Efficient Storage and Processing Of High Volume Network Monitoring Data

Ms. S. Malathy[1], Ms. D. Saranya[2], Ms. S. Naveena[3],Mr. D. Rajesh Kumar[4]
Asst. Professor[1,2,3,4],
Department of Information Technology,
Bannari Amman Institute of Technology, Sathyamangalam -638401, India.
E-mail: ksmalathy@bitsathy.ac.in[1], saranya.d@bitsathy.ac.in [2], naveena@bitsathy.ac.in[3] rajeshkumard@bitsathy.ac.in[4]

Abstract-Network traffic monitoring describes the use of the system that constantly monitors a computer network for a slow or failing component and notifies the network administrator I case of outages. Monitoring modern networks involves storing and transferring huge amount of data. To overcome with this problem a technique is used which allows number of operations performing directly on the transformed data with a controlled loss of accuracy. The results show that the transformed data closely approximates the original data(within 5% relative error) while achieving compression ratio of 20%.A sensibility analysis show that technique allows to trade off the accuracy on different input fields, while scalability analysis indicate that the technique scales with input size spanning up to three orders of magnitude. With the increasing sophistication of attacks, there is a need for network security monitoring systems that store and examine very large amounts of historical network flow data. An efficient storage infrastructure should provide both high insertion rates and fast data access. Traditional row-oriented Relational Database Management Systems (RDBMS) provide satisfactory query performance for network flow data collected only over a period of several hours.

Keywords-*Network monitoring, traffic analysis, monitoring data compression, space utilization, traffic analysis.*

*****

I INTRODUCTION

Network monitoring for a corporate network is a critical IT function that can save money in network performance, employee productivity and infrastructure cost overruns .A network monitoring system monitors an internal network for problems. Telecom Operators have to constantly monitor the network for a number of task like management, provisioning, service offering, etc.[4] Such as billing require the analysis of the data in real time. Network provisioning is performed on a set of statically indicator calculated over the last month and year. These task monitoring infrastructures have been presented such as relying on data reduction techniques to keep the amount of data manageable. Similarly these issues have been early addressed by the network research community.

Cloud storage is an important service of cloud networking, which allows the data owners to move the data from their networking systems to the cloud. The new paradigm of data hosting service also introduces new security challenges. Owners would worry that the data could be lost in the cloud. This is because data loss could happen in any infrastructure, no matter with high degree of reliable measures cloud service providers. It could discard the data that have not been accessed or rarely accessed to save the storage space and claim that the data are still correctly stored in the cloud. Therefore, owners need to be convinced that the data are correctly stored in the cloud. Owners can check the data integrity based on two-party storage auditing protocols. In this the third-party auditing is a natural choice for the storage auditing in cloud networking. A third-party auditor has expertise and capabilities can do a more efficient work and convince both cloud service providers and owners. The auditing protocol should have the following properties they are Confidentiality, Dynamic Auditing, and Batch Auditing. Some existing remote integrity checking methods can only serve for static archive data. It cannot be applied to the auditing service, since the data in the cloud can be dynamically updated. Thus, an efficient and secure dynamic auditing protocol is desired to convince data owners that the data are correctly stored in the cloud.

II LITERATURE SURVEY

SECURE STORAGE IN HIGH VOLUME NETWORK:

A spatially efficient data representation is produced that allows approximated computations in network monitoring logs, with no need for decompression stage. In order to ease the description of the technique, of its characteristics, and the challenges it must deal with, a specific format of monitoring log, but the technique can be applied to more complex monitoring data. The log format considered, analogous to flow-level traffic traces such as NetFlow ones, is constituted of records, each comprising four fields they are timestamp, source IP, destination URL, and load.

Each record represents a single HTTP session originated by a source IP to retrieve the given URL1, and the amount of data that has been exchanged. This format represents an example of data commonly logged by operators. To be able to perform computations directly on the representation,a technique is applied which converts the log in numerical matrices, where each row/column is a vector of real numbers. The technique comprises two main phases. The first one is called pre-processing phase and provides output as a fully numeric matrix representation of the original data. The second phase is called factorization phase which transforms the numeric matrix into a couple of sparse matrices which approximate the original data, while having a smaller memory footprint. The techniques are:

PRE-PROCESSING

Session splitting with a continuous stream of monitoring data, there is the necessity to split it in parts (chunks) with a finite number of sessions in order to apply the subsequent mapping and processing. This can be performed in several ways, either by considering the possibility to have overlapping between subsequent chunks or forcing all of them to be disjoint, and allowing for fixed or varying size chunks according to different stopping criteria. The chunk size and the splitting policy affect

i) The size of each compact representation chunk
ii) The data visible to the approximation algorithm, and thus both the compression efficiency and the approximation error
iii) The scope of the operations that can be performed directly on the representation can be applied intra-chunk and inter-chunks

SESSION SPLITTING

When dealing with a continuous stream of monitoring data, there is the necessity to split it in parts (chunks) with a finite number of sessions in order to apply the subsequent mapping and processing. The possibility to have overlapping between subsequent chunks or forcing all of them to be disjoint. Allowing for fixed- or varying-size chunks according to different stopping criteria. The data visible to the approximation algorithm, and thus both the compression efficiency and the approximation error. The scope of the operations that can be performed directly on the representation. The splitting can be performed before or after the URL filtering. , a fixed split size parameter is used to control the number of sessions allowed for each run of the approximation algorithm.

URL FILTERING

The proposed technique provides the possibility to control the presence of HTTP sessions with rare servers. It means of a URL filtering threshold parameter, defined as the minimum number of occurrences a URL must exhibit in the log in order to be considered. It also affects some sources.

LABEL CODING

The field URL (destination URLs) is stored as a list of unique elements. URLs are ranked by number of occurrences. It mapped to an integer equal to their rank order. a shorter code is assigned to more frequent items. The values in the field source IP are stored as a list with unique elements. The label corresponding to a source IP is given by its position. The load field is already numerical in nature. This way the size of mapping affects the total size of the representation within a percentage that decreases with the growth of the number of sessions.

NORMALIZATION AND WEIGHTING

This is done by dividing the code (ranking) by the total number of srcIDs. The specific application of the technique, different degrees of approximation can be tolerated on the different fields. A valuable property for this optimization is that, differently from the session split process.

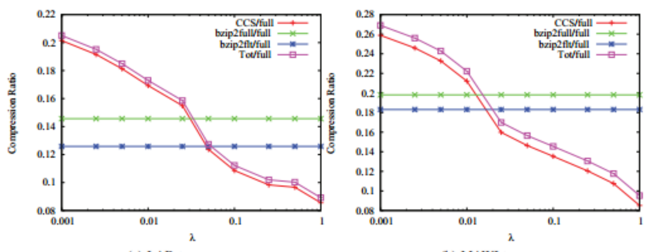
III OBSERVATIONS:

The total size of the data in the new representation format or compressed version as well as the size of specific components is compared against the size of the original data. The considered quantities are:

- CCS: size in bytes of the Compressed Column Sparse representation of the C matrix alone;
- Tot: sum of the size in bytes of: CCS, T matrix, bzip2-compressed ordered list of URLs, and bzip2-compressed ordered list of source IDs. These components, with a few metadata (namely the four integers representing the dimensions of the matrices and one float per field representing its scaling factor), are all needed to rebuild the original data (URL-filtered and approximated);
- bzip2flt: size in bytes of the URL-filtered and bzip2-compressed version of the original data;
- bzip2full: size in bytes of the bzip2-compressed version of the original data with no filtering;
- Full: size in bytes of the original data with no filtering. The CCS component is calculated in bits as: nnz·(base size +[log2(cols)])+[log2(nnz)]·(rows + 1) where nnz is the number of non-zero elements of the matrix, rows and cols are the dimensions of the matrix, and [.] is the ceiling function. This value corresponds to the size occupancy of a sparse matrix, represented as Compressed Column Sparse (or Compressed Sparse Column), where indexes are binary coded, and each element is represented with base size bits. In the considered case, base size is 32, rows and cols are the

dimensions of matrix C, respectively K and N in the notation of the previous section. The matrix T is represented as M ·K values of length base size bits each.

OBSERVATION RESULTS:



(a) LAB trace (b) MAWI trace

Figure 3.1 Compression ratios

IV CONCLUSION

Current work is also focused on evaluating the trade-off in accuracy vs. compression in relation to different precisions of matrix element representation, and on evaluating the accuracy obtainable after the data-mining operations, in relation to different use cases. Finally, we extend and apply the proposed technique to monitoring data from traffic analysis and characterization [5] and quality of service parameters [6].

V. REFERENCES

[1] P. B. Ros, G. Iannaccone, J. S. Cuxart, D. A. Lo´pez, and J. S. Pareta, "Load shedding in network monitoring applications," in Proc. 2007 USENIX Annual Technical Conference

[2] A. Dainotti, A. Pescap`e, and G. Ventre, "A packet-level characterization of network traffic," in *Proc. 2006 CAMAD*, pp. 38–45.

[3] R. Karrer, I. Matyasovszki, A. Botta, and A. Pescap`e, "Experimental evaluation and characterization of the *magnets* wireless backbone," in *Proc. 2006 WINTECH*, pp. 26–33.

[4] M. Munk, J. Kapusta, and P. Svec, "Data preprocessing evaluation for web log mining: reconstruction of activities of a web visitor," Procedia Computer Science, vol. 1, no. 1, pp. 2273–2280, 2010.

[5] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes, "Cisco systems netflow services export version 9," RFC 3954, Oct. 2004, Tech. Rep.

[6] J. Zujovic and O. G. Guleryuz, "Complexity regularized pattern matching," in Proc. 2009 IEEE International Conference on Image Processing.