ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

Parallel Mining of Frequent Itemsets Using FIMN on Neo4j

M.Jayashree[1], M.Barathi[2]

PG Scholar, Dept. of CSE, Ganadipathy Tulsi's Jain Engineering College, Vellore, Tamil Nadu, India

Professor, Dept. of CSE, Ganadipathy Tulsi's Jain Engineering College, Vellore, Tamil Nadu, India

ABSTRACT: As the data size increases globally day to day on web world, the utilization and enhancement of data retrieval and persistence is required. The current parallel mining techniques involves the clustering and frequent itemsets mining which critically lacks in automatic parallelization, load balancing, data distribution, and fault tolerance on large clusters. The existing system works on top of FiDoop framework, which possesses the latency and low threshold on handling large clusters. In order to ensure the automatic parallelization and dynamic shrading and automatic instance, the new framework for frequent itemset mining FIMN is built on top of Neo4j graph database applying the Random forest algorithm which undergoes the several classifications of data uploaded. In Neo4j, MapReduce jobs are implemented to complete the mining task. In the MapReduce work, the mappers freely decay itemsets; the reducers perform mix operations by developing little ultra metric trees, and the real mining of these trees independently. A survey on this FIMN framework on Neo4j includes it will has high scalability and efficiency on application servers and there is no latency issues.

KEYWORDS: FIMN, frequent itemsets, MapReduce, Random Forest, Neo4j.

I. INTRODUCTION

Frequent itemset mining (FIM) is one of the best known and most prominent data mining techniques. Frequent itemsets mining is a center issue in association rule mining (ARM), sequence mining, and the like. Accelerating the process of FIM is basic and key, since FIM utilization represents a critical bit of mining time because of its high calculation and input/output (I/O) power.

Frequent itemset mining is a data analytics strategy that was initially produced for market basket analytics. It goes for discovering regularities in the shopping behavior of the clients of supermarkets, mail order organizations, and online shops. Specifically, it tries to distinguish sets of items that are frequently purchased together. When recognized, such arrangements of related items might be utilized to streamline the association of the offered items on the racks of a market or the pages of a mail-order inventory or Web shop, or may give insights which items may advantageously be packaged.

Frequent pattern mining (FPM) assumes a fundamental part in mining association. Parallel frequent pattern detection algorithms exploit parallel and distributed computing resources to mitigate the sequential bottlenecks of current frequent pattern mining algorithms. In this manner, parallel FPM algorithm accomplish better versatility and execution, so they are drawing in much consideration in the data mining research group.

Frequent patterns are prevalent in real-life data, such as sets of items bought together in a superstore and some molecular fragments frequently appearing in a certain class of molecules with similar functions. Frequent pattern mining has been successfully applied to association rule mining, pattern-based classifications and clustering, and finding correlated items, and has become an essential data mining task. Although many efficient serial FPM algorithms have been proposed in recent years with a wide variety of implementation details, they can be differentiated by their pattern lattice enumeration strategies, support counting methods, and search space pruning techniques.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

II. RELATED WORK

Frequent itemsets mining algorithm can be isolated into two classifications [2], [7], in particular, Apriori and FP-growth schemes. Apriori is a classic algorithm using the generate-and-test step that produces an extensive number of candidate itemsets; Apriori needs repeated scanning of entire database. To reduce the time required for examining databases, proposed a novel approach called FP-growth, which abstains from producing candidate itemsets. Most created parallel FIM algorithms were based upon the Apriori algorithm. Unfortunately, in Apriori-like parallel FIM algorithms, every processor needs to filter a database numerous times and to trade an excessive number of candidate itemsets with different processors. In this manner, Apriori-like parallel FIM solutions endure potential issues of high I/O and synchronization overhead, which make it aggressive to scale up these parallel algorithms. This problem has been tended to by the execution of FP-growth like parallel algorithms. A drawback of FP-growth like parallel calculations, in any case, lies in the infeasibility to develop in-memory FP trees to suit extensive scale databases. This issue turns out to be more articulated with regards to massive and multidimensional databases.

III. PRELIMINARY

In this section, we first briefly review the FiDoop. Then, we summarize the basic idea of the Random Forest algorithm. To facilitate the presentation of FIMN, we introduce the MapReduce programming framework.

A. *Frequent Itemset Mining:*

Frequent Itemset Mining is one of the most critical and time-consuming tasks in association rule mining (ARM), an often-used data mining task, provides a strategic resource for decision support by extracting the most important frequent patterns that simultaneously occur in a large transaction database [3]. A typical application of ARM is the famous market basket analysis.

Let I=I1, I2, … , Im be a set of m distinct attributes, T be transaction that contains a set of items such that T ⊆I, D be a database with different transaction records Ts. An association rule is an implication in the form of X⇒Y, where X, Y ⊂I are sets of items called itemsets, and X∩Y = ∅. X is called antecedent while Y is called consequent, the rule means X implies Y. There are two important basic measures for association rules, support(s) and confidence(c). Since the database is large and users concern about only those frequently purchased items, usually thresholds of support and confidence are predefined by users to drop those rules that are not so interesting or useful. The two thresholds are called minimal support and minimal confidence respectively. Support(s) of an association rule is defined as the percentage/fraction of records that contain X ∪Y to the total number of records in the database. Suppose the support of an item is 0.1%, it means only 0.1 percent of the transaction contain purchasing of this item. Confidence of an association rule is defined as the percentage/fraction of the number of transactions that contain X ∪Y to the total number of records that contain X. Confidence is a measure of strength of the association rules, suppose the confidence of the association rule X⇒Y is 80%, it means that 80% of the transactions that contain X also contain Y together.

In general, a set of items (such as the antecedent or the consequent of a rule) is called an itemset. The number of items in an itemset is called the length of an itemset. Itemsets of some length k are referred to as k-itemsets.

Generally, an association rules mining algorithm contains the following steps:

- The set of candidate k-itemsets is generated by 1-extensions of the large (k -1) -itemsets generated in the previous iteration.
- Supports for the candidate k-itemsets are generated by a pass over the database.
- Itemsets that do not have the minimum support are discarded and the remaining itemsets are called large k itemsets.

This process is repeated until no more large itemsets are found.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

B. *Fidoop:*

Aiming to improve data storage efficiency and to avert building conditional pattern bases, FiDoop incorporates the concept of FIU-tree rather than traditional FP trees [1]. After *h*-itemsets are generated in phase 1, an iterative process is repeatedly running to construct *k*-FIU trees and to discover frequent *k*-itemsets until the *k* value is from *M* to 2. In other words, the construction of *k*-FIU trees and the discovery of frequent *k*-itemsets are executed in a sequential way. Even worse, it is nontrivial to construct *k*-FIU-tree, which is the most important and time consuming phase. Then constructs a *k*-FIU tree by decomposing each *h*-itemset into *k*-itemsets, where $k + 1 \leq h \leq M$. Then, the union of original *k*-itemsets is calculated to construct the *k*-FIU tree. The generation of the *k*-itemsets requires decomposing all possible *h*-itemsets ($h > k$); thus, the decomposition process is sequentially performed starting from long to short itemsets. As such, we improve the serial FIUT algorithm as follows.

1) The first phase of FIUT involving two rounds of scanning a database is implemented in the form of two MapReduce jobs. The first MapReduce job is responsible for the first round of scanning to create frequent one-itemsets. The second MapReduce job scans the database again to generate *k*-itemsets by removing infrequent items in each transaction.

2) The second phase of FIUT involving the construction of a *k*-FIU tree and the discovery of frequent *k*-itemsets is handled by a third MapReduce job, in which *h*-itemsets ($2 \leq h \leq M$) are directly decomposed into a list of $(h - 1)$ itemsets, $(h - 2)$ itemsets,…., and two-itemsets. In the third MapReduce job, the generation of short itemsets is independent to that of long itemsets. In other words, long and short itemsets are created in parallel by our parallel algorithm.

C. *Random Forest:*

Aim Random forests are an aggregate learning method for classification, regression and other tasks, which operate by constructing a multiple decision trees at processing time and outputting the mode of the classes or mean prediction of the separate trees [11]. It is a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1 \ldots x_n$ with responses $Y = y_1 \ldots y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to the samples.

The common element in all of these procedures is that for the kth tree, a random vector Θ_k is generated, independent of the past random vectors $\Theta_1, \ldots, \Theta_{k-1}$ but with the same distribution; and a tree is grown using the training set and Θ_k, resulting in a classifier $h(\mathbf{x},\Theta_k)$ where $\mathbf{x}$ is an input vector. For instance, in bagging the random vector Θ is generated as the counts in N boxes resulting from N darts thrown at random at the boxes, where N is number of examples in the training set. In random split selection Θ consists of a number of independent random integers between 1 and K. The nature and dimensionality of Θ depends on its use in tree construction.

D. *Mapreduce Framework:*

MapReduce is a popular data processing paradigm for efficient and fault tolerant workload distribution in large clusters. A MapReduce computation has two phases, namely, the Map phase and the Reduce phase. The Map phase splits an input data into a large number of fragments, which are evenly distributed to Map tasks across a cluster of nodes to process. Each Map task takes in a key-value pair and then generates a set of intermediate key-value pairs. After the MapReduce runtime system groups and sorts all the intermediate values associated with the same intermediate key, the runtime system delivers the intermediate values to Reduce tasks. Each Reduce task takes in all intermediate pairs associated with a particular key and emits a final set of key-value pairs. MapReduce applies the main idea of moving computation towards data, scheduling map tasks to the closest nodes where the input data is stored in order to maximize data locality.

Hadoop is one of the most popular MapReduce implementations. Both input and output pairs of a MapReduce application are managed by an underlying Hadoop distributed file system [13]. At the heart of HDFS is a single NameNode a master server managing the file system namespace and regulates file accesses. The Hadoop runtime

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

system establishes two processes called JobTracker and TaskTracker. Job-Tracker is responsible for assigning and scheduling tasks; each TaskTracker handles mappers or reducers assigned by JobTracker.

When Hadoop exhibits an overwhelming development momentum, a new MapReduce programming model Spark attracts researchers' attention. The main abstraction in Spark is a resilient distributed dataset (RDD), which offers good fault tolerance and allows jobs to perform computations in memory on large clusters. Thus, Spark becomes an attractive programming model to iterative MapReduce algorithms.

IV. OVERVIEW OF FIMN

The Frequent itemset mining on Neo4j (FIMN) is a framework mainly developed for the purpose of finding frequent itemsets in a database. In which random forest algorithm is applied for classification of data provided to the mining engine. The classification is done on three factors HLM, (High, low and medium). The classification process is handled by MapReduce job which performs the dynamic shrading. The shrading involves running the separate instance for each individual job, which ensures the parallelization. It is new data partitioning method to well balance computing load among the cluster nodes.

Random forest is a trademark term for an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the classes output by individual trees. Random forests are collections of trees, all slightly different. It randomizes the algorithm, not the training data. How you randomize depends on the algorithm, for pick randomly from the k best options. It generally improves decision trees decisions.

Unlike single decision trees which are likely to suffer from high variance or high Bias Random Forests use averaging to find a natural balance between the two extremes. A random forest is a Meta estimator that fits a number of classifiable decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.

ARCHITECTURE DIAGRAM

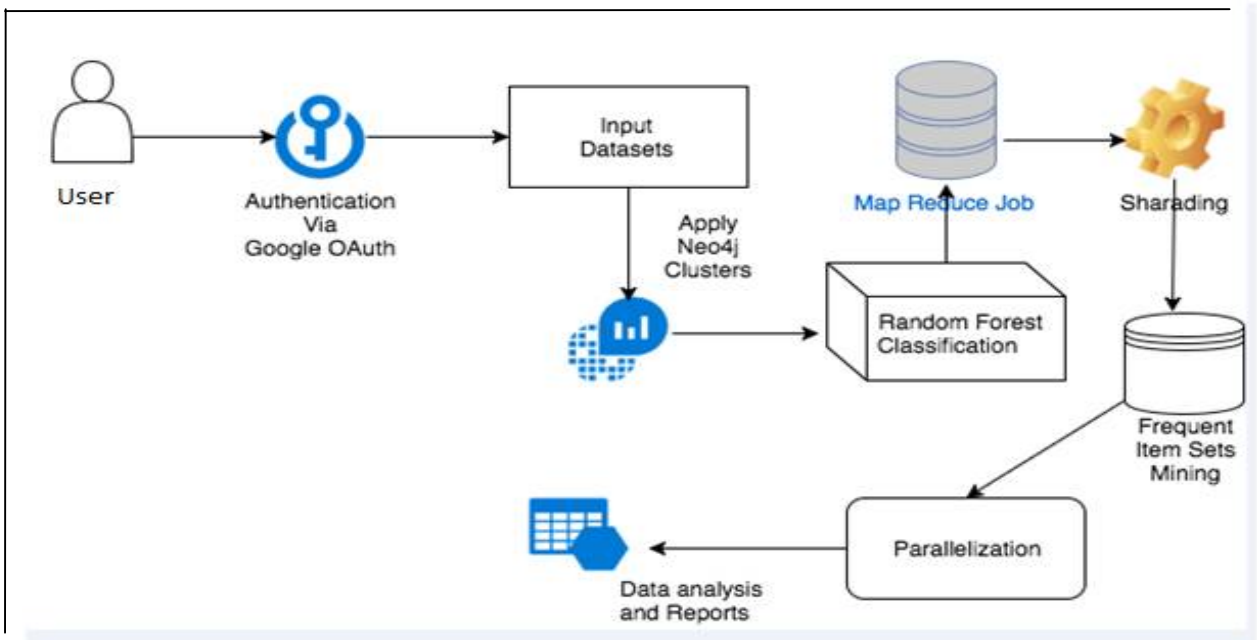

Figure 1. System Architecture

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

This Figure 1 shows the overall architecture of the system. The user first login through Google OAuth and inputs the datasets collected. Apply this to the Neo4j clusters then it will perform the random forest classification and MapReduce job will initialize the parallelization through shrading. The frequent itemsets will be reported and analysis, the output will be displayed on Neo4j.

Dataset Collection

OAuth is a service that is complementary to and distinct from OpenID. OAuth is also distinct from OATH, which is reference architecture for authentication, not a standard for authorization. However, OAuth is directly related to OpenID Connect (OIDC) since OIDC is an authentication layer built on top of OAuth 2.0.

As the application deals with patients health records, the dataset collection is done based on the health history of several patients for a specific period of last 10 years. This information is collected on gathering information sources from internet and other hospital databases. The patient emancipation is generally deals with the health care based on the perspective that better outcomes are achieved when patients become active participants in their own health management. All the information collected is stored in CSV file as a comma separated values.

Apply to Neo4j

Neo4j implements the property graph model efficiently down to the storage level. As opposed to graph processing or in-memory libraries, neo4j provides full database characteristics including acid transaction compliance, cluster support, and runtime failover, making it suitable to use graph data in production scenarios.

Some particular features make neo4j very popular among users, developers:

- Materializing of relationships at creation time, resulting in no penalties for complex runtime queries.
- Constant time traversals for relationships in the graph both in depth and in breadth due to efficient representation of nodes and relationships.
- All relationships in neo4j are equally important and fast, making it possible to materialize and use new relationships later on to “shortcut” and speed up the domain data when new needs arise.

Compact storage and memory caching for graphs, resulting in efficient scale-up and billions of nodes in one database on moderate hardware.

Random Forest Classification

With development of the information technology, the scale of data is increasing quickly. The massive data poses a great challenge for data processing and classification. Random Forest algorithm is a commonly used algorithm applied to data classification. But traditional Random Forest algorithm is not fit for the massive data. Google App Engine’s hadoop platform is a widely used open-source implementation of Google's distributed file system and the MapReduce framework, for scalable distributed computing or cloud computing. MapReduce programming model provides an efficient framework for processing large datasets in an extremely parallel mining. And it comes to being the most popular parallel model for data processing in cloud computing platform. However, designing the traditional machine learning algorithms with MapReduce programming framework is very necessary in dealing with massive datasets. In this project, we use scalable random forest approach to process the advertisement dataset.

Run Mapreduce for Clusters

The three MapReduce jobs of our proposed FIMN framework are described in detail.

- The first MapReduce job discovers all frequent items or frequent one-itemsets. In this phase, the input of Map tasks is a database, and the output of Reduce tasks is all frequent one-itemsets.
- The second MapReduce job scans the database to generate *k*-itemsets by removing infrequent items in each transaction.
- The last MapReduce job—the most complicated one of the three—constructs *k*-FIU-tree and mines all frequent *k*-itemsets.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

Frequent Itemsets Mining

Datasets in modern data mining applications become excessively large; therefore, improving performance of FIM is a practical way of significantly shortening data mining time of the applications. The dataset given as input to the Neo4j graph database, FIMN framework will mine the frequent itemsets by classifying the data using random forest algorithm then it will be processed by Mapreduce technique.

Report Generation

Thus the Frequent Itemsets mined will be displayed using the Neo4j graph database which was build to efficiently store, handle and query highly connected elements in your data model. With a powerful and flexible data model you can represent your real-world, variably structured information without a loss of richness.

The various terminologies related to this project have been explained in this section

Big data

Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications.

Data classification

In the field of data management, data classification as a part of Information Lifecycle Management (ILM) process can be defined as a tool for categorization of data to enable/help organization of data.

Distributed Computing

Distributed computing is a field of computer science that studies distributed systems. A distributed system is a software system in which components located on networked computers communicate and coordinate their actions by passing messages.

Clustering

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

V. CONCLUSION

In this paper, we propose a new framework for frequent that works based on a novel structure called Nodeset to facilitate the process of mining frequent itemsets. Hence, depending on the nodesets, an efficient framework called FIMN (Frequent Itemsets Mining using Neo4j) is proposed to mine frequent itemsets in databases. The main advantage of the proposed algorithm would be the speed and on the other side, it encodes each and every node present on the tree (Pre-Order and Post Order). It consumes very less memory to run the task on dense datasets for parallelization.

REFERENCES

1. D Yaling Xun, Jifu Zhang, and Xiao Qin, "FiDoop: Parallel Mining of Frequent Itemsets Using MapReduce" IEEE, 2015.
2. Ming-Yen Lin, Pei-Yu Lee and Sue-Chen Hsueh, "Apriori-based Frequent Itemset Mining Algorithms on MapReduce", 2012.
3. Sotiris Kotsiantis and Dimitris Kanellopoulos, "Association Rules Mining: A Recent Overview", 2006.
4. ZHANG Yuzhou, WANG Jianyong and ZHOU Lizhu, "Parallel Frequent Pattern Discovery: Challenges and Methodology", 2007.
5. M. J. Zaki, "Parallel and distributed association mining: A survey," IEEE Concurrency, vol. 7, no. 4, pp. 14–25, Oct./Dec. 1999.
6. I. Pramudiono and M. Kitsuregawa, "FP-tax: Tree structure based generalized association rule mining," in Proc. 9th ACM SIGMOD Workshop Res. Issues Data Min. Knowl. Disc., Paris, France, 2004, pp. 60–63.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

7. R. Agrawal, T. Imieli´nski, and A. Swami, "Mining association rules between sets of items in large databases," ACM SIGMOD Rec., vol. 22, no. 2, pp. 207–216, 1993.
8. J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," Data Min. Knowl. Disc., vol. 8, no. 1, pp. 53–87, 2004.
9. D. Chen et al., "Tree partition based parallel frequent pattern mining on shared memory systems," in Proc. 20th IEEE Int. Parallel Distrib. Process. Symp. (IPDPS), Rhodes Island, Greece, 2006, pp. 1–8.
10. L. Liu, E. Li, Y. Zhang, and Z. Tang, "Optimization of frequent itemset mining on multiple-core processor," in Proc. 33rd Int. Conf. Very Large Data Bases, Vienna, Austria, 2007, pp. 1275 1285.
11. Leo Breiman, "RANDOM FORESTS", University of California, January 2001.
12. J. Neerbek, "Message-driven FP-growth," in Proc. WICSA/ECSA Compan. Vol., Helsinki, Finland, 2012, pp. 29–36.
13. D. Borthakur, "Hdfs architecture guide," HADOOP APACHE PROJECT http://hadoop. apache. org/common/docs/current/hdfs design. pdf, 2008.
14. M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, vol. 10, 2010, p. 10.

BIOGRAPHY

JAYASHREE.M is a PG Scholar in the Computer Science Department, Ganadipathy Tulsi's Jain Engineering College, Vellore. She received a B.E(CSE) degree in 2015 from under Anna University, Ganadipathy Tulsi's Jain Engineering College, Vellore. Her research interests are Data Mining, Big Data, Cloud Computing, etc.

Dr.M.BARATHI is Professor in Computer Science and Engineering Department in Ganadipathy Tulsi's Jain Engineering College, Vellore, Tamil Nadu, India.