MASC: a Bitmap Index Encoding Algorithm for Fast Data Retrieval

Han Wang*, Zhen Chen†, Yuhao Wen‡, Junwei Cao, Guodong Peng, Wen-Liang Huang§, Ziwei Hu‖, Jing Zhou and Jinghong Guo
Research Institute of Information Technology, Tsinghua University, Beijing, China
Department of Electronic Engineering, Tsinghua University, Beijing, China
Department of Automation, Tsinghua University, Beijing, China
Email: *han-wang@outlook.com, †zhenchen.nslab@gmail.com, ‡wenyhinthu@gmail.com
§China Unicom Groups Labs
Financial Street, Xicheng District, Beijing, China, 100140
Email: huangwenliang@gmail.com
‖State Grid Smart Grid Research Institute, Beijing, China, 102209

***Abstract*—Fast retrieval in archival traffic data is essential for network security and forensic analysis. Bitmap Index is a data structure enabling fast search over large data collections in a limited time, but the space consumption is always a problem. WAH, PLWAH and COMPAX are proposed for compressing bitmap indexes for less storage. In this paper, a new bitmap index encoding scheme, named MASC, is proposed to further improve the compression ratio without impairing the query performance. Instead of being limited to a fixed length (31 bits) in PLWAH and COMPAX, the stride size can be as long as possible to encode consecutive zero bits and nonzero bits in a more compact way. Instead of piggyback used in PLWAH, a new structure in MASC called carrier is introduced as piggyback in PLWAH only carries an individual nonzero bit. We also generalize the traditional literal word concept in PLWAH and COMPAX. The validity of MASC encoding scheme is demonstrated with the application in Internet Traffic Archival system. Based on experiments with real Internet traffic data set from CAIDA, MASC has a better compression ratio than PLWAH and COMPAX2 without the penalty in query performance.**

***Index Terms*—traffic archival; network forensic; network security; bitmap index encoding; bitmap index compression; PLWAH; COMPAX**

I. Introduction

Indexes enable fast search over large data collections in a limited time. But traditional indexes have huge space consumption, which spurs the research on index compression for space and performance issues. Index compression is both interested in theory and practice. Gonzalo Navarro [1] et al. present a comprehensive survey on this research topic. Falk Scholer [2] et al. find that index compression not only saves the space cost, but also accelerates the query speed. Vo Ngoc Anh and Alistair Moffat [3] et al. propose a word-aligned binary codes for inverted index compression for full-text search engine. Jeff Dean [4] introduces the Block-Based Index Format, Byte-Aligned Variable-length Encodings and then Group Variant Encoding schemes used in several generations of Google search engine for space optimization and performance improvement.

A bitmap index is a structure that can accelerate search queries on low-cardinality attributes. It is useful in scientific data and traffic archival. But the space consumption is always a serious problem. Ming-Chuan Wu [5] et al. propose the bitmap index encoding method and its usage in data warehouse. Kesheng Wu [6] et al. propose WAH (Word-Aligned Hybrid) compression scheme for bitmap indexes and give a practical implementation called Fastbit [7]. As the state-of-art, PLWAH (Position List Word-Aligned Hybrid) [8] and COMPAX (COMPressed Adaptive indeX) [9] compression scheme are proposed for bitmap indexes to make further improvement for the WAH scheme. COMPAX2 is the new version of COMPAX scheme which provides an extended codebook and has better compression ratio with similar query speed compared with COMPAX.

In this paper, a new bitmap index encoding scheme, named MASC (**MA**ximized **S**tride with **C**arrier), is proposed to further improve the compression ratio without impairing the query performance. MASC uses the stride size as long as possible, not limited to 31 bits in PLWAH and COMPAX, to encode the consecutive zero bits and nonzero bits in a more compact way. MASC records origin bitmap index sequences into a new designed format. We demonstrate the validity of MASC with the application in Internet Traffic Archival system. Based on our experiments using real Internet traffic data set from CAIDA, MASC has better compression ratio than PLWAH and COMPAX2 for about 10%.

This paper is organized as follows: Section 2 introduces the background of bitmap index encoding scheme. Section 3 describes details in the design and encoding procedure of MASC. Section 4 presents the applications of the proposed MASC in Internet traffic Archival system. The experiment result is also presented with real Internet traffic trace from CAIDA in Section 4. Finally, we conclude this paper with future work in Section 5.

II. Bitmap Index

A bitmap index is a structure that can accelerate the process of searching queries. Its format is shown as follows in Table I.

TABLE I
AN EXAMPLE OF BITMAP INDEX

Row ID	Column Number	Bitmap Index			
		=1	=2	=3	=4
1	4	0	0	0	1
2	3	0	0	1	0
3	2	0	1	0	0
4	3	0	0	1	0
5	4	0	0	0	1
6	1	1	0	0	0

The shortcoming of bitmap index is that it requires large storage space, which has plenty of room to be improved. A considerable amount of bitmap index encoding algorithms have been raised and several of them are widely-used.

WAH is proposed by K. Wu, E. J. Otoo, and A. Shoshani [6]. WAH introduces the method of dividing 31-bit chunks into fill chunks (all of 31 bits are 0) and literal chunks (the rest), then encodes all fill chunks into a single fill word. WAH performs well when there are huge amount of consecutive 0's in the origin bit sequence.

F. Deli'ege and T. B. Pedersen proposed PLWAH [8]. While WAH finishes encoding after combining fill chunks, PLWAH tries to encode the fill word and its next literal word together if the literal word is nearly-identical to a 0-fill word. As a result, PLWAH has better compression ratio than WAH in general.

Unlike PLWAH, F. Fusco, M. Stoecklin and M. Vlachos propose COMPAX [9], which improves WAH in a different way. COMPAX also tries to combine literal words and fill words after dividing them. However, COMPAX introduces a codebook which enables the algorithm to encode original bit sequence in more paths. Actually, COMPAX divides bit sequence into F (fill), L (literal), FLF (fill-literal-fill), and LFL (literal-fill-literal) types and performs quite well compared with WAH and PLWAH.

III. MASC BITMAP INDEX ENCODING ALGORITHM

A. MASC Encoding Scheme

We propose a bitmap index encoding algorithm which can reduce the bitmap index size with the comparable query performance with the state-of-art algorithms, i.e., COMPAX2 and PLWAH. For clarity, a chunk is a fixed size block in 0-1 sequences for encoding operation. For the comparison purpose, we set chunk to 31 bits as well as indicated in COMPAX2 and PLWAH, while the bit order is from MSB to LSB.

In essential, MASC makes an encoding stride as long as possible and use a concept called *carrier* instead of *piggyback* in PLWAH. The design details of MASC are introduced as follows.

1) A 0-fill word encodes a sequence of consecutive zero bits. For example, the 0-fill word in Fig. 1 encodes 6 chunks (6×31) and 5 consecutive zero bits (191 bits in total).

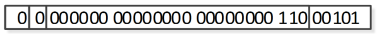

Fig. 1. 0-fill word.

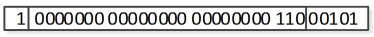

Fig. 2. 1-fill word.

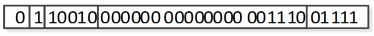

Fig. 3. 1-carried 0-fill word.

2) A 1-fill word encodes a sequence of consecutive nonzero bits. For example, the 1-fill word in Fig. 2 encodes 6 chunks (6×31) and 5 consecutive nonzero bits (191 bits in total).
3) A 1-carried 0-fill word encodes a sequence of consecutive zero bits followed by consecutive nonzero bits (at most 30 bits, 1 bit less than a complete chunk).

In Fig. 3, the 2^{nd} bit is used as *carrier flag*. When the flag is set to 1, it means that the word is a 1-carried 0-fill word. The $3^{rd} - 7^{th}$ bit is *carrier*, counting the amount of carried consecutive nonzero bits. The $8^{th} - 27^{th}$ bits are used as a counter, counting chunks of consecutive zero bits. The $28^{th} - 32^{nd}$ bit is an additional counter, counting consecutive zero bits that less than a chunk. In Fig. 3, the entire word represents $14 \times 31 + 15$ consecutive zero bits followed by 18 consecutive nonzero bits.

Actually a 1-carried 0-fill word also belongs to 0-fill word, however a **pure** 0-fill word does not carry any nonzero bits with its carrier flag set to 0, while 1-carried 0-fill word carries up to 30 nonzero bits with its carrier flag set to 1. So in following sections, if not special specified, when we mention a 0-fill word, it means the word does not have a carrier.

B. MASC Encoding Procedure

The encoding steps in MASC are explained in detail as follows.

1) Uncompressed bitmap index is divided into equal chunks of 31 bits. There is an example shown in Fig. 4.

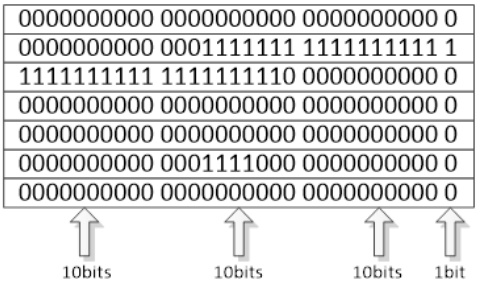

Fig. 4. Uncompressed bitmap index.

2) From the very first chunk, we count consecutive zero bits or nonzero bits and encode them into MASC form.

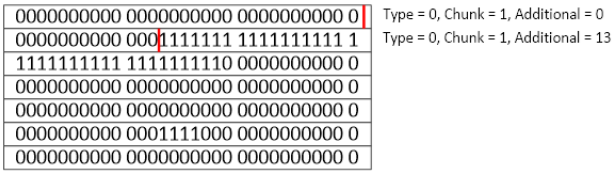

Fig. 5. Dealing with the first 0-fill word.

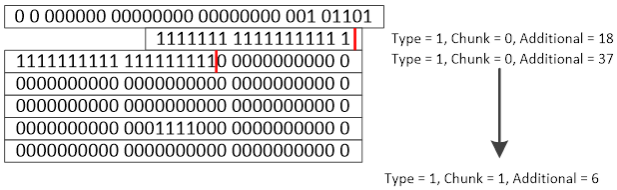

Fig. 6. Dealing with the first 1-fill word.

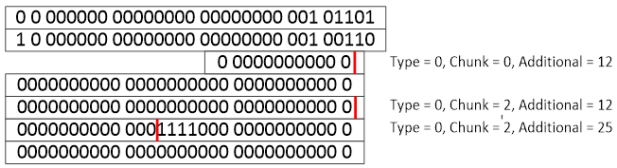

Fig. 7. Dealing with the second 0-fill word (step 1).

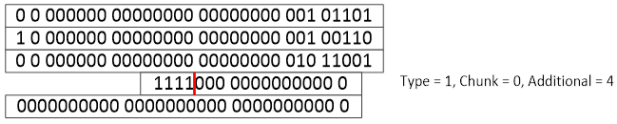

Fig. 8. Dealing with the second 0-fill word (step 2).

Fig. 5 to Fig. 9 shows detailed procedure based on the example presented in Fig. 4.

In Fig. 5, all bits in the first chunk are zero bits, and the first 13 bits in the second chunk are zero bits too. So we encode them into a 0-fill word. However, we have to check the next word to determine whether to combine these two words into a 1-carried 0-fill word or not. Then we come to Fig. 6.

In Fig. 6, the last 18 bits in the second chunk and the first 19 bits in the third chunk are nonzero bits. The total number is 37, exceeding 30 which is the limit of 1-carried 0-fill word. So we encode them into a single 1-fill word, and the former 0-fill word would not have any carrier.

The last 12 bits in the third chunk, the whole 4^{th} and 5^{th} chunk, and the first 13 bits in the 6^{th} chunk are zero bits. As a result, we encode them into a single 0-fill word. We should still check if we have to combine these zero bits and following consecutive nonzero bits into 1-carried 0-fill word.

The 13^{th} to 16^{th} bit in the 6^{th} chunk are nonzero bits, no more than the carrier limit. So we encode the four consecutive nonzero bits and the former zero bits into a 1-carried 0-fill word.

The last 14 bits in the 6^{th} chunk and the whole 7^{th} chunk are zero bits. So we encode them into a 0-fill word without

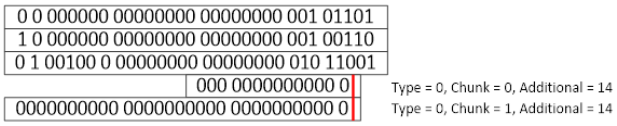

Fig. 9. Dealing with the last 0-fill word.

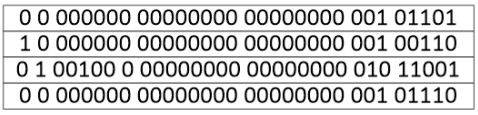

Fig. 10. The encoding result using MASC.

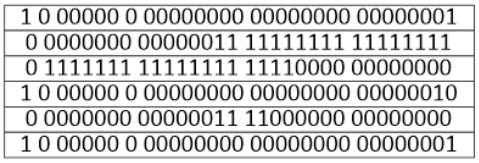

Fig. 11. The encoding result using PLWAH.

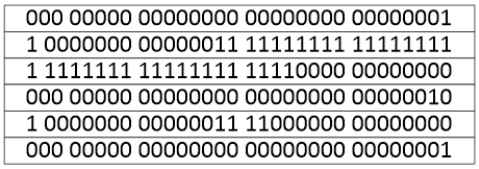

Fig. 12. The encoding result using COMPAX.

carrier. Till now we have finished the encoding process of MASC and Fig. 10 is the bit sequence after encoding the origin bitmap index by MASC.

Final results of PLWAH and COMPAX2 are shown in Fig. 11 and Fig. 12 respectively. It is clearly observed that both PLWAH and COMPAX2 encode the original bit sequence into 6 words, while MASC's result consumes only 4 words in all.

C. Comparison among MASC, PLWAH and COMPAX2

The concept *carrier* in MASC and *piggyback* in PLWAH are similar. However, a *carrier* can carry at most 30 nonzero bits while PLWAH can *piggyback* only a single nonzero bit. Besides, we generalize the concept of literal word and eventually obsolete this concept. As a consequence, several (no more than 30) nonzero bits can be carried by the former 0-fill word and output a 1-carried 0-fill word, while PLWAH has to encode them in a literal word or two literal words in the worst condition when the consecutive nonzero bits locate in two adjacent chunks. Considering zero bits' and nonzero bits' distribution in real data set, zero bits and nonzero bits appear usually in batch especially after being sorted by the hash value of each record. Thus MASC can perform better than PLWAH.

Though both MASC and COMPAX2 use the concept of filled words, there are many differences between them. In Section 4, we will conduct experiments to show MASC's advantages over the other two encoding schemes.

D. Query Table

Inspired by Group Variant Encoding scheme used in Google [4], we make some modification on plain MASC algorithm to raise the query efficiency. An extra bit, i.e. the second bit is used as a flag too. Similar to the 256-entry table in Jeff Dean's talk [4], we introduce the concept of query table which contains *type tag* and *position offset* in an encoding window. The encoding window size is typically set to 4k in our experiment carried out in Section 4.

The second bit of 1-fill words is set to 1. Fig. 13 shows the new 1-fill word.

1	1000000 00000000 00000000 110	00101

Fig. 13. A 1-fill word after revision.

The encoded bit sequence shown in Fig. 10 is converted into those in Fig. 14 as the original bit sequence shown in Fig. 4. The italics bit shows the difference between two results.

0 0 000000 00000000 00000000 001 01101
1 *1* 000000 00000000 00000000 001 00110
0 1 00100 0 00000000 00000000 010 11001
0 0 000000 00000000 00000000 001 01110

Fig. 14. The new encoding result from original bit sequence in Fig. 4.

Query table is used during query process. It consists of two parts: *type tag* and *position offset*.

A *type tag* shows whether the word contains nonzero bits. The second bit in a 0-fill word is set to 0 and all of the rest's second bit is 1, as a consequence the type tag can be directly duplicated from the second bit of words.

The *position offset* consists of chunk offset and bit offset. Chunk offset shows the number of chunks from the first one of current encoding window, while bit offset is determined by the number of bits from the first one of current chunk. The number of bits needed to represent the bit offset is 5. While bits needed for chunk offset is flexible, we set it to 7 in this example. Fig. 15 shows the whole query table for the example in Fig. 14.

0	0000000	00000
1	0000001	01101
1	0000010	10011
0	0000101	10001

Fig. 15. Query table for the example in Fig. 4.

Consider a computer can carry out a bitwise sum in a seconds and carry out a logical sum in b seconds. Let total words after encoding procedure to be W. Before introducing query table, assume that the average time for querying for a particular record is A and the average time querying for all records hit by a particular prerequisite (e.g. querying for all packets satisfying the first byte of source IP in traffic is 166) is B. Then we have $A = 0.75Wa+Wb$, and $B = 1.5Wa+2Wb$. If we compress the whole records together, using a query table, the time for querying for a particular record remains unchanged, but the time for querying for all records belong to a particular prerequisite (such as the same query above) declines to $Wa + Wb$.

IV. Applications

A. Traffic Forensic with MASC

Cisco [12] predicts that the volume of Internet traffic will quadruple between 2011 and 2016 reaching 1.3 Zettabytes per year in 2016. According to the internal statistics of China Unicom [20], mobile user traffic increases rapidly with CAGR (Compound Annual Growth Rate) of 135%. From the data of China Unicom in 2013, its monthly records are more than 2 trillion (2×10^{12}), monthly data volume is over 525TB, and has reached 5PB.

Luca Deri and Francesco Fusco [13] [14] propose MicroCloud-based flow aggregation for fixed and mobile networks. This architecture is used to provide real-time traffic monitoring and correlation in large distributed environments.

P. Giura and N. Memon [15] propose NetStore, a column oriented storage with IP address based on inverted indexes for fast retrieval. Each of the segments within a column in NetStore is compressed independently. They also discuss different possible compression methods. As Netstore maintains the strict time order of packets, it does not consider the reordering of packet based on flow level and utilizing bitmap indexing compression in its system.

CNSMS (collaborative Network security management system) [16], vCNSMS (Virtualized Collaborative Network Security Management System) [21] and TIFAflow (Time machine + FAstbit) [17] are used for traffic acquisition and aggregation for forensic analysis. CNSMS is an architecture for traffic acquisition with TIFAflow and its UTM (Unified Threat Management) appliance for traffic aggregation used in forensic analysis in a cloud computing based security center. TIFAflow is a software based on probe that combines TIFA [18] with fastbit indexing to provide granular data storage. It may be operated as an independent prober or integrated into CNSMSs UTM appliance.

A 10 Gbps network link can arrive at a maximum of 14.8 million packets per second. It is a big challenge to index these packets in one second. For any mobile network operator manages several such links, even records only flow statistics, the volume of resulting data could easily reach Terabytes in one year. If all mobile traffic data is recorded for forensic analysis, the volume of the data could easily reach Petabytes. That remains a major challenge to a mobile network operator that it must accommodate and index such big data for further analysis.

We apply the proposed MASC encoding scheme to record the origin bitmap index sequences into a new format. In the following sections, we evaluate the performance of three bitmap index compression encodings using the real data trace from CAIDA. Our experiment results suggest that MASC has the best performance among all of the three encoding schemes.

B. Performance Evaluation with Real Data

We choose the real Internet trace data from CAIDA [10] to evaluate three bitmap indexes encoding schemes.

TABLE II
CAIDA DATASET ATTRIBUTES

Column	Type	Bytes
Source IP	int	4
Destination IP	int	4
Source Port	short	2
Destination Port	short	2
Protocol	byte	1

We make comparison among MASC and the state-of-art algorithms, i.e., PLWAH and COMPAX2. There are totally 13,578,496 IPv4 packets in the CAIDA's data set. We pick out source IPs in our experiment for bitmap indexing.

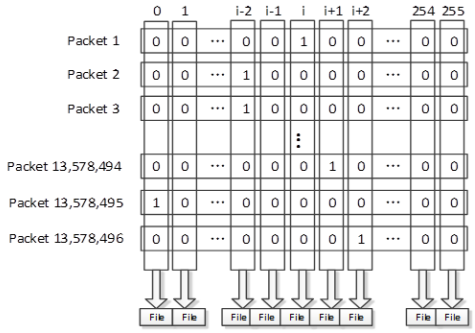

Fig. 16. Bitmap indexes of a byte from Source IP in Real Internet Trace.

At first we calculate hash values of each packet's five tuples and reord packets by hash values. This procedure is similar to oLSH function in COMPAX for better compression ratio. The difference is the hashing method we used is the uniform hashing instead of locality hashing.

After that, packets belonging to the same flow can be aggregated. Then we pick out each byte in source IP address (4 bytes in all) of each packet, converting each byte into bitmap index and writing the bitmap index into files for further encoding. After that, each file corresponds to a column of bitmap index as shown in Fig. 16. Then we encode 256×4 bitmap files by three encoding algorithms, i.e., PLWAH, COMPAX2 and MASC respectively, to evaluate the encoding schemes and make comparison among them.

From Fig. 17 to Fig. 19, it is clearly shown that MASC has a better compression ratio. MASC's disk consumption is 18.07% less than PLWAH's and 16.59% less than COMPAX2.

Fig. 20 shows the relationship between nonzero bits, which is equal to the number of matching results, and room consumption. The figure shows that COMPAX2 and PLWAH have their own comfort zone. PLWAH has better performance while the number of matching results is smaller and COMPAX2 wins when it becomes bigger. Their curves intersect at a point and this characteristic is also illustrated by Fig. 12 in F. Fusco's paper [11]. However, MASC consumes the least space among the three methods under almost all circumstances. From this figure, we can also observe that MASCs curve is always below the other two curves, which proves that MASC has the best compression ratio compared with COMPAX2 and PLWAH.

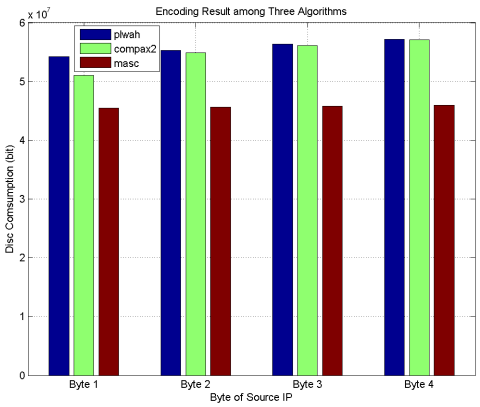

Fig. 17. The size of Compressed Source IP using 3 different schemes.

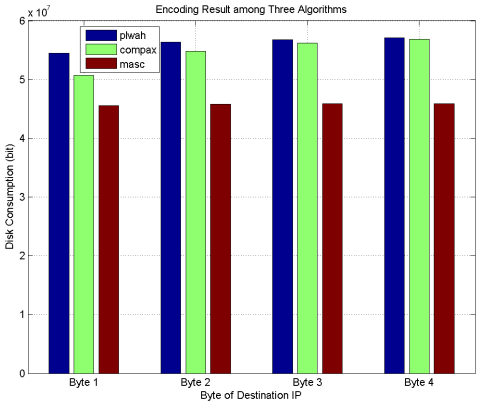

Fig. 18. The size of Compressed Destination IP using 3 different schemes.

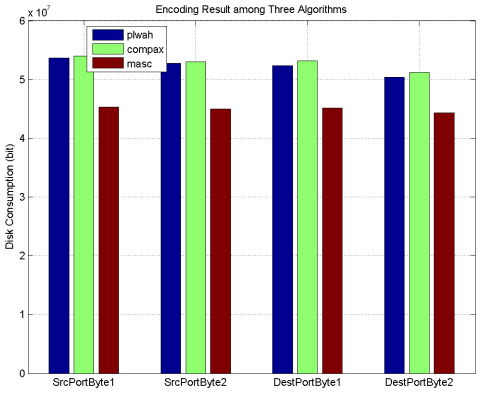

Fig. 19. The size of Compressed Ports using 3 different schemes.

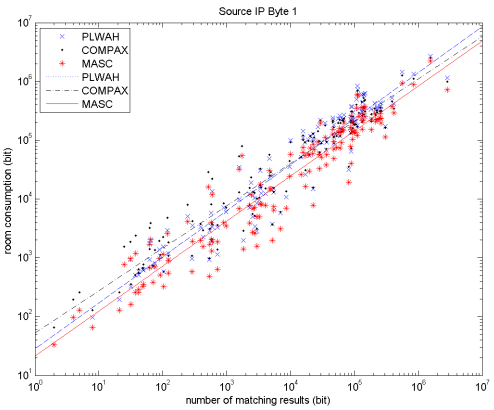

Fig. 20. Relationship between number of matching results and room consumption in byte 1 of Source IP.

C. The Benefit of Query Table for Query Performance

To pick out all records belonging to a particular prerequisite (such as source IP = 166.*.*.*), the plain MASC has to make more than 2.15M bitwise sum operations and 1.43M logical sum operations on average. After complementing MASC with query table, now we can make 1.43M bitwise sum operations and 700K logical sum operations. That is because the type tag lessen the workload of bitwise sum and the position offset reduces the number of logical sum operations, since position offset frees MASC from calculating positions once and again.

Meanwhile, PLWAH has to carry out more than 1.74M bitwise sum and 870K logical sum operations, and COMPAX2 makes 2.05M bitwise sum and 850K logical sum operations. As a result, MASC with query table has comparable querying efficiency as PLWAH and COMPAX2.

V. Conclusion

In this paper, we propose a new bitmap index encoding algorithm named MASC. Instead of 31 bits in PLWAH and COMPAX, MASC uses the stride size as long as possible to encode the consecutive zero bits and nonzero bits in a more compact way. In essential, MASC also uses a concept called *carrier* instead of *piggyback* in PLWAH as *piggyback* only represents the individual nonzero bit. We also generalize the traditional literal word concept in PLWAH and COMPAX by the new designed encoding format. As inspired by index format from Google, we also introduce the query table so that the query process can be far better than the plain MASC. In the experiments based on real Internet data trace, MASC shows a better compression ratio than PLWAH and COMPAX schemes without impairing the query performance in practice.

In the future, we will carry out more experiments on the compression ratio and query time to make all-round evaluation on the new algorithm.

Acknowledgment

This work is supported in part by Ministry of Science and Technology of China under National 973 Basic Research Program (No. 2013CB228206 and No.2012CB315801), National Natural Science Foundation of China (grant No. 61233016), and China NSFC A3 Program (No.61140320).

References

[1] G. Navarro and V. Mkinen, *Compressed full-text indexes*. ACM Computing Surveys (CSUR) 39, No. 1 (2007): 2.

[2] F. Scholer, H. E. Williams, J. Yiannis and J. Zobel, *Compression of inverted indexes for fast query evaluation*. In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 222-229. ACM, 2002.

[3] V. N. Anh and A. Moffat, *Inverted index compression using word-aligned binary codes*. Information Retrieval 8, No. 1 (2005): 151-166.

[4] J. Dean. *Challenges in building large-scale information retrieval systems: invited talk*. In Proceedings of the Second ACM International Conference on Web Search and Data Mining, pp. 1-1. ACM, 2009.

[5] M. Wu, A. P. Buchmann, and P. Larson, *Encoded Bitmap Indexes and Their Use for Data Warehouse Optimization*. Shaker, 2001.

[6] K. Wu, E. J. Otoo, and Arie Shoshani, *Optimizing bitmap indices with efficient compression*. ACM Transactions on Database Systems (TODS) 31, no. 1 (2006): 1-38.

[7] *FastBit*. An Efficient Compressed Bitmap Index Technology. https://sdm.lbl.gov/fastbit/.

[8] F. Deli'ege and T. B. Pedersen, *Position list word aligned hybrid: optimizing space and performance for compressed bitmaps*. Proc. of the 13th Int. Conf. on Extending Database Technology, EDBT '10, 2010.

[9] F. Fusco, M. Stoecklin, and M. Vlachos, *NET-FLi: On-the-fly Compression, Archiving and Indexing of Streaming Network Traffic*. Proceedings of the International Conference on Very Large DataBases (VLDB), pp. 13821393, 2010.

[10] CAIDA, http://www.caida.org.

[11] F. Fusco, M. Vlachos, and M. Stoecklin, *Real-time creation of bitmap indexes on streaming network data*, The VLDB Journal-The International Journal on Very Large Data Bases 21, no. 3 (2012): 287-307.

[12] *Cisco Visual Networking Index Forecast (2011-2016)*. http://www.cisco.com/web/solutions/sp/vni/vni_forecast_highlights.

[13] L. Deri, and F. Fusco, *MicroCloud-based Network Traffic Monitoring*, IFIP/IEEE International Symposium on Integrated Network Management (IM), 2013.

[14] L. Deri, and F. Fusco, *Realtime MicroCloud-based flow aggregation for fixed and mobile networks*, 9th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC), pp.96-101, 2013.

[15] P. Giura and N. Memon, *NetStore: an efficient storage infrastructure for network forensics and monitoring*. In Recent Advances in Intrusion Detection, pp. 277-296. Springer Berlin Heidelberg, 2010.

[16] Z. Chen et al., *Cloud Computing-Based Forensic Analysis for Collaborative Network Security Management System*, Tsinghua Science and Technology, pp. 40-50, vol.18, No. 1, 2013.

[17] Z. Chen et al., *TIFAflow: enhancing traffic archiving system with flow granularity for forensic analysis in network security*, Tsinghua Science and Technology, vol. 18, No. 4, 2013.

[18] G. Maier, R. Sommer, H. Dreger, A. Feldmann, V. Paxson, and F. Schneider, *Enriching Network Security Analysis with Time Travel*, Proceedings of the ACM SIGCOMM 2008 conference on Data communication, New York, USA, pp. 183-194.

[19] J. Li et al., *TIFA: Enabling Real-Time Querying and Storage of Massive Stream Data*, Proc. of International Conference on Networking and Distributed Computing (ICNDC), 2011.

[20] W. Huang, Z. Chen, W. Dong, and H. Li, *Mobile Internet big data platform in China Unicom*, Tsinghua Science and Technology, Volume 19, Issue 1, pp. 95-101, Feb. 2014.

[21] Z Chen, W Dong, H Li, P Zhang, X Chen, and J Cao, *Collaborative network security in multi-tenant data center for cloud computing*, Tsinghua Science and Technology, 19 (1), pp. 82-94, Feb. 2014.

[22] J. I. Cohen, *Bitmap index compression*, Oracle Corporation, http://www.google.com/patents/US6205442.

[23] BBC, Byte aligned data compression, DEC/Oracle, http://www.google.com/patents/US5363098.

[24] WAH, Word aligned bitmap compression method, data structure, and apparatus, UC Berkeley, http://www.google.com/patents/US6831575.

[25] COMPAX, Network analysis, IBM, http://www.google.com/patents/US20120054160.