Page Rank : GCP

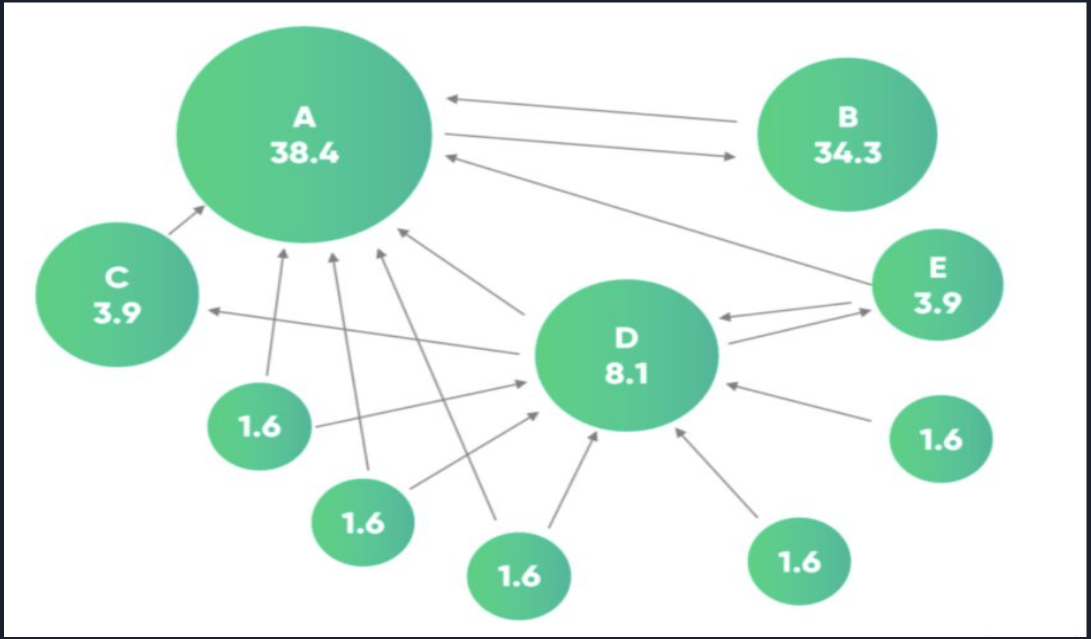
San Francisco Bay University
Dipali Gajera

Contents

1. INTRODUCTION
2. DESIGN
3. SET UP GCP
4. IMPLEMENTATION
5. TEST
6. CONCLUSION
7. REFERENCE

INTRODUCTION

- Pagerank is a search engine established by Google that was later adopted by webmasters to assess the quality of a website with reference to backlinking and SEO .

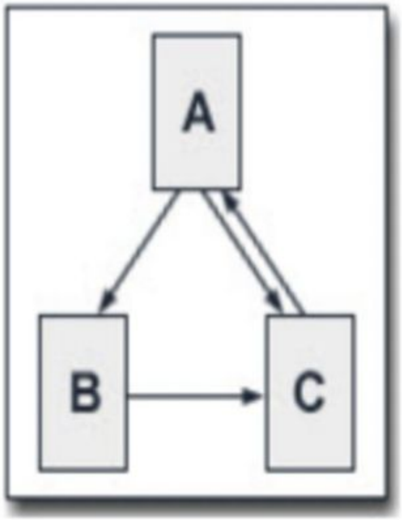


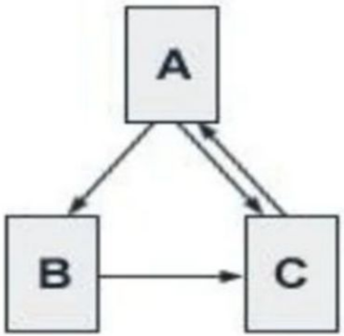
- The higher the PageRank, the more a website has a chance to rank on top pages in Google search engine.
- Pagerank also indicates the domain authority, and the high PageRank indicates high value and authority website which means if a high PageRank website links to another website with an anchor tag, the chances for another website of being crawled and indexed increases.

DESIGN

Assuming

- the initial PageRank value for each webpage is 1.
- the damping factor is 0.85
- the relation of the webpages is:





Consider an imaginary web of 3 web pages.
And the inbound and outbound link structure is as shown in the figure. The calculations can be done by following method :

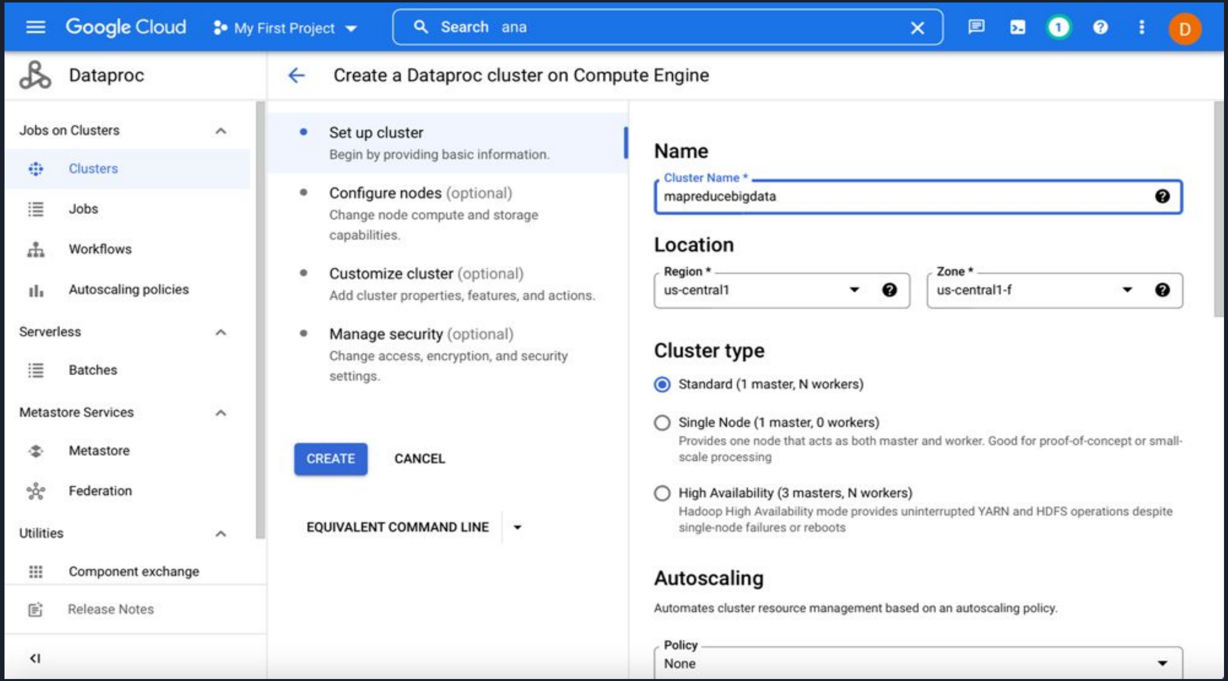
$$\begin{aligned} PR(A) &= 0.5 + 0.5\, PR(C) \\ &= 0.5 + (0.5*1) \\ &= 1 \end{aligned}$$

$$\begin{aligned} PR(B) &= 0.5 + 0.5\, (PR(A) / 2) \\ &= 0.5 + 0.5\, (1/2) \\ &= 0.5 + (0.5 * 0.5) \\ &= 0.5 + 0.25 \\ &= 0.75 \end{aligned}$$

$$\begin{aligned} PR(C) &= 0.5 + 0.5\, ((PR(A) / 2) + PR\,(B)) \\ &= 0.5 + 0.5\, (1/2 + 0.75) \\ &= 0.5 + 0.5\, (1.25) \\ &= 0.5 + 0.625 \\ &= 1.125 \end{aligned}$$

SET UP GCP

- ❖ First, Enable the Google Compute Engine API
- ❖ Create and configure the Dataporc Cluster and launch it.
- ❖ Connecting the master node through shell (SSH)



Create a Dataproc cluster on Compute Engine

- Set up cluster
  Begin by providing basic information.
- Configure nodes (optional)
  Change node compute and storage capabilities.
- Customize cluster (optional)
  Add cluster properties, features, and actions.
- Manage security (optional)
  Change access, encryption, and security settings.

Name

Cluster Name *: mapreducebigdata

Location

Region *: us-central1

Zone *: us-central1-f

Cluster type

● Standard (1 master, N workers)

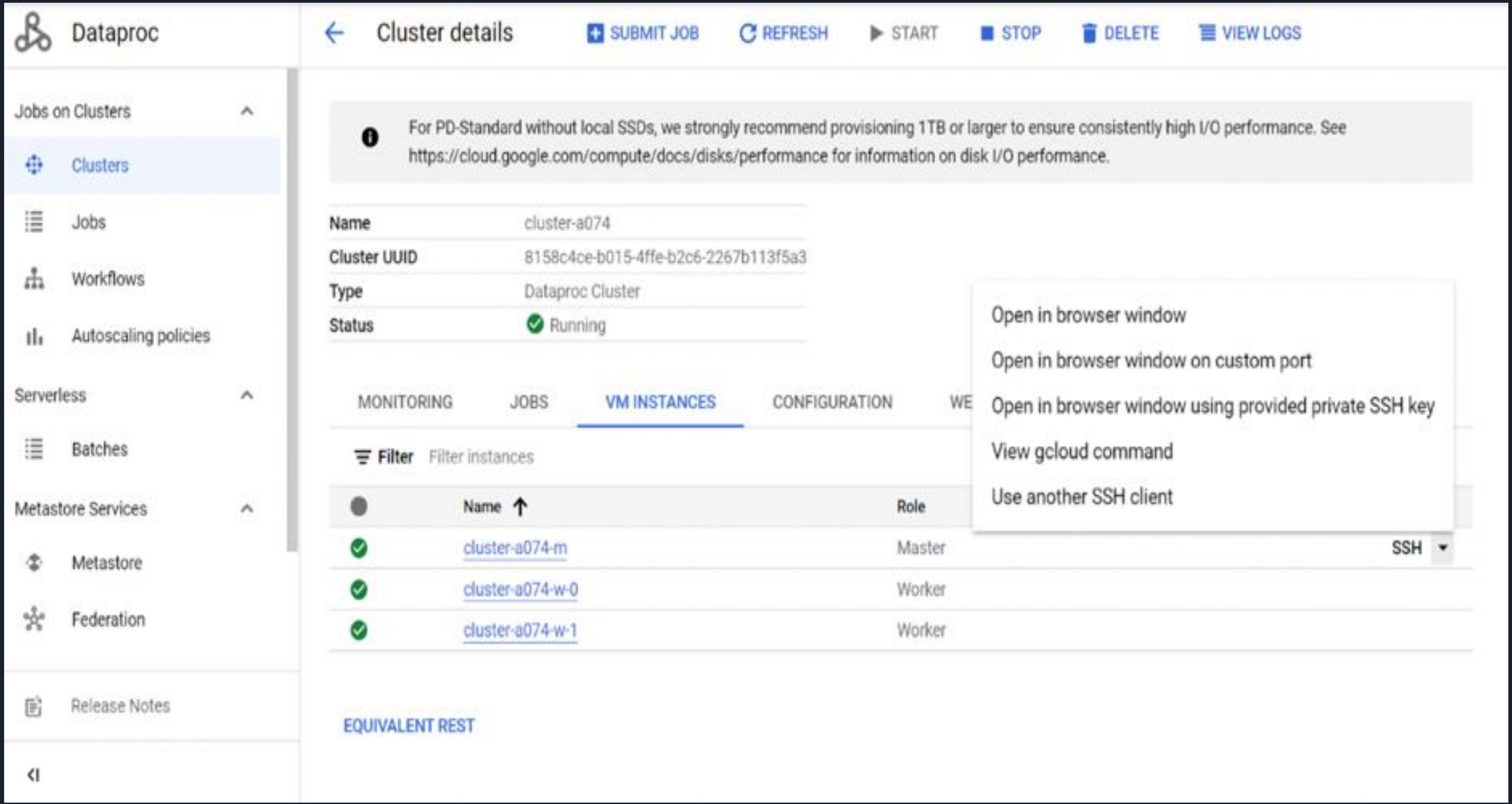
○ Single Node (1 master, 0 workers)
Provides one node that acts as both master and worker. Good for proof-of-concept or small-scale processing

○ High Availability (3 masters, N workers)
Hadoop High Availability mode provides uninterrupted YARN and HDFS operations despite single-node failures or reboots

Autoscaling

Automates cluster resource management based on an autoscaling policy.

Policy: None

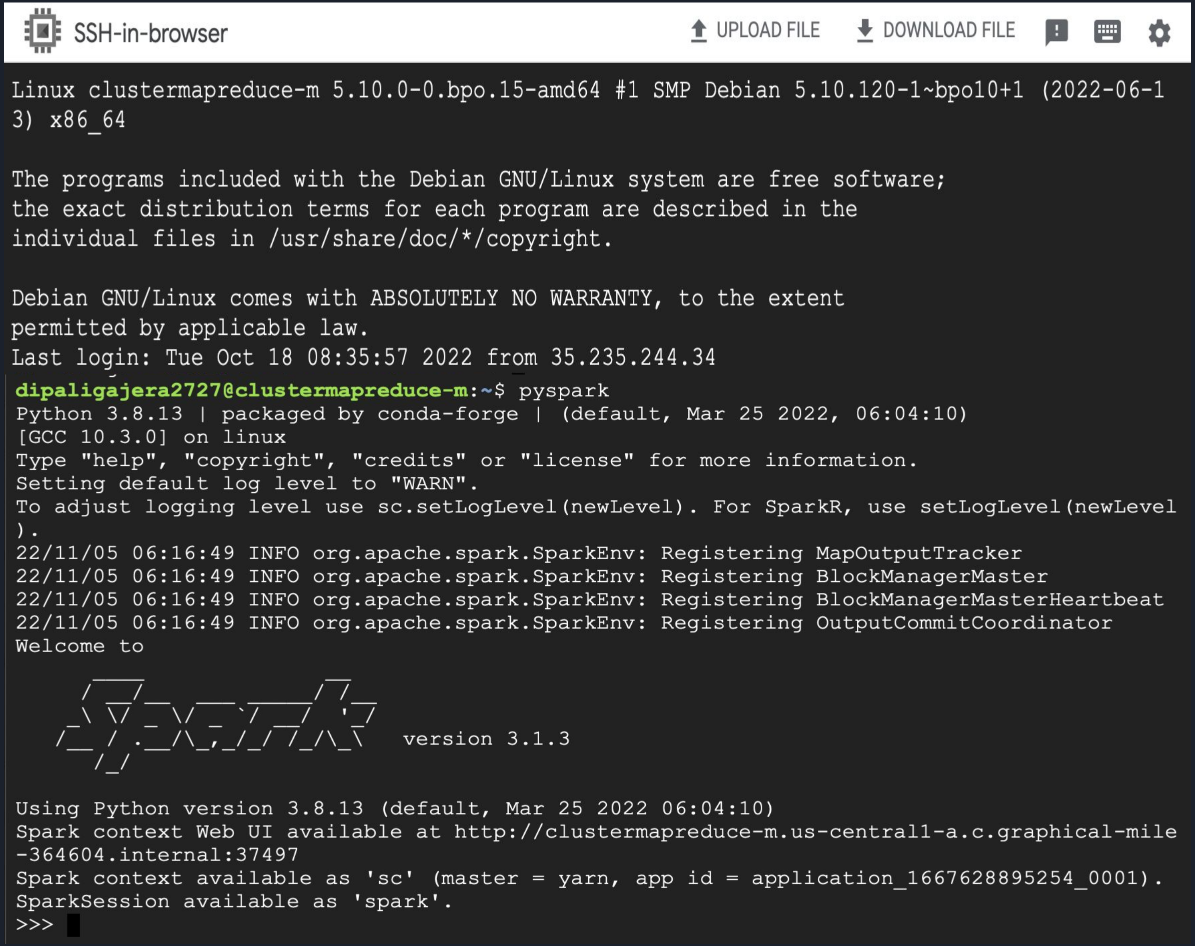


For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. See https://cloud.google.com/compute/docs/disks/performance for information on disk I/O performance.

Name	cluster-a074
Cluster UUID	8158c4ce-b015-4ffe-b2c6-2267b113f5a3
Type	Dataproc Cluster
Status	✓ Running

●	Name ↑	Role	
✓	cluster-a074-m	Master	SSH ▾
✓	cluster-a074-w-0	Worker	
✓	cluster-a074-w-1	Worker	

Open in browser window
Open in browser window on custom port
Open in browser window using provided private SSH key
View gcloud command
Use another SSH client
EQUIVALENT REST

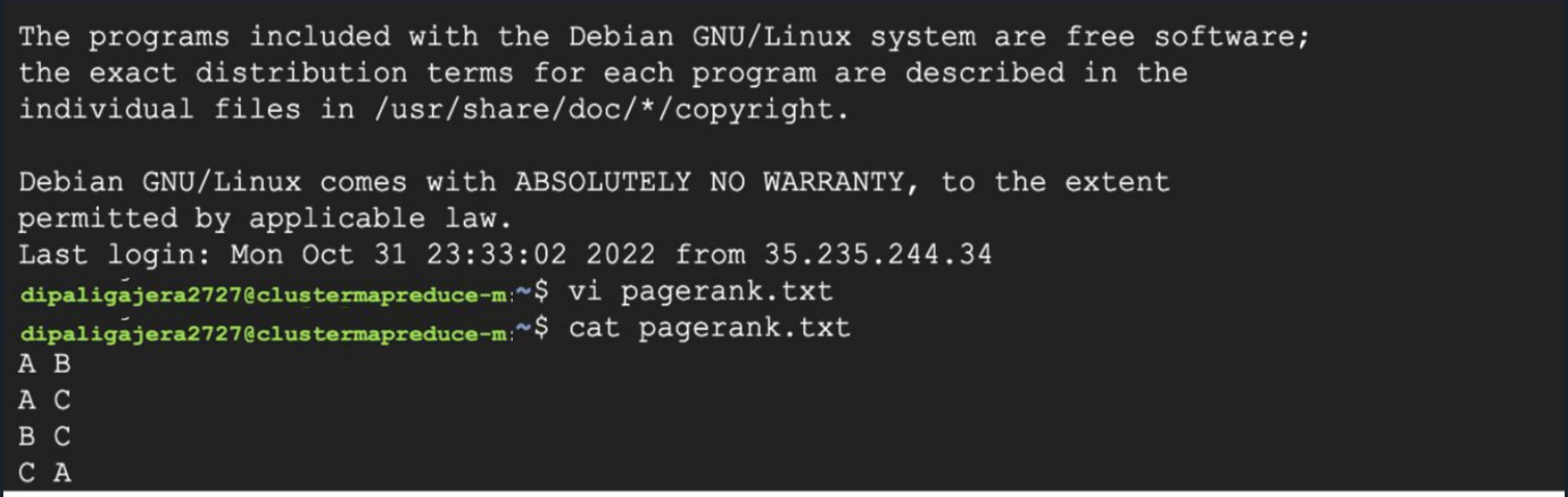


```
Linux clustermapreduce-m 5.10.0-0.bpo.15-amd64 #1 SMP Debian 5.10.120-1~bpo10+1 (2022-06-1
3) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct 18 08:35:57 2022 from 35.235.244.34
dipaligajera2727@clustermapreduce-m:~$ pyspark
Python 3.8.13 | packaged by conda-forge | (default, Mar 25 2022, 06:04:10)
[GCC 10.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel
).
22/11/05 06:16:49 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/11/05 06:16:49 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/11/05 06:16:49 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/11/05 06:16:49 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.1.3
      /_/

Using Python version 3.8.13 (default, Mar 25 2022 06:04:10)
Spark context Web UI available at http://clustermapreduce-m.us-central1-a.c.graphical-mile
-364604.internal:37497
Spark context available as 'sc' (master = yarn, app id = application_1667628895254_0001).
SparkSession available as 'spark'.
>>>
```

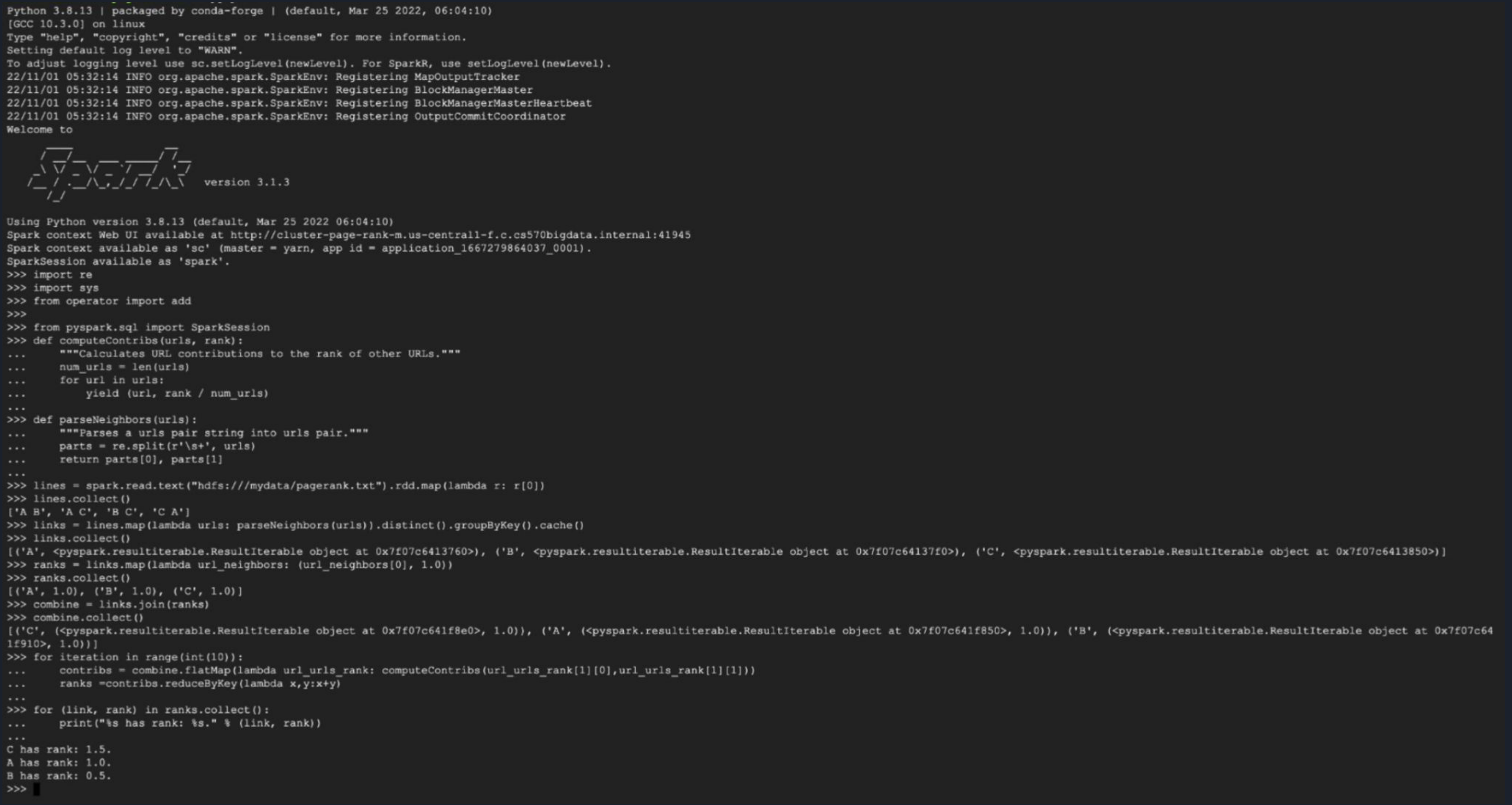
IMPLEMENTATION

- Make Data file: data.txt
  - Data File:
    - A B
    - A C
    - B C
    - C A
- hdfs dfs -mkdir hdfs:///mydata: Creation of directory
- hdfs dfs -mkdir put data.txt hdfs :///MyFile
- Program File : vi PageRank.py
- Run the program on PySpark :
  - spark submit PageRank.py hdfs:///MyFile/data.txt 1

❖ Page Rank_PySpark on GCP



```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Oct 31 23:33:02 2022 from 35.235.244.34
dipaligajera2727@clustermapreduce-m:~$ vi pagerank.txt
dipaligajera2727@clustermapreduce-m:~$ cat pagerank.txt
A B
A C
B C
C A
```



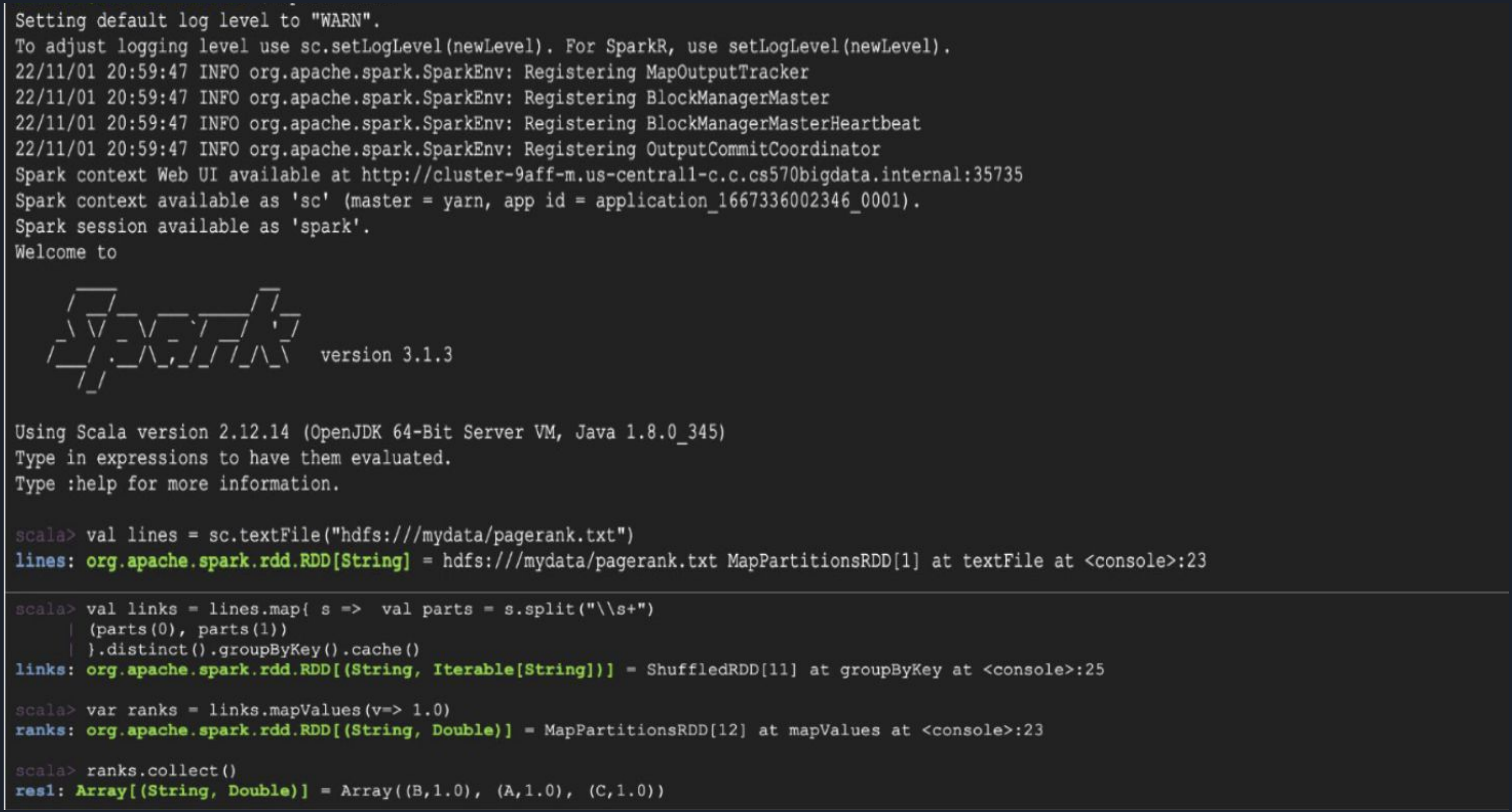
```
Python 3.8.13 | packaged by conda-forge | (default, Mar 25 2022, 06:04:10)
[GCC 10.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/11/01 05:32:14 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/11/01 05:32:14 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/11/01 05:32:14 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/11/01 05:32:14 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.1.3
      /_/

Using Python version 3.8.13 (default, Mar 25 2022 06:04:10)
Spark context Web UI available at http://cluster-page-rank-m.us-central1-f.c.cs570bigdata.internal:41945
Spark context available as 'sc' (master = yarn, app id = application_1667279864037_0001).
SparkSession available as 'spark'.
>>> 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]
...
>>> lines = spark.read.text("hdfs:///mydata/pagerank.txt").rdd.map(lambda r: r[0])
>>> lines.collect()
['A B', 'A C', 'B C', 'C A']
>>> links = lines.map(lambda urls: parseNeighbors(urls)).distinct().groupByKey().cache()
>>> links.collect()
[('A', <pyspark.resultiterable.ResultIterable object at 0x7f07c6413760>), ('B', <pyspark.resultiterable.ResultIterable object at 0x7f07c64137f0>), ('C', <pyspark.resultiterable.ResultIterable object at 0x7f07c6413850>)]
>>> ranks = links.map(lambda url_neighbors: (url_neighbors[0], 1.0))
>>> ranks.collect()
[('A', 1.0), ('B', 1.0), ('C', 1.0)]
>>> combine = links.join(ranks)
>>> combine.collect()
[('C', (<pyspark.resultiterable.ResultIterable object at 0x7f07c641f8e0>, 1.0)), ('A', (<pyspark.resultiterable.ResultIterable object at 0x7f07c641f850>, 1.0)), ('B', (<pyspark.resultiterable.ResultIterable object at 0x7f07c64
1f910>, 1.0))]
>>> for iteration in range(int(10)):
...     contribs = combine.flatMap(lambda url_urls_rank: computeContribs(url_urls_rank[1][0],url_urls_rank[1][1]))
...     ranks =contribs.reduceByKey(lambda x,y:x+y)
...
>>> for (link, rank) in ranks.collect():
...     print("%s has rank: %s." % (link, rank))
...
C has rank: 1.5.
A has rank: 1.0.
B has rank: 0.5.
>>> 
```

❖ Page Rank_PySpark on GCP

- ❖ Create Cloud Storage Bucket and Dataproc Cluster
- ❖ Connect to the shell (SSH)
- ❖ $ export SCALA_HOME=/usr/local/share/scala
- ❖ $ export PATH=$PATH:$SCALA_HOME/

Commands to perform:

```
$ export SCALA_HOME=/usr/local/share/scala
$ export PATH=$PATH:$SCALA_HOME/
Vi data.txt
hdfs dfs -mkdir hdfs :///MyFile
hdfs dfs -mkdir put data.txt hdfs :///MyFile
hdfs dfs -ls hdfs:///MyFile
```

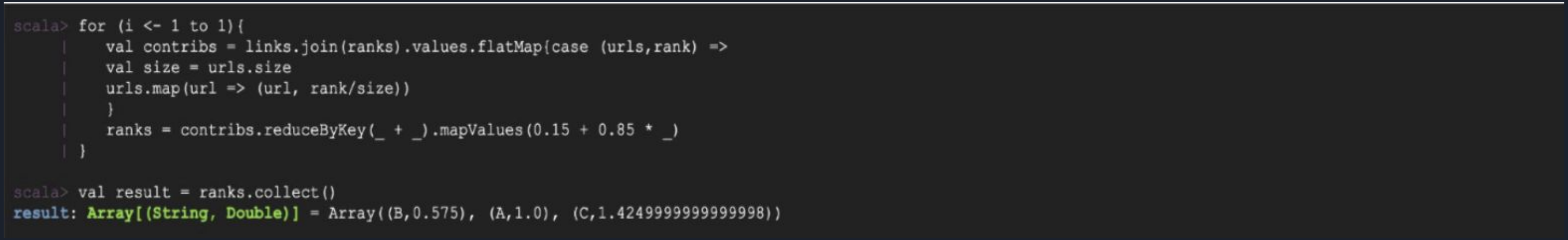


```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/11/01 20:59:47 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/11/01 20:59:47 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/11/01 20:59:47 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/11/01 20:59:47 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
Spark context Web UI available at http://cluster-9aff-m.us-central1-c.c.cs570bigdata.internal:35735
Spark context available as 'sc' (master = yarn, app id = application_1667336002346_0001).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.3
      /_/

Using Scala version 2.12.14 (OpenJDK 64-Bit Server VM, Java 1.8.0_345)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val lines = sc.textFile("hdfs:///mydata/pagerank.txt")
lines: org.apache.spark.rdd.RDD[String] = hdfs:///mydata/pagerank.txt MapPartitionsRDD[1] at textFile at <console>:23
```

```
scala> val links = lines.map{ s =>  val parts = s.split("\\s+")
     | (parts(0), parts(1))
     | }.distinct().groupByKey().cache()
links: org.apache.spark.rdd.RDD[(String, Iterable[String])] = ShuffledRDD[11] at groupByKey at <console>:25

scala> var ranks = links.mapValues(v=> 1.0)
ranks: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[12] at mapValues at <console>:23

scala> ranks.collect()
res1: Array[(String, Double)] = Array((B,1.0), (A,1.0), (C,1.0))
```

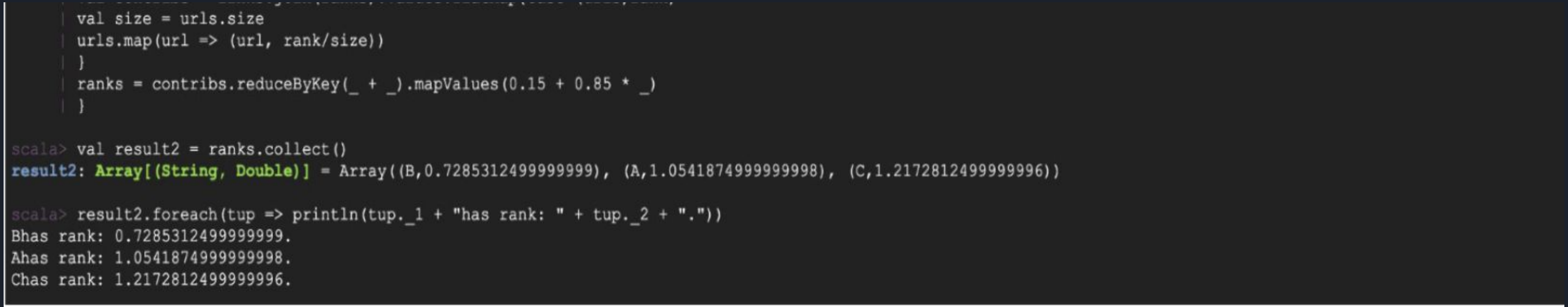
TEST

❖ 1st iteration:



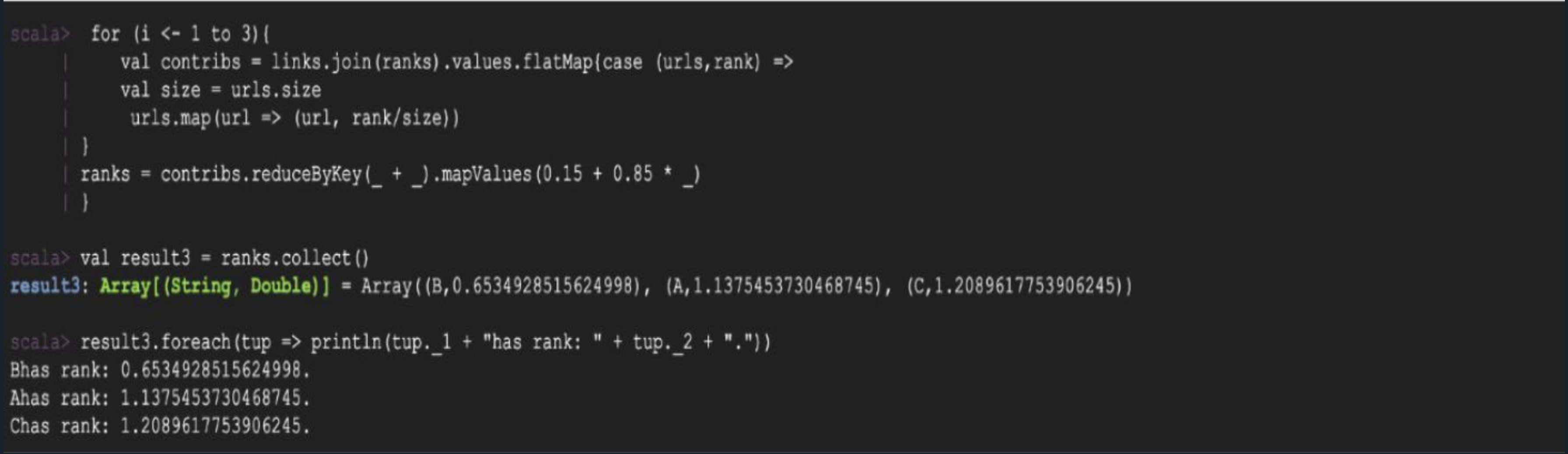
```
scala> for (i <- 1 to 1){
     |     val contribs = links.join(ranks).values.flatMap{case (urls,rank) =>
     |     val size = urls.size
     |     urls.map(url => (url, rank/size))
     |     }
     |     ranks = contribs.reduceByKey(_ + _).mapValues(0.15 + 0.85 * _)
     | }

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

❖ 2nd iteration:



```
     | val size = urls.size
     | urls.map(url => (url, rank/size))
     | }
     | ranks = contribs.reduceByKey(_ + _).mapValues(0.15 + 0.85 * _)
     | }

scala> val result2 = ranks.collect()
result2: Array[(String, Double)] = Array((B,0.7285312499999999), (A,1.0541874999999998), (C,1.2172812499999996))

scala> result2.foreach(tup => println(tup._1 + "has rank: " + tup._2 + "."))
Bhas rank: 0.7285312499999999.
Ahas rank: 1.0541874999999998.
Chas rank: 1.2172812499999996.
```

❖ 3rd iteration:



```
scala>  for (i <- 1 to 3){
     |     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 * _)
     | }

scala> val result3 = ranks.collect()
result3: Array[(String, Double)] = Array((B,0.6534928515624998), (A,1.1375453730468745), (C,1.2089617753906245))

scala> result3.foreach(tup => println(tup._1 + "has rank: " + tup._2 + "."))
Bhas rank: 0.6534928515624998.
Ahas rank: 1.1375453730468745.
Chas rank: 1.2089617753906245.
```

CONCLUSION

- I find PySpark to be simpler to learn and understand, while Scala is a more "prettier" programming language for use in production.

- Because of this, many developers write their code in PySpark before moving to Scala for production.

REFERENCE

- https://hc.labnet.sfbu.edu/~henry/npu/classes/machine_learning/text_summarization/slide/Graph_Base.html
- https://www.geeksforgeeks.org/page-rank-algorithm-implementation/
- https://en.wikipedia.org/wiki/PageRank

THANK YOU