Thomas Munoz Vasquez

CSC 4760

Assignment 5

Due April 12, 2024

# Spark GraphX

The PageRank algorithm serves as a metric for determining the significance of web pages, with its core concept being that a page gains importance if it is linked to by other important pages. The dataset employed in this program takes the form of an adjacency list, which illustrates relationships between vertices within a graph. The dataset creates a directed graph, facilitating the calculation of PageRank values assigned to each node.

To begin, the program loads an adjacency list file and reads its contents. Each line in the file is then split into an array of strings. The first items in the array are converted into vertices, as they correspond to the vertices in the adjacency list. Subsequently, edges are created using the first elements as the source vertex and mapping the remaining elements to destination vertices. A directed graph is then created using the vertices and edges. Next, the PageRank algorithm is applied to calculate the PageRank for each vertex in the graph. The algorithm is set to run for 30 iterations with a damping factor of 0.15. The program creates a new variable to store the vertices of the graph along with their PageRank values. It calculates the sum of all PageRank values in the new variable and divides each PageRank value by the total PageRank sum. Finally, the results are saved to a text file on the local disk. The

implementation provided more accurate results, as the expected PageRank values are rounded.

```scala
import org.apache.spark.graphx.{GraphLoader, Graph,Edge}
import org.apache.spark.SparkContext

//loads file
val file = sc.textFile("02AdjacencyList.txt")

//split each line into an array of strings
val linesArray = file.map(line => line.split(" ").map(_.toLong))

//maps first item of array, map vertices to 1
val firstItemstoVertice = linesArray.flatMap(_.headOption).map(vertice => (verti

//edges
val create_edges = linesArray.flatMap{arr =>
  val src = arr.head
  arr.tail.map(dest => Edge(src, dest, 0))
}

//new graph
val graph = Graph(firstItemstoVertice, create_edges)

val vertices = graph.vertices.collect()
val edges = graph.edges.collect()

//calculate pagerank
val PageRankGraph1 = graph.staticPageRank(30, 0.15)

//stores pagrerank with vertices.
val PageRankVertices1 = PageRankGraph1.vertices

//total pagerank sum
val dSum1 = PageRankVertices1.map(values => values._2).sum()

//each pagerank is divided by pagerank sum
val PageRankVertices11 = PageRankVertices1.mapValues(prValue=> prValue / dSum1)

//displays pagerank
val PageRankValues1 = PageRankVertices11.collect()
println("PageRank:")
println(PageRankValues1.mkString("\n"))

//save result to textfile
PageRankVertices1.map { case (vertexId, pageRank) =>
    s"$vertexId: $pageRank"
  }
  .coalesce(1)  // Reduce to a single partition
  .saveAsTextFile("results")
```

*Figure 1. PageRank Code.*

```
Pagerank.py    spark_pagerank_graphx.scala ×    file.txt    02AdjacencyList.txt
Examples > spark_pagerank_graphx.scala
1  import org.apache.spark.graphx.{GraphLoader, Graph,Edge}
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   ...   bash - Examples

scala> import org.apache.spark.graphx.{GraphLoader, Graph,Edge}
import org.apache.spark.graphx.{GraphLoader, Graph, Edge}

scala> import org.apache.spark.SparkContext
import org.apache.spark.SparkContext

scala>

scala> //loads file

scala> val file = sc.textFile("02AdjacencyList.txt")
file: org.apache.spark.rdd.RDD[String] = 02AdjacencyList.txt MapPartitionsRDD[869] at
 textFile at <console>:40

scala>

scala> //split each line into an array of strings

scala> val linesArray = file.map(line => line.split(" ").map(_.toLong))
linesArray: org.apache.spark.rdd.RDD[Array[Long]] = MapPartitionsRDD[870] at map at <
console>:40

scala>

scala> //maps first item of array, map vertices to 1

scala> val firstItemstoVertice = linesArray.flatMap(_.headOption).map(vertice => (ver
tice,1))
firstItemstoVertice: org.apache.spark.rdd.RDD[(Long, Int)] = MapPartitionsRDD[872] at
 map at <console>:40

scala>

scala> //edges

scala> val create_edges = linesArray.flatMap{arr =>
     |   val src = arr.head
     |   arr.tail.map(dest => Edge(src, dest, 0))
     | }
create_edges: org.apache.spark.rdd.RDD[org.apache.spark.graphx.Edge[Int]] = MapPartit
ionsRDD[873] at flatMap at <console>:40

scala>

scala> //new graph

scala> val graph = Graph(firstItemstoVertice, create_edges)
graph: org.apache.spark.graphx.Graph[Int,Int] = org.apache.spark.graphx.impl.GraphImp
l@4499dde5
```

*Figure 2. Running code (1).*

```
Pagerank.py   spark_pagerank_graphx.scala   file.txt   02AdjacencyList.txt
Examples > spark_pagerank_graphx.scala
1   import org.apache.spark.graphx.{GraphLoader, Graph,Edge}
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   bash - Examples

scala> val vertices = graph.vertices.collect()
vertices: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((4,1), (2,1), (1,1),
 (3,1), (5,1))

scala> val edges = graph.edges.collect()
edges: Array[org.apache.spark.graphx.Edge[Int]] = Array(Edge(1,2,0), Edge(2,3,0), Edg
e(2,4,0), Edge(3,4,0), Edge(4,1,0), Edge(4,5,0), Edge(5,3,0))

scala>

scala> //calculate pagerank

scala> val PageRankGraph1 = graph.staticPageRank(30, 0.15)
PageRankGraph1: org.apache.spark.graphx.Graph[Double,Double] = org.apache.spark.graph
x.impl.GraphImpl@57e1a5fa

scala>

scala> //stores pagrerank with vertices.

scala> val PageRankVertices1 = PageRankGraph1.vertices
PageRankVertices1: org.apache.spark.graphx.VertexRDD[Double] = VertexRDDImpl[1272] at
 RDD at VertexRDD.scala:57

scala>

scala> //total pagerank sum
```

```
gerank.py   spark_pagerank_graphx.scala   file.txt   02AdjacencyList.txt
Examples > spark_pagerank_graphx.scala
1   import org.apache.spark.graphx.{GraphLoader, Graph,Edge}
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   bash - Examples

scala> val dSum1 = PageRankVertices1.map(values => values._2).sum()
dSum1: Double = 5.000000000000001

scala>

scala> //each pagerank is divided by pagerank sum

scala> val PageRankVertices11 = PageRankVertices1.mapValues(prValue=> prValue / dSum1
)
PageRankVertices11: org.apache.spark.graphx.VertexRDD[Double] = VertexRDDImpl[1281] a
t RDD at VertexRDD.scala:57

scala>

scala> //displays pagerank

scala> val PageRankValues1 = PageRankVertices11.collect()
PageRankValues1: Array[(org.apache.spark.graphx.VertexId, Double)] = Array((4,0.29545
97724206865), (2,0.1622382181527363), (1,0.15555492363830503), (3,0.23119216214996713
), (5,0.15555492363830503))

scala> println("PageRank:")
PageRank:

scala> println(PageRankValues1.mkString("\n"))
(4,0.2954597724206865)
(2,0.1622382181527363)
(1,0.15555492363830503)
(3,0.23119216214996713)
(5,0.15555492363830503)

scala>

scala> //save result to textfile

scala> PageRankVertices1.map { case (vertexId, pageRank) =>
     |     s"$vertexId: $pageRank"
     |   }
res25: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1282] at map at <console>:
41

scala>   .coalesce(1)  // Reduce to a single partition
res26: org.apache.spark.rdd.RDD[String] = CoalescedRDD[1283] at coalesce at <console>
:41

scala>   .saveAsTextFile("results")

scala>

scala>
```

*Figure 3. Running code (2)*

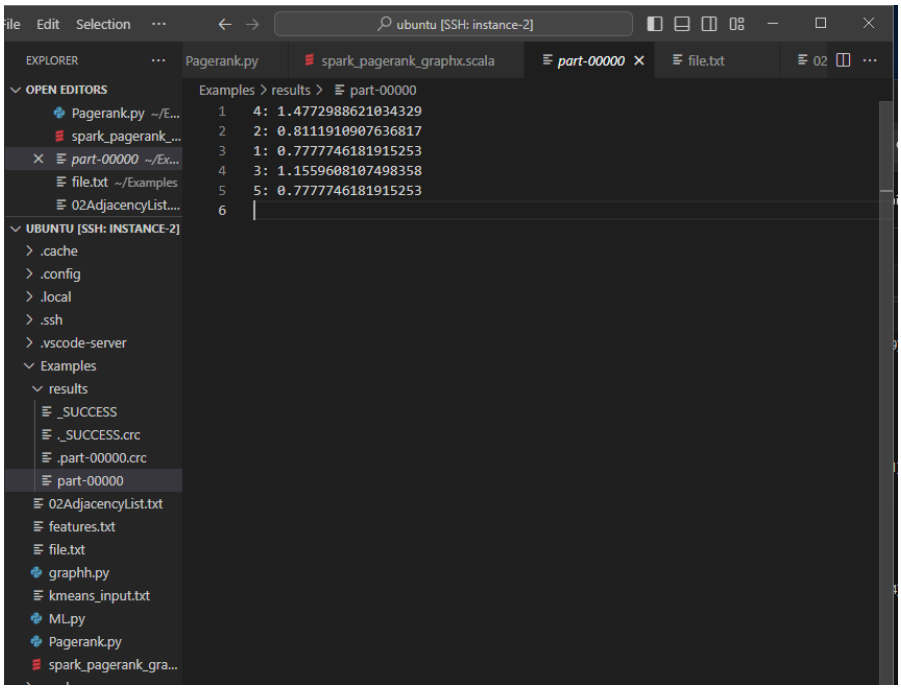


*Figure 4. New file and output.*