

# Page Rank: GCP



SFBU  
Navya Kandimalla.

# Contents

1. Introduction
2. Design
3. Set Up GCP
4. Implementation
5. Test
6. Enhancement Ideas
7. Conclusion
8. Reference



# Introduction

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

The original PageRank algorithm was described by Lawrence Page and Sergey Brin in several publications. It is given by

$$PR(A) = (1-d) + d (PR(T1)/C(T1) + \dots + PR(Tn)/C(Tn))$$

Where

PR(A) is the PageRank of page A,  
PR(Ti) is the PageRank of pages Ti which link to page A,  
C(Ti) is the number of outbound links on page Ti and  
d is a damping factor which can be set between 0 and 1.

# Explanation

We consider a small web of three pages, A, B, and C, with links from page A to pages B and C, from page B to page C, and from page C to page A. Page and Brin state that the damping factor  $d$  is often set around 0.85.

If there is no user input, PageRank will remain constant. Impact of a web page's PageRank of  $1-d$  the least PageRank Input Web Pages

The better, the more input web pages.

An input web page's PageRank should be as high as possible



# SET UP GCP

- Enable the google compute engine API
- Create the Dataproc Cluster and configure it.
- Launch the Dataproc Cluster
- Connecting the master to the SSH.

The screenshot shows the Google Cloud Platform console interface for the Dataproc Clusters page. The URL in the browser is `console.cloud.google.com/dataproc/clusters?authuser=1&project=mystical-accord-366002`. The page header includes the Google Cloud logo, the project name 'My First Project', a search bar for clusters, and a list of actions: CREATE CLUSTER, REFRESH, START, STOP, DELETE, and a dropdown for REGIONS. There is also a link to '+ 5 RECOMMENDED ALERTS' and a 'SHOW INFO PANEL' button.

The main content area displays a table of clusters. The table has the following columns: Name, Status, Region, Zone, Total worker nodes, Scheduled deletion, Cloud Storage staging bucket, and Created. A filter bar at the top of the table says 'Filter Search clusters, press Enter'. The table contains one cluster entry:

Name	Status	Region	Zone	Total worker nodes	Scheduled deletion	Cloud Storage staging bucket	Created
cluster-	Stopped	us-central1	us-central1-	2	Off	<a href="#">dataproc-staging-us-central1-248446181593-</a>	Oct 20, 2022, 6:38:59

The left sidebar shows the navigation menu with the following items: Clusters (selected), Jobs, Workflows, Autoscaling policies, Serverless, Batches, Metastore Services, Federation, Utilities, Component exchange, and Workbench.



## Cluster details

+ SUBMIT JOB

🔄 REFRESH

▶ START

■ STOP

🗑 DELETE

☰ VIEW LOGS

ⓘ Consider using Auto Zone rather than selecting a zone manually. See <https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/auto-zone>

[MORE](#)

Name	cluster-pagerank
Cluster UUID	73b8aa59-4899-4815-aff2-7df9535f71eb
Type	Dataproc Cluster
Status	⏸ Stopped

MONITORING

JOBS

**VM INSTANCES**

CONFIGURATION

WEB INTERFACES

☰ Filter Filter instances



●	Name ↑	Role	
○	<a href="#">cluster-pagerank-m</a>	Master	SSH ▾
○	<a href="#">cluster-pagerank-w-0</a>	Worker	
○	<a href="#">cluster-pagerank-w-1</a>	Worker	

[EQUIVALENT REST](#)

```
navyaannampelly@cluster-9185-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).
22/10/22 09:30:42 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/10/22 09:30:42 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/10/22 09:30:42 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
22/10/22 09:30:42 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
Welcome to
```



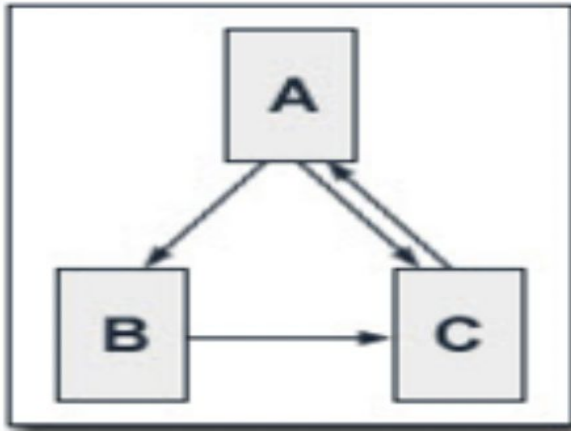
version 3.1.3

```
Using Python version 3.8.13 (default, Mar 25 2022 06:04:10)
Spark context Web UI available at http://cluster-9185-m.us-central1-a.c.mystical-accord-366002.internal:43851
Spark context available as 'sc' (master = yarn, app id = application_1666430372883_0001).
SparkSession available as 'spark'.
>>> █
```

# Manual Solution and Design

## Assuming

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

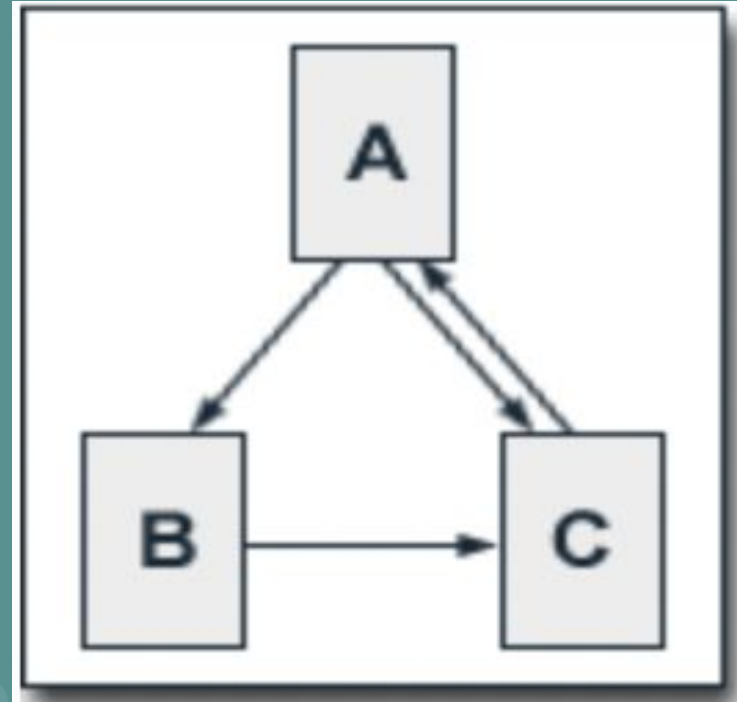




# Manual Solution and Design

## First Iteration:

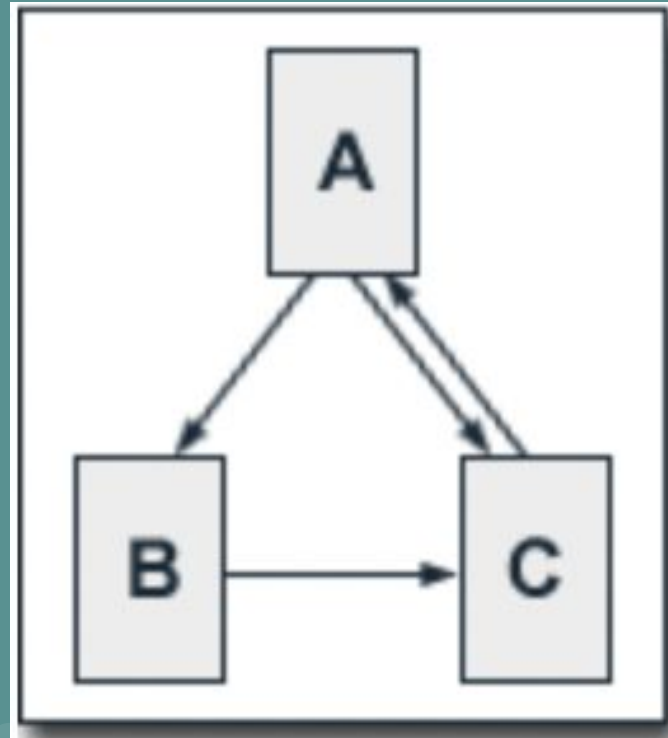
1. **PR(A)**  
 $= (1-d) + d * (PR(C) / 1)$   
 $= (1-0.85) + 0.85 * (1)$   
 $= 1$
2. **PR(B)**  
 $= (1-d) + d * (PR(A) / 2)$   
 $= (1-0.85) + 0.85 * 0.5$   
 $= 0.575$
3. **PR(C)**  
 $= (1-d) + d * (PR(A) / 2 + PR(B) / 1)$   
 $= (1-0.85) + 0.85 * (0.5 + 1)$   
 $= 1.425$



# Manual Solution and Design

## Second Iteration:

1.  $PR(A)$   
 $= 1 - 0.85 + 0.85 * 1.425$   
 $= 1.36125$
2.  $PR(B)$   
 $= 1 - 0.85 + 0.85 * 0.5$   
 $= 0.575$
3.  $PR(C)$   
 $= 1 - 0.85 + 0.85 * 1.075$   
 $= 1.06375$



# Implementation

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.

```
navyaannampelly@cluster-pagerank-m:~$ cat /usr/share/doc/*/copyright
BSOLUTELY NO WARRANTY, to the extent
```

```
Last login: Mon Oct 31 23:33:02 2022 from 35.235.244.34
```

```
nakhtar@cluster-page-rank-m:~$ vi pagerank.txt
```

```
nakhtar@cluster-page-rank-m:~$ cat pagerank.txt
```

```
A B
```

```
A C
```

```
B C
```

```
C A
```

---

# Implementation



SSH-in-browser

A B

A C

B C

C A

~

~

~

~

~

~

~

# PageRank + Scala + GCP

Create new directory on cluster i-e mydata:

```
hdfs dfs -mkdir hdfs:///mydata
```

```
navyaannampelly@cluster-pagerank-m:~$ hdfs dfs -mkdir hdfs:///mydata
```

Keep the pagerank.txt file into hdfs directory:

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

```
navyaannampelly@cluster-pagerank-m:~$ hdfs dfs -put pagerank_data.txt hdfs:///mydata
```

```
navyaannampelly@cluster-9185-m:~$ hdfs dfs -mkdir hdfs:///mydata
```

```
navyaannampelly@cluster-9185-m:~$ hdfs dfs -put ml-100k hdfs:///mydata
```

```
navyaannampelly@cluster-9185-m:~$ hdfs dfs -ls hdfs:///mydata
```

```
Found 1 items
```

```
drwxr-xr-x  - navyaannampelly hadoop          0 2022-10-22 09:28 hdfs:///mydata/ml-100k
```

```
navyaannampelly@cluster-9185-m:~$
```

```
navyaannampelly@cluster-9185-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).
```

```
22/10/22 09:30:42 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
```

```
22/10/22 09:30:42 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
```

```
22/10/22 09:30:42 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
```

```
22/10/22 09:30:42 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
```

```
Welcome to
```



```
version 3.1.3
```



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))
```



# Result

## First 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))
```

# Result

## Second 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.
```

# Result

## Third 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.
```

# Enhancement Ideas

- ❖ With  $n$  iterations, we may test the pagerank algorithm.
- ❖ We can determine the pagerank of large websites with numerous links.

# Conclusion

- ❖ The PageRank value of a page  $u$  is determined by the PageRank values of each page  $v$  in the set  $B_u$  (the set of all pages connecting to page  $u$ ), divided by the quantity  $L(v)$  of links from page  $v$ .
- ❖ For the algorithm to calculate the PageRank, a damping factor of 0.85 is used. It is comparable to the income tax that the government withholds from one even if it has already paid him.

# References

- <http://lintool.github.io/Cloud9/docs/content/pagerank.html>
- <http://people.revoledu.com/kardi/tutorial/PageRank/index.htm>
- <http://stat.rutgers.edu/home/tzhang/papers/nips07-ranking.pdf>
- <https://hc.labnet.sfbu.edu/~henry/npu/classes/mapreduce/week1/syllabus.html>



THANK  
YOU