CS5560PA: Parallel Algorithms
Assignment – 5
Apache Spark – GraphX

Submitted By Team 7
Rama Charan Pavan Jakkepalli
Sri Charan Pamuru
Pratap Rao

The case study that has been considered includes the parallel processing class and how team 7 is connected to the main domain.

1. **Connected Components:**

The following are the vertices:

```
16/12/05 12:07:54 INFO DAGScheduler: Job 0 finished: zipWithIndex at GraphxAlgorithms.scala:34, took
vertices
16/12/05 12:07:54 INFO SparkContext: Starting job: foreach at GraphxAlgorithms.scala:61
```

```
16/12/05 12:07:54 INFO Shu
(0,degree)
(1,phd)
(2,pratap)
(3,mayanka)
(4,cs)
(5,umkc)
(6,algorithms)
(16,studies)
(17,course)
(7,parallel algorithms)
(8,fall semester)
(9,sricharan)
(10,university)
(11,pavan)
(12,team7)
(13,master)
(14,member)
(15,department)
```

The edges for the graph can be shown as follows:

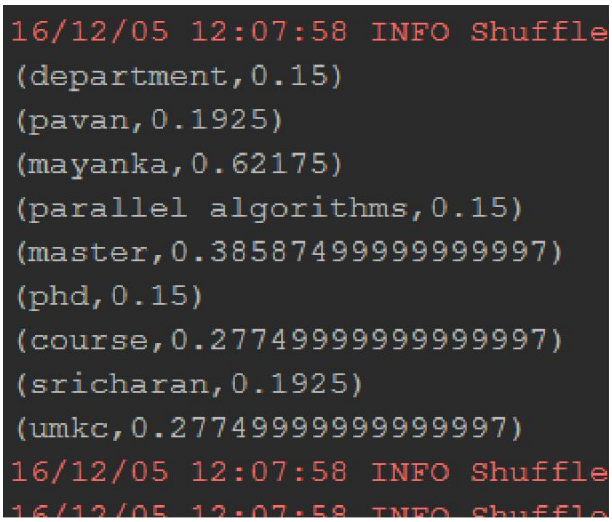
```
16/12/05 12:07:54
edges:
16/12/05 12:07:54
```

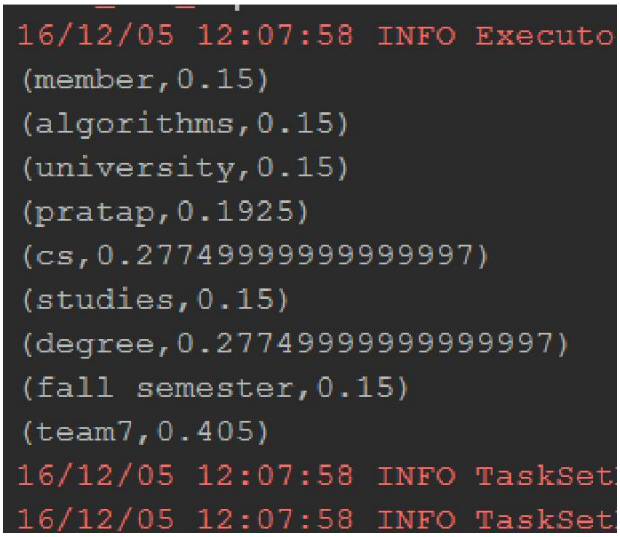
```
16/12/05 12:07:55 IN
Edge(0,13,is)
Edge(1,3,studies)
Edge(6,12,designs)
16/12/05 12:07:55 IN
Edge(16,0,part of)
16/12/05 12:07:55 IN
Edge(7,12,studies)
16/12/05 12:07:55 IN
Edge(8,17,part of)
16/12/05 12:07:55 IN
16/12/05 12:07:55 IN
16/12/05 12:07:55 IN
16/12/05 12:07:55 IN
Edge(10,5,is)
Edge(12,3,guides)
Edge(14,2,is)
Edge(14,9,is)
Edge(14,11,is)
Edge(15,4,is)
16/12/05 12:07:55 IN
```

2. **Page Rank:**

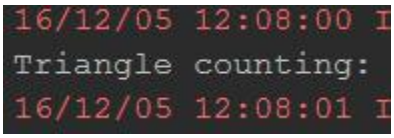
   In order to get the page rank, the following can be shown:

```
16/12/05 12:07:58
Page Rank:
16/12/05 12:07:58
```





3. **Triangle counting:**



```
16/12/05 12:
(4,0)
(16,0)
(0,0)
(8,0)
(12,0)
16/12/05 12:
16/12/05 12:
[rdd_31_1, r
(13,0)
(1,0)
(17,0)
(9,0)
(5,0)
16/12/05 12:
(14,0)
16/12/05 12:
(6,0)
[rdd_31_2, r
(10,0)
16/12/05 12:
(2,0)
. . . .
16/12/05
(15,0)
(11,0)
(3,0)
(7,0)
16/12/05
```

4. **Partitioned edges:**

```
val unpartitionedGraph: Graph[Int, Int] = ...val numPartitions: Int = 128

def getTripletPartition(e: EdgeTriplet[Int, Int]): PartitionID = ...

// Get the triplets using GraphX, then use Spark to repartition

themval partitionedEdges = unpartitionedGraph.triplets

  .map(e => (getTripletPartition(e), e))

  .partitionBy(new HashPartitioner(numPartitions))

  *.map(pair => Edge(pair._2.srcId, pair._2.dstId, pair._2.attr))*

val partitionedGraph = Graph(unpartitionedGraph.vertices, partitionedEdges)
```

Algorithm comparison:

The graph is plotted based upon the understanding of how the input csv file has the nodes and edges which are weighted against each other. The sample input can be the files that has been provided. The output has been shown above in the form of screenshot.

Complexity and Time performance:

O(v+e)
Time taken: 2 seconds

Comparative evaluation:

Giraph: inspired by Pregel, but requires neighbors to perform processing.

GraphLab: there is abstraction to split the large graphs in a cluster and allows the processing in both synchronous and asynchronous ways.

Powergraph: ideal for graphs based onn natural law

Stratosphere: alternative to spark. DAG specification and execution engines are present.

SPARQLVerse: Extreme scalability

Genomix: data parallel genome assembler for the Genome processing.