A GENERALIZED DIGITAL DATABASE TEXT COMPRESSION SCHEME COMPARED WITH ASCII

Sushil Kumar[1], Dr. Anoop Kumar Chaturvedi[2]
[1]Research Scholar, LNCTU, Bhopal, [2]LNCTU, Bhopal
[1]Sushil.tit@gmail.cm, [2]anoop.chaturvedi77@g mail.com

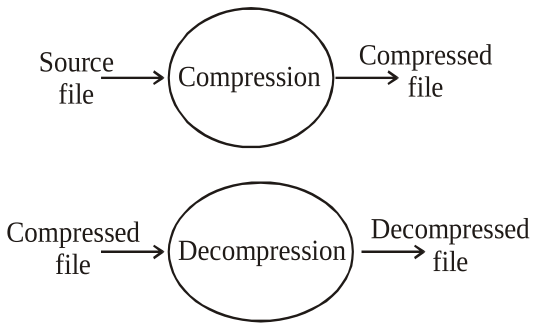
Abstract- In this paper we investigate that for experimental result shows the proposed scheme provides significant improvement in compression efficiencies. The main proposal of this paper is to reduce the memory space and the transmission time. We will also produce an experimental result which shows that our technique achieves good compression ratios. Reduction of compression ratio collection of large amount of data in almost all application is very important. In today’s era without computer application one can never thinks of database. Whenever database comes compression plays very important role. Application may be stock exchange, banking, reservations in almost every field compression becomes primary things. With the help of compression one can make very big data in very less memory space. Disc input and output performance has bottleneck for very large databases. Database compression can be used to reduce disk input output bandwidth requirements for very big amount of data transfers. The authors explores and propose techniques managing the compressed database such that standard operations like retrievals, inserts, deletes and modifications are supported.

Keywords- Compression, Compression Ratio, Delta code, Differential Method, Fixed Length Coding (FLC), Huffman after using Fixed Length Code (HFLC), Lossy Compression, Nonlossy Compression, RLE (Run Length Encoding), Temporal Database, LZW (Lampel Ziv Welch).

1. INTRODUCTION

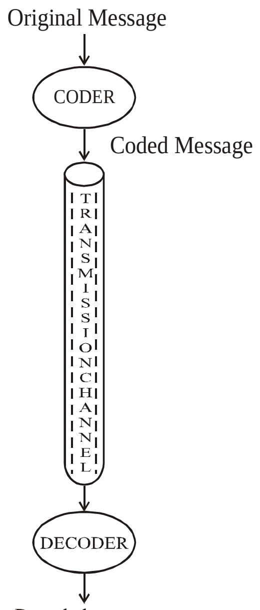
DATA Compression can be viewed as a means for efficient representation of a digital source of data such as text, image, sound or any combination of all these types such as video. The final goal of data compression is to represent a source in digital form with as few bits as possible while getting the minimum requirement of reconstruction of the original. Any compression will not work unless a means of decompression is also provided due to the nature of data compression.

A compression algorithm is often called compressor and the decompression algorithm is called de-compressor. The compressor and de-compressor can be located at two ends of a communication channel, [1] at the source and at the destination respectively as shown in fig.(1) also known as coder and decompression as decoder as shown in fig.(2).



Compressor and Decompressor

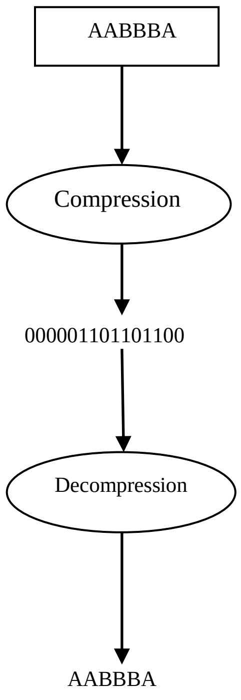
Fig. (1)



Coder and Decoder

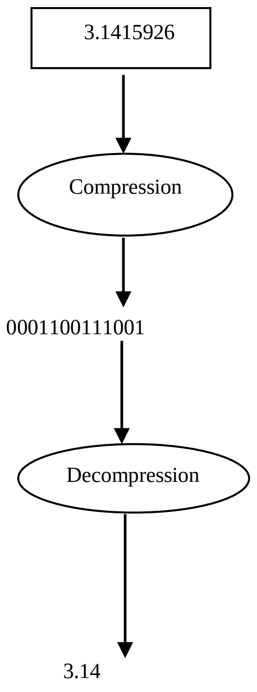
Fig. (2)

There are two major families of compression technique; they are called lossless and lossy compression. A compression is lossless only if it is possible to get exactly original data from the compressed version. There is no loss of information during the compression process. Fig. (3) will give the exact idea about lossless compression. Lossless compression is used when the original data of a source are so important that one cannot afford to lose any data, for example medical images.



Lossless Compression

Fig. (3)



Lossy Compression

Fig. (4)

A compression technique is lossy if it is not possible to reconstruct the original exactly from the compressed version. Lossy compression is called irreversible compression since it is impossible to recover the original data exactly by decompression. Fig. (4) shows one of the major used examples of lossy compression.

DATA Compression is the science and art of representing information in a compact form [7].
The data may also classified as text, audio, image and video while the real digital data format consists of 0's and 1's in a binary format

- Text data are usually represented by 8-bit extended ASCII code or EBCDIC having extension .txt, .tex, .doc.
- Binary data include data base file spreadsheet data, excitable files and program codes having extension as .bin.
- Image data are represented often by a two dimensional array of pixels in which each pixel is associated with its color code having extension as .bmp and .psd.
- Graphics data are in the form of vectors or mathematical equations, for example data format is .png (portable network graphics).
- Sound data are represented by a wave function having extension as .wav [6].

2. BACKGROUND ON TEXT COMPRESSION TECHNIQUE

Lossy compression achieves better compression by losing some information. When compressed stream is decompressed, the result is not identical data stream. Such a method makes sense especially in compressing images movies or sounds. In contrast, text files, especially files containing computer programs, may become worthless if even one bit get modified. Such files should be compressed only by a lossless compression method.
A data compression method is called universal if the compressor and de-compressor or do not know the statistics of the input strain [12].
There are different compression techniques

2.1 RLE:

If a data item d occurs n consecutive times in the input stream, replace the n occurrences with the single pair nd. The n consecutive occurrences of a data item are called a run length of n, and this approach to data compression is called run length encoding or RLE.
The runs are replaced by a tuple (r, l, s) for (run-flag, run-length, run symbol) respectively, where s is a member of alphabet of symbols but r and l are not.
For example: String KKKKKKKKK, containing a run of 9 k's can be replaced by triple('r',9,'k'), or a sort unit r9k consisting of the symbol r, 9 and k where r represents the case of 'repeating symbol', 9 means '9 times of occurrence' and k indicates that this should be interpreted as 'symbol k' (repeating 9 times).
Another example
Input is GGG ○○○ ○○○ BCDEFG ○○ 55GHJK ○ LM77777.

Output is r_3Gr_6n_6BCDEFG$r_2n_9$55GHJK ○ LM$r_5$7.
Solution is:

1. For first three G encoded as r_3G.
2. Next six space encoded r_6.
3. The non-repeating symbols BCDEFG encoded by n_6BCDEFG.
4. Next two spaces r_2.
5. Nine non repeated $n_9$55GHJK ○ LM.
6. Next five 7's are encoded by $r_5$7.

The mathematical model for Run-Length is so-called Markov Model [22].

1. If two characters are repeated then there is increase in size, so in place of compression it becomes expansion in size.

2. And for three repeated characters there is no compression.

3. But four onwards there will be actual compression starts.

The method will be advantageous when repetition is more than three.
RLE compression data can be further compressed using 5-bits and 6-bits coding scheme.

2.2 LZW:

LZW is a general compression algorithm capable of working on almost any type of data. It is generally fast in both compressing and decompressing data and does not require the use of floating-point operations. Also LZW writes compressed data as bytes and not as words.
LZW is referred as a substitution or dictionary-based encoding algorithm. The algorithm builds a data dictionary (also called a translation table or string table) of data occurring in an uncompressed data stream. Patterns of data (substrings) are identified in the data stream and are matched to entries in the dictionary. If the substring is not present in the dictionary, a code phrase is created based on the data content of the substring, and it is stored in the dictionary. The phrase is then written to the compressed output stream.
Data Compression technique on text Files: A comparison study has been done by Haroon Altarawneh et. al., he has taken different methods of data compression English text files, LZW, Huffman, Fixed Length Coding (FLC) and Huffman after using Fixed Length Code (HFLC). He evaluated a test on these algorithms on different text files or different file sizes and taken a comparison in terms of comparison: Size, Ratio, Time (Speed) and entropy. And they found that LZW is the best algorithm in all the compression scales [25].
According to them LZW is a general compression algorithm capable of working on almost any type of data. It creates a table of strings commonly accruing in the data being compressed, and replaces original data with reference into the table. LZW Compression replaces strings of characters into a single code. Compression occurs when a single code is output instead of a string of characters. It starts with a dictionary of all the single character with indexes. It starts expanding the dictionary as information gets send through. Pretty soon, redundant strings will be coded as a single bit, and compression has occurred.
The drawback is, Compression usually does not begin until a sizable table has been built.
They have lastly concluded that LZW is the best in all compression scale, especially on the large files, than Huffman, HFLC and FLC respectively. FLC is a good technique if the source file contains little number of different symbols (less than 16). Huffman gives better than HFLC and the second one need more time and more calculations but it is better than FLC.

3. PROPOSED NEW TECHNIQUE

- Characters can be coded into 5-bits coding.
- Memo can be coded into 6-bits coding.
- Dates can be coded into 16-bits coding.
- Time can also be coded into 16-bits coding [17].

4. IMPLIMENTATION OF PROPOSED TECHNIQUE

We have proposed some new techniques that can be implemented as given below:

4.1 Character

In general 8-bit ASCII code have been used for representing character, but when one declare any attribute to be of character type they often interested only in alphabet character from A-Z or a-z.
So it has redefined the coding in the following way.

Table 1. Modified Characters Code for **Character** Field

DECIMAL	CHARACTER	DECIMAL	CHARACTER
0	NOTHING	16	p
1	a	17	q
2	b	18	r
3	c	19	s
4	d	20	t
5	e	21	u
6	f	22	v
7	g	23	w
8	h	24	x
9	i	25	y
10	j	26	z
11	k	27	SPACE
12	l	28	END OF LINE
13	m	29	COMMA
14	n	30	FULL STOP
15	o	31	‘ ‘

So in this way a compact and complete representation of information is possible.

4.2 *Memo*

The memo field often includes character other than alphabets like number underscore plus minus etc. and it is found that at most 61 symbols are used in general so in this situation 6 bits are sufficient to represent them. Out of these 32 are the same as that in the previous case of character and the remaining 32 could be used for the following purposes.

Table 2. Modified Characters Code for **Memo** Field

DECIMAL	CHARACTER	DECIMAL	CHARACTER
32	“ “	48.	&
33	0	49.	*
34	1	50.	(
35	2	51.	)
36	3	52.	-
37	4	53.	_
38	5	54.	=
39	6	55.	+
40	7	56.	<
41	8	57.	>
42	9	58.	?
43	!	59.	/
44	@	60.	:
45	#	61.	;
46	$	62.	|
47	^	63.	\

So in this way complete information is possible.

5. RESULTS AND EVALUATION OF PROPOSED TECHNIQUE

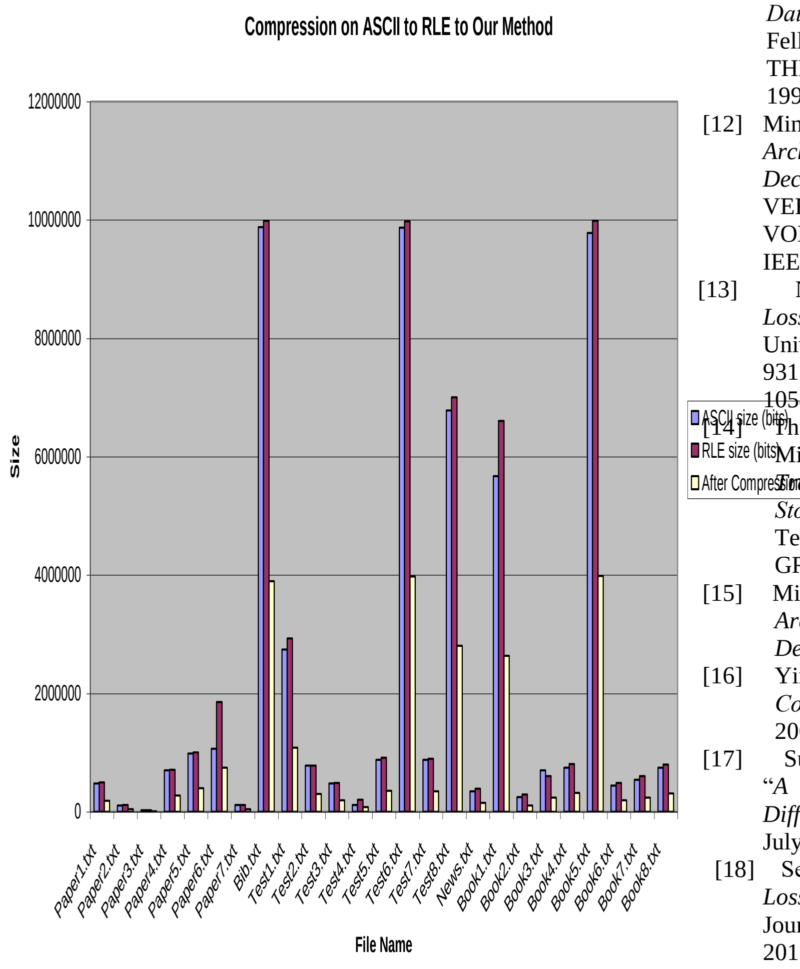
We have evaluated our proposed compression techniques on existing RLE method. Below Table1 has taken some 21 of different files with their size in number of bits. Then compressed it with the help of RLE compression and then on that we applied our methods then one can see our method is better. Below the table one can see that original size i.e. ASCII size in bits is 55094077. But the same file size has been increased to 58054222 bits. And when it comes to our methods it becomes only 22937800 bits. The same comparison is also can be seen graphically given below in last page of this paper. As in table one can see that almost all cases the size of file is increased when come to RLE compression. And this is true which we have discussed earlier in this paper. In the table some files have been taken on that experiment has been done. In modern database systems table structure is also stored together with the database file so that any application can make use of it. When we consider this compression scheme we will store this structure without any modification, it is only the data that will be stored according to this new scheme. The file also store some additional words like field separator, end of record to mark and distinguish separate attribute and record, and these will be there in proportion to the number of records in the file.

Table 3. List of files before Compression and after Compression

File Name	**ASCII size (bits)**	**RLE size (bits)**	**After Compression**
Paper1.txt	482978	493899	188675.67
Paper2.txt	103251	114142	44543.87
Paper3.txt	25677	26067	10038.44
Paper4.txt	697867	707952	278734.70
Paper5.txt	985567	999755	399902.00
Paper6.txt	1067865	1853265	741306.00
Paper7.txt	118957	118859	46543.60
Bib.txt	9876789	9987652	3894177.85
Test1.txt	2738527	2927652	1082770.74
Test2.txt	778965	776852	301740.80
Test3.txt	479211	489923	195969.20
Test4.txt	117528	206431	81662.32
Test5.txt	874321	911211	354464.39
Test6.txt	9865421	9974322	3978934.79
Test7.txt	879437	898344	349978.78
Test8.txt	6778943	7001233	2800999.67
News.txt	347954	391221	149976.47
Book1.txt	5674312	6600311	2640007.66
Book2.txt	248968	290142	110077.43
Book3.txt	698437	607421	240997.40
Book4.txt	746389	805072	321919.98
Book5.txt	9778899	9983241	3982176.40
Book6.txt	439567	491010	191303.01
Book7.txt	543268	602477	239990.80
Book8.txt	744979	795768	310908.20
TOTAL	**55094077**	**58054222**	**22937800**

6. CONCLUDING REMARK

In our paper we have taken original size of different files. On that applied RLE technique and then on RLE we applied our own technique. We have presented a very new compression technique for Text and Memo. Our technique will give better results, so that the proposed technique has better performance, compared to other techniques currently used in the Data-base. The CPU utilization will also increases due to compression. Also compression is beneficiate to Input and Output performance. Database compression is used to reduce the disc input output bandwidth requirements for bigger data transfer. The above technique will give 62.5% of compressions. Table 1 and table 2 will have modified chart according to our technique. Table 3 contains practical data on which our compression is giving better compression compare to RLE. Also the graph below can conclude that our compression is better than RLE technique.



REFERENCES

[1] A.S. Tanenbaum *"Computer Network"* (Fourth Edition Prentice-Hall of India Limited).

[2] Cormack, G. V. 1985. *"Data Compression on a Database System"*. Commun. ACM 28 12, (Dec.), 1336-1342.

[3] Debra A. Ielwer and Daniel S. Hirschberg *"Data Compression"* –IEEE JUNE 2002.

[4] Navathe S.B ,Elmasn R. *"Fundamentals of Database System"* (Pearson Education).

[5] Pujari. A. K "*Data Mining Technique"* (University Press).

[6] Reghbati, H.K *"An Overview of Data Compression Technique"* IEEE computer (1981).

[7] Saloman D. *"Data Compression The Complete Reference"* Springer, 3rd Edition (2004).

[8] William Stallings, *"Network Security Essentials Application and Standard"* (Pearson Education).

[9] Holger Kruse, Amar Mukherjee, *"Data Compression Using Text Encryption"* FL 32816 Page No. 1068-0314/97 Years 1997 IEEE Department of Computer Science University of Central Florida Orlando, 32816.

[10] Jianzhong Li and Hong Gao *"Efficient Algorithms for On-line Analysis Processing On Compressed Data Warehouses"* Harbin Institute of Technology, China.

[11] En-hui Yang and John C. Kieffer, *"On the Performance of Data Compression Algorithms Based Upon String Matching"* Fellow IEEE, IEEE TRANSACTIONS ON INFORMATION THEORY, VOL, 44, NO. 1, JANUARY 1998 0018-9448 1998 IEEE.

[12] Ming-Bo Lin, Member and Yung-Yi Chang, *"A New Architecture of a Two-Stage Lossless Data Compression and Decompression Algorithm"* IEEE TRANSACTIONS ON VERY LARGE SCALEINTEGRATION (VLSI) SYSTEMS, VOL, 17, NO, 9, SEPTEMBER 2009 1063-8210 Years 2009 IEEE.

[13] N. Magotra, W. McCoy', S. Stearns' Dept. of EECE, *"A Lossless Data Compression In Real Time F. Livingston."* University of New Mexico, Albuquerque, NM 87131: Dept, 9311, Sandia National Laboratory, Albuquerque, NM 87185 1058-6393/95 year 1995 IEEE.

[14] Thanos Makatos, Yannis Klonatos, Manolis Marazakis, Michail D. Flouris, and Angelos Bilas, *"ZBD: Using Transparent Compression at the Block Level to Increase Storage Space Efficiency"*, Foundation for Research and Technology – Hellas (FORTH), P.O. Box 2208, Heraklion, GR 71409, Greece, 978-07695-2/10, © 2010 IEEE.

[15] Ming-Bo Lin, Member, IEEE, and Yung-Yi Chang, *"A New Architecture of a Two-Stage Lossless Data Compression and Decompression Algorithm"*, 1063-8210, ©2009 IEEE.

[16] Ying Li and Khalid Sayood, *"Lossless Video Sequence Compression Using Adaptive Prediction"*, 1057-7149, © 2007 IEEE

[17] Sushil Kumar, Dr. Sarita S. Bhadauria, Dr. Roopam Gupta, "*A Digital Compression Scheme Using Delta and Differential Methods*", IJCA (0975-8887) Volume 25 – No.7, July 2011, page No. 18 – 25.

[18] Senthil Shanmugasundaram, Robert Lourdusamy, "*IIDBE: A Lossless Text Transform for Better Compression*" International Journal of Wisdom Based Computing, Vol. 1 (2), August 2011, Page No. 1 – 6.

[19] Tanakorn Wichaiwong, Kitti Koonsanit, Chuleerat Jaruskulchai, "*A Simple Approach to Optimized Text*

Compression's Performance" 4th International Conference on Web Services Practices, IEEE Computer Society, 978-0-7695-3455-8/08, Page no. 66 – 70.

[20] M. Baritha Begum, Dr. Y. Venkataramani, "*An Efficient Text Compression for Massive Volume of Data*" IJCA (0975 - 8887), Volume 21 – No. 5, May 2011, page No. 5 – 9.

[21] Md. Nasim Akhtar, Md. Mamunur Rashid, Md. Shafiqul Islam, Mohammod Abul Kashem, Cyrll Y. Kolybanov, "*Position Index Preserving Compression for Text Data*" JCS&T Vol. !! No. 1, April 2011, Page No. 9 – 14.

[22] S. R. Kodituwakku, U.S. Amarasinghe, "*Comparison of Lossless Data Compression Algorithms for Text Data*" IJCSE Vol. 1 No. 4 416-425, ISSN : 0976-5166, Page No. 416-4125.

[23] Rexline S. J, Robert L, "*Dictionary Based Preprocessing Methods in Text Compression – A Survey*" IJWBC, Vol. 1 (2), August 2011, Page No. 13-18.

[24] Umesh S. Bhadade, Prof. A. I. Trivedi, "*Lossless Text Compression using Dictionaries*" IJCA (0975 - 8887) Volume 13- No. 8, January 2011, Page No. 27-34.

[25] Haroon Altarawneh, Mohammad Altarawneh, "Data Compression Techniques on Text Files: A Comparison Study", IJCA (0975 - 8887) Volume 26- No.5, july2011.