Program 4: Parallelizing Shortest-Path Search, using a BFS approach on Spark

Name: Pooja Nadagouda

The following algorithm is used to find the shortest path using BFS in an undirected weighted graph.

The user inputs for this algorithm are:

- Text file name containing the graph adjacency matrix along with edge weights,
- Source vertex
- Destination vertex

The graph text file is read in lines (stored as RDD). These RDD lines are converted to a PairRDDs using spark transformation mapToPair where key is vertex and value is Vertex Data (attributes - neighbors, previousDistance, status, distance). To do this mapping 2 functions are used i.e., getVertex(rdd_line) and getVertexData(rdd_line). Each line of graph text file is read to get the vertex and vertex data attributes. For all vertices the data attributes are initialized with corresponding neighbors as list of Tuple<VertexName,Weight>, distance=0, prev=Integer.MAX_VALUE and status="INACTIVE".
Only for source vertex the status is set to "ACTIVE". By doing so our BFS while loop is executed atleast once. Condition used to end the while loop is checking if number of active vertices is greater than 0. Active vertices are checked using filter function on status attribute of vertex data.

Inside while loop following 3 spark transformation/action functions are used:

1. faltMapToPair -> here we check for vertices which have active status. When an active vertex is found we iterate through its neighbors and assign new distance value to the neighbor vertices. The new distance is sum of active vertex's distance and weight of current neighbor. All such neighbors are added to the list with Tuple<vertexName, vertexData> where vertexData contains new distance attribute and all other attributes are null.

2. reduceByKey -> here the list created in previous step is checked for duplicate keys i.e., active vertices neighbors are computed for new distance resulting in duplicate vertices. Hence these duplicate vertices with different vertexData are reduced to a single entry. For this reduction the following rules are followed:
   - if one of the vertexData has distance 0 and other has distance greater than 0 then the vertexData with distance greater than 0 is taken.
   - if both vertexData have distance greater than 0 then the smallest distance vertexData is taken.

   In both these cases if the vertexData chosen has null attributes, then care is taken to initialize its other Attributes to non-null values.

3. mapToValues -> Here all the vertex data is checked to see if the current distance attribute has lesser value than the prev attribute. If yes, then the prev attribute value is changed to distance attribute value and this vertex's status attribute is made ACTIVE. If no, then the distance attribute value is changed to prev attribute value.

Finally, we check if active vertices exist. The count of active vertices is updated to continue with next iteration of while loop.
Once while loop ends then the destination vertex's distance is printed out.

Performance:

Computing Nodes	Average Cores Per Node	Total Cores	Elapsed Time (In Seconds)	Performance Improvement (W.R.T. My 1 Node 1 Core)
1	1	1	222.201	1
1	2	2	157.531	1.41
1	3	3	152.782	1.45
1	4	4	148.714	1.49
2	1	2	184.937	1.20
2	2	4	127.690	1.74
2	3	6	124.662	1.78
2	4	8	122.235	1.82
3	1	3	200.201	1.12
3	2	6	142.568	1.56
3	3	9	134.909	1.64
3	4	12	117.161	1.89
4	1	4	208.883	1.06
4	2	8	129.017	1.72
4	3	12	123.468	1.79
4	4	16	117.541	1.89
5	1	5	196.948	1.13
5	2	10	130.705	1.70
5	**3**	**15**	**114.373**	**1.94**
5	4	20	120.411	1.85

Best performance is offered with 5 computing nodes with 3 cores.

Execution screenshots:

For 1 node:

```
[poojan26@cssmpi1h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi1h:58170
--executor-memory 1G --total-executor-cores 1 ShortestPath.jar graph.txt 0 1500
2022-11-23 12:07:37 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes
where applicable
shortest path: 41
Elapsed time:222201
[poojan26@cssmpi1h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi1h:58170
--executor-memory 1G --total-executor-cores 2 ShortestPath.jar graph.txt 0 1500
2022-11-23 12:11:57 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes
where applicable
shortest path: 41
Elapsed time:157531
[poojan26@cssmpi1h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi1h:58170
--executor-memory 1G --total-executor-cores 3 ShortestPath.jar graph.txt 0 1500
2022-11-23 12:16:10 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes
where applicable
shortest path: 41
Elapsed time:152782
[poojan26@cssmpi1h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi1h:58170
--executor-memory 1G --total-executor-cores 4 ShortestPath.jar graph.txt 0 1500
2022-11-23 12:33:19 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes
where applicable
shortest path: 41
Elapsed time:148714
```

For 2 nodes:

```
[poojan26@cssmpi1h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi1h:58170
--executor-memory 1G --total-executor-cores 1 ShortestPath.jar graph.txt 0 1500
2022-11-23 11:38:45 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:184937
[poojan26@cssmpi1h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi1h:58170
--executor-memory 1G --total-executor-cores 2 ShortestPath.jar graph.txt 0 1500
2022-11-23 11:41:58 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:127690
[poojan26@cssmpi1h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi1h:58170
--executor-memory 1G --total-executor-cores 3 ShortestPath.jar graph.txt 0 1500
2022-11-23 11:44:37 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:124662
[poojan26@cssmpi1h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi1h:58170
--executor-memory 1G --total-executor-cores 4 ShortestPath.jar graph.txt 0 1500
2022-11-23 11:47:11 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:122235
```

For 3 nodes:

```
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 1 ShortestPath.jar graph.txt 0 1500
2022-11-23 09:57:45 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:200201
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 2 ShortestPath.jar graph.txt 0 1500
2022-11-23 10:01:53 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:142568
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 3 ShortestPath.jar graph.txt 0 1500
2022-11-23 10:04:32 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:134909
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 4 ShortestPath.jar graph.txt 0 1500
2022-11-23 10:07:05 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:117161
```

For 4 nodes:

```
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 1 ShortestPath.jar graph.txt 0 1500
2022-11-23 10:17:43 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:208883
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 2 ShortestPath.jar graph.txt 0 1500
2022-11-23 10:23:23 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:129017
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 3 ShortestPath.jar graph.txt 0 1500
2022-11-23 10:32:28 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:123468
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 4 ShortestPath.jar graph.txt 0 1500
2022-11-23 10:34:50 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:117541
```

For 5 nodes:

```
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 1 ShortestPath.jar graph.txt 0 1500
2022-11-23 10:54:21 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:196948
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 2 ShortestPath.jar graph.txt 0 1500
2022-11-23 10:58:46 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:130705
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 3 ShortestPath.jar graph.txt 0 1500
2022-11-23 11:01:36 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:114373
[poojan26@cssmpi8h prog4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/spark-submit  --class  ShortestPath  --master  spark://cssmpi8h:58170
--executor-memory 1G --total-executor-cores 4 ShortestPath.jar graph.txt 0 1500
2022-11-23 11:03:59 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
shortest path: 41
Elapsed time:120411
```

Pros and cons of the algorithm used:

1. Writing code and parallelizing for BFS is simpler when compared to complex algorithms like Dijkstra's shortest path.

2. Using spark framework, BFS shortest path parallelized code highly varies from sequential code. Unlike Spark framework, OpenMP framework there is almost no difference between parallel code and sequential code. We need to come up with a completely new code using spark transformations/actions. Hence here more effort is required to come up with parallel code.

3. For an undirected weighted graph BFS does not perform efficiently. In case of typical BFS for finding shortest path, it follows the strategy of number of edges from the source vertex and does not re-visit the visited nodes. But here if a node is already visited there is a still a possibility that it can be re-visited and recomputed for distance from a path through its child node that may have more edges but may have lesser edge weight. Hence this increases the time complexity. In case of typical BFS time complexity is O(V+E) V-vertices, E-edges. But here the complexity is greater than O(V+E) due to revisits.

4. With parallelization the performance is increased, we can see from above performance table the sequential execution (1 node & 1 core) takes more time when compared to multiple nodes with multiple cores. Thus, even though the time complexity increases due to revisits the good performance is maintained with parallelization.

5. Using this algorithm, it is easier to only find the numeric distance between source vertex and destination vertex. But to keep track of the path from source to destination may be complicated due to the revisits to a vertex.

6. This algorithm may not work correctly for negative weighted edges.

7. In case of classic sequential BFS used for finding shortest path, the algorithm terminates once the destination vertex becomes visited. But the algorithm used here, should not terminate on finding the destination vertex because there can be other paths to reach the destination vertex which have lower edge weights i.e., shorter distance edges.

Source code for BFS shortest path:

```
import org.apache.log4j.Level;
import org.apache.log4j.Logger;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import scala.Tuple2;
import java.util.ArrayList;
import java.util.List;


public class ShortestPath {

    /*
     * This method returns the vertex un each line from graph.txt file.
     */
    public static String getVertex(String inputLine){
        String vertex = "";
        int i=0;
        while (inputLine.charAt(i)!='='){
            vertex = vertex + inputLine.charAt(i);
            i++;
        }
        return vertex;
    }

    /*
     * This method returns vertex data (neighbours, weight, prev, status)
     * attributes from each line of graph.txt file
     */
    public static Data getVertexData(String inputLine, String vertex){
        Data vertexData = new Data();
        String line = inputLine.substring(vertex.length()+1);
        String[] neighbours = line.split(";");
        for (String n: neighbours) {
            String[] nAttributes = n.split(",");
            vertexData.neighbors.add(new
Tuple2<>(nAttributes[0],Integer.parseInt(nAttributes[1])));
        }
        vertexData.distance=0;
        vertexData.prev=Integer.MAX_VALUE;
        vertexData.status="INACTIVE";
        return vertexData;
    }


    public static void main(String[] args) {

        //To print only error logs on terminal
        Logger.getLogger("org").setLevel(Level.ERROR);

        String inputFile = args[0];      //input file containing graph
        String sourceVertex = args[1];   //source vertex
        String destVertex = args[2];     //destination vertex

        SparkConf conf = new SparkConf( ).setAppName( "BFS-based Shortest Path Search" );
        JavaSparkContext jsc = new JavaSparkContext( conf );
        JavaRDD<String> lines = jsc.textFile( inputFile );

        //starting the timer
        long startTime = System.currentTimeMillis();
```

```
        //creating mapPairs->network for JavaRDD->lines
        //reads the graph.txt and creates JAVAPairRDD with vertex->vertexData
        JavaPairRDD<String, Data> network = lines.mapToPair( line -> {
            String vertex = getVertex(line);
            Data vertexData = getVertexData(line, vertex);
            if (vertex.equals(sourceVertex)){
                vertexData.status="ACTIVE";
            }
            return new Tuple2<>(vertex, vertexData);
        });

        //checking number of active vertices
        long activeNodes = network.filter(vertex -> {
            return vertex._2().status.equals("ACTIVE");
        }).count();

        while (activeNodes>0) {

            // If a vertex is "ACTIVE", create Tuple2( neighbor, new Data( ... ) ) for
            // each neighbor where Data should include a new distance to this neighbor.
            // Add each Tuple2 to a list. Don't forget this vertex itself back to the
            // list. Return all the list items.
            JavaPairRDD<String, Data> propagatedNetwork = network.flatMapToPair(vertex->
{
                List<Tuple2<String, Data>> list = new ArrayList<>();
                String vstatus = vertex._2().status;
                int vdist = vertex._2().distance;

                if (vstatus.equals("ACTIVE")){
                    for (Tuple2<String, Integer> n: vertex._2().neighbors) {
                        String nvertex = n._1();

                   // Distance should not be calculated for
                   // source vertex as distance from   itself is 0

                            if (nvertex.equals(sourceVertex)){
                                continue;
                            }
                            int nweight = n._2();
                            int newdistance = vdist + nweight;
                            Data nd = new Data();
                            nd.distance = newdistance;
                            list.add(new Tuple2<String, Data>(nvertex, nd));
                        }
                        vertex._2().status = "INACTIVE";
                    }
                    list.add(new Tuple2<>(vertex._1(), vertex._2()));
                    return list.iterator();
                } );

            // For each key, (i.e., each vertex), find the shortest distance and
            // update this vertex' Data attribute.
            network = propagatedNetwork.reduceByKey( ( k1, k2 ) ->{
                //if there are 2 vertex enteries one with distance 0 and other greater
than 0
                // then the greater one is taken i.e. distance of vertex from source is
assigned.
                if (k1.distance==0 && k2.distance>0 /* && k1.distance<k2.distance*/){
                    int dist = k2.distance;
                    k2 = k1;
                    k2.distance = dist;
                    return k2;
```

```
                } else if (k2.distance==0 && k1.distance>0){
                    int dist = k1.distance;
                    k1 = k2;
                    k1.distance = dist;
                    return k1;
                }
                //if there are 2 vertex enteries with both distnace > 0 then the smaller
one is taken
                else{
                    if (k1.distance<k2.distance){
                        if (k1.neighbors==null || k1.neighbors.isEmpty()){
                            int dist = k1.distance;
                            k1 = k2;
                            k1.distance = dist;
                        }
                        return k1;
                    }else{
                        if (k2.neighbors==null || k2.neighbors.isEmpty()){
                            int dist = k2.distance;
                            k2 = k1;
                            k2.distance = dist;
                        }
                        return k2;
                    }
                }
            });

             // If a vertex' new distance is shorter than prev, activate this vertex
            // status and replace prev with the new distance.
            network = network.mapValues( value -> {
                 // if the distance is not 0 and current distance computed is lesser
                // than previous distance then previous distance is updated and vertex
status is activated
                if (value.distance!=0 && value.distance<value.prev){
                    value.prev = value.distance;
                    value.status = "ACTIVE";
                }
                // if distance is not 0 and current distance computed is greater than
previously computed distance
                // then current distance is changed back to previous distance. Hre the
vertex status is not activated
                // as distance is not changed.
                else if (value.distance!=0 && value.distance>value.prev){
                    value.distance = value.prev;
                }
                return  value;
            });

            //checking for active vertices
            activeNodes = network.filter(vertex -> {
                return vertex._2().status.equals("ACTIVE");
            }).count();
        }

        // printing the shortest distance of destination vertex from source vertex
        for (Tuple2<String, Data> n :network.collect()) {
            if (n._1().equals(destVertex)){
                System.out.println("shortest path: " + n._2().distance);
            }
        }

        //ending the timer
        long endTime = System.currentTimeMillis();
```

```
        long elapsedTime = endTime - startTime;
        System.out.println("Elapsed time:"+ elapsedTime);

    }

}
```

LAB 4:

Set up of spark and executing simple test functions:

```
[poojan26@cssmpi1h lab4]$ ~/spark-2.3.1-bin-hadoop2.7/bin/pyspark
Python 2.7.5 (default, Jun 28 2022, 15:30:04)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2022-11-22 17:02:45 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.1
      /_/

Using Python version 2.7.5 (default, Jun 28 2022 15:30:04)
SparkSession available as 'spark'.
>>> document = sc.textFile( "sample.txt" )
>>> numbers = sc.parallelize( [0, 1, 2, 3, 4, 5] )
>>> len( document.flatMap( lambda s: s.split( ) ).collect( ) )
11
>>> numbers.max( )
5
>>>
```

Execution of MyClass program:

```
2022-11-22 17:07:59 INFO  TaskSchedulerImpl:54 - Removed TaskSet 0.0, whose tasks have all completed, from pool
2022-11-22 17:07:59 INFO  DAGScheduler:54 - ResultStage 0 (count at MyClass.java:22) finished in 0.406 s
2022-11-22 17:07:59 INFO  DAGScheduler:54 - Job 0 finished: count at MyClass.java:22, took 0.449106 s
#words = 11
2022-11-22 17:07:59 INFO  SparkContext:54 - Starting job: max at MyClass.java:23
2022-11-22 17:07:59 INFO  DAGScheduler:54 - Got job 1 (max at MyClass.java:23) with 1 output partitions
2022-11-22 17:07:59 INFO  DAGScheduler:54 - Final stage: ResultStage 1 (max at MyClass.java:23)
2022-11-22 17:07:59 INFO  DAGScheduler:54 - Parents of final stage: List()
2022-11-22 17:07:59 INFO  DAGScheduler:54 - Missing parents: List()
2022-11-22 17:07:59 INFO  DAGScheduler:54 - Submitting ResultStage 1 (ParallelCollectionRDD[2] at parallelize at MyClass.java
:18), which has no missing parents
2022-11-22 17:07:59 INFO  MemoryStore:54 - Block broadcast_2 stored as values in memory (estimated size 1864.0 B, free 366.0
MB)
2022-11-22 17:07:59 INFO  MemoryStore:54 - Block broadcast_2_piece0 stored as bytes in memory (estimated size 1236.0 B, free
366.0 MB)
2022-11-22 17:07:59 INFO  BlockManagerInfo:54 - Added broadcast_2_piece0 in memory on cssmpi1h.uwb.edu:46417 (size: 1236.0 B,
 free: 366.3 MB)
2022-11-22 17:07:59 INFO  SparkContext:54 - Created broadcast 2 from broadcast at DAGScheduler.scala:1039
2022-11-22 17:07:59 INFO  DAGScheduler:54 - Submitting 1 missing tasks from ResultStage 1 (ParallelCollectionRDD[2] at parall
elize at MyClass.java:18) (first 15 tasks are for partitions Vector(0))
2022-11-22 17:07:59 INFO  TaskSchedulerImpl:54 - Adding task set 1.0 with 1 tasks
2022-11-22 17:07:59 INFO  TaskSetManager:54 - Starting task 0.0 in stage 1.0 (TID 1, localhost, executor driver, partition 0,
 PROCESS_LOCAL, 7910 bytes)
2022-11-22 17:07:59 INFO  Executor:54 - Running task 0.0 in stage 1.0 (TID 1)
2022-11-22 17:07:59 INFO  Executor:54 - Finished task 0.0 in stage 1.0 (TID 1). 824 bytes result sent to driver
2022-11-22 17:07:59 INFO  TaskSetManager:54 - Finished task 0.0 in stage 1.0 (TID 1) in 24 ms on localhost (executor driver)
(1/1)
2022-11-22 17:07:59 INFO  TaskSchedulerImpl:54 - Removed TaskSet 1.0, whose tasks have all completed, from pool
2022-11-22 17:07:59 INFO  DAGScheduler:54 - ResultStage 1 (max at MyClass.java:23) finished in 0.039 s
2022-11-22 17:07:59 INFO  DAGScheduler:54 - Job 1 finished: max at MyClass.java:23, took 0.043982 s
max = 5
2022-11-22 17:07:59 INFO  AbstractConnector:318 - Stopped Spark@3676ac27{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2022-11-22 17:07:59 INFO  SparkUI:54 - Stopped Spark web UI at http://cssmpi1h.uwb.edu:4040
2022-11-22 17:07:59 INFO  ContextCleaner:54 - Cleaned accumulator 1
2022-11-22 17:07:59 INFO  MapOutputTrackerMasterEndpoint:54 - MapOutputTrackerMasterEndpoint stopped!
```

Execution of JavaWordCountFlatMap program with single node:

```
2022-11-22 17:26:48 INFO  TaskSetManager:54 - Finished task 0.0 in stage 0.0 (TID 0) in 223 ms on localhost (executor driver)
 (1/2)
2022-11-22 17:26:48 INFO  TaskSetManager:54 - Finished task 1.0 in stage 0.0 (TID 1) in 212 ms on localhost (executor driver)
 (2/2)
2022-11-22 17:26:48 INFO  TaskSchedulerImpl:54 - Removed TaskSet 0.0, whose tasks have all completed, from pool
2022-11-22 17:26:48 INFO  DAGScheduler:54 - ResultStage 0 (count at JavaWordCountFlatMap.java:35) finished in 0.324 s
2022-11-22 17:26:48 INFO  DAGScheduler:54 - Job 0 finished: count at JavaWordCountFlatMap.java:35, took 0.369905 s
#words = 11
2022-11-22 17:26:48 INFO  SparkContext:54 - Invoking stop() from shutdown hook
2022-11-22 17:26:48 INFO  AbstractConnector:318 - Stopped Spark@b5ca6c2{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2022-11-22 17:26:48 INFO  SparkUI:54 - Stopped Spark web UI at http://cssmpi1h.uwb.edu:4040
2022-11-22 17:26:48 INFO  MapOutputTrackerMasterEndpoint:54 - MapOutputTrackerMasterEndpoint stopped!
2022-11-22 17:26:48 INFO  MemoryStore:54 - MemoryStore cleared
2022-11-22 17:26:48 INFO  BlockManager:54 - BlockManager stopped
2022-11-22 17:26:48 INFO  BlockManagerMaster:54 - BlockManagerMaster stopped
2022-11-22 17:26:48 INFO  OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:54 - OutputCommitCoordinator stopped!
2022-11-22 17:26:48 INFO  SparkContext:54 - Successfully stopped SparkContext
2022-11-22 17:26:48 INFO  ShutdownHookManager:54 - Shutdown hook called
2022-11-22 17:26:48 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-ca46d32e-780a-4aaa-a2ba-2af2c52060da
2022-11-22 17:26:48 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-fb36c80c-58b8-43f8-b5ab-d671a94517ee
[poojan26@cssmpi1h lab4]$
```

Execution of JavaWordCountFlatMap program with 4 nodes:

```
2022-11-23 13:05:43 INFO  TaskSchedulerImpl:54 - Removed TaskSet 0.0, whose tasks have all completed, from pool
2022-11-23 13:05:43 INFO  DAGScheduler:54 - ResultStage 0 (count at JavaWordCountFlatMap.java:35) finished in 0.283 s
2022-11-23 13:05:43 INFO  DAGScheduler:54 - Job 0 finished: count at JavaWordCountFlatMap.java:35, took 0.321003 s
#words = 11
2022-11-23 13:05:43 INFO  SparkContext:54 - Invoking stop() from shutdown hook
2022-11-23 13:05:43 INFO  AbstractConnector:318 - Stopped Spark@72efb5c1{HTTP/1.1,[http/1.1]}{0.0.0.0:4042}
2022-11-23 13:05:43 INFO  SparkUI:54 - Stopped Spark web UI at http://cssmpi1h.uwb.edu:4042
2022-11-23 13:05:43 INFO  MapOutputTrackerMasterEndpoint:54 - MapOutputTrackerMasterEndpoint stopped!
2022-11-23 13:05:43 INFO  MemoryStore:54 - MemoryStore cleared
2022-11-23 13:05:43 INFO  BlockManager:54 - BlockManager stopped
2022-11-23 13:05:43 INFO  BlockManagerMaster:54 - BlockManagerMaster stopped
2022-11-23 13:05:43 INFO  OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:54 - OutputCommitCoordinator stopped!
2022-11-23 13:05:43 INFO  SparkContext:54 - Successfully stopped SparkContext
2022-11-23 13:05:43 INFO  ShutdownHookManager:54 - Shutdown hook called
2022-11-23 13:05:43 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-ac6b2aff-d1f5-40ef-a082-bb48dde7d4f9
2022-11-23 13:05:43 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-02038c7d-c6eb-407f-8efa-71d1fe3796c2
```

Source code for JavaWordCountFlatMap program:

```
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.FlatMapFunction;
import java.util.Arrays;
import java.util.Iterator;

public class JavaWordCountFlatMap {

    public static void main(String[] args) {
        // configure spark
        SparkConf sparkConf = new SparkConf().setAppName("Java Word Count FlatMap")
       .setMaster("local[2]").set("spark.executor.memory","2g");
        // start a spark context
        JavaSparkContext sc = new JavaSparkContext(sparkConf);

        // provide path to input text file
```

```
        String path = "sample.txt";

        // read text file to RDD
        JavaRDD<String> lines = sc.textFile(path);

       // Java 8 with lambdas: split the input string into words
       //JavaRDD<String> words = lines.flatMap( s -> Arrays.asList( s.split( " " )
).iterator() );

        JavaRDD<String> words = lines.flatMap(new FlatMapFunction<String, String>() {
            @Override
            public Iterator<String> call(String t) throws Exception {
                return Arrays.asList(t.split(" ")).iterator();
            }
        });

        System.out.println( "#words = " + words.count( ) ); // print #words
    }
}
```