Data Compression

Edward Higgins

2018-07-06

Introduction

What is data compression?

- Used to reduce the number of bytes used to express the same information
- Can be:
  - lossy (eg. mp3, jpeg) $\rightarrow$ approximate representation of information
  - lossless (eg. .zip, .gz, .flac) $\rightarrow$ exact representation of information
- Example: `aaaaabbcccccc` => `5a,3b,6c`

Why do we care?

- Files can be big!
  - Experimental/Simulation data
  - Raw A/V
    - 24bit 96KHz audio: ~2GB per hour
    - 24bit 1080p 60Hz video: ~1.3TB per hour
  - Even plain text
    - Wikipedia, without revisions or multimedia: ~60GB
- Finite storage limits how much can be kept
- Finite bandwidth limits how quickly it can be transferred

Compression Algorithms

Run length encoding

- ‘Runs’ of data (repeated sequences) are represented as:
  - One copy of the data
  - Count of how many times it is repeated
- Example: `"0.00000000"` $\rightarrow$ `"101.80"` (one 0, one ., eight 0s)
- Originally designed for bitmap images, where large areas of white compress well
- Good for certain data types, but can increase file size

Dictionary encoding (eg. Lempel-Ziv variants, Snappy)

- Repeated strings are stored once, and referenced later
- Example:
  - `"O Romeo, Romeo, wherefore art thou Romeo?"`
  - `"O Romeo, \2, wherefore art thou \3?"`
- Many different variants:
  - How far back do you look for matches?
  - How do you perform the search?
  - How do you encode matches?
- Trade-off between compression ratio and compression time

Dictionary encoding: LZ77

```
------------------------------------------------------------------
 "abracadabrarray"

 <-- Search buffer --->    <-- Lookahead -->
 8  7  6  5  4  3  2  1  | 1  2  3  4  5  6 | Output (off, len, next)
 ------------------------+------------------+-----------------------
                         | a  b  r  a  c  a | ( 0, 0, a )
                         |                  |
                         |                  |
                         |                  |
                         |                  |
                         |                  |
                         |                  |

------------------------------------------------------------------
```

Dictionary encoding: LZ77

```
----------------------------------------------------------------------
 "abracadabrarray"

 <-- Search buffer --->    <-- Lookahead -->
 8  7  6  5  4  3  2  1  | 1  2  3  4  5  6 | Output (off, len, next)
 ------------------------+------------------+-------------------------
                         | a  b  r  a  c  a | ( 0, 0, a )
                      a  | b  r  a  c  a  d | ( 0, 0, b )
                         |                  |
                         |                  |
                         |                  |
                         |                  |
                         |                  |

----------------------------------------------------------------------
```

Dictionary encoding: LZ77

```
---------------------------------------------------------------------
 "abracadabrarray"

 <-- Search buffer --->    <-- Lookahead -->
 8  7  6  5  4  3  2  1  | 1  2  3  4  5  6 | Output (off, len, next)
 ------------------------+------------------+------------------------
                         | a  b  r  a  c  a | ( 0, 0, a )
                      a  | b  r  a  c  a  d | ( 0, 0, b )
                   a  b  | r  a  c  a  d  a | ( 0, 0, r )
                         |                  |
                         |                  |
                         |                  |
                         |                  |

---------------------------------------------------------------------
```

Dictionary encoding: LZ77

```
-----------------------------------------------------------------------
 "abracadabrarray"

 <-- Search buffer --->    <-- Lookahead -->
 8  7  6  5  4  3  2  1  | 1  2  3  4  5  6 | Output (off, len, next)
 ------------------------+------------------+------------------------
                         | a  b  r  a  c  a | ( 0, 0, a )
                      a  | b  r  a  c  a  d | ( 0, 0, b )
                   a  b  | r  a  c  a  d  a | ( 0, 0, r )
                a  b  r  | a  c  a  d  a  b | ( 3, 1, c )
                         |                  |
                         |                  |
                         |                  |

-----------------------------------------------------------------------
```

Dictionary encoding: LZ77

```
-----------------------------------------------------------------------
 "abracadabrarray"

 <-- Search buffer --->    <-- Lookahead -->
 8  7  6  5  4  3  2  1  | 1  2  3  4  5  6 | Output (off, len, next)
 ------------------------+------------------+-----------------------
                         | a  b  r  a  c  a | ( 0, 0, a )
                      a  | b  r  a  c  a  d | ( 0, 0, b )
                   a  b  | r  a  c  a  d  a | ( 0, 0, r )
                a  b  r  | a  c  a  d  a  b | ( 3, 1, c )
          a  b  r  a  c  | a  d  a  b  r  a | ( 2, 1, d )
                         |                  |
                         |                  |

-----------------------------------------------------------------------
```

Dictionary encoding: LZ77

```
-----------------------------------------------------------------------
 "abracadabrarray"

 <-- Search buffer --->    <-- Lookahead -->
 8  7  6  5  4  3  2  1  | 1  2  3  4  5  6 | Output (off, len, next)
 ------------------------+------------------+-----------------------
                         | a  b  r  a  c  a | ( 0, 0, a )
                      a  | b  r  a  c  a  d | ( 0, 0, b )
                   a  b  | r  a  c  a  d  a | ( 0, 0, r )
                a  b  r  | a  c  a  d  a  b | ( 3, 1, c )
          a  b  r  a  c  | a  d  a  b  r  a | ( 2, 1, d )
    a  b  r  a  c  a  d  | a  b  r  a  r  r | ( 7, 4, r )
                         |                  |

-----------------------------------------------------------------------
```

Dictionary encoding: LZ77

```
-----------------------------------------------------------------
 "abracadabrarray"

 <-- Search buffer --->    <-- Lookahead -->
 8  7  6  5  4  3  2  1  | 1  2  3  4  5  6 | Output (off, len, next)
 ------------------------+------------------+-----------------------
                         | a  b  r  a  c  a | ( 0, 0, a )
                      a  | b  r  a  c  a  d | ( 0, 0, b )
                   a  b  | r  a  c  a  d  a | ( 0, 0, r )
                a  b  r  | a  c  a  d  a  b | ( 3, 1, c )
          a  b  r  a  c  | a  d  a  b  r  a | ( 2, 1, d )
    a  b  r  a  c  a  d  | a  b  r  a  r  r | ( 7, 4, r )
 c  a  d  a  b  r  a  r  | r  a  y          | ( 3, 2, y )

-----------------------------------------------------------------
```

Symbol reordering

- Doesn't actually compress the data
- Improves effectiveness of algorithms (eg. run length encoding, dictionary encoding)
- Example: `"banana"` $\rightarrow$ `"bnnaaa"`

Symbol reordering: Burrows-Wheeler Transform

```
---------------------------------------------------------------------
 1. Input          2. Make all      3. Sort          4. Take the
                      rotations        columns          last column



                     ^banana$         anana$^b
                     $^banana         ana$^ban
                     a$^banan         a$^banan
  ^banana$           na$^bana         banana$^          bnn^aa$a
                     ana$^ban         nana$^ba
                     nana$^ba         na$^bana
                     anana$^b         ^banana$
                     banana$^         $^banana

---------------------------------------------------------------------
```

Entropy encoding

- Each unique symbol is given its own variable-length code
- More frequently used symbols are given shorter codes
- Eg. Morse code:
  - `e` is `.`
  - `z` is `--..` (8 times the lengh of `e`)

Entropy encoding: Huffman codes

"this is an example of a huffman tree"   36 characters = 288 bits

Letter	Count	ASCII code	Letter	Count	ASCII code
' '	7	00100000	's'	2	01110011
'a'	4	01100001	't'	2	01110100
'e'	4	01100101	'l'	1	01101100
'f'	3	01100110	'o'	1	01101111
'h'	2	01101000	'p'	1	01110000
'i'	2	01101001	'r'	1	01110010
'm'	2	01101101	'u'	1	01110101
'n'	2	01101110	'x'	1	01111000

Entropy encoding: Huffman codes

```
-------------------------------------------------------------

 7   4   4   3   2   2   2   2   2   2   1   1   1   1   1   1
' '  a   e   f   h   i   m   n   s   t   l   o   p   r   u   x







Join the 2 least frequent entries into a subtree...

-------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
------------------------------------------------------------

  7   4   4   3   2   2   2   2   2   2   1   1   1   1    2
 ' '  a   e   f   h   i   m   n   s   t   l   o   p   r    *
                                                          / \
                                                         u   x




Repeat for all pairs with frequency of 2...

------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
--------------------------------------------------------------------

 7   4   4   3   2   2   2   2   2   2     2       2       2
' '  a   e   f   h   i   m   n   s   t     *       *       *
                                          / \     / \     / \
                                         l   o   p   r   u   x




Join the 2 least frequent entries into a subtree...

--------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
--------------------------------------------------------------------

 7   4   4   3   2   2   2   2   2   2     2           4
' '  a   e   f   h   i   m   n   s   t     *           *
                                          / \       __/ \__
                                         l   o     *       *
                                                  / \     / \
                                                 p   r   u   x




Repeat for all pairs with frequency of 4...

--------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
---------------------------------------------------------------------


  7   4   4   3   2     4       4       4               4
 ' '  a   e   f   h     *       *       *               *
                       / \     / \     / \__         __/ \__
                      i   m   n   s   t     *       *       *
                                           / \     / \     / \
                                          l   o   p   r   u   x




Re-sort the list...

---------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
---------------------------------------------------------------------

  7   4   4     4       4       4               4         3   2
 ' '  a   e     *       *       *               *         f   h
               / \     / \     / \__         __/ \__
              i   m   n   s   t     *       *       *
                                   / \     / \     / \
                                  l   o   p   r   u   x



Join the 2 least frequent entries into a subtree...

---------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
---------------------------------------------------------------------

  7   4   4     4       4       4               4           5
 ' '  a   e     *       *       *               *           *
               / \     / \     / \__         __/ \__       / \
              i   m   n   s   t     *       *       *     f   h
                                   / \     / \     / \
                                  l   o   p   r   u   x




Re-sort the list...

---------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
---------------------------------------------------------------------

  7     5     4   4     4       4       4               4
 ' '    *     a   e     *       *       *               *
       / \             / \     / \     / \__         __/ \__
      f   h           i   m   n   s   t     *       *       *
                                           / \     / \     / \
                                          l   o   p   r   u   x




Join the 2 least frequent entries into a subtree,
and repeat for all pairs with frequency 8...
---------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
---------------------------------------------------------------------

  7     5       8           8                   8
 ' '    *       *           *                   *
       / \     / \       __/ \__         ______/ \______
      f   h   a   e     *       *       *               *
                       / \     / \     / \__         __/ \__
                      i   m   n   s   t     *       *       *
                                           / \     / \     / \
                                          l   o   p   r   u   x


Re-sort the list...

---------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
---------------------------------------------------------------------

    8           8                   8                 7     5
    *           *                   *                ' '    *
   / \       __/ \__             ______/ \______               / \
  a   e     *       *       *               *             f   h
           / \     / \     / \__         __/ \__
          i   m   n   s   t     *       *       *
                               / \     / \     / \
                              l   o   p   r   u   x


Join the 2 least frequent entries into a subtree...

---------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
---------------------------------------------------------------------

    8           8                   8                  12
    *           *                   *                   *
   / \       __/ \__         ______/ \______           / \__
  a   e     *       *       *               *        ' '    *
           / \     / \     / \__         __/ \__           / \
          i   m   n   s   t     *       *       *         f   h
                               / \     / \     / \
                              l   o   p   r   u   x


Re-sort the list

---------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
---------------------------------------------------------------------

   12           8           8                   8
    *           *           *                   *
   / \__       / \       __/ \__         ______/ \______
 ' '    *     a   e     *       *       *               *
       / \             / \     / \     / \__         __/ \__
      f   h           i   m   n   s   t     *       *       *
                                           / \     / \     / \
                                          l   o   p   r   u   x


Join the 12-8 and 8-8 subtrees

---------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
----------------------------------------------------------------------


         20                          16
          *                           *
     ____/ \____             ________/ \________
    *           *           *                   *
   / \__       / \       __/ \__         ______/ \______
 ' '    *     a   e     *       *       *               *
       / \             / \     / \     / \__         __/ \__
      f   h           i   m   n   s   t     *       *       *
                                           / \     / \     / \
                                          l   o   p   r   u   x
Join the final 20-16 subtrees...

----------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
---------------------------------------------------------------------
                                                    The Huffman Tree
                   0 <- * -> 1
           ____________/ \____________
          *                           *
     ____/ \____             ________/ \________
    *           *           *                   *
   / \__       / \       __/ \__         ______/ \______
 ' '    *     a   e     *       *       *               *
       / \             / \     / \     / \__         __/ \__
      f   h           i   m   n   s   t     *       *       *
                                           / \     / \     / \
                                          l   o   p   r   u   x
---------------------------------------------------------------------
```

Entropy encoding: Huffman codes

```
---------------------------------------------------------------------
 "this is an example of a huffman tree"     36 characters = 145 bits

  Letter | Count | Huff. code       Letter | Count | Huff. code
 --------+-------+-----------      --------+-------+-----------
   ' '   |  7    | 000               's'   |  2    | 1011
   'a'   |  4    | 010               't'   |  2    | 1100
   'e'   |  4    | 011               'l'   |  1    | 11010
   'f'   |  3    | 0010              'o'   |  1    | 11011
   'h'   |  2    | 0011              'p'   |  1    | 11100
   'i'   |  2    | 1000              'r'   |  1    | 11101
   'm'   |  2    | 1001              'u'   |  1    | 11110
   'n'   |  2    | 1010              'x'   |  1    | 11111

---------------------------------------------------------------------
```

Common Formats

Deflate

- Used in .zip, zlib (.gzip, .png, ssh, . . .), Intel® QuickAssist Technology
- Combination of LZ77 and Huffman coding
- Good compromise between compression ratio and compression speed

BZip2

- Uses Burrows-Wheeler and Move-To-Front transforms to make data more compressible
- Run length encoding and Huffman encoding then used to compress the data
- Compared to Deflate:
  - Higher compression ratio
  - Similar decompression speeds
  - Much slower compression speeds

LZMA

- Used in 7z (windows) and xz (unix) formats, and many package distributions (deb, rpm, . . .)
- Uses a modified LZ77 algorithm with range encoding (an entropy encoding algorithm)
- Higher compression ratios than bzip2, with better decompression times

LZ4

- High speed compression with reasonable compression ratio
- LZ77-esque dictionary encoding with no entropy encoding
- Stores data in ‘blocks’

```
|----|----|----|----------------|--------|----|
| t1 | t2 | L1 | literal string | offset | L2 |
|----|----|----|----------------|--------|----|

t1 + L1 = length of literal string
t2 + L2 = length of match
```

- Implemented in many ZFS filesystem implementations

How do they compare?

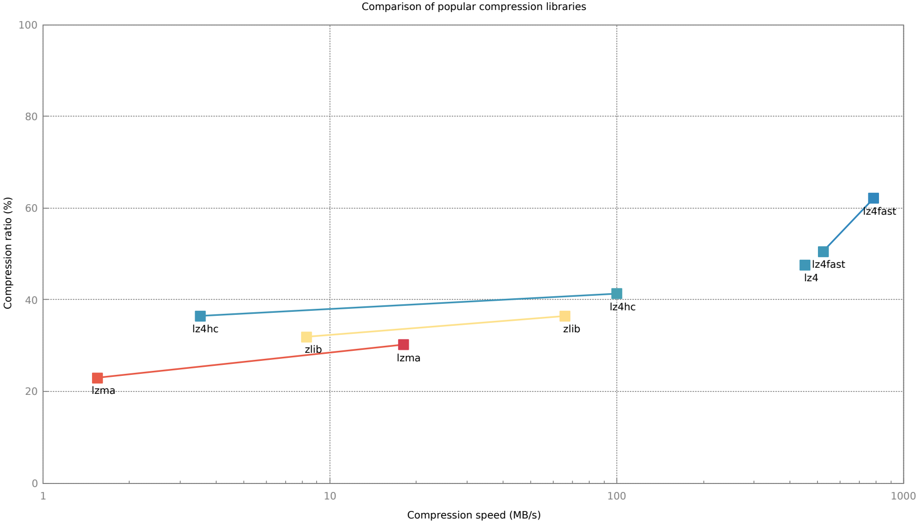

Figure 1: Compression speeds

How do they compare?

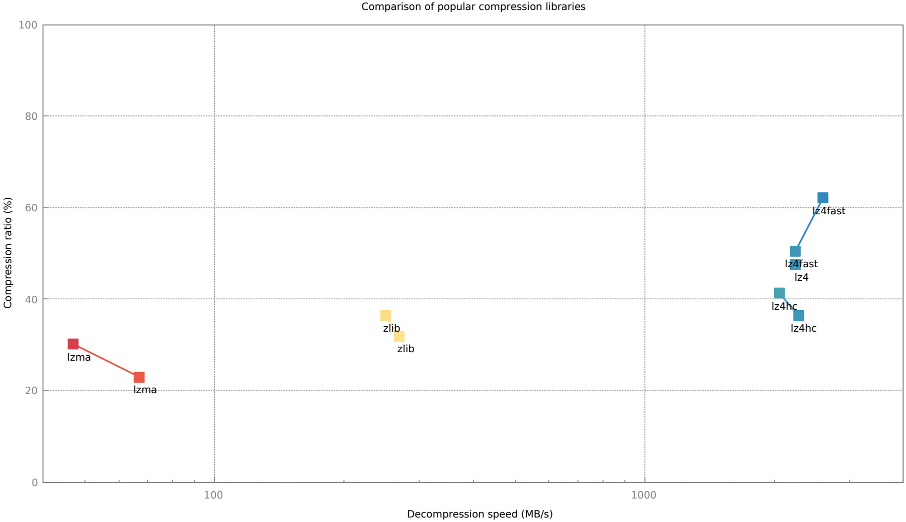

Figure 2: Decompression speeds