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
changjh13@mails.tsinghua.edu.cn

Wen-Liang Huang
China Unicom Groups Labs
China Unicom Groups Beijing, China
huangwenliang@gmail.com

ABSTRACT
Archiving of the Internet traffic is essential for analyzing network events in the field of network security and network forensics. To achieve fast searching in archival traffic data, the bitmap index requires 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
C.1.3 [Processor Architectures]: Other Architecture Style—Heterogeneous (hybrid) system; C.2.3 [Network Operations]: Network monitoring.

Keywords
Big Data, Bitmap index, Index compression, Codebook, WAH, PLWAH, COMPAX.

1. INTRODUCTION
1.1 Bitmap Index
Indexing is the core technology underlying answering queries on a large-scale archival data. Although inverted indexing is widely used in web and full-text retrieval, it is not suitable for the retrieval of archival Internet traffic data. In a search engine such as Google, indexing is the critical infra-structure to support its services. And index compression is an extremely important problem to improve user experiences.

Bitmap indexing is another promising approach for processing complex queries in scientific data management database [3], which is a special kind of database index that uses bitmaps. It was first proposed by P. O’Neil in the design of Model 204 commercial database.

1.2 Bitmap Compression Technique
It is widely accepted that bitmap indexing [4-5] is a practical solution that satisfies all requirements for large-scale and high-speed archival storage, indexing and data querying on network flow information, while incurring big space consumption.

The Approximate Encoding, (AE), is an example of a lossy bitmap compression scheme. An AE compressed bitmap index returns false-positives but is not guaranteed to return false-negatives. The accuracy of an AE compressed bitmap index ranges from 90% to 100%.

In run-length encoding, continuous sequences of bits are represented by one bit of the same value and the length of the sequence. The BBC (Byte-aligned Bitmap Code) [1] and the WAH are both lossless compression schemes based on run-length encoding. In terms of the performance, the WAH compression scheme is one of the most efficient CPU scheme, and is faster than BBC. However, WAH occupies a lot of storage, up to an average of 60% storage overhead over BBC.

<table>
<thead>
<tr>
<th>RowID</th>
<th>column</th>
<th>Bitmap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>=1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1.1 An example of the Bitmap index

As state-of-art, PLWAH (Position List Word-Aligned Hybrid) [2] and COMPAX compression scheme are proposed for bitmap indexes to make further improvement for the WAH scheme. They all find a good balance between compression and query response.

2. PLWAH+ CODING SCHEME
2.1 Definitions for Chunks
Bitmap is compressed by column. A column in a bitmap is a sequence of bits divided into chunks of 31 bits to ensure they are fit into the L1 cache. All the processes being carried out are based on the chunks which are more suitable for modern CPU architecture. At first, we will classify each chunk into different types. We define type for a chunk as below.

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

1-Filled Chunk: If the 31bits 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 their preceding Fill 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 0-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 as shown below:

- **LF**: For a continuous 2-tuple in the sequence, if the first element is NL 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 words.

### 2.3 Bit-Represented CodeBook

In this section, the final result of every codewords is represented by 4 bytes. The details are shown as follows:

For Literal, we add an ‘0’ before the 31 bits as a flag for identifying.

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

In the FL and LF words, the first five bits which represent first dirty bit position can’t be zero while as is shown above, while 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.

### 3. Experiment and Results

In our experiments, the network flow data from CAIDA [6] is parsed using libpcap library, and the fields of 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 network flow data is sorted to achieve faster searching in archival. Thus, the ratio of NI chunk is much lower than the original data. However, PLWAH+ has a better compression ratio than PLWAH with about 20% reduction of the amount of the literal words which cannot be piggybacked by PLWAH.

### 4. Extra Discussion

In this paper, we mainly discuss the PLWAH+32 which is typical and more suitable for the modern CPU architecture. To be more suitable for query response, we also adopt the method dividing the segment of 128 chunks, which maintains a good balance between compression and query response.

Firstly, the definition of an LF word that can piggyback more NI words, which is not considered in PLWAH.

Furthermore, PLWAH+ further classifies NI into two kinds, i.e., Nearly Identical 0 fill word (NI-0 word) and Nearly Identical 1 fill word (NI-1 word). According to a number of tests, a prediction can be naturally made that the PLWAH+ is more suitable to the complex database than PLWAH.

### 5. REFERENCES


