Available online at www.sciencedirect.com

ScienceDirect

Procedia - Social and Behavioral Sciences 129 (2014) 518 – 526

Procedia Social and Behavioral Sciences

ICIMTR 2013

International Conference on Innovation, Management and Technology Research, Malaysia, 22 – 23 September, 2013

An Efficient Log File Analysis Algorithm Using Binary-Based Data Structure

Sallam Osman Fageeri[a*], Rohiza Ahmad[b]

[a,b]Department of Computer & Information Science Universiti Teknologi, PETRONAS, Malaysia

Abstract

Log files provide valuable insight to previous history of system's usages. Using the information from a log file can help improve future access of a system. However, log files often contain huge amount of data which require significant amount of time to be processed. Even though some popular algorithms already exist to handle this, it is still an open challenge for researchers to further improve them. Hence, in this paper, a novel binary-based approach for frequency mining of a database log file is presented. The approach includes the use of a new algorithms along with its supportive data structures. Construction of the approach began with evaluation of some of the existing methods and identifying their drawbacks. From there the new algorithms were developed and tested. Initial experimentation of the approach reveals a significant improvement in terms of the execution time of the log file's frequency mining calculation.

Keywords: Log File; Frequency Mining; Binary-based Algorithm; Data Structure

1. Introduction

The Log file is an important document which is often kept by most systems such as e-business websites, and database systems, etc., to record their activities or transactions. Log files are usually flat files, where, each line or record in those files contains at least a timestamp, a combination of one or more event identifiers and the actual log message containing information of the event executed (Nagappan & Robinson 2011). For example, a log file containing database transactions might have information such as

[*] Corresponding author. *Tel.: 605 - 368 7477*
*E-mail a*ddress: rohiza_ahmad@petronas.com.my.

doi:10.1016/j.sbspro.2014.03.709

the type of transaction executed (e.g., the SELECT statement), the tables (e.g., STUDENT table) and the attributes (e.g., studentId and studentName) involved. In other words, log files provide the history and audit trail of events (Ilenia fronza, Alberto sillitti 2013). Due to the trace information kept, log files can be used for various purposes such as rolling back some records to their original data, identifying popular items to be recommended to customers and ranking most popular search results.

Furthermore, the more data kept in log files usually mean the more accurate results can be extracted from them. For example, no significant information can be concluded from a two lines log file as compared to log file which contains million lines of transactions. However, the huge size of log files incurs costs in the form of execution time when they are to be analyzed. Hence, reducing the latency of a user's request due to the time taken to analyze the log files has gained the attention of many researchers over the past few years (Gao et al. 1999; Jeswani et al. 2012; Sharifi et al. 2012; Murayama et al. 2012). With the increasing demand for more relevant user information, it has become essential to discover hidden information from the large and huge data repository such as the log files (Yu, 2011) in a shorter time period.

In this paper, a novel technique is presented which can help the analysis of the log files, in particular the frequency mining, to be less time consuming. The technique uses a binary- based solution for expediting the process. The solution presented in this paper includes the data structures and the algorithm that can be used for the process. As to proof that the technique works, a performance comparison of the technique to another popular log file analysis technique, i.e., Apriori technique, is also provided and discussed. Furthermore, since different systems capture different contents in their log files, in this paper, database log file is chosen to show the application of the technique proposed.

2. Literature review

This section reviews some of the previous research works which are related to log file usage and the approach used in analyzing them. The work discussed in (Helmy et al., 2008), for example, describes the purpose of analyzing the information stored in log files was to manage the bandwidth and the server capacity. For that purpose, the author has discussed various types of log files used such as transfer log, agent log, error log and referrer log. The Paper describes the works that must be done on IIS web server log, from the step of raw log file, till step before mining process can be initialized. However, in the paper, the author has not discussed the approach used in analyzing those log files in details. The paper describes the works that must be done on IIS web server log, from the step of raw log file, till step before mining process be initialized.

Besides the above, another work as reported in (Suneetha, 2009) describes a method that can help a system administrator of a web based application to improve the performance of the application. The paper is concerned with analysis of the web log data of the NASA website. Through analyzing the website's log file using the approach of in-depth analysis, valuable information can be found such as topmost errors and potential website visitors. In addition, the system performance has also been able to be improved by taking into consideration the occurred system errors which were captured in the log file. In this work, the log file's content was also used to capture the most active days and the least active days of the server. By having the above information, the proper days for shutting down the server can be determined. As for future work, the author has planned to use some data mining techniques such as classification, clustering and association to further improve the finding of frequent access patterns.

Furthermore, in Stermsek et al., (2007), user profiles, in particular user interests, are derived through analysis of the server log file and the meta data of the page content. The server log file takes some important and mandatory information such as user-id, requested page, user session information, and its content related meta-data for analysis. After implementing the analysis, a graph of files is obtained. The graph provides a view page of the user interest. In this work, the authors used combination of two analysis methods, i.e., statistical and graph analysis. For the log file analysis, the weight factor (amount of time spent on a page) is chosen instead of the frequency (number of times a page is accessed). The justification for that is due to the authors' belief that the time spent on a specific web page requested by a user is more useful as compared to the idea of just requesting a specific page for the purpose of finding links to other interesting pages. In addition to the above, in Mishra (2012) the authors have used a data mining algorithm called FP- Growth for exploring the most frequent access patterns. The access patterns are again generated based on the web log file data. Using the mentioned algorithm, the expected user interests can be easily determined. Moreover, the authors also claimed that the method is efficient for mining both short frequent pattern as well as long frequent pattern.

Based on extensive readings, it can be said that FP-growth (Han & Pei, 2000) is one of the popular algorithms besides Apriori (Agrawal, 1994) which has been constantly referred by most researchers in frequency mining of log files of database transactions. The algorithm has been used and sometimes extended by other authors such as in (Zhang & Ruan 2009; Vyas et al. 2010; Singh & Ram 2013; Li et al. 2012). FP-growth uses a data structure called FP-tree to achieve the designed goal. In the algorithm, whenever a line of a log file entry is read.

In comparison to the above, Apriori uses breadth-first approach in determining the most frequent access pattern (Burdick, et al., 2001). The algorithm starts with comparing the frequency of access to individual available items. The most frequently accessed item will be selected and paired with each one of the available items. Again, frequency of access will be counted and compared. The most frequent accessed pair(s) will be selected and combined with a third item. The comparison of frequencies, the selection of most frequently accessed combination(s) and the addition of an item to selected combination(s) activities will be repeated several times until one winner emerged. Apriori has been used by many research including those reported in (Ilayaraja 2013; Parack et al. 2012; Luan et al. 2012).

After looking at the existing algorithms, it can be seen that there are some drawbacks to them especially in terms of their time performance. This is largely due to the inefficient method in comparing an item or combination of them to the content of the log file. Usually string comparison will be used. If the item to be found contains multiple items, each of the items will need to be extracted first. After that, each possible combination will be constructed and their number of occurrences in the log file will need to be counted. The above process will require multiple scanning of the log file which obviously incurs unnecessary additional time cost. Besides, the cost in terms of space usage is also huge when handling the analysis process. Hence, we believe our approach which will be presented next takes care both of the issues mentioned above.

3. System architecture

Before presenting the log file analysis technique, we would like to highlight that this work is actually a small part of our bigger research which is to maximize the pre-fetching of cache content for mobile clients during period of disconnection. As such, Fig (1) depicts the system model or framework designed for the study. As illustrated, the framework is based on a 3-tier client/server architecture. The tiers are the client tier, the application server tier, and the database tier.

The clients in the first tier work as requesters of the service, while the application server in the second tier works as responder. A client can issue a request to the application server using a known structured language (e.g., SQL) asking about a part of the data which are kept in the database server of the third tier. The application server provides the process management services such as request receiver, main log file, request process engine and repository summary which are needed to provide response to the user request. In this paper, we will be concentrating on the repository summary part of the second tier where a novel technique for analyzing the log file will be presented and discussed.

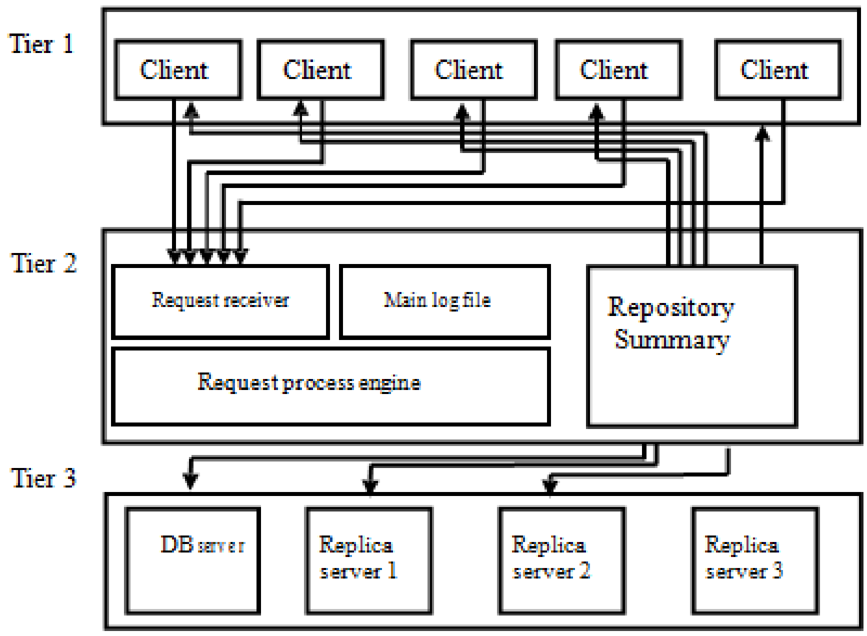

Figure 1: Binary-based pre-fetching cache content framework

4. Request Process Engine

With the assumption that data requested during a period of time are usually related to each other, we believe that the log file can help us to predict what are the needed data that can be pre-fetched to the client's cache later on. The previous requested items which are registered in the log file may lead to the finding of relationships among the items. Furthermore, the requested items which have some relationships may take the pattern and behavior of the related data which can be accessed in the nearest future. The one important part to the above is the use of data mining technique for discovering association between the related items in the log file (can also be called as the access history).

Consider the following assumption to find the association between the database tables and its attributes: Let $I = \{i_1, i_2 \ldots i_n\}$ be a set of binary data items. Let D be a Database and T be a set of database transactions, such that $\forall\ T \in D$, $T \subseteq I$. A transaction T contains a set of items X if and only if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \phi$. A rule $X \geq Y$ holds in the transaction set D with confidence c if c% of the transactions in D that contain X also contain Y. The rule $X \Rightarrow Y$ has support s in the transaction set D where s% of transactions in D contains X also contain Y. The problem of finding association rule can be decomposed as:

- Find all frequent set of items in the log file.
- Use the frequent item to generate the possible combination and association between the other related items.

5. Proposed Log File Preprocessing Technique

In this part, we present our Binary-based technique for conducting the frequency analysis of the log file. For the purpose of simplifying our discussion on the technique, in this paper we will highlight a database log file which contains user queries in the form of database tables only. However, using similar technique there should be no problem to include attributes as well in the log file. Table 1 shows sample content of a database log file that is used in this study (Note: *TID* stands for transaction id and *items used* stands for database tables queried).

Table 1. Sample Content of database log file

TID	Item Used
100	A, C, D
200	B,C,E
300	A,B,C,E
400	B,E

As mentioned earlier, our proposed technique is composed of the data structures as well as the algorithms for the frequency analysis activity. Fig (2) shows the three main stages involved in the technique, which are pre, during and post scanning of the log file. To make the discussion clearer, the data structures and algorithms required for the stages are also presented in the figure.

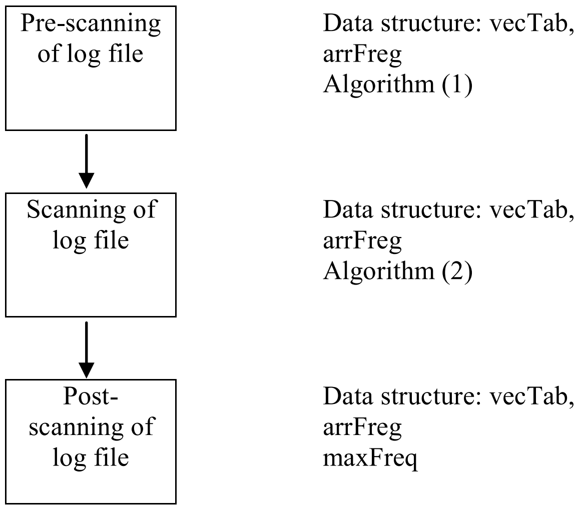
Pre-scanning of log file
Data structure: vecTab, arrFreg
Algorithm (1)
Scanning of log file
Data structure: vecTab, arrFreg
Algorithm (2)
Post-scanning of log file
Data structure: vecTab, arrFreg
maxFreq

Figure 2. Main stages of binary-based log file frequency analysis technique

5.1 Stage 1: Pre-scanning of log file

In stage 1 of the technique, two data structures will be created, the first is a vector named *vecTab* which contains the name of the actual tables read from a database and the other is a one dimensional array named *arrFreq* which will be used to store the frequency of certain combination of the tables queried. The size of *arrFreq* will depend on the number of database tables available. For example, if the number of tables n, is 5, then the number of rows will be $2^n = 2^5 = 32$ with index 0 to 31. The number of rows corresponds to the number of possible combinations of the 5 tables. This idea is adopted from the truth table concept in Boolean algebra. If we have 3 elements, e.g., A, B and C, then the potential combinations will be $2^3 = 8$ which are: null, A, B, C, AB, AC, BC and ABC. Table 1 shows the truth table representation of A, B and C, and its meaning in our technique.

Table 2. Truth table representation of technique (1=True, 0=False).

C	B	A	Combination	Array Index
0	0	0	null	0
0	0	1	A	1
0	1	0	B	2
0	1	1	AB	3
1	0	0	C	4
1	0	1	AC	5
1	1	0	BC	6
1	1	1	ABC	7

Having *arrFreq* in such a way will ease the process of incrementing relevant frequency. For example if we have 3tables named as A, B and C which are kept in *vecTab* locations 0, 1 and 2 respectively, then a query containing A and C will increment the frequency at index 5 of *arrFreq* by 1. This index is determined as follows: $2^0 + 2^2 = 1 + 4 = 5$ (Note: the first exponent, 0, is the location at which A is stored in *vecTab* and the second exponent, 2, is the location of C in *vecTab*). Looking back at the truth table in Table (2), we can see that index 5 is for TFT or 101 which is the binary for 5.

Algorithm (1) presents the algorithm used to populate the vector, i.e., *vecTab*, and create *arrFreq*.

```
Begin
    Query table names from database
    Set index to 0
    While not end of result set
        vecTab[index] ← $table_i$ Increment index by 1
    end while
    Create arrFreq[$2^n$]
    End
```

Algorithm 1. Populating *vecTab* and create *arrFreq*

5.2 Stage 2: Scanning of log file

The algorithm for this stage starts by reading the first record in the log file. For each table name mentioned in the record, its numeric code will be found by mapping it to the index where it is stored in *vecTab*. The code will be added to a formula named loc (see Formula 1) for determining which *arrFreq*'s frequency should be increased by 1.

$$Loc = \sum_{i=1}^{n} 2^{map(table_i)}$$

where $map(table_i)$ is the index of $table_i$ in *vecTab* Formula (1)

Reason for the formula has been presented in the previous discussion of the *arrFreq* array.

Algorithm 2 below presents the whole process of scanning the log file and populating the *arrFreq* array.

```
Begin
While not Eof (log)
    read transaction T from log for each $_{table_i}$ in T
        exp ← map($table_i$)
        loc ← loc + $2^{exp}$
    end for
    arrFreq[loc] ← arrFreq[loc] + 1 end while
End
```

Algorithm 2. Scanning log file and populating *arrFreq*

5.3 Stage 3:Post-scanning of log file

Once *arrFreq* has been populated based on the content of the log file, the array may be sorted using any sorting method to rank the frequencies along with their array indices from highest to lowest. The results of the sorting can be saved in another array named maxFreq, for example. Content of *maxFreq* can be something like Table 3.

Table 3. Sorted frequencies and their indices (*maxFreq*)

	Index	Frequency
0	13	100
1	29	98
	:	:

After that, finding which combination(s) is the most accessed will just require a conversion of the topmost index to its binary equivalent. For example, referring to Table (2) earlier, the highest frequency is 100 which is located at index 13, then the binary for 13 is 1101_2 which is DCA. In other words, the most frequently accessed term is the combination of D, C and A.

6. Experimental result

Based on our study, we undertook a set of experiments to evaluate the efficiency and accuracy based on the log file prediction methods. To validate the Binary-based technique we compare the result with the famous Apriori algorithm, the experiments were conducted on Intel® core™ 2 Duo CPU, 1.57GHz, and 01 GB of RAM computer. It was implemented in visual studio.net. The dataset used in the experiment is synthetic, which is provided by the QUEST generator of data generated from IBM's Almaden lab. The data set contains 62.5_{KB} to 4000_{KB} of records. Running the Apriori algorithm and our algorithm using the same dataset reveals that our algorithm managed to outperform Apriori in any data size of transactions, but when the dataset items were small. Furthermore, our algorithm shows tremendous difference in time performance when the size of data set grows larger. Fig (3) shows the result which revealed the enhancement performance of our new technique when compared with Apriori algorithm. The enhancement is in terms of execution time taken to complete the whole process from reading the content of the log file to the finding of the most frequently accessed term.

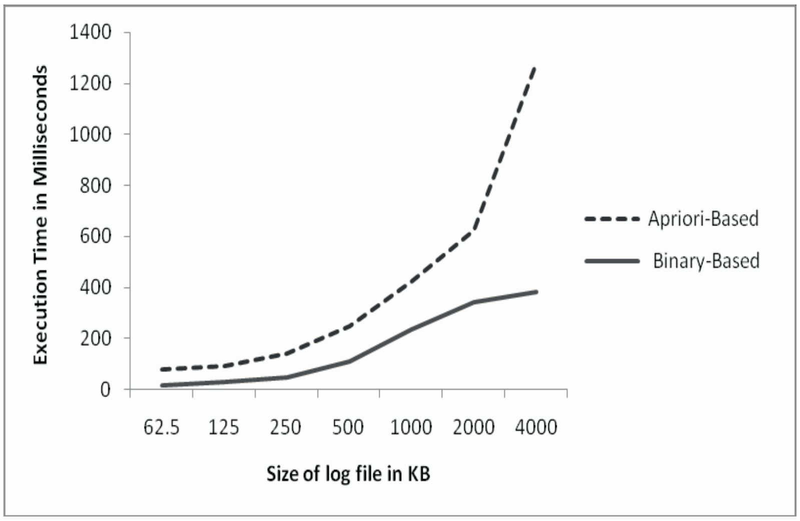

Figure 3. Execution time Binary-based vs. Apriori-based

7. Conclusion

Log files are often time contain huge amount of data which require significant amount of time to be processed. Thus, in this paper, a novel binary-based approach to perform frequency analysis of a database log file is presented. The contribution of this paper is a novel binary-based log file frequency analysis technique which is composed of the data structures needed and the algorithms. By implementing the technique, faster analysis of the log file can be done even though the amount of data might be huge. Furthermore, once the last array, *maxFreq* has been created, very easily relevant items can be retrieved and recommended to users. Comparing a binary-based concept with apriori algorithm shows significant improvement in execution time performance.

References

Agrawal, R., 1994. Fast Algorithms for Mining Association Rules 1 Introduction. *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 1215, 1–32.

Burdick, D., Calimlim, M. & Gehrke, J., 2001. MAFIA : A Maximal Frequent Itemset Algorithm for Transactional Databases. *IEE*E, 443–452.

Gao, L., Zhang, Z. & Towsley, D., 1999. Catching and Selective Catching : Efficient Latency Reduction Techniques for Delivering Continuous Multimedia Streams. *Proc. 1999 ACM Multimedia Conf*,.203–206.

Han, J. & Pei, J., 2000. Mining frequent patterns by pattern-growth: methodology and implications. *ACM SIGKDD explorations newsletter 2.2*,.14–20.

Helmy, M. et al., 2008. Data Pre-processing on Web Server Logs for Generalized Association Rules Mining Algorithm, 36(December).

Ilayaraja, M., 2013. Mining Medical Data to Identify Frequent Diseases using Apriori Algorithm. *Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*,194–199.

Ilenia fronza, Alberto sillitti, G., 2013. failure prediction based on log files using random indexing and support vector machines. *The journal of system and software*.

Jeswani, D. et al., 2012. Minimizing Latency in Serving Requests through Differential Template Caching in a Cloud. *2012 IEEE Fifth International Conference on Cloud Computing*, 269–276.

Li, N. et al., 2012. Parallel Implementation of Apriori Algorithm Based on MapReduce. *2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp.236–241. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6299286 [Accessed May 15, 2013].

Luan, R. et al., 2012. A dynamic improved apriori algorithm and its experiments in web log mining. *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, (Fskd), pp.1261–1264. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6234032.

Mishra, R., 2012. Discovery of Frequent Patterns from Web Log Data by using FP-Growth algorithm for Web Usage Mining. , 2(9), 311–318.

Murayama, D. et al., 2012. Low-Latency Dynamic Bandwidth Allocation for 100 km Long-Reach EPONs. *Journal of Optical Communications and Networking*, 5(1), 48.

Nagappan, M. & Robinson, B., 2011. Creating Operational Profiles of Software Systems by Transforming their Log Files to Directed Cyclic Graphs, 54–57.

Parack, S., Zahid, Z. & Merchant, F., 2012. Application of Data Mining in Educational Databases for Predicting Academic Trends and Patterns. *IEEE International Conference on Technology Enhanced Education (ICTEE)*, 1–4.

Sharifi, A. et al., 2012. 2012 IEEE / ACM 45th Annual International Symposium on Microarchitecture Addressing End-to-End Memory Access Latency in NoC-Based Multicores, 294–304.

Singh, J. & Ram, H., 2013. Improving Efficiency of Apriori Algorithm Using. *International Journal of Scientific and Research Publications*, 3(1), 1–4.

Stermsek, G., Strembeck, M. & Neumann, G., 2007. A User Profile Derivation Approach based on Log-File Analysis. *In Proceedings of IKE*, 258–264.

Suneetha, K.R., 2009. Identifying User Behavior by Analyzing Web Server Access Log File. , 9(4), 327–332.

Vyas, Z. V. et al., 2010. Modified RAAT (Reduced Apriori Algorithm Using Tag) for Efficiency Improvement with EP(Emerging Patterns) and JEP(Jumping EP). *2010 International Conference on Advances in Computer Engineering*, pp.238–240. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5532838 [Accessed May 15, 2013].

Yu, X., 2011. A Novel Approach to Mining Access Patterns. In *3rd International Conference on Awareness Science and Technology (iCAST)*.

Zhang, C. & Ruan, J., 2009. A Modified Apriori Algorithm with Its Application in Instituting Cross-Selling Strategies of the Retail Industry. *2009 International Conference on Electronic Commerce and Business Intelligence*, pp.515–518. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5189531 [Accessed May 15, 2013].