Dynamic Log Analysis

András Lukács, Zsolt Nagy

Abstract—**This article reviews a new log analysis solution based on multidimensional data cubes. It introduces the process of dynamic log analysis, including real-time log compression during the log management, a new log parsing language, and the efficient regular expression processing engine for this language. The article assesses the new online analytic processing (OLAP) tool implemented for log analysis. The algorithm and software technology developments appearing in the resulting system are creating a new class of real time log processing and analysis tools.**

Index Terms—**log analysis, real-time compression, regular expression, OLAP, bitmap index**

I. INTRODUCTION

The article introduces the log analysis tool developed by KÜRT Co. and its technology. This is a professional software system capable of analyzing log data generated by large complex IT systems.

Companies and organizations put an increasing emphasis on the protection of their data and information systems. Log analysis is a technology-intensive method of IT security services. Log files describing the operation of computers, networks and applications are constantly collected. Logs tell the user what has happened to which system or device and when the event occurred. Logs data allow important events to be recognized like attacks to the network, policy violence, fraud, and technical defaults can also be detected and predicted. Last, but not least, logs may be suitable for detailed tracking of business processes.

As logs are generated in large quantities even in middle sized IT systems (they record as many as a million events per second), the collection and the storage of logs for longer than a few days can pose a serious problem. Processing the accumulated data is a long lasting and resource intensive challenge. Generally, only a hundred logged events out of a billion hold information about an incident, and only two of these require intervention; thus an analysis system is required that is able to process this large amount of logs automatically by efficient pattern recognition and filtering.

Manuscript received June 23, 2012. This work was supported in by the National Development Agency (Hungary) under the grant GOP-1.1.1-09/1-2009-0034 "Confidentiality-guaranteed, distributed data collection, real-time log analysis and automatic intervention system" financed by the European Union.

A. Lukács is with KÜRT Information Management and Data Recovery Co. and Department of Computer Science, Eötvös Loránd University (e-mail: lukacs@cs.elte.hu)

Zs. Nagy is with KÜRT Information Management and Data Recovery Co. (e-mail: zsolt.nagy@kurt.hu)

Current log analysis systems are based on monitoring the frequency of events (incidents), and the co-occurrence of particular previously defined events and simple rules. It often takes several hours to produce alerts and analysis results because of the limited speed of log processing methodology hindering timely intervention and damage prevention or even making them completely impossible. To solve these problems, we designed a completely novel concept of log analysis compared to previous solutions. The presented algorithm and software technology developments appearing in our system lead to a new class of real time log processing and analysis tools.

The concept of dynamic log analysis bridges rule based analysis tools of the past decade and future solutions based on fully automatic pattern recognition and semantics. The main idea of dynamic log analysis places human professionals in the center of decision making and supports them by (semi)automatic tools in every task.

Implementing the concept of dynamic log analysis involves many technological challenges. The first problem is the immense amount of log data which serve as the basis of the analysis. It is not uncommon to see an IT system producing more than 100 TB of log data every year. Processing such a large amount of data is clearly a *big data* problem [1], [2]. The second issue is how to read the state of the IT system from the logs during the (risk-) analysis. A further important criterion is the openness of the log analysis system. The analysis system has to be capable of receiving and understanding various types of log data, while on the other hand it has to be able to represent the new risks universally. It is also important that the reports and graphs must be understandable by professionals, and they have to be easily added to the existing processes of risk analysis.

The second section of the article describes the necessary steps for log collection and log normalization, which include real time compression, a new language for log processing with its general regular expressions engine effectively processing complex events composed by multiple log lines. The third section introduces the online analytic processing (OLAP) solution used for the analysis, which – besides the ordinary functions – is able to retrieve the original log lines belonging to the queried cell due to a multi-level indexing technique.

A unique feature of the developed OLAP engine is the extremely fast query response time for the regular OLAP operations, and for the log line retrieval too. The fourth section deals with analysis techniques based on the data stored in the OLAP cube. Finally in the fifth section we sum up the novelties of the log analysis tool, and we mention its use cases.

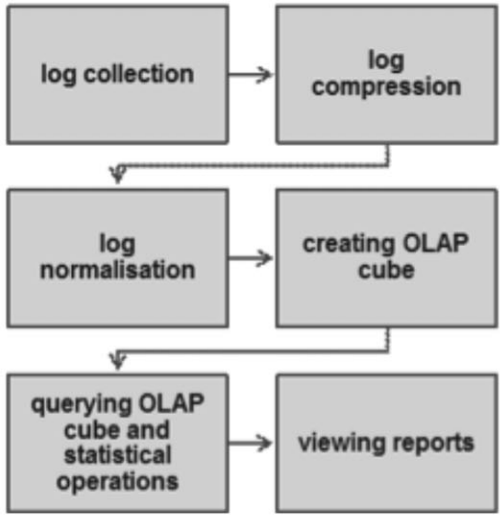

Fig. 1. Proposed process of dynamic log analysis.

II. LOG COLLECTION AND LOG NORMALIZATION

The collection process of logs generated by IT systems is supported by many well developed solutions [3]. Thus the first step in log processing is the long term storage of collected logs while maintaining efficient data access. Most log collection systems apply lossless compressors, for example *gzip*, to store the log files in a compressed format. Although such solutions can compress the log data to the tenth in size and so the reading speed from disk can increase with an order of magnitude, the data access speed of a log analysis tool also depends on the speed of decompression. Since compression rate and (de)compression speed are basically inversely proportional, it is a non-trivial problem to find the optimal solution for data compression and implementation, respectively.

The data access speeds can be well characterized with the achieved acceleration for data access speed of compressed logs compared to the data access speed of uncompressed logs. We have examined several real-time, extreme high-speed compressors, and *lz4* [4] showed the best results, which is optimized for data access based on a Lempel-Ziv algorithm. This comparison also included standard *gzip* (version 1.3.12) [5], *lzop* (version 1.02rc1) based on Lempel-Ziv-Oberhumer algorithm to optimize decompression [6], and *pigz* [7], which utilises the gzip parallel multi core architectures effectively.

Collected and stored logs are generated by heterogeneous devices (different operation systems, network devices, applications), thus the log formats are not standard; there are thousands of different log types to work with. On the other hand, logs are usually simple text files with information pieces separated by a character; they generally do not hold any deeper structures (like nested parentheses). This means that log parsing is typically a text processing problem. During this process, it has to be decided whether an input log matches a previous format or not, and then the needed data must be retrieved from the log for further processing. Because of the heterogenity of logs, the log analysis system must be easily expandable with new log formats, and log format descriptors created in an environment must be easily transferable to other similar environments.

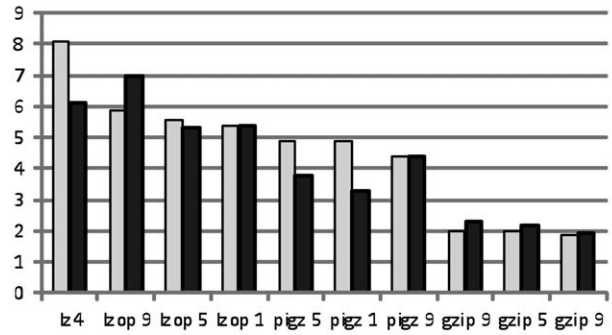

Fig. 2. Data access speed achieved by the use of different compressors evaluated on two data sets (light and dark columns). The height of columns show the achieved acceleration rate on compressed logs compared to the data access speed of uncompressed logs. Numbers behind gzip, lzop and pigz indicate the applied compression ratio (1 is the smallest and fastest; 9 is the largest and slowest). The compression speed of lzop 9 és gzip 9 is substantially worse than the results for other compressors.

Empirical evidence shows that log formats can be described with regular grammars like formal languages [8]. Furthermore it is known that a deterministic finite automaton (DFA) can be created for every regular language, and this automaton allows words belonging to a certain language to be recognized in linear time [9]. This theoretical solution also ensures that these automata are actually feasible in practice, and that they scale adequately in case of grammars describing log formats. The next step of pre-processing involves a further difficulty, since the data actually needed for the analysis must be retrieved from the log lines. In that phase, the filtering and transformation steps determined by professionals give the opportunity to introduce available risk analyst knowledge into the log analysis tool. Although the recognition of defined patterns can be tackled by the DFA-based algorithms, the extraction of a recognized pattern in itself surpasses the boundaries of DFA-based approaches. Therefore, the empirical testing of the available implementations for evaluating regular expressions was necessary.

Examining multiple regular expression processing softwares, we came to the following conclusions. The *flex* (Fast Lexical Analyzer) [10] is well suited to log format recognition. Flex provides the user with the opportunity to build a DFA from several regular expressions, which enables quick recognition. However, a technical problem is posed by the limited size of useable DFA, and the code does not support UTF8 coding, therefore the source code might require some modifications. A further problem of the conditional rules applicable to the extraction of patterns required to the analysis is that these rules reduce the speed of processing drastically.

Google *RE2* [11] is a regular expression matching library based on a highly efficient automaton theory. It offers high speed DFA and NFA (Non-deterministic finite automaton) based analysis. In case of NFA analysis, it is possible to retrieve parts from the input. In DFA mode, the matching

algorithm reads through the input once, pacing the automaton by characters. If we would like to retrieve the data, the DFA matching will not be appropriate. In that case NFA matching will be done, but the speed will be reduced to such an extent that it will be inadequate for log analysis.

The RE2 offered a good basis for the development of a prototype suitable for matching numerous samples simultaneously. The Set interface of RE2 made it possible to match multiple regular expressions with one DFA simultaneously. The speed of matching several thousand patterns simultaneously with such a DFA was similar to the speed of recognizing only one pattern. Although the size of the automaton can grow exponentially with the number of patterns, the size of a DFA built from almost a hundred thousand patterns still remains under 1 GB.

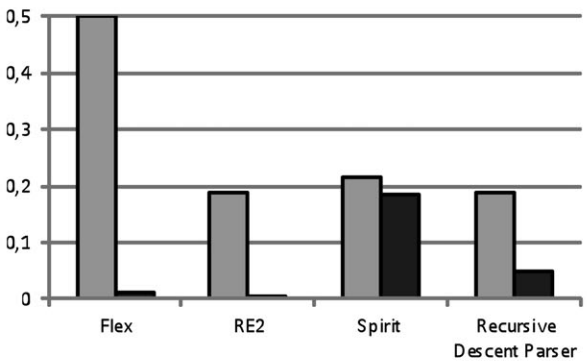

Fig. 3. Comparison of solutions applicable during the log normalization step. Lighter columns show the speed of pattern recognition, darker columns show the speed of pattern retrieval measured in GB/s when processing the test data set.

The *Spirit* [12] template metaprogram library developed for text parsing is part of the Boost program library [14]. With the help of Spirit, parsers can be generated in compile time for environment independent and regular language classes. An advantage of that is the possibility to optimize the parser during the translation, which may result in a significant increase in speed. The Spirit uses the recursive descent parser strategy [15], which - complemented by a state of the art programming approach – proved to be efficient in log format recognition and retrieval, too.

As text parsing rules are defined by a professional, a C++ translation step is required in case of Spirit for creating an actual text parsing program; this is not acceptable because the special (e.g.: legal) requirements of log analysis demand a high degree of stability. Although the compilation of the parser is feasible during run-time, optimizations during the translation were not possible in that case, so the speed decreased significantly. This led to the re-implementation of the recursive descent parser used by Spirit. The processing speed fell to the quarter, however, the resulting speed of text parsing and aggregation – measured in EPS (events per second) unit used by log analysis tools – was around 100 000 EPS on one core evaluated on a test data sample. This performance is comparable with the performance of the fastest log analysis tools available on the market. At the same time the log analysis tool has its own modular descriptive language designed for log analysis; highly efficient analyzers can be compiled during run-time.

III. MEANS OF ACHIEVING THE DATA MODEL

The starting point of the data model used in the log analysis is the multidimensional data cube applied in the OLAP (online analytical processing) tools [16]. Log records are represented in an aggregated form in the data cube. The log lines are represented with numbers in the data cube. Each number shows the corresponding counts of log lines where all special fields are the same. This comes from the experience of log analysis professionals; counts and other derived data based on counts (like statistics) provide excellent input for efficient analysis. Compared to the amount of log lines, there are generally much less nonzero numbers in the data cubes, so the data cube is a significant compression of the parsed log data. An important advantage of the multidimensional data model is the hierarchy of dimensions, which appears naturally in most cases, and can be exploited during the analysis. An example for this is the date, where the structure is the following: month, day, hour, minute etc. An extension compared to the ordinary OLAP implementations is that for each cell the corresponding log lines can be searched and retrieved due to a multiple inverse indexing.

Although the idea to use multidimensional data cubes and OLAP tools for special types of log lines occurred at the end of the second millennium especially in case of web server logs [17] [18], the OLAP has not been used yet for general purpose log analysis. The main reason for that might be the insufficient scaling of the available OLAP engines on big data. Nevertheless an early and sporadic source mentioning the use of OLAP tools in log analysis is worth noting [19].

In course of implementing the multidimensional data model of the log analysis tool, we posed and examined the question of applicability for several existing OLAP tools. We examined the applicability of OLAP implemented with relational database manager (ROLAP), and the use of memory resident OLAP (MOLAP) in log analysis. The examined ROLAP solution was an application executing the required OLAP functions needed for log analysis. This ROLAP solution is based on a MySQL [20] relational database manager. During the tests we used the *icCube* MOLAP tool [21]. As the performance of the tested tools was not sufficient for using it in a dynamic log analysis tool, a new OLAP engine implementation made it possible to fully meet the diverse needs (Fig. 4.)

The implemented OLAP engine (*Colap v1*) can quickly answer the count cube queries with the help of so-called

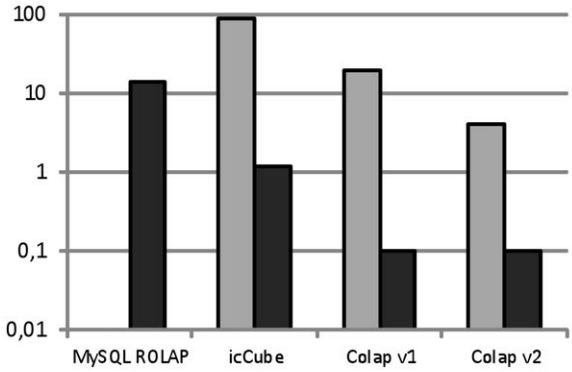

Fig. 4. Comparison of performances measured with different OLAP engines. The test data set was a log file with 6.5 million lines, it was 1.5 GB in raw size. Light columns indicate the loading speed of the gzip compressed test data from the memory to the OLAP tool and the building time of the data cube measured in seconds. Dark columns indicate the average time of OLAP operations on the built-up data cube measured in seconds. In case of MySQL ROLAP, the data loading time is not applicable.

compressed bitmap indexes. On one hand, the bitmap index is an incidence vector built upon all of the possible values of all the dimensions as coordinates; this vector describes one particular cell of the data cube by giving the values of the defining dimensions. On the other hand, the value of the corresponding cell also appears as the last element of the vector. This model is called indexed count cube model [21]. The bitmap index is a true representation of the cube. Every OLAP query can be traced back to the product of AND operations on the bit vectors of indexed count cube and of additions on the count values of cube record set. Vectors can be stored in a compressed format, AND operations can be done effectively even in this compressed format.

There are numerous methods for compressing bitmap indexes. Most solutions are based on run-length encoding, where sequences of the same values following each other in the bit vector are given with two data; the element of the sequence and the number of repetitions. The Byte-aligned Bitmap Code is used in most relational database management system. Another option is the application of Word-Aligned Hybrid (WAH) coding [22]. In case of particular fields like time, columns based bitmap vector encoding increases speed significantly [23]. This solution is the base of the OLAP engine version implemented by us, which could be further improved in terms of operation time by fine tuning the applied encoding (*Colap v2*).

IV. Analysis

Aggregated data cubes created from logs are the basis of the next steps in the analysis. Because of the diversity of log source systems, other tools were needed for the expansion and merging of the data cubes. Several operations available on multidimensional data cubes were implemented for the final phase of the analysis. Some of these are regular OLAP operations, others are statistical, event pattern comparison tools. What these tools have in common is that they create a low (two or three) dimension aggregate of the data cube storing the most detailed information, which thus can be visualized. OLAP queries can be done in a language similar to MDX (MultiDimensional Expressions) query language [25].

The log analysis tool can be managed via a web page, which is also the locus of displaying generated reports and visualizations (Fig. 5.). The number of feasible reports are inexhaustible, essentially all the information demanded by the analysts can be supplied by the system in real time. With the aid of the reporting system, one can configure other services starting from alerts to the periodic summary reports through the interface for defining reports.

The attained OLAP engine of the log analysis tool makes interactive analysis possible even when dealing with large data sets. Operations on the data cube are finished typically under one second. If the analyst wishes to look at the original log lines from the count cube cells, the indexing technology makes it possible to reach them under one second no matter how large the data set is.

Fig. 5. A typical screen image from the analysis module of the system: With color mapping, the heat map visualizes the frequencies of log patterns given in the data cube in hourly breakdown. The lighter (originally red) color indicates large number of occurrences, the darker (originally blue) indicates low occurrence rate or no occurrences. The sample represents information from logs collected about a week long period.

V. CONCLUSION

In the article we introduced a new log analysis solution based on multidimensional data cubes. We described the structure of the log analysis tool and explained the new technologies in modules attaining some steps in log management. According to that, we described in detail the real-time compression, the new log processing language able to represent professional knowledge for log normalization with the corresponding regular expressions, and the engine processing complex events described by multiple log lines. A new solution is the online analytic processing (OLAP) used for the analysis. Important features of the developed OLAP engine is the extremely fast response time for the queries, for regular OLAP cube operations, and also for data retrieval.

The presented log analysis tool offers effective monitoring technology for all companies using information technology extensively. Possible use cases cover multiple areas from traditional security monitoring, through real time exploration of operational risks, to business intelligence analysis of the monitored IT devices, helping to enhance efficiency.

Methods and algorithms developed during our research might be utilized in many areas outside log analysis. One of the promising alternative use cases is the processing of large amount of short text messages, or the analysis of extreme large data sets, which allows us to analyze the general behavior of many systems outside the field of informatics.

REFERENCES

[1] C. Lynch, “Big data: How do your data grow?” Nature, vol. 455, no. 7209, pp. 28-29, 2008.
[2] K. Cukier, ”Data, data everywhere”, The Economist, 25 February 2010. http://www.economist.com/specialreports/displaystory.cfm?story_id=15557443
[3] K. Scarfone and P. Hoffman, ”Guide to computer security log management: Recommendations of the National Institute of Standards and Technology”, 2006. http://purl.fdlp.gov/GPO/LPS115454
[4] http://code.google.com/p/lz4/
[5] http://www.oberhumer.com/opensource/lzo/
[6] http://www.lzop.org/
[7] http://zlib.net/pigz/
[8] S. Srinivasan, A. Amir, P. Deshpande, and V. Zbarsky, ”On business activity modeling using grammars”, 14th International Conference on World Wide Web (WWW '05), ACM, New York, pp. 1046-1047, 2005.
[9] K. Thompson, ”Regular expression search algorithm”, Communications of the ACM vol. 11, no. 6, pp. 419–422, 1968.
[10] http://flex.sourceforge.net/
[11] http://code.google.com/p/re2/
[12] J. de Guzman and D. Nuffer, ”The Spirit Library: Inline Parsing in C++”, C/C++ Users Journal, vol. 21, no. 9, p22, 2003. http://www.drdobbs.com/cpp/184401692
[13] D. Abrahams and A. Gurtovoy, C++ Template Metaprogramming - Consepts, Tools, and Techniques from Boost and Beyond, Addison Wesley Professional, 2004.
[14] http://boost-spirit.com/
[15] W. H. Burge, Recursive Programming Techniques, Addison-Wesley, 1975.
[16] S. Chaudhuri and U. Dayal, ”An overview of data warehousing and OLAP technology”, SIGMOD Rec. vol. 26, no. 1, pp. 65-74., 1997.
[17] O. R. Zaiane, M. Xin, and J. Han, ”Discovering Web access patterns and trends by applying OLAP and data mining technology on Web logs”, Proceedings of the Advances in Digital Libraries Conference, pp. 19-29, 1998.
[18] Q. Chen, U. Dayal, and M. Hsu, "An OLAP-based Scalable Web Access Analysis Engine", International conference on data warehousing and knowledge discovery (DaWaK), LNCS, vol. 1874, pp. 210-223, 2000.
[19] L. Y. S. Clement, ”Log Analysis as an OLAP Application - A Cube to Rule Them All”, SANS Institute, http://www.sans.org/reading_room/whitepapers/logging/log-analysis-olap-application-cube-rule_1152, 2003.
[20] http://www.mysql.com/
[21] http://www.iccube.com/
[22] I. Spiegler, R. Maayan,. "Storage and retrieval considerations of binary data bases". Information Processing and Management: an International Journal, vol. 21, no. 3, pp. 233–254, 1985.
[23] K. Wu, E. J. Otoo, and A. Shoshani, ”Optimizing bitmap indices with efficient compression”, ACM Transactions on Database Systems, vol. 31, pp. 1-38, 2006.
[24] E. O'Neil, P. O'Neil, and K. Wu, "Bitmap Index Design Choices and Their Performance Implications," 11th International Database Engineering and Applications Symposium (IDEAS 2007), pp. 72-84, 2007
[25] T. Niemi, J. Nummenmaa, and P. Thanisch, ”Constructing OLAP cubes based on queries”. 4th ACM international workshop on Data warehousing and OLAP (DOLAP '01). ACM, pp. 9-15, 2001.
(All internet sources are as of May 2012.)

András Lukács received his M. Sc degree at the Eötvös Loránd University in 1992, and and his Ph. D in mathematics at the Hungarian Academy of Sciences in 1998. From 1995 to 2011 he was a Research Fellow at the Computer and Automation Research Institute of the Hungarian Academy of Sciences, where he was a co-founder and Head of the Data Mining and Websearch Group. He has been involved in and coordinated several national, European and industrial data mining related projects. Since 2012 he has been a Senior Research Fellow at the Department of Computer Science, Eötvös Loránd University, Budapest, Hungary. Dr. Lukács current interests include data mining, networks and algorithms of big data.

Zsolt Nagy received his M. Sc and doctor degree in law from the Faculty of Law at the University of Debrecen in 2002, and a post gradual degree in 2011 from the Corvinus University of Budapest as jurist-economist. From 2003 he is a PhD student at the Faculty of Law and Political Science at the University of Pécs where he is doing research on legal informatics. From 2002 till 2008 he worked at NetLock Ltd as PKI consultant, from 2008 to present he is working at KÜRT Co, Budaörs, Hungary. Currently, he is the head of R&D at KÜRT Co. Dr. Nagy’s recent interests include electronic signature and time stamp in the e-business administration, security of critical infrastructures and log analysis.