Large-Scale Graph Processing Project Using Pregel

Jingmin Yu*

School of Information Science and Engineering, Ningbo University, Ningbo, Zhejiang 315211, China
**Corresponding author. Email: 15988873180@163.com*

ABSTRACT
How to compute large-scale data is an important topic. The graph is an important type of data. Pregel is a system that is suitable for processing large-scale graphs. Pregel which sets two states of vertexes determines the program according to the states of vertexes. Because of the partitions of vertexes parted by Pregel, Pregel can sufficiently compute the large-scale graphs. In many traditional graph algorithms, Dijkstra's algorithm is an important shortest path algorithm that can give all shortest distances between the source vertex and other vertexes. The model uses the Pregel framework to implement Dijkstra's algorithm and uses large-scale graphs as input. The results under different conditions such as different cores and the number of vertexes are expressive and reasonable because of the features of Pregel.

Keywords: *graph algorithm, pregel, big data*

1. INTRODUCTION

With the development of the internet, more and more algorithms and technology have been invested to compute large-scale data, such as MapReduce [1] and BigTable [2]. These models provide different frameworks to compute and process large-scale data.

A graph which is an important kind of structured data can be used in high frequency in big data working. The graphs which contain vertices and edges have more different features than traditional graphs [3]. Many graphs' algorithms can solve some specific problems on Graph. For example, the shortest path algorithm which can help users to find the shortest path on a graph is an efficient and important algorithm. Depth-first search and breadth-first search use a specific way to scan a finite and undirected graph. When the depth-first search was used by the public, the algorithm has helped users to thread mazes since the nineteenth century [1], [4].

Although the data that need to be processed become bigger and bigger, these traditional graph algorithms can be useful for large-scale data. Many scientists proposed many new frameworks which support the algorithms to compute the large-scale graph data. Pregel is one of the efficient frameworks which are focused on graph data.

2. BACKGROUND

2.1 Pregel

Pregel which was proposed in 2010 is a system for processing and computing large graphs [5]. Pregel sets two states of each vertex in a directed graph and uses new functions to change the state of each vertex.

The two states of each vertex are active and inactive. Some functions are used to deliver messages between two vertexes. At the beginning of a Pregel program, all vertexes are active. As shown in Figure1, if a vertex does not send and receive any messages, the state of the vertex will change to inactivity.

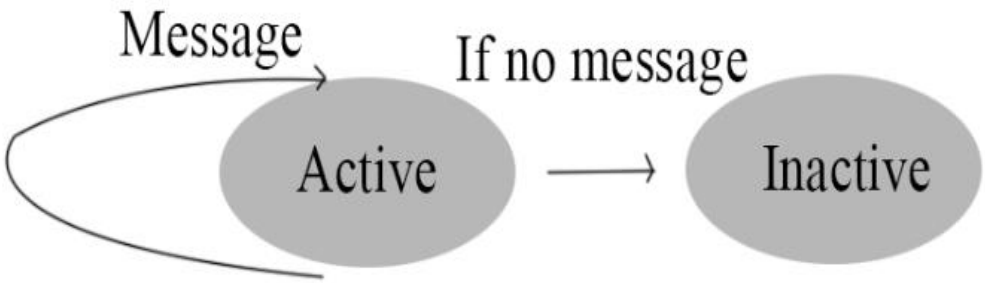

Figure 1: Vertex State [5]

Superstep is a new mechanism in Pregel. In a superstep, every active vertex runs the functions of sending and receiving messages and the functions which are defined by users. User can put their algorithms in the functions which are defined by users. For example, the attitudes of a vertex can be changed. If all vertexes become inactive, the Pregel program finishes.

If users use Pregel to process big graphs, the big graphs have many partitions. Each partition contains a part of edges and vertexes. A cluster of machines can run the copies of the program [5]. There are master machines and worker machines that can run one or more parts of the graph. The basic architecture of a cluster of machines ensures that Pregel can be distributed and parallel.

2.2 Resilient Distributed Datasets (RDDs)

RDD is a distributed memory abstraction that makes programmers perform in-memory computations on large clusters [6]. RDDs have abilities to store and organize large-scale data.

Because iterative algorithms such as graph algorithms produce many intermediate results which can be used in later steps. The traditional methods of storing data are inefficient in these conditions. RDDs become an important data structure.

RDD is a cluster of distributed data. An RDD only can be created by deterministic operations which use on other RDDs. This project uses RDD to store the vertexes, edges, and attitudes of vertexes of the graph.

2.3 Dijkstra's algorithm

Dijkstra's algorithm is a basic one of the shortest path algorithms [7]. Dijkstra's algorithm can evaluate the shortest distances between the source vertex to other vertexes.

Firstly, Dijkstra's algorithm sets the distances which are between two nonadjacent vertexes as infinite, and the distance of the source as zero. Then the algorithm begins to traverse the other adjacent vertexes and update the distances of adjacent vertexes by adding the length of edges as the distances. If the algorithm finds that the distance of the before vertex with the length of the edge is shorter than the old distance of the current vertex, the old distance will be changed. Finally, if all vertexes have their own shortest path, the algorithm can be finished.

3. GRAPH MODEL

This section discusses the main model implementing the shortest path algorithm by using Pregel API.

3.1 main model

Based on the algorithm mentioned in subsection 2.2, the model combines the principle of Pregel and Dijkstra's algorithm.

At the beginning of the program, the program traverses all vertexes in the graph and determines whether all vertexes are inactive. The graph that contains large-scale vertexes and edges uses RDD as the data structure to store and manage the data. If all vertexes are not inactive, the program will begin a new turn of computing. In the process during computing which means a user-defined vertex-program vprog, the active vertexes will send messages to adjacent vertexes. The processing of sending messages is a user-supplied function in the Pregel model.

Then the adjacent vertexes determine whether the distances of current vertexes need to be updated. If the attitude of the vertex needs to be changed, the vertex becomes an inactive vertex and updates its attitude. As shown in Figure2, in the Pregel model, a vertex that does not send or receive any messages means inactivity. Contrasting, the state of a vertex can be remaining and not do other operating. Finally, if all vertexes become

inactive vertexes, the program gets the shortest distance between the other vertexes and the source vertex.

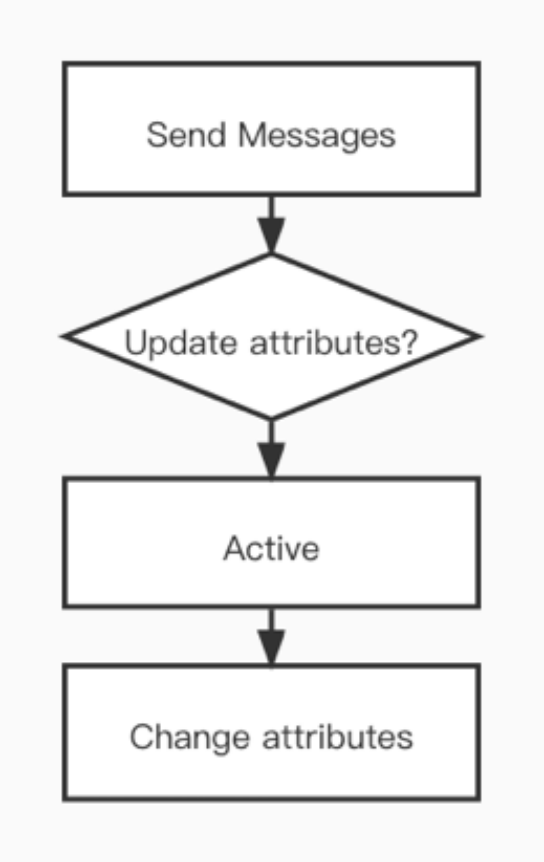

Figure 2: A part of the program

3.2 Implementation

According to the Spark document, there are some classes supporting Pregel programs in GraohX [8]. The program uses 3 operations in Pregel API to achieve the goal of the algorithm.

Table 1 illustrates the meaning and implementation of each operation. At mergeMsg operation, the specific code is like the code of vprog. Because the mergeMsg always is used in the optimization of a program.

Table 1: specific operations

Operation	Meaning	Implementation
vprog	Run on each vertex, receive the messages, and compute a new vertex value	min((long)dst, (long)newdst)

sendMsg	received messages in the current iteration	if(srcAttr + attr) < dstAttr)
		return srcAttr+attr;
		else
		return empty;

mergeMsg	Merge two incoming messages of type A into a single message of type A	min((long)a,(long)b)

4. EVALUATION

4.1 environment

The program runs on a PC that has 2 GHz 4 cores Intel Core i5, 16 GB memory, and MAC operating system.

4.2 results

There are many different results of each running. According to Figure 3, at the same condition, the seconds

have a linear increase with the increase of the number of vertexes. The linear increase illustrates the parallelism of Spark.

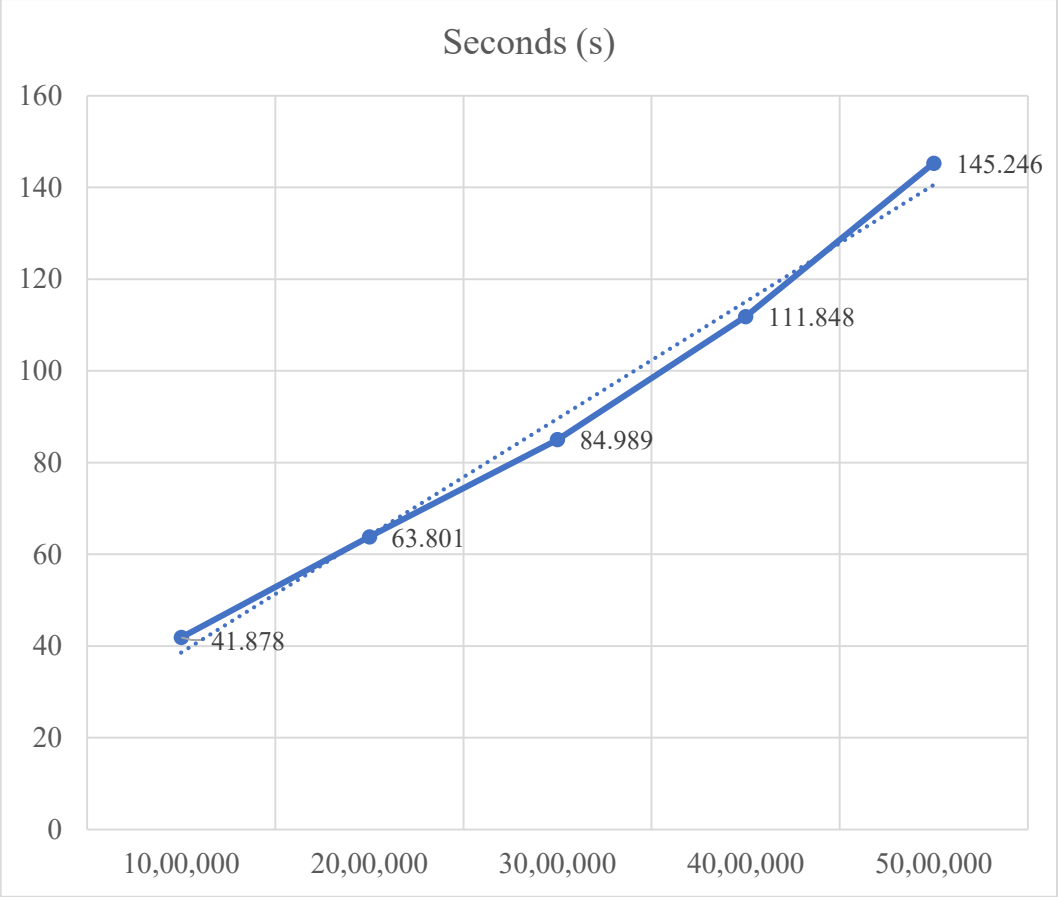

Figure 3: Running results of different graphs with numbers of vertexes at same condition (1 executor, 1 core)

Table 2: running results on different executors and same cores which run on each executor

Vertexes	Executors	Cores	Seconds
1,000,000	1	1	41.878s
1,000,000	2	1	37.229s
1,000,000	3	1	36.767s
1,000,000	4	1	36.749s

According to Table 2, the decrease of seconds does not show a linear change. A possibility of the reason for the unilinear decrease is Standalone mode as the operating mode. In Standalone mode, because of the setting of cores, if the worker has enough cores and memory, multiple executors from the same application can be started on the same worker [8].

5. CONCLUSIONS

The research has used the graph programming model and had a successful result by using Pregel API in GraphX. First, the research has described the model and found that the Pregel is a useful and efficient method for computing large-scale graphs. Second, the research has described Dijkstra's algorithm which is a traditional and important graph algorithm. Third, I have created the program implementing Dijkstra's algorithm by using Pregel. Then I have described and analyzed these experiment results which have understandable phenomena such as a linear increase of running seconds. According to the evaluation, large-scale graphs can be recognized as an important data structure in the big data field. There are many useful methods and frameworks to compute large-scale graph data.

Besides, there are not fully clear results under different executors and cores. The different results are different between our image and experiments. The deep reason for these results needs to do more research. I

expect that more research focusing on executors and cores can achieve shorter computing speed and higher accuracy under limited hardware conditions such as fewer executors and cores.

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," 2004.

[2] F. Chang *et al.*, "Bigtable: A Distributed Storage System for Structured Data," 2006. [Online]. Available: www.cnn.com

[3] W.-K. Chen, *Applied Graph Theory : Graphs and Electrical Networks.* 2014.

[4] S. Even and G. Even, *Graph Algorithms, second edition*, vol. 9780521517188. Cambridge University Press, 2011. doi: 10.1017/CBO9781139015165.

[5] G. Malewicz *et al.*, "Pregel: A system for large-scale graph processing," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 135–145, 2010, doi: 10.1145/1807167.1807184.

[6] M. Zaharia *et al.*, "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing," 2012.

[7] H. Ortega-Arranz, D. R. Llanos, and A. Gonzalez-Escribano, *The Shortest-Path Problem: Analysis and Comparison of Methods*, vol. 1, no. 1. Morgan & Claypool Publishers LLC, 2014. doi: 10.2200/S00618ED1V01Y201412TCS001.

[8] "GraphX - Spark 3.2.0 Documentation." http://spark.apache.org/docs/latest/graphx-programming-guide.html#pregel-api (accessed Nov. 19, 2021).