Volume 6, Issue 1, January 2016 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Review on Big Graph Partitioning for Recommendation Systems

Chitra Setia*, Supriya Panda

Computer Science Department, The North Cap University, Gurgaon,
Haryana, India

Abstract— With the rapid and continuous increase in data graphs, it has become very difficult in today's scenario to manage, organize, and analyze patterns from them. Hence there are various methods like triangulation method, partitioning methods, etc which are implemented so as to split the data into clusters based on locations, likes and dislikes etc. and run them on various parallel machines. Also there are several streaming algorithms, which can be used if the vertices are not present on the machines and are coming in as a continuous stream. In this paper, we have tried to illustrate some of these methods to develop recommendation systems, which can be used to enhance the business based on several inputs and recommendations of multiple people on any product like watches, books, movies etc thereby increasing the revenue of the companies and also benefit the customers to determine the quality and usefulness of the product and other important features.

Keywords— Big Graphs, Big graph databases, Triangulation method, k-mutual friend subgraph, Streaming.

I. INTRODUCTION

The word Big graph has become a buzzword now-a-days as the magnitude and scalability of the data both structured and unstructured is increasing at a very fast pace. The machines have started accumulating data from various resources like satellites which are monitoring the earth and taking pictures 24-hours a day, social networking sites collecting trillions of data, cloud services, sensors, RFID, etc. Thus the progression has started from employees generating data to users and machines generating data and it is estimated that by the end of 2020, we will reach the end of the English alphabet that is the Zeta Bytes era. Big data can be summarized fully with the help of 5Vs properties; Volume, Variety, Veracity, Velocity and Value. Thus a big graph can be said as a complex interconnected topology just like the human brain, which possesses trillions of neurons linked together through synapses. Similarly, the big graph can be implemented in social networks also called as the social graph. For example, Twitter, Facebook etc., wherein we can visually identify the relationships among nearly 1- billion users. As the data is increasing continuously, the first major problem that arises is how to collect, organize, analyse and process this tsunami of information. The second important issue is how to visually identify the relationships and the interconnections between huge amounts of data and the individuals generating this data. For example, on social networking sites like Facebook which is one of the most widely used social networking service in the world with over 800 million people using the service each month [1].

Hence the solutions comes up with the emerging new tools like Apache Hadoop; a scalable, open source, fault-tolerant Virtual Grid operating system architecture for data storage and processing and uses HDFS (Hadoop Distributed File System) which is fault-tolerant, high-bandwidth clustered storage architecture, [4] along with powerful parallel framework called MapReduce which is carried on in 4 steps. Split function which partitions the data to distribute among several nodes and through them a set of key/value pairs are generated called map along with shuffle process which merges the pair with the same key. Finally the reduce function summarize and process the results into the final answer and then output the result into HDFS (Hadoop Distributed File System) [5]. Moreover, second problem can be solved by exploiting the most simple and extensively used data structure called **graphs** and the graph databases are used to manage the highly interconnected data just like the key-value pair. Several tools like Neo4j;one of the world's leading graph database, GraphLab ,Pregel, HypergraphDB and many other graph databases are provided in order to identify, analyze and visualize the relationships and works with a flexible network structure of nodes and relationships rather than static tables. In this paper, we will summarize how to create a social graph and explore various graph partitioning algorithms to split data into cohesive subgraphs and analyse how closely they are related.

II. THE BASICS OF BIG GRAPH DATABASES

A. Graphs

The graph theory has been widely exploited in big data graph analytics wherein the graph can be defined as an ordered pair G = (V, E) where V denotes vertices or nodes like World Wide Web. pages, individuals, their id's, etc., and E signify the edges or the lines connecting the nodes and representing the properties and information related to the nodes like the hyperlinks, wherein the link structure of a website can be represented as a directed graph in which the vertices represent the web pages and the directed edges represent links from one page to another. Moreover, they also represent the relationships in between the nodes. For example, A and B are 2 nodes representing 2 friends and both are connected by an edge Label: "Knows" that signifies A knows B. But in order to manage approximately one trillion edges graph, the

graph databases are used to capture the relationships among the entities, especially in a social graph, where the vertices are considered as nodes and edges as relationships between actors.

B. Subgraph

Generally, a subgraph S of a graph G can be defined as in where S belongs to G i.e., where set of vertices V and edges E are all subset of G and the k-mutual friend Subgraph can be defined as a subgraph which is connected such that each and every edge is supported by at least k -pairs of edges forming a triangle with that edge within G.

C. Cohesive Subgraph

After identifying the set of edges and vertices, the major step is to identify the cohesive groups and split the data across large clusters which will then be called as a cohesive subgraph and these cohesive subgraphs are then processed in parallel on different processors by using various key value pairs along with several online and offline algorithms.

III. VARIOUS GRAPH MODELS

The k –mutual friend subgraph mentioned above is actually derived from clique problem and the k-core.

A. Clique Model

The clique is actually a connected graph where the restriction is that all the edges and the vertices must be fully connected to each other and the clique problem is related to finding subgraphs in a graph where each pair of elements that is every vertex must be fully connected to each other by an edge.

B. k-core

- The k-core model or **the degenerate graph** is actually a relaxation of the clique problem and fits to the real life scenario as each node or specifically all the individuals cannot be connected to each other. Hence k-core, where value of k relative minimum size of degree of each vertex, is a maximal connected subgraph of G in which all vertices have degree at least **k**, i.e., maximal connected subgraph.
- The k-core model can be illustrated diagrammatically in the following Fig.1.

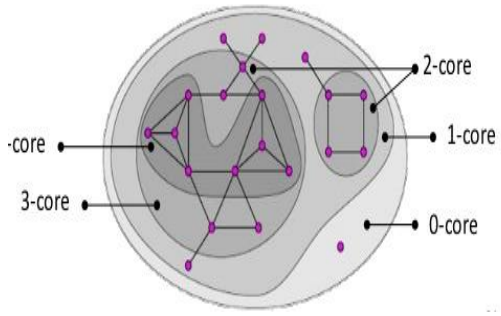

Fig. 1 Example of a k-core model

C. DN-Graph

There is also a spectacular dependency between the mutual friend concept and DN (dense neighbourhood) graph. A DN graph G′(V ′,E′ , λ) is a connected subgraph of G′(V ′,E′) of graph G(V,E) that must satisfy the following 2 conditions:

1) Each and Every pair of vertices must share at least, λ common neighbours. This is similar to k-mutual friend subgraph model but the second condition makes it different [2].

2) For any v ∈ V \V ′, λ (V′ U (union) {v}) < λ; for any v ∈ V ′, λ (V ′ − {v}) <= λ. Thus the restriction is imposed that in order to insert or delete just one vertex, the maximal subgraph must necessarily attain the local maximum that is the between any connected vertices, the lower bound λ must be maximized locally. But this sometimes leads to redundant graphs, which increases the overall complexity and hence is not suitable for processing large graphs [2].

D. k-plex

The k-plex model can be defined as a maximal subgraph in which each and every member or vertex of the subgraph must be connected to at least n-k other members and vertices.

IV. GRAPH V/S RELATIONAL DATABASES

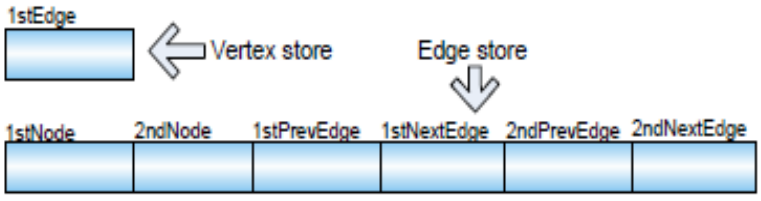

Fig.1 Graph Database Structures

The graph structures are very much beneficial, in spite of the fact that it deals with the highly interconnected data; they also have the ability to run 150 times faster as compared to the relational databases. This is because it uses a technique called Index free Adjacency wherein each node maintains direct references to its adjacent nodes such that all the inbounds and out bounds directly points to the nodes which by some means are related to each other. Thus its speed increases automatically as query times depends upon the number of traversals and the amount of graph searched irrespective of the size of the data. Hence as shown above in Fig. 2, there are 2 storage primitives: vertex store and edge store.

A. Vertex Store

Vertex Store can be defined as the store wherein each vertex can be represented as one integer which is the offset of the first relationship this node participates in.

B. Edge Store

We use Edge store can be defined as the store in which each edge can be represented with the help of six integers. The first integer actually represents offset of the first vertex and the second integer represents the offset of the second vertex. The remaining four integers are in the sequence like the offset of the previous edge of the first vertex, the offset of the next edge of the first vertex, the offset of the previous edge of the second vertex and finally the offset of the next edge of the second vertex. These 6 integers are represented in the form of doubly linked list in the memory because this model possesses a valid benefit.

Also, Table I below summarizes some more differences between Relational databases and graph Databases in.

TABLE I DIFFERENCE BETWEEN RELATIONAL AND GRAPH DATABASES

Index	**Data Storage**		
	Relational Databases	**Graph databases**	**Comments**
1	A relational database has to do lot of calculations to analyze how things are connected.	Graph Databases are much faster than relational databases because of the index free adjacency.	Graph databases are much faster and better
2	Various JOIN operations are needed which are very expensive.	They don't require any Join operations and can manage large datasets naturally with high flexibility.	Graph Databases are better as no JOIN operations
3	In relational databases, data is stored in a tabular fashion with rigid data types.	The graph databases are less dependent or nearly independent on the rigid schema and hence can easily adapt the evolving schemas	The graph databases are better as they are independent of any schemas.

In today's scenario, almost 80% (800 million) of the world's population has facebook and twitter accounts through which people are socially connected to each other directly or indirectly and having lot of mutual friends[2]. Thus the mutual friend structure is used to capture the tie strength in a quantitative manner for social network analysis.

V. GRAPH TRIANGULATION METHOD

Now the major problem that arises is how to find the best possible solutions so as to discover the maximal k-mutual friend subgraph. One method could be to use graph triangulation such that if there are 2 vertices connected to each other and sharing one common neighbour then a triangle will be formed with one common neighbour. Hence in a similar fashion, several triangles can be formed and the all the remaining ones which are not involved in any triangle formation will be considered as unwanted and unsatisfied edges for finding mutual friends or implementing friends of friends query. However this method is simple and straightforward solution, but the complexity of computing such triangles is very high as the worst case of this computation is that it eliminates one edge at a time and hence the number of loops required for calculation of complexity is |E|. Thus we can imagine that this method is nearly impossible for a dense graph. Thus an improved algorithm is proposed based on an observation that when we delete an edge, only the triangle counts of the edges which are forming the triangles with that edge are decreased. So we can find all the affected edges and then decrease the triangle counts for them. Thus it eliminates the useless iterations, but the flaw remains the same and that is it is not suitable for large scale graph processing as we cannot store such huge data on the disk. It needs O (|E|) space complexity.

VI. GRAPH PARTITIONING METHOD

Another solution to analyze and extract useful information and find patterns from trillions of URL's and online social networks like Facebook having trillions users which comes up as a saviour is to split or partition the data into large clusters and then parallel and distributed algorithms can be implemented for the computations. This method is actually called as balanced graph partitioning. The main objective of this method is to reduce the number of cross partition edges with the constraint that the number of nodes in each partition must be approximately even. Partitioning the graph into clusters is a very standard solution as in real life scenario the graphs are not random at all. They may be related to each other either geographically or by any other domain likes some Nobel causes which each member support etc. Thus it assures that the balanced partitions will be far better than the random cuts. Although it is a standard approach, but the main question is how to move such large amounts of data; gigabytes or terabytes from one network to another because this communication is largely expensive and causes the network links to become saturated.

A. Machine and Data Graphs

In order to find and measure inconsistency of the network bandwidth, a machine graph is designed where the machines represent the vertices, the connection between the 2 machines are represented by the edges and the weights on the edges represent the bandwidth between the two machines. Also it is assumed that all the machines possess equal computational power and memory in order to keep the concept simple. The bandwidth of the network between 2 machines can be measured by sending the data of 8 MB. And the computations will be done based on the iterations as if there are N machines then N iterations will be needed and for every iterations (N/2) machines are measured efficiently, for example if there are 4 machines, then they are divided into 2 machine in each pair and then the one with higher bandwidth can be demonstrated by a thick edge in the model. Now as the machine graph is ready a partition sketch can be built where the root represents the input graph and the leaf nodes represents the graph partitions which have been developed by various algorithms. This partition or bisection is done with the help of 3 steps:

1) Coarsening: Coarsening is done in which the graph becomes compact and results in a smaller graph with each iteration.

2) Partitioning: The next step is partitioning which splits the graph further into 2 parts using GGGP (greedy graph growing partitioning)

3) Uncoarsening: Uncoarsening is just the reverse process of coarsening process.

Thus, firstly the graph must be stored in a machine by using a simple hash function and then bisection must be done keeping in mind the machine and the data graph must be mapped by keeping the network bandwidth saturated.

Hence a Graph Loader is needed which performs 2 functions:

1) It is used to load or read the serial graph data onto a cluster from the memory.

2) Moreover, it helps in decision making so as to find the location or decide the cluster where the node is to be loaded.

Thus the balanced graph partitioning guarantees low communication overhead among different machines as each partition contains $\approx n/k$ vertices, where n, k are the total number of vertices and machines, respectively[3]. But, the question is what if no graph is present in memory to be partitioned. This is called as streaming based partitioning. The term streaming can be defined as the process, where the data is not stored or available anywhere on the disk like the metadata and is processed just as a steady and continuous stream dynamically. Hence streaming partitioning is required when the entire graph is not present in memory and the nodes needs to be processed as and when they arrive. Thus a streaming graph model can be defined with cluster of k machines, each with memory capacity C, such that total memory capacity, kC is large enough to hold the graph.[3] The vertices and the edges reach in the form of a stream and the partitioner decides the particular machine where each vertex needs to be placed. Although the decision may take some time, hence a buffer of size C is introduced to place the arriving vertices which by some reason have to wait to be processed.

Let us consider that the v is the vertex that arrives at time t in the stream. |S| denotes the total elements in the set and the capacity constraint C is applied on each partition. Thus there are several heuristics algorithms which can be used to choose the index **ind** of the partition.

1) Balanced: Select a partition of minimal size and then assign v to it.

2) Chunking: The stream is first divided into chunks of size C and then the partitions are occupied in the order: ind = [t/C].

3) Hashing: Hashing is the method which is used in several real time systems This method helps us to locate the vertices very quickly and efficiently, the hash function used to find the index ind is $H(v) = v \pmod k + 1$.

4) Deterministic Greedy: The v is assigned to that partition where maximum edges are present.

5) Balance Big: It is actually used to differentiate between the high degree and the low degree nodes, if v vertex which is coming is of high degree then balanced is used else Deterministic Greedy is used.

6) Prefer Big: Let us consider a buffer of size C. Wherein all the high degree nodes are first chosen using balanced and then more nodes are streamed in to be kept in the buffer. If low degree nodes are present in the buffer, then remove it by using Deterministic Greedy.

7) Avoid Big: It is completely opposite to that of the Prefer Big where when the high degree nodes are nodes are present in the buffer, they are removed using deterministic greedy.

8) Greedy Big: In this process, all the small nibbles (1 NB = 4 bits) are found and then a partition is selected for each nibble using Deterministic Greedy.

Also 3 stream orders are considered namely BFS, DFS and Random:

1. BFS (Breadth First Search): Breadth first search or BFS is created by first choosing a start node from each of the connected component and then traverse in the order of the BFS.

2. DFS (Depth First Search): this ordering will be same as BFS but the DFS will be used instead of BFS.

3.Random: The random ordering is a standard stream ordering for theoretically analyzing algorithms in which all the vertices will come in a way which is given by the permutation totally random.

VII. IMPLEMENTATION

We can implement all these above heuristics as well as the stream orders based on the requirements for example: we can build a graph based recommendation system which can be used increase the revenue and enhance the business according to needs of the people by analyzing patterns. Also customers can use these reviews and feedbacks from various friends and people so as to determine the quality of brands and decide which are better and advantageous to them. For example: The recommendation system built in the below figure which are distinguished among clusters with several brands in the market like Titan, Casio, Fastrack etc which are then being generalized based on the age groups among customers which are either visiting or buying it. Generally the recommendation systems can be demonstrated into 3 categories; content based collaborative based and hybrid filtering [6].

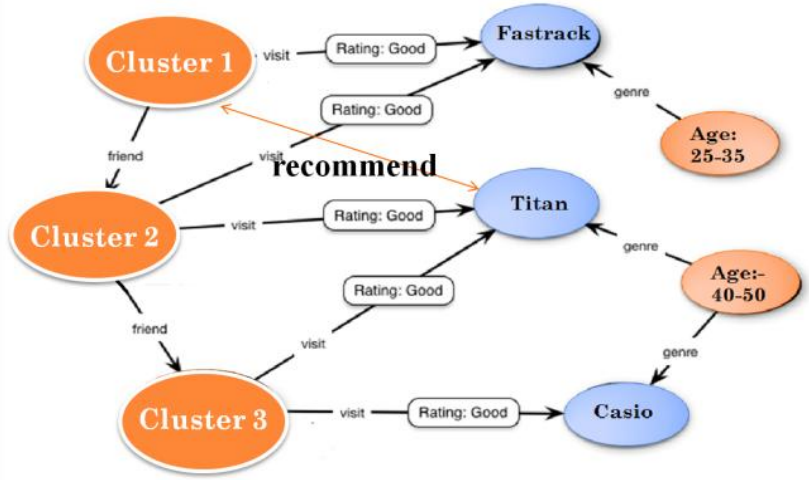

Fig.3 Feedback Systems

VIII. CONCLUSION

In this paper, we have elaborated how the graphs and graph databases are more advantageous as compared to the relational databases. Also various theories and methods have been defined like friends of friends or the mutual friends, cohesive subgraph theory etc. to determine the tie - strength. Moreover several partitioning and streaming partitioning methods are described , which can be implemented in feedback systems. Hence in future a framework can be designed, which uses facebook interface or the friends of friends query and the partitioning methods to divide the vertices into clusters and use the feedbacks and the reviews from customers in order to test the products before purchasing.

REFERENCES

[1] Bakshy, Eytan, et al., The role of social networks in information diffusion, Proceedings of the 21st international conference on World Wide Web. ACM, 2012.

[2] Zhao, Feng, and Anthony KH Tung,Large scale cohesive subgraphs discovery for social *network visual analysis*, Proceedings of the VLDB Endowment 6.2 (2012): 85-96.

[3] Stanton, Isabelle, and Gabriel Kliot, Streaming graph partitioning for large distributed graphs, Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012.

[4] Manikandan, Shankar Ganesh, and Siddarth Ravi. "Big Data Analysis Using Apache Hadoop." *IT Convergence and Security (ICITCS), 2014 International Conference on*. IEEE, 2014.

[5] Yang, Juan, et al. "The research and design of parallel recommendation algorithm based on mapreduce." *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*. Vol. 1. IEEE, 2012.

[6] Stanescu, Ana, Swapnil Nagar, and Doina Caragea. "A Hybrid Recommender System: User Profiling from Keywords and Ratings." *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*. Vol. 1. IEEE, 2013.

