Final Project M1

CS 327E
October 30, 2017

Final Project Overview

Goals:

- Learn the basics of distributed computing and Spark (Spark Core and Spark SQL)
- Gain hands-on exposure to EMR*
- Develop ETL pipelines with PySpark and Postgres
- Enrich IMDB database with new data sources

Format:

- Weekly milestones, 6 total
- Continue working in pairs
- Monday: reading quiz, new concepts, assignment sheet, project work
- Wednesday: project work
- Friday: milestone submission (except for Thanksgiving week)

***EMR is not covered by free tier.** Pricing for 1-node cluster on m3.xlarge is $0.34 per hour / $8.2 per day / $246 per month.

Final Project Milestones

- **M1:** ETL movie ratings data (source: Movielens). Due date: 11/03.
- **M2:** ETL movie tags data (source: Movielens). Due date: 11/10.
- **M3:** ETL movie ticket sales data (source: The-Numbers). Due date: 11/17.
- **M4:** ETL Bollywood data (source: Cinemalytics). Due date: 12/01.
- **M5:** Group presentations. Week of 12/4 – 12/11.
- **M6:** Technical reports. Due date: 12/11.

1) The MapReduce programming model consists of a user-provided map function and a user-provided reduce function.

A) True
B) False

2) The fundamental abstraction in Spark is called:

A) Discretized Stream
B) Resilient Distributed Dataset
C) B+ Tree
D) Distributed Hash table

3) What type of operation is the **map** function in Spark?

A) A transformation
B) An action
C) An event
D) All of the above

4) What type of operation is the **reduce** function in Spark?

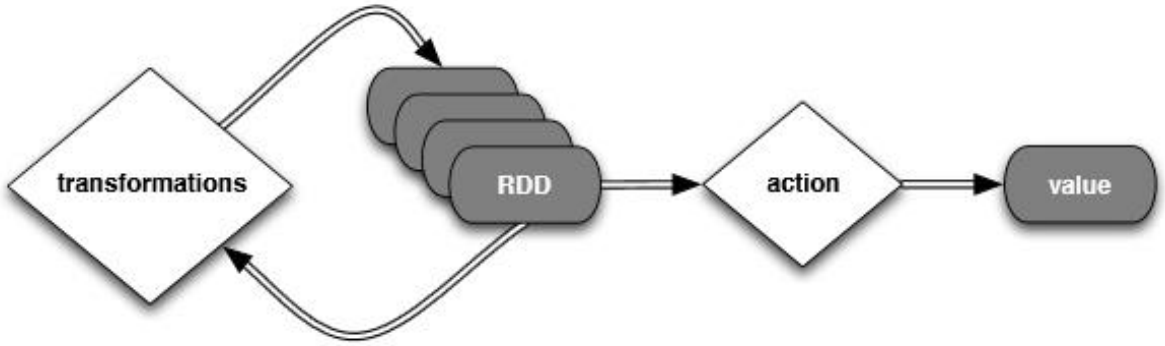
A) A transformation
B) An action
C) A sample
D) All of the above

5) Which of these AWS services provides a Spark cluster?

A) CloudFormation
B) Athena
C) Kinesis
D) Elastic MapReduce

RDD Key Concepts

- *RDD* = Partitioned collection of records across a Spark cluster
- Operations on RDD = transformations and actions
- *Base RDD* created from file(s)
- *Transformed RDD* created by applying transformations and actions to *Base RDD*



Spark Transformations

- **Map:** call map on an RDD and pass it a function as a parameter. Map applies the function to each element of the input RDD. It returns a new RDD as output.

```
>>> numbers = sc.parallelize(range(10))
>>> numbers.map(lambda x: x*10).collect()
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>> def times_ten(x):
...   return x*10
...
>>> numbers.map(times_ten).collect()
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
>>>
```

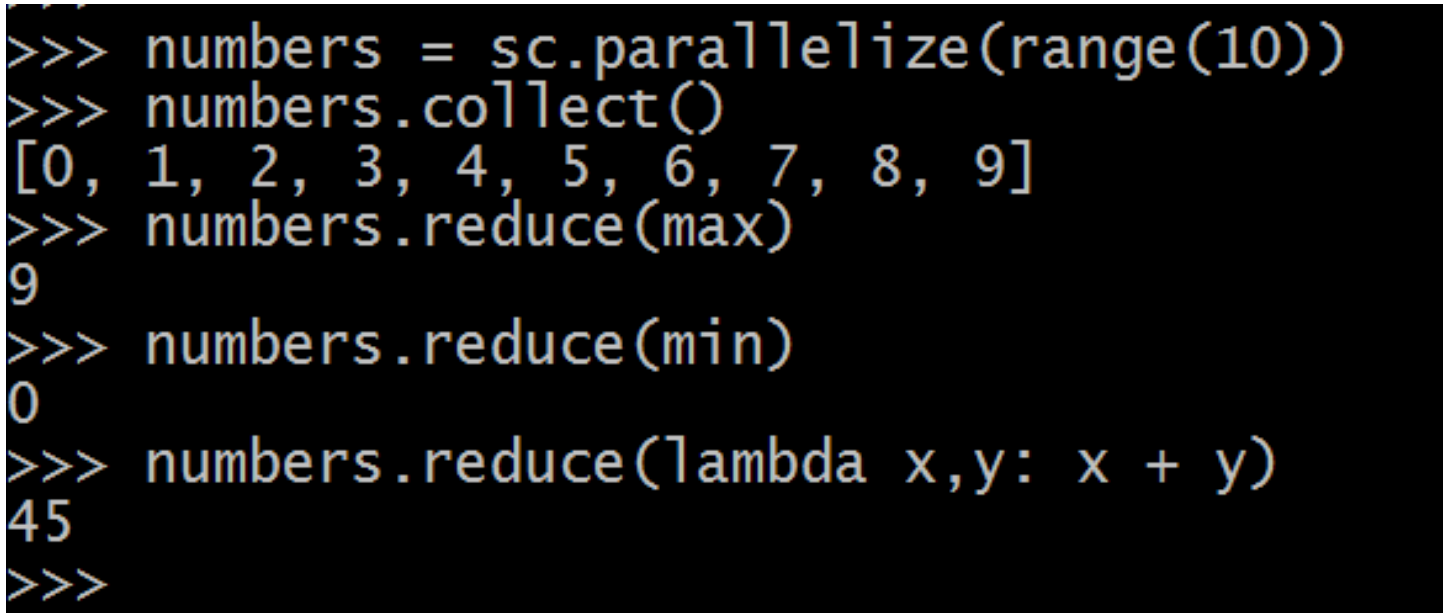
Spark Transformations

- **Filter:** works like a SQL `where` clause. Filter is called on an RDD and provided a function to filter. Spark calls the function on each element of the RDD. If the function returns true, the element will be passed to the output RDD.

```
>>> numbers = sc.parallelize(range(10))
>>> def is_even(x):
...   if (x % 2) == 0: return True
...   else: return False
...
>>> numbers.filter(is_even).collect()
[0, 2, 4, 6, 8]
>>>
>>> numbers.collect()
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
```

Spark Actions

- **Reduce:** calculates a single aggregate over all the elements of an RDD. Requires a function that is both associative and commutative. Spark applies the function to pairs of elements again and again until there is only one output left.



Spark Actions

- **ReduceByKey:** works like the SQL `group by`. Calculates an aggregate value for each key in a key pair RDD. Requires a function that is both associative and commutative. Spark applies the function to pairs of values again and again until there is only one output left for each key.

```
>>> import operator
>>> states = sc.parallelize(["TX", "TX", "TX", "NY", "NY", "VT", "CA", "CA"])
>>>
>>>
>>> states.map(lambda x: (x, 1)).collect()
[('TX', 1), ('TX', 1), ('TX', 1), ('NY', 1), ('NY', 1), ('VT', 1), ('CA', 1), ('CA', 1)]
>>>
>>>
>>> states.map(lambda x: (x, 1)).reduceByKey(operator.add).collect()
[('CA', 2), ('VT', 1), ('TX', 3), ('NY', 2)]
>>>
```

Spark Programming Guide:

https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html#transformations

https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html#actions

Milestone 1

http://www.cs.utexas.edu/~scohen/projects/m1-assignment.pdf