RESEARCH ARTICLE OPEN ACCESS

A Navel Approach on Big Data with Graph Connectivity Algorithms in Distributed Environment

Ahmed UnnisaBegum [1], Reem Taher Abbas Elaqi [2],
Mohammed Ashfaq Hussain [1], Samar Mansour Hassen [1]
Lecturer Jazan University [1], Student Jazan University [2]

ABSTRACT
Graph Connectivity is the single most persistent characteristic of today's networks environment. Networks with even a modest degree of complexity are not random, which means connections are neither evenly distributed nor static. simple statistical analysis alone fails to sufficiently describe connected systems. Therefore, most big data analytics today do not effectively model the connectedness of real-world systems and have fallen short in extracting value from huge volumes of interrelated data. Therefore, with the increasing of large scale graphs, designing scalable systems for processing and analysing large scale graphs has become one of the timeliest problems facing the big data research community. In general, distributed processing of big graphs is a challenging task due to their size and the inherent irregular structure of graph computations. In this paper, we discuss how can the graphs connected in distributed environment to handling large chunks of the data.
Keywords :- Big data, Graph connectivity algorithms, Distributed data, networks, GPS.

I. INTRODUCTION

Graph analytics, also known as network analysis. It is an exciting new area for analytics capacities. In the business area it has a focus on graph analysis is the ability to use it for social network influencer analysis. Marketing managers in particular are interested in identifying social network influencers because they are potential aim for marketing campaigns trying to trigger chain reactions among social network communities to buy products and services[1]. However, many more potential use cases for graph analysis exist, including several widely used business use cases such as detecting financial crimes fraud detections, Optimizing routes in the airlines, Conducting research in life sciences including medical research, disease pathologies networks as well as helping prevent cybercrime attacks on computer networks and so on[2]. Here we are not talking about visualizations. Graphs are made up of nodes and edges. The mathematical term for nodes is vertices. You may see articles or hear mathematically related people talking about a vertex or vertices and edges. In applying in the real world, nodes or vertices can be people, such as customers and employees, they can be places such as airports, buildings, cities and towns, distribution centres, houses, landmarks, retail stores, shipping ports and so on [3]. Vertices can also be things such as assets, bank accounts, computer servers, customer accounts, devices, grids, molecules, policies, products, twitter handles, URLs, web pages. Edges can be elements that represent relationships such as emails, likes and dislikes, payment transactions, phone calls, social networking and so on. Edges can be directed, that is, they have a one-way direction arrow to represent a relationship from one node to another. Graph analysis may highlight dominant edges. For instance, a large number of payments between bank accounts may indicate money laundering activity. We may begin to identifying communities within a social network and then for one or more specific communities you may want to identify the influencers within that community and the people linking multiple communities [4].

Graph analytics using centrality analysis can be applied in this scenario. Understanding networks can be hugely advantageous, but understanding graph theory is a crucial element in accessing of big data. In this paper we mentioned, "big data" has very profitable information in networks, while algorithms help sort through networks the best. And this article brings to light a very necessary prerequisite in understanding and analyzing networks and understanding graphs[3].

II. HANDLING BIG DATA WITH GRAPHS

Graphs theory is one of the most important and interesting areas of computer science. Graphs have been implemented in real life in a lot of use cases. If you use a GPS on your phone or a GPS device and it shows you a driving direction to a place, behind the scene there is an efficient graph that is working for you to give you the best possible direction. In a social network you are connected to your friends and your friends are connected to other friends and so on. This is a massive graph running in production in all the social networks that you use[5]. You can send messages to your friends or follow them or get followed all in this graph.

Social networks or a database storing driving directions all involve massive amounts of data and this is not data that can be stored on a single machine, instead this is

distributed across a cluster of thousands of nodes or machines. This massive data is nothing but big data. Big data comprises huge amount of data distributed across a distributed across a cluster of machines. Hence, in actuality, it's not a single node graph and we have to build a graph that spans across a cluster of machines and also a graph that spans across a cluster of machines would have vertices and edges spread across different machines and this data in a graph won't fit into the memory of one single machine[6]. Consider your friend's list on Facebook; some of your friend's data in your Facebook friend list graph might lie on different machines and this data might be just tremendous in size.

The following are the areas where graph analytics is extremely useful and used frequently.

Path analytics: As the name suggests, this analytics approach is used to figure out the paths as you traverse along the nodes of a graph. There are many fields where this can be used—simplest being road networks and figuring out details such as shortest path between cities, or in flight analytics to figure out the shortest time taking flight or direct flights [7].

Connectivity analytics: As the name suggests, this approach outlines how the nodes within a graph are connected to each other. So using this you can figure out cluster of thousands of machines. Building graphs based on this massive data has different challenges such as, due to the vast amount of data involved, the data for the graph is how many edges are flowing into a node and how many are flowing out of the node. This kind of information is very useful in analysis. For example, in a social network if there is a person who receives just one message but gives out say ten messages within his network then this person can be used to market his favorite products as he is very good in responding to messages.

Community Analytics: Some graphs on big data are huge. But within these huge graphs there might be nodes that are very close to each other and are almost stacked in a cluster of their own. This is useful information as based on this you can extract out communities from your data. For example, in a social network if there are people who are part of some community, say marathon runners, then they can be clubbed into a single community and further tracked.

Centrality Analytics: This kind of analytical approach is useful in finding central nodes in a network or graph. This is useful in figuring out sources that are single handedly connected to many other sources. It is helpful in figuring out influential people in a social network, or a central computer in a computer network

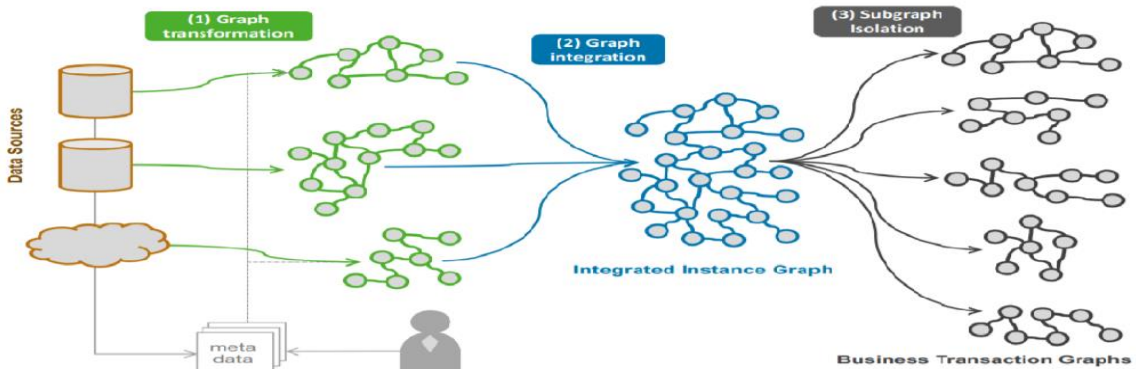

Fig1 : Data Integration and Analysis Frame work

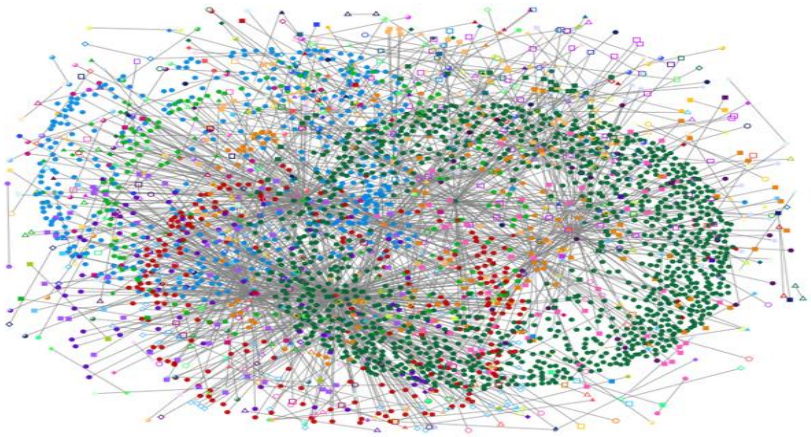

Fig 2: The Massive collection of nodes in Big data

III. GRAPH CONNECTIVITY ALGORITHMS FOR BIG DATA

Big data architecture includes ingesting, protecting, processing and transforming data into file systems or database structures. Analytics tools and queries can be run in the environment to get intelligence from data, which outputs to a variety of different vehicles. A graph is a mathematical structure comprising of nodes or vertices, connected by edges. While the nodes represent the different entities of the system, the edge is illustrative of the relationship between them. However, if graphs are extrapolated to the context of data sciences, they are rather powerful and organized data structures. These, in turn, represent complex dependencies in the data. Graph analytics is used to model pairwise relationships between people and/or objects in any system. This would help one in generating insights into the strength and direction of the relationship. The edges are the more critical component, might connect nodes to other nodes or its properties[8].

In graph processing frameworks, we distinguish between partitioning techniques based on three classifications. First, does the input require a preprocessing step or does the algorithm work on On-streaming. On-streaming graph partitioning reduces overhead by limiting computation to a single pass, while a preprocessing step is impeded less by time and other resource restrictions. Several frameworks that optimize for out-of-core processing require a preprocessing step to convert input into an optimized storage format. TOTEM partitioning uses the degree distribution of the input graph, which requires an extra pass over the data. This process is not the input data, but the network architecture to enable optimized partitioning. Although not required, on-the-fly algorithms can still benefit considerably from a preprocessing step. Salihoglu and Widom (GPS) report an improvement in run time by up to 2.5x using this approach[9]. Preprocessed data can be used for multiple runs. Generally, the surveyed frameworks work with a simple streaming partitioning method to minimize the overhead of data loading. Typical algorithms for this purpose are random, range, and round-robin partitioning. A simple abstraction, i.e. hash(key) mod R, where R is the number of partitions and hash is a user-defined function, allows for a certain degree of customization. Using the identity function for hash results in a round-robin partitioning, while using a cryptographic hash function results in a more random partitioning. The second distinction in partitioning techniques for graph processing is made by the support for dynamic repartitioning during execution, rather than using the same static distribution from start. Runtime behavior of an algorithm can be unpredictable, so using an adaptive partitioning method can improve performance. As there is extra communication required for reassigning data, it's only beneficial for applications with more than a few iterations[10]. Note that dynamic repartitioning is implicit for load balanced Dataflow frameworks that do not separate static and state data.

Finally, we concludethat the graph frameworks facilitate efficient analysis on data streams. We notice that a lot of frameworks focus on the volume aspect of big data in distributed environment, but ignore the challenge in velocity. A lot of data is constantly changing and continuous bulk analysis of the entire input causes significant overhead, especially when the majority of the result does not change.

IV. CONCLUSION

Big data is inevitable for any modern organization. As big data analytics can help companies create better value for their customers and themselves. But relying on the traditional data management and analytics tools is not enough, as data-points continue to develop and grow in the interconnected world. To make the most out of the interconnected nature of things, you should consider using

graph databases.Finally, we identified and presented a set of the current open research challenges and also presented some of the valued directions for future research. In general, we trust that there are still many opportunities for new innovations and optimizations in the domain of large scale graph processing. So, in a world of interconnected data, understanding the value of data alone is not enough, you also need to look for the different ways things connect with each other.

REFERENCES

[1] M. Becker, W. Degenhardt, J. Doenhardt, S. Hertel, G. Kaninke, and W. Keber, "A probabilistic algorithm for vertex connectivity of graphs," Inf. Proc. Letters 15 (1982), pp. 135-136.

[2] S. Even, "An algorithm for determining whether the connectivity of a graph is at least k," SIAM Journal of Computing 4 (1975), 393-396.

[3] Big Data Spatial and Graph User's Guide and Reference. http://docs.oracle. com/cd/E69290 01/doc.44/e67958/toc.htm, accessed: 2016-03-16.

[4] Gelly: Flink Graph API. https://ci.apache.org/projects/flink/ flink-docs-master/apis/batch/libs/gelly.html, Accessed: 2016-03-15.

[5] Cheng R. et al.: Kineograph: taking the pulse of a fast-changing and connected world. In: Proc. EuroSys (2012).

[6] W. Fan, J. Li, S. Ma, N. Tang, Y. Wu, and Y. Wu, "Graph pattern matching: from intractable to polynomial time," Proc. VLDB Endow., vol. 3, no. 1-2, pp. 264–275, Sep 2010.

[7] A. Fard, M. U. Nisar, J. A. Miller, and L. Ramaswamy, "Distributed and scalable graph pattern matching: Models and algorithms," International Journal of Big Data (IJBD), vol. 1, no. 1, 2014.

[8] Gonzalez J. E. et al.: GraphX: Graph Processing in a Distributed Dataflow Framework. In: Proc. OSDI (2014).

[9] H. Nagamochi and T. Ibaraki, "Computing edge connectivity in multigraphs and capacitated graphs," SIAM J. Disc Math 5 (1992), pp. 54- 66.

[10] A. Fard, S. Manda, L. Ramaswamy, and J. A. Miller, "Effective caching techniques for accelerating pattern matching queries," in Big Data (Big Data), 2014 IEEE International Conference on. IEEE, 2014, pp. 491–499.