Enhanced Probabilistic Static Load-Balancing of Parallel Mining of Frequent Item set

Chetan A. Gole[1] Prof. Yogesh S. Patil[2] Prof. Dinesh D. Patil[3]
[1]ME Student [2]Assistant Professor [3]Head of Dept.
[1,2,3]Department of Computer Science & Engineering
[1,2,3]SSGBCOET, Bhusawal, Maharashtra India

Abstract— *To propose a protocol for secure mining of association rules in horizontally distributed databases. The current leading protocol is that of Kantarcioglu and Clifton. Our protocol, like theirs, is based on the Fast Distributed Mining (FDM) algorithm of Cheung et al. which is an unsecured distributed version of the Apriori algorithm. The main ingredients in our protocol are two novel secure multi-party algorithms — one that computes the union of private subsets that each of the interacting players hold, and another that tests the inclusion of an element held by one player in a subset held by another. Our protocol offers enhanced privacy with respect to the protocol. In addition, it is simpler and is significantly more efficient in terms of communication rounds, communication cost and computational cost.*

Keywords— Data mining, frequent sequence mining, parallel algorithms, static load-balancing, probabilistic algorithms

I. INTRODUCTION

Frequent pattern mining is an important data mining technique with a wide variety of mined patterns. The mined frequent patterns can be sets of items (item sets), sequences, graphs, trees, etc. Frequent sequence mining The GSP algorithm presented is the first to solve the problem of frequent sequence mining. As the frequent sequence mining is an extension of item set mining, the GSP algorithm is an extension of the Apriori algorithm [3]. The Apriori and the GSP algorithms are breadth-first search algorithms. The GSP algorithm suffers with similar problems as the Apriori algorithm: it is slow and memory consuming. As a consequence of the slowness and memory consumption of algorithms described in [3], other algorithms were proposed. The two major ideas in the frequent sequence mining are those of Zaki and Pei, Han. These two algorithms use the so-called prefix-based equivalence classes (PBECs in short), i.e., represent the pattern as a string and partition the set of all patterns into disjoint sets using prefixes. The two algorithms differ only in the datastructures used to control the search.The algorithms are fast. How-ever, when the sequential algorithm runs for too long there is a need for parallel algorithms. Such as the one described in this paper. There is a very natural opportunity toparallelize an arbitrary frequent sequence mining algorithm: partition the set of allfrequent sequences using the PBECs.

The PBECs are created, scheduled, and executed on the processors. Because the PBECs are scheduled once, we talk about static load-balance of the computation. This approach has one advantage: it prevents repeated huge transfers of data among nodes (the data is transferred once among processors); and one disadvantage: estimating the size of a PBEC is a computationally hard problem. Currently, there do not exist scalable parallelization of these algorithms. There are two kinds of parallel computers: shared memory machines and distributed memory machines. Parallelizing on the shared memory machines is easier than parallelizing on distributed memory machines. The dynamic load-balancing is easy on shared memory machines, as the hardware supports easy parallelization: the processor has access to the whole database. For this work, distributed memory machines, i.e., cluster of workstations, was used. Sampling technique that statically load-balance the computation of parallel frequent item set mining process, are proposed in [13]. In these three papers, the so-called double sampling process and its three variants were proposed. This work extends the idea presented in [14] to parallel frequent sequence mining algorithm. The double sampling process is enhanced by introducing weights that represents the relative processing time of the algorithm for a particular PBEC.The paper is organized as follows: Section 2 de-scribes basic notion used through the whole article, Section 3 overviews the sequential Prefixspan algorithm, Section 4 shows related work and discuss the difficulties and challenges of parallel mining of frequent sequences. In Section 5, we briefly overviews the proposed algorithm. overviews theory behind sampling and approximate counting a standard tool for solving #P-hard problems.

1	h(1; 2; 3)(1; 2; 4)(2; 4; 5)i
2	h(1; 4)(2; 3; 4; 5)(1)(2; 4; 5)i
3	h(3; 4; 5)(1; 2; 3)(1)i
4	h(3)(1)(2; 3; 5)i
5	h(2; 5)(1; 3; 5)(1; 2; 5)(2; 4; 5)i

Fig-1: An example database, which will be used to demonstrate the algorithm.

II. RELATED WORK

1. Keying hash functions for message authentication

Authors: M. Bellare, R. Canetti And H. Krawczyk-

The use of cryptographic hash functions like MD5 or SHA-1 for message authentication has become a standard approach in many applications, particularly Internet security protocols. Though very easy to implement, these mechanisms are usually based on ad hoc techniques that lack a sound security analysis.Presented new, simple, and practical constructions of message authentication schemes based on a cryptographic hash function. Our schemes, NMAC and HMAC, are proven to be secure as long as the underlying hash function has some reasonable cryptographic strength. Moreover we show, in a quantitative way, that the schemes retain almost all the security of the underlying hash function. The performance of our schemes is essentially that of the underlying hash function. Moreover they use the hash function (or its compression function) as a black box, so that widely available library code or hardware can be used to implement them in a simple way, and replace ability of the underlying hash function is easily

2. FairplayMP - A system for secure multi-party computation

Authors: A. Ben-David, N. Nisan, and B. Pinkas

They have present FairplayMP (for "Fairplay Multi-Party"), a system for secure multi-party computation. Secure computation is one of the great achievements of modern cryptography, enabling a set of untrusting parties to compute any function of their private inputs while revealing nothing but the result of the function. In a sense, FairplayMP lets the parties run a joint computation that emulates a trusted party which receives the inputs from the parties, computes the function, and privately informs the parties of their outputs. FairplayMP operates by receiving a high-level language description of a function and a configuration file describing the participating parties. The system compiles the function into a description as a Boolean circuit, and performs a distributed evaluation of the circuit while revealing nothing else. FairplayMP supplements the Fairplay system , which supported secure computation between two parties. The underlying protocol of FairplayMP is the Beaver-Micali-Rogaway (BMR) protocol which runs in a constant number of communication rounds (eight rounds in our implementation). We modified the BMR protocol in a novel way and considerably improved its performance by using the Ben-Or-Goldwasser-Wigderson (BGW) protocol for the purpose of constructing gate tables. We chose to use this protocol since we believe that the number of communication rounds is a major factor on the overall performance of the protocol. We conducted different experiments which measure the effect of different parameters on the performance of the system and demonstrate its scalability. (We can now tell, for example, that running a second-price auction between four bidders, using five computation players, takes about 8 seconds.)

3. Secret sharing homomorphisms: Keeping shares of a secret

Authors: J.C. Benaloh lackley and Shamir

Independently proposed schemes by which a secret can be divided into many shares which can be distributed to mutually suspicious agents. This paper describes a homomorphism property attained by these and several other secret sharing schemes which allows multiple secrets to be combined by direct computation on shares. This property reduces the need for trust among agents and allows secret sharing to be applied to many new problems. One application described gives a method of verifiable secret sharing which is much simpler and more efficient than previous schemes. A second application is described which gives a fault-tolerant method of holding verifiable secret-ballot elections.

4. Privacy-preserving graph algorithms in the semi-honest model

Authors: J. Brick ell and V. Shmatikov

We consider scenarios in which two parties, each in possession of a graph, wish to compute some algorithm on their *joint graph* in a privacy-preserving manner, that is, without leaking any information about their inputs except that revealed by the algorithm's output. Working in the standard secure multi-party computation paradigm, we present new algorithms for privacy-preserving computation of APSD (all pairs shortest distance) and SSSD (single source shortest distance), as well as two new algorithms for privacy-preserving set union. Our algorithms are significantly more efficient than generic constructions. As in previous work on privacy-preserving data mining, we prove that our algorithms are secure provided the participants are "honest, but curious

5. A fast distributed algorithm for mining association rules-

D.W.L. Cheung, J. Han, V.T.Y. Ng, A.W.C. Fu, and Y. Fu Says that With the existence of many large transaction databases, the huge amounts of data, the high scalability of distributed systems, and the easy partitioning and distribution of a centralized database, it is important to investigate efficient methods for distributed mining of association rules. The study discloses some interesting relationships between locally large and globally large item sets and proposes an interesting distributed association rule mining algorithm, FDM (fast distributed mining of association rules), which generates a small number of candidate sets and substantially reduces the number of messages to be passed at mining association rules. A performance study shows that FDM has a superior performance over the direct application of a typical sequential algorithm. Further performance enhancement leads to a few variations of the algorithm.

III. METHODOLOGY

The implementation was executed on the CESNET metacentrum on the zegox cluster. Each zegox's node contains two Intel E5-2620 equipped with 1 Infiniband. Nodes were exclusively allocated for these measurements and used a maximum of 5 cores per node (to avoid influences from other jobs).The speedup of the proposed algorithm is evaluate on five datasets 1) data from our faculty web (fitweb);2) protein dataset (dataset with single item per event);3)three artificially created datasets using the IBM data generator. It should be noted that the execution of the sequential (parallel) algorithm on the protein dataset is optimized using the fact that there is one item per event. The first real-world dataset, the fitweb dataset, which was constructed from the http log files, by considering fetches by one ip address. One event was made from ids of the resources fetched in a window of 10 seconds by one IP address. From the transactions, items were removed if presented in every transaction. The datasets generated using the IBM generators are denoted using a string ThnumiIhnumiPhnumiPLhnumiSLhnumiTLhnumi. The parameters of the datasets generated using the IBM generator are the following: 1) T stands for number of transactions in thousands; 2) I stands for number of items in thousands 3) P is the number of patterns used for generation of one event; 4) SL is the average number of events in a transaction 5) TL is the average transaction number of items in an event. It should be noted that the IBM generator does not generate entirely random datasets. The IBM generator embeds sequential patterns in transactions, see [2] for details. It is well known that the real datasets and the datasets generated using IBM generator have different characteristics. The sampling parameters and the database sizes are shown in Figure 5. In the case of the datasets generated by the IBM database generator, we use the sample

of size of 5% of the database. In the case of protein dataset and fit web dataset, we set the same absolute size of the sample as for the generated datasets with 100'000 transactions. The supports were set, so that the execution time is in the range of minutes up to hours for every dataset (maximum 25 hours for the T1000I1P500PL5SL5TL15 dataset).

In Figure 4 are shown the speedups of our method. All of the proposed methods have speedups up to 20– 32 on 40 processors for lower values of support (the speedup 32 on 40 processors is for the protein dataset). These three methods exhibits similar performance on the datasets generated using the IBM generator. The speedups are lower, for higher values of support. For example, the T1000I0.3P500PL5SL5TL15 dataset has quite good speedups for supports 10'000 and 8'750 and bad speedups for supports 30'000 and 20'000. The reason is that the number of frequent sequences is very small: 892 for the support 30'000 and 4948 for the support 20'000. The execution time for the support 20'000 is 384 seconds (6 minutes). The Amdahl's law applies in such a case. It does not make much sense to parallelize such tasks and one should expect bad speedups. These results demonstrate the performance of the algorithm on small tasks. Similar problems, e.g., bad speedup for higher values of support, can be seen on other datasets From the speedup graphs, see Figure 4, it is not clear whether the k-depth weighting tree performs al-ways better then the plain sampling method, The purpose of the k-depth weighting tree and the sample weighting tree is to handle the datasets with unbalanced support and the number of embed-dings over frequent sequences, see Section 5. The experiments show that the estimates of the running time using the two weighting approaches performs no worse than the unweight (plain) estimate.

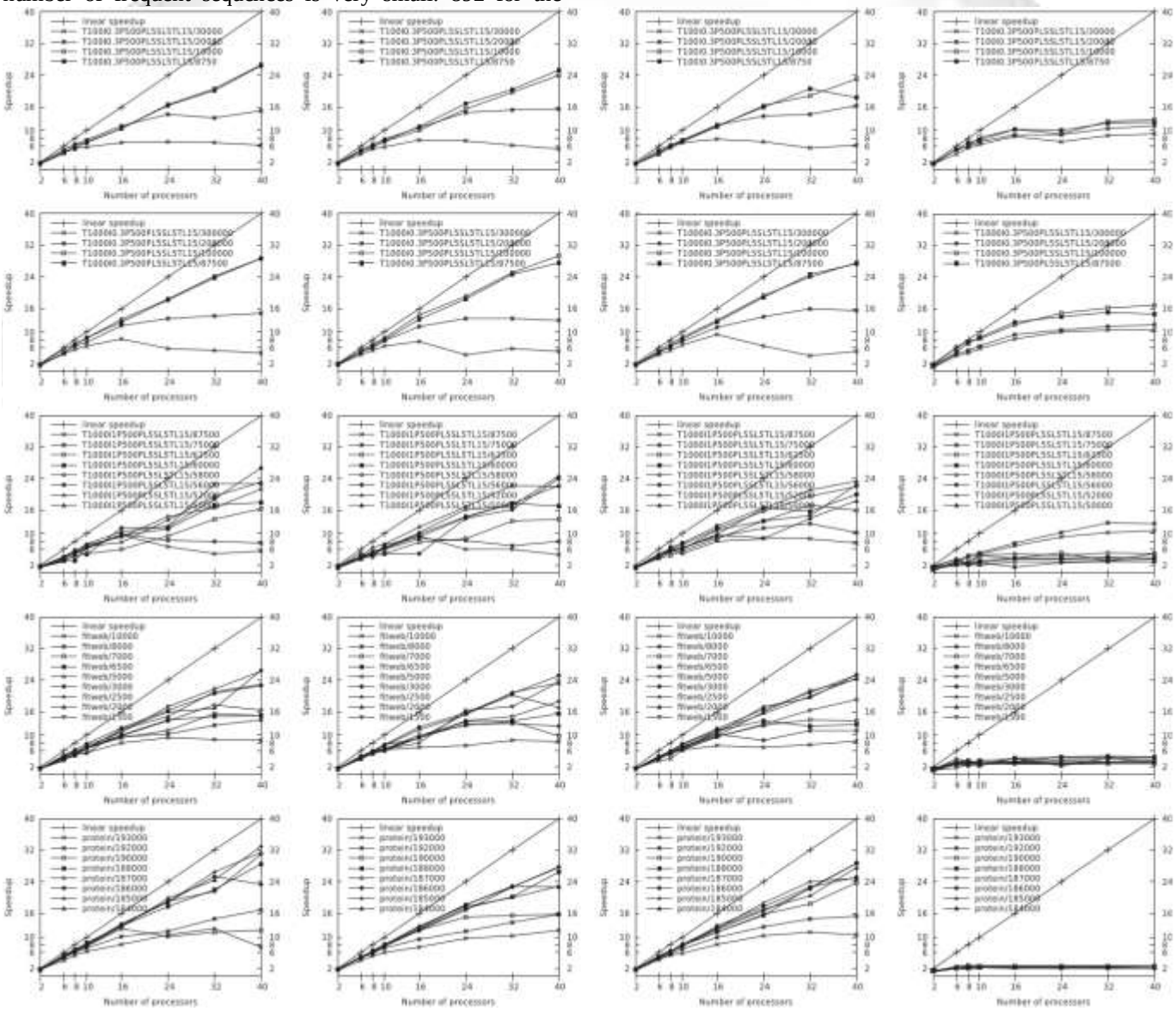

Fig-2: Each line of graphs represents one dataset. In each line the graphs are organized as follows (from left to right): 1) k-depth weighting tree; 2) sample weighting tree; 3) plain; 4) selective sampling

IV. RESULTS & DISCUSSION

Conclusion from the experiments is that there exists some lower bound on the amount of computation. The bound is given by the amount and length of frequent sequences. It is also possible to observe that there exist a particular number of processors with highest speedup for a particular support. An example is the T1000I1P500PL5SL5TL15 dataset with supports 87'500 and 75'000.When the support has a high value, the amount of the frequent sequences is small and they tend to be short. The exact numbers are unimportant as they can change among different datasets. performs worse than our new approach. One of the reason could be that the selective sampling makes PBECs with prefix of size 1 and there is no obvious ex-tension to longer prefixes of size>1. In these experiments, the speedups are reason-able when the number of frequent sequences is approximately greater than $10^0$000. The bad speedups for T100I0.3P500PL5SL5TL15 (supports 30'000 and 20'000) and T1000I0.3P500PL5SL5TL15 (supports 300'000, 200'000) is clearly given by the amount of computation (which is connected to the number of resulting frequent sequences). The same holds for all computations done on the datasets with two highest support values.

V. CONCLUSION & FUTURE SCOPE

Proposed a protocol for secure mining of association rules in horizontally distributed databases that improves significantly upon the current leading protocol in terms of privacy and efficiency. One of the main ingredients in our proposed protocol is a novel secure multi-party protocol for computing the union (or intersection) of private subsets that each of the interacting players hold. Another ingredient is a protocol that tests the inclusion of an element held by one player in a subset held by another. Those protocols exploit the fact that the underlying problem is of interest only when the number of players is greater than two. Can apply Genetic Algorithm on Generated Item set to get more appropriate results.

REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.
[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD Conference*, pages 439–450, 2000.
[3] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *STOC*, pages 503–513, 1990.
[4] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Crypto*, pages 1–15, 1996.
[5] A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP - A system for secure multi-party computation. In *CCS*, pages 257–266, 2008.
[6] J.C. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret. In *Crypto*, pages 251–260, 1986.
[7] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *ASIACRYPT*, pages 236–252, 2005.
[8] D.W.L. Cheung, J. Han, V.T.Y. Ng, A.W.C. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. In *PDIS*, pages 31– 42, 1996.
[9] D.W.L Cheung, V.T.Y. Ng, A.W.C. Fu, and Y. Fu. Efficient mining of association rules in distributed databases. *IEEE Trans. Knowl. Data Eng.*, 8(6):911–922, 1996.
[10] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.
[11] A.V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, pages 217–228, 2002. [12] R. Fagin, M. Naor, and P. Winkler. Comparing Information without Leaking It. *Communications of the ACM*, 39:77–85, 1996.
[13] M. Freedman,Y. Ishai, B. Pinkas, and Rheingold. Keyword search and oblivious pseudorandom functions. In *TCC*, pages 303–324, 2005.
[14] M.J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, pages 1–19, 2004.
[15] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.