utilized (Check it is in Spark MLlib or ML).

Reference

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.