Parallel Breadth First Search on Large Graphs

REPORT: Performance Analysis

Prateek Kumar Singh

Introduction:

Graph:

A non-linear data structure made up of vertices and edges is called a graph. In a graph, the edges are the lines or arcs that link any two nodes, while the vertices are occasionally also referred to as nodes. A graph is more strictly made up of a collection of vertices (V) and a set of edges (E). The graph has the prefix G(E, V).

Many problems in everyday life can be resolved with graphs. Networks are depicted using graphs. The networks could be telephone, circuit, or city-wide path networks. Graphs are also employed in social media sites like Facebook and linkedIn. For instance, each individual is represented by a node on Facebook, each node is a structure that houses data like a person's name, gender, location, and other attributes.

Breadth First Search:

Breadth First Search(BFS) is a search strategy that can be used to identify linked graph elements.
Breadth-first search on graphs can be applied when a start node, also known as a search key, is explicitly specified and appropriate measures are taken to avoid following the same vertex again.

Analysis:

The number of reducer tasks is set to 1 in the lab's framework for the k-means clustering.

Why only 1 Reducer?
The task requires us to compare new results with the previous ones and that is why we choose to keep only one Reducer. If we had more than one reducers, there would have been multiple outputs and it would have been cumbersome to carry out the comparisons of the new results with the previous results. Since we would have to merge all the segmented outputs before comparing it to the original.

Disadvantage:
Due to the fact that there is only one reducer, it limits parallelization because it may operate on one or more partitions. Also there is under utilization of the cluster. Using only one reducer makes the re-try very computationally expensive.

Problem with one Reducer:
When the rate at which data is accessed is insufficient to satisfy predetermined system requirements, a performance bottleneck develops.
If the MapReduce operation involves a sizeable quantity of data, a single reducer would in fact constitute a bottleneck.

Solution:
The only strategy that should work in the event of huge data is to employ several reducers because the Hadoop MapReduce engine guarantees that all values associated with the same key are routed to the same reducer.