Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра системного проектування

Оброблення надвеликих масивів даних

Лабораторна робота №3

Виконала:

студентка групи ДА-11мп

Молчанова В.С.

Київ – 2021 р.

## Завдання:

Pеалізація та розуміння алгоритму “Page Rank”

## Хід роботи:

В даній реалізації враховується проблема «западин» (sinks): сторінок, що не мають вихідних посилань та які порушують роботу алгоритму, накопляючи в собі ранг. Для того щоб розподілити їх ранг між іншими сторінками, у функції handleSinks до кожної западини додаються посилання на усі інші сторінки.

Також під час реалізації алгоритму виникала проблема зі сторінками, на які немає посилань. Наприклад, це сторінка А. Алгоритм ітеративно обробляє список вихідних посилань кожної сторінки urls і мапить їх імена до їх рангу. Але так як сторінка А не зустрічається в жодному з цих списків, після першої ж ітерації вона зникала з обчислень. Для вирішення цієї проблеми на кожній ітераціїї таким сторінкам задається ранг 1-d, адже формула

Diagram, text

Description automatically generated

за відсутністю посилань Ті зводиться до 1-d.

### Текст програми:

Файл BigData.scala

import org.apache.spark.{SparkConf, SparkContext}  
  
object BigData {  
 def main(args: Array[String]) = {  
 val (lab, output) = *parseParameters*(args)  
 val conf = new SparkConf().setAppName("appName").setMaster("local")  
 val sc = new SparkContext(conf)  
 val result =  
 if (lab == "2" ) *getInvertedIndices*(sc, "D:/Programming/spark-3.2.0-bin-hadoop3.2/input/lab2/")  
 else *getPageRank*(sc)  
 sc.stop()  
 *writeToFile*(result, output)  
 }  
  
 def parseParameters(args: Array[String]): (String, String) = {  
 val lab = if (args.length > 0) args(0) else "2"  
 val output = if (args.length > 1) args(1) else "output.txt"  
 (lab, output)  
 }  
  
 def writeToFile(result: String, filePath: String): Unit ={  
 val pw = new PrintWriter(new File(filePath ))  
 pw.write(result)  
 pw.close()  
 }  
}

Файл PageRank.scala

import org.apache.spark.SparkContext  
import org.apache.spark.rdd.RDD  
  
object PageRank {  
 def getPageRank(sc: SparkContext, iterations:Int=50, d:Double=0.85): String = {  
 val lines = sc.textFile("D:\\Programming\\spark-3.2.0-bin-hadoop3.2\\input\\lab3\\pages.txt")  
 val linksWithoutSinks = lines.map(l => l.split("\\W")).map(xs => (xs.head, xs.tail))  
 val links = *handleSinks*(linksWithoutSinks).persist()  
 var ranks = links.map(l => (l.\_1, 1.0))  
 for (\_ <- 1 to iterations) {  
 val linksWithRanks = links.join(ranks) *// t-> (c(t), pr(t))* val citationRanks = linksWithRanks.values.flatMap{ case (urls, rank) =>  
 urls.map(url => (url, rank / urls.length)) *// a -> pr(tx)/c(tx)* }  
 val newRanks = citationRanks.reduceByKey(\_ + \_).mapValues((1-d) + d \* \_)  
 *// allows to keep ranks of pages for which there are no links* ranks = ranks.leftOuterJoin(newRanks) mapValues {  
 case (\_, *Some*(newRank)) => newRank  
 case (\_, None) => 1 - d *// if there are no links to page, expression (1 - d) + d\*sum(links) becomes (1-d)* }  
 }  
 val result = ranks.collect()  
 val n = result.length  
 result.map(r => "Page %1s has rank %2f".format(r.\_1, r.\_2/n))  
 .mkString("Result:\n", "\n", "").replace(",", ".")  
 }  
  
 *// sink is page with no outer links  
 // this function adds to every sink a link to every other page (including sinks)* def handleSinks(linksWithoutSinks: RDD[(String, Array[String])]): RDD[(String, Array[String])] = {  
 val persistedLinks = linksWithoutSinks.persist()  
 val sinks = persistedLinks.flatMap(p => p.\_2).distinct().map(l => (l, 1)).leftOuterJoin(linksWithoutSinks)  
 .filter(l => l.\_2.\_2.isEmpty).keys  
 val allPages = sinks.union(persistedLinks.keys).collect()  
 val sinksEdges = sinks.map(s => (s, allPages.filter {\_ != s}))  
 val linksWithSinks = persistedLinks.union(sinksEdges)  
 persistedLinks.unpersist()  
 linksWithSinks  
 }  
}

### Вхідні дані

Файл pages.txt

b c

c b

d a b

e b d f

f b e

g b e

h b e

i b e

j e

k e

### Результат виконання

Graphical user interface, text, application

Description automatically generated

### Контрольні питання:

*1) Опишіть типи даних, які ви використовували для ключів та значень на різних етапах MapReduce. Якщо ви серіалізували дані в текст і назад, опишіть, як ви заклали вміст тексту в потік даних.*

* Ім’я сторінки => масив імен сторінок, на які вона посилається (String => Array[String])
* Ім’я сторінки => її ранг (String => Double) (як результат етапу Map та як результат етапу Reduce)
* Ім’я сторінки => її старий ранг, опція нового рангу (String => (Double, Option[Double])) (для збереження сторінок, на які нема посилань)

*2) Яку масштабованість, якщо така є, ви бачите в цій системі?*

Мені здається, що цю систему достатньо легко масштабувати, адже якщо зчитувати дані не з одного файлу, а з багатьох, які можуть бути результатом роботи алгоритмів по збору даних про посилання, то ця система з мінімальною зміною зможе зібрати велику кількість даних з усіх цих файлів та обробити її.

*3) Опишіть процес тестування вашого коду перед його запуском на великому наборі даних.*

Під час виконання роботи я виявила два проблемних моменти, вартих уважного тестування: «западини» та сторінки без вхідних посилань. Обидва таких типів сторінок представлені в тестовому файлі (сторінка а – западина, сторінки g, h, I, j, k не мають вхідних посилань).