PLWAH+: A Bitmap Index Compressing Scheme based on PLWAH

Jiahui Chang, Zhen Chen*,
Wenxun Zheng, Yuhao Wen, Junwei Cao
Research Institute of Information Technology,
Tsinghua University
Tsinghua National Lab for Information Science and Technologies (TNList), Beijing, China
zhenchen@tsinghua.edu.cn
changjh13@mails.tsinghua.edu.cn

Wen-Liang Huang
China Unicom Groups Labs
China Unicom Groups
Beijing, China
wlhuang@chinaunicom.cn

ABSTRACT
Archiving of the Internet traffic is essential for analyzing network events in the field of network security and network forensics. The bitmap index is widely used to achieve fast searching in archival traffic data requiring a large storage space. As current state-of-art, WAH, PLWAH and COMPAX are proposed for compressing bitmap indexes. In this paper, a new bitmap index compression scheme, named PLWAH+ (Position List Word Aligned Hybrid Plus), is introduced, based on PLWAH. With less storage consumption, PLWAH+ is more suitable for indexing in large-scale and high-speed network traffic.

Categories and Subject Descriptors
E.4 [**Coding and Information Theory**]: Data compaction and compression*;* C.2.3 [**Network Operations**]: Network monitoring.

Keywords
Internet traffic, bitmap coding, bitmap index, data retrieval, traffic analysis, traffic forensic.

1. INTRODUCTION
Indexing is the core technology underlying answering queries on a large-scale archival data. Bitmap index is designed for quick retrieval of archival Internet traffic data. A bitmap index example is shown in *Table 1*.

Bitmap indexing uses a bit vector to indicate the certain values of the index, which was firstly proposed by P. O'Neil in the design of Model 204 commercial database. [6-7]

Table 1 An example of the Bitmap index

RowID	column	Bitmap			
		=1	=2	=3	=4
1	1	1	0	0	0
2	2	0	1	0	0
3	4	0	0	0	1
4	3	0	0	1	0
5	2	0	1	0	0
6	4	0	0	0	1

ANCS'14, October 20–21, 2014, Los Angeles, CA, USA
ACM 978-1-4503-2839-5/14/10.
http://dx.doi.org/10.1145/2658260.2661777

The technologies used in Bitmap index database includes bitmap indexing [4], bitmap compression and classification. Currently, the state-of-art bitmap index compression algorithms are BBC [1], WAH [3], PLWAH [2], COMPAX [8], SECOPAX [9] etc.

Based on the observation of the result of the WAH performance, this paper improves upon existing work by offering a lossless bitmap compression technique that outperforms PLWAH on both storage and performance perspective. Especially, we propose the PLWAH+ (Position List Word Aligned Hybrid Plus) bitmap compression scheme.

2. PLWAH+ CODING SCHEME
2.1 Definitions for Chunks
In PLWAH+ compression scheme, a bit vector is divided into chunks of 31 bits to ensure they are fit into the L1 cache. At first, we will classify each chunk into different types. Types for a chunk are defined as below:

0-Filled Chunk: If the 31 bits of a chunk are all '0', we call the chunk 0-Filled Chunk.

1-Filled Chunk: If the 31 bits of a chunk are all '1', we call the chunk 1-Filled Chunk.

Literal Chunk: If a chunk cannot be classified into 0-Filled Chunk or 1-Filled Chunk, it is called a Literal Chunk.

Dirty Bit: If only a few bits in a Literal Chunk are different from Filled Chunk, they are all called Dirty Bit. Furthermore, they can be divided into 1-Dirty Bit (1 bit) and 0-Dirty Bit (0 bit).

NI Chunk: If a Chunk is a Literal Chunk with less than 4 Dirty Bit, it is called a NI Chunk. The NI Chunk can be divided into two parts as follows:

NI-0 Chunk: If a Chunk is nearly identical to the '0' sequence with less than 4 1-Dirty Bits, it is called a NI-0 Chunk.

NI-1 Chunk: If a Chunk is nearly identical to the '1' sequence with less than 4 0-Dirty Bits, it is called a NI-1 Chunk.

2.2 Definitions for Codewords
After the categorization of the chunks, we begin to encode the bitmap roughly into the codewords as shown below:

0-Fill: If there are some continuous 0-Filled Chunks, replace them with a 0-Fill codeword which indicates the number of the replaced chunks.

1-Fill: If there are some continuous 1-Filled chunks, replace them with a 1-Fill codeword which indicates the number of the replaced chunks.

Obviously, 0-Fill and 1-Fill are two types of Fill.

Last but not least, we do the ultimate encoding based on rough encoding as shown below:

LF: For a continuous 2-tuple in the sequence, if the first element is a NI Chunk and the second codeword is a Fill, this 2-tuple is encoded into a LF codeword, including NI-0-0-Fill, NI-1-0-Fill, NI-0-1-Fill, and NI-1-1-Fill.

FL: For a continuous-2-tuple in the sequence, if the first element is a Fill and the second codeword is a NI Chunk, this 2-tuple is encoded into a FL codeword, including 0-Fill-NI-0, 0-Fill-NI-1, 1-Fill-NI-0, and 1-Fill-NI-1.

Literal: If a Literal Chunk survives after the encoding procedure with FL and LF, it's called a Literal codeword.

So far, the whole process of PLWAH+ compression has finished. The result is composed of Fill, Literal, LF, and FL codewords.

2.3 Bit-Represented CodeBook

In this section, the final result of every codeword is represented by 4 bytes.

As for Literal, we add a '0' before the 31 bits as a flag for identifying.

And it is easy to find that a Fill word has 23 bits for storing a counter. However, only the lowest byte will be used later in the experiment.

In the FL and LF codewords, the first five bits which represent first dirty bit position can't be zero while, as is shown in *Figure 1*, the second, third and fourth are flexible. So the number of the dirty bits in the NI Chunk is no more than four. The counter of the FL and LF words can be represented within 8 bits, corresponding to the segment of 3,968 (128*31) bit sequence.

The details are shown in *Figure 1*:

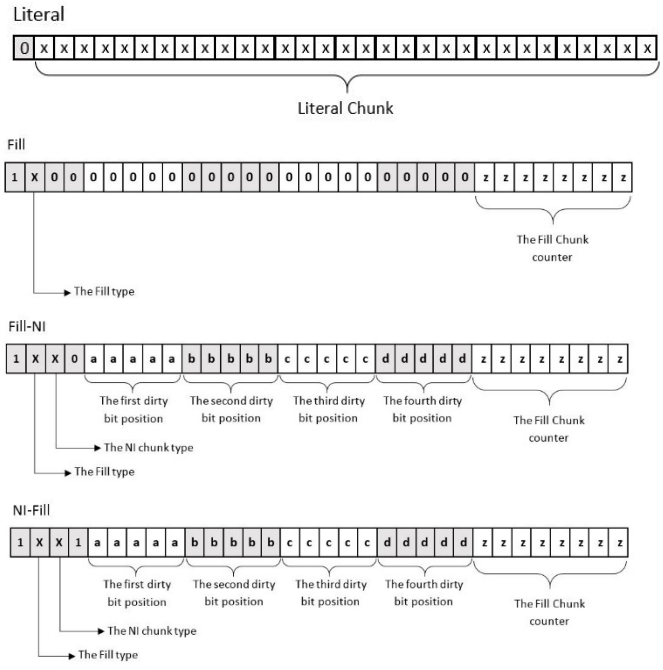

Figure 1 Codebook

3. EXPERIMENT AND RESULTS

In our experiments, the network flow data from CAIDA [5] is parsed using *libpcap* library, and the fields named source IP, source port, destination IP, destination port and protocol ID are extracted from the *pcap* archive, all of which are saved into a plain text file. The row ID of that file is the same as the record ID.

The original data size is 13,581,810 multiplied by 14 bytes, equaling to 47,536,335 Dwords (64 bits). The final compressed files sum up to 20,516,573 Dwords. As shown in *Figure 2*, we can see that the result of PLWAH+ reduces about 3% of the storage comparing to PLWAH and the compression ratio reaches roughly 43% with about 20% reduction of the amount of the literal words.

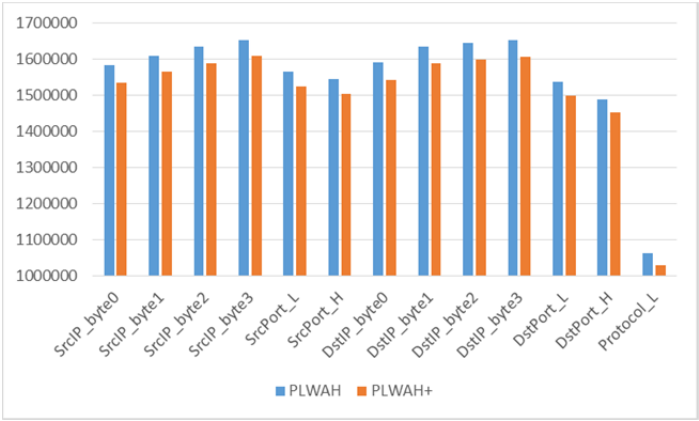

Figure 2 Used storage in Dword (64bits)

4. CONCLUSION

In this paper, we mainly discuss the PLWAH+ which is typical and more suitable for the modern CPU architecture. The concept of the NI-1 chunk and the LF word which are not considered in PLWAH makes PLWAH+ more suitable for indexing in large-scale and high-speed streaming network data.

5. REFERENCES

[1] G. Antoshenkov, Byte-aligned bitmap compression, in: DCC'95, p. 476, 1995.

[2] F. Deli`ege and T. B. Pedersen. Position list word aligned hybrid: optimizing space and performance for compressed bitmaps. EDBT '10, 2010.

[3] Wu, Kesheng, Ekow J. Otoo, and ArieShoshani. "Compressing bitmap indexes for faster search operations." SSDBM'02, pp. 99-108, 2002.

[4] Van Schaik, Sebastiaan et al. "A memory efficient reachability data structure through bit vector compression." ACM SIGMOD' 2011, pp.913-924, 2011.

[5] CAIDA, www.caida.org.

[6] O'Neil, Patrick E. "Model 204 architecture and performance." High Performance Transaction Systems. Springer Berlin Heidelberg, 39-59, 1989.

[7] O'Neil, Patrick, and DallanQuass. "Improved query performance with variant indexes." ACM SIGMOD Record, vol. 26, no. 2, pp. 38-49, 1997.

[8] Fusco, F., Stoecklin, M. P., and Vlachos, M. Net-fli: on-the-fly compression, archiving and indexing of streaming network traffic. Proceedings of the VLDB Endowment, 3(1-2), 1382-1393, 2010.

[9] Yuhao Wen, et al., SECOMPAX A bitmap index compression algorithm, ICCCN HotData'14, Shanghai, China, 2014.