PageRank on GCP

CONTENTS OF THIS TEMPLATE

Table of contents

01

Introduction

PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results.
It works by evaluating both the quantity and quality of links to a page.

This recursive nature of ranking forms the basis of PageRank, which considers not just the number of incoming links but also the importance of the pages providing these links, creating a reliable gauge of a page's relevance and authority on the web.

Design

02

Technology used for this project and Sections are:

GCP- Pyspark

GCP -Scala

03

Implementation

Three subsections involved.

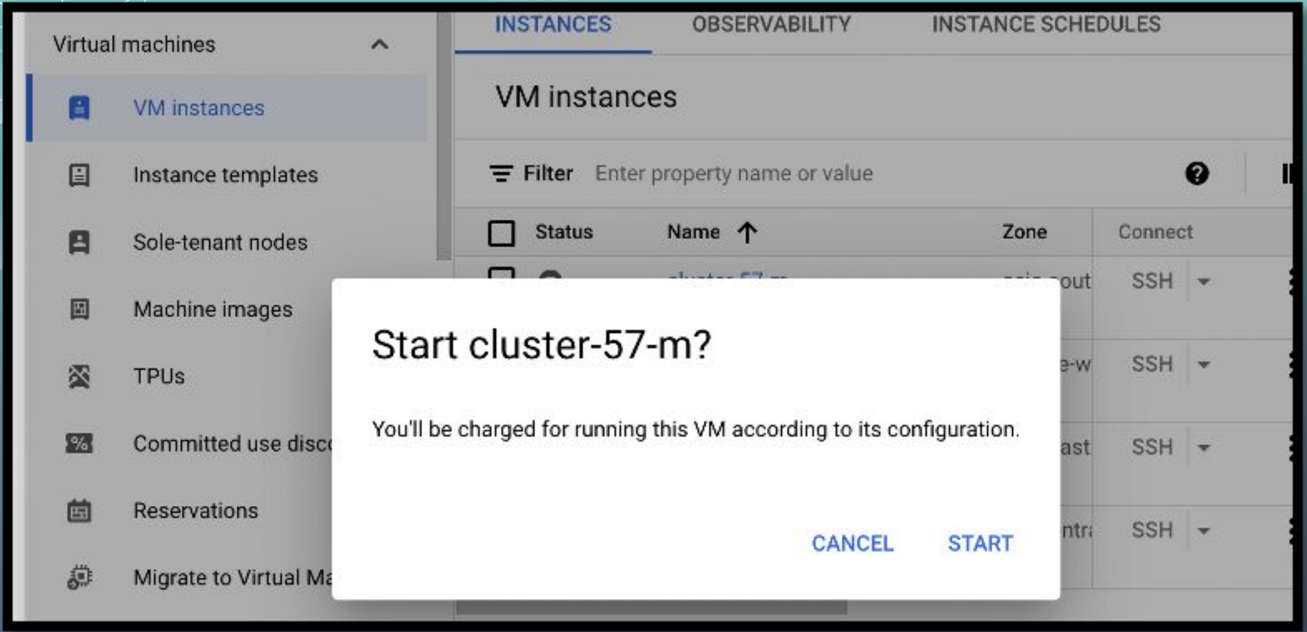
STEP 1: PySpark On GCP

Steps of implementing PageRank with Pyspark on GCP:

- Set up environment(cluster) on GCP for Pyspark
- Create input data file for this problem
- Create Pyspark program for PageRank

Note: To run the project, need to copy the data file to HDFS file system.

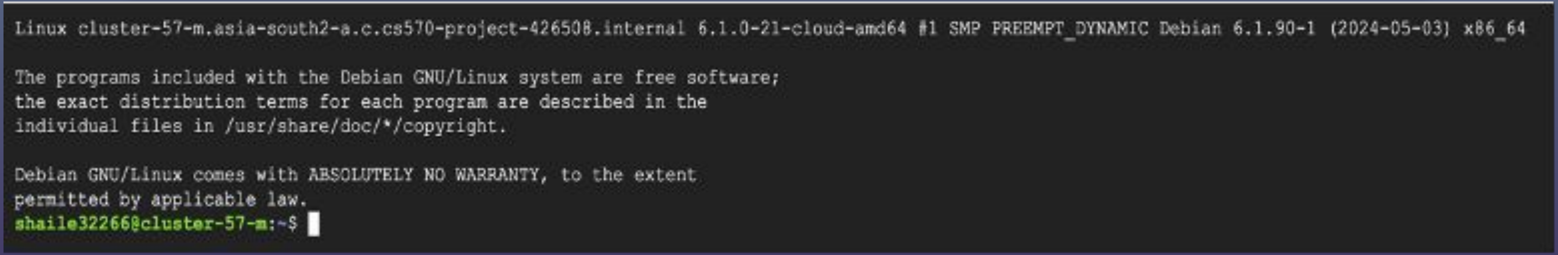
I. Setup PySpark Cluster



Start cluster-57-m?

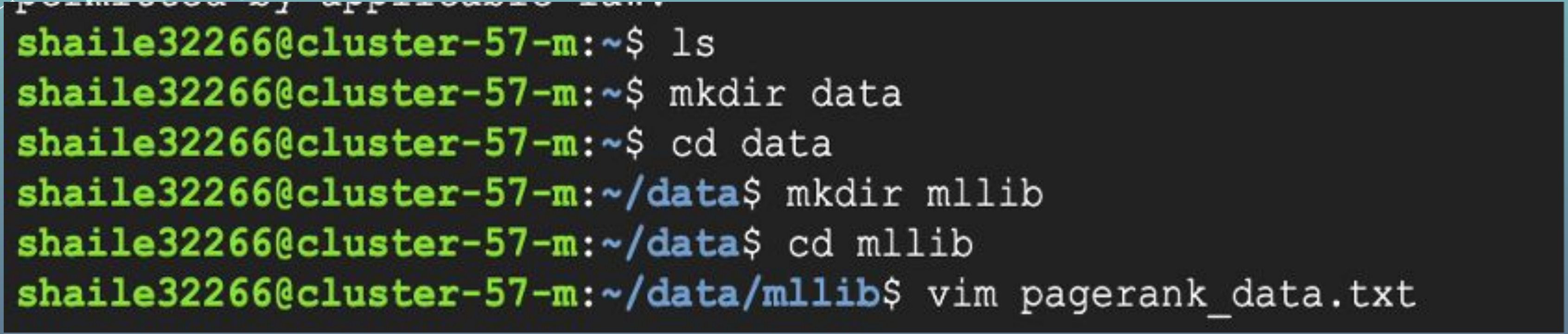
You'll be charged for running this VM according to its configuration.

CANCEL START

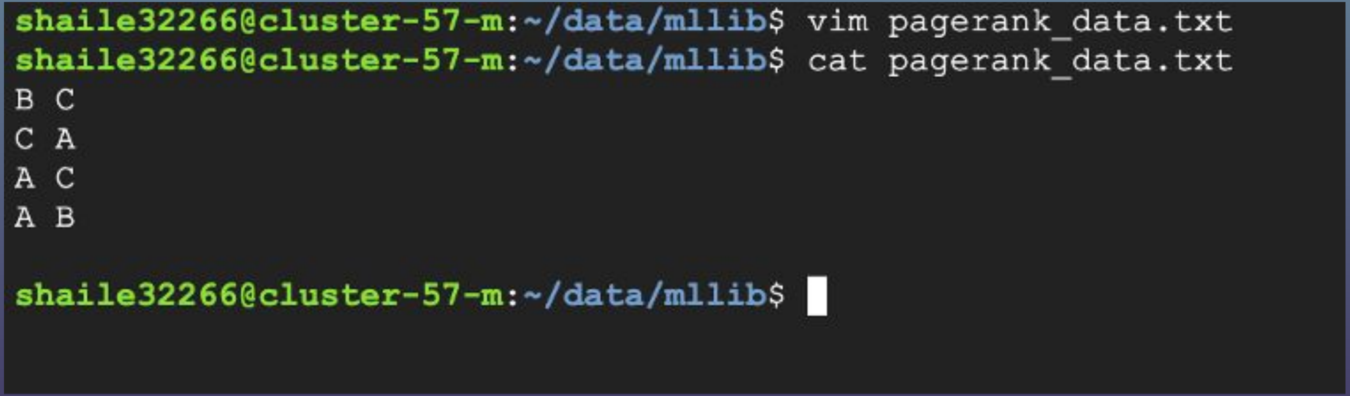


```
Linux cluster-57-m.asia-south2-a.c.cs570-project-426508.internal 6.1.0-21-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.90-1 (2024-05-03) x86_64

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.
shaile32266@cluster-57-m:~$
```

2. Create Input File

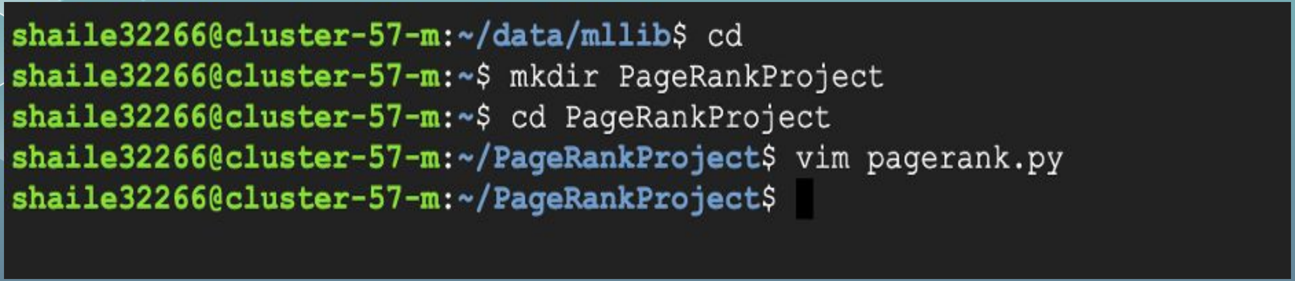


```
permitted by applicable law.
shaile32266@cluster-57-m:~$ ls
shaile32266@cluster-57-m:~$ mkdir data
shaile32266@cluster-57-m:~$ cd data
shaile32266@cluster-57-m:~/data$ mkdir mllib
shaile32266@cluster-57-m:~/data$ cd mllib
shaile32266@cluster-57-m:~/data/mllib$ vim pagerank_data.txt
```

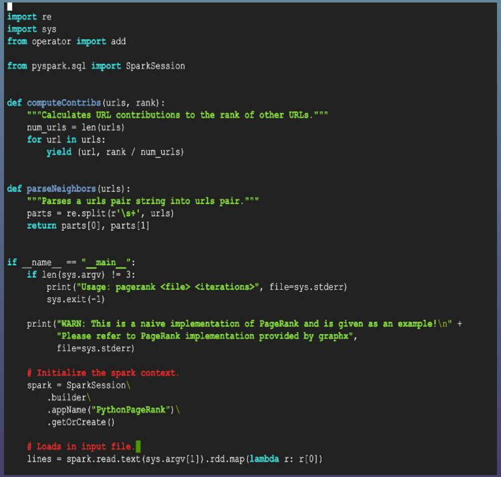


```
shaile32266@cluster-57-m:~/data/mllib$ vim pagerank_data.txt
shaile32266@cluster-57-m:~/data/mllib$ cat pagerank_data.txt
B C
C A
A C
A B

shaile32266@cluster-57-m:~/data/mllib$ 
```

2. Create PageRank PySpark Code



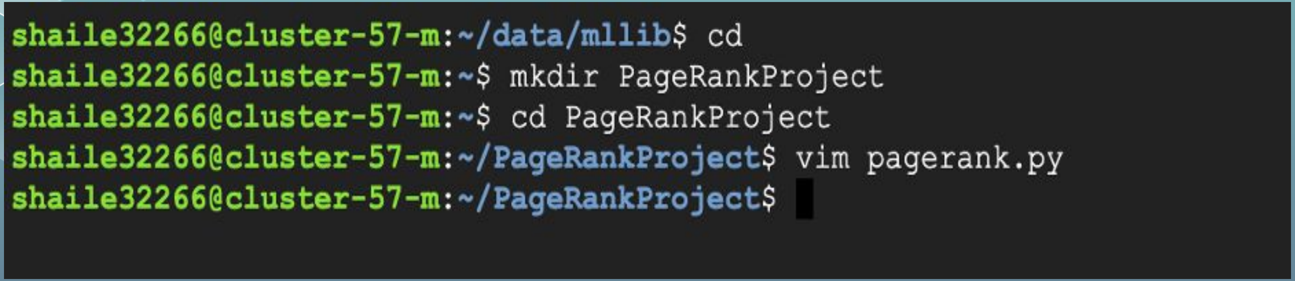
```
shaile32266@cluster-57-m:~/data/mllib$ cd
shaile32266@cluster-57-m:~$ mkdir PageRankProject
shaile32266@cluster-57-m:~$ cd PageRankProject
shaile32266@cluster-57-m:~/PageRankProject$ vim pagerank.py
shaile32266@cluster-57-m:~/PageRankProject$
```



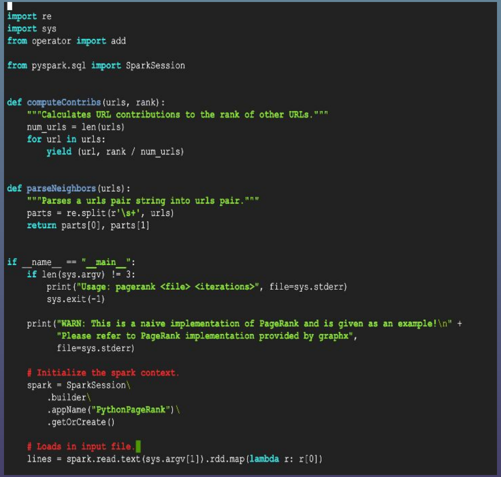
```python
import re
import sys
from operator import add

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()

    # Loads in input file.
    lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])
```

For a clear visual visit Github repository provided at the end of this presentation.

3. Create PageRank PySpark Code



```
shaile32266@cluster-57-m:~/data/mllib$ cd
shaile32266@cluster-57-m:~$ mkdir PageRankProject
shaile32266@cluster-57-m:~$ cd PageRankProject
shaile32266@cluster-57-m:~/PageRankProject$ vim pagerank.py
shaile32266@cluster-57-m:~/PageRankProject$
```



```
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()

    # Loads in input file.
    lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])
```

For a clear visual visit Github repository provided at the end of this presentation.

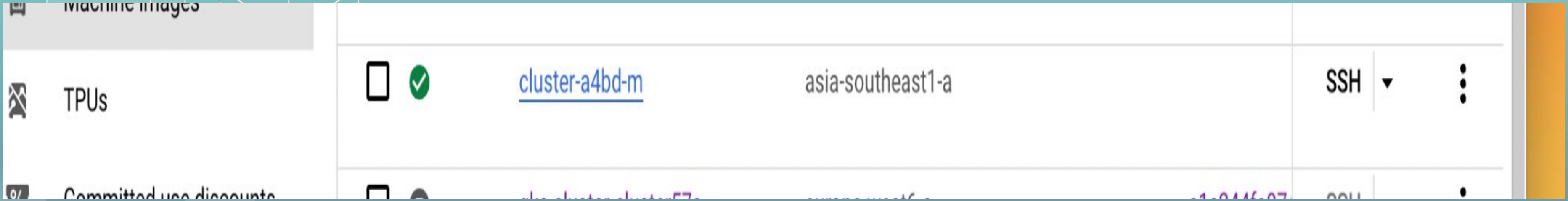
STEP 2: Scala On GCP

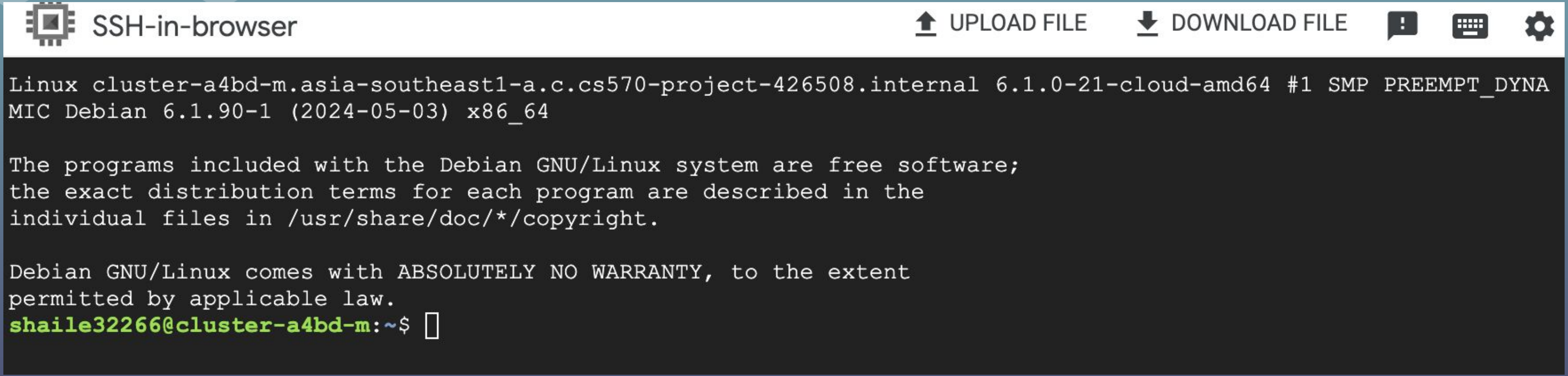
Steps of implementing PageRank with Scala on GCP:

- Set up environment(cluster) on GCP for Scala
- Create input data file for this problem
- Create Scala program for PageRank

Note: To run the project, need to copy the data file to HDFS file system.

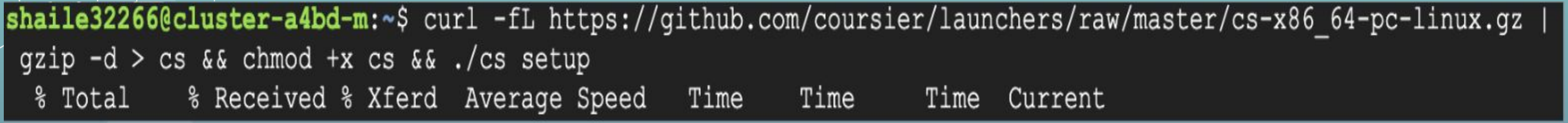
I. Setup Scala Cluster



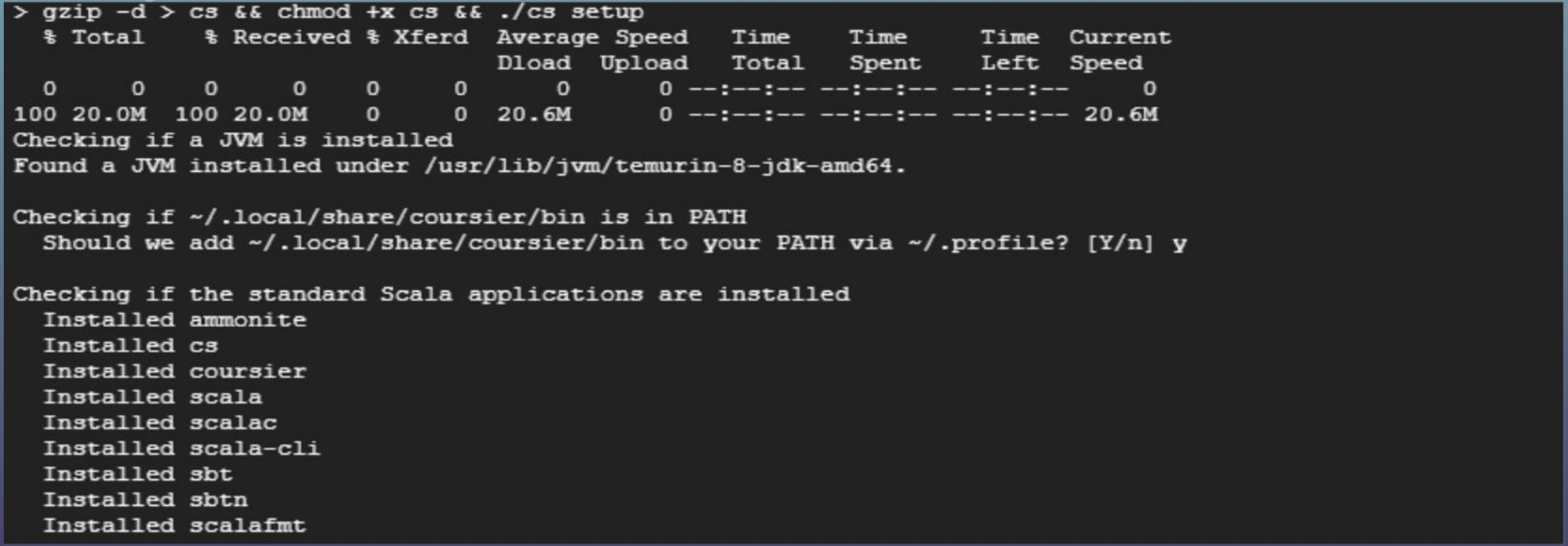


```
Linux cluster-a4bd-m.asia-southeast1-a.c.cs570-project-426508.internal 6.1.0-21-cloud-amd64 #1 SMP PREEMPT_DYNA
MIC Debian 6.1.90-1 (2024-05-03) x86_64

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.
shaile32266@cluster-a4bd-m:~$
```

2. Install Coursier Launcher for Linux

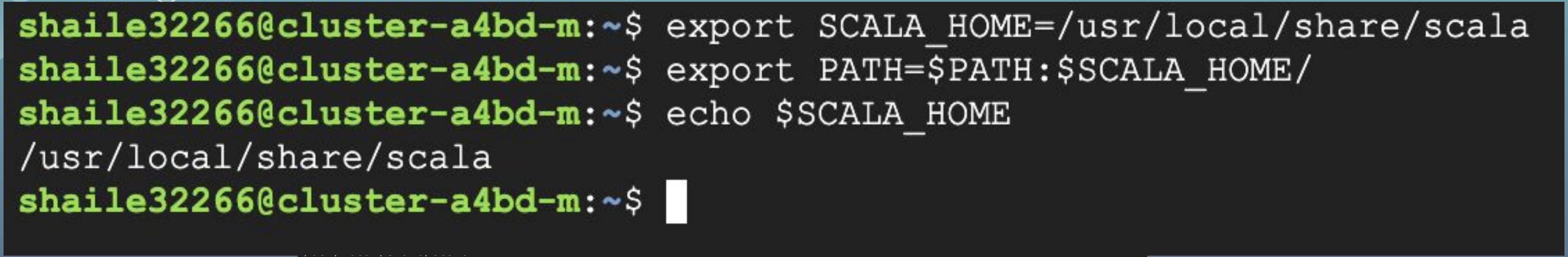


```
shaile32266@cluster-a4bd-m:~$ curl -fL https://github.com/coursier/launchers/raw/master/cs-x86_64-pc-linux.gz |
 gzip -d > cs && chmod +x cs && ./cs setup
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
```

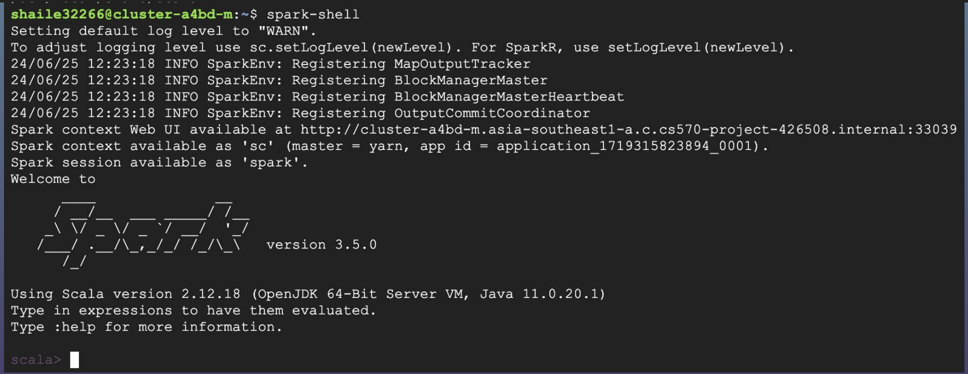


```
> gzip -d > cs && chmod +x cs && ./cs setup
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 20.0M  100 20.0M    0     0  20.6M      0 --:--:-- --:--:-- --:--:-- 20.6M
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
```

3. Set SCALA_HOME and update PATH for Scala

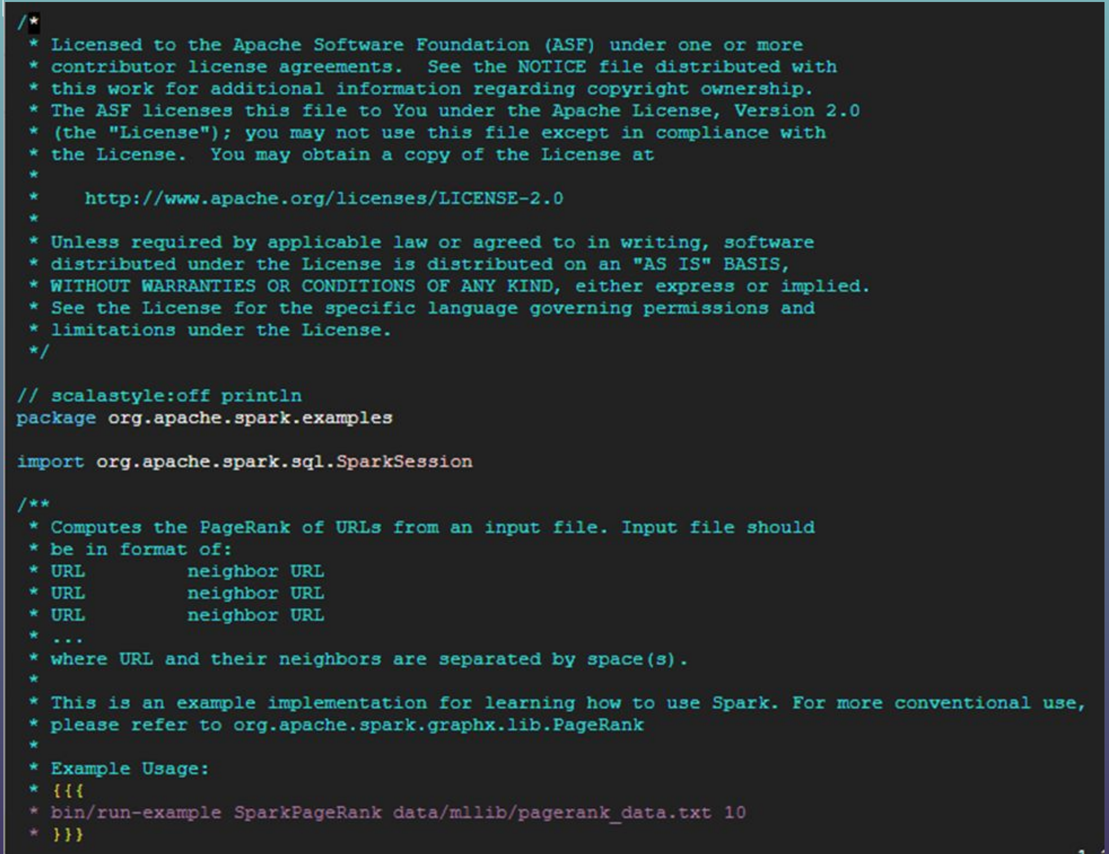


```
shaile32266@cluster-a4bd-m:~$ export SCALA_HOME=/usr/local/share/scala
shaile32266@cluster-a4bd-m:~$ export PATH=$PATH:$SCALA_HOME/
shaile32266@cluster-a4bd-m:~$ echo $SCALA_HOME
/usr/local/share/scala
shaile32266@cluster-a4bd-m:~$
```



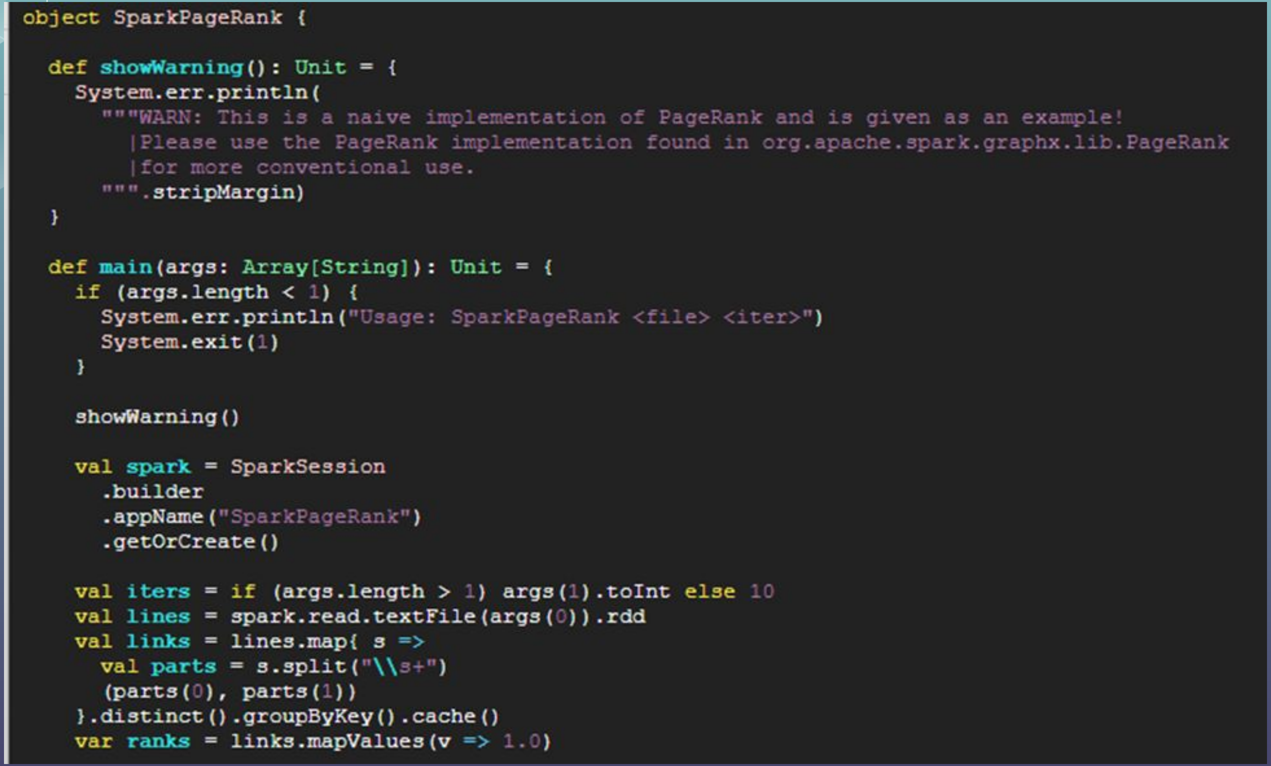
```
shaile32266@cluster-a4bd-m:~$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/06/25 12:23:18 INFO SparkEnv: Registering MapOutputTracker
24/06/25 12:23:18 INFO SparkEnv: Registering BlockManagerMaster
24/06/25 12:23:18 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/06/25 12:23:18 INFO SparkEnv: Registering OutputCommitCoordinator
Spark context Web UI available at http://cluster-a4bd-m.asia-southeast1-a.c.cs570-project-426508.internal:33039
Spark context available as 'sc' (master = yarn, app id = application_1719315823894_0001).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.0
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 11.0.20.1)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Open sparkshell and copy paste this code



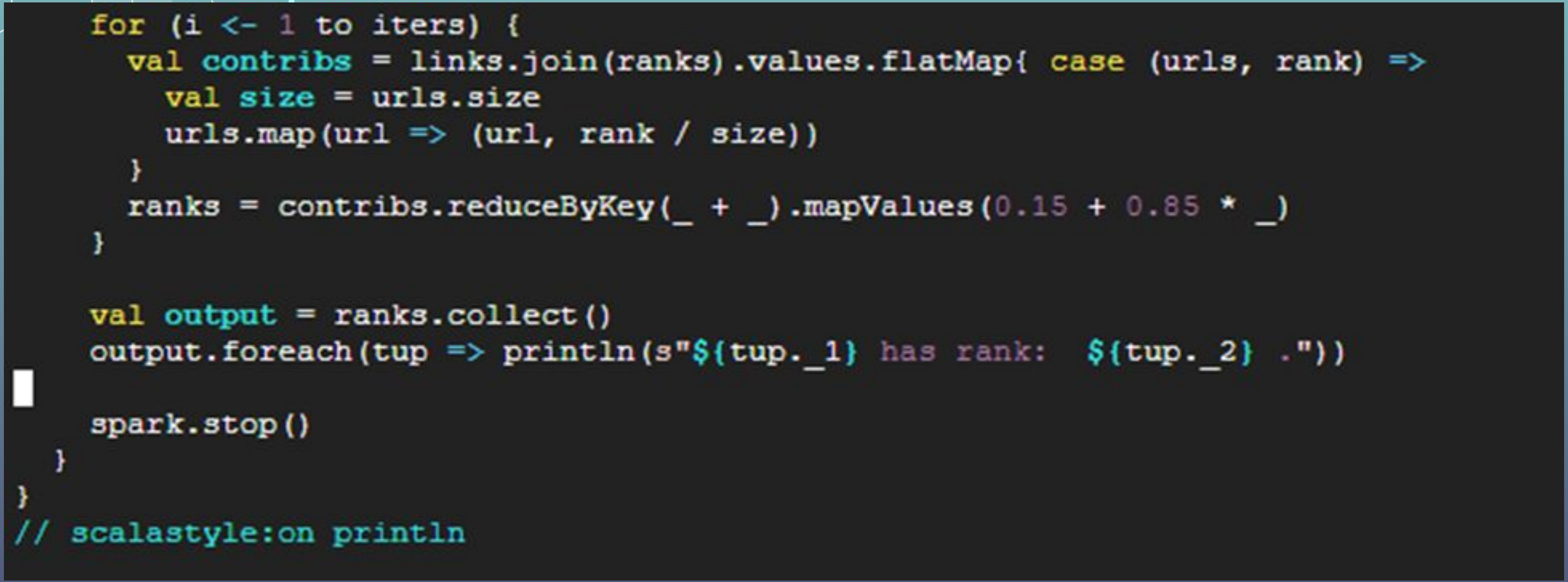
```
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *    http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

// scalastyle:off println
package org.apache.spark.examples

import org.apache.spark.sql.SparkSession

/**
 * Computes the PageRank of URLs from an input file. Input file should
 * be in format of:
 * URL         neighbor URL
 * URL         neighbor URL
 * URL         neighbor URL
 * ...
 * where URL and their neighbors are separated by space(s).
 *
 * This is an example implementation for learning how to use Spark. For more conventional use,
 * please refer to org.apache.spark.graphx.lib.PageRank
 *
 * Example Usage:
 * {{{
 * bin/run-example SparkPageRank data/mllib/pagerank_data.txt 10
 * }}}
```

Scala Code



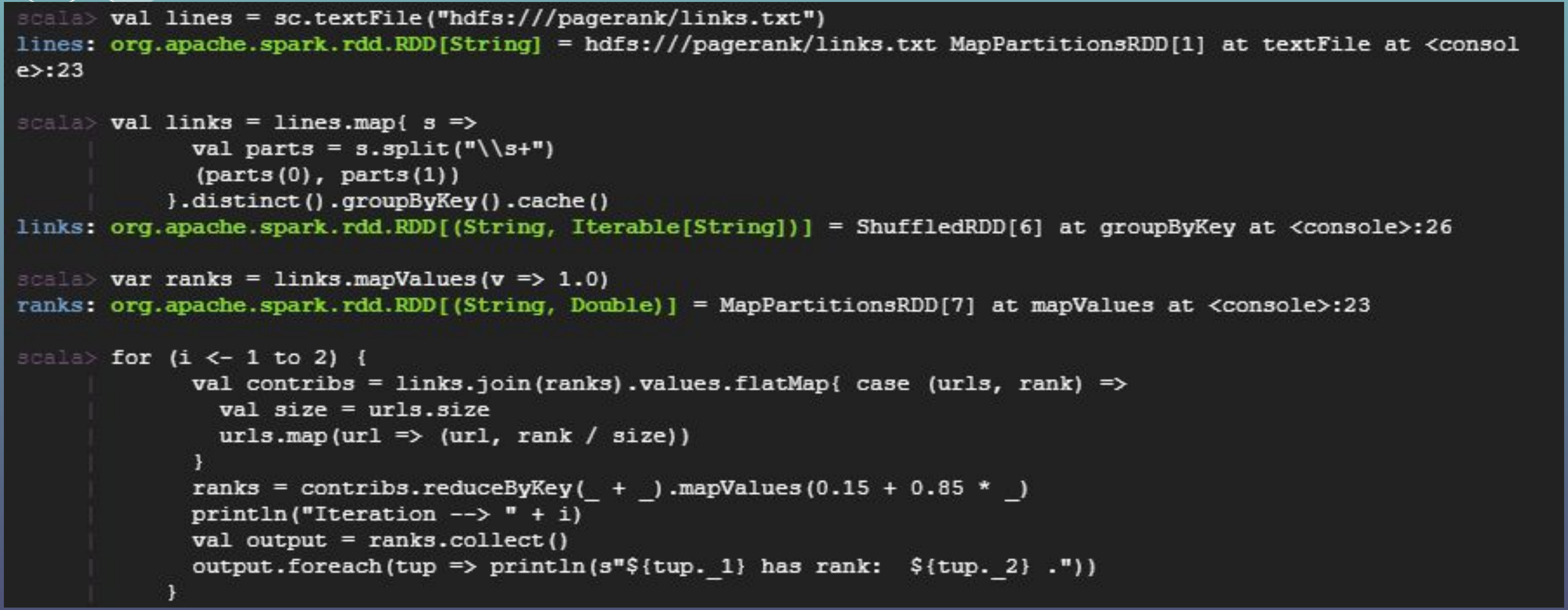
```scala
object SparkPageRank {

  def showWarning(): Unit = {
    System.err.println(
      """WARN: This is a naive implementation of PageRank and is given as an example!
        |Please use the PageRank implementation found in org.apache.spark.graphx.lib.PageRank
        |for more conventional use.
      """.stripMargin)
  }

  def main(args: Array[String]): Unit = {
    if (args.length < 1) {
      System.err.println("Usage: SparkPageRank <file> <iter>")
      System.exit(1)
    }

    showWarning()

    val spark = SparkSession
      .builder
      .appName("SparkPageRank")
      .getOrCreate()

    val iters = if (args.length > 1) args(1).toInt else 10
    val lines = spark.read.textFile(args(0)).rdd
    val links = lines.map{ s =>
      val parts = s.split("\\s+")
      (parts(0), parts(1))
    }.distinct().groupByKey().cache()
    var ranks = links.mapValues(v => 1.0)
```

Scala Code



```scala
    for (i <- 1 to iters) {
      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(s"${tup._1} has rank:  ${tup._2} ."))

    spark.stop()
  }
}
// scalastyle:on println
```

Improved Scala Code



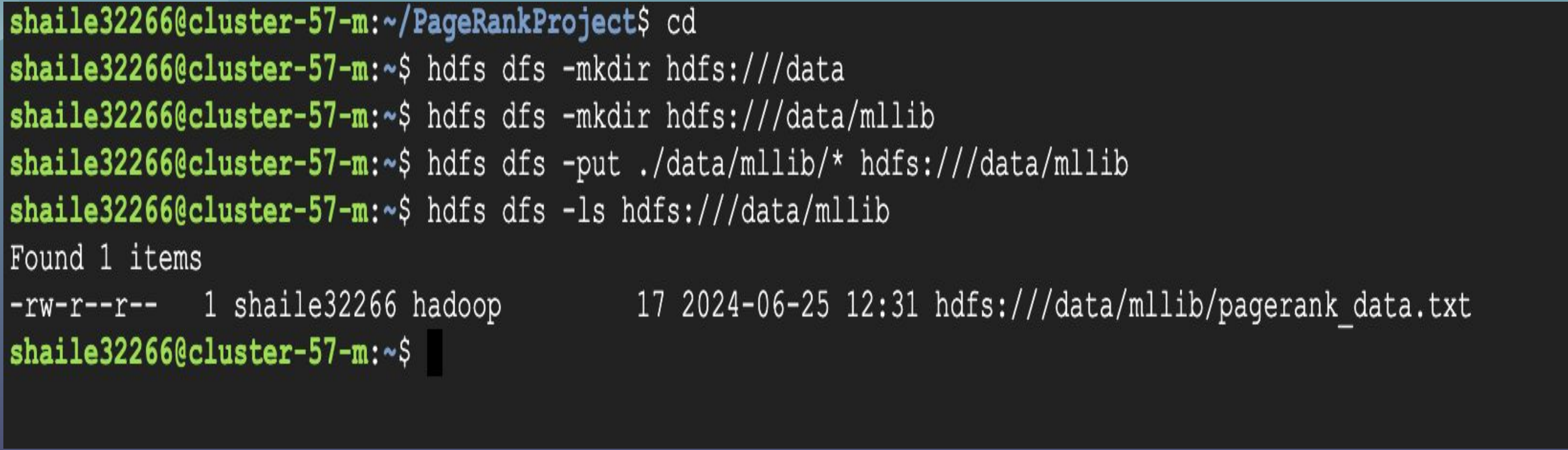
```
scala> val lines = sc.textFile("hdfs:///pagerank/links.txt")
lines: org.apache.spark.rdd.RDD[String] = hdfs:///pagerank/links.txt MapPartitionsRDD[1] at textFile at <consol
e>:23

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

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 * _)
     |        println("Iteration --> " + i)
     |        val output = ranks.collect()
     |        output.foreach(tup => println(s"${tup._1} has rank:  ${tup._2} ."))
     |      }
```

04

Testing

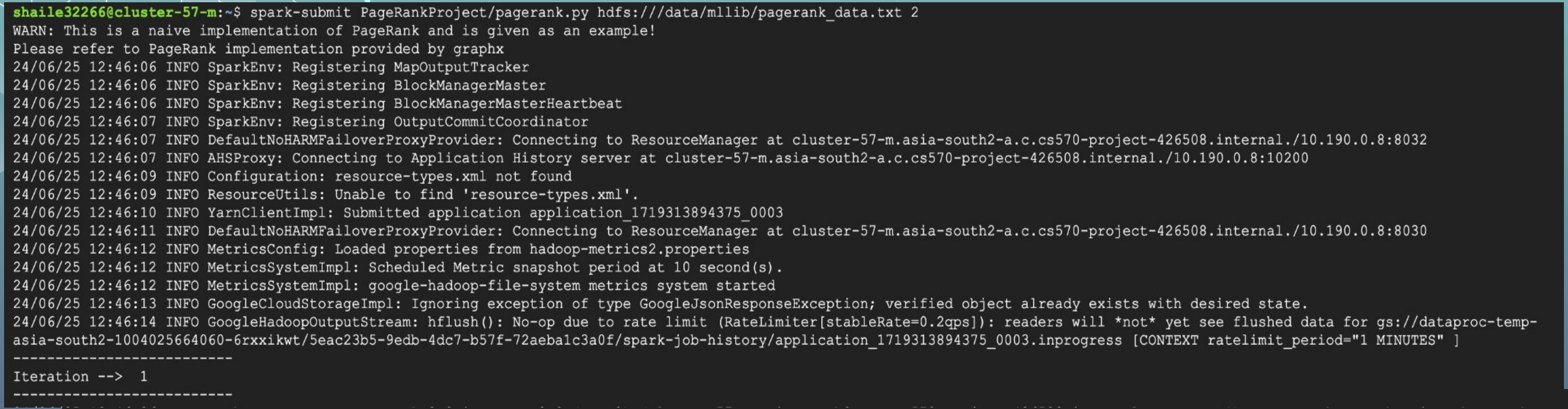
Run the program with input data file to test.

Pyspark: Create HDFS file system and copy input data file from local to hdfs:

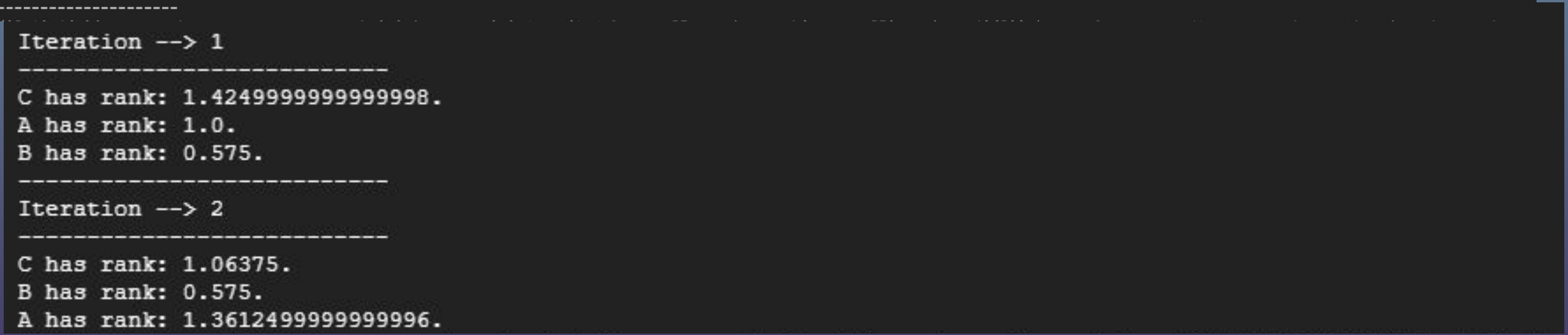


```
shaile32266@cluster-57-m:~/PageRankProject$ cd
shaile32266@cluster-57-m:~$ hdfs dfs -mkdir hdfs:///data
shaile32266@cluster-57-m:~$ hdfs dfs -mkdir hdfs:///data/mllib
shaile32266@cluster-57-m:~$ hdfs dfs -put ./data/mllib/* hdfs:///data/mllib
shaile32266@cluster-57-m:~$ hdfs dfs -ls hdfs:///data/mllib
Found 1 items
-rw-r--r--   1 shaile32266 hadoop         17 2024-06-25 12:31 hdfs:///data/mllib/pagerank_data.txt
shaile32266@cluster-57-m:~$
```

Test: Pyspark



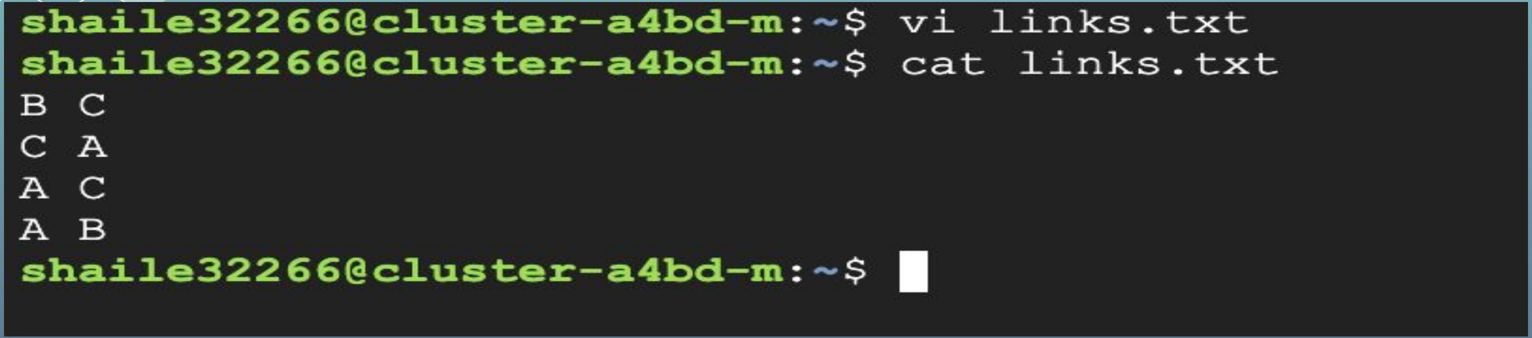
```
shaile32266@cluster-57-m:~$ spark-submit PageRankProject/pagerank.py hdfs:///data/mllib/pagerank_data.txt 2
WARN: This is a naive implementation of PageRank and is given as an example!
Please refer to PageRank implementation provided by graphx
24/06/25 12:46:06 INFO SparkEnv: Registering MapOutputTracker
24/06/25 12:46:06 INFO SparkEnv: Registering BlockManagerMaster
24/06/25 12:46:06 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/06/25 12:46:07 INFO SparkEnv: Registering OutputCommitCoordinator
24/06/25 12:46:07 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at cluster-57-m.asia-south2-a.c.cs570-project-426508.internal./10.190.0.8:8032
24/06/25 12:46:07 INFO AHSProxy: Connecting to Application History server at cluster-57-m.asia-south2-a.c.cs570-project-426508.internal./10.190.0.8:10200
24/06/25 12:46:09 INFO Configuration: resource-types.xml not found
24/06/25 12:46:09 INFO ResourceUtils: Unable to find 'resource-types.xml'.
24/06/25 12:46:10 INFO YarnClientImpl: Submitted application application_1719313894375_0003
24/06/25 12:46:11 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at cluster-57-m.asia-south2-a.c.cs570-project-426508.internal./10.190.0.8:8030
24/06/25 12:46:12 INFO MetricsConfig: Loaded properties from hadoop-metrics2.properties
24/06/25 12:46:12 INFO MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/06/25 12:46:12 INFO MetricsSystemImpl: google-hadoop-file-system metrics system started
24/06/25 12:46:13 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with desired state.
24/06/25 12:46:14 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2qps]): readers will *not* yet see flushed data for gs://dataproc-temp-
asia-south2-1004025664060-6rxxikwt/5eac23b5-9edb-4dc7-b57f-72aeba1c3a0f/spark-job-history/application_1719313894375_0003.inprogress [CONTEXT ratelimit_period="1 MINUTES" ]
--------------------------
Iteration -->  1
--------------------------
```



```
Iteration --> 1
----------------------------
C has rank: 1.4249999999999998.
A has rank: 1.0.
B has rank: 0.575.
----------------------------
Iteration --> 2
----------------------------
C has rank: 1.06375.
B has rank: 0.575.
A has rank: 1.3612499999999996.
```

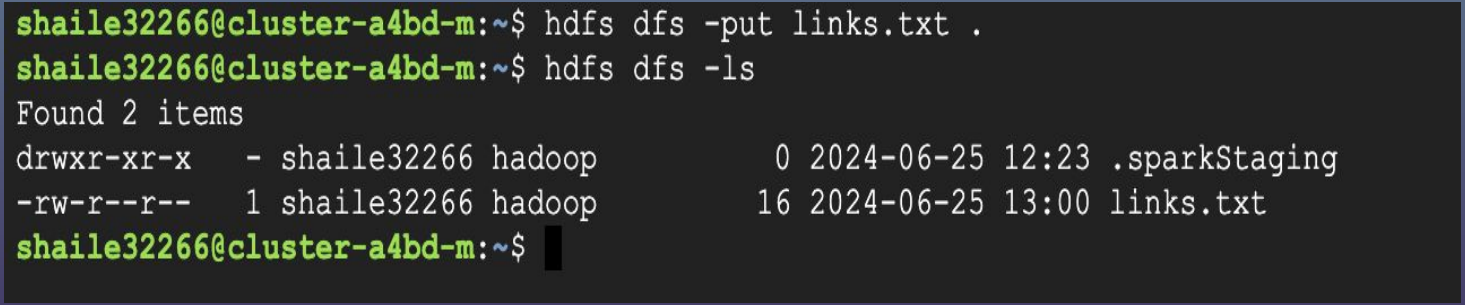
Test: Scala

1. Create file: links.txt



```
shaile32266@cluster-a4bd-m:~$ vi links.txt
shaile32266@cluster-a4bd-m:~$ cat links.txt
B C
C A
A C
A B
shaile32266@cluster-a4bd-m:~$ 
```

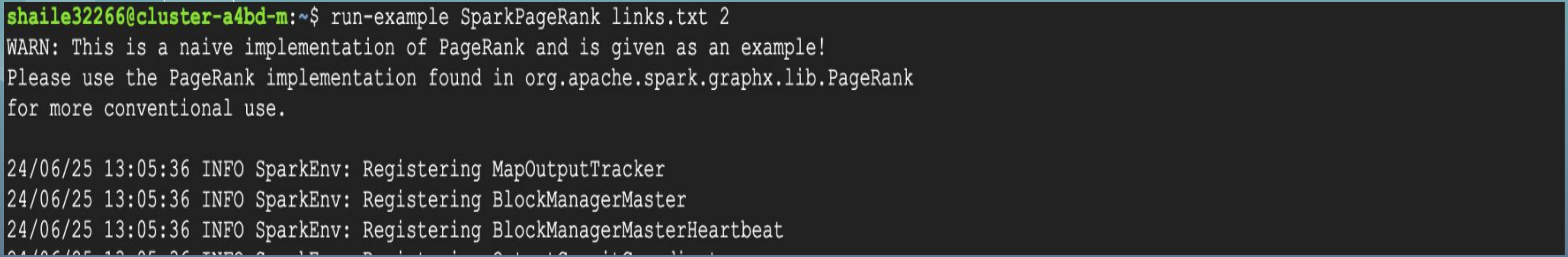
2. Copy input file from local to HDFS



```
shaile32266@cluster-a4bd-m:~$ hdfs dfs -put links.txt .
shaile32266@cluster-a4bd-m:~$ hdfs dfs -ls
Found 2 items
drwxr-xr-x   - shaile32266 hadoop          0 2024-06-25 12:23 .sparkStaging
-rw-r--r--   1 shaile32266 hadoop         16 2024-06-25 13:00 links.txt
shaile32266@cluster-a4bd-m:~$ 
```

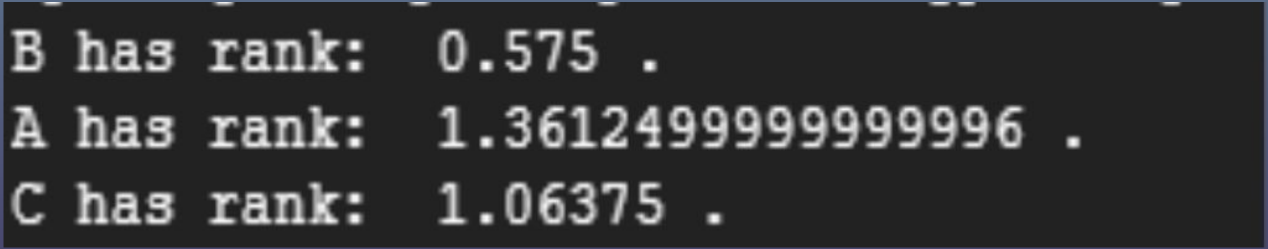
Test: Scala

3. Run-example SparkPageRank links.txt 2



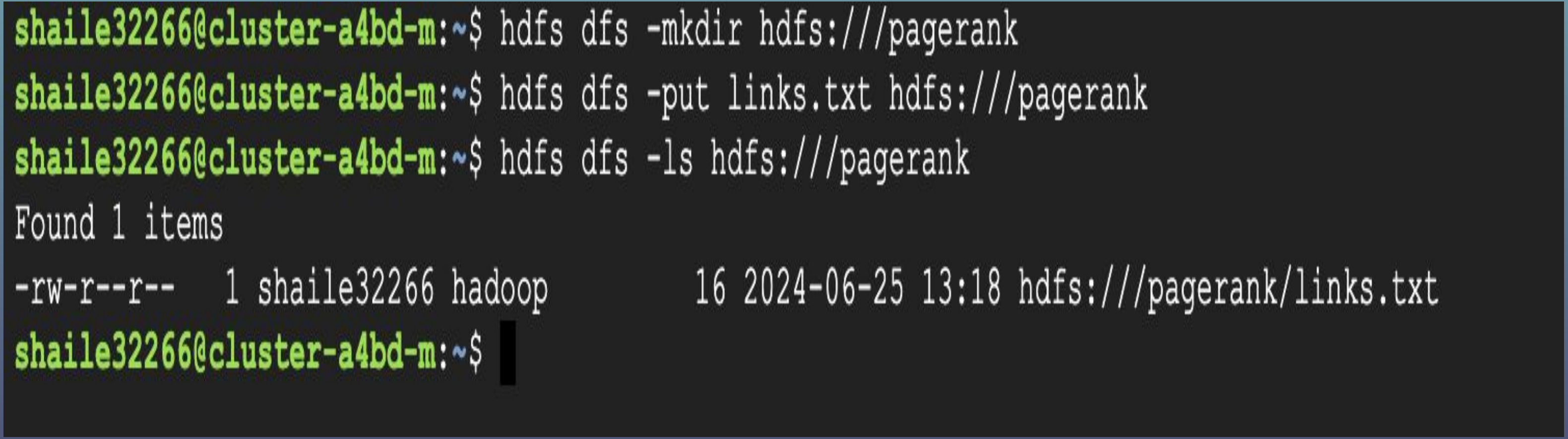
```
shaile32266@cluster-a4bd-m:~$ run-example SparkPageRank links.txt 2
WARN: This is a naive implementation of PageRank and is given as an example!
Please use the PageRank implementation found in org.apache.spark.graphx.lib.PageRank
for more conventional use.

24/06/25 13:05:36 INFO SparkEnv: Registering MapOutputTracker
24/06/25 13:05:36 INFO SparkEnv: Registering BlockManagerMaster
24/06/25 13:05:36 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
```

4. Result



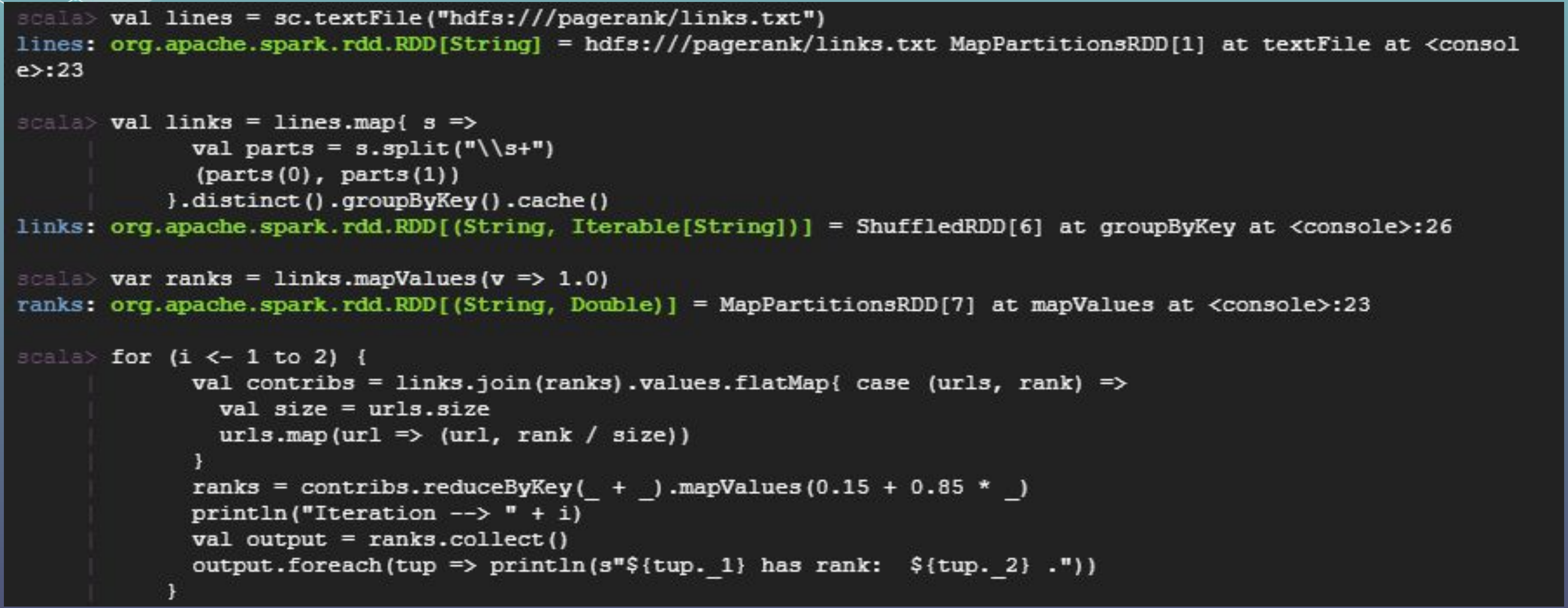
```
B has rank:  0.575 .
A has rank:  1.3612499999999996 .
C has rank:  1.06375 .
```

5. Test in Spark Shell with Modified Code



```
shaile32266@cluster-a4bd-m:~$ hdfs dfs -mkdir hdfs:///pagerank
shaile32266@cluster-a4bd-m:~$ hdfs dfs -put links.txt hdfs:///pagerank
shaile32266@cluster-a4bd-m:~$ hdfs dfs -ls hdfs:///pagerank
Found 1 items
-rw-r--r--   1 shaile32266 hadoop         16 2024-06-25 13:18 hdfs:///pagerank/links.txt
shaile32266@cluster-a4bd-m:~$
```

The modified code



```
scala> val lines = sc.textFile("hdfs:///pagerank/links.txt")
lines: org.apache.spark.rdd.RDD[String] = hdfs:///pagerank/links.txt MapPartitionsRDD[1] at textFile at <consol
e>: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> var ranks = links.mapValues(v => 1.0)
ranks: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[7] at mapValues at <console>:23

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 * _)
     |        println("Iteration --> " + i)
     |        val output = ranks.collect()
     |        output.foreach(tup => println(s"${tup._1} has rank:  ${tup._2} ."))
     |      }
```

Enhancement 05

- Integrate Google Cloud Storage (GCS) for data management.
- Optimize Spark configurations for better performance.
- Implement advanced data partitioning strategies.
- Explore real-time data processing with Apache Kafka.
- Create interactive dashboards for PageRank visualization.
- Utilize GCP’s autoscaling for resource management

Conclusion

06

- Implemented PageRank with PySpark and Scala on GCP.
- Gained practical experience in cloud-based big data processing.
- Achieved efficient handling of large datasets.
- Demonstrated scalability and high performance on GCP.
- Enhanced skills in cloud computing and data analytics.
- Ready to apply techniques to broader data projects.

GITHUB LINK

https://github.com/Sharon20222/Cloud-Computing/tree/main/Spark/PageRank

References

- https://www.scala-lang.org/

- https://spark.apache.org/docs/latest/api/python/index.html