LZ4 Compression Library

Zhe Zhang

Agenda

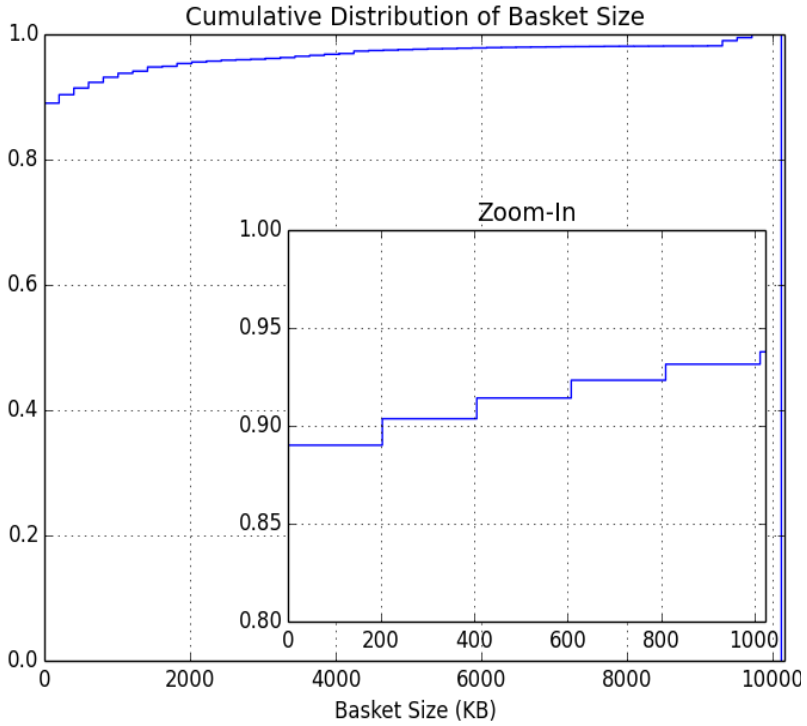
- LZ4 on a CMS file
- LZ4 on dummy files

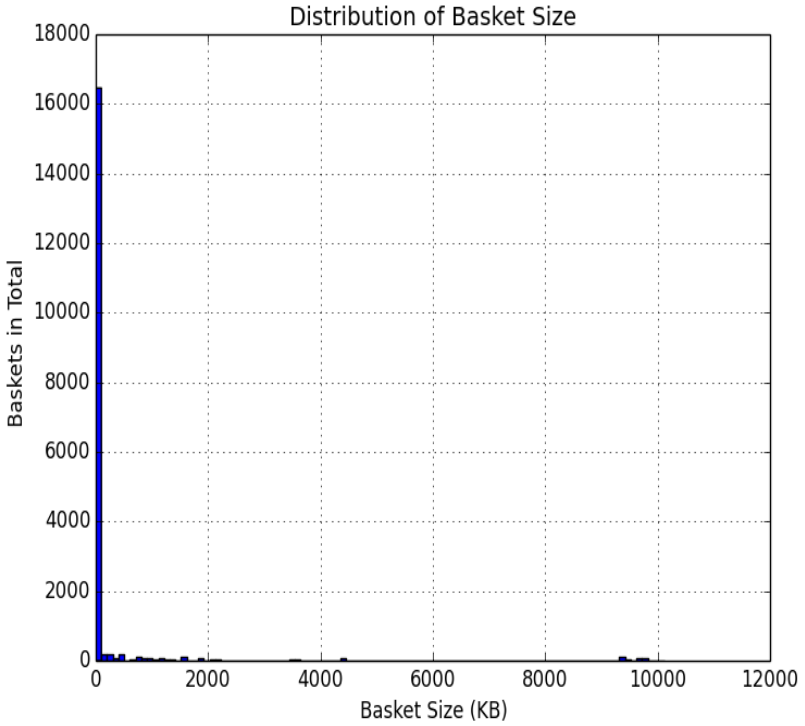
Agenda

- LZ4 on a CMS file
- LZ4 on dummy files

Distribution of Block Size

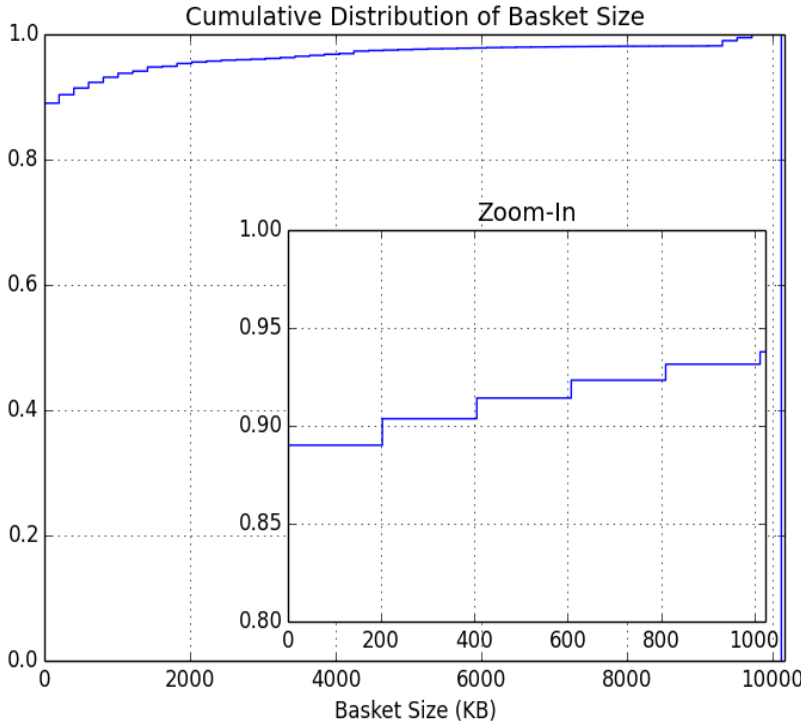
CMS file:
https://root.cern.ch/files/CMS_7250E9A5-682D-DF11-8701-002618943934.root

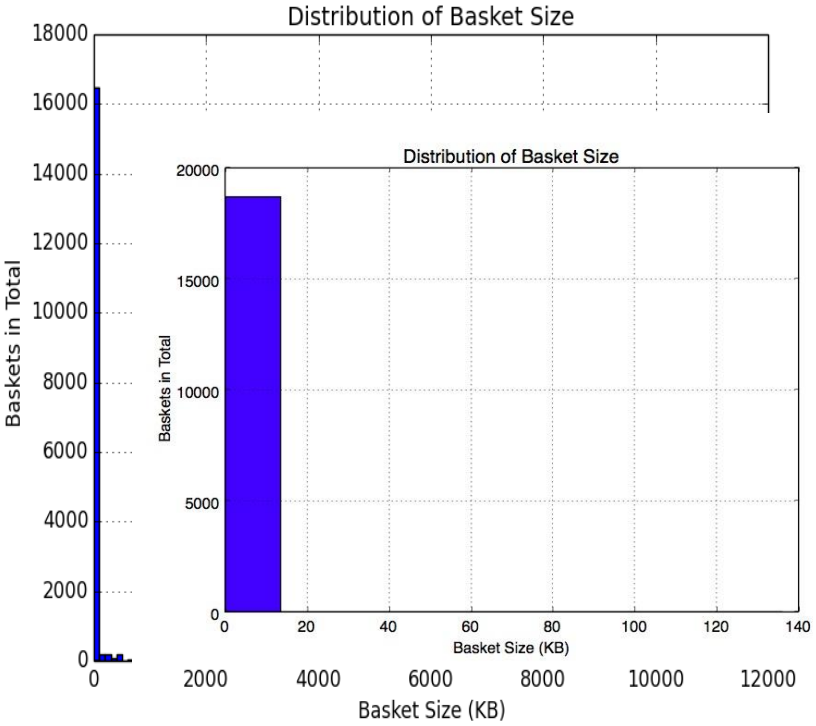




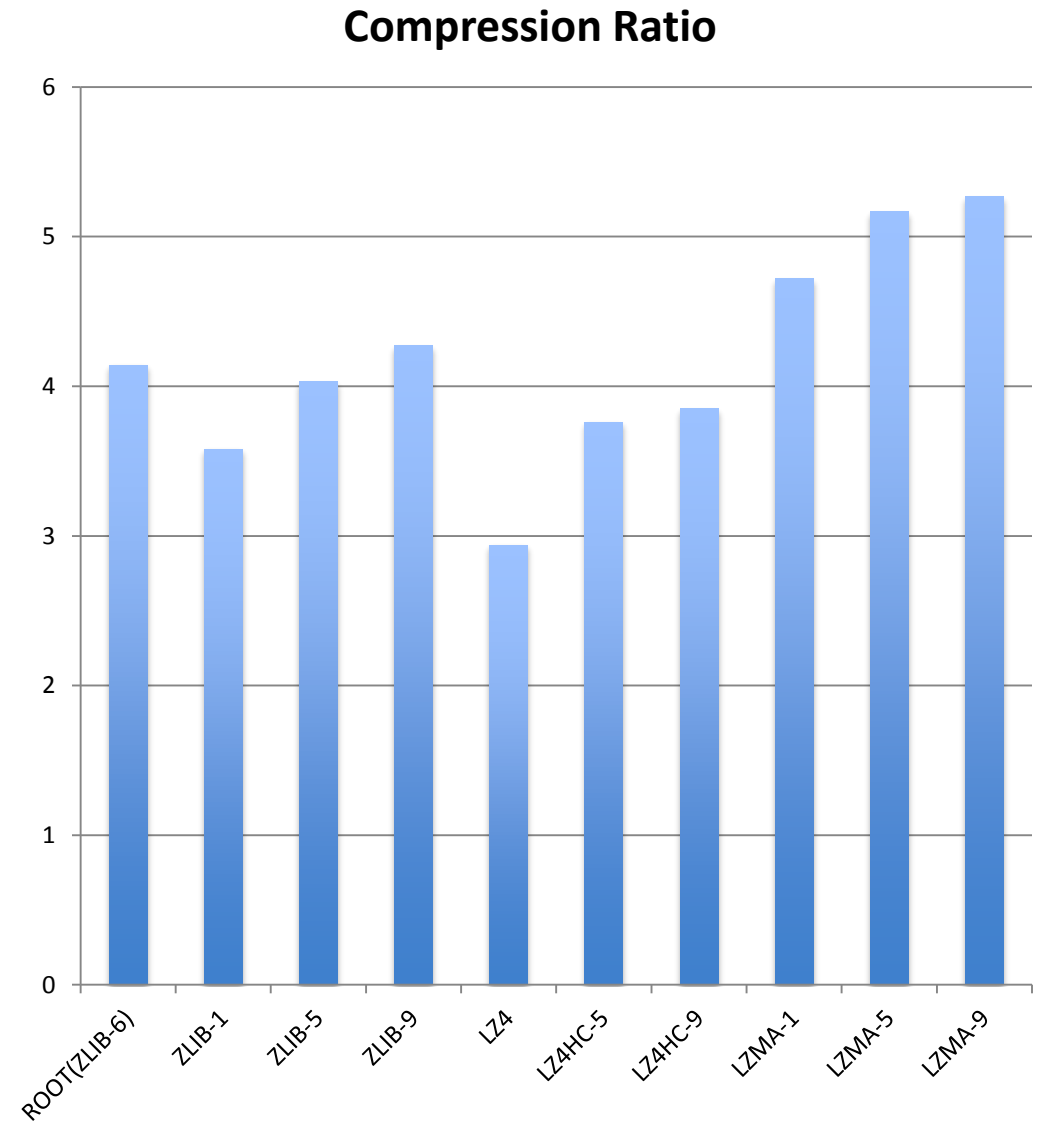
Distribution of Block Size

Most of baskets in this CMS file: < 20 KB



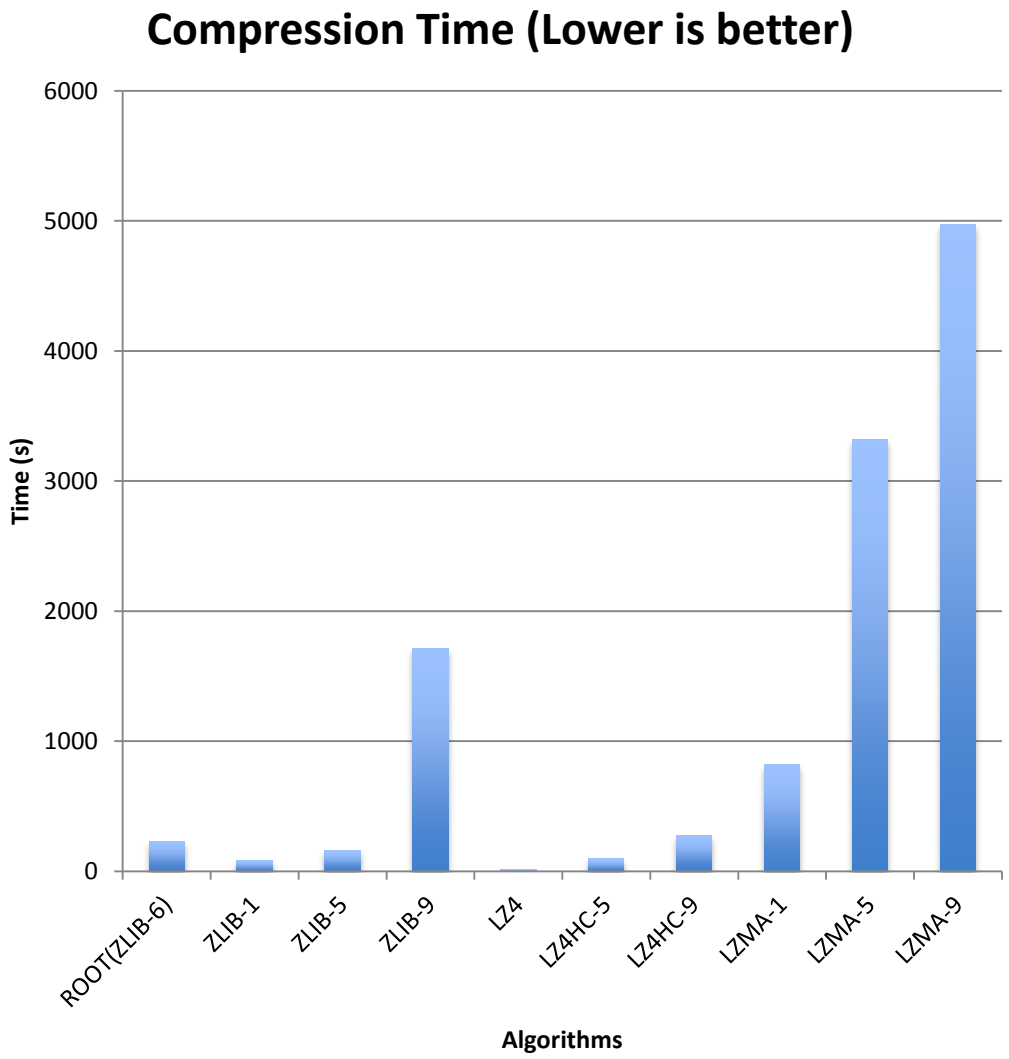


Compression Ratio



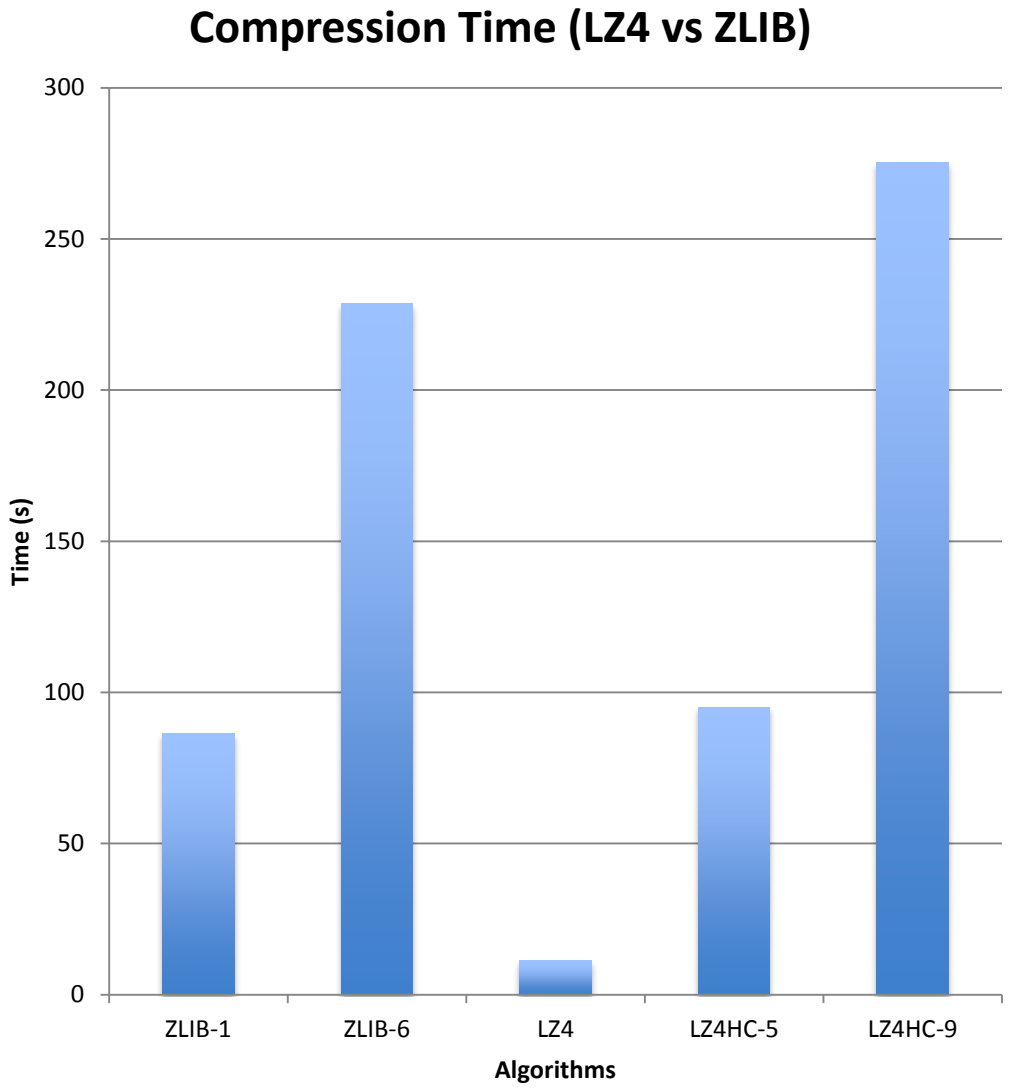
- LZMA has highest compression ratios at all levels
- LZ4HC-5 and LZ4HC-9 sit between Zlib-1 and Zlib-6(ROOT)

Compressing Time(I)



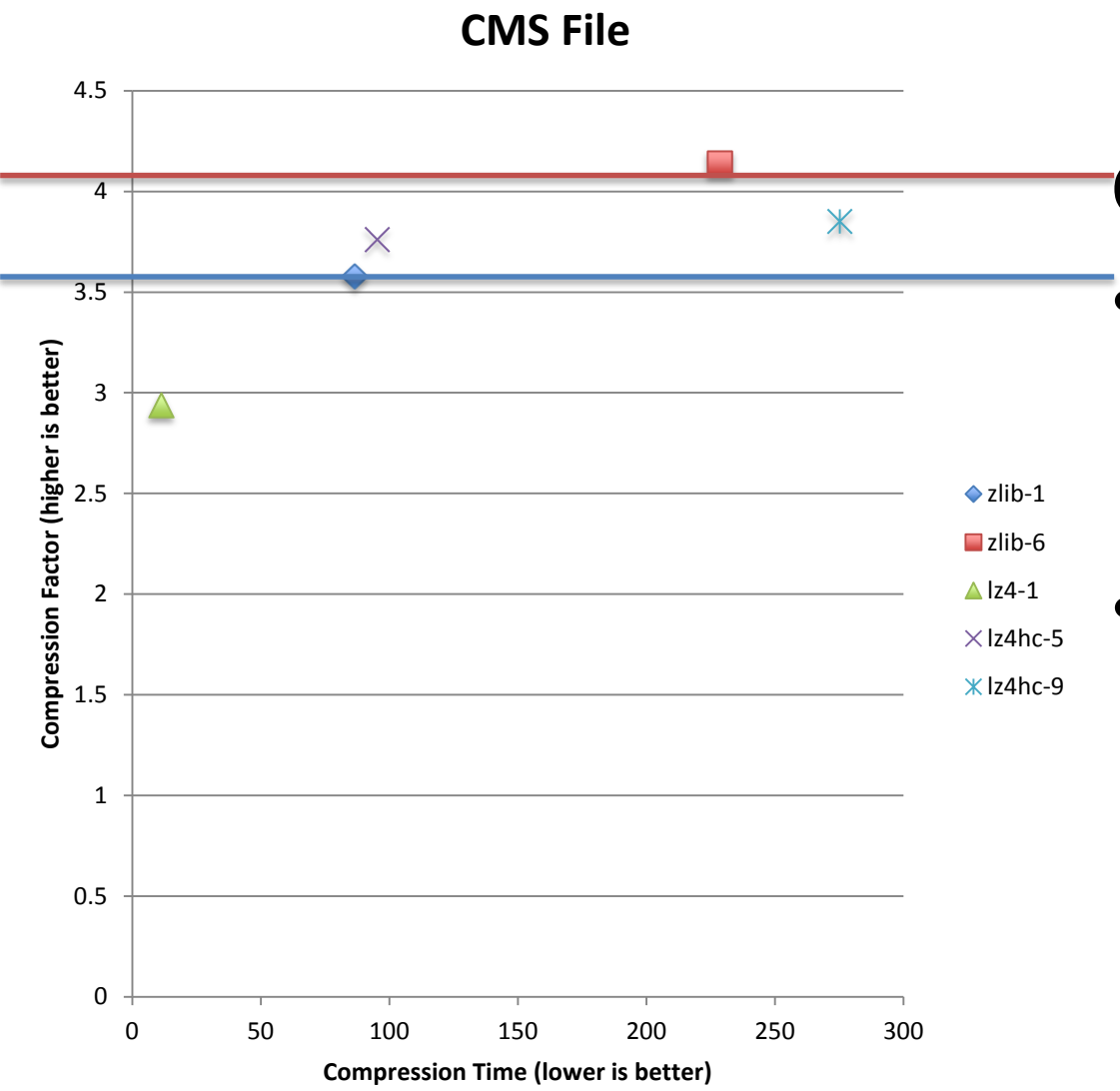
- LZMA took long time to compress data
- LZ4 is faster than ZLIB at same compression level

Compressing Time(II)



- Compression speed of ROOT(Zlib-6) is between LZHC-5 and LZHC-9

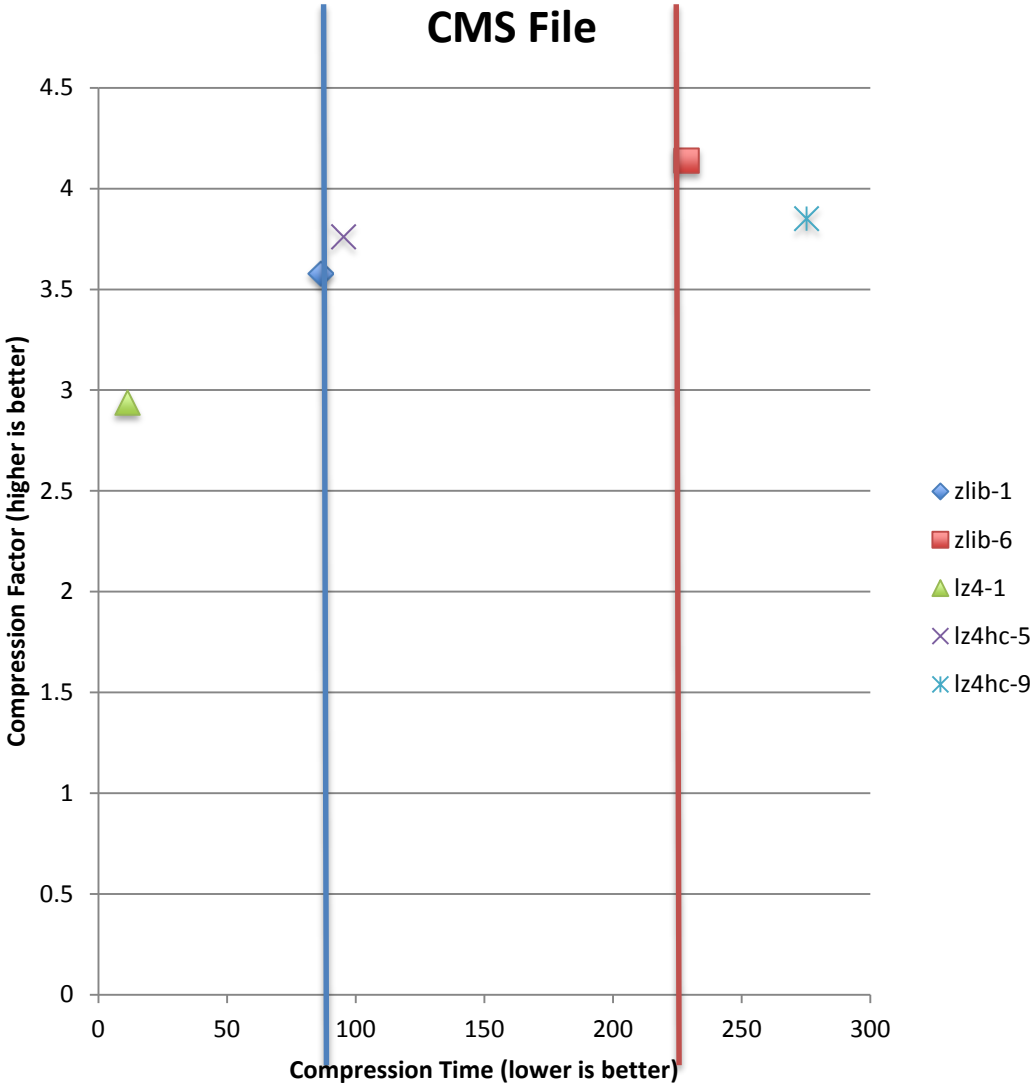
Compressing Time vs. Compression Factor(I)



Compression factor:

- LZ4HC-5 and LZ4HC-9 fall between Zlib-1 and Zlib-6(ROOT)
- LZ4 has lower compression factor than others

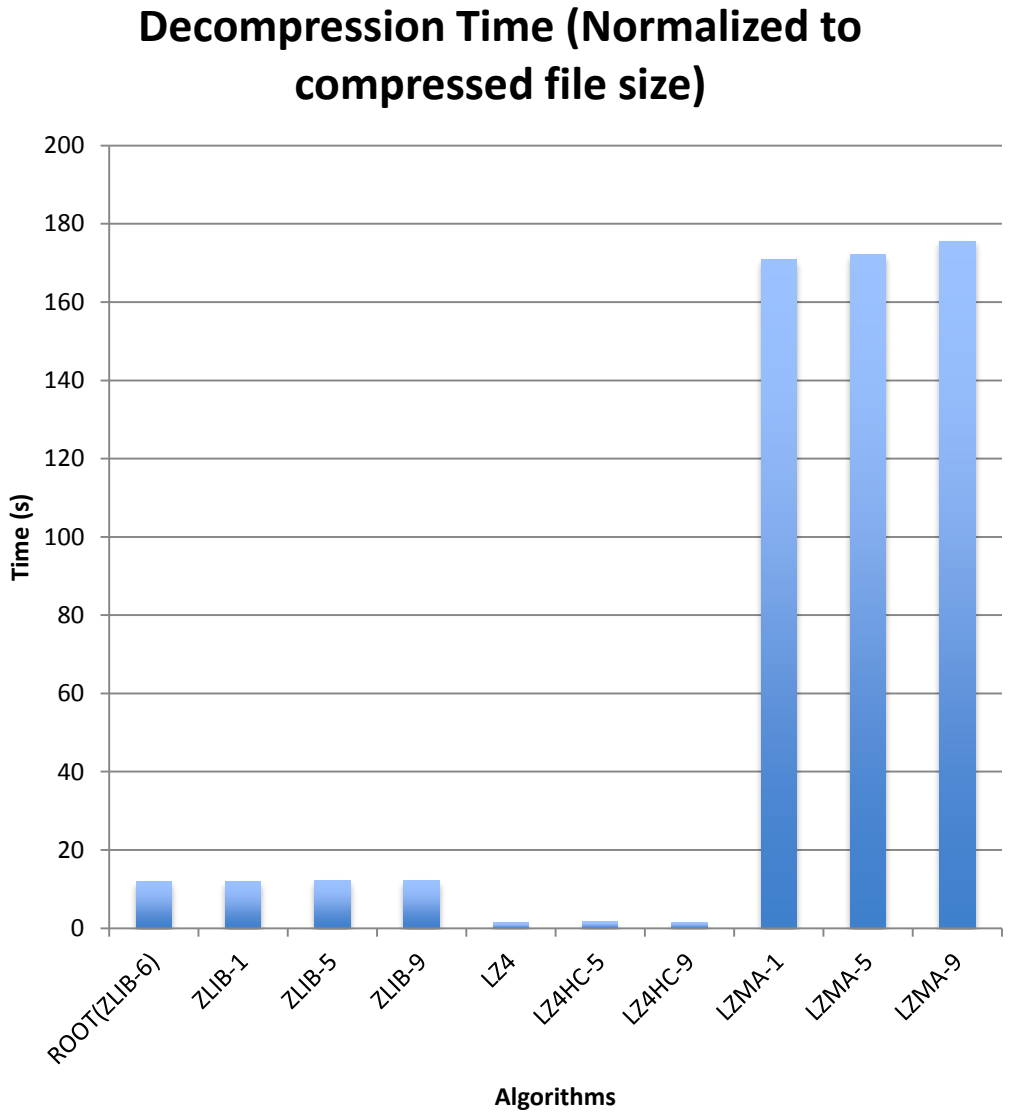
Compressing Time vs. Compression Factor(II)



Compression Time:

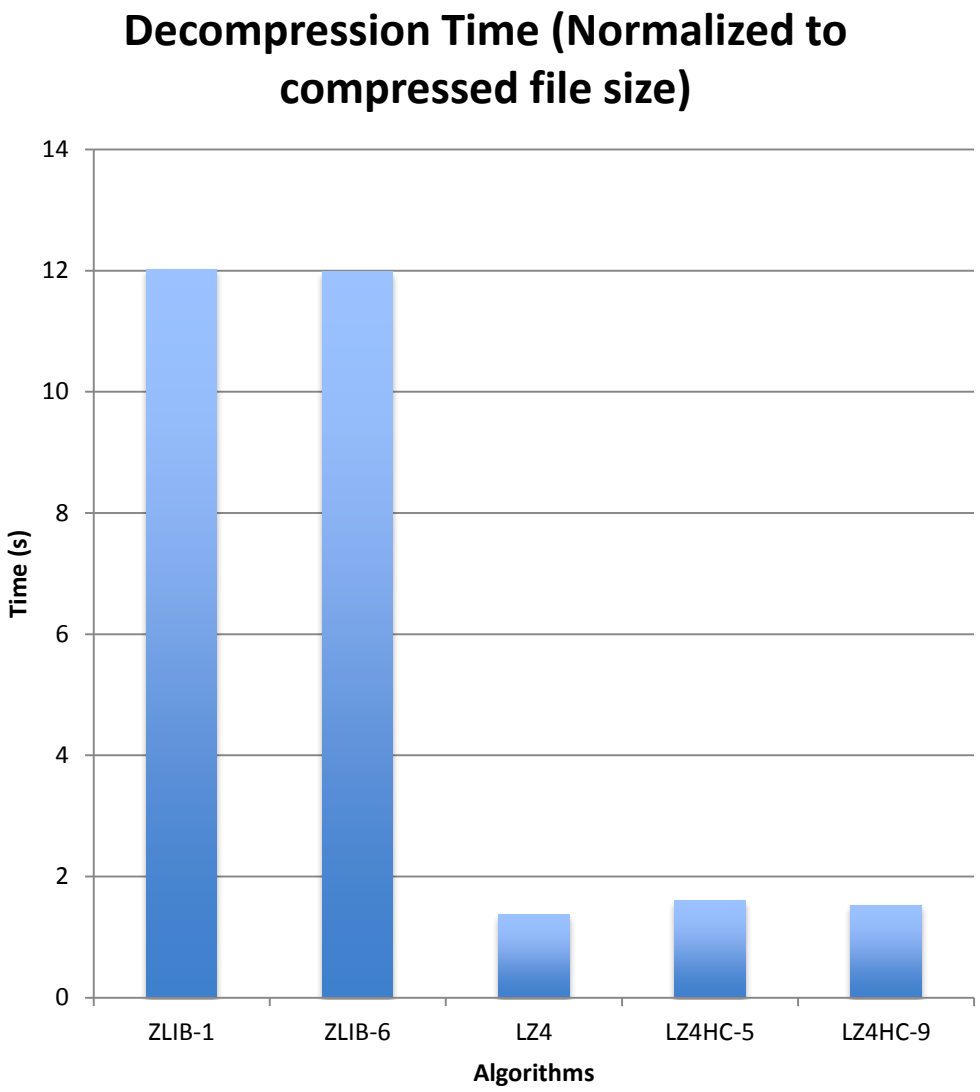
- Compression speed of ROOT(Zlib-6) is between LZHC-5 and LZHC-9
- Zlib-1 is between LZ4 and LZ4HC-5

Decompressing Time(I)



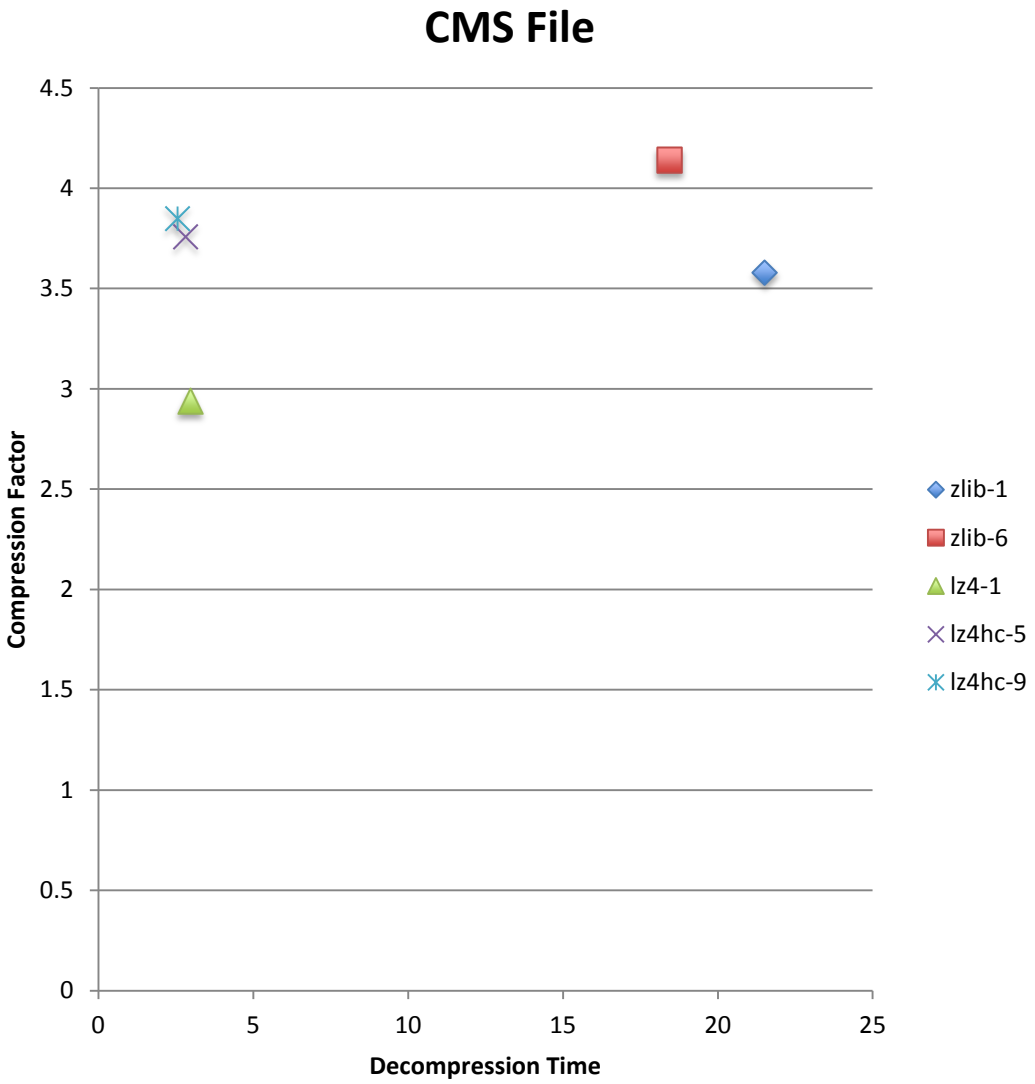
- LZMA are slowest at all compression levels
- LZ4 are fastest at all compression levels

Decompressing Time(II)



- LZ4 is 4-7 times faster than ROOT(Zlib-6)

Decompressing Time(III)



- LZ4 related algorithms are located away from Zlib
- LZ4HC-5 and LZ4HC-9 are faster while maintaining good compression factors

Agenda

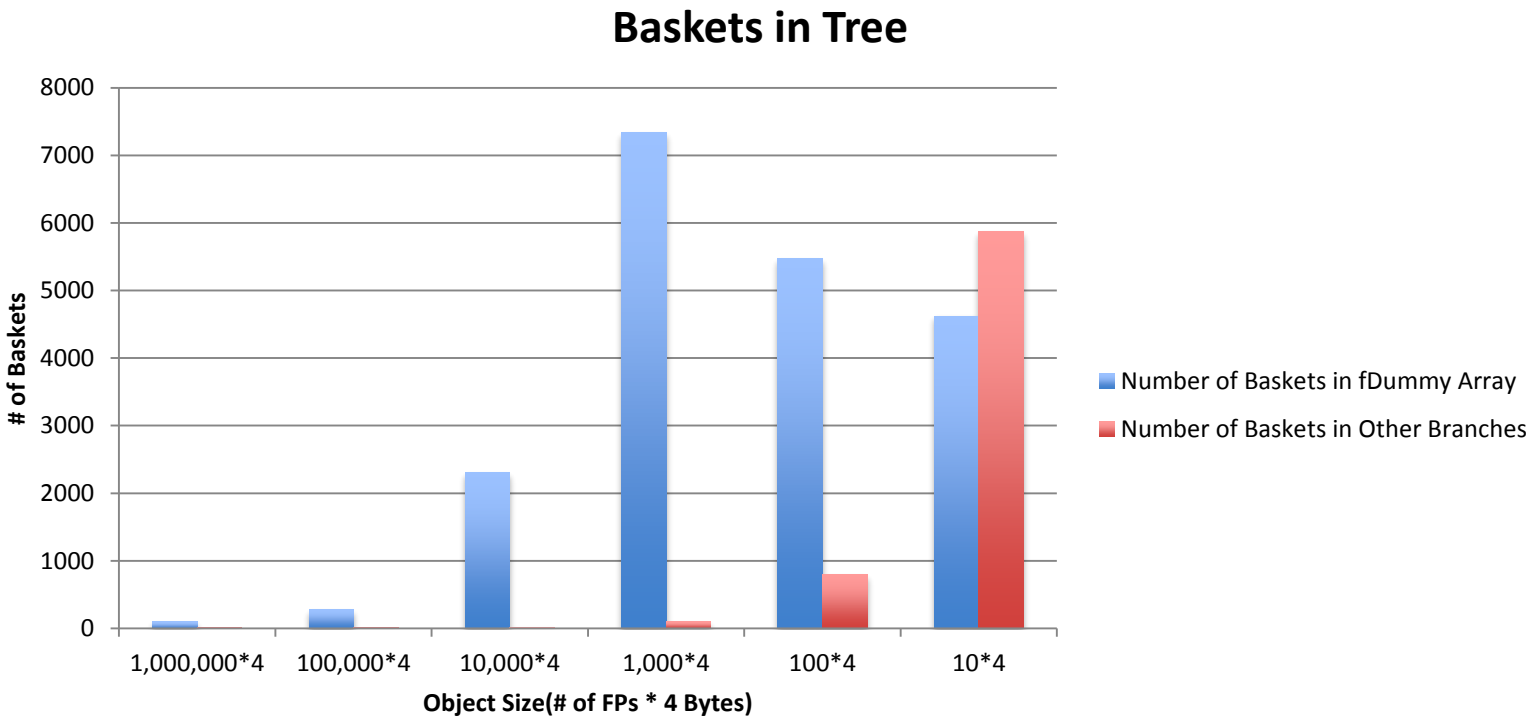
- LZ4 on a CMS file
- LZ4 on dummy files

Test Setup

- Each dummy object contains multiple FPs
- Dummy object's size is ranging from 40 B to 4 MB
- All tests contain equal amount of object data

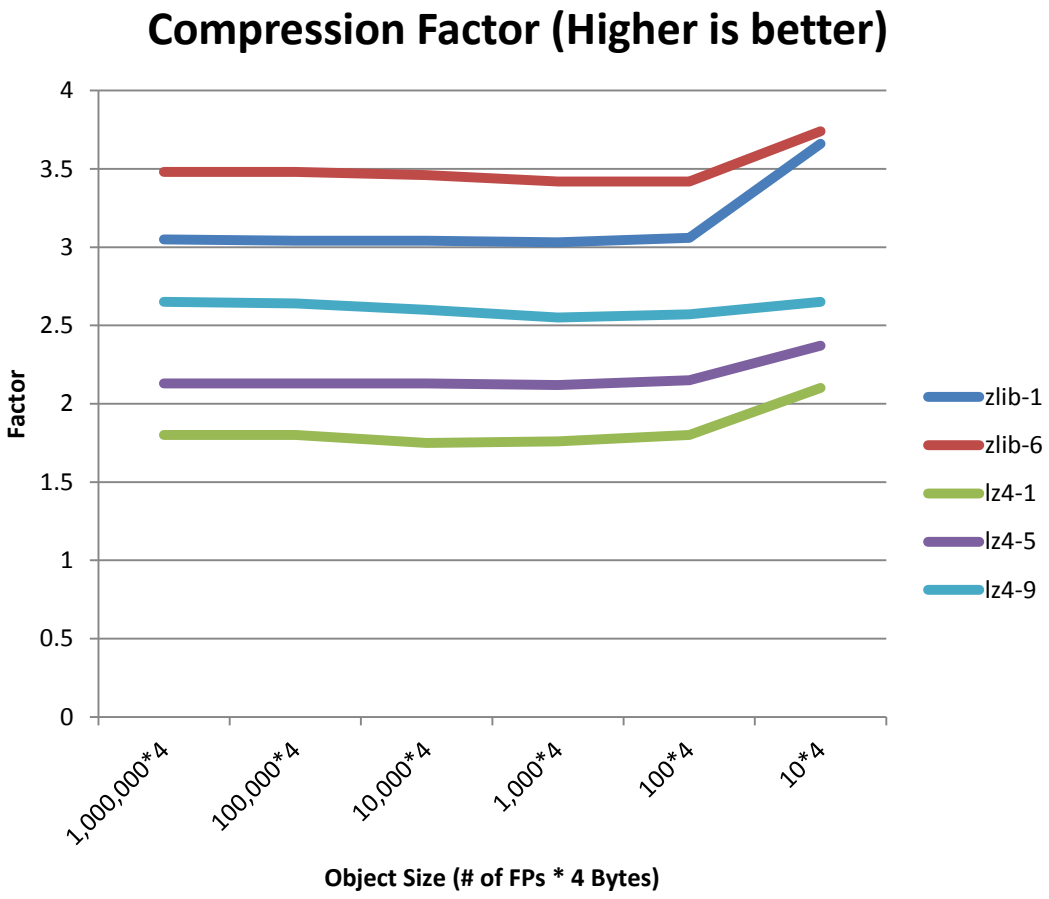
e.g.
Each object size = 4 MB, # of object = 100
Each object size = 400 KB, # of object = 1,000
Each object size = 40 KB, # of object = 10,000
Each object size = 4 KB, # of object = 100,000
Each object size = 400 B, # of object = 1,000,000
Each object size = 40 B, # of object = 10,000,000

Distribution of Baskets



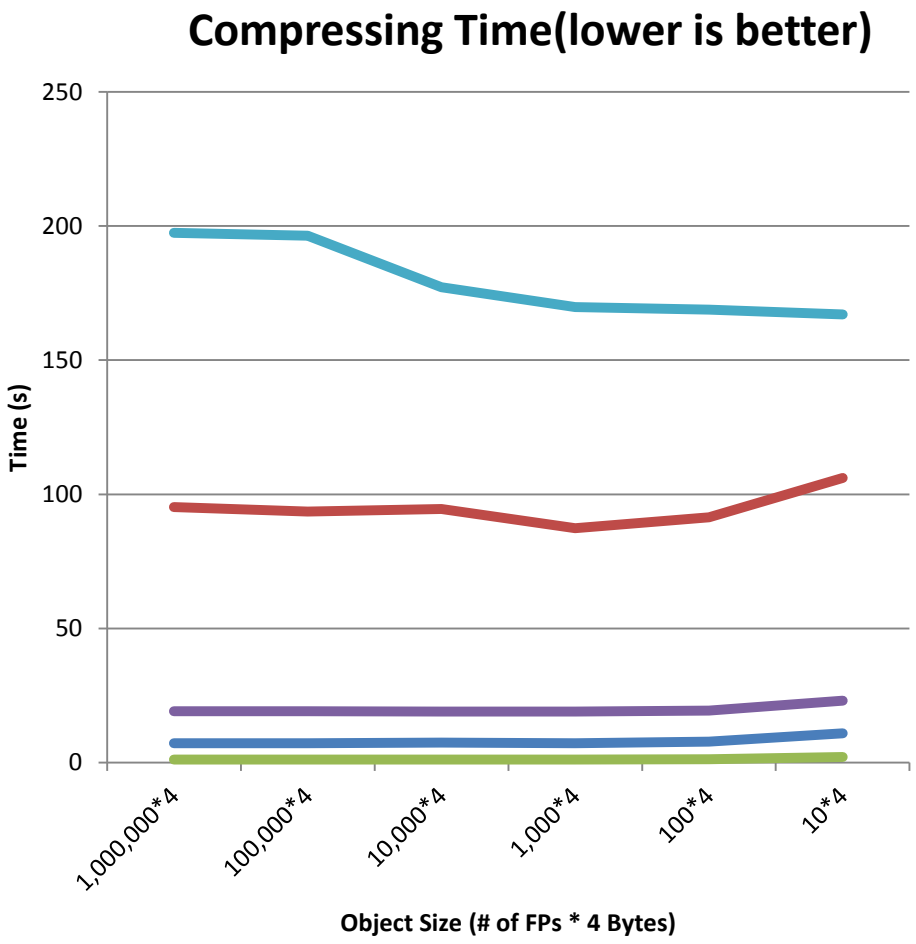
- Most of data are stored in the branch of fDummy array, too large or too small objects generate less baskets and thus more storage efficient
- Smaller objects generate more baskets in other branches (etc. fRefTable, fSize)

Compression Factor



- LZ4 is not storage efficient comparing to ZLIB

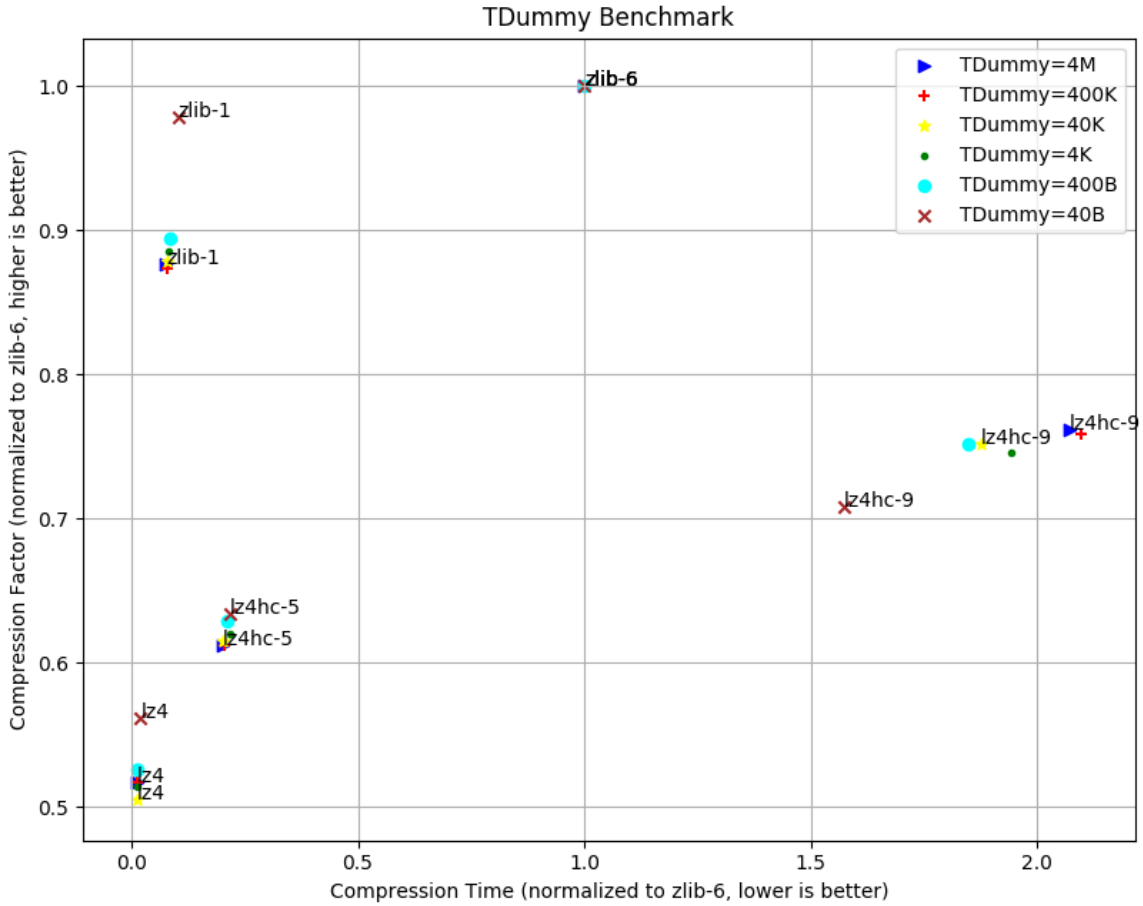
Compressing Time



- Compressing Time:

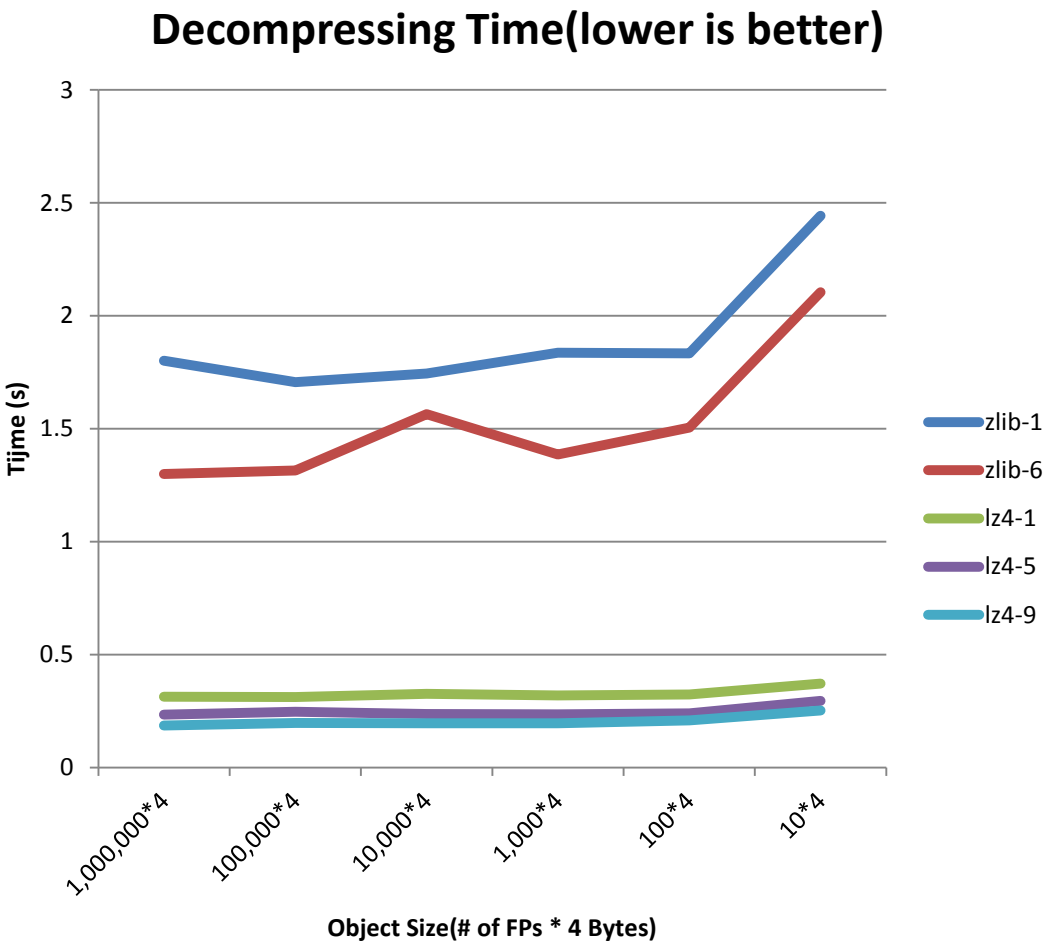
LZ4-9 > ZLIB-6 > LZ4-5 > ZLIB-1 > LZ4-1

Compression Time vs. Compression Factor



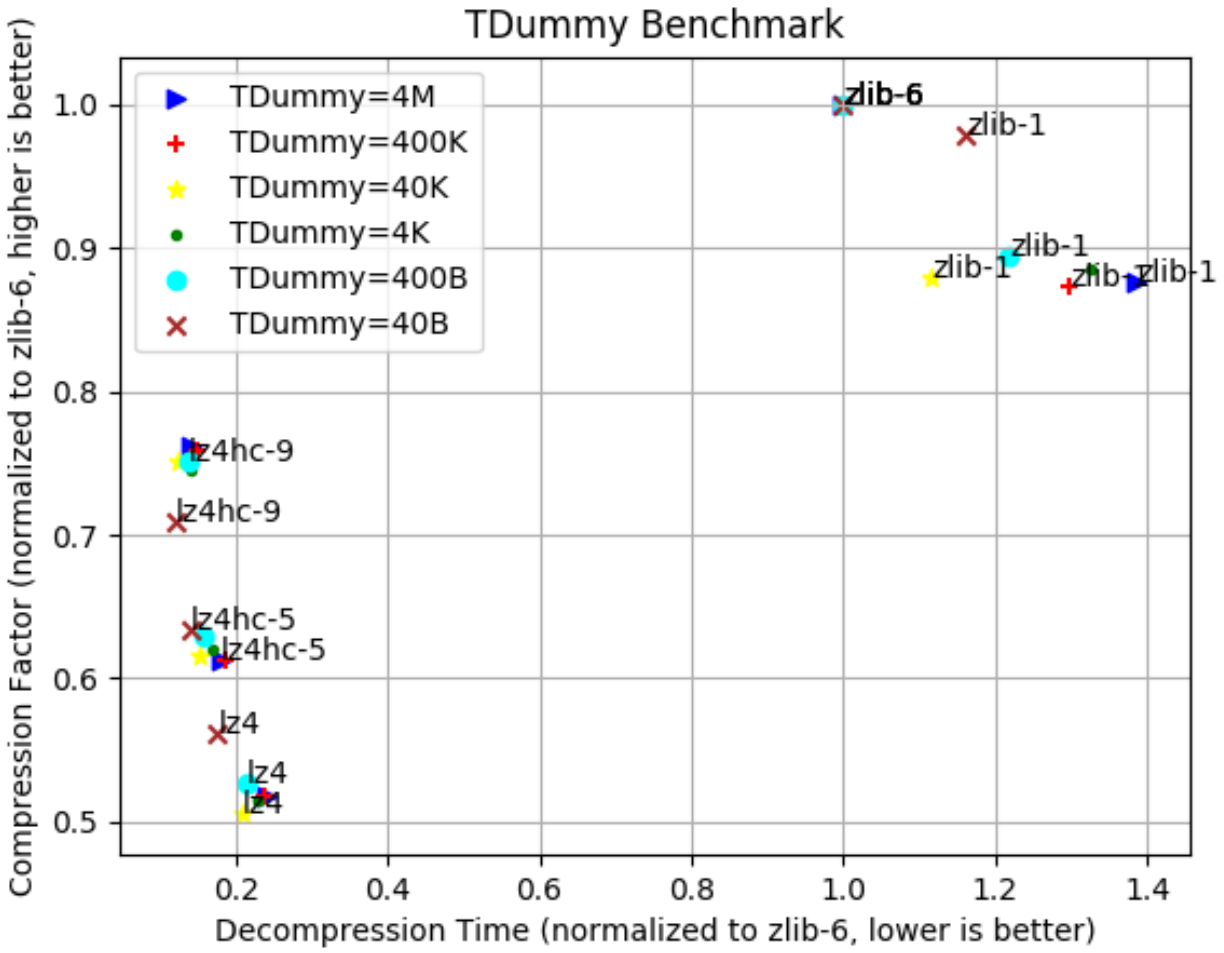
- LZ4 and LZ4HC-5 have fast compression speed but low compression ratio
- Zlib-1 are good on both compression speed and compression ratio

Decompressing Time



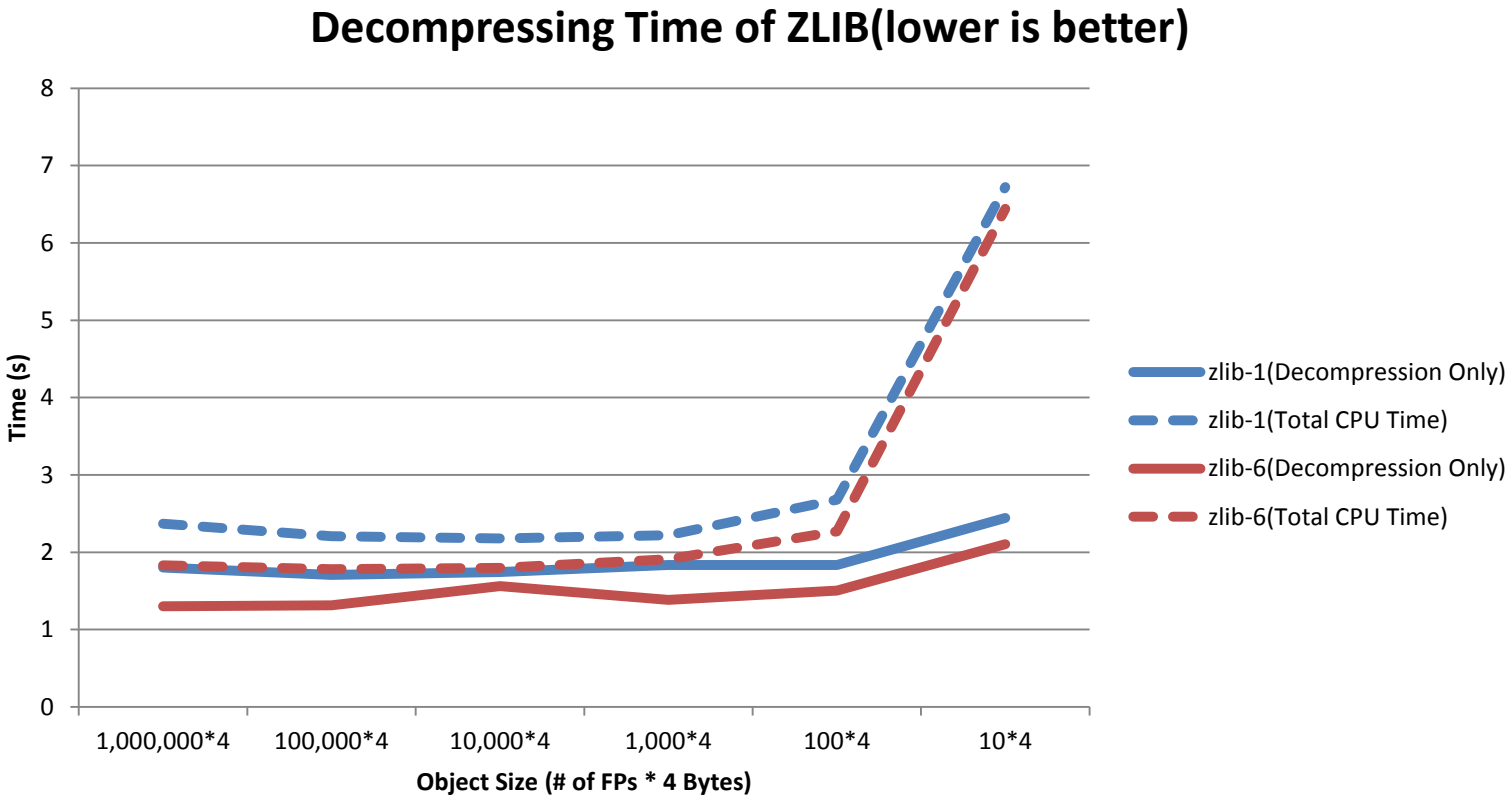
- LZ4 outperforms ZLIB at all levels.

Decompression Time vs. Compression Factor



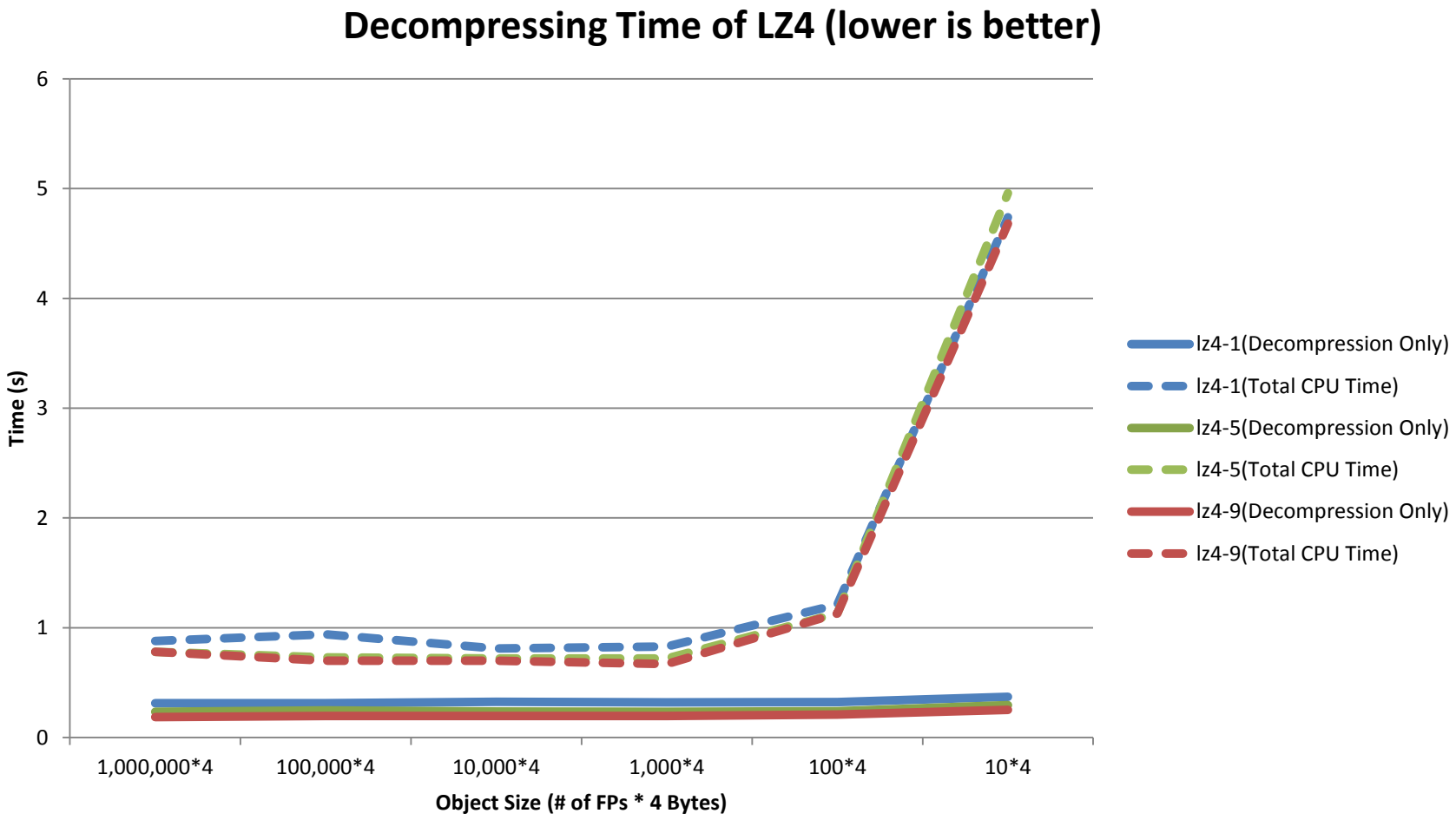
- LZ4 has better decompression speed but lower compression ratio

Decompressing Time(ZLIB)



- A majority of CPU cycles are spent on doing decompression
- Still a significant portion are doing something else(etc. deserialization)
- As object size becomes tiny, more CPU cycles are consumed by other stuff

Decompressing Time(LZ4)



- Since LZ4 has faster decompressing speed, other work (etc. deserialization) seems to contribute more

Conclusions

- Compression Time:
  - LZ4-9 > ZLIB-6 > LZ4-5 > ZLIB-1 > LZ4-1
- Decompression Time:
  - LZ4 outperforms ZLIB
- Compression Ratio:
  - For large baskets, Zlib has higher compression ratio than LZ4

Appendix: Basket Sizes

Object Size = 4M

```
*Tree    :T         : An example of a ROOT tree
*Entries :      100 : Total =       400018327 bytes
*        :          : Tree compression factor =   3.
*Branch  :dummy
*Entries :      100 : BranchElement (see below)
*Br    0 :fUniqueID : UInt_t
*Entries :      100 : Total  Size=       1283 bytes
*Baskets :        5 : Basket Size=      11771 bytes
*Br    1 :fBits     : UInt_t
*Entries :      100 : Total  Size=       1687 bytes
*Baskets :        5 : Basket Size=      11771 bytes
*Br    2 :fSize     : Int_t
*Entries :      100 : Total  Size=       1247 bytes
*Baskets :        5 : Basket Size=      11771 bytes
*Br    3 :fDummy    : Float_t fDummy[fSize]
*Entries :      100 : Total  Size=  400011250 bytes
*Baskets :      100 : Basket Size=    4000084 bytes
*Br    4 :TRefTable : List of branch numbers with re
*Entries :      100 : Total  Size=       1402 bytes
*Baskets :        0 : Basket Size=      32000 bytes
```

Object Size = 400K

```
*Tree    :T         : An example of a ROOT tre
*Entries :     1000 : Total =       400062073
*        :          : Tree compression factor
*Branch  :dummy
*Entries :     1000 : BranchElement (see below
*Br    0 :fUniqueID : UInt_t
*Entries :     1000 : Total  Size=       4883
*Baskets :        5 : Basket Size=      51200
*Br    1 :fBits     : UInt_t
*Entries :     1000 : Total  Size=       8887
*Baskets :        5 : Basket Size=      51200
*Br    2 :fSize     : Int_t
*Entries :     1000 : Total  Size=       4847
*Baskets :        5 : Basket Size=      51200
*Br    3 :fDummy    : Float_t fDummy[fSize]
*Entries :     1000 : Total  Size=  400035076
*Baskets :      273 : Basket Size=    7999488
*Br    4 :TRefTable : List of branch numbers w
*Entries :     1000 : Total  Size=       8602
*Baskets :        0 : Basket Size=      32000
```

Object Size = 40K

```
*Tree    :T         : An example of a ROOT tree
*Entries :    10000 : Total =       400530265 bytes
*        :          : Tree compression factor =   3.04
*Branch  :dummy
*Entries :    10000 : BranchElement (see below)
*Br    0 :fUniqueID : UInt_t
*Entries :    10000 : Total  Size=      40883 bytes
*Baskets :        5 : Basket Size=      51200 bytes
*Br    1 :fBits     : UInt_t
*Entries :    10000 : Total  Size=      80968 bytes
*Baskets :        6 : Basket Size=      51200 bytes
*Br    2 :fSize     : Int_t
*Entries :    10000 : Total  Size=      40847 bytes
*Baskets :        5 : Basket Size=      51200 bytes
*Br    3 :fDummy    : Float_t fDummy[fSize]
*Entries :    10000 : Total  Size=  400294572 bytes
*Baskets :     2301 : Basket Size=    7996416 bytes
*Br    4 :TRefTable : List of branch numbers with refe
*Entries :    10000 : Total  Size=      80857 bytes
*Baskets :        3 : Basket Size=      32000 bytes
```

Object Size = 4K

```
*Tree    :T         : An example of a ROOT tree
*Entries :   100000 : Total =       403661817 bytes  Fil
*        :          : Tree compression factor =   3.03
*Branch  :dummy
*Entries :   100000 : BranchElement (see below)
*Br    0 :fUniqueID : UInt_t
*Entries :   100000 : Total  Size=     401599 bytes  Fil
*Baskets :       13 : Basket Size=      51200 bytes  Com
*Br    1 :fBits     : UInt_t
*Entries :   100000 : Total  Size=     804340 bytes  Fil
*Baskets :       38 : Basket Size=      51200 bytes  Com
*Br    2 :fSize     : Int_t
*Entries :   100000 : Total  Size=     401531 bytes  Fil
*Baskets :       13 : Basket Size=      51200 bytes  Com
*Br    3 :fDummy    : Float_t fDummy[fSize]
*Entries :   100000 : Total  Size=  401289302 bytes  Fil
*Baskets :     7336 : Basket Size=    7952384 bytes  Com
*Br    4 :TRefTable : List of branch numbers with refere
*Entries :   100000 : Total  Size=     804527 bytes  Fil
*Baskets :       37 : Basket Size=      32000 bytes  Com
```

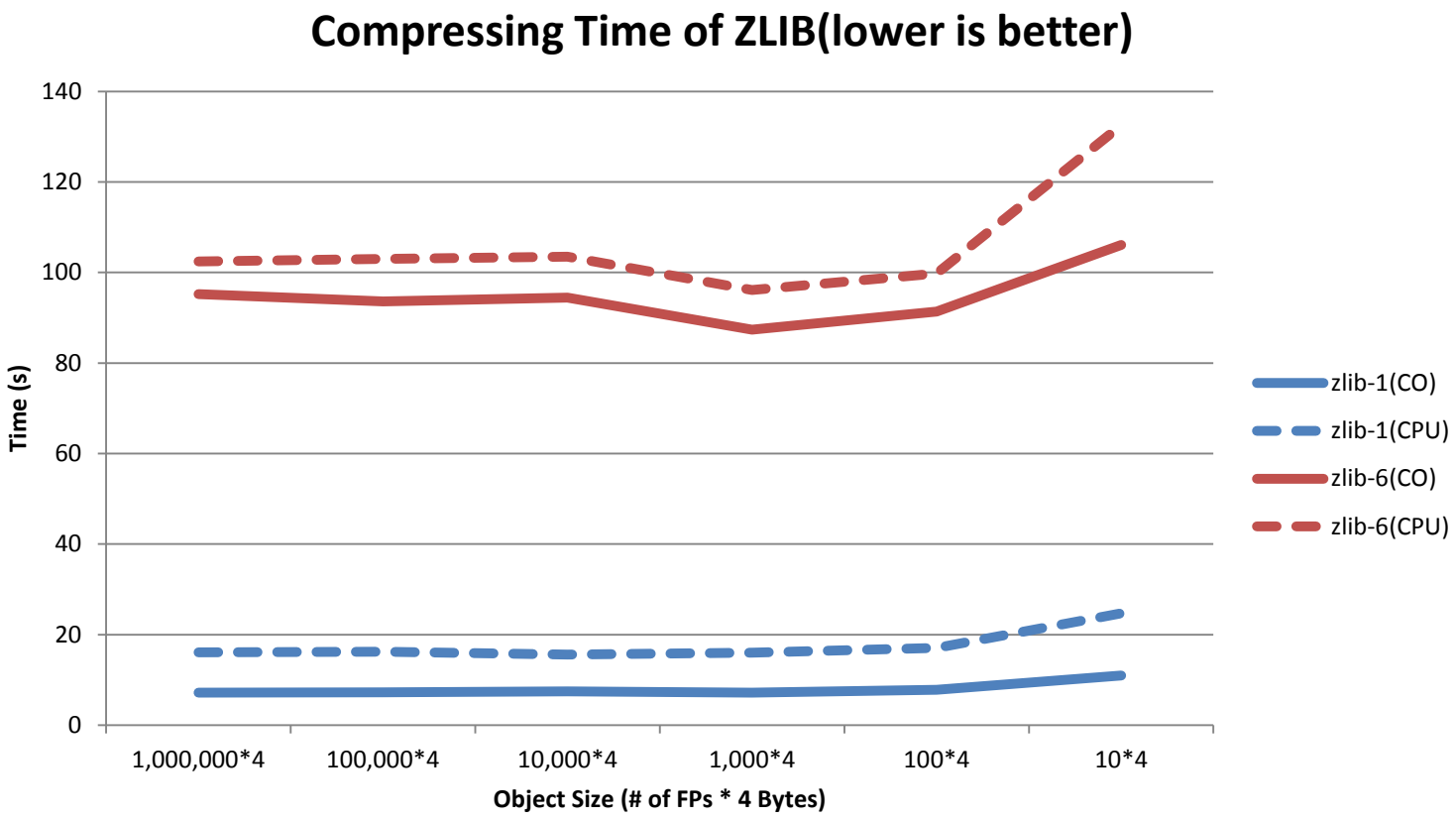
Object Size = 400B

```
*Tree    :T         : An example of a ROOT tree
*Entries :  1000000 : Total =       429641913 bytes
*        :          : Tree compression factor =   3
*Branch  :dummy
*Entries :  1000000 : BranchElement (see below)
*Br    0 :fUniqueID : UInt_t
*Entries :  1000000 : Total  Size=    4010024 bytes
*Baskets :       98 : Basket Size=      74240 bytes
*Br    1 :fBits     : UInt_t
*Entries :  1000000 : Total  Size=    8023144 bytes
*Baskets :      222 : Basket Size=     149504 bytes
*Br    2 :fSize     : Int_t
*Entries :  1000000 : Total  Size=    4009616 bytes
*Baskets :       98 : Basket Size=      74240 bytes
*Br    3 :fDummy    : Float_t fDummy[fSize]
*Entries :  1000000 : Total  Size=  405573994 bytes
*Baskets :     5472 : Basket Size=    7553024 bytes
*Br    4 :TRefTable : List of branch numbers with r
*Entries :  1000000 : Total  Size=    8043257 bytes
*Baskets :      375 : Basket Size=      32000 bytes
```

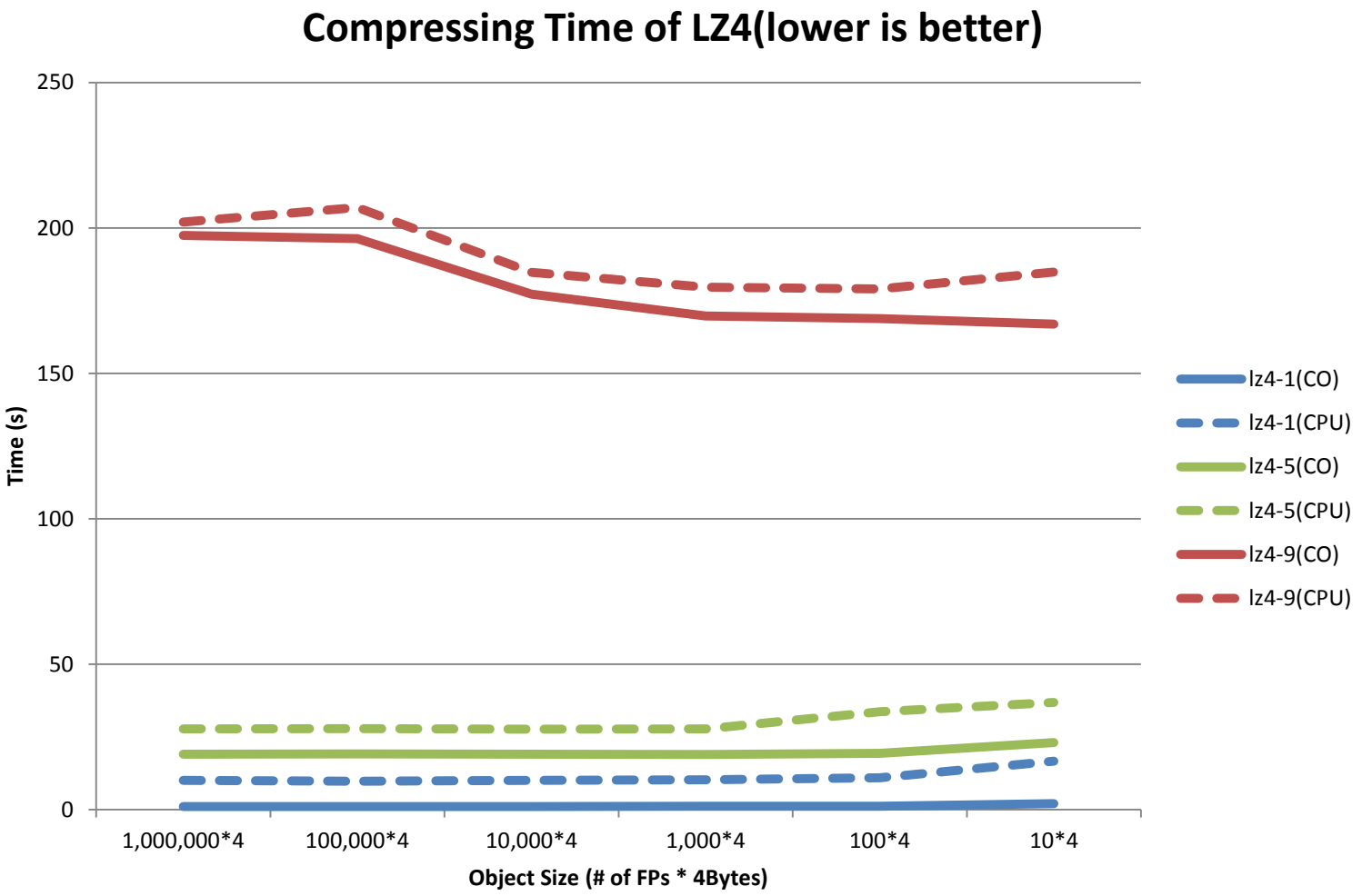
Object Size = 40B

```
*Tree    :T         : An example of a ROOT tree
424*Entries : 10000000 : Total =       691077948 bytes
   *        :          : Tree compression factor =   3.
*Branch  :dummy
*Entries : 10000000 : BranchElement (see below)
...*
   *Br    0 :fUniqueID : UInt_t
 27*Entries : 10000000 : Total  Size=   40045620 bytes
23 *Baskets :      446 : Basket Size=     463360 bytes
...*
   *Br    1 :fBits     : UInt_t
258*Entries : 10000000 : Total  Size=   80124349 bytes
37 *Baskets :     1227 : Basket Size=     929280 bytes
...*
   *Br    2 :fSize     : Int_t
 29*Entries : 10000000 : Total  Size=   40043820 bytes
51 *Baskets :      446 : Basket Size=     463360 bytes
...*
   *Br    3 :fDummy    : Float_t fDummy[fSize]
782*Entries : 10000000 : Total  Size=  450504294 bytes
94 *Baskets :     4622 : Basket Size=    5218304 bytes
...*
   *Br    4 :TRefTable : List of branch numbers with re
276*Entries : 10000000 : Total  Size=   80403027 bytes
28 *Baskets :     3757 : Basket Size=      32000 bytes
...*
```

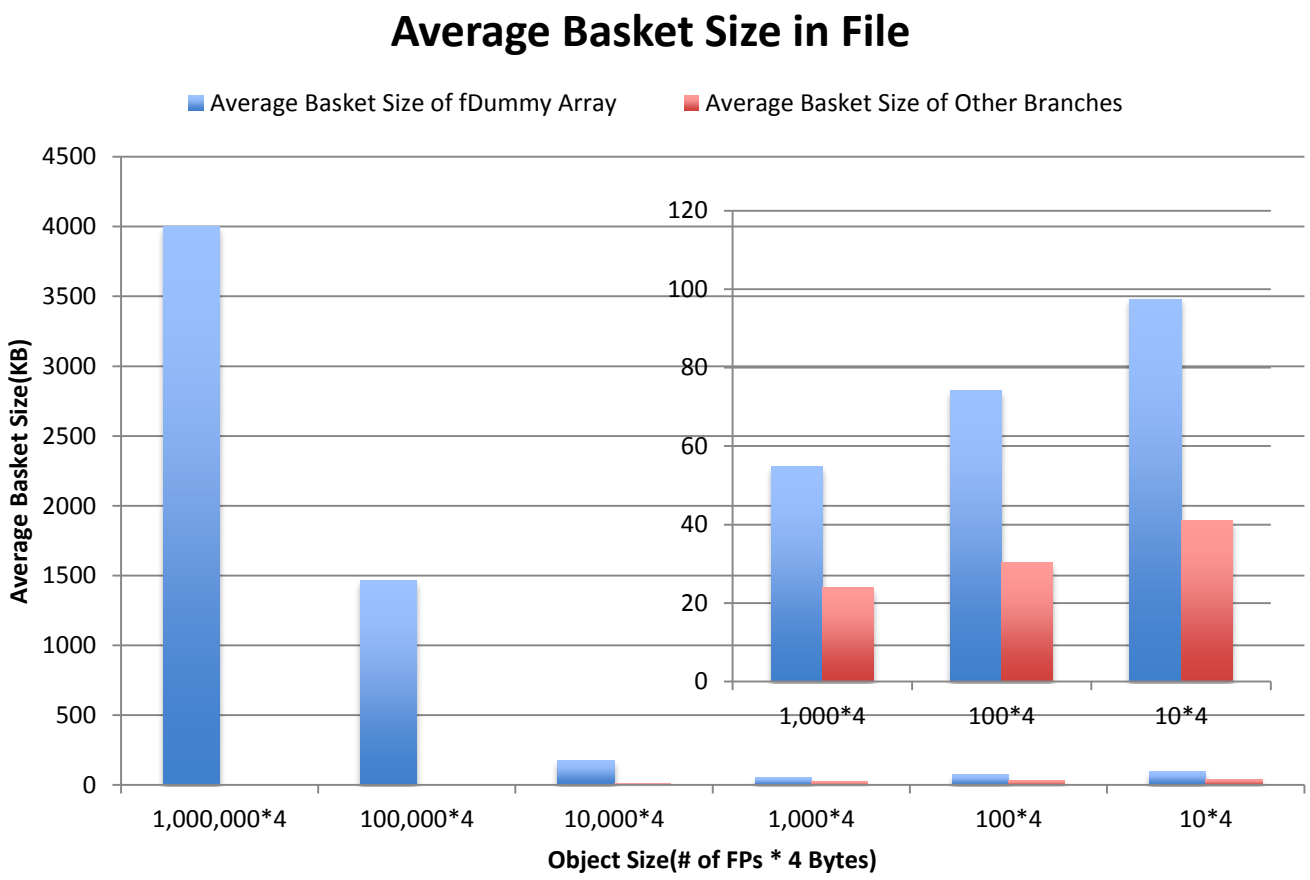
Appendix: Compressing Time of ZLIB



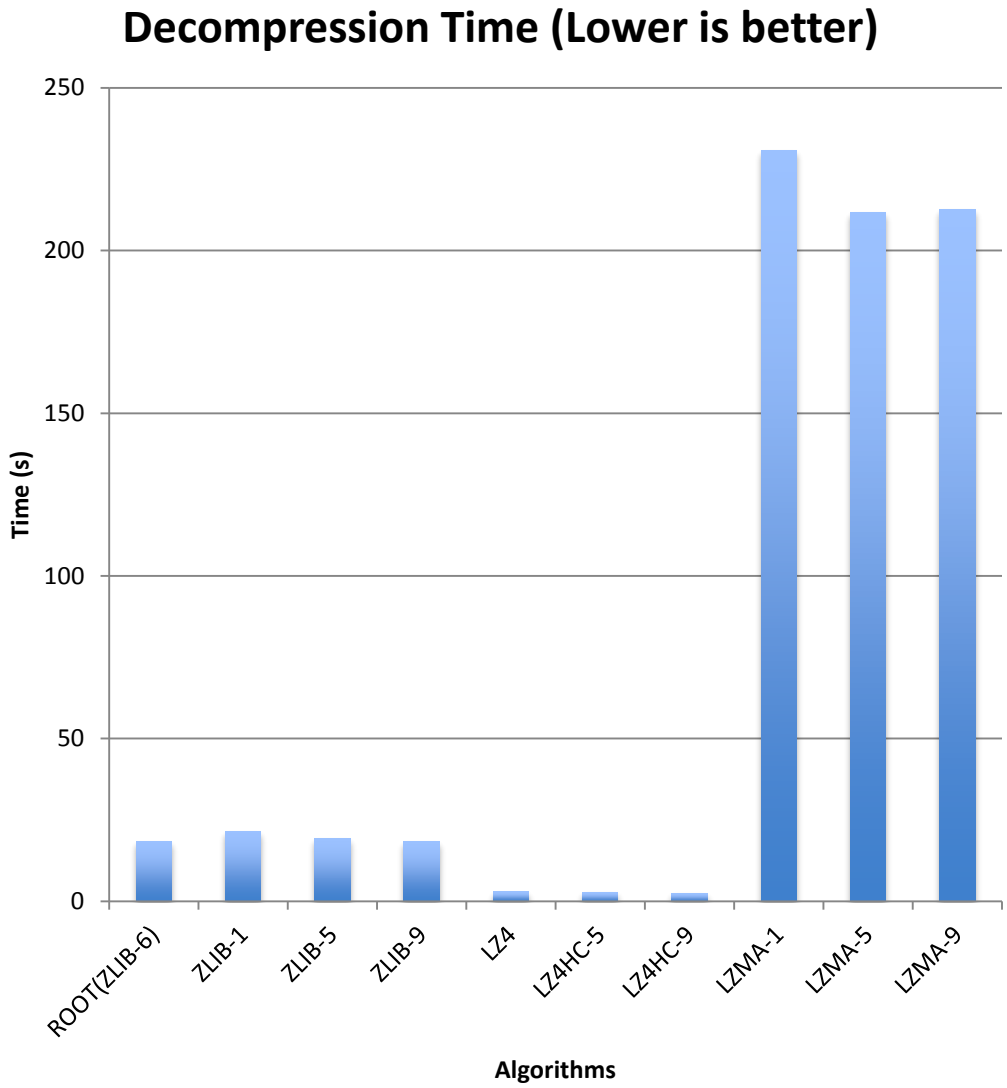
Appendix: Compressing Time of LZ4



Average Basket Size

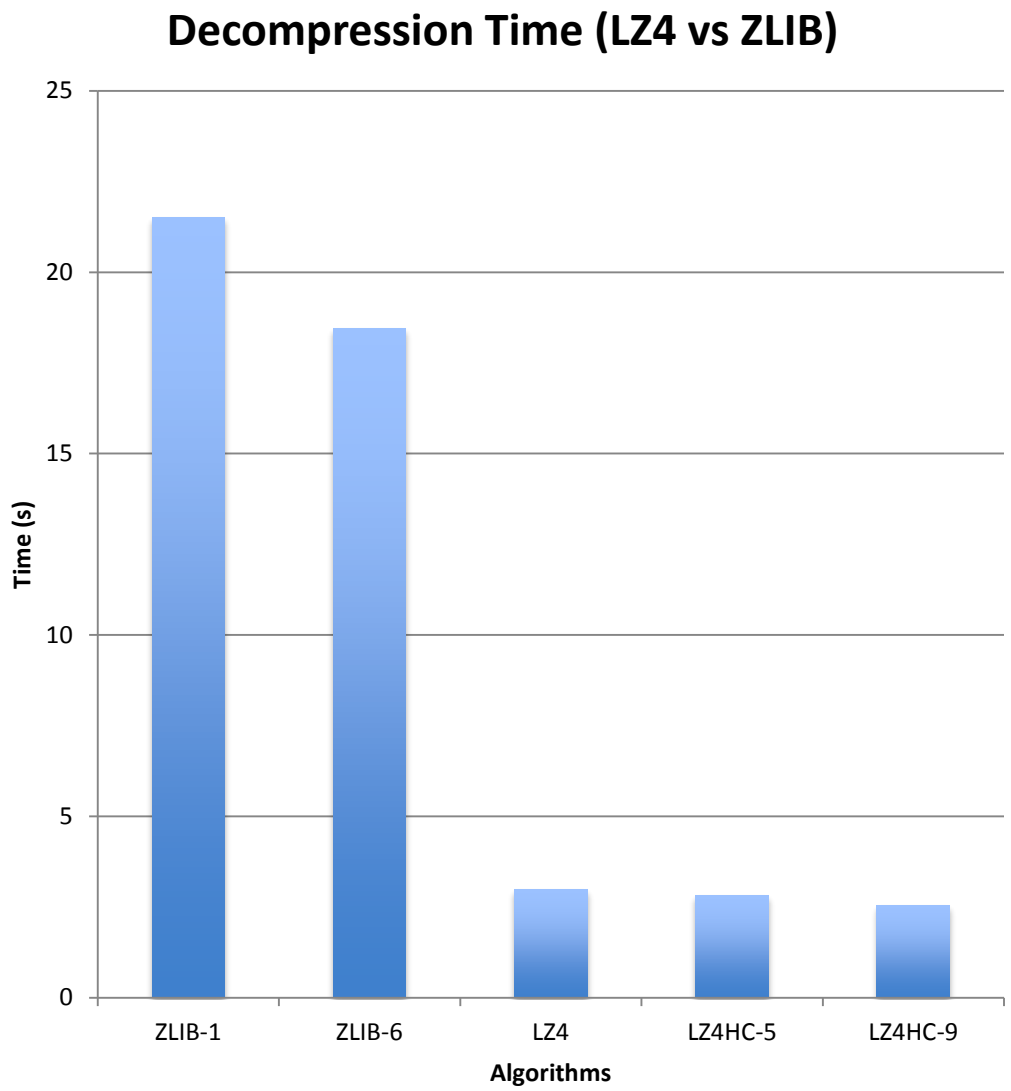


Decompressing Time(I)



- LZMA are slowest at all compression levels
- LZ4 are fastest at all compression levels

Decompressing Time(II)



- LZ4 is 4-7 times faster than ROOT(Zlib-6)