Apache Spark - GraphX

Consider an interesting case study, construct a graph using Apache Spark Graphx library and find the following partition algorithms for the graph

i Connected components
ii Page rank
iii Triangle counting
iv Partition Strategies

Justify how each of these algorithms relevance and use in your case study.

Submit the following

a) Code through Github

b) Report with Algorithm, Sample Input and Output, Complexity and Time Performance.

c) Bonus for comparative evaluation with other techniques

Sol:

We considered three busses which travel to the five cities. These cities are the vertices of our graph.

These vertices have the verticesID

```
val vertices=Array((1L, ("SFO")),(2L, ("ORD")),(3L,("DFW")),(4L,("BRW")),(5L,("JFD")))
val vRDD= sc.parallelize(vertices)
```

Edges are the routes between these bus stations. These edges should have the source and destination.

```
val edges = Array(Edge(1L,2L,1800),Edge(2L,3L,800),Edge(3L,4L,1400)),Edge(4L,5L,400)),Edge(5L,1L,1400))
val eRDD= sc.parallelize(edges)
// Array(Edge(1,2,1800), Edge(2,3,800))
```

To create a graph we need vertices RDD and edge RDD.

```
val nowhere = ("nowhere")
val graph = Graph(vRDD,eRDD, nowhere)

graph.vertices.collect.foreach(println)


graph.edges.collect.foreach(println)
```

Page Rank:
PageRank measures the importance of each vertex in a graph, by determining which vertexes have the most edges with other vertexes. In our example we can use PageRank to determine which bus station are the most important, by measuring which bus station have the most connections to other bus stations.

```
val ranks = graph.pageRank(0.1).vertices

ranks.take(3)

val impbusstations = ranks.join(vRDD).sortBy(_._2._1, false).map(_._2._2)
impbusstations.collect.foreach(println)
```

Connected components

Connected component algorithm label the each connected component of the graph with IDs of least numbered vertices.

```
val connectedComponents = graph.connectedComponents().vertices
println("GCC:")
connectedComponents.foreachpartition(println)
```

Triangle Counting

A vertex is a part of the triangle which has the two adjacent edges. Triangle count algorithm is how many triangles passing through each vertex.

```
val trCounting=graph.triangleCount().vertices
println("Triangle counting:")
trCounting.foreach(println)
```

Connected components time complexity: O(q+vlogv).

Other techniques:

Since the properties of the vertices depend on the properties of the other vertices there are some iterative algorithms on such is pregel.

Pregel

With the implementation of pregel algorithm the message is send from one vertex to its neighbor.

To send a message with pregel we have to send the message in series of steps

Vertices receive the sum of the inbound message from the previous step

New value is calculated.

Send the message to next neighbor in the series.

Github Link: https://github.com/venkatagovardhan/Parallel-algorithms.git