ISSN No: - 2456 – 2165

The Conclusion of FS3-H A Sampling Based Method for Top-K Frequent Sub Graph Running and Comparison of Execution of Map Reduce Vs Spark on Dataset

Ganesh Vilas Muluk
M.E. Information Technology
Siddhant College of Engg.Sudumbre,
Pune, India.
mulukganesh@gmail.com

Prof. J. M. Pingalkar
M.E. Information Technology
Siddhant College of Engg.Sudumbre,
Pune, India.

Abstract—Frequent Subgraph Mining (FSM) is an important research task on graph data. Near about 90% searching in the world is frequent. It is challenging task to the search result with big data in efficient time, which unable to feet into memory and also have I/O performance limitations. The problem statement is to find the complete and feasible solution to FSM problem by building a distributed system which improves run up in memory, I/O performance, execution time and quality of result with the low-cost system using Hadoop. We are avoiding memory, I/O performance limitations of existing methods Frequent Subgraph Mining Algorithm in Hadoop (FSM-H) and A Sampling-based method for top-k Frequent Subgraph Mining (FS3) by combing these two methods and new technologies to implement FS3-H. We are doing an experiment on the graph by partitioning it into k-subgraph with multilevel graph partitioning (MGP) approach and k-means clustering algorithm on the decentralized cluster on low-cost systems connected with Java Remote Method Invocation (RMI) technology using Apache Spark framework which is built on Hadoop and alternative to Hadoop iterative Map-Reduce of FSM-H. Also, we are doing an experimental analysis on mining of graph by storing and fetching top-k frequent sub graph of FS3 parallel with user defined support in Hadoop and MySQL into decentralized distributed k-sub-queue manager. The proposed method will avoid limitations of existing methods. It has applications in various disciplines, including chemo informatics, bioinformatics, social sciences and various aspects of real life.

Keywords—Graph Storage, Graph Mining, FS3-H, Hadoop.

I. INTRODUCTION

Graph mining is the process of discovering, retrieving and analyzing nontrivial patterns in graph-shaped data. The main usage of FSM is for finding subgraph patterns that are frequent across a collection of graphs. This task is additionally useful in applications related to graph classification, and graph indexing. The existing algorithms for sub graph mining are not scalable to in social and biological large graphs. For instance, a typical protein-protein interaction (PPI) network contains a few hundred and a few thousands of known proteins interactions. The most efficient of the existing FSM algorithms with a large support value cannot mine frequent sub graphs in a reasonable amount of time from a small database of PPI networks. In this era of big data, we are collecting graphs of even larger size, so an efficient algorithm for FSM is in huge demand. Mining patterns from graph databases are challenging graph related operation like subgraph testing, have higher time complexity than the corresponding operations on item sets, sequences, and trees. The major challenging problems for Frequent Subgraph Mining are Sub graph isomorphism and an efficient scheme to enumerate all frequent sub graphs.

The all existing algorithms either incomplete or cannot produce output within the feasible amount of time. The algorithm either focuses on graph storage or graph retrieval. The traditional methods for data analysis and mining are not designed to handle the massive amount of data. In recent years many such methods implemented that is aware of the big data. The most successful framework for distributed computing is MapReduce, which adopts the data centric approach to distributed computing. The limitations of MapReduce was it computes support of candidate subgraph pattern over entire graph in a graph dataset and in distributed platform it was difficult to decide which graph is frequent. These limitations are avoided in FSM-H method [2], proposed by Mansurul A Bhuiyan, and Mohammad Al Hasan using an Iterative MapReduce framework. Y urong Zhong, Dan Liu proposed K-Means Clustering Algorithm Based on hadoop[3]. The limitation of FSM-H and proposed K-Means Clustering Algorithm is it I/O and memory overhead. Also, Iterative MapReduce framework does not support real-time data streaming. To avoid these limitations we propose to use Apache Spark Framework, which is alternative to MapReduce and it is very advance with additional functionalities. We are using MGP approach for graph partition. We are focusing on both graph storage and graph mining in proposed method. We are implementing

concept priority queue and queue manager to store top-k frequent subgraph of the FS3 method proposed by Tanay Kumar Saha and Mohammad Al Hasan [1]. In this method, the author stored top-k frequent subgraph in the priority queue with MCMH algorithm and queue evection strategy with user defined support. The limitations of this method are memory limitations of queue manager, I/O overheads, and subgraph computation overheads. We are avoiding these limitations by fetching subgraph from Hadoop cluster. Also, we are maintaining queue manager for each system in the network and also store some sort of result in serialized object. Then we are fetching results with parallel computing from sub-queue manager and combine its result as output. We will reduce redundant search using Exceptions method proposed by Shingo Okuno, Tasuku Hiraishi, Hiroshi Nakashima, Masahiro Yasugi and Jun Sese[4].

II. RELATED WORK

There are various types of sub graph mining algorithm in hadoop. But the best of them implemented as follows: a) FSM-H: Frequent Subgraph Mining Algorithm in Hadoop and b) FS^3 : A Sampling based method for top-k Frequent Subgraph Mining. FSM-H uses an iterative MapReduce based framework. FSM-H is complete as it returns all frequent subgraphs for a given user defined support. It uses distributed file system to optimize I/O performance. FSM-H generates a complete set of frequent subgraphs for a given minimum support. To ensure completeness, it constructs and retains all patterns that have a non-zero support in the map phase of mining and then in the reduce phase, it decides whether a pattern is frequent by aggregating its support from different computing nodes. FSM-H runs in an iterative fashion, where the output from the reducers of iteration i-1 is used as an input for the mappers in the iteration i. The reducers of iteration I then find the true frequent subgraph (of size i) by aggregating their local supports. Iterative Map Reduce can be defined as a multi staged execution of Map and Reduce function pair in a cyclic fashion i.e. the output of the stage i reducers is used as an input of the stage i+1 mappers. The external condition decides the job termination. FS^3 is a sampling base method for mining top-k frequent subgraphs of a given size, p. FS^3 works on Fixed Size Subgraph Sample. FS^3 does not perform costly subgraph isomorphism test. FS^3 performs a Markov Chain Monte Carlo (MCMC) sampling over the space of fixed-size subgraphs to find frequent subgraph. The FS^3 works with two stages on a graph database G, and a size value p, FS^3 samples subgraph of size-p from the database graph using a 2-stage sampling. Stage-1: FS^3 chooses one of the database graph Gi uniformly. Stage-2: FS^3 chooses a size-p subgraph of g using MCMC. The sampling distribution of the second stage is biased such that it oversamples the graphs that are likely to be frequent over the entire graph. FS^3 use innovative queue mechanism to hold the top-k frequent subgraphs as sampling process. FS^3 uses queue evection strategy to maintain subgraph in queue mechanism.

III. LITERATURE SURVEY

Tanay Kumar Saha and Mohammad Al Hasan, "FS^3: A Sampling based method for top-k Frequent Subgraph Mining", 2014 IEEE International Conference on Big Data.[1]

In this paper, the author proposed a sampling based method FS^3, which mines a small collection of most frequent. FS^3 performs a Markov Chain Monte Carlo (MCMC) sampling over the space of fixed-size subgraphs FS^3 is equipped with an innovative queue manager. It stores the sampled subgraph in a finite queue such that the top-k positions in the queue contain the most frequent subgraphs. FS^3 is efficient to obtain most frequent subgraphs for the database of large graphs.

Mansurul A Bhuiyan, and Mohammad Al Hasan, "FSM-H: Frequent Subgraph Mining Algorithm in Hadoop", 2014 IEEE International Congress on Big Data.[2]

In this paper, the author proposed a frequent subgraph mining algorithm based on iterative MapReduce-based framework called FSM-H. FSM-H is complete as it returns all the frequent subgraphs with given user-defined support, and it is efficient and optimized. It maintains higher level of abstraction and uses data centric approach of distributed system. It uses distributed file system that is particularly optimized to improve the I/O performance.

Y urong Zhong, Dan Liu, "The Application of K-Means Clustering Algorithm Based on Hadoop", 2016 IEEE International Conference on Cloud Computing and Big Data Analysis.[3]

Spatial data contains some kind of property information of space feature and also has the spatial feature of space or location. The traditional clustering methods and clustering analysis focuses feature of space object's properties, but ignores the geographical coordinate information of the space, and the spatial proximity of objects. K - Means spatial clustering algorithm combines geographic location and property feature to solve geographical position and property feature spatial data.

Shingo Okuno, Tasuku Hiraishi, Hiroshi Nakashima, Masahiro Yasugi and Jun Sese, "Reducing Redundant Search in Parallel Graph Mining using Exceptions.", 2016 IEEE International Parallel and Distributed Processing Symposium Workshops.[4]

In this paper, the author proposed an implementation of a parallel graph mining algorithm having exception handling features using a task-parallel language. The performance of a backtrack search algorithms is poor and a worker may search a useless sub tree, can result in a large amount of redundant search. This drawback can be implemented efficiently using the task-parallel language with exception handling mechanism by all running tasks parallel in a try block with an exception are automatically aborted. The author applied this abort mechanism algorithm using task-parallel language Tascell COPINE to the graph mining.

Esraa Al-sharoa and Selin Aviyente, "Evolutionary spectral graph clustering through subspace distance measure", 2016 IEEE Statistical Signal Processing Workshop (SSP).[5]
Today's world the massive amounts of high-dimensional data are increasingly gathered. Much of this is streaming big data that is either not stored or stored only for short periods of time. It is important to be able to reduce the dimensionality of this data in a streaming fashion. Clustering is a common way of reducing the dimensionality of data. In this paper, the author proposed an evolutionary spectral clustering approach for community detection in dynamic networks. In this paper, the author proposed method to obtain smooth cluster assignments by minimizing the subspace distance between consecutive times.

IV. PROPOSED SYSTEM MODEL

The problem statement is to find the complete and feasible solution to FSM problem by building a distributed system which improves run up in memory, I/O performance, execution time and quality of result with the low-cost system using Hadoop. Our objective is:

- Experimental analysis on medium size graph by partitioning it into k-subgraph with appropriate graph partition and using clustering algorithm on decentralized cluster on low cost system with Apache Spark framework which is built on Hadoop and alternative to Hadoop Map-Reduce to improve run up in memory and I/O performance.
- Experimental analysis on Mining of graph by storing and fetching top-k frequent subgraph into decentralized distributed k-sub-queue manager to improve execution time and quality of result with user defined support in Hadoop.

A. The System Will Execute Using Below Procedure

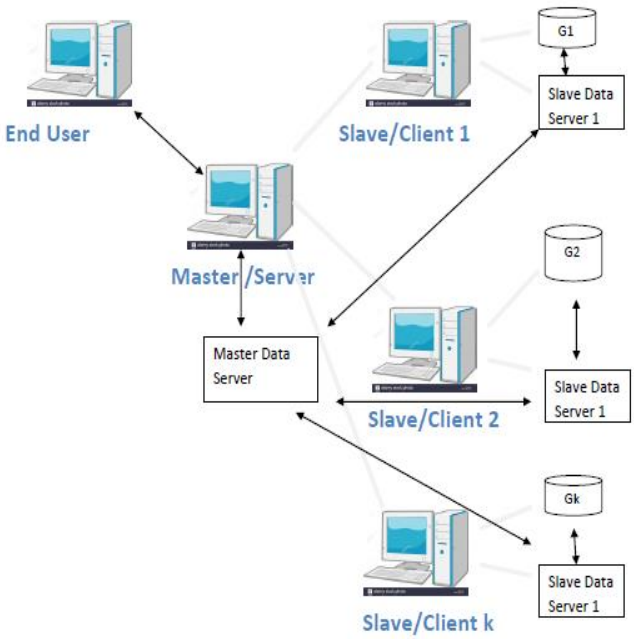

Fig. 1: Proposed System Architecture

We are storing each graph on the separate low-cost system connected in the network. Again this single graph partition is either stored in Hadoop cluster or again partitioned into subgraph partitions as shown in figure 3.4.2. This process repeats until external termination condition or unable to make graph partition. Each graph partition is stored on Hadoop system with k-means clustering algorithm. Each system works as master or slave server depending on the network. The system is connected with each other using java RMI technology. We are using MySQL to check user defined support of subgraph. The System implementation is divided in to 3 modules as follows, 1) System administration and maintenance module 2) Graph Storing module 3)Graph Mining.

a). System Administration and Maintenance Module

This module includes user registration, user authentication and system authorization process. This module also includes user reports, graph visualization and business intelligence reports.

b). Graph Storing Module

This module includes the following process:

1. Extraction of social network data of users from MySQL to Hadoop using sqoop.
2. Transformation of data to proper format
3. Load data into Hadoop cluster.

In this module we are fetching social network graph data with sqoop and then applying graph partition algorithm, Iterative Map Reduce algorithm, K-Means clustering algorithm with Apache Spark in Hadoop on graph data. We are storing subgraph on every data server. The graph partition and partition index is maintained by graph partition and index manager by coordinating with Host Communication Client and Server. The Host Communication Client and Server know master and slave server details and level of graph partition. If any changes occur in graph then data server automatically update its reflection to sub-queue manager. Implementation in this module focuses on storage of graph in well-ordered form such that it will easily available to access within minimum time.

c). Graph Mining Module

This module focuses on graph mining. This module focus on improve execution time and aware of quality of result. In this module is using queue manager mechanism. In this module we are using concept of sub-queue manager mechanism. We have implemented queue manager on every data server. We are storing subgraph results on physical disk and it's index in sub-queue manager. When data server receives request for graph data then it firstly check graph partition and index manager and get address of data server where subgraph is stored. The target data server receives request for data , then it look up data in sub-queue manager if he found it return precomputed result with index provided by queue manager from hard disk or lookup into hadoop cluster with routine graph mining phase . After this the master data server accepts results from slave data server and combines it as ouput. Then

it returns it to it's master data server or send response to user if itself .super master data server.

B. Keyword Points

a). ETL

The process of extracting data from source systems and bringing it into the data warehouse is commonly called ETL, which stands for extraction, transformation, and loading.[6] We are using MySQL as transactional database, Hadoop and Spark as data mining tool. We are importing data from MySQL to Hadoop by using sqoop, by filtering graph data result as per user requirement. After this process next process is graph partitioning and graph clustering.

b). Graph Partitioning

The graph partitioning problem formulation for an undirected graph G = (V, E) asks for a division of V into k pairwise disjoint subsets (partitions) such that all partitions are of the approximately equal size and the edge-cut, which is the NP-hard problem. Scoring rules and objective functions are important problems.

c). Graph Clustering

Clustering is an important tool for investigating the structural properties of data. The clustering refers to the grouping of similar objects in the same cluster. Clustering graphs usually refer to the identification of vertex subsets that have significantly more internal edges than external ones.

d). Hadoop

Hadoop is an open source, Java-based programming framework that supports the processing and storage of extremely large data sets in a distributed computing environment.[7] It is part of the Apache project sponsored by the Apache Software Foundation. Hadoop makes it possible to run applications on systems with thousands of commodity hardware nodes, and to handle thousands of terabytes of data. Its distributed file system facilitates rapid data transfer rates among nodes and allows the system to continue operating in case of a node failure.[8] This approach lowers the risk of catastrophic system failure and unexpected data loss, even if a significant number of nodes become inoperative. Consequently, Hadoop quickly emerged as a foundation for big data processing tasks, such as scientific analytics, business and sales planning, and processing enormous volumes of sensor data, including from internet of things sensors.[9]

C. Mathematical Module:

a). Frequent Sub-graph Mining

Let, G = {G1,G2, . . . , Gn} be a friend graph database,
where each Gi ∈ G, ∀i = {1 . . .n} represents a labeled, undirected and connected graph.

Now, t(g) = {Gi : g ⊆Gi ∈ G}, ∀i = {1 . . .n}, is the support-set of the graph g (here the subset symbol denotes a subgraph relation).
Thus, t(g) contains all the graphs in G that has a subgraph isomorphic to g.
The cardinality of the support-set is called the support of g.
g is called frequent if support $\geq \pi^{min}$, where π^{min} is predefined/user-specified minimum support (minsup) threshold.[2]

b). Connected Subgraph:

For a given graph
$G = (V,E)$, we call $G' = (V',E')$ a connected subgraph1 of G iff all of the following criteria hold.

(1) $V' \subseteq V$
(2) $E' = \{(u, v) \mid u, v \in V'\} \cap E$
(3) $\forall u, v \in V' : \exists\{(u1, v1), \cdots, (un, vn)\}$ being the path between u and v where $u1 = u$, $vn = v$, $(ui, vi) \in E'$, $ui = vi-1$ $(1 < i \leq n)$

Note that E' is uniquely defined by V', and thus we may let E' be denoted by (V').

V. EXPERIMENTAL RESULT

We are using 2 different Hadoop multimode clusters for our project. Each multimode cluster has 2 Hadoop data node and 2 Apache Spark worker nodes. Both multimode clusters are connected to each other by using data server written with java RMI technology. Total four computer systems are required with 2GB RAM, Dual Core processor and Minimum 15 GB Hard disk. Each machine we have installed MySQL, Java, Hadoop, Apache Spark, Sqoop, Eclipse and Net beans IDE, Apache Tomcat Server. We are generating our own dataset from social networking project 'My Network'. We store relational data in MySQL database and non-relational data in Hadoop. We import graph data into Hadoop from MySQL by using sqoop tool. It is part of ETL operation. Then we apply graph partition and clustering on input data. Now entire data storage process complete. Now this stored graph data is input for graph mining.

Dataset size	Support	1 master data server	2 master data server
200K	1%	2471	1332
200K	4%	1718	1002
200K	7%	1203	721
400K	1%	2604	1509
400K	4%	2134	1737
400K	7%	2134	1217
600K	1%	2805	1637
600K	4%	2341	1467
600K	7%	2305	1337
800K	1%	3173	1853
800K	4%	2773	1623
800K	7%	2573	1453
1000K	1%	3702	2021
1000K	4%	3282	1809
1000K	7%	2786	1559

Table 1: Performance of Map Reduce-FSG (in seconds)

Dataset size	Hadoop (Minutes)	Spark (Minutes)
800MB	15	3
1GB	18	4.4
2GB	32	5.5
3GB	45	7

Table 2: Performance of Hadoop and Spark

VI. CONCLUSION

We conclude that FS^3-H, a sampling-based method for storing and finding the frequently induced subgraph of a given size input graphs on distributed system help to find the complete and feasible solution to large graph mining within minimum possible time and memory using parallel computing. We can use new technologies that will help to improve run up in memory and I/O performance of system and algorithm. The frequent subgraph mining is the topic of research and we can continue study to finding the complete and feasible solution to large graph mining.

VII. FUTURE WORK

In the future work of this will to use of custom ETL tool at the data source, so that all unnecessary patterns and unstructured data will be removed from the data source. In the future work we will do experiment on this system with more number of data nodes in cluster and more number of Hadoop clusters connected to each other. Also we will try to improve execution time of algorithm by using a set of GPU (Graphics processing unit) to accelerate data mining. Another important work is to implement this system to the relational database, such that system will work with real-time data.

VIII. REFERENCES

[1]. Tanay Kumar Saha and Mohammad Al Hasan, ―FS^3: A Sampling based method for top-k Frequent Subgraph Mining, 2014 IEEE International Conference on Big Data.

[2]. Mansurul A Bhuiyan, and Mohammad Al Hasan, ―FSM-H: Frequent Subgraph Mining Algorithm in Hadoop, 2014 IEEE International Congress on Big Data.

[3]. Y urong Zhong, Dan Liu. ―The Application of K-Means Clustering Algorithm Based on Hadoop‖, 2016 IEEE International Conference on Cloud Computing and Big Data Analysis.

[4]. Tasuku Hiraishi, Hiroshi Nakashima, Shingo Okuno, Masahiro Yasugi and Jun Sese, ―Reducing Redundant Search in Parallel Graph Mining Using Exception‖, 2016 IEEE International Parallel and Distributed Processing Symposium Workshops.

[5]. Esraa Al-sharoa and Selin Aviyente, ―Evolutionary Spectral Graph Clustering Through Subspace Distance Measure‖ 2016 IEEE Statistical Signal Processing Workshop (SSP).

[6]. Oracle® Database Data Warehousing Guide 10g Release 2 (10.2) B14223-02

[1] https://www.pinterest.com/pin/728738783425575367

[2] https://eworks.in/bigdata-consulting/

[3] D.Vasudha, K.Ishthaq Ahamed , " AIMS: Adaptive Improved MapReduce Scheduling in Hadoop" , International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) , Volume 6, Issue 4, July - August 2017 , pp. 249-256 , ISSN 2278-6856.