Advanced Programming

Lesson 4

Hortonworks

Learning Objectives

- After you complete this lesson you should be able to:
  - Use Pair RDD functions
    - Joins
    - GroupBy
    - Key reordering
  - Use more Core RDD functions
  - Use the Spark documentation to find proper API
  - Understand basic partitions in Spark
  - Explain Job/Stage/Task and creation of the DAG
  - View the Spark UI

Hortonworks

Pair RDD Transformation: mapValues()

- Performs a map only on the values, not changing the keys

```
rdd=sc.parallelize([(“a”, [“dog”, “cat”, “bird”]), (“b”, [“this”])])
rdd.mapValues(lambda x: len(x)).collect()
[(‘a’, 3), (‘b’,1)]
```


Hortonworks

Pair RDD Transformation: groupByKey()

- Returns an rdd with a grouping of values by key

```
rdd = sc.parallelize([('a', 1), ('a', 2), ('b', 1), ('b', 3)])
output = rdd.groupByKey()
for (key, values) in output.collect():
    print ‘key: ‘,key
    for value in values: print ‘\t’, value
[(‘a’, [1, 2]), (‘b’, [1, 3])]
```


Hortonworks

Pair RDDs Transformation: Joins

- All joins supported (inner, full outer, left outer, right outer)

```
rdd = sc.parallelize([(‘a’, 1), (‘a’, 2), (‘b’, 1), (‘b’, 3)])
rdd1 = sc.parallelize([(‘a’, 2), (‘a’, 3), (‘b’, 1), (‘c’, 3)])

rdd1.leftOuterJoin(rdd).collect()
[('a', (2, 1)), ('a', (2, 2)), ('a', (3, 1)), ('a', (3, 2)),
 ('c', (3, None)), ('b', (1, 1)), ('b', (1, 3))]
```


Hortonworks

Pair RDD Transformation: Reordering the K/V

- A lot of times we need to reorder our K/V pair for processing
  – This is called pattern matching

```
rdd = sc.parallelize([('a', (1,2)), ('a', (2,3))])
Python
rdd.map( lambda (k, (v1,v2)) : ((k,v1) ,v2)).collect()
[(('a', 1), 2), (('a', 2), 3)]
Scala
rdd.map{case (k,(v1,v2) => ((k, v1), v2)}.collect()
[(('a', 1), 2), (('a', 2), 3)]
```


Hortonworks

More Pair RDD operations

- Transformations
  - keys() returns just the keys
  - values() returns just the values
  - subtractByKey() retruns rdd minus the keys in rdd1
- Actions
  - lookup(key) returns all the values for that particular key

More RDD Transformations

- keyBy() create a KV pair with the specified key
- zipWithIndex() like zip, but also returns index of element
- union() returns an RDD of both rdds
- zip() returns an RDD of K/V where rdd element is key and rdd1 element is value

Using the Spark API Documentation

- http://spark.apache.org/docs/<version>/api/scala/
- http://spark.apache.org/docs/<version>/api/python/
- Version can be
  - "1.3.1"
  - "1.5.0"
  - "latest" for the newest release

Hortonworks

Lab 2

- In this lab we want to do the following:
  - Find the top 5 states with the most males
  - Get the top 3 wines for each state

- Lab tip*

```
myString = ("This is my string")
rdd=sc.parallelize([myString]).map(lambda line: line.split(" "))
            .map(lambda line: (line[0], line[3])).take(1)
[('This', 'string')]
```


Hortonworks

Basic Spark Partitioning/Parallelism

- White Board
- Notes:
  - Defaults to number of blocks a file is on HDFS
  - Can be set with spark.default.parallelism
    - Default is num of cores on local, or total cores on all executors
    - This setting is for operations with no parent RDD's
  - Shuffle based operations (join, reduceByKey, etc) take the largest number in the parent
  - Shuffle based operations take an optional param for partitions
    - rdd.reduceByKey(lamda a, b: a+b, **20**) will have 20 partitions

Hortonworks

Wide vs Narrow Operations

- Wide operations require a shuffling of data (normally)
  - reduceByKey
  - groupByKey
  - repartition
  - join
- Narrow operations can be executed locally
  - map
  - filter
  - flatMap

Hortonworks

Understanding the Shuffle

- White board how a shuffle works
- Extra Notes/Gotchas
  - Lots of parallelism + lots of reducers = lots of small files
    - Not a big deal, except when you open more then 32k files on a system
    - spark.shuffle.consolidateFiles=False
  - Shuffle the minimal amount of data
    - Avoid groupByKeys
    - Filter/Distinct Early!
  - Embrace the Shuffle
    - Not necessarily bad

Hortonworks

Spark Task/Stage/Job

- Task is a unit of work
- Stage is a group of tasks separated by a wide operation
- A job is a grouping of stages

- The next stage cannot start before all the tasks in the previous stage have finished

Hortonworks

Using the spark ui

- For stand alone spark:
  localhost:4040
- For YARN integrated cluster, navigate with Hue
- UI offers valuable insight into the application
- Demo

Hortonworks

© Hortonworks Inc. 2011 – 2014. All Rights Reserved

Hortonworks

Conclusion and Key Points