

# PAGERANK\_SPARK

...

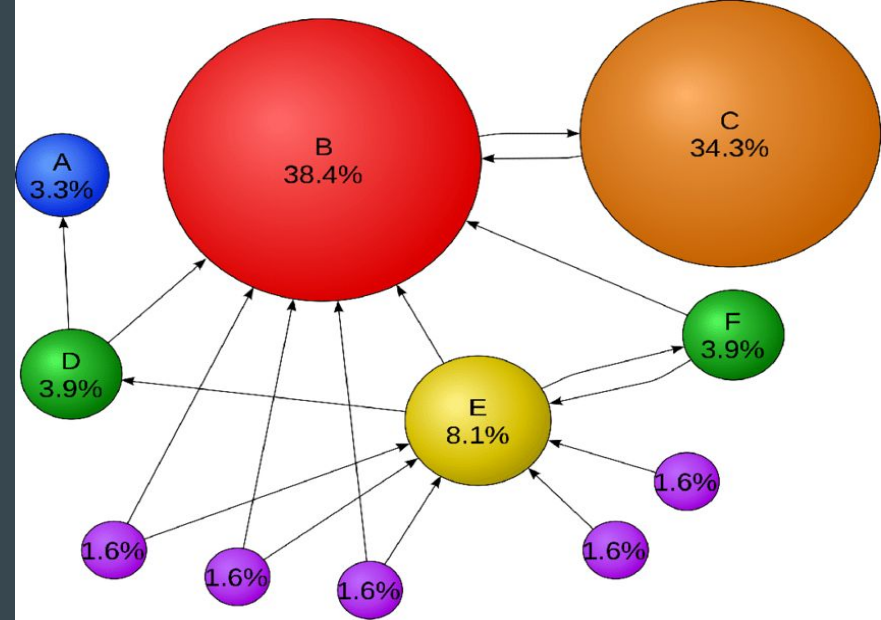
FATEMA\_NAGORI

# TABLE OF CONTENT

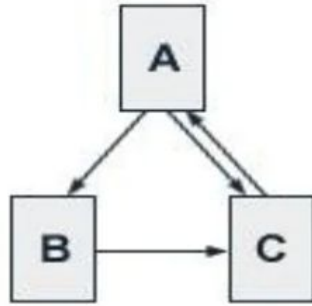
- Introduction
- Design
- Implementation
- Test
- Enhancement Ideas
- Conclusion
- References

# INTRODUCTION

- PageRank (PR) is an algorithm used by Google Search to rank web pages in their search engine results.
- PageRank is a way of measuring the importance of website pages.



# DESIGN



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 :

$$\begin{array}{lll} \text{PR(A)} = 0.5 + 0.5 \text{ PR(C)} & \text{PR(B)} = 0.5 + 0.5 (\text{PR(A)} / 2) & \text{PR(C)} = 0.5 + 0.5 ((\text{PR(A)} / 2) + \text{PR(B)}) \\ = 0.5 + (0.5 * 1) & & \\ = 1 & = 0.5 + 0.5 (1/2) & = 0.5 + 0.5 (1/2 + 0.75) \\ & = 0.5 + (0.5 * 0.5) & = 0.5 + 0.5 (1.25) \\ & = 0.5 + 0.25 & = 0.5 + 0.625 \\ & = 0.75 & = 1.125 \end{array}$$

# DESIGN

A web page does not have input will have:

constant PageRank:  $1-d$

the smallest PageRank Input Web Pages' impact to the PageRank of a web page:

The more Input Web Pages the better.

The higher PageRank of an Input Web Page the better..

An iterative algorithm that performs many joins, so it is a good use case for RDD partitioning.

The algorithm maintains two datasets:

(pageID, link List) elements containing the list of neighbors of each page,

(pageID, rank) elements containing the current rank for each page.

# SETUP GCP

- Enable the Google Cloud Compute Engine API
- Create, Configure and Launch a Google Cloud Dataproc cluster
- Connecting to the Master Node using Secure Shell (ssh)

The screenshot shows the Google Cloud console interface for creating a Dataproc cluster. The top navigation bar includes the Google Cloud logo, a dropdown menu for 'New CS570', and a search bar. The left sidebar contains navigation links for 'Dataproc', 'Jobs on Clusters', 'Clusters', 'Jobs', 'Workflows', 'Autoscaling policies', 'Serverless', 'Batches', and 'Metastore Services'. The main content area is titled 'Create a Dataproc cluster on Compute Engine' and features a list of steps: 'Set up cluster' (selected), 'Configure nodes (optional)', 'Customize cluster (optional)', and 'Manage security (optional)'. The 'Set up cluster' section is further divided into 'Name', 'Location', and 'Cluster type'.

**Name**

Cluster Name \*  
cluster-6926

**Location**

Region \*  
us-central1

Zone \*  
us-central1-a

**Cluster type**

☒ Standard (1 master, N workers)

☐ Single Node (1 master, 0 workers)

## Dataproc

Workflows  
Autoscaling policies

Serverless ^

Batches

Metastore Services ^

Metastore

Federation

### Cluster details

SUBMIT JOB

REFRESH

START

STOP

DELETE

VIEW LOGS



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-6926
Cluster UUID	55d54331-cc70-469b-88c3-c3bbe1dff51c
Type	Dataproc Cluster
Status	Running

MONITORING

JOBS

VM INSTANCES

CONFIGURATION

WE



Filter instances



Name ↑

Role

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

## Google Cloud Data Catalog API

[https://ssh.cloud.google.com/v2/ssh/projects/new-cs570/zones/us-central1-a/instances/cluster-6926-m?authuser=1&hl=en\\_US&projectN...](https://ssh.cloud.google.com/v2/ssh/projects/new-cs570/zones/us-central1-a/instances/cluster-6926-m?authuser=1&hl=en_US&projectN...)

[https://ssh.cloud.google.com/v2/ssh/projects/new-cs570/zones/us-central1-a/instances/cluster-6926-m?authuser=1&hl=en\\_US&p...](https://ssh.cloud.google.com/v2/ssh/projects/new-cs570/zones/us-central1-a/instances/cluster-6926-m?authuser=1&hl=en_US&p...)

SSH-in-browser

```
Linux cluster-6926-m 5.10.0-0.bpo.15-amd64 #1 SMP Debian 5.10.120-1-bpo10+1 (2022-06-13) 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: Wed Oct 5 19:59:54 2022 from 35.235.244.33
```

```
fnagori@cluster-6926-m:~$
```

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

cluster-6926

# Launch Spark shell at GCP

```
fnagori@cluster-6926-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).
22/10/05 20:09:02 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/10/05 20:09:02 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/10/05 20:09:02 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/10/05 20:09:02 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
Welcome to
```



```
Using Python version 3.8.13 (default, Mar 25 2022 06:04:10)
Spark context Web UI available at http://cluster-6926-m.us-central1-a.c.new-cs570.internal:34509
Spark context available as 'sc' (master = yarn, app id = application_1664999425862_0001).
SparkSession available as 'spark'.
>>> # The SparkContext 'sc' is already defined.
>>> # Set the Logging Level to ERROR only
>>> sc.setLogLevel("ERROR")
>>> # Set the Logging Level to ERROR only
>>> sc.setLogLevel("ERROR")
```



# Implementation-PageRank\_PySpark\_GCP

## STEP:

- Prepare Data:vi pagerank\_data.txt
- Create a directory (folder) to store the data: hdfs  
dfs -mkdir hdfs:///mydata
- Copy the date to HDFS: hdfs dfs -put  
pagerank\_data.txt hdfs:///mydata
- Prepare the program: vi pagerank.py
- Running the program with Pyspark: spark-submit  
pagerank.py hdfs:///mydata/pagerank\_input.txt 1

## INPUT FILE FORMAT

A B  
A C  
B C  
C A

# TEST-PageRank\_PySpark\_GCP (1st-iteration)

```
nagorifatmal@cluster-5854-m-0:~$ hdfs dfs -put pagerank input.txt hdfs:///mydata
nagorifatmal@cluster-5854-m-0:~$ spark-submit pagerank.py hdfs:///mydata/pagerank_input.txt 1
WARN: this is a naive implementation of PageRank and is given as an example!
Please refer to PageRank implementation provided by graphx
22/11/01 21:29:35 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/11/01 21:29:35 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/11/01 21:29:35 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/11/01 21:29:35 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
22/11/01 21:29:35 INFO org.sparkproject.jetty.util.log: Logging initialized @3688ms to org.sparkproject.jetty.util.log.Slf4jLog
22/11/01 21:29:35 INFO org.sparkproject.jetty.server.Server: jetty-9.4.40.v20210413; built: 2021-04-13T20:42:42.668Z; git: b881a572662e1943a14ae12e7e1207989f218
b74; jvm 1.8.0_345-b01
22/11/01 21:29:35 INFO org.sparkproject.jetty.server.Server: Started @3846ms
22/11/01 21:29:36 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@24e81266(HTTP/1.1, (http/1.1)){0.0.0.0:44567}
22/11/01 21:29:37 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at cluster-5854-m-0.us-central1-b.c.new-cs570.internal./
10.128.0.18:10200
22/11/01 21:29:37 INFO org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider: Failing over to rml
22/11/01 21:29:37 INFO org.apache.hadoop.io.retry.RetryInvocationHandler: java.net.ConnectException: Call From cluster-5854-m-0/10.128.0.18 to cluster-5854-m-1.
us-central1-b.c.new-cs570.internal.:8032 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apach
e.org/hadoop/ConnectionRefused, while invoking ApplicationClientProtocolPBClientImpl.getNewApplication over rml after 1 failover attempts. Trying to failover af
ter sleeping for 764ms.
22/11/01 21:29:37 INFO org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider: Failing over to rm2
22/11/01 21:29:39 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found
22/11/01 21:29:39 INFO org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/11/01 21:29:40 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted application application_1667337547764_0002
22/11/01 21:29:41 INFO org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider: Failing over to rml
22/11/01 21:29:41 INFO org.apache.hadoop.io.retry.RetryInvocationHandler: java.net.ConnectException: Call From cluster-5854-m-0/10.128.0.18 to cluster-5854-m-1.
us-central1-b.c.new-cs570.internal.:8030 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apach
e.org/hadoop/ConnectionRefused, while invoking ApplicationMasterProtocolPBClientImpl.registerApplicationMaster over rml after 1 failover attempts. Trying to fai
lover after sleeping for 693ms.
22/11/01 21:29:41 INFO org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider: Failing over to rm2
22/11/01 21:29:43 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonRespons
eException; verified object already exists with desired state.
C has rank: 1.4249999999999998.
A has rank: 1.0.
B has rank: 0.575.
22/11/01 21:29:56 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@24e81266(HTTP/1.1, (http/1.1)){0.0.0.0:0}
nagorifatmal@cluster-5854-m-0:~$
```

# TEST-PageRank\_PySpark\_GCP (10th-iteration)

```
22/11/01 21:31:39 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@3babe07(HTTP/1.1, (http/1.1)){0.0.0.0:0}
nagorifatmal@cluster-5854-m-0:~$ spark-submit pagerank.py hdfs:///mydata/pagerank input.txt 10
WARN: This is a naive implementation of PageRank and is given as an example!
Please refer to PageRank implementation provided by graphx
22/11/01 21:32:13 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/11/01 21:32:13 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/11/01 21:32:13 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/11/01 21:32:13 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
22/11/01 21:32:13 INFO org.sparkproject.jetty.util.log: Logging initialized @3781ms to org.sparkproject.jetty.util.log.Slf4jLog
22/11/01 21:32:13 INFO org.sparkproject.jetty.server.Server: jetty-9.4.40.v20210413; built: 2021-04-13T20:42:42.668Z; git: b881a572662e1943a14ae12e7e1207989f218
b74; jvm 1.8.0_345-b01
22/11/01 21:32:13 INFO org.sparkproject.jetty.server.Server: Started @3932ms
22/11/01 21:32:13 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@ab57f85(HTTP/1.1, (http/1.1)){0.0.0.0:42679}
22/11/01 21:32:14 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at cluster-5854-m-0.us-centrall-b.c.new-cs570.internal./
10.128.0.18:10200
22/11/01 21:32:14 INFO org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider: Failing over to rml
22/11/01 21:32:14 INFO org.apache.hadoop.io.retry.RetryInvocationHandler: java.net.ConnectException: Call From cluster-5854-m-0/10.128.0.18 to cluster-5854-m-1.
us-centrall-b.c.new-cs570.internal.:8032 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apach
e.org/hadoop/ConnectionRefused, while invoking ApplicationClientProtocolPBClientImpl.getNewApplication over rml after 1 failover attempts. Trying to failover af
ter sleeping for 1312ms.
22/11/01 21:32:16 INFO org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider: Failing over to rm2
22/11/01 21:32:17 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found
22/11/01 21:32:17 INFO org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/11/01 21:32:18 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted application application_1667337547764_0004
22/11/01 21:32:19 INFO org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider: Failing over to rml
22/11/01 21:32:19 INFO org.apache.hadoop.io.retry.RetryInvocationHandler: java.net.ConnectException: Call From cluster-5854-m-0/10.128.0.18 to cluster-5854-m-1.
us-centrall-b.c.new-cs570.internal.:8030 failed on connection exception: java.net.ConnectException: Connection refused; For more details see: http://wiki.apach
e.org/hadoop/ConnectionRefused, while invoking ApplicationMasterProtocolPBClientImpl.registerApplicationMaster over rml after 1 failover attempts. Trying to fai
lower after sleeping for 1249ms.
22/11/01 21:32:20 INFO org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider: Failing over to rm2
22/11/01 21:32:22 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonRespons
eException: verified object already exists with desired state.
A has rank: 1.1667391764027368.
B has rank: 0.6432494117885129.
C has rank: 1.1900114118087488.
22/11/01 21:32:37 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@ab57f85(HTTP/1.1, (http/1.1)){0.0.0.0:0}
nagorifatmal@cluster-5854-m-0:~$
```

# Implementation-PageRank\_Scala\_GCP

## SETUP Scala Environment at GCP

- Create a Cloud Storage bucket
- Create a Dataproc cluster
- Connecting to the Master Node using Secure Shell (ssh)
- install scala
- `$ export SCALA_HOME=/usr/local/share/scala`
- `$ export PATH=$PATH:$SCALA_HOME/`

## Scala program:

- 1.Prepare data: vi pagerank\_data.txt
- 2.create a directory (folder) to store the data:
  - `hdfs dfs -mkdir hdfs:///mydata`
  - `hdfs dfs -put pagerank_data.txt hdfs:///mydata`
  - `hdfs dfs -ls hdfs:///mydata`
- spark-shell

# Launch Scala on GCP

```
fnagori@cluster-0386-m:~$ export PATH=$PATH:$SCALA_HOME/
```

```
fnagori@cluster-0386-m:~$ scala
```

```
Welcome to Scala 2.12.14 (OpenJDK 64-Bit Server VM, Java 1.8.0_332).
```

```
Type in expressions for evaluation. Or try :help.
```

# Implementation-PageRank\_Scala\_GCP

## COMMANDS:

```
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()
```

INPUT FILE  
FORMAT

A B

A C

B C

C A



# SUBMIT JOB AT GCP

Google Cloud

New CS570 ▾

Q

Search Products, resources, docs (/)

21

F

Dataproc

Jobs on Clusters ▴

Clusters

**Jobs**

Workflows

Autoscaling policies

Serverless ▴

Batches

Metastore Services ▴

Metastore

Federation

Release Notes

<|

← Submit a job

Job ID \*  
job-b54ca707

Region \*  
us-central1  
Specifies the Cloud Dataproc regional service, which determines what clusters are available.

Cluster \*  
cluster-0386

Job type \*  
Spark

**Main class or jar \***  
gs://dataproc-staging-us-central1-1049843925247-hmciannh/HelloWorld.jar  
The fully qualified name of a class in a provided or standard jar file, for example, com.example.wordcount, or a provided jar file to use the main class of that jar file

Jar files  
Jar files are included in the CLASSPATH. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hd

Archive files

Starting cluster... X

# TEST-PageRank\_Scala\_GCP (1ST iteration)

```
scala> 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 * _)  
}
```

```
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>
```

```
scala> val output = ranks.collect()  
output: Array[(String, Double)] = Array((B,0.644434510492888), (A,1.1633753188067952), (C,1.1921901707003153))
```

```
scala> output.foreach(tup => println(tup.1 + " has rank: " + tup.2 + "."))
```

```
B has rank: 0.644434510492888.  
A has rank: 1.1633753188067952.  
C has rank: 1.1921901707003153.
```



# TEST-PageRank\_Scala\_GCP (2nd iteration)

```
scala>

scala> val output = ranks.collect()
output: Array[(String, Double)] = Array((B,0.644434510492888), (A,1.1633753188067952), (C,1.1921901707003153))

scala> output.foreach(tup => println(tup._1 + " has rank: " + tup._2 + "."))
B has rank: 0.644434510492888.
A has rank: 1.1633753188067952.
C has rank: 1.1921901707003153.
```

```
scala> for (i <- 1 to 2) {
|   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>

scala> val output = ranks.collect()
output: Array[(String, Double)] = Array((B,0.6444286991654888), (A,1.163373267750066), (C,1.1921980330844435))

scala> output.foreach(tup => println(tup._1 + " has rank: " + tup._2 + "."))
B has rank: 0.6444286991654888.
A has rank: 1.163373267750066.
C has rank: 1.1921980330844435.
```

# TEST-PageRank\_Scala\_GCP (10 iteration)

```
scala> val lines = sc.textFile("hdfs:///mydata/pagerank_input.txt")
lines: org.apache.spark.rdd.RDD[String] = hdfs:///mydata/pagerank_input.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[6] at groupByKey at <console>:26

scala>

scala> var ranks = links.mapValues(v => 1.0)
ranks: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[7] at mapValues at <console>:23

scala>

scala> 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 * _)
    | }

scala> val output = ranks.collect()
[Stage 0:>                                     (0 + 2) [Stage 1:>                                     (0 + 2)
output: Array[(String, Double)] = Array((B,0.6432494117885129), (A,1.1667391764027368), (C,1.19
00114118087488))

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

## Python vs Scala

1

Performance

2

Simplicity

3

Concurrency

4

Type Safety

5

Productivity and  
Ease of Use

6

Advanced Features

# CONCLUSION:



Both the program results for Scala and Python was same like manual result.

From a more personal point of view, I consider **PySpark** easier to learn and read, however for production **Scala** is a much "prettier" programming language. For this reason, many people program in PySpark and when all the code is validated, they go to production in Scala.

# REFERENCES

[PAGE RANK EXAMPLES](#)

[SCALA AT GCP](#)

[PYTHON AT GCP](#)

[EXAMPLES OF THEORY](#)

