732A54/TDDE31 Big Data Analytics

Exercise Session

Huanyu Li



Agenda

- Aims of this exercise session
- Review
 - ✓ Map-Reduce: Working with key-value pairs
 - ✓ Lambda functions
- How to design and write PySpark code
- Lab introduction and exercises
 - ✓ Conceptual design
 - ✓ Write PySpark code
- *How to work on heffa



Aims

- Give you an overview of the labs
- Help you to understand how to design and write code using Spark in python
- Exercises: start to solve the assignments in the labs



Map-Reduce: Working with key-value pairs

- Data elements: key-value pairs
- Python's tuple structure fit this key-value pair: (key, value)
 (1,2), (1,3), (1,4), (1,5), (2,2), (2,3)
- > A tuple is a sequence of immutable Python objects

```
('a', 3), (1, (3, 4)), ((1,1), 2)
```

Accessing elements done with [index]

```
x = (3, (c', [1])), y = ((3, a'), (c', [1]))

x[0] = 3, x[1] = x[-1] = (c', [1]), x[1][1] = [1]

y[0] = (3, a'), y[0][0] = 3, y[1][1] = [1]
```

'Shuffle' operations by a key work on RDDs containing built-in Python tuples

- √ 'repartition' operations
- √ 'byKey' operations
- √ 'join' operations



Lambda functions— a way to pass function to a RDD operation

General form

lambda arguments: expression

> Examples:



RDD - Operations

	$map(f:T\Rightarrow U)$:	:	$RDD[T] \Rightarrow RDD[U]$
	$filter(f: T \Rightarrow Bool)$:	:	$RDD[T] \Rightarrow RDD[T]$
	$flatMap(f: T \Rightarrow Seq[U])$:	:	$RDD[T] \Rightarrow RDD[U]$
	<pre>sample(fraction : Float) :</pre>	:	$RDD[T] \Rightarrow RDD[T]$ (Deterministic sampling)
	groupByKey() :	:	$RDD[(K, V)] \Rightarrow RDD[(K, Seq[V])]$
	$reduceByKey(f:(V,V) \Rightarrow V)$:	:	$RDD[(K, V)] \Rightarrow RDD[(K, V)]$
Transformations	union() :	:	$(RDD[T], RDD[T]) \Rightarrow RDD[T]$
	join() :	:	$(RDD[(K, V)], RDD[(K, W)]) \Rightarrow RDD[(K, (V, W))]$
	cogroup() :	•	$(RDD[(K, V)], RDD[(K, W)]) \Rightarrow RDD[(K, (Seq[V], Seq[W]))]$
	crossProduct() :	:	$(RDD[T], RDD[U]) \Rightarrow RDD[(T, U)]$
	$mapValues(f : V \Rightarrow W)$:	•	$RDD[(K, V)] \Rightarrow RDD[(K, W)]$ (Preserves partitioning)
	/		$RDD[(K, V)] \Rightarrow RDD[(K, V)]$
	partitionBy(p : Partitioner[K]):	:	$RDD[(K, V)] \Rightarrow RDD[(K, V)]$
	count() :	R	$RDD[T] \Rightarrow Long$
	collect() :	R	$RDD[T] \Rightarrow Seq[T]$
Actions			$RDD[T] \Rightarrow T$
	- ` ,		$RDD[(K, V)] \Rightarrow Seq[V]$ (On hash/range partitioned RDDs)
	save(path: String):	C	Outputs RDD to a storage system, e.g., HDFS

Table 2: Transformations and actions available on RDDs in Spark. Seq[T] denotes a sequence of elements of type T.

- You need more than the above to solve all assignments in the lab.
 - ✓ PySpark library: https://spark.apache.org/docs/1.6.1/api/python/pyspark.html
 - ✓ Spark 1.6 programming guide: https://spark.apache.org/docs/1.6.0/programming-guide.html



Agenda

- Aims of this exercise session
- Review
 - ✓ Map-Reduce: Working with key-value pairs
 - ✓ Lambda functions
- How to design and write PySpark code
- Lab introduction and exercises
 - ✓ Conceptual design
 - ✓ Write PySpark code
- *How to work on heffa



Word Count – Conceptual Design

- In terms of map-reduce programming model, how to form key, value pair and what kind of transforms are needed.
- During reduce process, what functions are needed on the values.

Mapper

Shuffle Reduce flatmap, (The, 1) The (The, 1) (weather, 1) weather map (weather, 1) (of. 1) (of. 1) (vesterday, 1) yesterday (yesterday, 1) (The, 1) (was, 1) was (was, 1) Data (weather, 1) (The, 2) (warm, 1) add warm (warm, 1) (of, 1) (weather, 1) (yesterday, 1) (of, 1) (was, 1) (yesterday, 1) (warm, 1) (was, 1) (The, 1) (The, 1) (warm, 2) flatmap, (warm, 1) (The, 1) The map max (max. 1) temperature (temperature, 1) The weather of yesterday was warm (max, 1) (temperature. 15 (15, 1)The max temperature is 15 degree degree (degree, 1) (is, 1) Tomorrow is also warm (max, 1) (15, 1)(max, 1) (temperature, 1) (degree, 1) (temperature, 1) (is, 1) (is, 2) (15, 1)(warm, 1) (15, 1)(degree, 1) (degree, 1) (Tomorrow, 1) (Tomorrow, 1) flatmap, (is, 1) (also, 1) map (Tomorrow, 1) (also, 1) Tomorrow (Tomorrow, 1) (is. 1) (also, 1) also (also, 1) (warm, 1) warm



How to write PySpark code

- > Step 1. To create a SparkContext object which tells Spark how to access a cluster.
- Step 2. To create distributed datasets (RDD)
 - ✓ Use external datasets by local file system or HDFS
- > Step 3. RDD operating: transformation, action



- Step 1. To create a SparkContext object which tells Spark how to access a cluster.
- Step 2. To create distributed datasets (RDD)

Mapper

Use external datasets by local file system or HDFS

Step 3. RDD operating: transformation, action Shuffle Reduce The (The, 1) flatmap (The, 1) (weather, 1) weather (weather, 1) (of, 1) (of, 1) (yesterday, 1) yesterday (yesterday, 1) (The. 1) (was, 1) Data (was, 1) (weather, 1) (The, 2) (warm, 1) add warm (warm-1) (of, 1) (weather, 1) (yesterday, 1) (of. 1) (was, 1) (yesterday, 1) (warm, 1) (was, 1) (The, 1) (The, 1) (warm, 2) (warm, 1) (The, 1) The flatmap max (max. 1) (temperature, 1) temperature The weather of yesterday was warm (max, 1) (is, 1) (temperature, 15 (15, 1)The max temperature is 15 degree degree (degree, 1) (is, 1) Tomorrow is also warm (max, 1) (15, 1)(max. 1) (temperature, 1) (degree, 1) (temperature, 1) (is, 1) add (is, 2) (15, 1)(warm, 1) (15, 1)(degree, 1) (degree, 1) (Tomorrow, 1) (Tomorrow, 1) (is, 1) flatmap (also, 1) (Tomorrow, 1) (also, 1) Tomorrow (Tomorrow, 1) (is, 1) (is, 1) (also, 1) also (also, 1) (warm, 1) warm

- from pyspark import SparkContext
- sc = SparkContext(appName = "exercise test")____
- step 2 news_file = sc.textFile("/user/x_huali/data/news.txt") =
- words = news_file.flatMap(lambda line: line.split(" "))_
- word_count = words.map(lambda word: (word, 1)) =
- counts = word_count.reduceByKey(lambda v1, v2: v1+v2)
- counts.saveAsTextFile("word_count_result")_____ step 3: RDD action



step 1

step 3: RDD transformation(s)

Agenda

- Aims of this exercise session
- Review
 - ✓ Map-Reduce: Working with key-value pairs
 - ✓ Lambda functions
- How to design and write PySpark code
- Lab introduction and exercises
 - ✓ Conceptual design
 - ✓ Write PySpark code
- *How to work on heffa



Lab Introduction

- Working with the historical meteorological data from Swedish Meteorological Hydrological Institute (SMHI)
 - ✓ The data includes air temperature and precipitation readings from 812 stations in Sweden.
- > Three labs
 - ✓ BDA1 Spark: 6 assignments
 - In general, you need to do filtering, grouping, aggregating ... over the data.
 - Map-reduce programming model, PySpark,
 - working with key-value pairs
 - ✓ BDA2 Spark SQL: Redo the 6 assignments in BDA1 with Spark SQL
 - ✓ BDA3 Machine Learning with Spark:
- Example: Find highest temperature for a certain period
 - ✓ temperature-readings.csv

Headers for temperature-readings.csv

Station number Date	Time	Air temperature (in °C)	Quality ³
---------------------	------	-------------------------	----------------------

102170;2013-11-01;06:00:00;6.8;G 102170;2013-11-01;18:00:00;3.8;G 102170;2014-11-02;06:00:00;5.8;G 102170;2014-11-02;18:00:00;-1.1;G 102170;2015-11-03;06:00:00;-0.2;G 102170;2015-11-03;18:00:00;5.6;G 102170;2015-11-04;06:00:00;6.5;G

.....



Find the highest temperature in 2014 and 2015.

Show the year and highest temperature in the result

- Conceptual design
 - ✓ Understand the question and data
 - ✓ How to form key, value pair and what RDD operations are needed.
 - ✓ What operations are needed during mapping and reducing?
- Write pyspark code

Headers for temperature-readings.csv

|--|

```
102170;2013-11-01;06:00:00;6.8;G
```

102170;2013-11-01;18:00:00;3.8;G

102170;2014-11-02;06:00:00;5.8;G

102170;2014-11-02;18:00:00;-1.1;G

102170;2015-11-03;06:00:00;-0.2;G

102170;2015-11-03;18:00:00;5.6;G

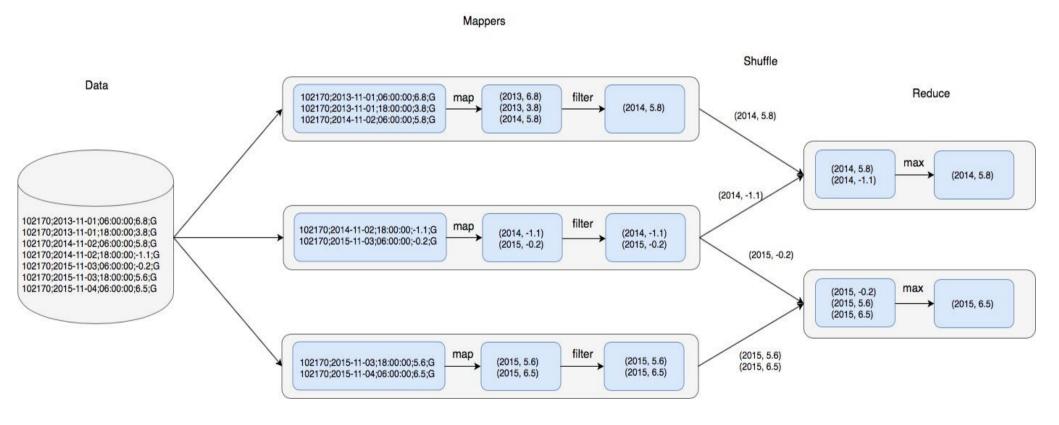
102170;2015-11-04;06:00:00;6.5;G

.....



Solution – Conceptual design

- Extract year as key and temperature as value
- > Filter data (2014 and 2015)
- > Reduce by key and then compare each two values to get the higher temperature.





- Step 1. To create a SparkContext object which tells Spark how to access a cluster.
 - Step 2. To create distributed datasets (RDD) from HDFS.
- Step 3. RDD operating: transformation, action

Solution

Question: Find the highest temperature in 2014 and 2015.

```
from pyspark import SparkContext
  def max_temperature(a,b):
      if a>=b:
3
           return a
      else:
           return b
  sc = SparkContext(appName = "exercise test")
  temperature_file = sc.textFile("/user/x_huali/data/temperature-readings.csv")
  lines = temperature_file.map(lambda line: line.split(";"))
  year_temperature = lines.map(lambda x: (x[1][0:4], float(x[3])))
  year_temperature = year_temperature.filter(lambda x: int(x[0])==2014 or int(x[0])==2015)
  #max_temperatures = year_temperature.reduceByKey(lambda a,b: a if a>=b else b)
  #max_temperatures = year_temperature.reduceByKey(max)
  max_temperatures = year_temperature.reduceByKey(max_temperature)
  max_temperatures.saveAsTextFile("max_temperature_2014_2015")
  line 7: create SparkContext object
  line 8: get the file on hdfs, absolute path needed! '/user/USERNAME/...' or
   'hdfs:namenode:8020/user/USERNAME/...'
  line 9: transform the data by splitting each line
```

- line 10: transform the data by extracting year and temperature as tuple
- line 11: filter data by year
- line 14: reducer, to get the max temperature,
 - line 12, line 13, line 14 show the different ways of passing functions to Spark
- line 15: save result in a directory



For the first assignment in BDA1

- 1) What are the highest temperatures measured each year for the period 1950-2014. Provide the listed sorted in the descending order with respect to the maximum temperature
- Exercise (30 mins)
 - ✓ Conceptual design (how to form key, value pairs and what RDD operations are needed).
 - ✓ Write pyspark code (Pseudocode)

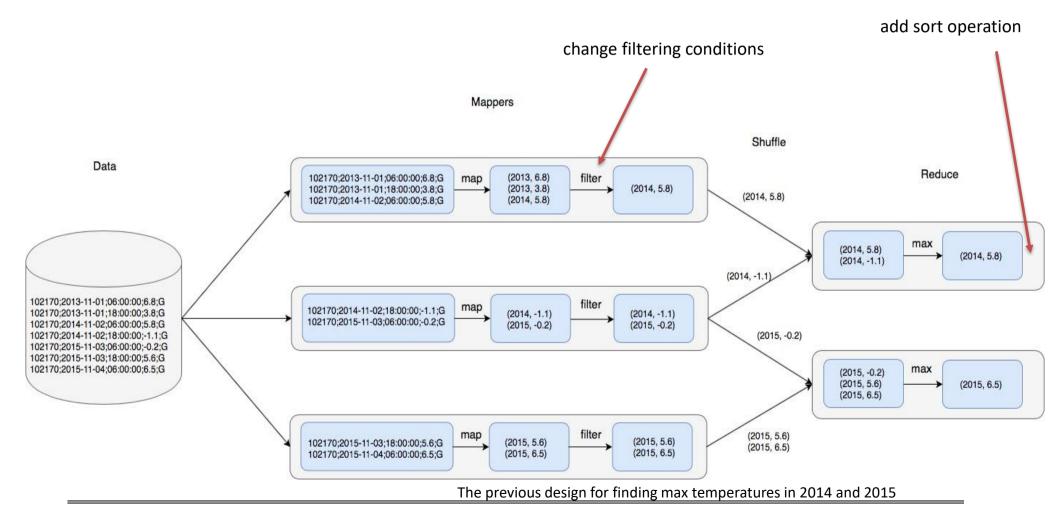
Headers for temperature-readings.csv

Station number	Date	Time	Air temperature (in °C)	Quality ³	
102170;2013-11-01 102170;2013-11-01 102170;2014-11-02 102170;2014-11-02 102170;2015-11-03 102170;2015-11-03	2;18:00:00;3.8;G 2;06:00:00;5.8;G 2;18:00:00;-1.1;G 3;06:00:00;-0.2;G 3;18:00:00;5.6;G		•	te a SparkContext e distributed data	•



Solution – Conceptual design

- Extract year and temperature
- Filter data (2014 and 2015) 1950-2014
- Reduce by key to get maximum
- > Sort





Solution

- Pre steps: Distribute your data
- Step 1. To create a SparkContext object which tells Spark how to access a cluster.
- Step 2. To create distributed datasets (RDD) from HDFS.
- Step 3. RDD operating: transformation, action

```
from pyspark import SparkContext
  def max_temperature(a,b):
      if a >= b:
          return a
      else:
5
           return b
  sc = SparkContext(appName = "exercise test")
  temperature_file = sc.textFile("/user/x_huali/data/temperature-readings.csv")
  lines = temperature file.map(lambda line: line.split(";"))
  year_temperature = lines.map(lambda x: (x[1][0:4], float(x[3])))
  year_temperature = year_temperature.filter(lambda x: int(x[0])>=1950 and int(x[0])<=2014)
  #max_temperatures = year_temperature.reduceByKey(lambda a,b: a if a>=b else b)
  #max_temperatures = year_temperature.reduceByKey(max)
  max_temperatures = year_temperature.reduceByKey(max_temperature)
  max_temperaturesSorted = max_temperatures.sortBy(ascending = False, keyfunc=lambda k: k[1])
  max_temperaturesSorted.saveAsTextFile("max_temperature")
```

- line 7: create SparkContext object
- line 8: get the file on hdfs, absolute path needed! '/user/USERNAME/...' or 'hdfs:namenode:8020/user/USERNAME/...'
- line 9: transform the data by splitting each line
- line 10: transform the data by extracting year and temperature as tuple
- line 11: filter data by a time period
- line 14: reducer, to get the max temperature,
 - line 12, line 13, line 14 show the different ways of passing functions to Spark
- line 15: sort result by temperature
- line 16: save result in a directory



*How to work on heffa

Data paths

- Heffa locally: /nfshome/hadoop_examples/shared_data temperature-readings.csv, precipitation-readings.csv temperatures-big.csv
- Heffa hdfs: /user/common/732A54
 temperatures-big.csv
- https://www.ida.liu.se/~732A54/lab/station_data.zip stations.csv stations-Ostergotland.csv



*How to work on heffa

Commands

- ssh -X
- scp file.txt username@heffa.nsc.liu.se:Desktop
- hdfs dfs -ls /user/username/
- hdfs dfs -mkdir data
- cd /nfshome/hadoop_examples/shared_data
- hdfs dfs -copyFromLocal ./temperature-readings.csv /user/username/data

-



www.liu.se

