Assignment
Rooting Trees with Apache Spark

Akash Sharma
UBIT-as475
as475@buffalo.edu
Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY 14214

26 November 2021

1 Introduction

This assignment is to implement and analyze efficient Apache spark program to find the ultimate root of tree.

2 Code used in rooting the tree

```
def splitST(line):
fields = line.split(" ")
Source = int(fields[0])
Target = int(fields[1])
return (Source, Target)

    if __name__ == "__main__":
conf = SparkConf().setMaster("local[*]").setAppName("find the root")
sc = SparkContext(conf = conf)
sc.setLogLevel("ERROR")
StartTime = time.time()
lines = sc.textFile(sys.argv[1])
rdd = lines.map(splitST)
length=rdd.count()
root = rdd.filter(lambda x:x[0] == x[1])
reverserdd = rdd.map(lambda x: (x[1],x[0]))

    while True:
root = root.join(reverserdd).groupByKey().mapValues(tuple).flatMap(lambda
x:x[1]).map(lambda x: (x[1],x[0]))
if root.count() == length:
break

    root.coalesce(1).saveAsTextFile(sys.argv[2])
sc.stop()
print("time : ", time.time()-StartTime)
```

2.1 Above code approch 1

In the above, By observing the property of the join above program is written. Firstly the root is obtain by filtering the rows for and then after reversing the coloumn of the tree root is combined with reverserdd. In this approch getting successful result but it is taking to extra time due to the skewness of the obtain in the join.

3 Another tried code

```
import pyspark
from pyspark import SparkConf, SparkContext
from collections import defaultdict
import sys
import time
conf = SparkConf().setMaster("local[*]").setAppName("root of all vertex")
sc = SparkContext(conf = conf)

    def splitVE(line):
line = line.split(" ")
edge = int(line[0])
vertex = int(line[1])
parent[edge] = vertex
addEdge(edge, vertex)
parent = int(findPar(vertex))
return (edge , parent[edge])

    def splitVE1(result):
result = result.split(",")
edge = int(result[0])
vertex = int(result[1])
addEdge(edge, vertex)
return (edge , parent[edge])

    def addEdge(u,v):
parent[u] = int(findPar(v))
```

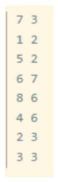
```
    def findPar(vertex):
parent[vertex] = edge
if(parent.get(vertex)==None or parent.get(vertex)==vertex) :
parent[vertex] = int(vertex)
return vertex;
else:
return findPar(parent[vertex])

    if __name__ == "__main__": line = sc.textFile(sys.argv[1])
StartTime = time.time()
sc.setLogLevel("ERROR")
parent = defaultdict(list)
rdd = line.map(splitVE)
rdd1 = rdd.map(splitVE1).sortByKey()
rdd2 = rdd1.map(splitVE1).sortByKey(ascending=False)
final = rdd2.map(splitVE1)
answer=final.collect()
r = sc.parallelize(answer)
r.saveAsTextFile(sys.argv[2])
sc.stop()
print("time : ", time.time()-StartTime)
```

3.1 Above code approach 2

In the above code, recursion is used to get the the ultimate root of the tree , in this code tried to obtain the root of the recursively, in this getting the parent but code is not efficient.

4 Output obtain after running the code

4.1 Input tree given



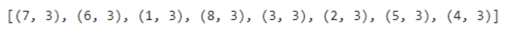
```
7 3
1 2
5 2
6 7
8 6
4 6
2 3
3 3
```

Figure 1: Input Tree

After giving above input getting below output.

5 Output obtain after running the code

5.1 Input tree given



```
[(7, 3), (6, 3), (1, 3), (8, 3), (3, 3), (2, 3), (5, 3), (4, 3)]
```

Figure 2: Output Tree