Assignment 3 - PartB

Group8: Xin Jin & Hongyi Wang & Yuzhe Ma

December 4, 2017

1 GraphX: Application 1

1.1 Solution

We run the PageRank algorithm on the Live Journal data(soc-LiveJournal1.txt) and use the Pregel API implemented in Graphx to calculate the page rank.

1.2 Compare the application's completion time

- Completion time of the original application: 33 min.
- Completion time of the new graphx application: 4.7 min.

1.3 Compute the amount of network/storage read/write bandwidth used during the application lifetimes

Table 1: Metrics for disk and network comparison

Performance for sync/async Distributed Training				
Application	Network Rec	Network Trans	Disk Read	Disk Write
Original	11.41 G	11.07 G	0.61 G	24.23 G
GraphX	12.79 G	12.33 G	0.043 G	12.32 G

1.4 Compute the number of tasks for every execution

- The number of tasks for every execution of the original application: 16.
- The number of tasks for every execution of the new GraphX application: 32

1.5 Does GraphX provide additional benefits while implementing the PageRank algorithm? Explain and reason out the difference in performance, if any

It is obvious that using GraphX to implement the PageRank algorithm is much faster. Besides using RDD to do the in memory computation, GraphX provides flexible APIs to construct the graph and do the operations. As PackRank algorithm itself is graph based, after construction the graph with GraphX API, it is efficient to do the computation.

2 GraphX: Application 2

2.1 Find the number of edges where the number of words in the source vertex is strictly larger than the number of words in the destination vertex

First, we read twitter data as vertex and then write a checkCommonWord function to generate the qualified edges. After that, we use the triplets API to calculate the satisfied edges number.

2.2 Find the most popular vertex

Refer to Neighborhood Aggregation example, we use graph.aggregateMessages to count the outcome degree of each vertex. After that, we use a customized cmp function to do the reduce to select the most popular vertex.

2.3 Find the average number of words in every neighbor of a vertex

Refer to Neighborhood Aggregation example, we use graph.aggregateMessages to count the outcome degree of each vertex and the sum of the neighbor words number. After that, we use a reduce function to compute the average neighbor words length.

2.4 Find the most popular word across all the tweets

In order to find the most popular word, we just need to use the vertex to flatmap each word, and then write a cmp function to do the reduce to get the most popular word.

2.5 Find the size of the largest subgraph which connects any two vertexes

By using the connected components of graph, we could labels each connected component of the graph with the ID of its lowest-numbered vertex. After that, we only need a map and a reduce operations to get the size of the largest component of the graph.

2.6 Find the number of time intervals which have the most popular word

First, we use the same way in question4 to find the most popular word. Then we use RDD of vertexes to do a contain selection. After that we could do a reduce to sum the appearance of corresponding time intervals.