International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-3 Issue-1, October 2013

Two Tier Data Compression Method for Real-Time Databases

Nikita Bhatia, Richa Srivastava

***Abstract*—Modern day applications handle large volumes of data. These applications involve real-time data manipulations to be carried within time constraints. So real-time databases are used in most of the real-time applications. For efficient utilisation of database, with no compromise on speed, various compression methods are used to compress the data in a real-time database. In this paper, we propose a two-stage compression process. This process uses two algorithms- Swinging Door algorithm and LZSH algorithm. The survey indicates that this two stage compression results in highly compressed data. The compression time is always less than the database computational time.**

***Index Terms*—Real-time database, Data Compression, Swinging Door algorithm, LZSH algorithm.**

I. INTRODUCTION

Traditionally, real-time systems manage their data in application dependent structures. As real-time systems evolve, their applications become more complex and require access to more data. The main drawback of this approach is that, as the applications grow in complexity and amount of data, the code which deals with data management becomes increasingly difficult to develop and maintain. It thus becomes necessary to manage the data in a systematic and organized fashion. Database management systems provide tools for such organization, so in recent years there has been interest in merging database and real-time technology. The resulting integrated system, which provides database operations with real-time constraints, is generally called a real-time database system (RTDBS). Real-time database systems are the most promising alternative to manage the data with a structured and systematic approach. There are few applications that need to use real-time databases. Some of them have stringent timing requirements. For example, Internet Service Management, the traffic has grown tremendously in recent years. The service provider streamlines allocation of resources to subscribers, resource management, authentication, authorization, accounting and controls all the relevant network elements. Such processes should be carried out at processing time of the order of microseconds. The service provider databases should be strong enough to cater to about millions of users with such small processing periods. So for such applications, real-time databases are employed. [1]

Manuscript published on 30 October 2013.
* Correspondence Author (s)
Nikita Bhatia*, Department of Electronics and Telecommunication, Vivekanand Education Society's Institute of Technology, Mumbai, India.
Richa Srivastava, Department of Electronics and Telecommunication, Vivekanand Education Society's Institute of Technology, Mumbai, India.

Data stored in databases keep growing as a result of businesses requirements for more information. A big portion of the cost of keeping large amounts of data is in the cost of disk systems, and the resources utilized in managing the data. Databases contain all types of data, text, image, audio, video.

To optimize data storage within database, data needs to be compressed. Compression reduces size of the file and hence saves space while storing it. It also reduces the time required for retrieving data from database. A typical compression technique may offer space savings, but only at a cost of much increased query time against the data. There are several places where data can be compressed, either external to the database, or internally, within the DBMS software.[1]

For further optimization, data can be compressed for more than one time. In this paper, we present a two stage compression algorithm for real-time databases. There are two types of compression techniques-lossy compression and lossless compression. It is highly advisable to use lossy compression during first stage of compression. The input for first stage compression is the original data which contains identical information. So, loss of identical data is affordable. At this stage, the output is compressed data. This compressed data is further compressed at second stage. The second stage should be lossless compression technique. This is so because data is already compressed and in the event of losses it may happen that a huge chunk of vital original may be lost.

In this paper we present a lossy real-time database compression technique, "Swinging door algorithm ". The second stage lossless compression technique presented is "LZSH algorithm".

II. SWINGING DOOR ALGORITHM

Swinging door algorithm [2] is a lossy compression algorithm. According to this algorithm, a value is not stored if the time between that value and the last stored value is less than the compression minimum time. The exception to this rule is when a value changes from good to bad status or bad to good status. And a value is always stored if the time between that value and the last stored value is greater than the compression maximum time.

As shown in the Fig. 1, after initialization of the system, the first step is to receive the signal values of a first data point. In the next step, error offset values are established above and below the first data point, from which two lines, called doors, point to each next successive point. The doors each pivot about their respective offset points. The upper door, initially pointing straight down, turns only counterclockwise, and the lower door, initially pointing straight up, turns only clockwise. The doors are rotated accordingly to accommodate points by slope comparison method.

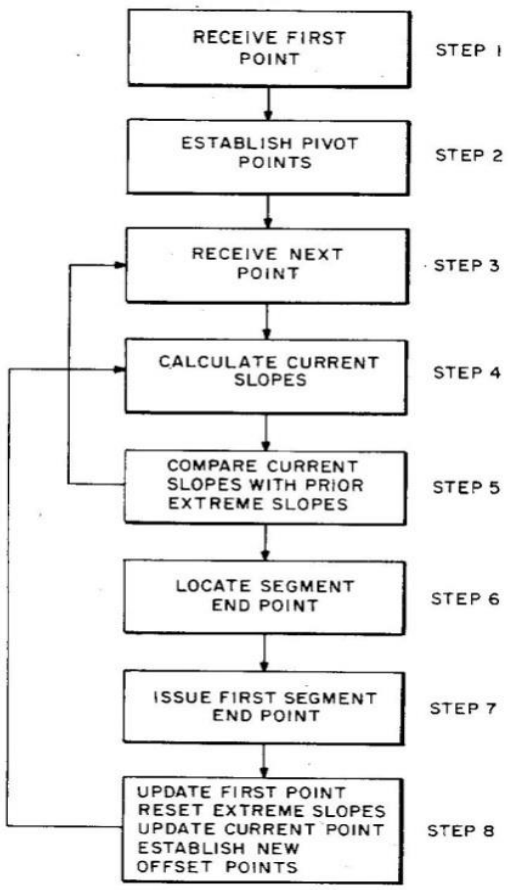

Fig. 1

The slope comparison method involved in this algorithm is can be explained with the help of Fig. 2. For the given data stream to be compressed, a configuration record is initially recorded at the head of the data stream. The x axis in the figure represents time and the y axis represents data point value.

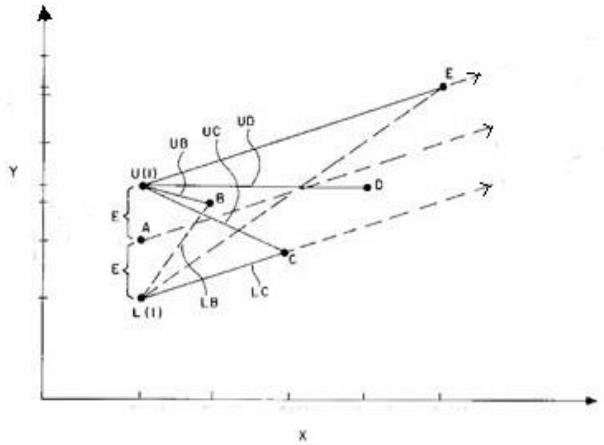

Fig. 2

Step 1: After the first point A is received, two pivot points L and U are established by setting error bound above and below
Step 2: Second data point B is received.
Step 3: Two segments UB and LB are constructed from the upper and lower pivots U and L to point B. These are the swinging doors that pivot on the offset points U and L. The slopes of the two doors SU(1) and SL(1) respectively are calculated.

Step 4: The current slopes SU(1) and SL(1) are now compared with the previous maximum SU(MAX) and minimum SL(MIN) door slopes calculated during the current corridor set of points. In the present case, processing only the second data point in the corridor; there are no prior maximum and minimum slopes. The slopes SU(1) and SL(1) are then the maximum and minimum prior slopes and they are within their own limits.

Third data point C is received and steps 2 and 3 are repeated for it and its slopes SU(2) and SL(2) are calculated. Repeating step 4 for point C, the current upper SU(i) and lower SL(i) door slopes as exampled here by SU(2) and SL(2) are compared with the previous extreme maximum and minimum values for these values SU(MAX) and SL(MIN), exampled here by SU(1) and SL(1), and one of the following actions taken:

1. If SU(i) is greater than SU(MAX), update SU(MAX) with SU(i) and retain the values for (DY(i), DX(i)) as a maximum point Y(MAX), X(MAX).
2. If SL(i) is less than SL(MIN), update SL(MIN) with SL(i) and retain the values for (DY(i), DX(i)) as a minimum point Y(MIN), X(MIN).
3. If neither of the conditions 1. or 2. occur, then update the current last point D(i-1) that is inbounds, that is between the Swinging Doors, with the current point D(i) as the last inbounds point and repeat the process from step 2 above.
4. If SU is greater than SL, the doors are being opened wider than parallel, so additional processing occurs. First, a corridor end point C(i) is generated to represent the end of the current corridor.

In the displayed case of figure, the upper slope SU(1) of the upper door segment from U(1) to B is greater than the slope SU(2) of the segment from U(1) to C, that is the first upper door UB is more open than the second upper door UC. The upper door is then left open as defined by the first segment UB. The lower slope SL(1) of the lower door segment LB is greater than the slope SL(2) of segment LC as displayed, that is the second lower door from L(1) to C is more open than the first lower one from L(1) to point B. In this case the lower door is opened or swung wider to accommodate point C or more technically the minimum door slope SL(MIN) is updated with SL(2). The doors are now positioned as UB for the upper door, and LC for the lower door. The sequence of steps 2, 3 and continue to be repeated. Point D is received, and in this example the upper door UD is swung open to accommodate it. The upper door is always opened with increasing slopes; while the lower door is always opened with decreasing slopes but note that for the same point the upper door slope is always less than the lower door slope, so as the doors open their slopes approach equality. Each door is opened only to the extent necessary to include all earlier points, and only until the doors form parallels. Every time one of the doors is opened either a new upper or lower extreme point has been encountered that establishes a new extreme upper or lower boundary condition.

Finally, point E is received. To open the upper door to point E would increase the upper door slope from point U to point E as wide as the widest lower door slope which as depicted runs from point L to point C.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The result of the Swinging Door process is then a series of points being the corridor segment end points. All intermediate points have been dropped reducing the data stream according to the error width E.

The swinging door method is preferred as the processing is simple and direct. The calculation time for the swinging door compression becomes less significant as the rate of data compression increases; that is, the number of points within the error bounds of a segment gets large. The processing time for the calculation is similar to other compression procedures.

III. CHARACTERISTICS OF INTERMEDIATE DATA

Intermediate data is the data received from the lossy compressor based on the Swinging Door algorithm. This intermediate data has very high change rate which causes great challenges to the lossless compressor. Also after the first stage compression, intermediate data's each length offset pair is compared with three latest used pair. It is observed that 31% of length-offset pairs are equal to their last used pairs; 11% of them are equal to their second last used pairs and 6% of them are equal to their third last used pairs.

It is observed from the results that intermediate data's characteristics differ a lot from the original data. It has a high volatility as compared to original data. So the next step is to use a new lossless algorithm LZSH in second stage for compression of this data.

IV. LZSH ALGORITHM

LZSH is a lossless compression algorithm based on LZ77 but has two major improvements on the latter [3]. Firstly, LZSH uses one-bit flags named match-flag to indicate whether the next set of data is a single literal byte or a encoded length-offset pair. Secondly, LZSH uses two-bit flags named dic-flag to determine the encoding mode of the length-offset pair. Both the flags have been described in the Table 1.

match flag	dic flag	Compressed code
0	N/A	Next uncompressed literal byte.
1	00	Encoded length-offset pair.
	01	N/A (length-offset pair is equal to the last used pair)
	10	N/A (length-offset pair is equal to the second last used pair)
	11	N/A (length-offset pair is equal to the third last used pair)

Table 1

The LZSH compression process is shown in Fig. 4.
Step S01, first of all, find the longest match in LZ sliding window and store its length-offset pair in memory.
Step S02, the availability of the match found in step S01 needs to be judged. If the match's length is less than 3, set match-flag to '0' and output the next uncompressed literal byte; otherwise set match-flag to '1' and continues Step S06.
Step S06, compressor compares the length-offset pair with three latest used pairs. If the length-offset pair is equal to the last used pair, set dic-flag to '01'; if the length-offset pair is equal to the second last used pair, set dic-flag to '10'; if the length-offset pair is equal to the third last used pair, set dic-flag to '11'; otherwise set dic-flag to '00' and encode the length-offset pair.
Step S14, move the sliding window and keep LZSH compression process is repeated from Step S01.

LZSH is a byte-based model instead of bit-based model. To preserve byte alignment, eight continuous match-flags or four continuous dic-flags are clustered into groups preceded by a flag byte.

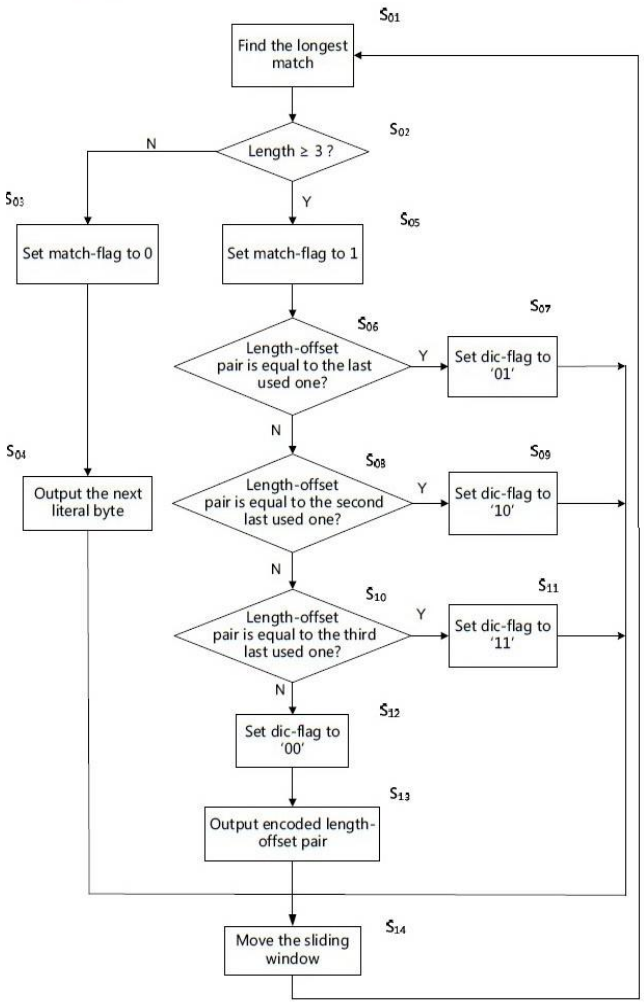

Fig. 4

V. PERFORMANCE TEST

The performance test of the lossless compression of intermediate data from lossy compressor is given in the following table:

Algorithm	Size of Intermediate Data	Size of Compressed Data	Compression Time
LZ77	50k	38.9k	64ms
LZW	50k	25.6k	118ms
LZSH	50k	23.8k	66ms

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

REFERENCES

[1] Web course on "Real-time Databases" Version 2, CSE, IIT Kharagpurhttp://nptel.iitk.ac.in/
[2] G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.
[3] Edgar H. Bristol, "Data compression for display and storage", US4669097 A, The Foxboro Company, May 1987
[4] Si-huiShu, Yi Shu, "A Two-Stage Data Compression Method For Real-time Database", 3rd International Conference on System Science, Engineering Design and Manufacturing Informatization, 2012

Nikita Bhatiais a student of final year of Electronics and Telecommunication. She studies at VES Institute of Technology, Mumbai. She has participated in various inter-collegiate and intra-collegiate technical events like technical paper presentation, technical presentation, etc. She presented a paper on "Intelligent Transport System" and won the presentation competition conducted by IEEE-VESIT in February 2013. She is a member of different technical societies like IEEE (Institute of Electrical and Electronics Engineers), CSI (Computer Society of India), ISTE (Indian Society for Technical Education).

Richa Srivastava is a final year student of Electronics and Telecommunication Engineering at V. E. S. Institute of Technology, Mumbai. She has participated in various inter-collegiate, intra-collegiate and national level technical paper presentations. She is an active member of two student chapters- ISTE-VESIT and IEEE-VESIT.