Michał Skrzypiec

Big data algorithms. Map-Reduce Model

1. Task 1

First task was to calculate the indegree and outdegree for every node in the graph. Having the graph described as a list of directed edges (node1, node2), the map operation was emitting: (node1, 1) (for outdegree) and (node2, 1) (for indegree). Reduce operation after collecting all elements with simillar key, added their coresponding values, resulting in a degree (in- or out-).

For single operation (calculating either indegree or outdegree), each edge was processed once during map operation. When it comes to nodes, each of them was processed about 8 times (average degree) during edge processing, so about 8 times in total.

For averaging the degree we couldn't just use a map-reduce operation, because there were nodes that didn't have either in- or out-degree and were therefore omitted from the corresponding result, which lost the information about the number of nodes. So we calculated nodes separately and used it to devide the sum of degree obtained with reduce. Calculation of nodes was another processing of every edge.

2. Task 2

Second task was to treat the network as an undirected one and for each node calculate its clustering coefficient. First step was to create an undirected graph. We did so by creating a pairs (Set(node1, node2), 1), where Set(node1, node2) was the key, so we omitted edge repetitions.

Another step was to create a map of a node beeing a key and a set of its neighbours beeing the value (adjecency map). First we mapped every edge (node1, node2) into two pairs (node1, Set(node2)) and (node2, Set(node1). Then reduce operation simply added sets.

Next map-reduce operation was the following:

Map: for every node and its set of neighbours (n_1, S_1)

- take every n_i from S_1 and find its neighbours S_i
- make an intersection of S_1 and S_i (S_r)
- count elements in S_r
- emit a pair (node, $|S_r|$)

Reduce added all elements of every S_r giving therefore number of edges between neighbours of n (times 2, because they were counted twice)

Last map operation was to create requested triples: (node, its clustering coefficient, its degree):

- calculate degree using adjecency map
- calculate clustering coefficient
- create the triple

Every node was processed twice once during creation of undirected graph. Another 2 times by adjecency matrix. Then during first mapping k, because of processing every neighbour of every node. In the end for every node was created a final result. So final number of times each node was processed would be 2 + 2 + k + 1 = k + 5 = ~19

When it comes to calculating the average value, we used a following map-reduce alorithm. Each pair (node, degree) was mapped to (degree, 1), where second variable kept the amount of currently processed elements. Reducing was deviding the sum of both elements, but previously multiplying them by old numbers of processed elements.