EFFICIENT MIXED MODE SUMMARY FOR MOBILE NETWORKS

Ahmed E. El-Din[1] and Rabie A. Ramadan[1]
[1] Computer Engineering Department, Cairo University
Cairo, Egypt

{Ahmed_ezzeldin@eng.cu.edu.eg, Rabie@rabieramadan.org}

ABSTRACT.

Cellular networks monitoring and management tasks are based on huge amounts of continuously collected data from network elements and devices. Log files are used to store this data, but it might need to accumulate millions of lines in one day. The standard name of this log is in GPEH format which stands for General Performance Event Handling. This log is usually recorded in a binary format (bin). Thus, efficient and fast compression technique is considered as one of the main aspects targeting the storage capabilities. On the other hand, based on our experience, we noticed that experts and network engineers are not interested in each log entry. In addition, this massive burst of entries can lose important information; especially those translated into performance abnormalities. Thus, summarizing log files would be beneficial in specifying the different problems on certain elements, the overall performance and the expected network future state. In this paper, we introduce an efficient compression algorithm based log frequent patterns. In addition, we propose a Mixed Mode Summary-based Lossless Compression Technique for Mobile Networks log files (MMSLC) as a mixed on-line and off-line compression modes based on the summary extracted from the frequent patterns. Our scheme exploits the strong correlation between the directly and consecutively recorded bin files for utilizing the online compression mode. On the other hand, it uses the famous "Apriori Algorithm" to extract the frequent patterns from the current file in offline mode. Our proposed scheme is proved to gain high compression ratios in fast speed as well as help in extracting beneficial information from the recorded data.

KEYWORDS. *Logs, Compression, Frequent Patterns*

1.INTRODUCTION

Cellular networks monitoring and management tasks are based on huge amounts of data that are continuously collected from elements and devices from all around the network [5][11]. Log files are considered the most famous storage file type for distributed systems like cellular networks. These files are collections of log entries and each entry contains information related to a specific event took place within the system. The importance of these files comes from the fact of using them to monitor the current status of the system, track its performance indicators, identifying frauds and problems to help in decision making in all aspects of the network [2][3].
Unfortunately, these files are recorded in a binary format (bin files) that needs huge effort to be parsed first, and then evaluated.

Initially, log files were used only for troubleshooting problems [8]. However, nowadays, it is used in many functions within most organizations and associations, such as system optimization and measuring network performance, recording users' actions and investigating malicious activities [2][3]. Unfortunately, these log files can be of order tens of gigabytes per day, and must be stored for a number of days as history. Thus, efficient and fast compression of these log files becomes very important issue facing the storage capabilities of the system. The essential problem falls not

only in the storage but also in the speed of extracting meaningful information from this huge data in a reasonable time.

Traditional compression techniques such as Lempel-Ziv [6] and Huffman Encoding [7] exploit the structures at the level of consecutive bytes of data. Therefore, the drawback of applying those techniques to cellular network log files is that in order to access a single tuple, the entire data page must be accessed first. On the other hand, those techniques did not give importance to the dependencies and the correlations between the attributes of the same log entries or between those of different log files [7].

Several compression techniques developed further targeting log files problems, and falls into two categories which are : 1) Offline compression in which it compresses the log files after been totally stored and it is considered as the most famous compressing approach; 2) Online compression in which the log file is compressed on the fly. Although offline compression leads to high compression ratios, this technique needs to scan the log file multiple times before compression. At the same time, online compression methods, in most of the cases, produces lower compression ratios than the offline compression methods.

Data mining and knowledge discovery methods are considered promising tools for systems' operators to gain more out of their available data [5][9]. These methods build models representing the data and can be used further in the rapid decision making. For large and daily updated log files, it is difficult or almost impossible to define and maintain a priori knowledge about the system, as new installed network components trigger new events. Fortunately, there is still a possibility to use Meta information that characterizes different types of log entries and their combinations. "Frequent patterns (FP)" is considered the most famous type of Meta information [7][8].

FP capture the common value combinations that occur in the logs. This type of presentation is quite understandable for experts and can be used to create hierarchical views. These condensed representations can be extracted directly from highly correlated and/or dense data.

Experts and network operators are not interested in each log entry. At the same time, this massive burst of log entries can lose important information, especially those translated into performance abnormalities of certain element in the network. In addition, indeed, rare combinations can be extremely interesting for system monitors. Thus from the experts and monitors point of view, frequent patterns summary can help fast detection of many of the network problems and abnormalities in the components performance over its lifetime.

In this paper, we propose a new lossless compression scheme based on mining frequent patterns named Mixed Mode Summary-based Lossless Compression for Mobile Networks log files. Our scheme first uses the famous Apriori technique for mining frequent patterns, assigns unique codes according to their compression gain, and uses these codes in compressing the file. MMSLC exploits the high correlation between the consecutively recorded log files by introducing mixed online and offline compression modes. MMSLC uses the FPs extracted from previous log files in offline mode to compress the current log files in online mode. At the same time, it extracts the frequent patterns from the current log files in offline mode to be used in compressing the new log files.

Our offline compression achieves high compression ratio, provides fast summary of the log file holding the frequent patterns to be used in network monitoring, while being able to restore all details if needed. MMSLC works on the attribute/tupelo-level to exploits the semantic structures in the recorded data. The rest of this paper is organized as follows: Section 2 introduces the

related work; GPEH log files structures and standards are elaborated in section 3; the frequent pattern generator algorithm is elaborated in section 4; MMSLC is introduced in section 5; MMSLC performance analysis is elaborated in section 6; finally this paper is concluded in section 7.

2 RELATED WORK

Jakub Swacha et. al. [1] described a lossless, automatic and fully reversible log file compression transformation. This transformation is not tuned for any particular type of logs, and is presented in variants starting from on-line compression variant to off-line compression one. They dealt with log files as plain text files in which every line corresponds to a single logged event. Thus, for the first variant, as the neighbouring lines are very similar in structure and content, this variant replaces tokens of a new line with references to the previous line, on byte level. However, single log often records events that may belong to more than one structural type. Thus, similar lines are not always blocked, but they are intermixed with lines differing in content or structure. The next variant searches the block, for each new line, for the line that returns the longest initial match is used as reference line. These variants are on-line schemes and addressed the local redundancy. However, this kind of compression affects the final compression ratio as it depends on the region around the line the algorithm and it searches for the reference. For off-line variant, it handles words frequency throughout the entire log. The dictionary has an upper limit and if reaches that, it became frozen. During the second pass, words within the dictionary are replaced with their respective indexes. The dictionary is sorted in descending order of frequencies; therefore frequent words have smaller index values than the rare ones. However, the limited size of dictionary may affect negatively on the final compression ratio as it freezes without adding higher frequent word. At the same time, compressing with respect to word frequencies does not give high compression ratio compared to that with respect to pattern frequencies.

Following the same concept, Rashmi Gupta et. at [4] proposed a multi-tiered log file compression solution, where each notion of redundancy is addressed by separated tier. The first tier handles the resemblance between neighbouring lines. The second tier handles the global repetitiveness of tokens and token formats. The third tier is general-purpose compressor which handles all the redundancy left. These tiers are optional and designed in several variants differing in required processing time and obtained compression ratio. However this compression scheme addresses the local redundancy and affects the final compression ratio as it depends on the region around the line the algorithm searches for the reference to compress the current line.

Kimmo et. at. [5] Presented a comprehensive log compression (CLC), method that uses frequent patterns and their condensed representations to identify repetitive information from large log files. A log file may be very large. During one day, a log file might accumulate millions of lines. Thus, for file evaluation of an expert, manual check is required. However, the most dangerous attacks are new and unseen for an enterprise defence system. CLC filters out frequently occurring events that hide other, unique or only a few times occurring events, without any prior domain knowledge. This separation makes it easier for a human observer to perceive and analyze large amounts of log data. However this scheme is used only in summarizing the frequent patterns in offline mode, and not used further in lossless compressing the corresponding log file.

These previously mentioned techniques and more others take into consideration that the log files are in a text format. However, the log files that we are dealing with are in a binary format. In order to deal with extracting such data, we had to deal with the recorded standard for each event. We stress on this part since we spend a lot of time working with different standards as well as their versions before reaching the step of compression and data mining. In the following, we briefly explain how the log files are recorded.

3 GPEH LOG FILES STRUCTURE AND RECORDING STANDARDS

Log files are used to record events that occured within the system. For cellular mobile and telecommunication, the events are related to every ascpet concerning the core network to the base stations ending with the cellular phones. Consecuently each log file can be of order tens of gigabytes per day, and must be stored for number of days as history. Each event within the log file follows certain format with respect to the different features recorded, the order and the size of each feature. Concerning the field of Telecommunication, these log files are in binary format. Thus for faster dealing with data, we parse each event and its corresponing messages into separate MySql database table. Finally we have database table for each unique GPEH event.

Taking GPEH Internal event "Admission Control Respose" as an example of the events recorded in log files. This event is captured whenever the admission control function responds to an admission request [15]. Table 1 shows the Structure of the event, the order of parameters, the number of bits of each parameter and the range of each parameter. Each single event recorded in the log file, has its own structure, parameters' order and sizes.

As mentioned before, the accumilated size of the log file is very huge. "Downlink Channelization Code Allocation" internal event for example accumilated nearly 1,622,900 rows (220 MB) for just 4 minites and "RRC Measurement Report" accumilated nearly 10GB for 15 minites and 68GB for 3 hours. Thus compressing these events is considered essential before storing as history.

Table 1. Internal Admission Control Event

Event ID 391 INTERNAL ADMISSION CONTROL RESPONSE		
EVENT_PARAM	PPS	Range
RESULT	3	0..3
FAILURE_REASON	8	0..78
……..		
GBR_UL	15	0..16000

4 FREQUENT PATTERN GENERATOR ALGORITHM

In this section, we state the basic terminologies that will be used further in the paper then introduce frequent pattern generator algorithm that will be used for lossless compression, as follow:

4.1.Basic Terminologies:

We assume normal relational table T with m records. Each record can be viewed as n (attribute - attribute value) pairs and each pair is considered as an item. Each record contains a set of n items and the group of items in the record is called itemset. The Support of an itemset is the number of records in the table T where this itemset occurs. The itemset is called Frequent Pattern (FP), if the support count is greater than or equal to a user-defined threshold. These frequent patterns can be used in summarizing the GPEH log files and compressing the data as well.

Pattern Gain is a measure of the contribution of this pattern in the table T [7]. The higher the gain is, the larger area covered by this pattern and the better compression can be gained by using this pattern. The storage gain of the pattern P can be calculated as in equation (1):

$$Gain(P) = \begin{cases} |P|.S(P), & if\ |P| > 1\ and\ S(P) > 1 \\ 0, & if\ |P| = 1\ or\ S(P) = 1 \end{cases} \quad (1)$$

where $|P|$ is the number of individual items in the pattern and $S(P)$ is the support of pattern P.

4.2.Frequent Pattern Generation:

Discovering all combinations of itemsets with support above the user-defined threshold requires multiple passes over the current data, and that is why Apriori algorithm is used [12][13][14]. The basic property used in Apriori algorithm and its extensions is that the large frequent itemsets is generated by joining smaller sized frequent itemsets, and removing the infrequent itemsets from the further search.

In each pass, the previously generated frequent itemsets are used, new items are added, and new candidate itemsets are generated. For the new pass, the support of these candidate itemsets is found and those with support higher than the threshold to be the seed for the next pass are selected. This process continues until no new itemsets are found in the data. The following pseudo code summarizes the algorithm:

1. The first pass on the data counts the frequency of each individual item to determine 1-itemsets and selects the frequent items as the next seeds.

2. The K^{th} pass consists of two phases.

(1) Use the frequent (K-1)-itemsets produced from the $(K-1)^{th}$ pass, add new 1-itemsets to generate the candidate K itemsets.
(2) Scan the data and calculate the support of each candidate K itemsets.
(3) Use the frequent K itemsets in the next pass.

3. Stop when there are no frequent itemsets.

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3
..	..	..
..	..	..

Figure 1: Simple GPEH log file.

For example, the relational table shown in Figure (1) has three columns; each column holds different values for three GPEH log file features. Using Apriori algorithm, the first scan gets the support and the gain for each individual attribute value in each column, as declared in figure 2. Using the user defined threshold, discard the frequent patterns with frequency below the specified threshold, this will keep faster compression by neglecting the infrequent patterns.

Using frequent patterns generated from the first pass, the different combinations of these features are generated as candidates for the next run, as shown in figure (3). During the second scan, the support and the gain for each itemset are calculated; the above threshold frequencies to be the candidates for the next run are also selected. The scheme stops when there is no an itemset above the threshold.

Finally, those frequent itemsets generated from the first run till the end are sorted in descending order with respect to their gain. Each frequent itemset is given a pattern in ascending order for big

overall compression gain, as shown in Figure (4). These patterns are used to compress the main log file in lossless mode.

Finding all combinations of itemsets with support above the user defined threshold could be solved using breadth-first search. However, it is not that efficient and it is considered time consuming especially for large GPEH log files. Thus, our scheme, as will be shown later, will compress the files based on selected features. Offline compression needs several passes on the current log file to extract the frequent patterns and to assign codes. At the same time, using old codes in online compressing all the upcoming log files, may causes lose of new frequent patterns and then affects negatively on the decision making process. Thus our proposed scheme mixes the offline and online compression modes to keep fresh summary data for accurate decision making. Traditional compression schemes operate in offline fashion to avoid losing new patterns. Our scheme made use of the strong correlations of the consecutive recorded log files and the previously stored frequent-pattern look-up table to compress the current log files in on-line fashion.

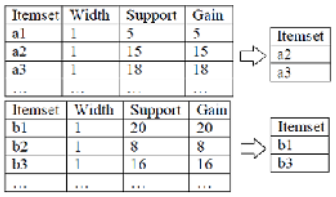

Figure 2: First Pass.

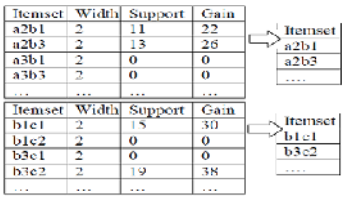

Figure 3: Second Pass.

Itemset	Gain	Pattern
$a_2b_1c_1$	40	P_0
$a_3b_1c_1$	22	P_2
…	…	…
b_1c_1	12	P_{14}
…	…	…
c_1	2	P_{24}
a_3	2	P_{25}

Figure 4: Codes assignment.

5 Mixed Mode Summary-Based Lossless Compression for log files (MMSLC):

In this section, our lossless compression scheme (MMLC) is introduced. The aim is to extract the frequent patterns from GPEH selected features, and then use these patterns to compress the GPEH log files. Our proposed algorithm falls into three phases are as follow:

5.1 Step1: Frequent Patterns Mining:

Log files hold several attributes, and each attribute has wide range of the different values. Compressing using all these attributes leads to huge storage save with unbounded time limit. On the other hand, experts and network operators are in need of fast summary from the current log file for certain features, as will be shown later in performance analysis section. Thus, for efficient and fast compression, summarizing GPEH log file will be with respect to some selected features. This would be beneficial in specifying the different problems on certain element, performance, and the expected future state.

Our scheme, MMLC, applies the frequent pattern Generator algorithm on those selected features. As discussed in the previous section, the frequent pattern Generator algorithm makes offline

summary from the different combinations of these selected features. This summary holds the patterns, their codes, their supports and gains.

5.2 Step2: Compression and decompression:

Our scheme assigns unique codes in the ascending order of the patterns' gains, as shown in figure 4. These codes are used for lossless compressing/decompressing lookup table for the GPEH log file. In addition, two other issues can lead to more compression gain:

1- There are some features with constant value, single value for the whole log file. MMSLC exploits these features, by replacing them with hints at the end of the compressed file, as shown in figure (5).
2- GPEH log entries are associated with time when the events are triggered. This time feature cannot be compressed for the wide variation of its values. However, some patterns are repeated number of times consequently for the same log file. MMSLU makes good use of this repetition by stating the start, the end time, and the frequency of the consecutive repetition, as shown in figure (6).

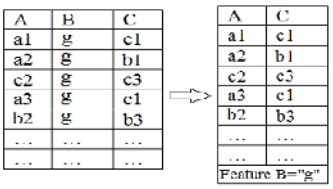

Figure 5: Compressing Uni-Valued Columns.

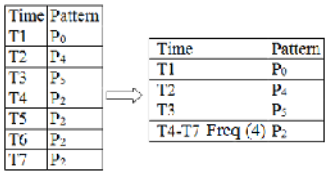

Figure 6: Compressing with respect to time.

5.3 Step 3: Mixed modes:

The compressed file with the used frequent pattern list represents the frequent and infrequent patterns along the scanned file. Although offline compression is the most widely used technique, this kind of compression delays the decision making process, that depends on the frequent and infrequent patterns obtained after scanning and compressing the log files. Thus, online compression can benefit decision makers for network monitoring, especially for huge amount of data reported by the telecommunication networks.

Our scheme proposed mixed mode between the offline frequent pattern mining process and the online compression using these patterns. MMSLC, mainly, assumes high correlation between the consecutive log files which is the case of cellular networks. Thus, the frequent patterns extracted from number of log files can be used to compress the direct next log files. As shown in figure (7), for the first history of log files, frequency patterns are extracted in offline mode. Then, the new log files are compressed in online mode using these patterns. During compression, MMSLC deals with these log files as new history and extracts new frequency patterns list in offline mode. Then, these patterns are used for compressing the new log files.

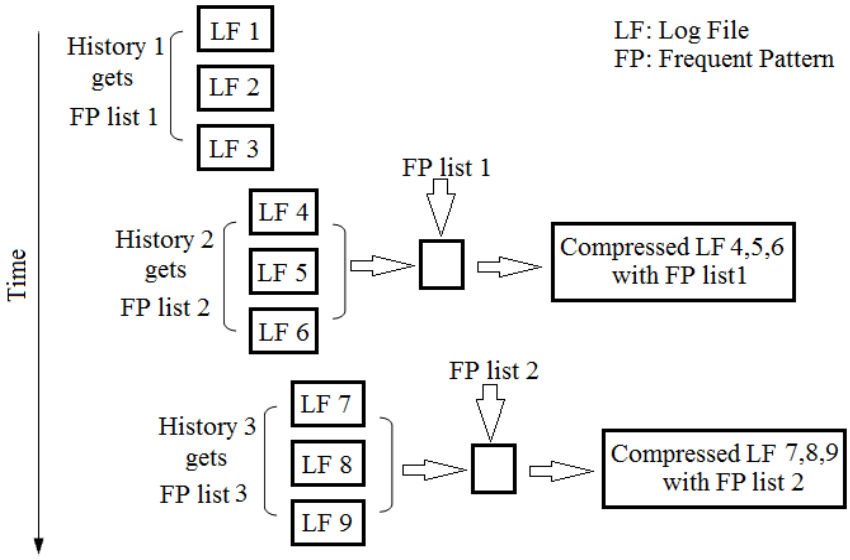

Figure 7: MMSLC Progress by Time.

6 Performance Analysis

In this section, we evaluate the performance of our proposed scheme, MMSLC. The scheme is implemented using java and the experiments are run on live recorded GPEH log files. Our algorithm runs on GPEH log files to compress and make summary with respect to certain parameter selected by experts. These parameters' frequencies are recorded in offline mode and the given patterns are presented according to their compression gain. Finally, these patterns are used to compress the log file. Our scheme was run on 3 different GPEH internal events as follows:

6.1 GPEH Events:

Our Scheme was run on GPEH "*System Block*" events to make summary on the blocked call problems recorded in the bin files. Experts usually collect the summary with respect to the different combinations of the cell id, the different source configuration and the different block reasons, instead of monitoring each record in the file that can leads to inefficient analysis of the system status. Figure 8 shows small part of this summary.

P:a11	c1=55406	Freq :1149
P:a12	c1=55407	Freq :1044
	.....	
P:a20	c1=55601	Freq :133
P:a0	SOURCE_CONF= 64	Freq :8542
P:a2	SOURCE_CONF= 65	Freq :5098
	.....	
P:a15	SOURCE_CONF= 70	Freq :484
P:a3	BLOCK_REASON= 33	Freq :4979
	.....	
P:a4	BLOCK_REASON= 41	Freq :3415
P:a1	BLOCK_REASON= 43	Freq :6448
P:b15	c1=55406 SOURCE_CONF= 64	Freq :1067
	.....	
P:b14	c1=55408 SOURCE_CONF= 64	Freq :1091
P:b12	c1=55406 BLOCK_REASON= 43	Freq :1149
	.....	
P:b10	c1=55408 BLOCK_REASON= 43	Freq :1185
P:b4	SOURCE_CONF= 64 BLOCK_REASON= 41	Freq :2255
	.....	
P:b1	SOURCE_CONF= 65 BLOCK_REASON= 33	Freq :4142
P:c0	c1=55602 SOURCE_CONF= 65 BLOCK_REASON= 33	Freq :2618
	.....	
P:c12	c1=55892 SOURCE_CONF= 65 BLOCK_REASON= 41	Freq :209

Figure 8: Part of the output Summary file from Block Calls internal event.

The second event was the "*Rab Establishment*" GPEH event, we collected summary with respect to the physical layer category, the source connection properties and the target configuration. The third event was the "*Soft handover Evaluation*" and we collected summary with respect to RNC module, the Event trigger and the Action. Experts can make use of the output summary for fast and accurate decision making, sample of the possible output summary of system block event are the source configuration shown in figure 9 and Cell ids shown in figure 10.

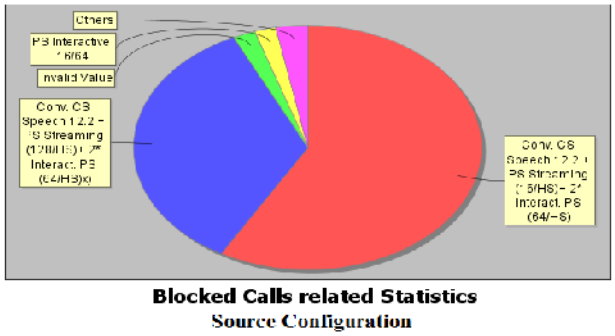

Figure 9: Source Configuration Summary representation.

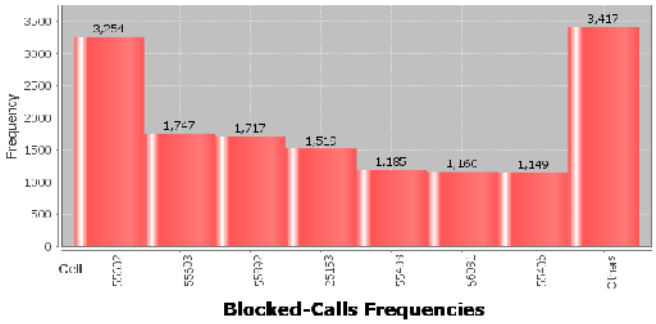

Figure 10: Cell ID Summary representation.

As stated before, in Step2, there are uni-valued features with constant value for the whole log file. MMLC exploits these features, states them in the summary file as shown in figure 11 and remove these columns from the compressed file. Also for the semi static valued columns, MMSLC gets the most frequent value in these fields, states them in the summary file as shown in figure 11. Thus for lossless compression, the log entries, with different values for those fields, will be written in the compressed file.

```
====StaticColumns====
   Column Name                    Value     Freq
SCANNERID                             2    15148
EVENTID                             431    15148
RNC1                                  9    15148
                    :::::::
REQUESTED_SERVICE_CLASS               0    15148
CURRENT_RAB_STATE                    31    15148
BLOCKING_PROCEDURE                    8    15148
     ----Partially StaticColumns---
   Column Name                    Value     Freq
UECONTEXT                          2188      149
REQUESTED_ORIGIN                      2    11981
REQUESTED_SF_DL                     640     9917
                    :::::::
TRAFFIC_CLASS_UL                      0    14957
GBR_UL                             8192    15120
GBR_DL                                0    15124
```

Figure 11: Static and Semi-Static Columns.

6.2 Memory and Time Analysis:

Figure 12 shows comparison between the main file, compressed file and the summary file in terms of the required storage memory. As shown, the compression gain is incomparable compared to the main file. These summaries are the most frequently used in telecommunications, instead of deep immerse in huge block of data.

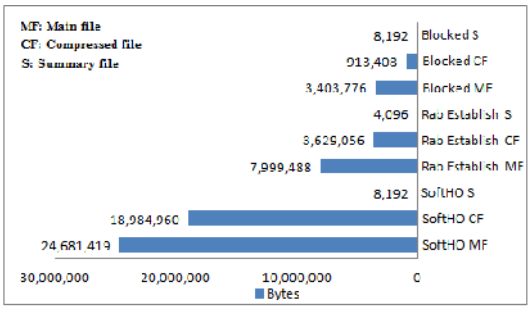

Figure 12: Memory needed to store the different types of files.

MMSLC allows faster online compression of log files using the offline summary of the previous "*Rab Establishment*" and "*Soft handover Execution*" events and this is very obvious in figure 13. This figure shows the time difference between offline compressing file after extracting its summary, which is the natural mode of most current compression techniques, and between online compressing the file based on offline saved summary on the three previously described GPEH events.

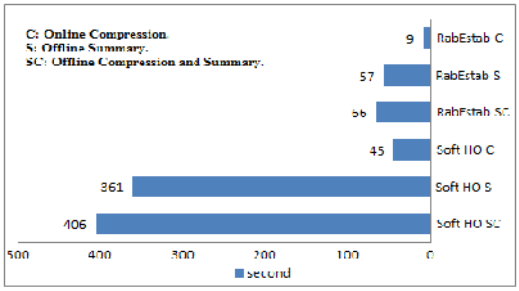

Figure 13: Time Compression.

6.3 GPEH Joint Events:

Our proposed scheme was applied on the join of three related GPEH events, named Internal_IMSI, Internal_SystemRelease and RRC_ConnectionRequest to collect statistics on the IMSI Roamers. IMSI Roamer is the mobile IMSI from outside the country and considered financial profit for the mobile system than the normal IMSI.

The collected statistics focus on displaying the number of Roamers connected to the current cells on Google Map and identifying the countries of these roamers.

As shown in figure 14, from the output summary of the events' join, this map shows the number of IMSI Roamers related to each cell with color indicators. These indicators divide the max number of Roamers into four groups, each indicates the importance of each cell. Green indicator means that this cell holds the number of roamers bigger than that of the yellow indicator. Thus for experts, cell with green indicator must have higher priority in maintenance and monitoring than that of the red indicator.

Figure 15 shows the same result in a different representation for experts; table shows the distribution of the roamers on the current active cells in the field.

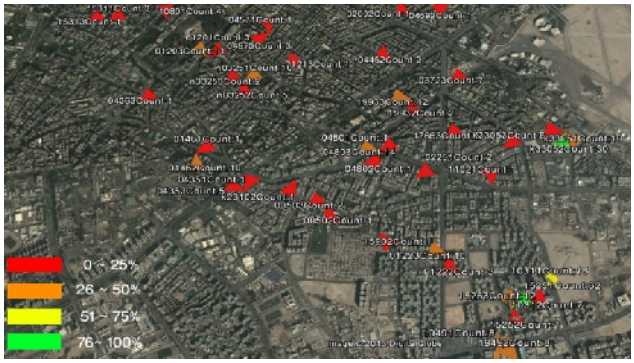

Figure 14: Google Map showing the number of Roamers connected to each cell.

Cells	Roamers Count
15251	32
k33052	30
19937	17
10311	16
k32994	15
04803	14
k36833	14
15253	12
19933	12
01203	11
00133	11
01223	10
01462	10
n03251	9

Figure 15: The number of Roamers connected to each cell.

Country distribution of IMSI Roamers is shown with two different representations, table and pie chart in figure 16 and 17 respectively. This kind of analysis is very important for experts from financial point of view.

Country	Roamers Count
Saudi Arabia	196
United Arab Emirates	50
Palestine	33
Jordan	20
Malaysia	16
Kuwait	13
Netherlands	13
Republic of Macedonia	9
Turkey	9
Libya	9
Denmark	8
China	8
Greece	6
Oman	6
Mexico	3
South Africa	3
Austria	3
Iraq	2

Figure 16: Country distribution of the current Roamers.

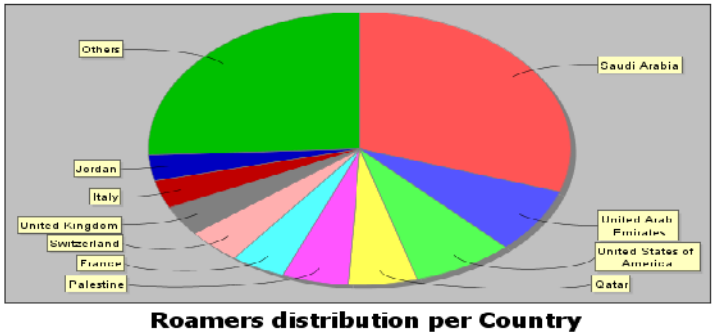

Figure 17: Pie Chart displaying the Top IMSI Roamers Countries.

7 CONCLUSION

In this paper, we introduced a new lossless compression scheme based on mining frequent patterns named Mixed Mode Summary-based Lossless Compression for log files (MMSLC). Our scheme uses the famous *Apriori technique* for mining frequent patterns to be used in compressing the file. Massive burst of entries can lose important information; especially those translated into performance abnormalities. In addition, experts and network engineers are not interested in each log entry. MMSLC provides summarization for the current log files that would be beneficial in specifying the different problems on certain elements, the overall performance and the expected network future state. MMSLC exploits the high correlation between the consecutively recorded log files by introducing mixed online and offline compression modes. MMSLC uses the FPs extracted from previous log files in offline mode to compress the current log files in online mode. At the same time, it extracts the frequent patterns from the current log files in offline mode to be used in compressing the new log files. Our proposed scheme is proved to gain high compression ratios in fast speed as well as help in extracting beneficial information from the recorded data.

ACKNOWLEDGMENTS.

This paper is part of “TEMPO” project funded by NTRA Egypt, and we also would like to express our gratitude to all TEMPO Team for their help and effort to accomplish this work.

REFERENCES

1. Christian Borgelt: Recursion Pruning for the Apriori Algorithm. Workshop on Frequent Itemset Mining Implementations-FIMI (2004).
2. Christopher H., Simon J. Puglisi, Justin Z.: Relative Lempel-Ziv Factorization for Efficient Storage and Retrieval of Web Collections. Proceedings of the VLDB Endowment (PVLDB), 2011, Vol. 5, No. 3, pp. 265-273.
3. Ericsson: General Performance Event Handling RNC Description, 2010.
4. Haiming Huang: Lossless Semantic Compression for Relational Databases. B.E., Renmin University of China, Beijing, P.R.China, 1998.
5. K. Hatonen: Data mining for telecommunications network log analysis, University of Helsinki, 2009.
6. Karahoca, A.: Data Mining Via Cellular Neural Networks In The GSM Sector. The 8th IASTED International Conference Software Engineering and Applications, Cambridge, MA, pp. 19-24, 2004.
7. Kimmo H., Jean-fancois B., Mika K., Markus M., Cyrille M.: Comprehensive Log Compression with Frequent Patterns. Data Warehousing and Knowledge Discovery DaWak, pp. 360-370, 2003.
8. Mr.Raj Kumar Gupta, Ms.Rashmi Gupta: An Evaluation of Log File & Compression Mechanism. International Journal of Advanced Research In Computer and Communication Engineering, VOL 1, ISSUE 2, 2012.

9. Piotr Gaqrysiak and Michal Okoniewski: Applying Data Mining Methods for Cellular Radio Network Planning. In Proceedings of the IIS'2000 Symposium on Intelligent Information Systems, Mieczyslaw, 2000.
10. Rakesh A., Heikki M., Ramakrishnan S., Hannu T., A. Inkeri V.: Fast Discovery of Association Rules. Advances in Knowledge dicovery and datamining book, pp. 307-328, American Association for Artificial Intelligence Menlo Park, CA, USA, 1996.
11. Rashmi Gupta, Raj Kumar Gupta, 2012. A Modified Efficient Log File Compression Mechanism for Digital Forensic in Web Environment. (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (4), 4878-4882.
12. Risto Vaarandi: A breadth-first algorithm for mining frequent patterns from event logs. In Proceedings of the 2004 IFIP International Conference on Intelligence in Communication Systems.
13. Salvatore O., Paolo P., Raffaele P.: Enhancing the Apriori Algorithm for Frequent Set Counting . International conference Data Warehousing and Knowledge Discovery-DaWak, pp. 71-82, 2001.
14. Sebastian D., Szymon G.: Subatomic field processing for improved web log compression. International Confereonce on Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2008.
15. Skibiński P., Swacha J.: Fast and efficient log file compression. CEUR Workshop Proceedings of 11th East-European Conference on Advances in Databases and Information Systems (ADBIS 2007).