ISSN:0975-9646
Pranavkumar Bhadane et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (4) , 2015, 4058-4061

Analysis of Frequent Item set Mining of Electronic Evidence using FP-Growth based on Map/Reduce

Pranavkumar Bhadane
P.G. Student,
SSVPS's BS Deore College of Engineering,
Dhule, 424005,India

R.V.Patil
Associate Professor,
SSVPS's BS Deore College of Engineering,
Dhule, 424005,India

***Abstract*— Association rules can mine the relevant evidence of computer crime from the massive data and association rules among data itemset, and further mine crime trends and connections among different crimes. They can help detect and leads case policies and prevent crime with given criterions. Frequent item set mining (FIM) plays a fundamental Associations, correlations and electronic evidence analysis area like many real-world data mining areas. FP-growth pattern of constant search for the most famous FIM algorithm. Incrementing data, time and space costs FP-growth will be mining algorithms bottleneck. Information and communication technologies in the world, with rapid advancements in the crimes committed are becoming technologically intensive. Use digital devices when crime, forensic examiners and practical frameworks which can pose as evidence to recover the data for analyzing the methods to adopt. Data Generation, Data Warehousing and Data Mining, are the three essential features involved in the investigation process. So that we proposed a a novel parallelized algorithm called PISPO based on the cloud-computing framework MapReduce, which is widely used to cope with largescale data and captures both the content and state to be distributed to the changed and original of the transactions dataset to SPO-tree.**

Keywords*— computer crime; PISPO; ISPO-tree; MapReduce; frequent itemset; data mining; association rules.

I. INTRODUCTION

The digital world has penetrated every aspect of today's generation, both in the space of human life and mind, not even sparing the criminal sphere of the world. According to Jim Christy, Director of Cyber Crime Institute, forensic science is the application of science to legal process and therefore against crime. Science and technology, and in fact or in a Court of law of evidence relating to the use of in the process. When crime is aided by or digital device (s), including forensic investigations using digital or cyber forensic categorized under. If only one computer or digital storage medium is digital tools, as we check in computer forensics. Computer forensic (a.k.a. digital forensic) forensic science, whose goal is to explain the current state of digital artifact is a branch.

Digital Forensic Research Workshop (DFRWS) has defined Digital Forensic Science as "the use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations". Digital forensic science covers computer forensics, network forensics, disk, firewall forensics, forensics, database device, mobile device forensics, software forensics, live system forensics, etc. Digital forensic incident (s) and professionals who have developed and applied advances has been described as driven. DFRW media analyze a digital forensic analysis, the other two code analysis and network analysis being identified as three main specific type. This paper digital forensic investigation process introduces a framework for the physical storage device. It is also a tool to access and analyze its contents flash drive is a specific case. Paper details stored on the Flash drive data preprocessing steps to bring out information.

The key contribution of this research is proposing and developing a novel tree structure for maintaining frequent patterns about electronic evidence in an incremental dataset.

We offer algorithm ISPO-tree (single pass ordered tree) based on both content and an innovative approach for parallelizing Map Reduce FP-FP-growth algorithm for a tree change also have captured the State of transactions in the dataset that intelligently on a large scale mining operations and functions in computational free shards Map Reduce jobs map. This computer failures with the ability to start from the tree can achieve near linear speedup.

II. LITERATURE SURVEY

[1] J. W. Han, J. Pei, and Y. W. Yin, Mining frequent patterns in transaction databases, time series databases, and many other kinds of databases has been studied popularly in data mining research. Most of the previous studies adopt an Apriori -like candidate set generation-and-test approach. Ho w ever, candidate set generation is still costly, especially when there exist prolic patterns and/or long patterns.

In this study, we propose a novel frequent pattern tree (FP-tree) structure, which is often about the pattern is compressed, important information for an extended prefix tree structure, the complete set of patterns and pattern piece by mining an efficient FP-tree based mining mode, FP-growth, develop. With three potential mining techniques AC hived: (1) a large database is a highly compact, much smaller data structure which avoids expensive, repeated database scans, (2) our FP-tree is narrow-based mining in

large numbers to avoid costly generation candidate set a pattern fragment growth method adopts, and (3) a split-divide and conquer method, small mined databases for a set of conditional conned patterns to decompose in mining operations, which dramatically reduces the search space. FP-growth method is efficient and scalable for both long and short mining our performance study shows patterns often, and an order of magnitude is ab out faster Apriori algorithm and even faster recently reported new frequent pattern mining methods.

[2]Le Wang∗ , Lin Feng∗ , Jing Zhang, Pengyu Liao, Mainstream parallel algorithms for mining frequent itemsets (patterns) were designed by implementing FP-Growth or Apriori algorithms on MapReduce (MR) framework. Current Mr FP-growth algorithm data between nodes cannot be evenly distributed, and Mr. Apriori algorithms use multiple processes map/reduce and even with the value of the generated several key value pairs; Tthesedisadvantages hinder their Performance. This paper proposes an algorithm FIMMR: it first of all local candidates as each data segment often item sets Khan, sorting applies strategies candidates, and then identifies global frequent item sets candidates. Experimental results show that FIMMR outperforms efficiency of PFP and SPC for quite some time; and minimum support threshold FIMMR small under one of the other two algorithms, order of magnitude improvements can achieve; Meanwhile, FIMMR of speedup is satisfactory.

[3] S. K. Tanbeer, C. F. Ahmed, and B. S. Jeong, et al, FP-growth algorithm using FP-tree has been widely studied for frequent pattern mining because it can give a great performance improvement compared to the candidate generation-and-test paradigm of *Apriori*. However, it still requires two database scans which are not applicable to processing data streams. In this paper, we present a novel tree structure, called CP-tree (Compact Pattern tree), that captures database information with one scan (*Insertion phase*) and provides the same mining performance as the FP-growth method (*Restructuring phase*) by dynamic tree restructuring process. Moreover, CP-tree can give full functionalities for interactive and incremental mining. Extensive experimental results show that the CP-tree is efficient for frequent pattern mining, interactive, and incremental mining with single database scan.

[4] Sankalp Mitra1, Suchit Bande2, Shreyas Kudale3, Advait Kulkarni4, Asst. Prof. Leena A. Deshpande, et al, As an important part of discovering association rules, frequent itemsets mining plays a key role in mining associations, correlations, causality and other important data mining tasks. Since some traditional frequent itemsets mining algorithms are unable to handle massive small files datasets effectively, such as high memory cost, high I/O overhead, and low computing performance, an improved Parallel FP-Growth (IPFP) algorithm and discuss its applications in this paper. In particular, a small files processing strategy for massive small files datasets to compensate defects of low read/write speed and low processing efficiency in Hadoop. Moreover, use of MapReduce to implement the parallelization of FP-Growth algorithm, thereby improving the overall performance of frequent itemsets mining. The experimental results show that the IPFP algorithm is feasible and valid with a good speedup and a higher mining efficiency, and can meet the rapidly growing needs of frequent itemsets mining for massive small files datasets.

[5] Jeffrey Dean and Sanjay Ghemawat, et al., MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. In many real world tasks such as paper model, expressible. Written in the functional style, the program automatically parallelized and executed on a large cluster of commodity machines. Description of the run-time system input data, a set of machines to determine the execution time of the program across, dealing with machine failures, and essential difference machine takes care of communication management. It's easy to use the resources of a large distributed system programmer for parallel and distributed systems without any experience with. Our Map Reduce implementation runs on a large cluster of commodity machines and highly scalable: a typical Map Reduce computation processes many terabytes of data on thousands of machines. Find the system easy to use programmer: Map Reduce programs have been implemented and hundreds of upwards of one thousand Map Reduce jobs are executed on Google groups every day.

III. FREQUENT ITEMSET MINING

Frequent itemset mining palys a key role in data mining that focuses on looking at sequences of actions or events, for example the order in which we get dressed. Shirt first? Pants first? Socks second item or second shirt if wintertime? Sequence analysis is used in a lot of different areas, and is also highly useful in games for finding behavioral patterns that lead to particular behaviors, for example a player quitting a game. Here is how it works.

Frequent item set mining, data base instances (also called transactions) that each features (also called items) takes the form of a set of numbers. For example, items purchased from a dataset in a social online games 4 transactions can join as:

Gurning, Awesomeness, beautiful pet shirt sword {}
Awesomeness, cute pet, healing potion shirt {}
Gurning, sword of healing potion {}
{Shirt Awesomeness, fancy hats, cute pet}

The frequent item set mining algorithm for work items (times at least a minimum quantity is present) support at least a minimum defined as those item sets, all is set. If support is set to 3, the following 1-itemsets (only one item) dataset described above can be found at: {sword of Grungni} and {beautiful pet}.

It is also possible to find a 2-itemset: {shirt Awesomeness, beautiful pet}, as three transactions are both beautiful pet shirt and Awesomeness.

IV. FREQUENT ITEMSET MIMING USING MAP/REDUCE

FIM using Map/Reduce comes with a large communication cost, i.e., the number of sets to be mined can be very large, moreover, the number of sets that have to be recounted can be very large as well. Implementing such partitioning technique in Hadoop is therefore prohibitive. A possible solution for the recounting part, is to mine the sub databases with a lower threshold, hence, decreasing the number of itemsets that might have been missed. However, another problem occurs: in fact, each shard may be local for local sub structure is very different from the rest of the data defines the database. As a result, the frequent item sets quite a few shards, though many of the sets are actually local structures and interesting so far around the world blow. This method since we divided the data instead of the search space are the space to deal with such problems to have. Therefore, No additional communication is required between mappers and no overlapping mining to check results. More specifically diffuses by using this technique, memory-wise for mining large datasets fit best.

V. FP-GROWTH ALGOTITHM

The FP-Growth Algorithm is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree). In his study, Han proved that his method outperforms other popular methods for mining frequent patterns, e.g. the Apriori Algorithm and the TreeProjection. In some later works it was proved that FP-Growth has better performance than other methods, including Eclat and Relim˙The popularity and efficiency of FP-Growth Algorithm contributes with many studies that propose variations to improve his performance.

After constructing the FP-Tree it's possible to mine it to find the complete set of frequent patterns. To accomplish this job, Han in presents a group of lemmas and properties, and thereafter describes the FP-Growth Algorithm as presented below.

Algorithm: FP-Growth

***Input*:** A database DB, represented by FP-tree constructed, and a minimum support threshold?

***Output*:** The complete set of frequent patterns.

***Method*:** call FP-growth (FP-tree, null).

Procedure FP-growth (Tree, a) {

(01) If Tree contains a single prefix path then // Mining single prefix-path FP-tree {

(02) Let P be the single prefix-path part of Tree;

(03) Let Q be the multipath part with the top branching node replaced by a null root;

(04) For each combination (denoted as ß) of the nodes in the path P do

(05) Generate pattern ß ∪ a with support = minimum support of nodes in ß;

(06) Let freq pattern set (P) be the set of patterns so generated;

}

(07) Else let Q be Tree;

(08) For each item ai in Q do {// Mining multipath FP-tree

(09) Generate pattern ß = ai ∪ a with support = ai .support;

(10) construct ß's conditional pattern-base and then ß's conditional FP-tree Tree ß;

(11) If Tree ß ≠ Ø then

(12) Call FP-growth (Tree ß, ß);

(13) Let freq pattern set (Q) be the set of patterns so generated;

}

(14) Return (freq pattern set (P) ∪ freq pattern set (Q) ∪ (freq pattern set (P) × freq pattern set (Q)))

}

When a single prefix path FP-tree, full set often patterns can be generated in three parts: single prefix path P, Q and their combinations multipath (01-03 and 14 lines). A single prefix path for the resulting pattern is that enumerations support its sub paths (04-06 lines). Thereafter, (line 03 or 07) multipath Q is defined and the resulting patterns are processed from it (lines 8 to13). Finally, in line 14 results found are returned as consistent patterns.

SPO-Tree:

Single Pass Ordered Tree SPO-Tree captures information with a single scan for incremental mining. All items in a transaction are inserted/sorted based on their frequency. The tree is reorganized dynamically when necessary. SPO-Tree allows for easy maintenance in an incremental or data stream environment.

CP-Tree

Although CAN tree offer simple single pass construction process, it usually lead poor compaction in tree size compared FP tree. Therefore, it is storage and runtime inefficient causing higher mining time since the item in the tree are not stored in frequency descending order.[4] CP-tree (Compact Pattern tree)[1], is a compact prefix-tree structure which is constructed with one database scan and provides the same mining performance as the FP-growth technique by efficient tree restructuring process. Build action mainly consists of two steps: Insert step inserts the transaction (s) CP-tree item appearance and update frequency counting I-list; According to related items and restructuring, according to the frequency of items that I list rearranges and tree nodes descending new I-list according to restructures. This step in restructuring the branch restructuring [2] (BRM). In restructuring the branch method this restructuring it unclassified path one after another sorting and I-list in descending order by frequency. CP-can tree from the tree and more negligible cost restructuring despite the tree data structure extremely compact; CP-a significant performance gain on overall runtime tree.

CAN-Tree

CAN Tree require only one database scan, this is differ from the FP tree that two database scan require. In CAN tree item arranged in some canonical order, which can be determined by user prior to the mining process or at a run time during mining process so it is unaffected by the item frequency unlike FPtree. Lexicographic order or item that

can be arranged in alphabetic order. Trees are good (i) item can totally order incremental updates is unaffected by changes in frequency caused by. (ii) At least her children as high as the sum of the frequencies of the frequency of node in the tree. Build tree can we similarly consistent pattern such as FP-tree approach and use the divide-and-conquer approach. That this traverse upwards in the database. Since items constantly in our Cantered Ordered, any insertions, deletions, and modifications of the transaction has no effect on the command tree item.

VI. CONCLUSIONS AND FUTURE WORK

The improved algorithm named as PISPO based on the character of evidence record, which needed to update a little sometime, which not only reduce the overall number of tree branches but also update the tree in real time. When adding new criminal record or some record has been changed, flag should distribute the item for a prefix tree updates. Items and flags in a transaction are inserted into the tree based on a descending order of frequency. The tree is reconstructed once the proportion of the edit distance of items in the sorted order changes above a certain threshold. This algorithm is based on a novel data and computation distribution scheme, which eliminates communication among computers virtually and makes it possible for us with the MapReduce model. We demonstrated that the algorithm is effective when on massive data scene should be mining. Our future work will extract and correlate the evidence stored by the jpg, rmvb, doc, wvm and so on based on non-relational database such as MongoDB.

REFERENCES

[1] J. W. Han, J. Pei, and Y. W. Yin, "Mining frequent patterns without candidate generation," Proceedings of the 2000 ACM SIGMOD, June, 2000, Dallas, TX USA, pp. 1-12.

[2] Le Wang∗ , Lin Feng∗ , Jing Zhang, Pengyu Liao, "An Efficient Algorithm of Frequent Itemsets Mining Based on MapReduce", Journal of Information & Computational Science 11:8 2014) 2809–2816 May 20, 2014.

[3] S. K. Tanbeer, C. F. Ahmed, and B. S. Jeong, et al, "CP-Tree: a tree structure for single-pass frequent pattern mining," Advances in Knowledge Discovery and Data Mining, Springer, LNCS, Vol. 5012, 2008, pp. 1022-1027.

[4] Sankalp Mitra1, Suchit Bande2, Shreyas Kudale3, Advait Kulkarni4, Asst. Prof. Leena A. Deshpande, "Efficient FP Growth using Hadoop - (Improved Parallel FP-Growth)", International Journal of Scientific and Research Publications, Volume 4, Issue 7, July 2014 1 ISSN 2250-3153.

[5] J. Dean, and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communications of the ACM (CACM), Vol. 51, No. 1, January 2008, pp. 107-113.

[6] S. L. Garfinkel, "Digital forensics research: the next 10 years," Digital Investigation (the Proceedings of DFRWS '10), Vol. 7, Supplement, August 2010, pp. s64-s73.

[7] L. Zhou, Z. Y. Zhong, and J. Chang, et al, "Balanced parallel FP-Growth with MapReduce" IEEE Youth Conference on Information Computing and Telecommunications, November, 2010, Beijing, China, pp. 243-246.

[8] S. G. Totad, G. RB, and P. P. Reddy, "Batch processing for incremental FP-tree construction," International Journal of Computer Applications, Vol. 5, No. 5, August 2010, pp. 28-32.

[9] Y. S. Koh, and G. Dobbie. "SPO-tree: efficient single pass ordered incremental pattern mining," Data Warehousing and Knowledge Discovery (DaWaK 2011), Springer, LNCS 6862, 2011, pp. 265-276.

[10] H. Y. Li, Y. Wang, and D. Zhang, et al, "Pfp: parallel fp-growth for query recommendation," Proceedings of the 2008 ACM conference on Recommender systems (RecSys '08), October 23-25, 2008, Lausanne, Switzerland, pp. 107-114.