# STUDY OF VARIOUS DATA COMPRESSION TOOLS

**Divya Singh[1], Vimal Bibhu[2], Abhishek Anand[3], Kamalesh Maity[4],Bhaskar Joshi[5]**

Senior Lecturer, Department of Computer Science and Engineering, AMITY University Greater Noida [1]
Assistant Professor, Department of Computer Science and Engineering, AMITY University Greater Noida [2]
B.Tech Scholar, Department of Computer Science and Engineering, AMITY University Greater Noida [3][4 ][5]

## Abstract

This paper focuses on the compression technique implemented by BZIP2 and its comparison to one of the existing compression techniques “GZIP” and “XZ”. This Paper also discusses the various algorithms used in BZIP2.The comparison between both the techniques include factors like Compressed file size upon comparison, compression time and decompression time. The techniques mentioned here are lossless.

***Keywords**:* Data Compression, Lossless, Lossy, BZIP2, GZIP, XZ, Burrows-Wheeler Transform, Huffman Coding, Deflate LZMA.

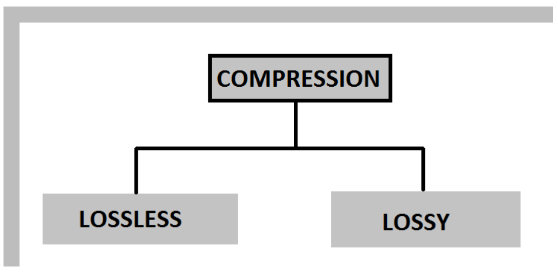
## 1. Introduction

Technology is changing and pacing every second. The output of today becomes the input of tomorrow. Everything needs to be short and precise and to do that we need short message or smaller data. To shorten the data, we need to compress the data. That’s how the idea of data compression came into existence.

Data compression means shortening the size of the data so as to reduce the disk size as well as data transmission becomes faster. Both are inter-related as once the data size reduces, the data that needs to be transmitted is short and hence it gets transmitted very fast. Basically, compression includes encoding the data in such a way that the redundant bits or letters which are occurring are removed upon compression.

Mainly, there are two types of data compression i.e, Lossy and Lossless data compression. When the redundant data bits are removed upon compression and regained after decompression, then this sort of compression is known as Lossless Data Compression while on the other hand if it is not recovered upon decompression then the compression is classified as Lossy Data Compression.

Examples of lossless data compression technique are “Text File Compression”. Here the letters in a text file upon decompression are not lost. Examples of lossy data compression techniques are image, video compression where few data is lost and is not regained upon decompression.



## 2. Introduction To BZIP2

BZIP2 is an open source advanced data compression program which uses Burrow-Wheeler Block-Sorting Text Compression Algorithm. It is not a file archiver instead is a single file compressor. Though bZip2 is a bit complicated than the other compression programs but we have researched that why it should be implemented especially in compression over the network.

BZIP2 compression tool is very efficient in compressing files as compared to the older algorithms like LZW and DEFLATE. It compresses the data in the blocks of 100-900 Kilobytes and then uses Burrows-Wheeler Transform algorithm to convert frequent occurring character sequences into string of identical letters.

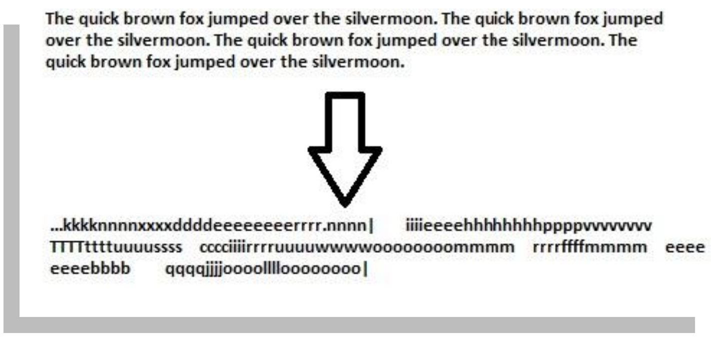
BZIP2 compress Heavy Files in Blocks.Block size affects the compression ratio and the amount of memory for compression and decompression.At Compression Time Block size stored in compressed file & at Decompression stage block size is used from header of the compressed file.

BZIP2 compressed files into blocks of 900kb long.Each Block is handled independently so if media error caused multi-block [.bz2] file to become damaged so due to block's of compressed file it may possible to recover data from undamaged clock in file.Each block is delimited by 48-bit pattern & each block ocntain own 32-bit CRC so damaged block is find out from undamaged block's.

### 2.1 BZIP2 Compression Process

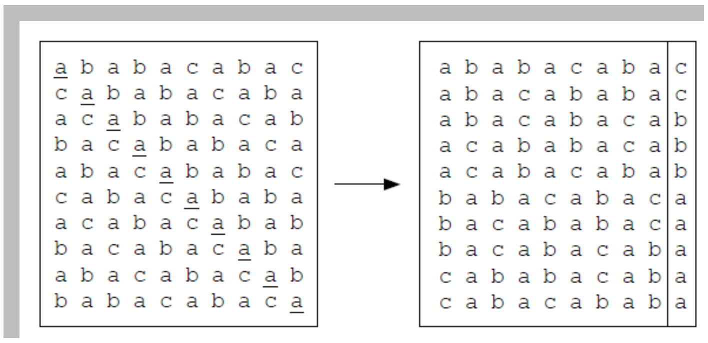
BZIP2 Generally a use following algorithms before compression to format data into blocks that can be compressed later and at the time of decompression, they follow reverse order.

1. Burrows–Wheeler transform (BWT): This algorithm takes an input of strings and rearranges them in such a way that most frequent letters assemble together so that the compression becomes very easy. The BWT begins by create a list of strings contain all cyclical rotations of original strings.Then List is sorted and last character of each rotation make a transformed string.Transformed string is formed by taking last column of the block. For better understanding, we take an example below:-



The quick brown fox jumped over the silvermoon. The quick brown fox jumped over the silvermoon. The quick brown fox jumped over the silvermoon. The quick brown fox jumped over the silvermoon.

...kkkknnnnxxxxddddeeeeeeeeerrrr.nnnn| iiiieeeehhhhhhhhppppvvvvvvvv TTTTttttuuuussss ccccciiiirrrruuuuwwwwoooooooommmm rrrrffffmmmm eeee eeeebbbb qqqqjjjjoooollllooooooooo|

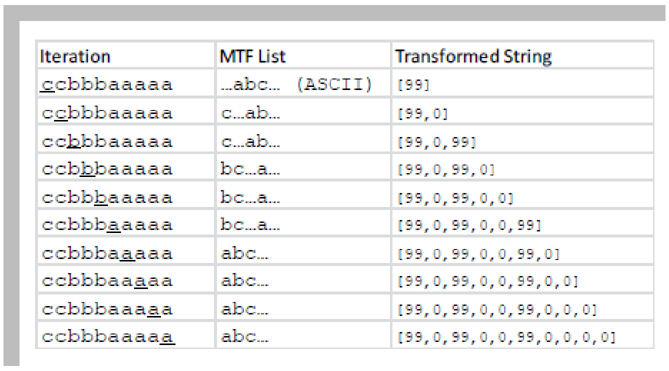


```
a b a b a c a b a c          a b a b a c a b a|c
c a b a b a c a b a          a b a c a b a b a|c
a c a b a b a c a b          a b a c a b a c a|b
b a c a b a b a c a          a c a b a b a c a|b
a b a c a b a b a c    -->   a c a b a c a b a|b
c a b a c a b a b a          b a b a c a b a c|a
a c a b a c a b a b          b a c a b a b a c|a
b a c a b a c a b a          b a c a b a c a b|a
a b a c a b a c a b          c a b a b a c a b|a
b a b a c a b a c a          c a b a c a b a b|a
```

Working of Burrows-Wheeler includes two Steps:-

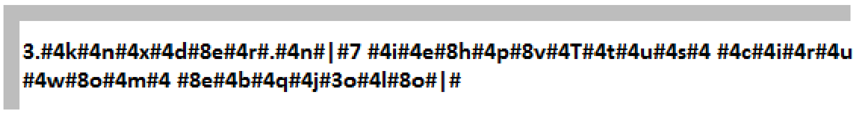
- Arrange the text into circular Array and arrange them using BWT delimiter.
- Block-sort the arranged array into lexicographical order.

2. MOVE-TO-FRONT Transform : The Move-to-front transform apply to improves effectivness of entropy encoding algorithm.MTF keep the recently used characters at front of the list.a new sequence output is generated of small number with more repeats.

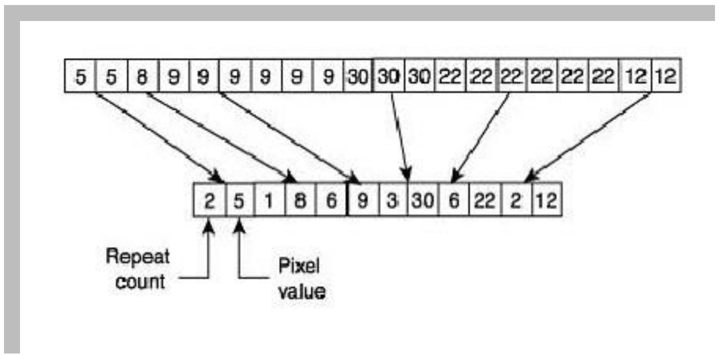


| Iteration | MTF List | Transformed String |
|---|---|---|
| ccbbbaaaaa | ...abc... (ASCII) | [99] |
| ccbbbaaaaa | c...ab... | [99,0] |
| ccbbbaaaaa | c...ab... | [99,0,99] |
| ccbbbaaaaa | bc...a... | [99,0,99,0] |
| ccbbbaaaaa | bc...a... | [99,0,99,0,0] |
| ccbbbaaaaa | bc...a... | [99,0,99,0,0,99] |
| ccbbbaaaaa | abc... | [99,0,99,0,0,99,0] |
| ccbbbaaaaa | abc... | [99,0,99,0,0,99,0,0] |
| ccbbbaaaaa | abc... | [99,0,99,0,0,99,0,0,0] |
| ccbbbaaaaa | abc... | [99,0,99,0,0,99,0,0,0,0] |

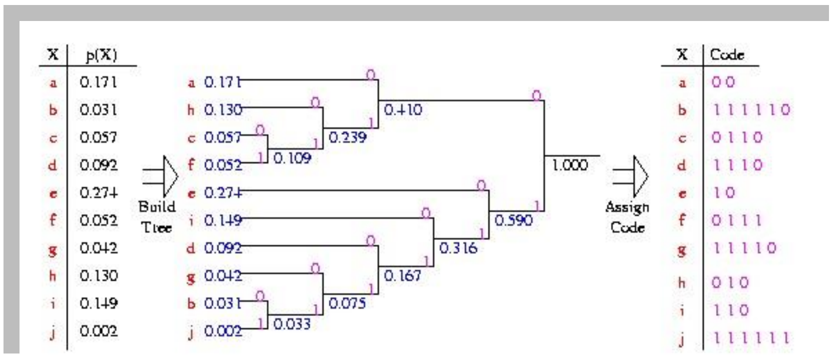
3. Run-length encoding (RLE): Once the sorting is done, the sorted string is encoded according to the frequency of the letters. This can be understood by the following example.

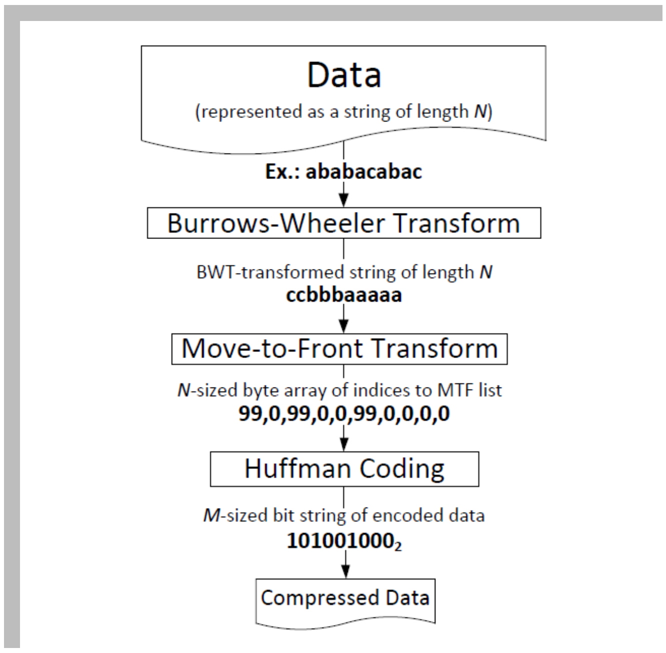


**3.#4k#4n#4x#4d#8e#4r#.#4n#|#7 #4i#4e#8h#4p#8v#4T#4t#4u#4s#4 #4c#4i#4r#4u #4w#8o#4m#4 #8e#4b#4q#4j#3o#4l#8o#|#**



4. Huffman coding: The run length encoded text generated in the previous step is taken as input and a Huffman Tree is created using the Huffman coding and the letters are encoded into their individual binary codes. The letter with minimum frequency gets the highest binary code and the letter with the maximum, gets the lowest.




Data
(represented as a string of length N)
Ex.: ababacabac
Burrows-Wheeler Transform
BWT-transformed string of length N
ccbbbaaaaa
Move-to-Front Transform
N-sized byte array of indices to MTF list
99,0,99,0,0,99,0,0,0,0
Huffman Coding
M-sized bit string of encoded data
101001000₂
Compressed Data

## 3. Introduction To GZIP

GZIP is one of the most advanced compressor program which replaced the older "compress" program in unix systems to achieve compression at a faster rate.

The Algorithms Used in GZIP Are :-
• DEFLATE: - This algorithm is the combination of LZ77 and Dynamic Huffman coding.

GZIP uses LZ77 algorithm and dynamic Huffman algorithm to compress data. first Gzip use LZ77 algorithm to compress data , then use dynamic Huffman algorithm to compress the result.GZIP program has an excellent integration with unix-files and it has ".gz" extension(file format).

## 4. Introduction To XZ

"XZ" is in the series with one of the best compression programs like "7-ZIP" which uses LZMA algorithm to compress the data. It is also a lossless data compressor.
Although, it is similar to "7-ZIP" program which uses LZMA2 algorithm but "XZ" has an extra edge over the 7-ZIP as it has integration with unix-file like "metadata".
This compressor program has its own file format which is ".xz" and it takes single file as input and not the multiple files. The most significant part of this compression program is that it can compress "tar" (archived) files too.

## 5. Compression & Decompression Analysis

In the following section we are analysing compression / decompression ratio of all the mentioned compression program.We using two different Processor to compress three different file and record compression size / decompression time / cpu-usage.

### 5.1 Test Condition

| | | |
|---|---|---|
| Processor | Intel Core i7-3770 | Intel Core 2 Duo T6500 |
| Architecture | x86 | x86 |
| Platform | Kalinux 1.0.6 | Kalinux 1.0.4 |

**Process :-**

1.First we Take Three Different Files with Filesize 1M , 10M , 100M
Text File contain Garbage Random Repeated Data.

2.Compress all files and analyze the time / compressed size / cpu usage / decompression time

**Results :-**

Once the Test is Over, We get the Following Results.

**size - filesize**

**ctime - compression time**

**csize - compressed size**

**dtime - decompression time**

**c-cpu - cpu usage during compression**

Note: Processor { Intel Core 2 Duo } - *Kalinux 1.0.4 x86*

| Algo | size[MB/KB] | ctime[s] | csize[kb] | dtime[s] | c-cpu[%] |
|---|---|---|---|---|---|
| bzip2 | 1.0/1048576 | 0.02 | 0.06 | 0.00 | 93 |
| gzip | 1.0/1048576 | 0.00 | 1.3 | 0.01 | 80 |
| xz | 1.0/1048576 | 0.17 | 0.296 | 0.00 | 99 |

| Algo | size[MB/KB] | ctime[s] | csize[kb] | dtime[s] | c-cpu[%] |
|---|---|---|---|---|---|
| bzip2 | 10.5/10485760 | 0.25 | 0.072 | 0.04 | 98 |
| gzip | 10.5/10485760 | 0.12 | 12.7 | 0.06 | 100 |
| xz | 10.5/10485760 | 1.47 | 1.7 | 0.02 | 99 |

| Algo | size[MB/KB] | ctime[s] | csize[kb] | dtime[s] | c-cpu[%] |
|---|---|---|---|---|---|
| bzip2 | 104.9/104857600 | 2.58 | 0.113 | 0.44 | 99 |
| gzip | 104.9/104857600 | 1.07 | 101.8 | 0.66 | 99 |
| xz | 104.9/104857600 | 14.23 | 15.4 | 0.55 | 99 |

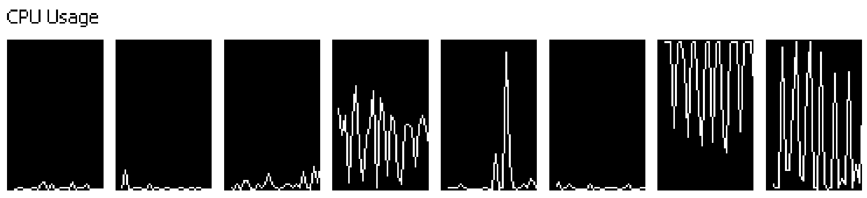
Note: Processor { Intel Core i7 } - *Kalinux 1.0.6 x86*

| Algo | size[MB/KB] | ctime[s] | csize[kb] | dtime[s] | c-cpu[%] |
|---|---|---|---|---|---|
| bzip2 | 1.0/1048576 | 0.02 | 0.06 | 0.00 | 54 |
| gzip | 1.0/1048576 | 0.01 | 1.3 | 0.00 | 88 |
| Xz | 1.0/1048576 | 0.08 | 0.296 | 0.00 | 89 |

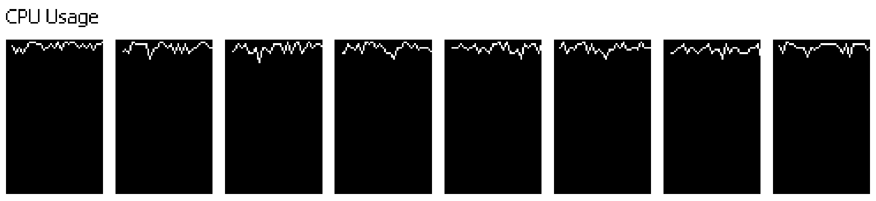
| Algo | size[MB/KB] | ctime[s] | csize[kb] | dtime[s] | c-cpu[%] |
|---|---|---|---|---|---|
| bzip2 | 10.5/10485760 | 0.12 | 0.072 | 0.02 | 97 |
| gzip | 10.5/10485760 | 0.06 | 12.7 | 0.03 | 93 |
| xz | 10.5/10485760 | 0.70 | 1.7 | 0.02 | 98 |

| Algo | size[MB/KB] | ctime[s] | csize[kb] | dtime[s] | c-cpu[%] |
|---|---|---|---|---|---|
| bzip2 | 104.9/104857600 | 1.02 | 0.113 | 0.40 | 98 |
| gzip | 104.9/104857600 | 0.60 | 101.8 | 0.55 | 98 |
| xz | 104.9/104857600 | 6.77 | 15.4 | 0.50 | 98 |

BZIP2 Compression Performance Over **Dual Quad-Core Xenon Processor**.When we compress the Large File by using BZIP2 on **Dual Quad-Core Xenon Processor**. We find that BZIP uses more than two cores to parallelize the work.



**CPU Usage – 7ZIP**



**CPU Usage – BZIP2**

## 6. Conclusion

From the above sets of results, we have concluded that bzip2 program has the best compression ratio (size after compression/size before compression). Though gzip has the least compression time but the compression size is greater than bzip2.

## References

[1] M. Burrows and D. J. Wheeler. A Block-Sorting Lossless Data Compression Algorithm. Technical Report 124, 1994.

[2] Brenton Chapin and Stephen R. Tate. Higher Compression from the Burrows-Wheeler Transform by Modified Sorting. In Data Compression Conference.

[3] P. Fenwick. Block-Sorting Text Compression — Final Report, 1996.

[4] David A. Huffman. A method for the construction of Minimum-Redundancy Codes — Proceedings of the Institute of Radio Engineers.

[4] David A. Huffman. A method for the construction of Minimum-Redundancy Codes — Proceedings of the Institute of Radio Engineers

[5] T.A.Welch. A Technique for High Performance Data Compression. IEEE Computer, Vol. 17, No. 6, June 1984.

[6] Giovanni Manzini. Wheeler. The Burrows-Wheeler Transform:Theory and Practice.

[7] JUERGEN ABEL. Improvements to the Burrows Wheeler Compression Algorithm: After BWT Stages.

[8] Jagadish H. Pujar,Lohit M. Kadlaskar. A New Lossless Method of Image Compression and Decompression Using Huffman Coding Techniques.

**1st Divya Singh –** Senior Lecturer at Amity University Greater Noida ( Email Address : divya1784@rediffmail.com )

**2nd Vimal Bibhu –** Assistant Professor at Amity University Greater Noida.

**3rd Abhishek Anand –** B.Tech Scholar at Amity University Greater Noida ( Email Address : abhi9312@gmail.com).

**4th Kamalesh Maity –** B.Tech Scholar at Amity University Greater Noida ( Email Address : mail.maity25@gmail.com).

**5th Bhaskar Joshi –** B.Tech Scholar at Amity University Greater Noida ( Email Address : bhaskarj078@gmail.com).