Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Data Compression on Columnar-Database Using Hybrid Approach (Huffman and Lempel-Ziv Welch Algorithm)

Dalvir Kaur *
Master of Technology in CSE,
Sri Guru Granth Sahib World University,
Fatehgarh Sahib, Punjab, India.

Kamaljeet Kaur
Assistant Professor, Department Of CSE,
Sri Guru Granth Sahib World
University, Fatehgarh Sahib, Punjab, India.

Abstract—Columnar Oriented Database is an enhance approach to service to service the needs of Business Intelligence. A Columnar Oriented Database Management system that stores content by columns rather than the row. This type of data differs from traditional database with regards to performance, storage requirements and easy to modification of the schema. One of the major advantage of column based database approach has easiest to understand is its effect on compression and possible to apply compression algorithm for each column. Compression is a technology foe reducing the quantity of data used to represent any content without reducing the quality of the data. In this paper we used the data compression on columnar –oriented database sing Hybrid approach. This consists of two lossless Algorithm Huffman Coding and Lempel-Ziv Welch Coding in MATLAB software.

Keywords—Columnar Database, coding, Decoding, Huffman coding, Hybrid approach, Lempel-Ziv Welch, Lossless compression.

I. INTRODUCTION

One of the most-often cited advantages of Column-Stores is data compression. Compression improves performance by reducing the size of data on disk, decreasing seek times, increasing the data transfer rate. Intuitively, data stored in columns is more compressible than data stored in rows[1][2]. Compression is useful because it helps reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth. Compression is possible for data that are redundant or repeated in a given test set. Compression method is broadly divided into lossless and lossy methods, lossless method which can reconstruct the original data exactly from the compressed data and lossy method which can only reconstruct an approximation of the original data[10][11].

In this paper we provide Hybrid approach on Columnar Database using Lossless Huffman Coding and Lempel-Ziv Welch Algorithm Features. These algorithms are used to compress the data image. In this paper we can discuses only the Huffman and LZW coding decoding on data images and show the image compression using MATLAB software.
Compression is achieved by removing one or more of the three basic data redundancies:
1) Coding redundancy, which is presented when less than optimal code words are used;
2) Inter pixel redundancy, which results from correlations between the pixels of an image;
3) Psycho visual redundancy, which is due to data that are ignored by the human visual system [13].

II. HUFFMAN ALGORITHM

A. Huffman Encoding

Huffman code is mapped to the fixed length symbols to variable length codes. Huffman algorithm begins, based on the list of all the symbols or data which are arranged in descending order of probabilities. According to the probabilities, the code words are assigned.[11] Shorter code words for higher probabilities and longer code words for smaller probabilities are assigned. Next it generates a binary tree, by the bottom-up approach with a symbol at
every leaf. This has some steps, in each step two symbols with the smallest frequencies are chosen, added to the top of the partial tree. The selected smallest frequencies symbols are deleted from the list, which are replaced by a secondary symbol denoting the two original symbols. Therefore, the list is reduced to one secondary symbol, which
denotes the tree is complete. Finally, assign a codeword for each leaf or symbol based on the path from the root node to symbols in the list [13] .

- Two families of Huffman Encoding have been proposed:-

1. Static Huffman Algorithms
2. Adaptive Huffman Algorithms

1. Static Huffman Algorithms calculate the frequencies first and then generate a common tree for both the compression and decompression processes . Details of this tree should be saved or transferred with the compressed file[12].
2. The Adaptive Huffman algorithms develop the tree while calculating the frequencies and there will be two trees in both the processes. In this approach, a tree is generated with the flag symbol in the beginning and is updated as the next symbol is read[12].

B. Features of Huffman coding

- Store/transmit a maximum amount of information in a minimum amount of space/time.
- Huffman codes are Prefix tree binary code trees.
- Code generate by the Huffman algorithm achieve the ideal code length up to the bit boundary[].

C. Huffman Decoding

After the code has been created, coding and/or decoding is accomplished in a simple look-up table manner. The code itself is an instantaneous uniquely decodable block code. It is called a block code, because each source symbol is mapped into a fixed sequence of code symbols. It is instantaneous, because each code word in a string of code symbols can be decoded without referencing succeeding symbols. It is uniquely decodable, because any string of code symbols can be decoded in only one way [11]. Thus, any string of Huffman encoded symbols can be decoded by examining the individual symbols of the string in a left to right manner [13]. After decoding the above Huffman codeword by using the Huffman tree we get the original message.

III. LEMPEL ZIV WELCH ALGORITHM

Lempel-Ziv-Welch (LZW) is a category of a dictionary–based compression method [11]. It maps a variable number of symbols to a fixed length code. LZW places longer and longer repeated entries into a dictionary. It emits the code for an element, rather than the string itself, if the element has already been placed in the dictionary. In a dictionary-based data compression technique, a symbol or a string of symbols generated from a source alphabet is represented by an index to a dictionary constructed from the source alphabet. In a dictionary based coding to build a
dictionary that has frequently occurring symbols and string of symbols. When a new symbol or string is found and that is contained in the dictionary, it is encoded with an index to the dictionary. Or else, the new symbol is not in the dictionary, the symbol or strings of symbols are encoded in a less efficient manner[14].

- Have 2 phases:
  - Building an indexed dictionary
  - Compressing a string of symbols

A. LZW Encoding

A high level view of the encoding algorithm is shown here:

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Emit the dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2[5].

A dictionary is initialized to contain the single-character strings corresponding to all the possible input characters. The algorithm works by scanning through the input string for successively longer substrings until it finds one that is not in the dictionary. When such a string is found, the index for the string without the last character (i.e., the longest substring that *is* in the dictionary) is retrieved from the dictionary and sent to output, and the new string (including the last character) is added to the dictionary with the next available code. The last input character is then used as the next starting point to scan for substrings [11].

In this way, successively longer strings are registered in the dictionary and made available for subsequent encoding as single output values. The algorithm works best on data with repeated patterns, so the initial parts of a message will see little compression. As the message grows, however, the compression ratio tends asymptotically to the maximum.

B. *LZW Decoding*

The decoding algorithm works by reading a value from the encoded input and outputting the corresponding string from the initialized dictionary. At the same time it obtains the next value from the input, and adds to the dictionary the concatenation of the string just output and the first character of the string obtained by decoding the next input value[14]. The decoder then proceeds to the next input value and repeats the process until there is no more input, at which point the final input value is decoded without any more additions to the dictionary.In this way the decoder builds up a dictionary which is identical to that used by the encoder, and uses it to decode subsequent input values. Thus the full dictionary does not need be sent with the encoded data; just the initial dictionary containing the single-character strings is sufficient [11].

C. *Example of LZW*

Eg: source: ABCABCBACAACBAA (15 bytes)

Encoding

Table1. Encoding using LZW

Input symbol	*Output symbol / index*	*Dictionary index*	*Dictionary value*
AB	A	256	AB
C	B	257	BC
A	C	258	CA
BC	256	259	ABC
B	C	260	CB
A	B	261	BA
C	A	262	AC
AA	258	263	CAA
CB	262	264	ACB
AA	261	265	BAA
(End Of File)	A		

Decoding:

While decoding the program will build dictionary as well [14].

Table 2. Decoding Using LZW

Input symbol / index	*Output symbol*	*Dictionary index*	*Dictionary value*
AB	A	256	AB
C	B	257	BC
256	C	258	CA
C	AB	259	ABC
B	C	260	CB
A	B	261	BA
258	A	262	AC
262	CA	263	CAA
261	AC	264	ACB
A	BA	265	BAA
(End Of File)	A		

IV. PROPOSED HYBRID APPROACH

This paper we discused proposed a Hybrid compression technique using the two lossless methodologies Huffman coding and Lempel Ziv Welch coding to compress data image. In the first stage, the image is compressed with Huffman coding and calculate the MSE,PSNR, CR and elapsed time of an data image. After that apply the Lempel ziv compression on same image and calculate the results .from the experiment results further we can use the both algorithms and recover the data image reduced the size of data image and calculate same results .

A. STEPS AND FLOW DIAGRAM

1. Understanding the columnar database.
2. Creation of columnar databases.
3. Read the data image on to the workspace of the mat lab.
4. Make double precision for image.
5. Call a function which will find the symbols (i.e. pixel value which is non-repeated).
6. Call a function which will calculate the probability of each symbol.
7. Probability of symbols are arranged in decreasing order and lower probabilities are merged and this step is continued until only two probabilities are left and codes are assigned according to rule that the highest probable symbol will have a shorter length code.

8. Further Lempel ziv Welch encoding is performed the LZW Dictionary results.
9. Apply Hybrid approach for data compression using Huffman coding and Lempel_Ziv Algorithm.
10. Analysis of compressed image and calculation of PSNR, MSE,CR and Elapsed time using Huffman compression and Lempel –Ziv compression.

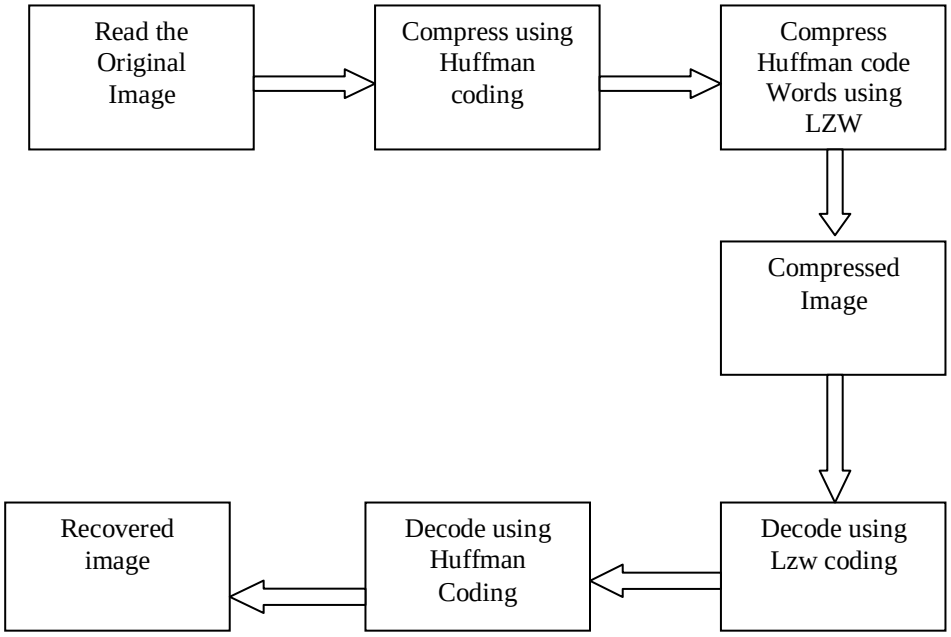

Figure 1 Flow Chart

B. Original Data Image
First we read the data image on the workplace of the matlab.

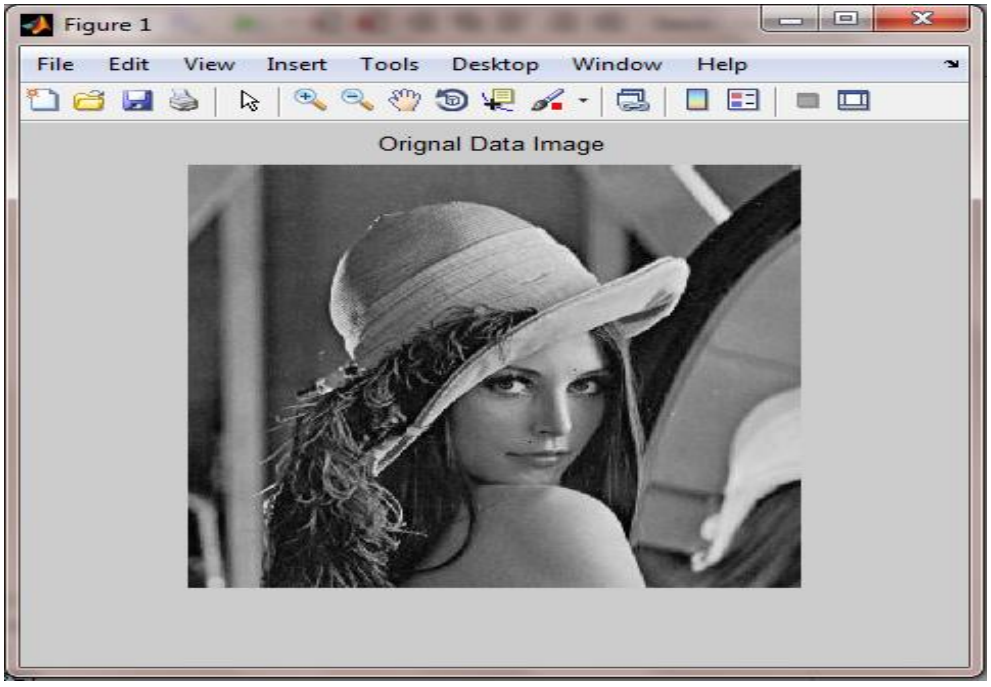

Figure 1 Original Data Image

C. Huffman Compression
After that we apply the Huffman coding on original image and calculate the different stages of encoding and decoding and calculate the MSE,PSNR,CR and Elapsed Time of data image.

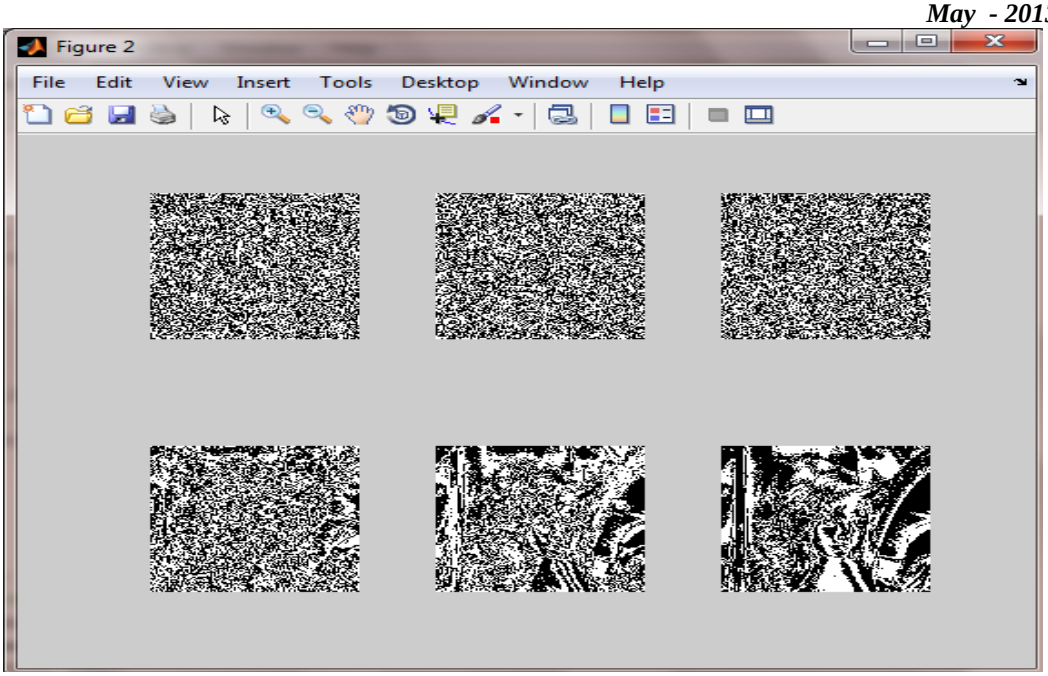

Figure 2 Huffman Coding Decoding Image

D. LZW Compression

After that we apply the Lempel Ziv Welch Algorithm on original image and calculate the different stages of encoding and decoding and calculate the MSE,PSNR,CR and Elapsed Time of data image.

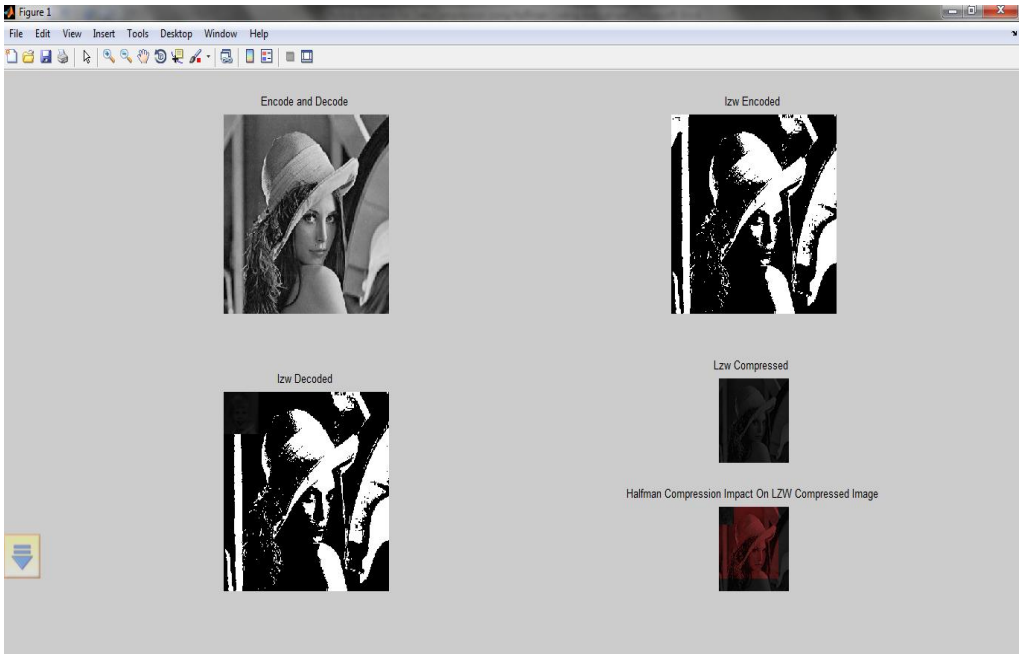

Figure 3 LZW Coding Decoding Image

V. CONCLUSION

This experiment confirms that the higher data redundancy helps to achieve more compression. The above presented new compression and decompression technique called as Hybrid approach based on Huffman coding and Lempel Ziv coding is very efficient technique for compressing the data image. There are different lossless data compression are used to compress the data image we use the but we use the Hybrid approach on columnar database in this paper apply only the Huffman and Lempel ziv Welch algorithm for compression on gray scale data images in future we can apply the some features of both algorithms on different data images and calculates the final results.

REFERENCES

[1] Abadi Daniel, R.Samuel Madden, Miguel C. Ferreira, "Integrating Compression and Execution in Column Oriented Database Systems",*SIGMOD 2006, Chicago, Illinois, USA*. June 27–29, 2006.

[2] Abadi Daniel J, Peter A. Boncz, Stavros Harizopoulos, *"Column-oriented Database Systems" VLDB '09,* August 24-28, 2009.

[3] Bajaj Punam Simranjit Kaur Dhindsa "A Vision towards Column-Oriented Databases" International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 4, June – 2012

[4] C. Saravanan and R. Ponalagusamy, "Lossless Grey-scale Image Compression using Source Symbols Reduction and Huffman Coding",International Journal of Image Processing (CSC Journals), Vol.3, Iss.5, pp.246-251, 2009.

[5] C. Saravanan, M. Surender "Enhancing Efficiency of Huffman Coding using Lempel Ziv Coding for Image Compression" International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January 2013

[6] Column-Oriented DBMS, Wikipedia.

[7] Data Compression types, Wikipedia.

[8] Mamta Sharma, "Compression Using Huffman Coding", International Journal of Computer Science and Network Security, Vol.10, No.5, May 2010.

[9] M.J.Weinberger, G.Seroussi, and G. Saprio. "LOCO-1 Lossless, Image Compression Algorithm: Principles and Standardization into JPEG-LS", IEEE trans. Image Processing, pp.1309 - 1324, August 2000.

[10] E.Guy Blelloch "Introduction to Data Compression "Computer Science Department Carnegie Mellon University January 31, 2013.

[11] R.S.AARTHI, D. MURALIDHARAN, P. SWAMINATHAN "DOUBLE COMPRESSION OF TEST DATA USING HUFFMAN CODE" Journal of Theoretical and Applied Information Technology 15 May 2012. Vol. 39 No.2

[12] S.R. Kodituwakku, U. S.Amarasinghe"COMPARISON OF LOSSLESS DATA COMPRESSION ALGORITHMS FOR TEXT DATA" *Department of Statistics & Computer Science, University of Peradeniya, Sri Lanka*2007.

[12] V.Gordon Cormack"Data compression on a database system" *ACM, 28(12):1336–1342*, 1985.

[13] http://www.skylondaworks.com/sc_huff.htm

[14] .http://www.TheLZWcompressionalgorithm.html

