CLOUD-COMPUTING SPARK PAGERANK

Student: Shih-Ching, Huang

ID: 19631

Professor: Henry Chang

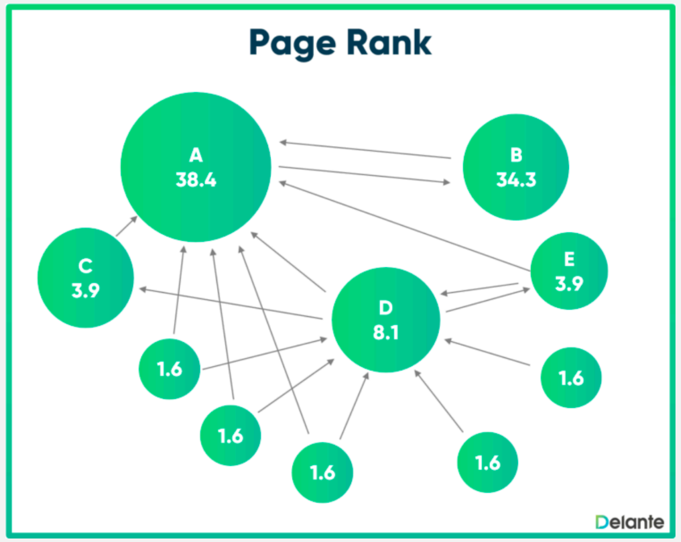
TA: Gu Liang

Course: CS570 - Big Data Processing & Analytics

TABLE OF CONTENT

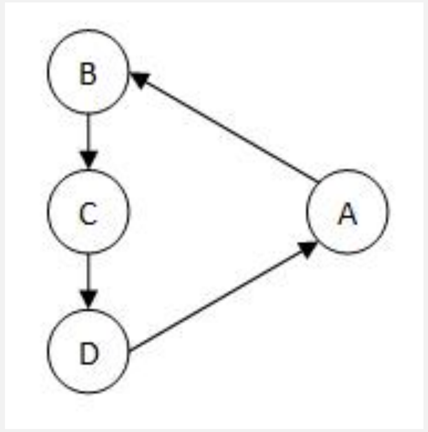
- Introduction
- Design
- Implementation
- Test
- Enhancement Ideas
- Conclusion
- References

INTRODUCTION



- PageRank (PR) is an algorithm used by Google Search to rank web pages in their search engine results. It is named after both the term "web page" and co-founder Larry Page. PageRank is a way of measuring the importance of website pages.
- The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page.
- PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process.
- The PageRank computations require several passes, called "iterations", through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.
- A probability is expressed as a numeric value between 0 and 1. A 0.5 probability is commonly expressed as a "50% chance" of something happening. Hence, a document with a PageRank of 0.5 means there is a 50% chance that a person clicking on a random link will be directed to said document.

DESIGN



- Assume a small universe of four web pages: A, B, C, and D. Links from a page to itself are ignored. Multiple outbound links from one page to another page are treated as a single link. PageRank is initialized to the same value for all pages. In the original form of PageRank, the sum of PageRank over all pages was the total number of pages on the web at that time, so each page in this example would have an initial value of 1. However, later versions of PageRank, and the remainder of this section, assume a probability distribution between 0 and 1. Hence the initial value for each page in this example is 0.25.

- The PageRank transferred from a given page to the targets of its outbound links upon the next iteration is divided equally among all outbound links.
- If the only links in the system were from pages B, C, and D to A, each link would transfer 0.25 PageRank to A upon the next iteration, for a total of 0.75.

$$PR(A) = PR(B) + PR(C) + PR(D).$$

- Suppose instead that page B had a link to pages C and A, page C had a link to page A, and page D had links to all three pages. Thus, upon the first iteration, page B would transfer half of its existing value (0.125) to page A and the other half (0.125) to page C. Page C would transfer all of its existing value (0.25) to the only page it links to, A. Since D had three outbound links, it would transfer one third of its existing value, or approximately 0.083, to A. At the completion of this iteration, page A will have a PageRank of approximately 0.458.

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}.$$

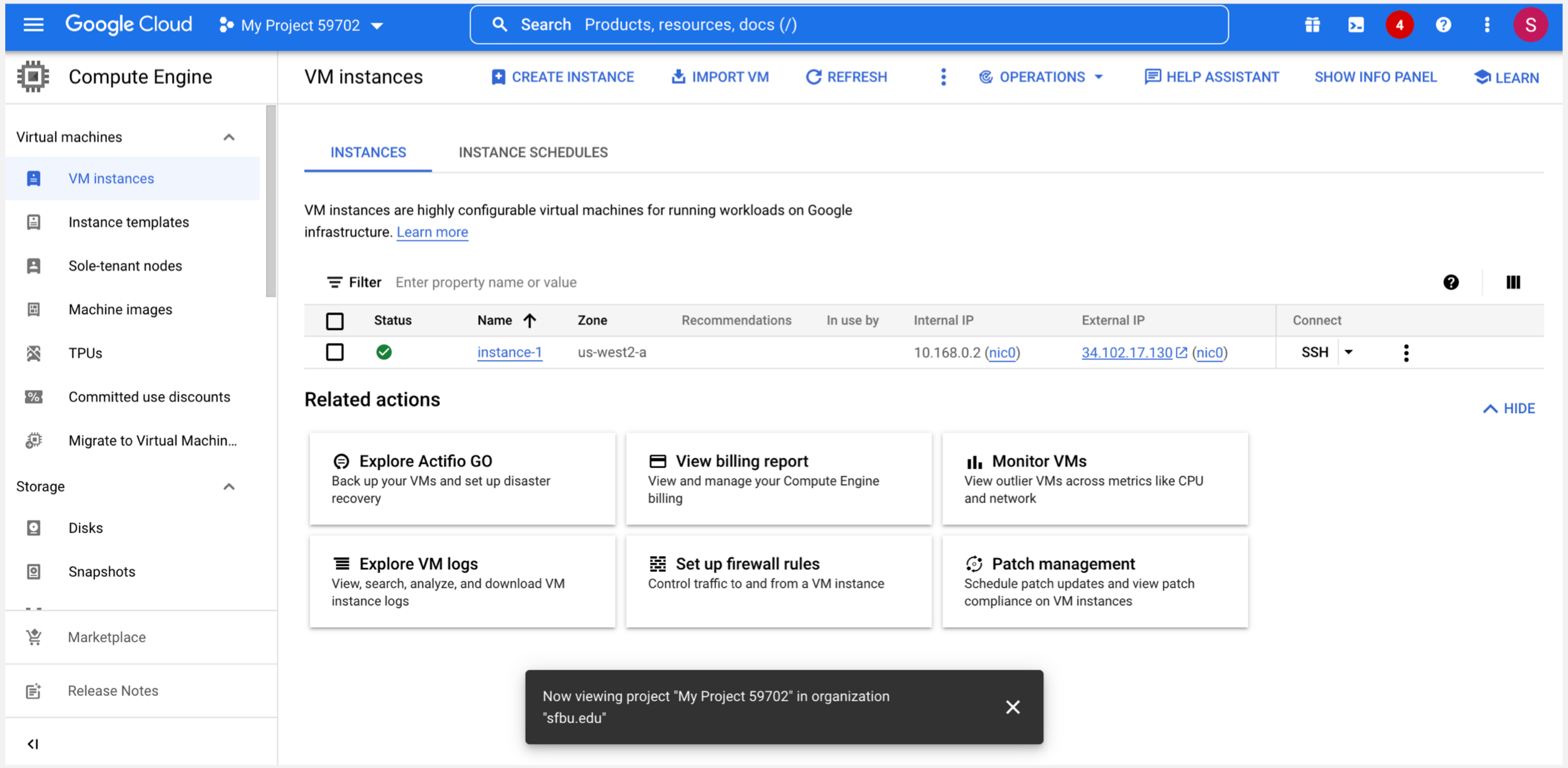
- In other words, the PageRank conferred by an outbound link is equal to the document's own PageRank score divided by the number of outbound links L( ).

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}.$$

- In the general case, the PageRank value for any page u can be expressed as:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

PREREQUISITE - GCP



VM instances are highly configurable virtual machines for running workloads on Google infrastructure. Learn more

	Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
☐	✓	instance-1	us-west2-a			10.168.0.2 (nic0)	34.102.17.130 (nic0)	SSH

Related actions

Explore Actifio GO
Back up your VMs and set up disaster recovery

View billing report
View and manage your Compute Engine billing

Monitor VMs
View outlier VMs across metrics like CPU and network

Explore VM logs
View, search, analyze, and download VM instance logs

Set up firewall rules
Control traffic to and from a VM instance

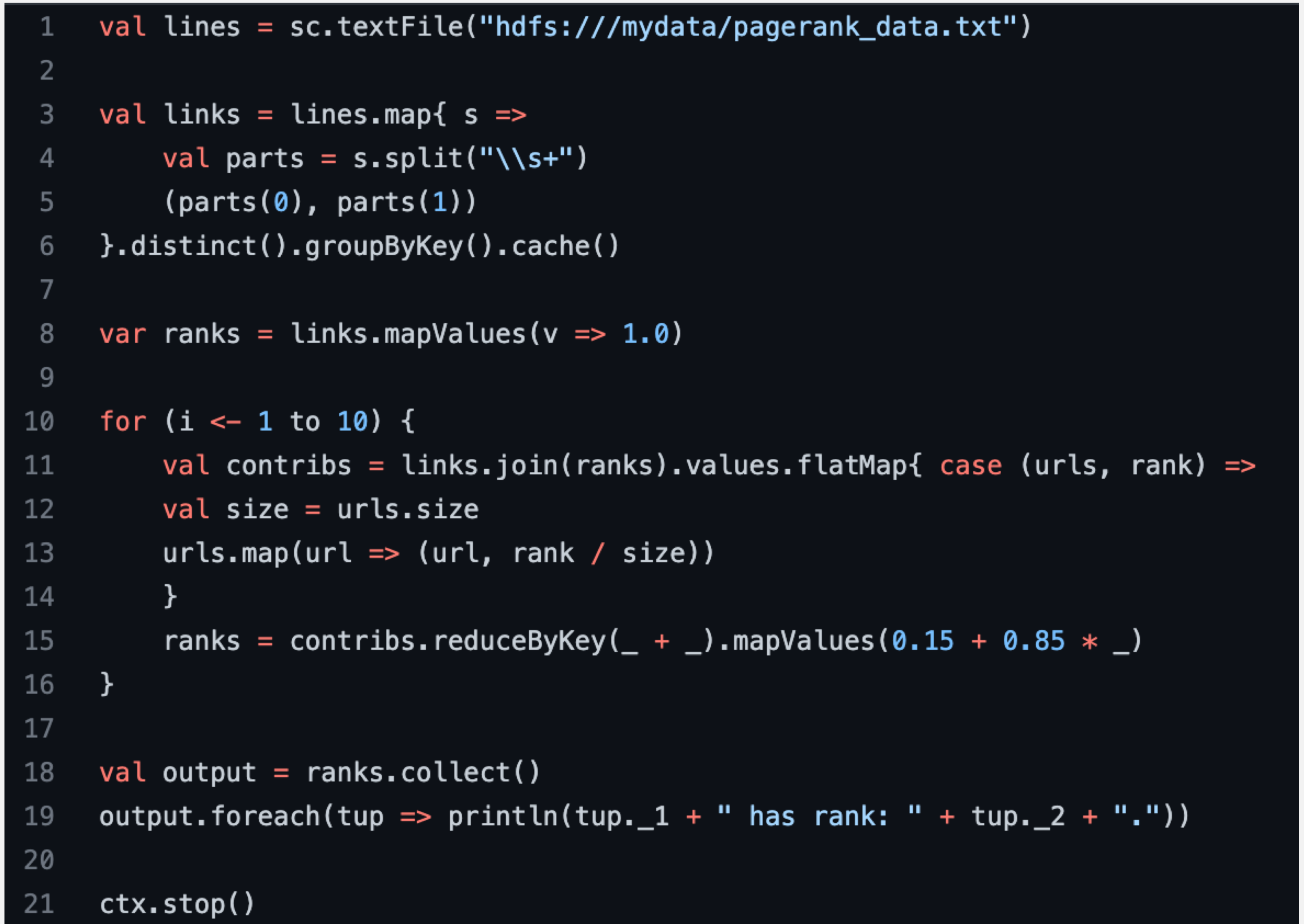
Patch management
Schedule patch updates and view patch compliance on VM instances

Now viewing project "My Project 59702" in organization "sfbu.edu"

IMPLEMENTATION

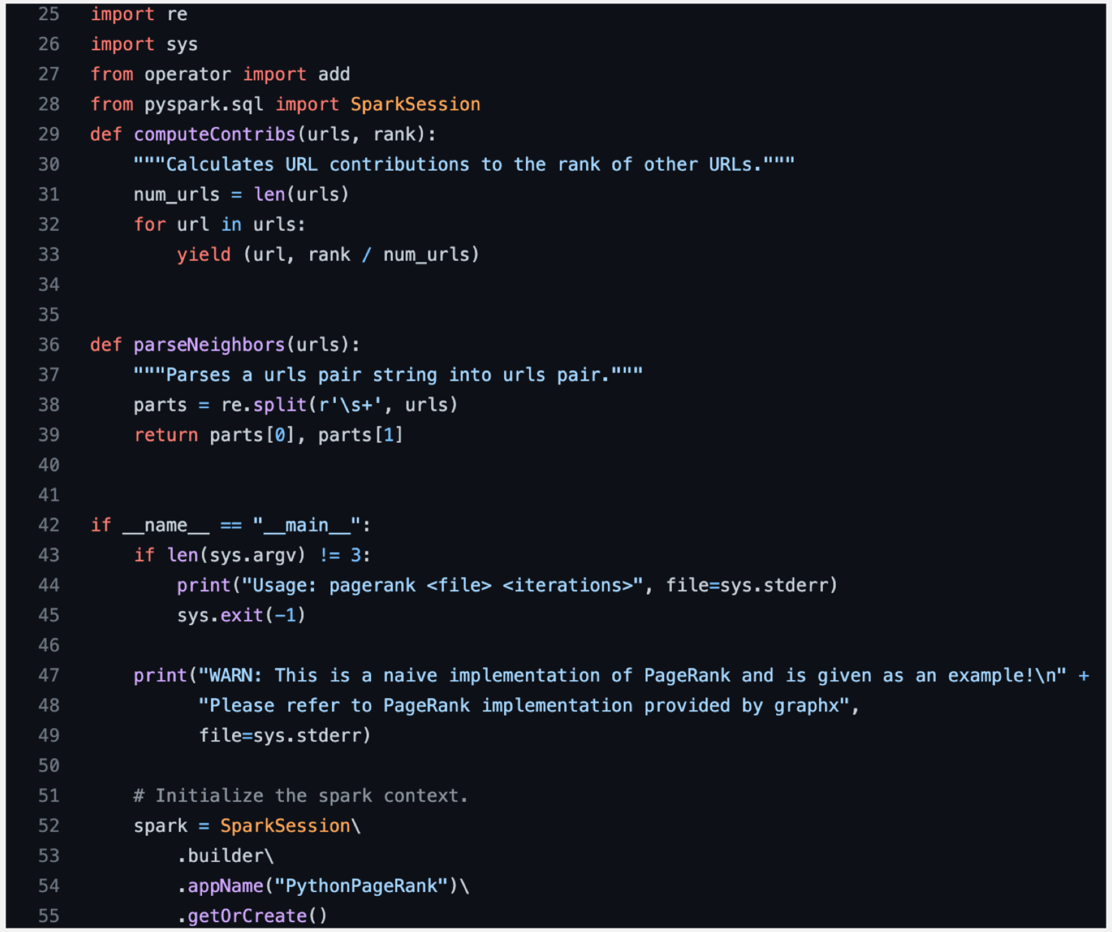
```
$ vi pagerank_data.txt
$ hdfs dfs -mkdir hdfs:///mydata
$ hdfs dfs -put pagerank_data.txt hdfs:///mydata
$ vi pagerank.py
$ spark-submit pagerank.py
$ hdfs:///mydata/pagerank_data.txt1
```

SPARKPAGERANK.SCALA



```scala
val lines = sc.textFile("hdfs:///mydata/pagerank_data.txt")

val links = lines.map{ s =>
    val parts = s.split("\\s+")
    (parts(0), parts(1))
}.distinct().groupByKey().cache()

var ranks = links.mapValues(v => 1.0)

for (i <- 1 to 10) {
    val contribs = links.join(ranks).values.flatMap{ case (urls, rank) =>
    val size = urls.size
    urls.map(url => (url, rank / size))
    }
    ranks = contribs.reduceByKey(_ + _).mapValues(0.15 + 0.85 * _)
}

val output = ranks.collect()
output.foreach(tup => println(tup._1 + " has rank: " + tup._2 + "."))

ctx.stop()
```

PAGERANK.PY

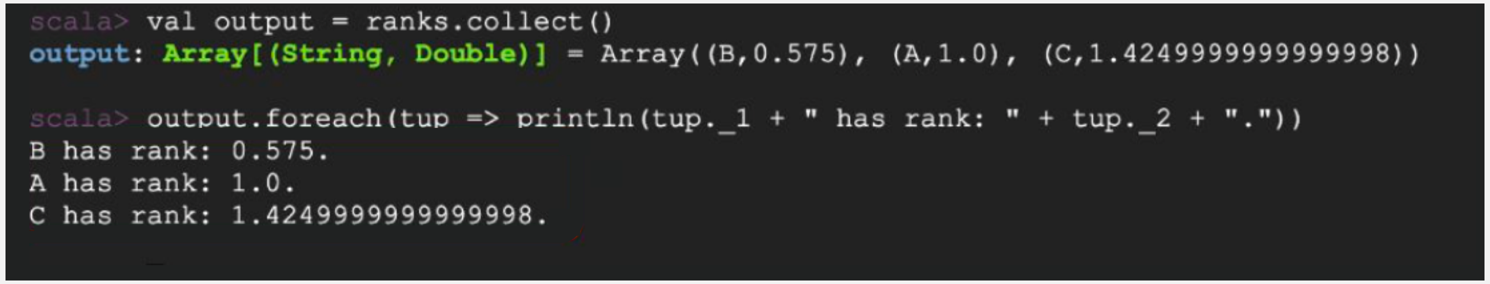


```python
import re
import sys
from operator import add
from pyspark.sql import SparkSession
def computeContribs(urls, rank):
    """Calculates URL contributions to the rank of other URLs."""
    num_urls = len(urls)
    for url in urls:
        yield (url, rank / num_urls)


def parseNeighbors(urls):
    """Parses a urls pair string into urls pair."""
    parts = re.split(r'\s+', urls)
    return parts[0], parts[1]


if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: pagerank <file> <iterations>", file=sys.stderr)
        sys.exit(-1)

    print("WARN: This is a naive implementation of PageRank and is given as an example!\n" +
          "Please refer to PageRank implementation provided by graphx",
          file=sys.stderr)

    # Initialize the spark context.
    spark = SparkSession\
        .builder\
        .appName("PythonPageRank")\
        .getOrCreate()
```

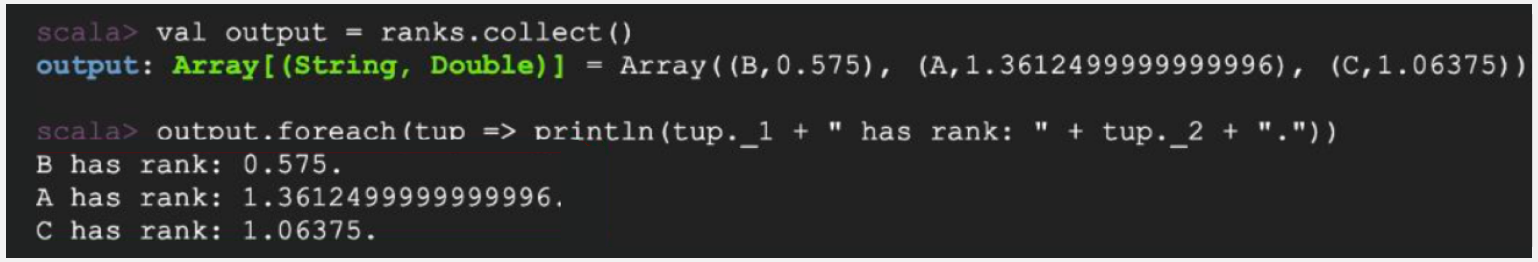
IMPLEMENTATION

```
$ curl -fL https://github.com/coursier/launchers/raw/ma ster/cs-x86_64-pc-
linux.gz | gzip -d > cs && chmod +x cs && ./cs setup

$ export SCALA_HOME=/usr/local/share/scala

$ export PATH=$PATH:$SCALA_HOME/

$ vi pagerank_data.txt

$ hdfs dfs -mkdir hdfs:///mydata

$ hdfs dfs -put pagerank_data.txt hdfs:///mydata

$ hdfs dfs -ls hdfs:///mydata
```

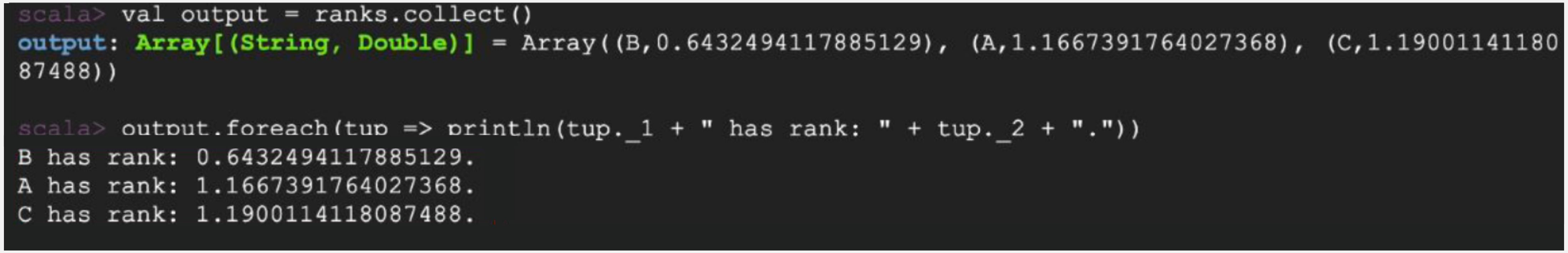
TEST



```
scala> val output = ranks.collect()
output: Array[(String, Double)] = Array((B,0.575), (A,1.0), (C,1.4249999999999998))

scala> output.foreach(tup => println(tup._1 + " has rank: " + tup._2 + "."))
B has rank: 0.575.
A has rank: 1.0.
C has rank: 1.4249999999999998.
```



```
scala> val output = ranks.collect()
output: Array[(String, Double)] = Array((B,0.575), (A,1.3612499999999996), (C,1.06375))

scala> output.foreach(tup => println(tup._1 + " has rank: " + tup._2 + "."))
B has rank: 0.575.
A has rank: 1.3612499999999996.
C has rank: 1.06375.
```



```
scala> val output = ranks.collect()
output: Array[(String, Double)] = Array((B,0.6432494117885129), (A,1.1667391764027368), (C,1.19001141180
87488))

scala> output.foreach(tup => println(tup._1 + " has rank: " + tup._2 + "."))
B has rank: 0.6432494117885129.
A has rank: 1.1667391764027368.
C has rank: 1.1900114118087488.
```

ENHANCEMENT IDEAS

- We can explore with comparing using Scala Spark and Python PySpark.

CONCLUSION

- We successfully run PageRank through Apache Spark on GCP.
- Apache Spark can use for data mining.

REFERENCES

- Course mateiral
- Page Rank Algorithm and Implementation