# Page Rank

Submitted By: Norina Akhtar
Student Id : 19643

# Table Of Contents

Introduction

Explanation

Manual Implementation / Design

implementation using Pyspark

PageRank + Scala + GCP

Enhancement Ideas

Conclusion

# Introduction

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

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

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.

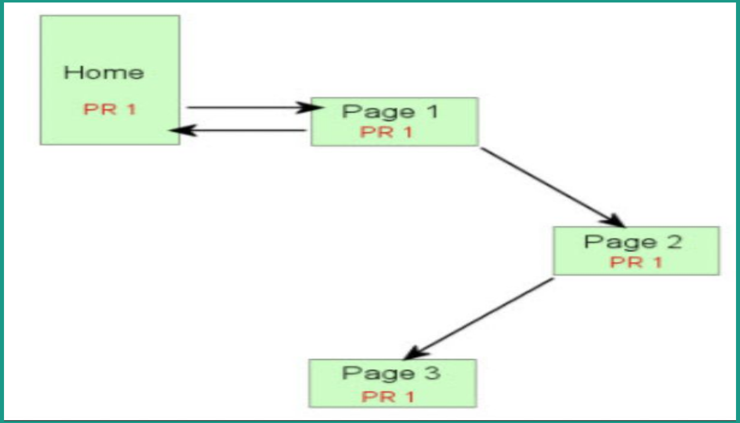
An iterative algorithm that performs many joins, so it is a good use case for RDD partitioning.

The algorithm maintains two datasets:

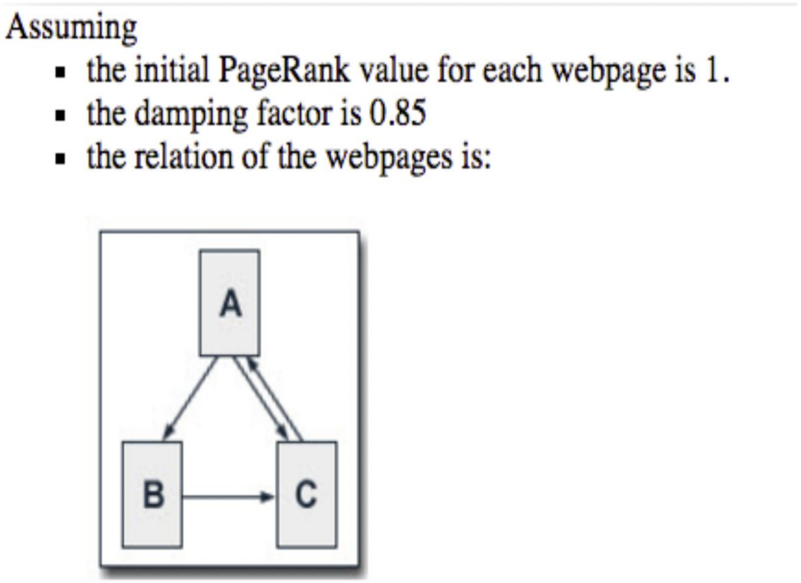
- (pageID, linked List) elements containing the list of neighbors of each page,
- (pageID, rank) elements containing the current rank for each page.

# Explanation

- We regard a small web consisting of three pages A, B, C, whereby page A links to the pages B and C , page B links to page C and page C links to Page A. According to Page and Brin , the damping factor d is usually set to 0.85.
- 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.



# Manual Implementation / Design



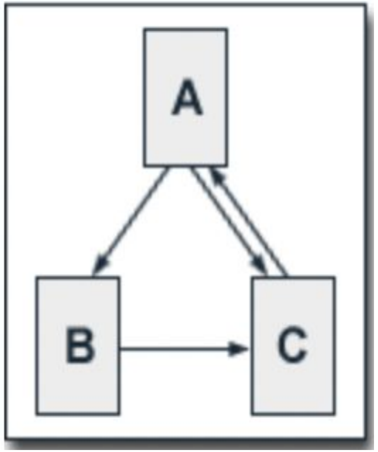
Assuming

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

# Manual Implementation / 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 Implementation

Second Iteration:

1. PR(A)
   = 1 – 0.85 + 0.85 * 1.425

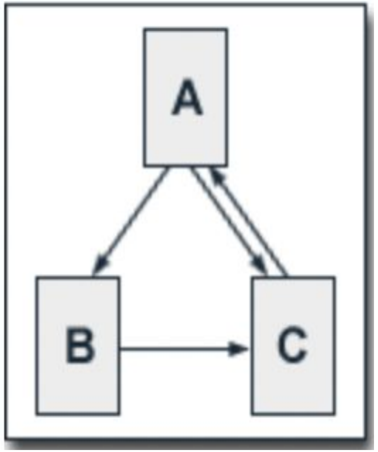
   = 1.36125

2. PR(B)
   = 1 – 0.85 + 0.85 * 0.5

   = 0.575

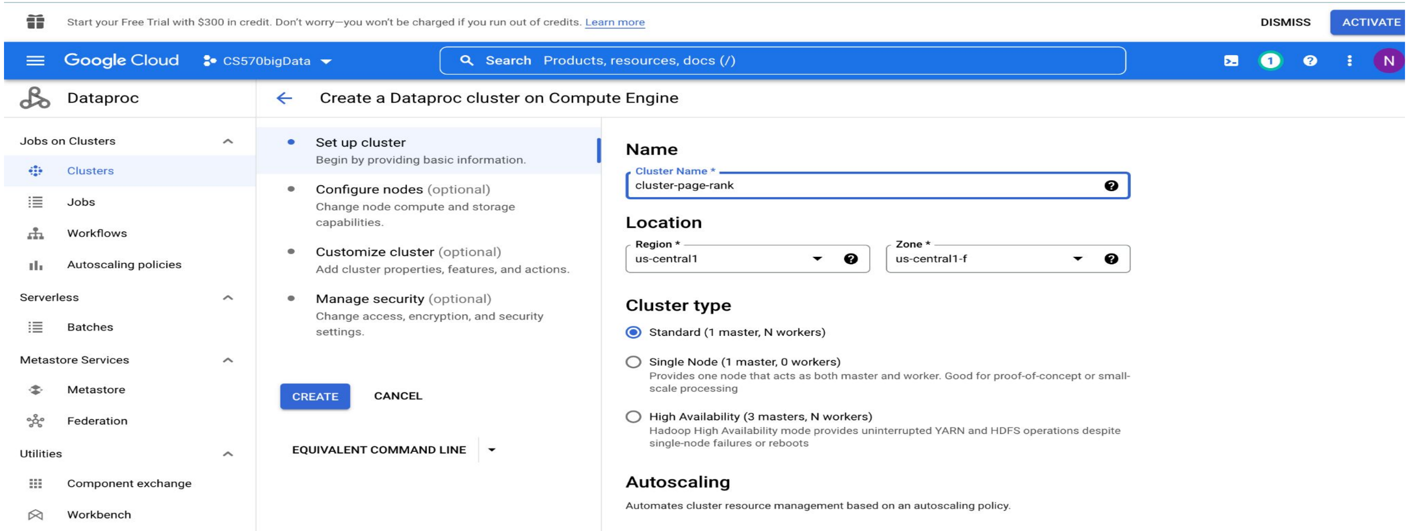
3. PR(C)
   1 – 0.85 + 0.85 * 1.075

   = 1.06375



# Implementation

Create cluster on GCP:



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 *: cluster-page-rank

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

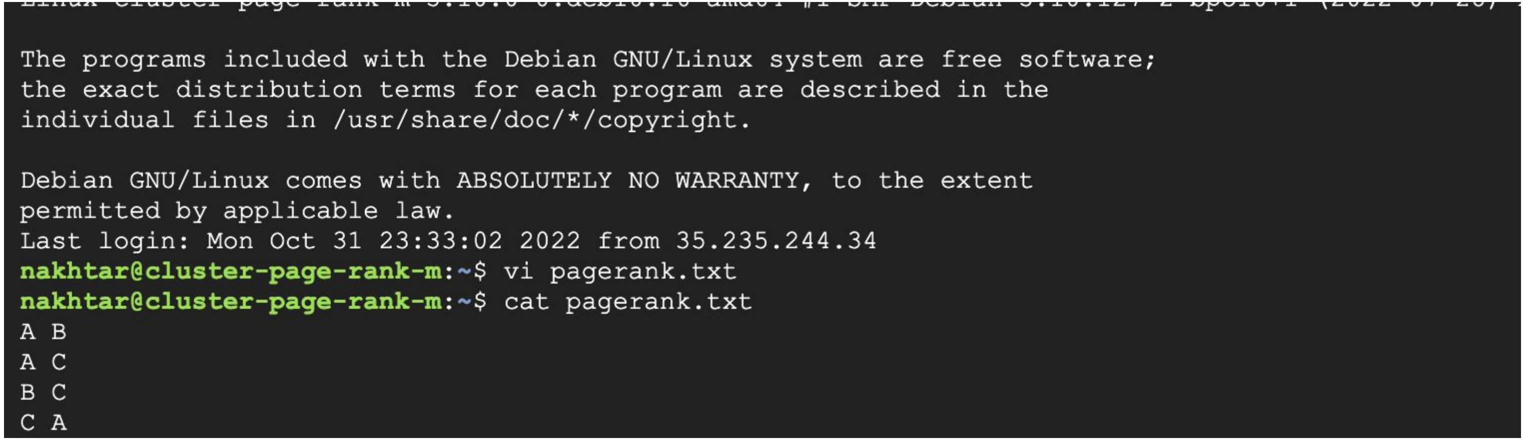
○ High Availability (3 masters, N workers)
Hadoop High Availability mode provides uninterrupted YARN and HDFS operations despite single-node failures or reboots

Autoscaling

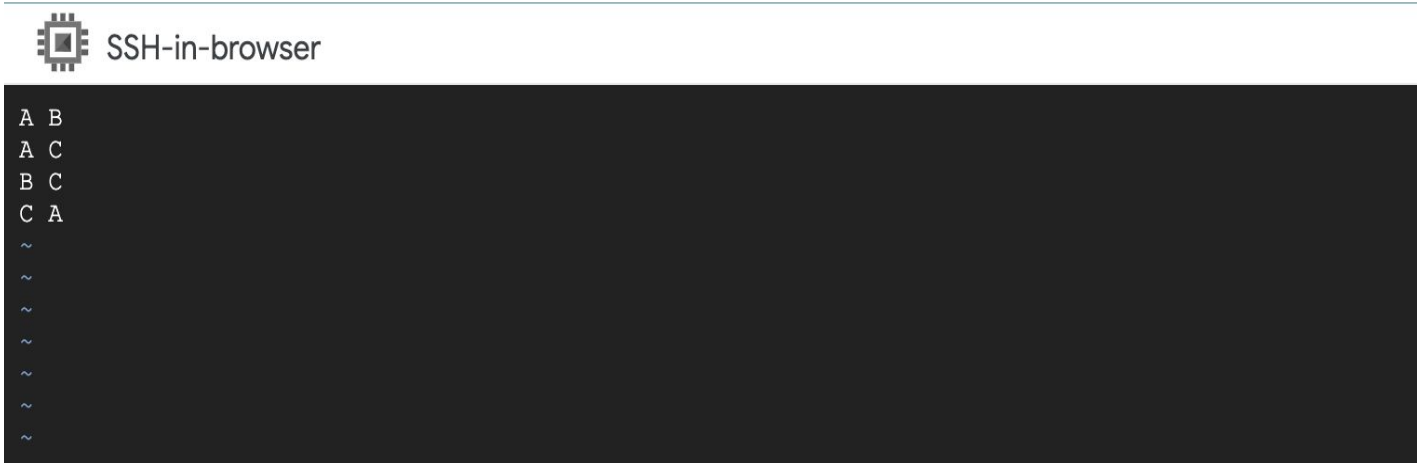
Automates cluster resource management based on an autoscaling policy.

# Implementation using Pyspark

Create new file:
 vi pagerank.txt



```
Linux cluster-page-rank-m 5.10.0-0.deb10.16-amd64 #1 SMP Debian 5.10.127-2~bpo10+1 (2022-07-28)

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
nakhtar@cluster-page-rank-m:~$ vi pagerank.txt
nakhtar@cluster-page-rank-m:~$ cat pagerank.txt
A B
A C
B C
C A
```

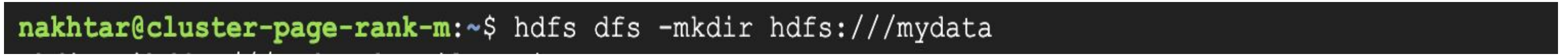


```
A B
A C
B C
C A
~
~
~
~
~
~
~
```

# Implementation

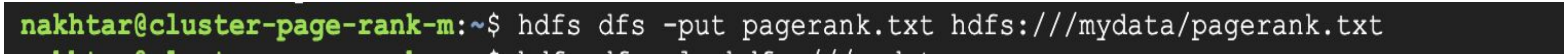
### Create new directory on cluster i-e mydata:

hdfs dfs -mkdir hdfs:///mydata

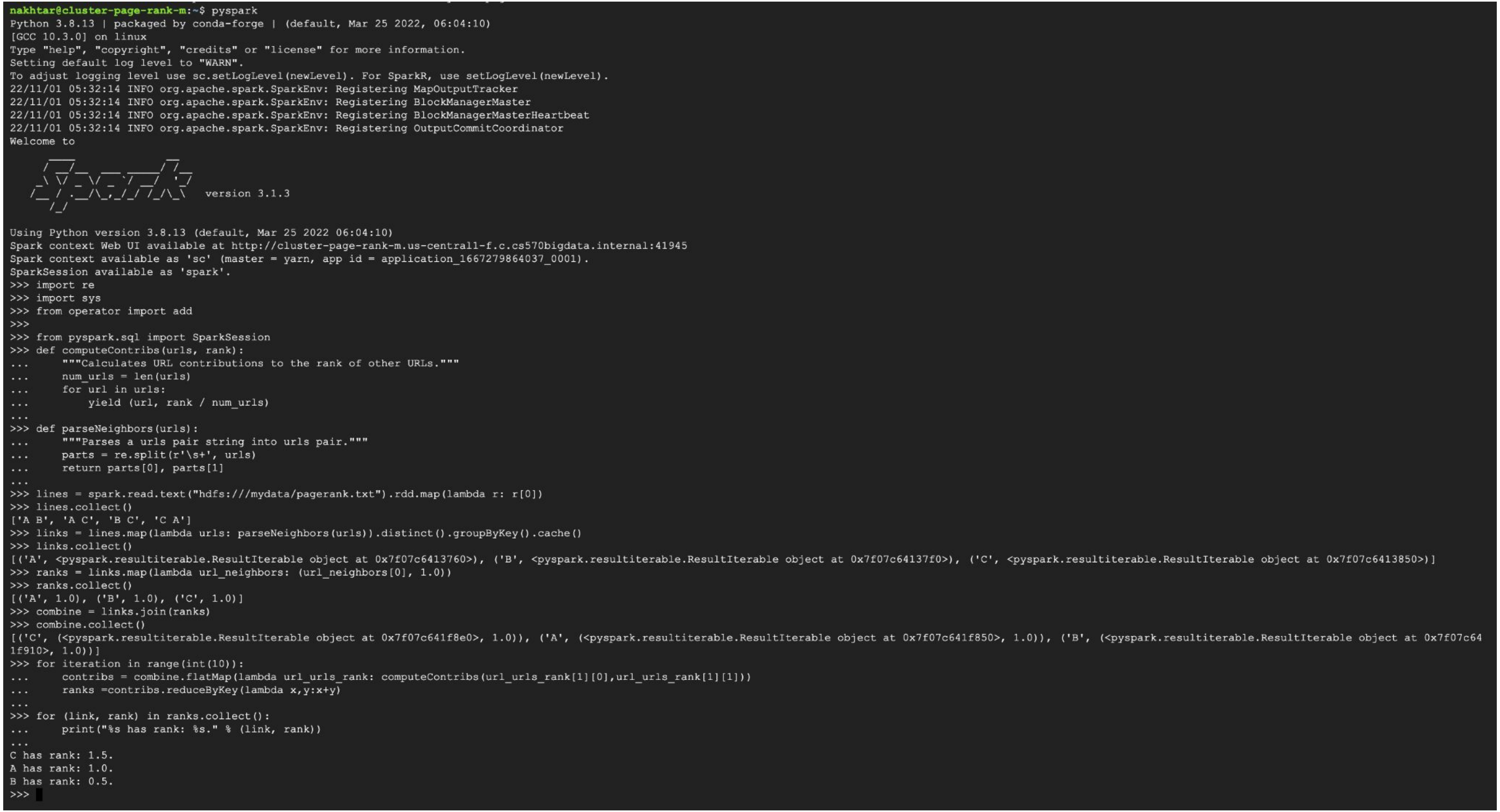


```
nakhtar@cluster-page-rank-m:~$ hdfs dfs -mkdir hdfs:///mydata
```

Put pagerank.txt file into hdfs directory:



```
nakhtar@cluster-page-rank-m:~$ hdfs dfs -put pagerank.txt hdfs:///mydata/pagerank.txt
```



```
nakhtar@cluster-page-rank-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).
22/11/01 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.
>>>
```

# PageRank + Scala + GCP

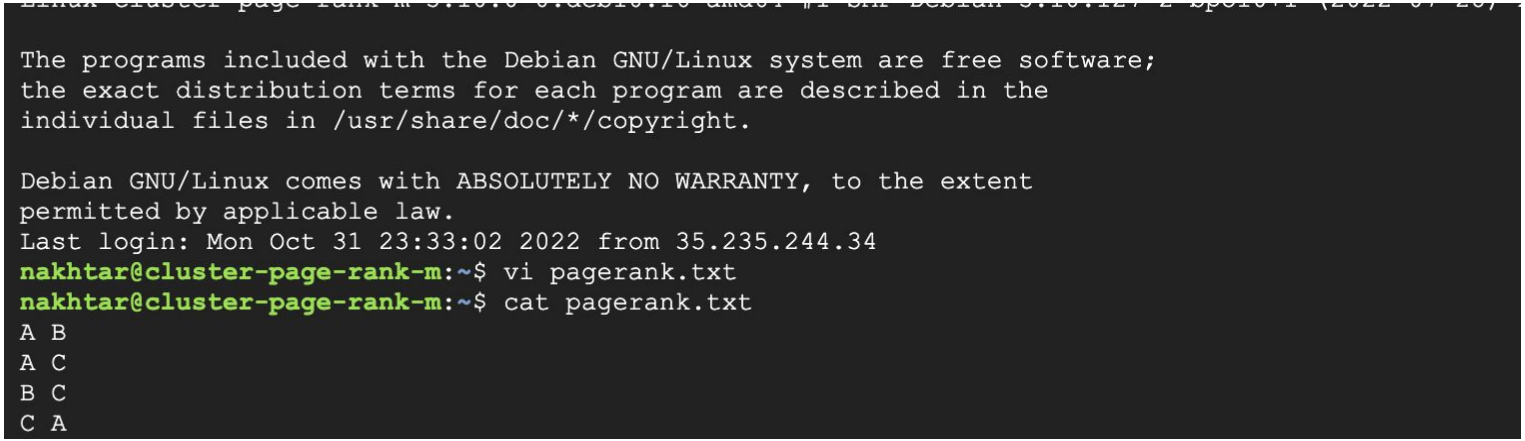
**Set up Scala on GCP:**

- Create cluster
- Install scala using these commands

```
 $ curl -fL https://github.com/coursier/launchers/raw/master/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/
```

# PageRank + Scala + GCP

Create pagerank.txt file to store input data

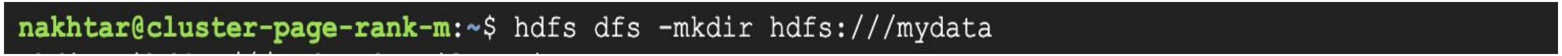


```
Linux cluster-page-rank-m 5.10.0-0.deb10.16-amd64 #1 SMP Debian 5.10.127-2~bpo10+1 (2022-07-28)

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
nakhtar@cluster-page-rank-m:~$ vi pagerank.txt
nakhtar@cluster-page-rank-m:~$ cat pagerank.txt
A B
A C
B C
C A
```

# PageRank + Scala + GCP

### Create new directory on cluster i-e mydata:

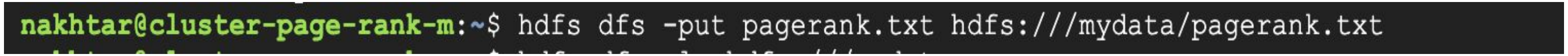
hdfs dfs -mkdir hdfs:///mydata



```
nakhtar@cluster-page-rank-m:~$ hdfs dfs -mkdir hdfs:///mydata
```

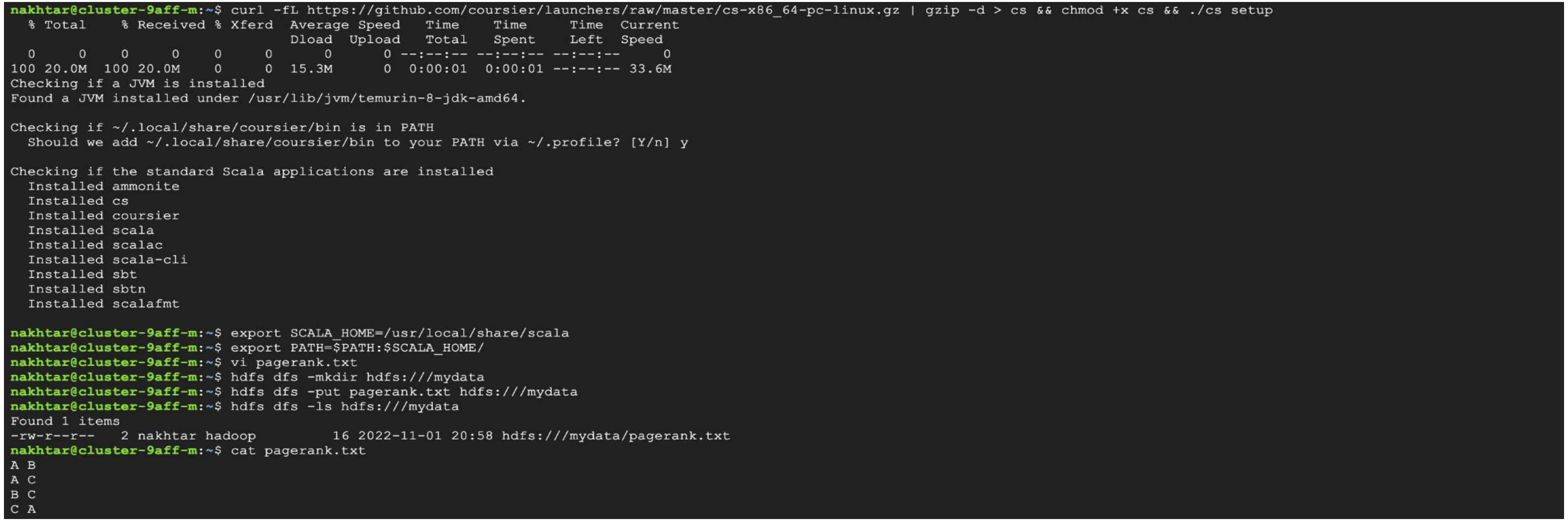
Put pagerank.txt file into hdfs directory:

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

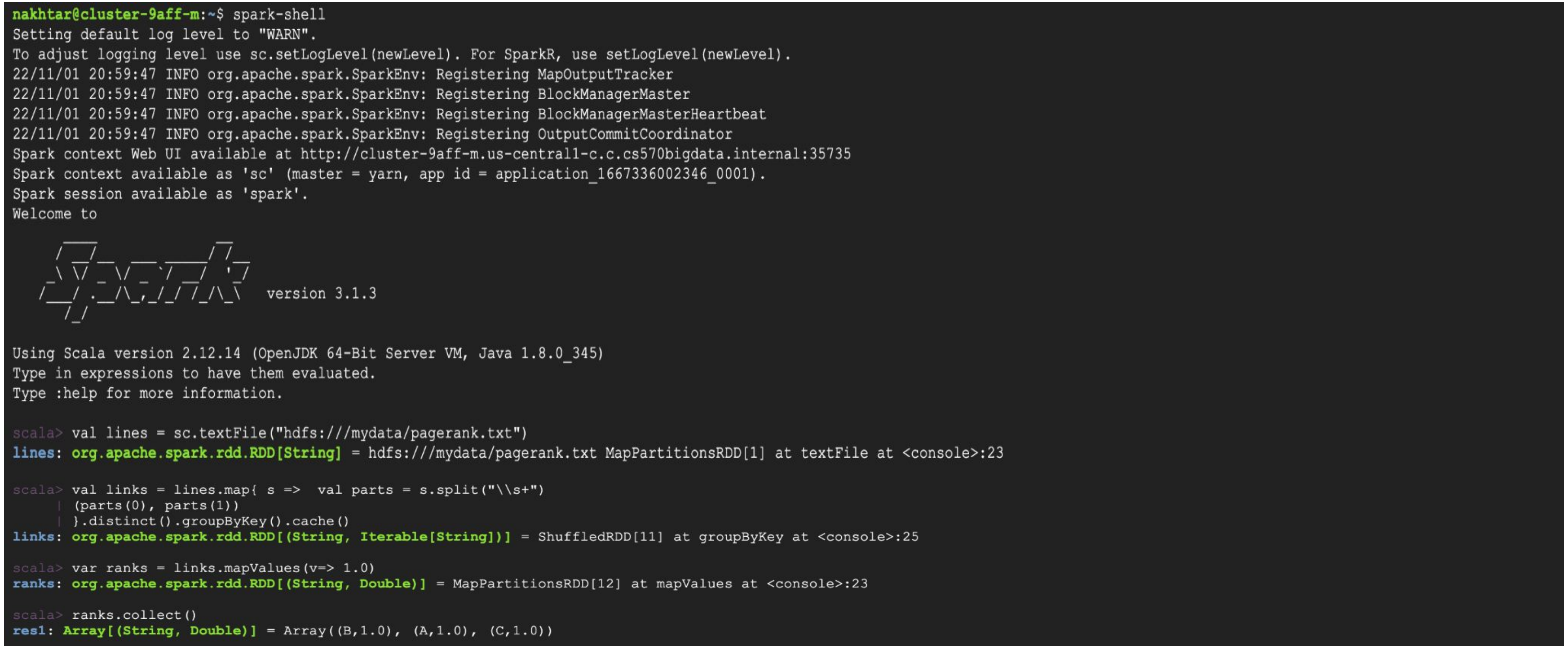


```
nakhtar@cluster-page-rank-m:~$ hdfs dfs -put pagerank.txt hdfs:///mydata/pagerank.txt
```

# Implementation using Scala



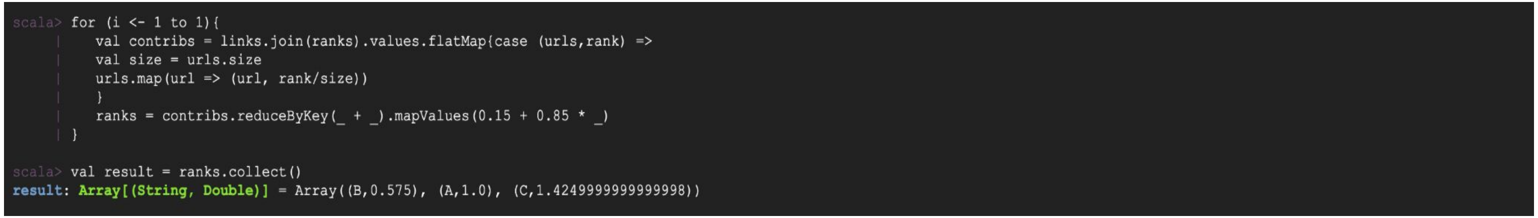
```
nakhtar@cluster-9aff-m:~$ curl -fL https://github.com/coursier/launchers/raw/master/cs-x86_64-pc-linux.gz | gzip -d > cs && chmod +x cs && ./cs setup
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 20.0M  100 20.0M    0     0  15.3M      0  0:00:01  0:00:01 --:--:-- 33.6M
Checking if a JVM is installed
Found a JVM installed under /usr/lib/jvm/temurin-8-jdk-amd64.

Checking if ~/.local/share/coursier/bin is in PATH
  Should we add ~/.local/share/coursier/bin to your PATH via ~/.profile? [Y/n] y

Checking if the standard Scala applications are installed
  Installed ammonite
  Installed cs
  Installed coursier
  Installed scala
  Installed scalac
  Installed scala-cli
  Installed sbt
  Installed sbtn
  Installed scalafmt

nakhtar@cluster-9aff-m:~$ export SCALA_HOME=/usr/local/share/scala
nakhtar@cluster-9aff-m:~$ export PATH=$PATH:$SCALA_HOME/
nakhtar@cluster-9aff-m:~$ vi pagerank.txt
nakhtar@cluster-9aff-m:~$ hdfs dfs -mkdir hdfs:///mydata
nakhtar@cluster-9aff-m:~$ hdfs dfs -put pagerank.txt hdfs:///mydata
nakhtar@cluster-9aff-m:~$ hdfs dfs -ls hdfs:///mydata
Found 1 items
-rw-r--r--   2 nakhtar hadoop         16 2022-11-01 20:58 hdfs:///mydata/pagerank.txt
nakhtar@cluster-9aff-m:~$ cat pagerank.txt
A B
A C
B C
C A
```



```
nakhtar@cluster-9aff-m:~$ spark-shell
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))
```

# Result

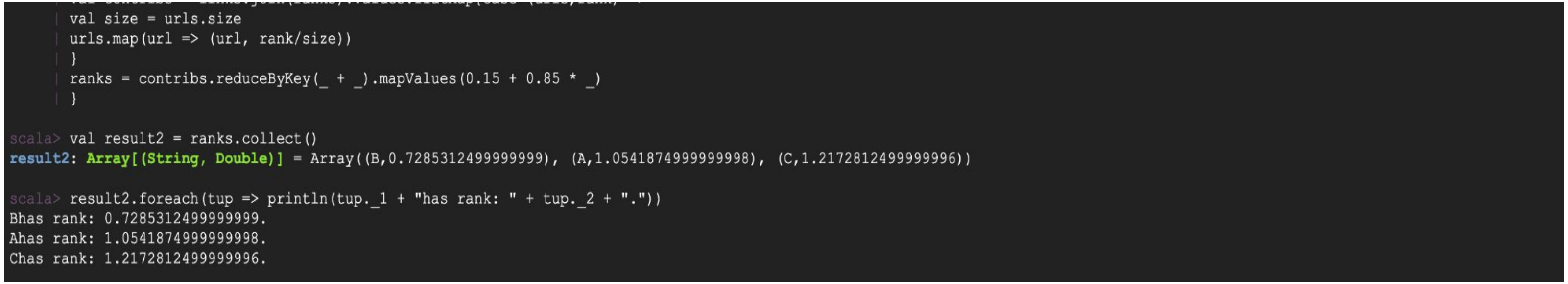
Ist Iteration:



```
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

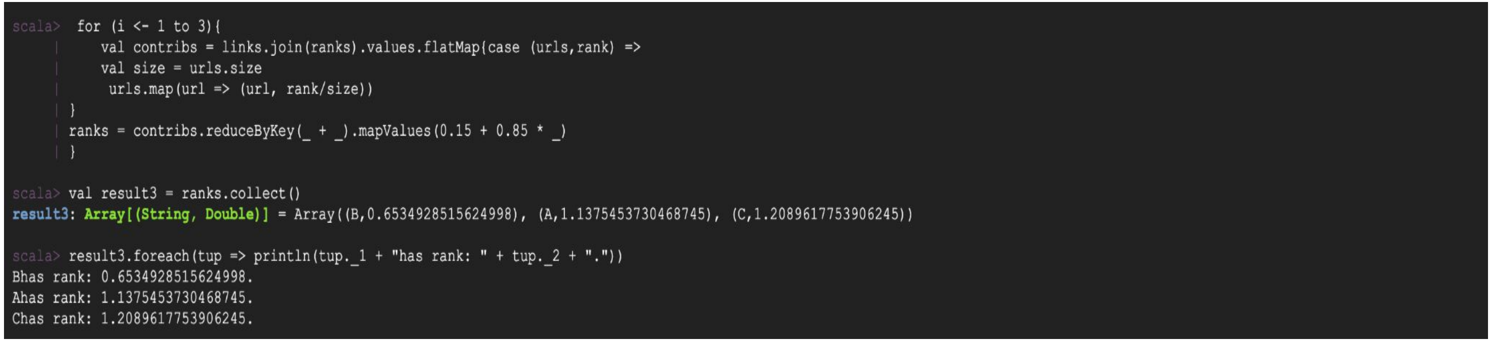
Second Iteration



```
       | 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

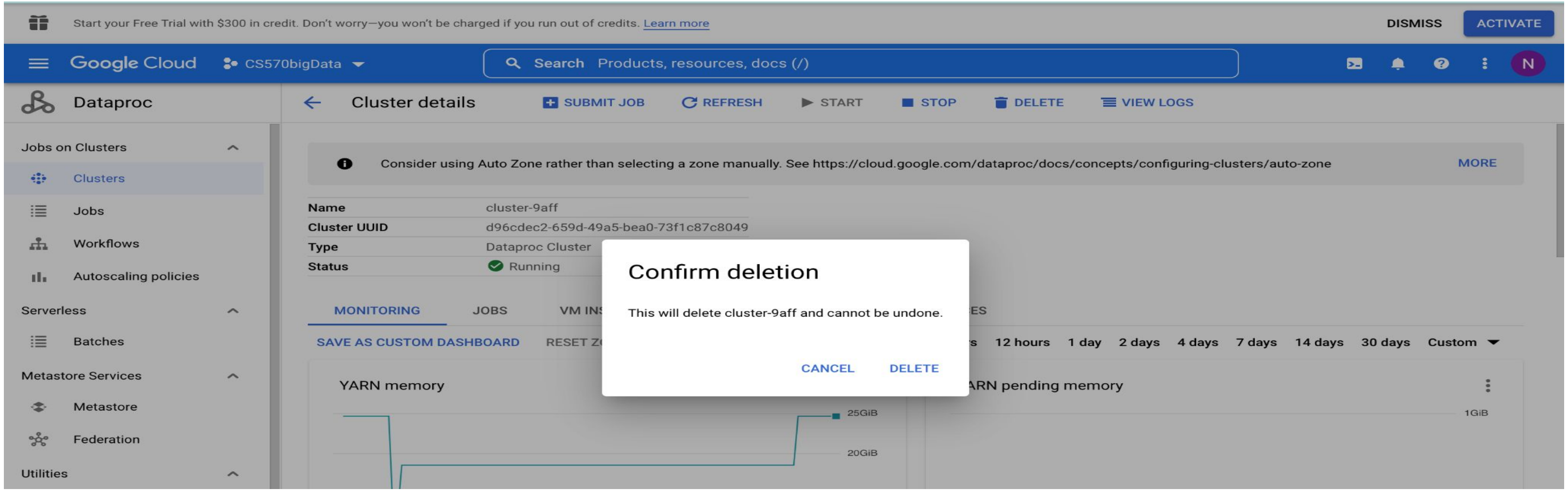


```
scala>  for (i <- 1 to 3){
      |     val contribs = links.join(ranks).values.flatMap{case (urls,rank) =>
      |     val size = urls.size
      |      urls.map(url => (url, rank/size))
      | }
      | ranks = contribs.reduceByKey(_ + _).mapValues(0.15 + 0.85 * _)
      | }

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.
```

So on…..

We can do it for n number iterations using for loop

# Shut down the Cluster



Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more

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

| | |
|---|---|
| Name | cluster-9aff |
| Cluster UUID | d96cdec2-659d-49a5-bea0-73f1c87c8049 |
| Type | Dataproc Cluster |
| Status | Running |

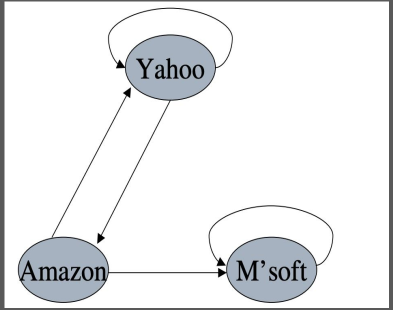
Confirm deletion

This will delete cluster-9aff and cannot be undone.

CANCEL DELETE

# Enhancement Ideas

- We can calculate  pagerank of big websites which have multiple links
- Compare scala and python performance
- We can test the the pagerank algorithm using n number of iteration



# Conclusion

Hence pagerank is expressed as :

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

The PageRank value for a page u is dependent on the PageRank values for each page v contained in the set Bu (the set containing all pages linking to page u), divided by the number L(v) of links from page v. The algorithm involves a damping factor(0.85) for the calculation of the PageRank. It is like the income tax which the govt extracts from one despite paying him itself

# References

- PageRank and design patterns for efficient graph algorithms
- Page Rank Tutorial
- A General Boosting Method and its Application to Learning Ranking Functions for Web
- Raise Your Google Ranking