Implementing Graph Pattern Mining for Big Data in the Cloud

Chandana Ojah
M.Tech in Computer Science & Engineering
Department of Computer Science & Engineering,
PES College of Engineering, Mandya
Ojah.chandana@gmail.com

Dr. MC Padma
Professor & HOD,
Computer Science & Engineering,
PES College of Engineering, Mandya
padmapes@gmail.com

Abstract – With the increasing popularity of various social networking sites, there is an explosive growth in data associated with these, so mining big data has become an important problem in the graph pattern mining research area. Graph mining helps to explore the patterns from networks or databases. Till now various graph mining techniques exist for mining frequent patterns for a graph database which contains relatively small sized graphs. But with the rapid arrival of the era of big data, traditional graph mining approaches have been unable to meet large data analysis needs. In this context, this paper proposes an adaptation to the big graph data mining approach especially in the field of social networks. The proposed approach is based on Hadoop plateform, and improves the efficiency by processing big data in distributed fashion. Again the proposed approach can be adapted to cloud environment which has the merits – load balancing, scalability and efficiency. Experiments have been conducted with real Facebook data set. The approach can be also adapted to dataset larger than experimented data.

Index Terms – *Big Data, Cloud computing, social network mining, Graph pattern mining*

*****

I. INRODUCTION

Recently, Big Data has become a very popular term in the whole IT industry. Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time. Big data has 4V characteristics-Volume, Variety, Velocity, and Value. One such field in Big Data is big graph data. Here, the data forms the shape of a graph or network. The data are connected to each other such as in various social networks (Facebook, Google+, twitter etc.) where edges represent interactions between people, communication networks (email communication networks with edges representing communication), citation network (nodes represent papers, edges represent citations), Amazon networks (nodes represent products and edges link commonly co-purchased products) and many more. Graph mining has been studied to explore the patterns from networks or databases. Graph mining has been studied to explore different patterns from networks or graph databases. In recent years, many applications have applied this method, such as [1], to discover infrastructure patterns in management database. Also it has been used in biomedical data such as in [2]. However, these approaches have limitations in supporting big graph data due to complexity of the problem. Experiments in [3] and [4] shows that numbers of nodes supported is approximately 1000. SUBDUE [5] and SPIDERMINE [] has demonstrated scalability, but they are not based on cloud computing environment.

The objective of our paper is to develop an approach for finding frequent patterns, both small and large patterns from large dataset from Facebook in Hadoop framework in the cloud. These patterns will be based on different features or attributes or personal data related to Facebook users such as birthday, work, education type etc. We can use these patterns for various further analysis such as studying the patterns to know students from which institutes are mostly related, employees from which organisations are mostly related etc. The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Use of cloud services eliminates our need to invest in hardware up front, so we can develop and deploy applications faster. We can launch as many or as few virtual servers as we need, configure security and networking, and manage storage. Also it enables us to scale up or down to handle changes in requirements or spikes in popularity, reducing our need to forecast traffic.

More specifically, this paper presents a cloud implementation, which selects some initial points from the graph to start the pattern mining approach. This algorithm uses Breadth First Search (BFS) approach for pattern growth as it is more efficient comparing to other graph traversing approaches. Also it is complete as compared to Depth First Search. Our approach consists of mainly 3 phases, Mining phase, Isomorphism Test phase and Pattern Pruning phase. The first phase of our approach is the pattern or sub-graph mining phase. Here we first mine patterns using BFS. Then we calculate the support count for each pattern. Support count is nothing but the total number of occurrences of the pattern or sub-graph in the large graph. The support count of each graph is required in the subsequent phases. After pattern mining phase, next phase is pattern isomorphism test phase. When we get the first single-edged patterns, it may happen that two patterns may be same. (E.g. let us deriving our patterns based on work-organisation. So, IBM-TCS and TCS-IBM is the same pattern). So, these two patterns are not two different patterns. The two are same pattern. So we have to increase the count of this pattern when we find two patterns as isomorphic. It is the last phase of our framework. Actually all the phases keep executing continuously step by step. After getting the initial single-edged patterns or sub-graphs, isomorphism test is conducted on the resulted patterns. The isomorphism test

results in increasing the count for those patterns which are isomorphic and the count of remaining patterns remains the same. As there is a minimum support count or threshold is attached with our approach, we will choose those patterns which have support count greater than the minimum support count or threshold.

II. RELATED WORKS

The related work forms three groups, graph pattern mining, and HADOOP and Amazon Web Services.

A. Graph Pattern Mining

Frequent graph pattern mining is a popular topic in graph pattern mining field. Before, emphasis was given to mining frequent patterns from a graph dataset, which contains different small sized graphs. But due to the increase in popularity of single large graphs such as the Web graph, Facebook graph, Twitter graph etc. has drawn attention to frequent pattern mining of single large graph. SUBDUE [5] can be considered as the most famous graph pattern mining approach for a single large graph. SUBDUE has the ability to discover substructures, compress the database, and represent concept of structure, but it is computation intensive. SEuS [3] and MoSS [4] are useful for smaller sized graphs, but they are not suitable for larger sized graphs. There have also been a few graph mining approaches developed on the cloud in past years. As in [6], authors introduced Mizan, which provides efficient fine-grained vertex migration to balance computation and communication. Gbase [7] provides parallel indexing mechanism for graph operations that both saves storage space and query responses. In c-SPIDERMINE [9], authors have introduced the top-k pattern mining approach in the cloud.

B. MapReduce and Hadoop

The Apache Hadoop [10] software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. MapReduce [11] is the heart of Hadoop. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The MapReduce concept is fairly simple to understand for those who are familiar with clustered scale-out data processing solutions.

C. AWS and EC2

Amazon Web Services (AWS) [12] is a collection of remote computing services, also called web services that make up a cloud computing platform offered by Amazon.com. These services are based out of 11 geographical regions across the world. The most central and well-known of these services are AmazonEC2 and Amazon S3. These products are marketed as a service to provide large computing capacity more quickly and cheaper than a client company building an actual physical server farm. Amazon Elastic Compute Cloud (EC2) [13] is a central part of Amazon.com's cloud computing platform, Amazon Web Services (AWS). EC2 allows users to rent virtual computers on which to run their own computer applications. EC2 allows scalable deployment of applications by providing a Web service through which a user can boot an Amazon Machine Image to create a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server instances as needed, paying by the hour for active servers, hence the term "elastic".

III. PROPOSED APPROACH

This section presents the details of our proposed method to mine frequent patterns from a large graph. Our experimental data is already partitioned, i.e. the single large graph has been partitioned into several sub-graphs. The framework for our proposed method can be shown as in figure 1.

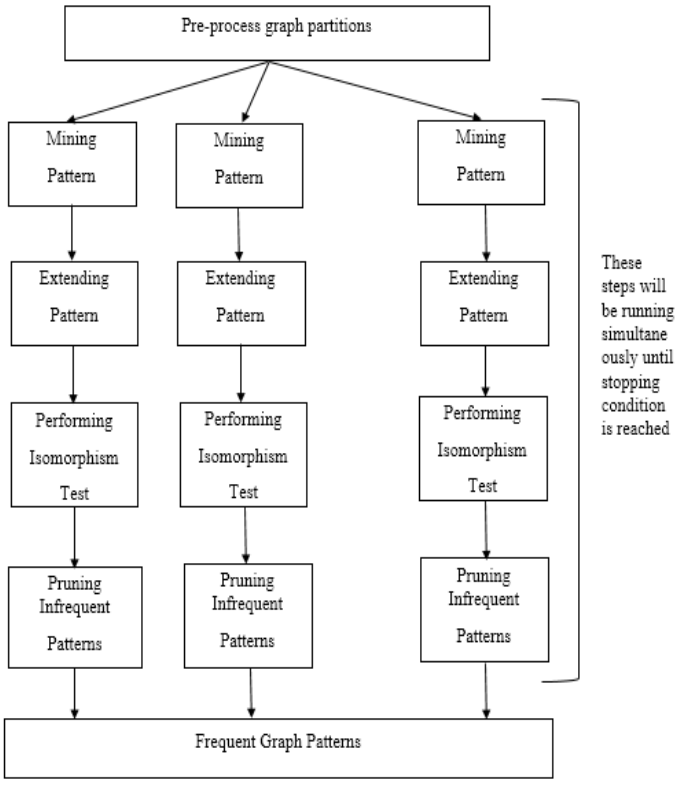

Figure 1: Framework of proposed approach

There are three phases in our proposed approach. Section IIIA addresses the first phase i.e. Mining Phase, section IIIB addresses the second phase i.e. Isomorphism Test phase, and lastly, section IIIC addresses the Pattern Pruning phase.

A. Mining Phase

In this section, we introduce our pattern mining approach which includes two parts. First, to start mining patterns in a

graph, we have to choose some initial points or nodes for mining. Here we have taken those points as initial points, which have the highest number of associativity. As we are dealing with social networks, the nodes of graph will be the people and edges will be the connection among people. Associativity of a node points to the number of connections of that node. The top-k nodes having highest associativity will be chosen as the initial points to start mining.

Algorithm 1: Initial node set

Require: k, number of initial nodes
Ensure: node set V = $\{v_i\}$

1: Sort the nodes in decreasing order of the no of time it appears
2: V = $\{v_i \mid 1 \leq i \leq k\}$
3: output V

After discovering the initial k-points to start mining, the points should be extended to discover single-edged graph patterns. For this extension purpose, we have used Breadth First Search approach in our proposed system.

Definition 1: *Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key') and explores the neighbour nodes first, before moving to the next level neighbours.*

Algorithm 2: Set Up Single-edged graphs

Require: V initial node set
Ensure: $G_{init} = \{g_1, g_2, \ldots, g_k\}$, k single edge graphs

1: for each $v_i \in V$
2: $G_{init} \square \{(v_i, v_o) \mid 1 \leq i \leq k, v_o \in G \Lambda v_o \in V\}$
3: Output G_{init}

B. *Isomorphism test phase*

In this section, Isomorphism test will be conducted among the patterns mined. After getting the initial single-edged patterns, the next step is to perform isomorphism test among the single-edged patterns. Isomorphism test ensures that two isomorphic patterns are not considered to be two different patterns. Besides, the count of that pattern should be increased.

Definition 2: *In graph theory, an isomorphism of graphs G and H is a bijection between the vertex sets of G and H , f: V(G) → V(H) such that any two vertices u and v of G are adjacent in G if and only if f(u) and f(v) are adjacent in H. This kind of bijection is generally called "edge-preserving bijection", in accordance with the general notion of isomorphism being a structure-preserving bijection.*

If an isomorphism exists between two graphs, then the graphs are called isomorphic and we write $G \simeq H$. *In the case when the bijection is a mapping of a graph onto itself, i.e., when G and H are one and the same graph, the bijection is called an automorphism of G.*

Algorithm 3: Isomorphism Test

Require: g1, g2

Ensure: true or false

1: create a pattern edge between every pair of nodes in each graph.

2: match the pattern graphs

3: **if** pattern graph G_p1 is isomorphic to pattern graph G_p2

4: output true

5: else output false

C. *Pattern Pruning Phase*

It is the last phase of our framework after mining and isomorphism test. Actually all the phases keep executing continuously step by step. After getting the initial single-edged patterns or sub-graphs, isomorphism test is conducted on the resulted patterns. The isomorphism test results in increasing the count for those patterns which are isomorphic and the count of remaining patterns remains the same. As there is a minimum support count or threshold is attached with our approach, we will choose those patterns which have support count greater than the minimum support count or threshold. In any pattern discovery mechanism this approach is taken as we all are mostly interested with frequently occurring patterns. The patterns or sub-graphs with higher count are more interesting to study. Thus in our approach in pruning phase, all the patterns or sub-graphs with support count less than the minimum support count or threshold will be pruned or discarded.

After pruning of infrequent patterns, again the frequent patterns are extended, isomorphism tests are done and the infrequent patterns are pruned. This process continues until we reach the stopping conditions. There are two stopping conditions, occurrence of either one terminates our approach.

1. All the graph patterns discovered in the previous step are infrequent.

2. All the nodes of the graph are visited. Once a node is visited by some pattern while extending, it is marked as visited and cannot be visited again.

Algorithm 4: Prune Infrequent Patterns

Require: $G_{intermediate}$ = intermediate set of graphs, t threshold

 ISSN: 2321-8169
3196 – 3200

Ensure: $G_{frequent}$ frequent set of graphs

1: **for** each $g_i \in G_{intermediate}$

2: calculate Sup_g, the support count of g_i

3: **if** $Sup_g > t$

4: $G \square g_i$

5: output $G_{frequent}$

Algorithm 5: Edge Extension

Require: $G_{i,}$ the input graph

Ensure: G_{ext} or null

1: **if** any nodes are left in the graph

2: $G_{ext =}$ add an edge to the subgraph using BFS

3: **if** all nodes are exhausted

4: prune the subgraph

5: output G_{ext}

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we present the performance evaluation of proposed approach with real data sets [14]. The setting of the environment will be introduced in section IV A. Result analysis will be discussed in section IV B.

A. Experimental environment

This paper implements our proposed approach on Hadoop 2.6.0 in a cloud computing environment consisting of 4 virtual machines. One node acts as master node and others serve as slave nodes. We are using AWS EC2 instances of type t2.macro, each one is having 1 vCPU, and 1 GiB memory.

B. Result analysis

Users have to give the input partitioned graphs, the number of initial nodes, the feature based on which the patterns will be discovered (work id, education type etc.) and the threshold value. Depending upon the feature chosen, the number of initial nodes and threshold, different patterns will be discovered.

1) Effect of number of initial nodes chosen on result:

As the initial nodes for mining can be specified by users, the number of patterns retrieved will vary if the selected feature and threshold are kept constant. In figure 2, we have recorded number of frequent patterns against number of initial nodes chosen for mining. (Feature: education;type , threshold:6).

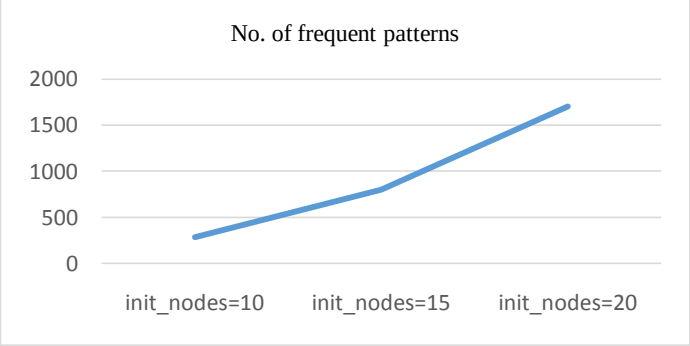

Fig 2: No. of frequent patterns against number of initial nodes.

2) Effect of threshold value chosen on result:

The threshold value also can be adjusted by user. The number of patterns will vary if the number of initial nodes for mining and feature is kept constant. In figure 3, we have recorded the number of frequent patterns against threshold value. (Feature: education;type, number of initial nodes:20).

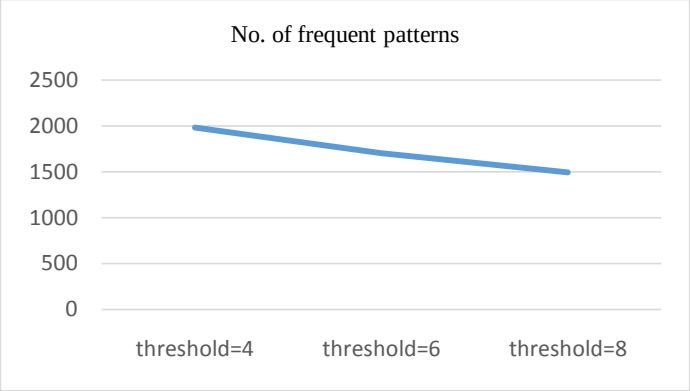

Fig 3: No. of frequent patterns against number of threshold values

V. CONCLUSION

Big data mining is now a hot research question, MapReduce brings a simple distributed programming. This paper proposes an approach based on Hadoop and cloud computing in handling big graph data. In the face of mining large data sets, the parallelization is a good solution. Our proposed approach is an approximation-based approach which will results in frequent graph patterns ranging from single-edged to n-edged graph patterns based on the initial node set, feature based on which patterns are searched for and the threshold value. The graph patterns can be studied for different purposes. Also we can get patterns from combination of feature sets from the patterns listed in the input graph. The proposed approach can be used for other social network graphs if the data format is same as the input dataset. In future work, more real big data sets can be examined for this approach. Moreover, we can improve the way of finding the initial nodes for pattern mining, pattern extension and stopping conditions.

REFERENCES

[1] P. Anchuri, M. J. Zaki, O. Barkol, R. Bergman, Y. Felder, S. Golan, and A. Sityon, "Graph mining for discovering infrastructure patterns in configuration management databases," *Knowledge and information systems*, vol. 33, no. 3, pp. 491– 522, 2012.

[2] V. Bonnici, R. Giugno, A. Pulvirenti, D. Shasha, and A. Ferro, "A subgraph isomorphism algorithm and its application to biochemical data," *BMC Bioinformatics*, vol. 14, no. Suppl7, p. S13, 2013.

[3] S. Ghazizadeh and S. S. Chawathe, "Seus: Structure extraction using summaries," in *Discovery science*. Springer, 2002, pp. 71–85.

[4] C. Borgelt, T. Meinl, and M. Berthold, "Moss: a program for molecular substructure mining," in *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*. ACM, 2005, pp. 6–15.

[5] L. B. Holder, D. J. Cook, and S. Djoko, "Substucture discovery in the subdue system." in *KDD Workshop*, 1994, pp. 169–180.

[6] Z. Khayyat, K. Awara, A. Alonazi, H. Jamjoom, D. Williams, and P. Kalnis, "Mizan: a system for dynamic load balancing in large-scale graph processing," in *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 2013, pp. 169–182.

[7] U. Kang, H. Tong, J. Sun, C.-Y. Lin, and C. Faloutsos, "Gbase: an efficient analysis platform for large graphs," *The VLDB Journal*, vol. 21, no. 5, pp. 637–650, 2012.

[8] http://www.cs.ucsb.edu/~xyan/papers/vldb11_spider.pdf

[9] Chun-Chien Chen, Kuan-Wei Lee, Chi-Chieh Chang, De-Nian Yang and Ming-Syan Chen "Efficient Large Graph Pattern Mining for Big Data in the Cloud".

[10] http://hadoop.apache.org/.

[11] http://en.wikipedia.org/wiki/MapReduce.

[12] http://en.wikipedia.org/wiki/Amazon_Web_Services.

[13] http://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud.

[14] "Standford large network dataset collection", http://snap.standford.edu/data/