Share Archiver v0.1

This document contains details about the share archiver and my thoughts/rationale during the creation. I still have much to learn ;)

Share Archiver's vision is to achieve faster downloads for all by improving online file sharing through small archive size with fast decompression.

Background:
The internet enables us to share files with each other freely, which let to file sharing platforms being used immensely. WeTransfer, just one of many, hit 1.5 million transfers a day alone, 17 files per seconds being downloaded, and that was back in 2013 (source). To enable single file download, multiple files are added into a single zip archive, on which the user later extracts the files. ZIP uses Deflate, which had been specified in 1996. Since then, stronger compression algorithms have been created. Having smaller archive sizes optimizes this process, leading to faster downloads and less data usage for the users overall.

Goal:
My vision was to make a little application to optimize this process. Since these files only get archived (compressed) once by the uploader and uploaded one, but downloaded and extracted up to a thousand times (or more, there really is no limit), my focus was to optimize both archive size and decompression speeds. For this purpose, I have started working on my personal project, Share Archiver.

Plan:
Using only strong asymmetric compressors with fast decompression speeds, creating an archive that compresses each single file with each compressor and adding the smallest compressed file to the archive should result in small archive size and still fast decompression (extraction meaning decompressing on a per file basis with de corresponding decompressor again).

Current Implementation v0.1:
Share Archiver is basically a simple python script that takes a path to a folder as input and copies all files into a folder called sharetmp. You can observe this by the application creating a "sharetmp" directory in the current working directory of the application. From there on, for each file, it generates a temporary directory (invisible) where it writes the output (the compressed file) of each compressor. Then it traverses the contents of this temp. Directory, finding the smallest file, and adding it (by moving it into the sharetmp folder, replacing the input file with the compressed file) and continuing with the next, until all copied files from the input folder have been replaced by the min size compressed version of themselves, at the end adding the content of the sharetmp folder into a .share archive output, then deleting the sharetmp folder.
Since development happened on my Ubuntu 20.04.3 LTS Desktop, this is currently a linux only application. I also want to use pyinstaller for a simpler distribution and prepackage python environment so it does not need to be present on the compression/decompression system. By default, pyinstaller creates an executable that can only be used by a system that uses the same

OS as the system that created the executable with pyinstaller. Following I will add the reasoning for compressors used, and compression profiles provided, and some ideas what I can work on for the next version of the share archiver.

Distribution
For distribution, I used pyinstaller and created a single file which includes binaries of the compressors built (with make) on my system. I tried to do the same on a Ubuntu18 virtual machine, to also support older versions (binaries generated by pyinstaller do not support older versions, so like a binary from ubuntu20 will not run on a ubuntu18, bot the other way around it will).
As another option, I created an executable with the folder of the compressors that will be used additionally, so the user can build those binaries on his system himself with make.

Compressors:
I was looking for strong asymmetric data compressors running on linux, that have good decompression speeds. See the Compressor Selection chapter for information on how I made this selection.
Share Archiver v0.1 makes use of the following great compressors shoutout here to the developers of these fantastic compressors, they did a great job and invested a lot of time for us/the community to profit from these efforts:
Zlib
Balz
LZMA (XZ)
GZip
Zstandard
PPmd
LZ4
Brotli
LZHAM
ZOPFLI
Lzturbo
GLZA
Balz
BSC
CSArc
NNCP
ZPAQ

Compression Profiles / Settings:
I defined the following Profiles: Fast, Default, Strong and Max.
The Default Profile uses all asymmetric compressors with their default settings, see additional notes.
The Strong Profile uses all asymmetric compressors with their strong settings, see additional notes.

The Fast Profile reduces compression time for usability and might affect resulting archive size. It does it by using only the three best asymmetric compressors in their default profile when used on the corpus. I wanted to cut time by compressing each file with only the three best asymmetric compressors instead of the 15 asymmetric compressors, best as in smallest resulting size on files since when choosing the compressors I already chose those that satisfy the decompression speeds criteria. Since the resulting archive only contains one compressed version of each file, therefore this profile will not affect decompression speed. More info on how I made that selection in the fast chapter.
The Max Profile was simply for my curiosity but is made available. This Profile should not be used for online file distribution as by using all available compressors (meaning additionally nncp and zpaq) decompression might take a long time and frustrate the users. It is simply meant to achieve the smallest achievable filesize. Compression time might also be long.

Comparison / Benchmarking
This is in no way a comprehensive test, but more like my personal tests to satisfy my curiosity. For these tests, I used the corpus from maximumcompression.com since compared to silesia and calgary and so forth, this seems more appropriate for the intended usecase of this archiver concerning file types and file sizes. (who just has large books stored as .txt files on their pc and wants to share them online? I guess only a smaller percentage of all uploaders).
I will just put the comparison to zip here, as I often find files being shared in zip archives online

11735040 maximumcompressioncorpus.share
14405261 maximumcompressioncorpus.zip

If you want to see the archive size comparison with compressors like 7z, xz and so forth, see Quick Size Comparison.

The default profile had the following times on my pc:
Compression, Default:
Original Size: 53134726
Archive Size: 11735040
Compression Ratio: 4.527869184936736
Program Execution Time (PET): 156.91 seconds
Decompression:
Program Execution Time (PET): 0.34 seconds

For detailed information with compression and decompression times and resulting archive size of each profile with the , read the Benchmarking chapter.

Multiprocessing
To decrease time, see multiprocessing

But it leads to more ram usage, see next topic

RAM
When using the application I once ran out of ram, therefore I added the memory saver option, read more in this chapter.

Future Versions Ideas:

- Provide a flag to make use of Wine, so Share can make also make use of compressors that were made for windows, like RAZOR and p8f, also strong ones like paq8pxd for max profile. Of course, if such a compressor has been used for one of the files, it can only be fully decompressed on a linux system that also has wine installed. But being able to make use of even more fantastic compressors might improve the application, and the compressors used in the fast profile might change and also the number of compressors used in the fast profile might change, depending on the tests.
- To avoid confusion, additionally to console output, the .sharel (for basic linux), .sharelw (linux with wine) and also .sharew (windows version) extensions could be used to indicate decompression system support for these archives.
- Since I would be integrating wine compressors, I could work on a windows executable. It would work by only using the windows supported compressors, or in other words, we simply deactivate the linux only compressors from the linux-wine-version mentioned in point 1.
- I could integrate a -general flag, that would create an archive being decompressable on both systems by only using compressors supported by windows and linux.
- The previously mentioned points can be merged together. A linux system with wine that compresses a .sharelw archive is already compressing every single file with every single integrated asymmetric compressor. Therefore, it will be able to, at the same time, generate a .sharel and .sharew archive additionally, without any additional compressing. It would simply filter by extensions and sort by filesize and only add supported files to that share archive. There will be additional filtering and file copying operations, but not any additional compression, which is the time consuming part.
- Save the original filesize of each file in the archive information file, and compare after extraction. Or the checksum? I could outprint mismatches of original values with decompressed values in the console output. (maybe with that also, a testarchive action? Would decompress each file and compare values and throw information in case of mismatch or give a green light if everything seems allright. But files are only decompressed into temporary folders, so no cleanup needed afterwards (deleting files). It would also outprint the time, giving an estimate to the creator of the archive of how long it will take to decompress that archive. Since fast decompression times was part of the design, this check should not take too long (if not max profile).
- Maybe again merge above point by providing a benchmark or test flag, it will simply after compressing the archive right away check the archive as described above, and additionally create a text file, that includes the time taken to compress (create the share archive), archive size, compression ratio, if the test was successful, and estimated decompression time (simply the test time, but it might not be exactly the same because of additional operations so therefore estimated decompression time)

- A GUI for decompression? Because more people would be decompressing than compressing.
- A GUI for compression/generation of share archive?
- Maybe a docker container with a linux with wine system running on it with share? It would be able to create the .sharel, .sharelw and .sharew archives in one go (as described in above points) also on a windows system. (with a mounted volume, where we can simply copy our files into, let the container run, and the files will have been replaced with the share archives after operations.
- Maybe above point but create a virtual appliance, a .cow2 file, i mean just some virtual machine image that can be spun up and does exactly that. If there were GUI versions, those would be on there and started up/opened automatically at startup. But maybe this is way too much already.

Anyway it has been interesting so far to work on this project as a hobby ^^
I want to add that in no way am I an expert in compression or anything like that, I just got interested and wanted to create this archiver. I am sure there is a lot of potential for improvements, as well in the code as also in things like the compressors chosen (maybe there are some great ones to add still) or the compressor options themselves to be more optimized and other things like that. I am learning new things all the time, and that is great, that's exactly how it should be and how i like it ^^ Nothing needs to be perfect, especially not from the beginning if it were so. Otherwise we would never start, or never show our work to anyone, which would delay us from getting constructive feedback and improving the hobby/application we are working on. And positive feedback/encouragement also increases the likelihood of continuing working on this project as a small hobby :)
Philip Hofmann

Compressor Selection

To get a list of compressors to use, I looked at different compression Benchmarks:

- Matt Mahoney's Benchmarks (disregarded benchmarks that did not include decompression speeds since this was an important metric I was looking for)
- Maximumcompression.com (loaded graphics from internet archive snapshots since live version does not show it)
- Squeezechart
- Compressionsratings.com
- World Compression Challenge
- Squash Compression Benchmark
- Lzturbo Compression Benchmark
- PeaZip (https://peazip.github.io/fast-compression-benchmark-brotli-zstandard.html)
- https://community.centminmod.com/threads/round-4-compression-comparison-benchmarks-zstd-vs-brotli-vs-pigz-vs-bzip2-vs-xz-etc.18669/
- https://github.com/inikep/lzbench/blob/master/lzbench18_sorted.md
- https://cran.r-project.org/web/packages/brotli/vignettes/brotli-2015-09-22.pdf

While looking at all this data, graphs, I took note/made a list of compressors that seem to have good compression ratio and fast decompression speeds.
The list I came up with included Razor, Zstandard, Brotli, Lzham, Bzip2, LZA, Packet, rzm, mcomp, zcm, and so forth…

I then disregarded any windows compressors for this version since I was developing on a linux for linux (no wine consideration yet, i wanted it to work on standar linux. Or, on standard ubuntu since thats the distro i am familiar with (love the budgie desktop environment with it). I know that there are other distros (and about distro hopping), or rather i tried zorin, manjaro, kali, parrot, arcolinux, whonix, tails, qubes, … all out as vms (i am aware they all are very different, i tried them because of the aspect they were focussing on, like zorin for windows switchers, manjaro for arch linux made easy, some security distros with privacy tools and mechanisms and virtualizations, learning distros etc . But this is the one that endured on my systems. Grew up, used longest and still use windows tho (dualboot), especially for gaming. Anyway i am getting waay off topic).

Anyway those that remained, I tried to get a hold of those binaries. Sometimes it was skimming encode.ru posts for the executables once posted by the creators. Or then there were easy accessible sites for it like with zpaq. To make it all short, it was subjective process, i did not have hard requirements (like a certain decompression speed minimum, but rather in comparison to all the ones I thought would fit when looking at all this data and graphs. It does not sound like it, but this process actually took just a lot of time (comparing compressors by using all this data)
When writing this I just realized, I maybe should have used pxz instead of just xz. I'll take it on my to do list to look into it.

Compression Profiles

Fast

The fast profile related to compression time, not decompression time. The reason is simple: Share should be fast decompressing by design, as I chose only compressors that were fast enough for my taste, see compressors selection section. The output of each file is its compressed version with such a compressor. The reason compression takes so much longer, is that in contrast to decompression, every single file is compressed multiple times. This can of course lead up to very long compression times with big files. Therefore I wanted to provide this Profile for this specific case where the user is not willing to invest the time compressing. So i am trying to approach similar share archive size while drastically lowering compression time.

For creating the fast profile, I reduce the number of compressors used which will drastically improve compression speeds. We leave them on the default profile/settings.
For this I needed a test / some baseline of data on which I can make that selection of compressors.
This assumes that each compressor compresses best with a certain filetype. So we choose a testset with some different filetypes in it and then use the min compressors.

For this test we wanted to try filetypes that will most likely be used by a normal user (not for example the silesia corpus, which includes and overrepresents whole books as text files. Who has a pc full of stored large books in single text files? And uploads them for others to download? This subjectively seems like an very specific usecase to me)
This is why I went with the testset from maximumcompression.com which has different everyday filetypes included

Testset from maximumcompression.com
https://www.maximumcompression.com/data/files/full_testset.7z on 06.10.2021

(PS full outputs at the end in [data output subsection](#))

'du -b' on the decompressed testset:
53138822

share compression with default
Archive Size: 11714560
Program Execution Time (PET): 182.83 seconds

If we look at the compressors that reached min file size (see full data section output below) we find the following compressors (or file extensions in this case):

Min size: brotli (2x), bsc (2x), xz (3x), ppmd, glza, csa
Compressors used: 14
Compressors that reached min sizes: 6
Compressor Reduction: 8

Can we reduce the number of compressors even further to increase compression speed? brotli, bsc and xz were used more than once, maybe we can replace the only-once-min-compressors with one of these if file differences are not too severe. So lets have a look at the specific instances where a compressor was min only once, filtered with min compressors (you find the whole filtered data according to min compressors in the data section output again, here only these specific instances):

```
  1010238  rafale.bmp.brotli
  782132  rafale.bmp.bsc
  1153481  rafale.bmp.csa
  982910  rafale.bmp.glza
  780727  rafale.bmp.ppmd
  976448  rafale.bmp.xz
min File:  /tmp/tmprcxu3m5m/rafale.bmp.ppmd
```

-> bsc is closest to ppmd min file size, we replace ppmd with bsc, so modified list of min compressors: brotli (2x), bsc (3x), xz (3x), glza, csa

```
  758927  FP.LOG.brotli
  528192  FP.LOG.bsc
  1043598  FP.LOG.csa
  508262  FP.LOG.glza
  659868  FP.LOG.ppmd
  842004  FP.LOG.xz
min File:  /tmp/tmp_s3c_q95/FP.LOG.glza
```

-> bsc closest to glza min file size, modified list of min compressors: brotli (2x), bsc (4x), xz (3x), csa

```
  1451351  AcroRd32.exe.brotli
  1605468  AcroRd32.exe.bsc
  1348393  AcroRd32.exe.csa
  1552739  AcroRd32.exe.glza
  1541891  AcroRd32.exe.ppmd
  1406152  AcroRd32.exe.xz
min File:  /tmp/tmpvi8voccq/AcroRd32.exe.csa
```

-> xz closest to csa min file size, modified list of min compressors: brotli (2x), bsc (4x), xz (4x)

Therefore the chosen compressors for the fast profiles are: xz, bsc and brotli
Since we reduced the number of used compressors to 3, this should significantly improve compression time. Let test it with the same testset.

Recompression with share with only xz, bsc and brotli compressors:
Archive Size: 11786240
Program Execution Time (PET): 124.67 seconds

Compared with Default compression:
Archive Size: 11714560
Program Execution Time (PET): 182.83 seconds

As expected, it reduces compression time but archive size went up.
(To be honest, I expected an even more drastic reduction of compression time since we reduced the number of used compressor so drastically.)

Just out of curiosity, I will benchmark the previously 6 min compressors against each other by compressing this whole testset with only one of those compressors enabled each time
brotli:
Archive Size: 12462080
Program Execution Time (PET): 124.21 seconds
bsc:
Archive Size: 12544000
Program Execution Time (PET): 3.67 seconds
xz:
Archive Size: 12441600
Program Execution Time (PET): 13.74 seconds
ppmd:
Archive Size: 12595200
Program Execution Time (PET): 8.59 seconds
glza:
Archive Size: 12544000
Program Execution Time (PET): 19.91 seconds
csa:
Archive Size: 13056000
Program Execution Time (PET): 3.72 seconds

brotli seems to be the offender here big time concerning compression time. Even tho from compression benchmarks seen online it should be comparable to xz compression speeds.

Let me test brotli settings of the library i am using

brotli default - no quality parameter given:
Archive Size: 12462080
Program Execution Time (PET): 124.21 seconds
brotli quality=11
Archive Size: 12462080
Program Execution Time (PET): 120.49 seconds

meaning the default of this library is highest compression setting. Let me adjust it for default settings.
some tests
quality = 10
Archive Size: 12687360
Program Execution Time (PET): 42.69 seconds
quality = 9
Archive Size: 13537280
Program Execution Time (PET): 4.87 seconds

quality=9 falls in the range of the other min compressor compression speeds. I will set this as new default for brotli.

recompress whole archive again whith share and all compressors (whole output again per file in full data output section):
Archive Size: 11735040
Program Execution Time (PET): 180.55 seconds

(We not that PET has not been affected that much, which might be because of multiprocesses)
New min compressor list: xz (5x), bsc (2x), ppmd, glza, csa
Compressors: 5
Lets have a look at them, selected by single times min compressors and filtered by min compressors, doing the same play again:

```
  528192 FP.LOG.bsc
  1043598 FP.LOG.csa
  508262 FP.LOG.glza
  659868 FP.LOG.ppmd
  842004 FP.LOG.xz
min File:  /tmp/tmpurlbzgo5/FP.LOG.glza
min File size in bytes:  508262
```

-> bsc closest to glza, new min compressor list: xz (5x), bsc (3x), ppmd, csa

```
  1605468 AcroRd32.exe.bsc
  1348393 AcroRd32.exe.csa
```

1552739 AcroRd32.exe.glza
1541891 AcroRd32.exe.ppmd
1406152 AcroRd32.exe.xz
min File: /tmp/tmpcnkim7lh/AcroRd32.exe.csa
min File size in bytes: 1348393

-> xz closest to csa but difference big enough to consider leaving csa in, lets benchmark afterwards times with csa in or out, otherwise replaced by xz

782132 rafale.bmp.bsc
1153481 rafale.bmp.csa
982910 rafale.bmp.glza
780727 rafale.bmp.ppmd
976448 rafale.bmp.xz
min File: /tmp/tmp0em2u5wz/rafale.bmp.ppmd
min File size in bytes: 780727

-> bsc replaces ppmd. New list: xz (5x), bsc (4x), csa ; otherwise xz (6x), bsc (4x)

lets test it:
xz,bsc,csa:
Archive Size: 11755520
Program Execution Time (PET): 13.88 seconds

xz, bsc:
Archive Size: 11806720
Program Execution Time (PET): 13.55 seconds

Speeds fast enough to leave csa in.

RESULT: Final "fast" profile uses xz, bsc and csa compressors.

--- full data output section ------

output of ls -ls on the testset (decompressed)

```
insgesamt 51916
  824   842468 Dez 25  2000 A10.jpg
 3784  3870784 Mär 27  2001 AcroRd32.exe
 3976  4067439 Feb  2  2003 english.dic
 4424  4526946 Feb 23  2003 FlashMX.pdf
20136  20617071 Feb  4  2003 FP.LOG
 3696  3782416 Sep  1  1997 MSO97.DLL
 4072  4168192 Feb 22  2003 ohs.doc
 4056  4149414 Nov  4  2002 rafale.bmp
 4028  4121418 Okt 27  1995 vcfiu.hlp
 2920  2988578 Apr 20  1996 world95.txt
```

---

output of compression, verbose, default profile, just the file comparisons

```
Archive Size: 11714560
Program Execution Time (PET):  182.83 seconds

drwx------  2     4096  .
drwxrwxrwt 32 root   root        16384  ..
  822838  ohs.doc__4168192.lzham
  835207  ohs.doc.balz
  787574  ohs.doc.brotli
  823642  ohs.doc.bsc
  906814  ohs.doc.bz2
  800476  ohs.doc.csa
  819409  ohs.doc.glza
  1001815  ohs.doc.gz
  1166965  ohs.doc.lz4
  816188  ohs.doc.lzt
  844517  ohs.doc.ppmd
  790732  ohs.doc.xz
  1009146  ohs.doc.zlib
  982036  ohs.doc.zopfli
  905129  ohs.doc.zst
min File:  /tmp/tmpyseaoz04/ohs.doc.brotli
min File size in bytes:  787574

drwx------  2     4096  .
drwxrwxrwt 32 root   root        16384  ..
  624106  world95.txt__2988578.lzham
```

```
  653799  world95.txt.balz
  557311  world95.txt.brotli
  467596  world95.txt.bsc
  577006  world95.txt.bz2
  634312  world95.txt.csa
  494110  world95.txt.glza
  863382  world95.txt.gz
  1515865  world95.txt.lz4
  597888  world95.txt.lzt
  495080  world95.txt.ppmd
  573456  world95.txt.xz
  873561  world95.txt.zlib
  829561  world95.txt.zopfli
  793131  world95.txt.zst
min File:  /tmp/tmpiblh4pii/world95.txt.bsc
min File size in bytes:  467596

drwx------  2     4096  .
drwxrwxrwt 32 root   root       16384  ..
  842003  A10.jpg__842468.lzham
  836321  A10.jpg.balz
  842473  A10.jpg.brotli
  829328  A10.jpg.bsc
  836468  A10.jpg.bz2
  835995  A10.jpg.csa
  847802  A10.jpg.glza
  841824  A10.jpg.gz
  842539  A10.jpg.lz4
  842501  A10.jpg.lzt
  839411  A10.jpg.ppmd
  842108  A10.jpg.xz
  841804  A10.jpg.zlib
  840286  A10.jpg.zopfli
  842498  A10.jpg.zst
min File:  /tmp/tmpqw2tpp1v/A10.jpg.bsc
min File size in bytes:  829328

drwx------  2     4096  .
drwxrwxrwt 32 root   root       16384  ..
  3732083  FlashMX.pdf__4526946.lzham
  3772339  FlashMX.pdf.balz
  3687867  FlashMX.pdf.brotli
  3729218  FlashMX.pdf.bsc
  3810876  FlashMX.pdf.bz2
```

```
  3746453  FlashMX.pdf.csa
  3788939  FlashMX.pdf.glza
  3829660  FlashMX.pdf.gz
  3988964  FlashMX.pdf.lz4
  3728483  FlashMX.pdf.lzt
  3774937  FlashMX.pdf.ppmd
  3699544  FlashMX.pdf.xz
  3832440  FlashMX.pdf.zlib
  3798256  FlashMX.pdf.zopfli
  3822760  FlashMX.pdf.zst
min File:  /tmp/tmpl6y7j_20/FlashMX.pdf.brotli
min File size in bytes:  3687867

drwx------  2     4096  .
drwxrwxrwt 32 root   root       16384  ..
  1900404  MSO97.DLL__3782416.lzham
  1895556  MSO97.DLL.balz
  1848184  MSO97.DLL.brotli
  1930560  MSO97.DLL.bsc
  2110954  MSO97.DLL.bz2
  1828394  MSO97.DLL.csa
  1902244  MSO97.DLL.glza
  2187940  MSO97.DLL.gz
  2950916  MSO97.DLL.lz4
  1864666  MSO97.DLL.lzt
  1915711  MSO97.DLL.ppmd
  1816656  MSO97.DLL.xz
  2189534  MSO97.DLL.zlib
  2117635  MSO97.DLL.zopfli
  2303613  MSO97.DLL.zst
min File:  /tmp/tmpo_sj1x0w/MSO97.DLL.xz
min File size in bytes:  1816656

drwx------  2     4096  .
drwxrwxrwt 32 root   root       16384  ..
  667002  vcfiu.hlp__4121418.lzham
  716321  vcfiu.hlp.balz
  625293  vcfiu.hlp.brotli
  641692  vcfiu.hlp.bsc
  712716  vcfiu.hlp.bz2
  714194  vcfiu.hlp.csa
  695354  vcfiu.hlp.glza
  839458  vcfiu.hlp.gz
  1331347  vcfiu.hlp.lz4
```

```
  673640  vcfiu.hlp.lzt
  691699  vcfiu.hlp.ppmd
  613092  vcfiu.hlp.xz
  847585  vcfiu.hlp.zlib
  795948  vcfiu.hlp.zopfli
  841659  vcfiu.hlp.zst
min File:  /tmp/tmp2n444dwi/vcfiu.hlp.xz
min File size in bytes:  613092

drwx------  2     4096  .
drwxrwxrwt 32 root   root       16384  ..
  1039742  rafale.bmp__4149414.lzham
  1074566  rafale.bmp.balz
  1010238  rafale.bmp.brotli
  782132  rafale.bmp.bsc
  890163  rafale.bmp.bz2
  1153481  rafale.bmp.csa
  982910  rafale.bmp.glza
  1254753  rafale.bmp.gz
  2317235  rafale.bmp.lz4
  1024005  rafale.bmp.lzt
  780727  rafale.bmp.ppmd
  976448  rafale.bmp.xz
  1261662  rafale.bmp.zlib
  1140303  rafale.bmp.zopfli
  1357980  rafale.bmp.zst
min File:  /tmp/tmprcxu3m5m/rafale.bmp.ppmd
min File size in bytes:  780727

drwx------  2     4096  .
drwxrwxrwt 32 root   root       16384  ..
  923525  english.dic__4067439.lzham
  863791  english.dic.balz
  865179  english.dic.brotli
  1183780  english.dic.bsc
  1221689  english.dic.bz2
  925328  english.dic.csa
  928125  english.dic.glza
  1049608  english.dic.gz
  1952339  english.dic.lz4
  876038  english.dic.lzt
  1028120  english.dic.ppmd
  851628  english.dic.xz
  1049443  english.dic.zlib
```

```
  887932  english.dic.zopfli
  1246393  english.dic.zst
min File:  /tmp/tmpn1xv5c4j/english.dic.xz
min File size in bytes:  851628


drwx------  2     4096  .
drwxrwxrwt 32 root   root       16384  ..
  921633  FP.LOG__20617071.lzham
  710813  FP.LOG.balz
  758927  FP.LOG.brotli
  528192  FP.LOG.bsc
  723415  FP.LOG.bz2
  1043598  FP.LOG.csa
  508262  FP.LOG.glza
  1333132  FP.LOG.gz
  2368379  FP.LOG.lz4
  732244  FP.LOG.lzt
  659868  FP.LOG.ppmd
  842004  FP.LOG.xz
  1449455  FP.LOG.zlib
  1258260  FP.LOG.zopfli
  1365597  FP.LOG.zst
min File:  /tmp/tmp_s3c_q95/FP.LOG.glza
min File size in bytes:  508262

drwx------  2     4096  .
drwxrwxrwt 32 root   root       16384  ..
  1501616  AcroRd32.exe__3870784.lzham
  1477648  AcroRd32.exe.balz
  1451351  AcroRd32.exe.brotli
  1605468  AcroRd32.exe.bsc
  1699590  AcroRd32.exe.bz2
  1348393  AcroRd32.exe.csa
  1552739  AcroRd32.exe.glza
  1728149  AcroRd32.exe.gz
  2443192  AcroRd32.exe.lz4
  1434485  AcroRd32.exe.lzt
  1541891  AcroRd32.exe.ppmd
  1406152  AcroRd32.exe.xz
  1732407  AcroRd32.exe.zlib
  1662377  AcroRd32.exe.zopfli
  1822497  AcroRd32.exe.zst
min File:  /tmp/tmpvi8voccq/AcroRd32.exe.csa
```

min File size in bytes: 1348393

Archive Size: 11714560
Program Execution Time (PET): 182.83 seconds

Filtered according to min compressors

  787574 ohs.doc.brotli
  823642 ohs.doc.bsc
  800476 ohs.doc.csa
  819409 ohs.doc.glza
  844517 ohs.doc.ppmd
  790732 ohs.doc.xz
min File: /tmp/tmpyseaoz04/ohs.doc.brotli

  557311 world95.txt.brotli
  467596 world95.txt.bsc
  634312 world95.txt.csa
  494110 world95.txt.glza
  495080 world95.txt.ppmd
  573456 world95.txt.xz
min File: /tmp/tmpiblh4pii/world95.txt.bsc

  842473 A10.jpg.brotli
  829328 A10.jpg.bsc
  835995 A10.jpg.csa
  847802 A10.jpg.glza
  839411 A10.jpg.ppmd
  842108 A10.jpg.xz
min File: /tmp/tmpqw2tpp1v/A10.jpg.bsc

  3687867 FlashMX.pdf.brotli
  3729218 FlashMX.pdf.bsc
  3746453 FlashMX.pdf.csa
  3788939 FlashMX.pdf.glza
  3774937 FlashMX.pdf.ppmd
  3699544 FlashMX.pdf.xz
min File: /tmp/tmpl6y7j_20/FlashMX.pdf.brotli

  1848184 MSO97.DLL.brotli
  1930560 MSO97.DLL.bsc

```
  1828394  MSO97.DLL.csa
  1902244  MSO97.DLL.glza
  1915711  MSO97.DLL.ppmd
  1816656  MSO97.DLL.xz
min File:  /tmp/tmpo_sj1x0w/MSO97.DLL.xz

  625293  vcfiu.hlp.brotli
  641692  vcfiu.hlp.bsc
  714194  vcfiu.hlp.csa
  695354  vcfiu.hlp.glza
  691699  vcfiu.hlp.ppmd
  613092  vcfiu.hlp.xz
min File:  /tmp/tmp2n444dwi/vcfiu.hlp.xz

  1010238  rafale.bmp.brotli
  782132  rafale.bmp.bsc
  1153481  rafale.bmp.csa
  982910  rafale.bmp.glza
  780727  rafale.bmp.ppmd
  976448  rafale.bmp.xz
min File:  /tmp/tmprcxu3m5m/rafale.bmp.ppmd

  865179  english.dic.brotli
  1183780  english.dic.bsc
  925328  english.dic.csa
  928125  english.dic.glza
  1028120  english.dic.ppmd
  851628  english.dic.xz
min File:  /tmp/tmpn1xv5c4j/english.dic.xz

  758927  FP.LOG.brotli
  528192  FP.LOG.bsc
  1043598  FP.LOG.csa
  508262  FP.LOG.glza
  659868  FP.LOG.ppmd
  842004  FP.LOG.xz
min File:  /tmp/tmp_s3c_q95/FP.LOG.glza

  1451351  AcroRd32.exe.brotli
  1605468  AcroRd32.exe.bsc
  1348393  AcroRd32.exe.csa
  1552739  AcroRd32.exe.glza
  1541891  AcroRd32.exe.ppmd
  1406152  AcroRd32.exe.xz
```

min File: /tmp/tmpvi8voccq/AcroRd32.exe.csa

---

output or recompression with default value of brotli now 9

```
drwx------  2     4096  .
drwxrwxrwt 31 root   root       16384  ..
  822838  ohs.doc__4168192.lzham
  835207  ohs.doc.balz
  831755  ohs.doc.brotli
  823642  ohs.doc.bsc
  906814  ohs.doc.bz2
  800476  ohs.doc.csa
  819409  ohs.doc.glza
  1001815  ohs.doc.gz
  816188  ohs.doc.lzt
  844517  ohs.doc.ppmd
  790732  ohs.doc.xz
  1009146  ohs.doc.zlib
  982036  ohs.doc.zopfli
  905129  ohs.doc.zst
min File:  /tmp/tmp3ast715v/ohs.doc.xz
min File size in bytes:  790732

drwx------  2     4096  .
drwxrwxrwt 31 root   root       16384  ..
  624106  world95.txt__2988578.lzham
  653799  world95.txt.balz
  613571  world95.txt.brotli
  467596  world95.txt.bsc
  577006  world95.txt.bz2
  634312  world95.txt.csa
  494110  world95.txt.glza
  863382  world95.txt.gz
  1515865  world95.txt.lz4
  597888  world95.txt.lzt
  495080  world95.txt.ppmd
  573456  world95.txt.xz
  873561  world95.txt.zlib
  829561  world95.txt.zopfli
  793131  world95.txt.zst
min File:  /tmp/tmp5q833fr0/world95.txt.bsc
min File size in bytes:  467596
```

```
drwx------  2     4096  .
drwxrwxrwt 31 root   root       16384  ..
  842003  A10.jpg__842468.lzham
  836321  A10.jpg.balz
  842473  A10.jpg.brotli
  829328  A10.jpg.bsc
  836468  A10.jpg.bz2
  835995  A10.jpg.csa
  847802  A10.jpg.glza
  841824  A10.jpg.gz
  842539  A10.jpg.lz4
  842501  A10.jpg.lzt
  839411  A10.jpg.ppmd
  842108  A10.jpg.xz
  841804  A10.jpg.zlib
  840286  A10.jpg.zopfli
  842498  A10.jpg.zst
min File:  /tmp/tmpyjlvnhsl/A10.jpg.bsc
min File size in bytes:  829328

drwx------  2     4096  .
drwxrwxrwt 31 root   root       16384  ..
  3732083  FlashMX.pdf__4526946.lzham
  3772339  FlashMX.pdf.balz
  3739955  FlashMX.pdf.brotli
  3729218  FlashMX.pdf.bsc
  3810876  FlashMX.pdf.bz2
  3746453  FlashMX.pdf.csa
  3788939  FlashMX.pdf.glza
  3829660  FlashMX.pdf.gz
  3988964  FlashMX.pdf.lz4
  3728483  FlashMX.pdf.lzt
  3774937  FlashMX.pdf.ppmd
  3699544  FlashMX.pdf.xz
  3832440  FlashMX.pdf.zlib
  3798256  FlashMX.pdf.zopfli
  3822760  FlashMX.pdf.zst
min File:  /tmp/tmpf3xyi0bo/FlashMX.pdf.xz
min File size in bytes:  3699544

drwx------  2     4096  .
drwxrwxrwt 31 root   root       16384  ..
  1900404  MSO97.DLL__3782416.lzham
```

```
  1895556  MSO97.DLL.balz
  2059040  MSO97.DLL.brotli
  1930560  MSO97.DLL.bsc
  2110954  MSO97.DLL.bz2
  1828394  MSO97.DLL.csa
  1902244  MSO97.DLL.glza
  2187940  MSO97.DLL.gz
  2950916  MSO97.DLL.lz4
  1864666  MSO97.DLL.lzt
  1915711  MSO97.DLL.ppmd
  1816656  MSO97.DLL.xz
  2189534  MSO97.DLL.zlib
  2117635  MSO97.DLL.zopfli
  2303613  MSO97.DLL.zst
min File:  /tmp/tmp63vya5bn/MSO97.DLL.xz
min File size in bytes:  1816656

drwx------  2     4096  .
drwxrwxrwt 31 root   root       16384  ..
  667002  vcfiu.hlp__4121418.lzham
  716321  vcfiu.hlp.balz
  706235  vcfiu.hlp.brotli
  641692  vcfiu.hlp.bsc
  712716  vcfiu.hlp.bz2
  714194  vcfiu.hlp.csa
  695354  vcfiu.hlp.glza
  839458  vcfiu.hlp.gz
  1331347  vcfiu.hlp.lz4
  673640  vcfiu.hlp.lzt
  691699  vcfiu.hlp.ppmd
  613092  vcfiu.hlp.xz
  847585  vcfiu.hlp.zlib
  795948  vcfiu.hlp.zopfli
  841659  vcfiu.hlp.zst
min File:  /tmp/tmp26mdv3wg/vcfiu.hlp.xz
min File size in bytes:  613092

drwx------  2     4096  .
drwxrwxrwt 31 root   root       16384  ..
  1039742  rafale.bmp__4149414.lzham
  1074566  rafale.bmp.balz
  1135314  rafale.bmp.brotli
  782132  rafale.bmp.bsc
  890163  rafale.bmp.bz2
```

```
  1153481  rafale.bmp.csa
  982910  rafale.bmp.glza
  1254753  rafale.bmp.gz
  2317235  rafale.bmp.lz4
  1024005  rafale.bmp.lzt
  780727  rafale.bmp.ppmd
  976448  rafale.bmp.xz
  1261662  rafale.bmp.zlib
  1140303  rafale.bmp.zopfli
  1357980  rafale.bmp.zst
min File:  /tmp/tmp0em2u5wz/rafale.bmp.ppmd
min File size in bytes:  780727

drwx------  2     4096  .
drwxrwxrwt 31 root   root       16384  ..
  923525  english.dic__4067439.lzham
  863791  english.dic.balz
  1048240  english.dic.brotli
  1183780  english.dic.bsc
  1221689  english.dic.bz2
  925328  english.dic.csa
  928125  english.dic.glza
  1049608  english.dic.gz
  1952339  english.dic.lz4
  876038  english.dic.lzt
  1028120  english.dic.ppmd
  851628  english.dic.xz
  1049443  english.dic.zlib
  887932  english.dic.zopfli
  1246393  english.dic.zst
min File:  /tmp/tmpj9ie92hn/english.dic.xz
min File size in bytes:  851628

drwx------  2     4096  .
drwxrwxrwt 31 root   root       16384  ..
  921633  FP.LOG__20617071.lzham
  710813  FP.LOG.balz
  893851  FP.LOG.brotli
  528192  FP.LOG.bsc
  723415  FP.LOG.bz2
  1043598  FP.LOG.csa
  508262  FP.LOG.glza
  1333132  FP.LOG.gz
  2368379  FP.LOG.lz4
```

```
  732244  FP.LOG.lzt
  659868  FP.LOG.ppmd
  842004  FP.LOG.xz
  1449455  FP.LOG.zlib
  1258260  FP.LOG.zopfli
  1365597  FP.LOG.zst
min File:  /tmp/tmpurlbzgo5/FP.LOG.glza
min File size in bytes:  508262

drwx------  2     4096  .
drwxrwxrwt 31 root   root        16384  ..
  1501616  AcroRd32.exe__3870784.lzham
  1477648  AcroRd32.exe.balz
  1644087  AcroRd32.exe.brotli
  1605468  AcroRd32.exe.bsc
  1699590  AcroRd32.exe.bz2
  1348393  AcroRd32.exe.csa
  1552739  AcroRd32.exe.glza
  1728149  AcroRd32.exe.gz
  2443192  AcroRd32.exe.lz4
  1434485  AcroRd32.exe.lzt
  1541891  AcroRd32.exe.ppmd
  1406152  AcroRd32.exe.xz
  1732407  AcroRd32.exe.zlib
  1662377  AcroRd32.exe.zopfli
  1822497  AcroRd32.exe.zst
min File:  /tmp/tmpcnkim7lh/AcroRd32.exe.csa
min File size in bytes:  1348393

Archive Size: 11735040
Program Execution Time (PET):  180.55 seconds
```

Default

Describe here what you did and why you did it: The creators of these applications are the experts, not me. They provide default values, which they will have carefully implemented or chosen as good values for usability, so I stick with those.
But could not find for blabla for lzturbo

zopfli calibrating numiterations silesia:ooffice file

They suggest on docu different values. But i am optimizing size with fast decompression, not compression speeds:

5:
6152192 --> 2992712 ooffice.zopfli
Program Execution Time (PET): 9.81 seconds

15:
6152192 --> 2992119 ooffice.zopfli
Program Execution Time (PET): 15.56 seconds

25:
6152192 --> 2992056 ooffice.zopfli
Program Execution Time (PET): 21.46 seconds

50:
6152192 --> 2991918 ooffice.zopfli
Program Execution Time (PET): 35.55 seconds

100:
6152192 --> 2991701 ooffice.zopfli
Program Execution Time (PET): 63.53 seconds

200:
6152192 --> 2991640 ooffice.zopfli
Program Execution Time (PET): 119.2 seconds

1000:
6152192 --> 2991464 ooffice.zopfli
Program Execution Time (PET): 573.94 seconds

...
Where is the end to incresing computational cost = smaller file size?

Such which value should I choose? I am optimizing for decompression and size not compression time, but a compressor that never finishes (on strong settings) also is of no practical value.
First time I encounter an unlimited range compression level setting for a compressor. hm. What value should one choose here?

glza ooffice
-x
6152192 --> 2492198 ooffice.glza
Program Execution Time (PET): 16.04 seconds

-x -p0.0
6152192 --> 2495065 ooffice.glza
Program Execution Time (PET): 14.12 seconds

-x -p3
6152192 --> 2515226 ooffice.glza
Program Execution Time (PET): 27.41 seconds

-x -p10
6152192 --> 2554776 ooffice.glza
Program Execution Time (PET): 41.53 seconds

ill just take -x alone

Strong

For this Profile, I left everything the same, but if the compressor provided some max compression option, like -cx instead of -c alone, I will call the compressors with that option. Or just the highest level compression option they provide, if for example the compression level expect an integer, just the highest integer allowed.

Max

For the Max Profile, we lift the limitation on decompression time. So basically any compressor were allowed and executed together with the other compressors, and as designed, always the smallest compressed version of the file will be added to the archive, taking into account those compressed with stronger, also symmetrical, compressors. In this version, ZPAQ and NNCP are integrated additionally.

Benchmarks

See the following sections:

Quick Size Comparison

This is just a quick comparison I did. I selected all files in this corpus, right clicked on my Ubuntu20, and there are multiple compression options. I compressed them just with each available option that was listed. Then I added the share archive created with the same input data in default profile. This is the output of listing all archives with their file sizes, sorted by size:

```
11735040 maximumcompressioncorpus.share
12177048 maximumcompressioncorpus.7z
12403504 maximumcompressioncorpus.tar.xz
12412975 maximumcompressioncorpus.tar.lzma
12412988 maximumcompressioncorpus.tar.lz
12523696 maximumcompressioncorpus.tar.7z
12654720 maximumcompressioncorpus.exe
13610427 maximumcompressioncorpus.tar.bz2
14405261 maximumcompressioncorpus.zip
14405261 maximumcompressioncorpus.cbz
15095465 maximumcompressioncorpus.tar.gz
15102263 maximumcompressioncorpus.war
15102263 maximumcompressioncorpus.jar
15102263 maximumcompressioncorpus.ear
15102263 maximumcompressioncorpus.crx
15102263 maximumcompressioncorpus.apk
15311263 maximumcompressioncorpus.tar.zst
20907094 maximumcompressioncorpus.tar.lz4
21334955 maximumcompressioncorpus.tar.lzo
21427739 maximumcompressioncorpus.tar.Z
53135336 maximumcompressioncorpus.ar
53136148 maximumcompressioncorpus.cpio
53144064 maximumcompressioncorpus.tar
```

53522432 maximumcompressioncorpus.iso

System Specs

Memory: 15.6 GiB
Processor: AMD® Ryzen 5 3600 6-core processor × 12
Graphics: NV168
Disk Capacity: 2.3 TB
OS Name: Ubuntu 20.04.3 LTS
OS Type: 64-bit
GNOME Version: 3.36.8
Windowing System: X11

All Profiles

Benchmarking results
Maximumcompression corpus

Compression, Fast:
Original Size: 53134726
Archive Size: 11755520
Compression Ratio: 4.5199809110953835
Program Execution Time (PET): 11.13 seconds
Decompression:
Program Execution Time (PET): 0.36 seconds

Compression, Default:
Original Size: 53134726
Archive Size: 11735040
Compression Ratio: 4.527869184936736
Program Execution Time (PET): 156.91 seconds
Decompression:
Program Execution Time (PET): 0.34 seconds

Compression, Strong:
Original Size: 53134726
Archive Size: 11376640
Compression Ratio: 4.670511328476597
Program Execution Time (PET): 254.39 seconds
Decompression:
Program Execution Time (PET): 0.31 seconds

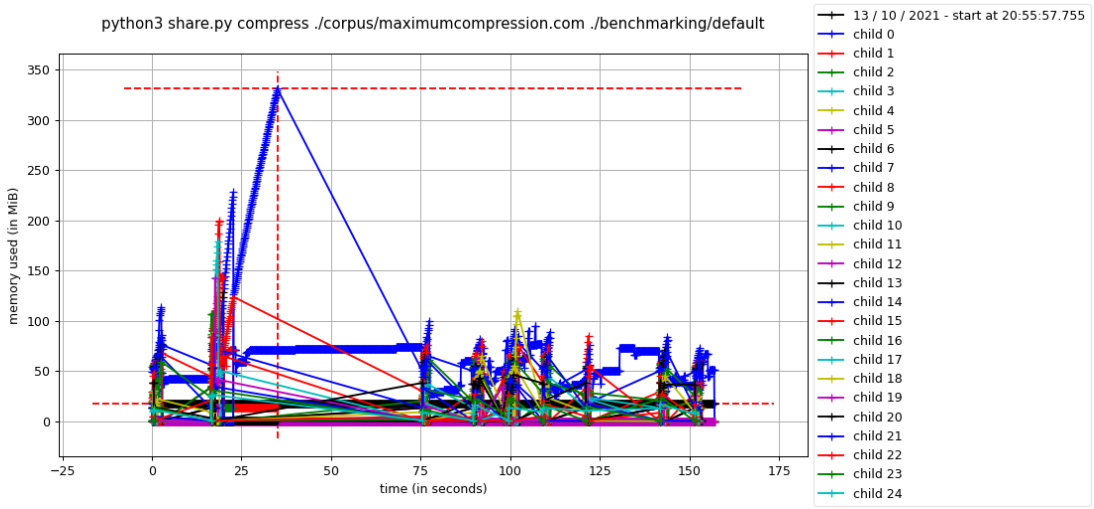
Compression, Max:

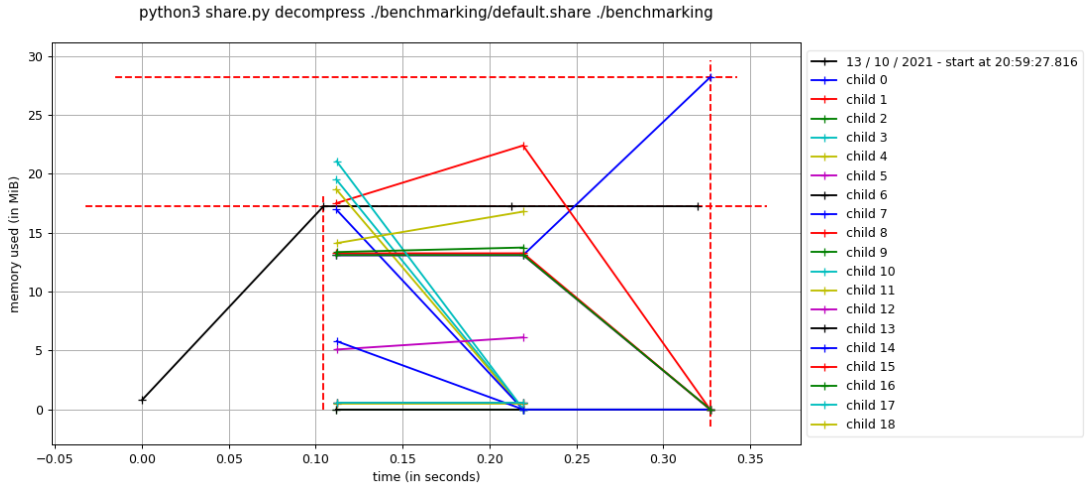
Original Size: 53134726
Archive Size: 11100160
Compression Ratio: 4.786843252709871
Program Execution Time (PET): 16464.23 seconds
Decompression:
Program Execution Time (PET): 6479.34 seconds

See Plots for additional Info.
Also see plots for memory saver effect (*ms*). It simply splits the number of concurrent processes to reduce memory usage. Results in longer compression time. For when running out of memory, will not effect the strong compressors like nncp (and zpaq; both only in max profile) since i do not let them run concurrently because of memory resource demands.

Default Profile

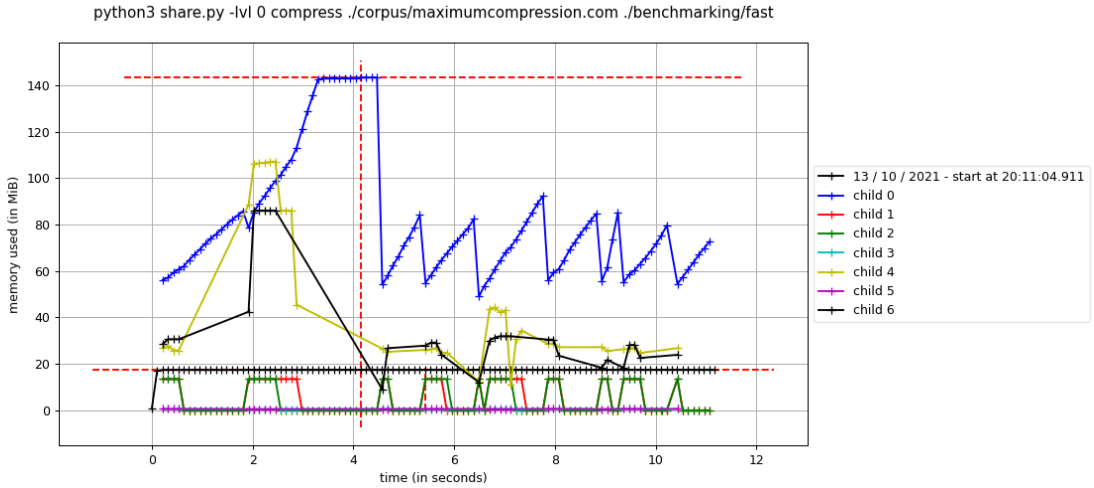
Compression, Default:
Original Size: 53134726
Archive Size: 11735040
Compression Ratio: 4.527869184936736
Program Execution Time (PET): 156.91 seconds
Decompression:
Program Execution Time (PET): 0.34 seconds

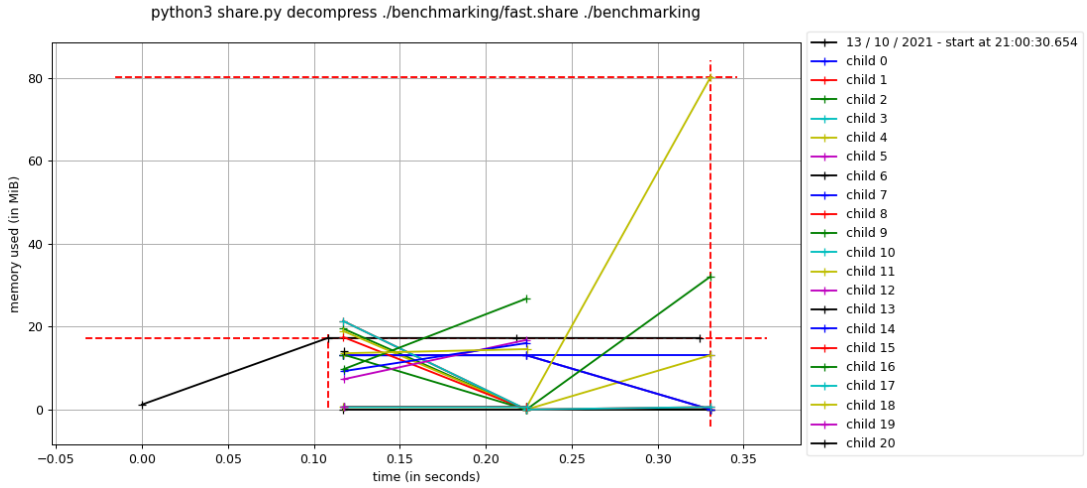




Fast Profile

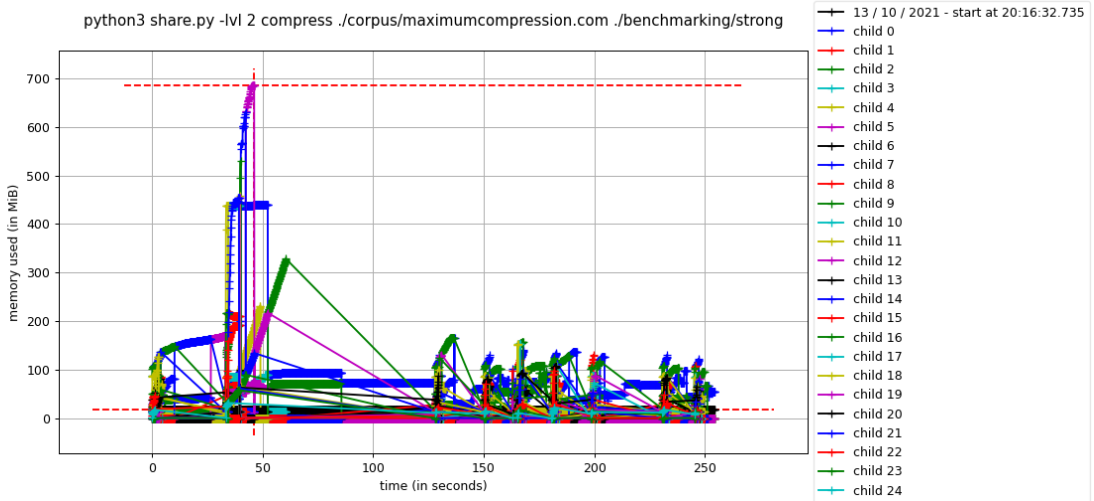
Compression, Fast:
Original Size: 53134726
Archive Size: 11755520
Compression Ratio: 4.5199809110953835
Program Execution Time (PET): 11.13 seconds
Decompression:
Program Execution Time (PET): 0.36 seconds

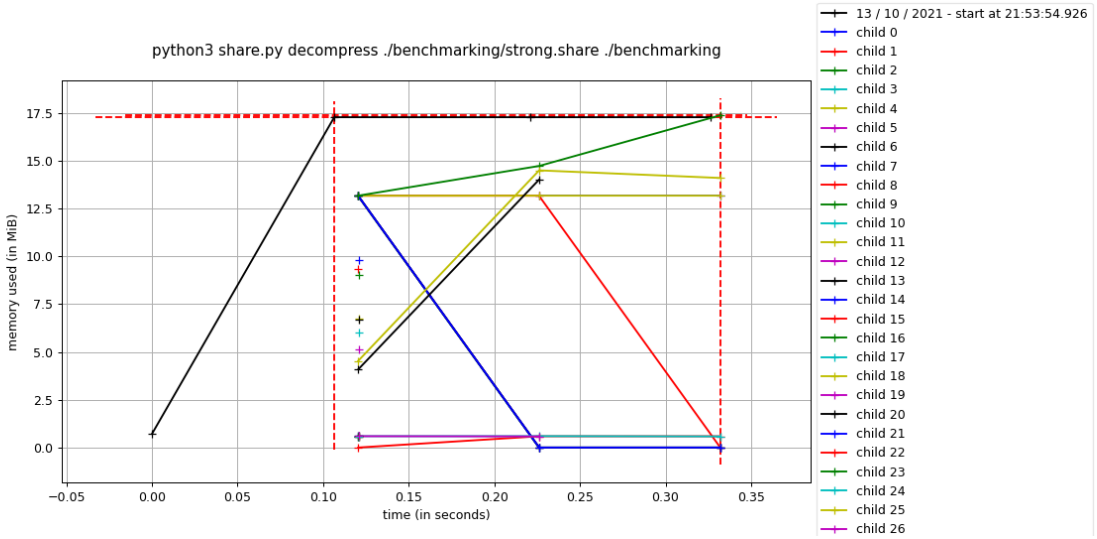




Strong Profile

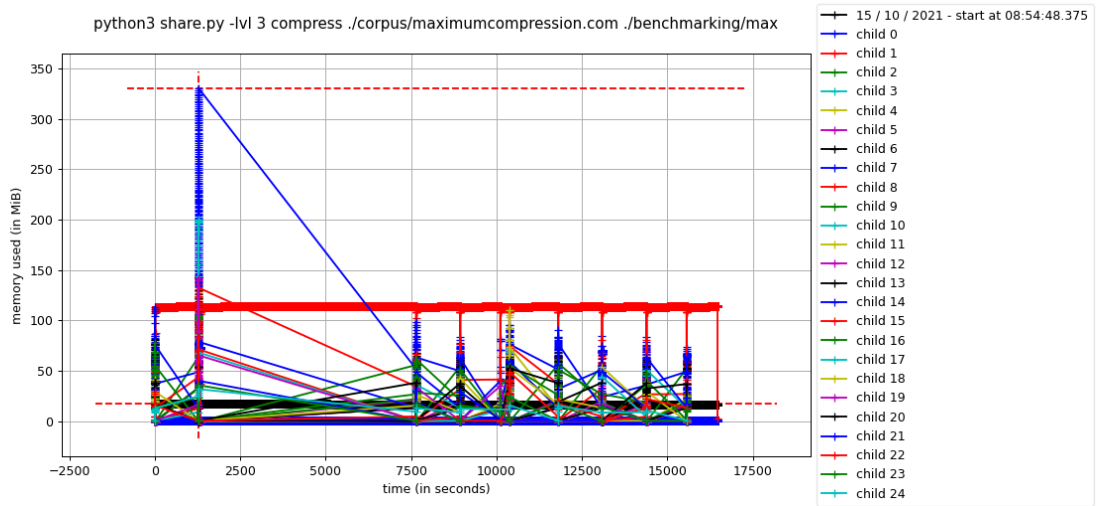
Compression, Strong:
Original Size: 53134726
Archive Size: 11376640
Compression Ratio: 4.670511328476597
Program Execution Time (PET): 254.39 seconds
Decompression:
Program Execution Time (PET): 0.31 seconds

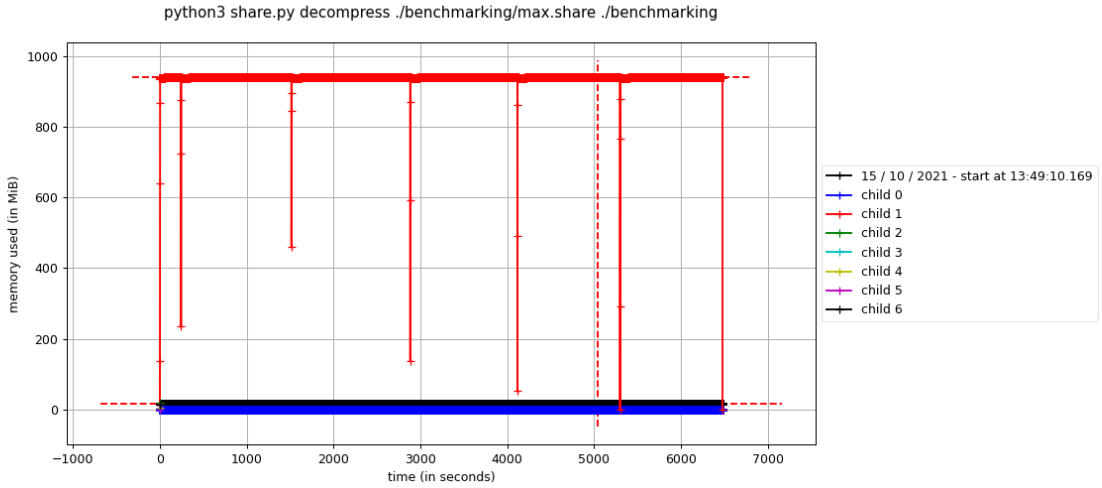




Max Profile

Compression, Max:
Original Size: 53134726
Archive Size: 11100160
Compression Ratio: 4.786843252709871
Program Execution Time (PET): 16464.23 seconds
Decompression:
Program Execution Time (PET): 6479.34 seconds




python3 share.py decompress ./benchmarking/max.share ./benchmarking
memory used (in MiB)
time (in seconds)
15 / 10 / 2021 - start at 13:49:10.169
child 0
child 1
child 2
child 3
child 4
child 5
child 6

Multiprocessing

To improve compression as well as decompression times, I used python multiprocessing to start the compression or decompression of each individual file concurrently. I then tested the results by compression the Silesia Corpus on a System available (same system for both times of course) with share archiver, default profile:

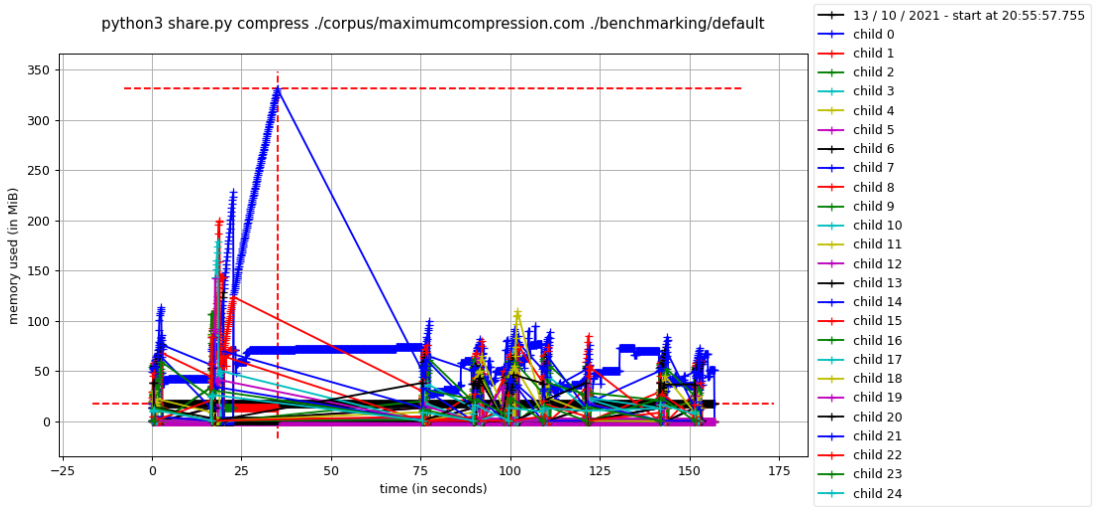
Synchronous (previous approach):
Compression:
Archive Size: 43827200
Program Execution Time (PET): 1878.27 seconds
Decompression:
Program Execution Time (PET): 5.27 seconds

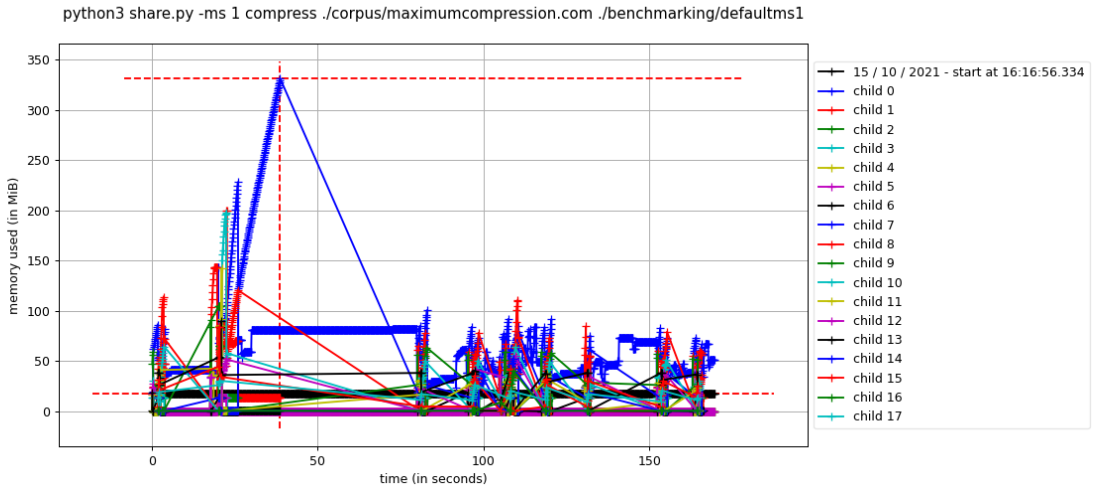
With multiprocessing (new):
Compression:
Archive Size: 43827200
Program Execution Time (PET): 959.12 seconds
Decompression:
Program Execution Time (PET): 1.58 seconds

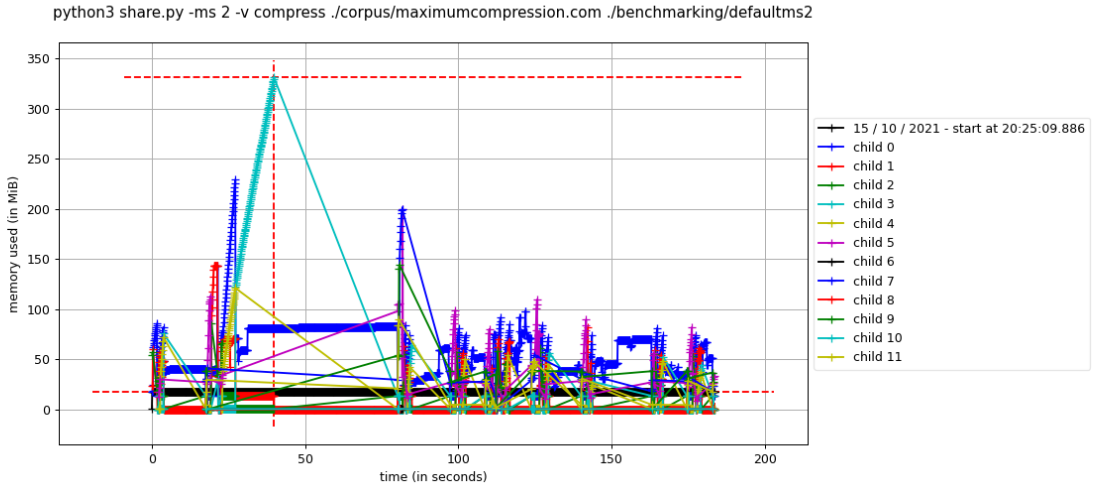
When executing and testing this, as expected, the memory usage was a lot higher. When using the Max Profile, it would start those strong compressors with greater requirements (like cmixes RAM requirements) simultaneously with all the other compressors, which gave me an out of memory error. Therefore the additional compilers activated when using the max profile will run only synchronously, after all the other compressors have run concurrently. I additionally implemented and provide a memory saver option, to limit this phenomenon by basically reverting multiprocessing, or in other words, to which level multiprocessing will be applied.

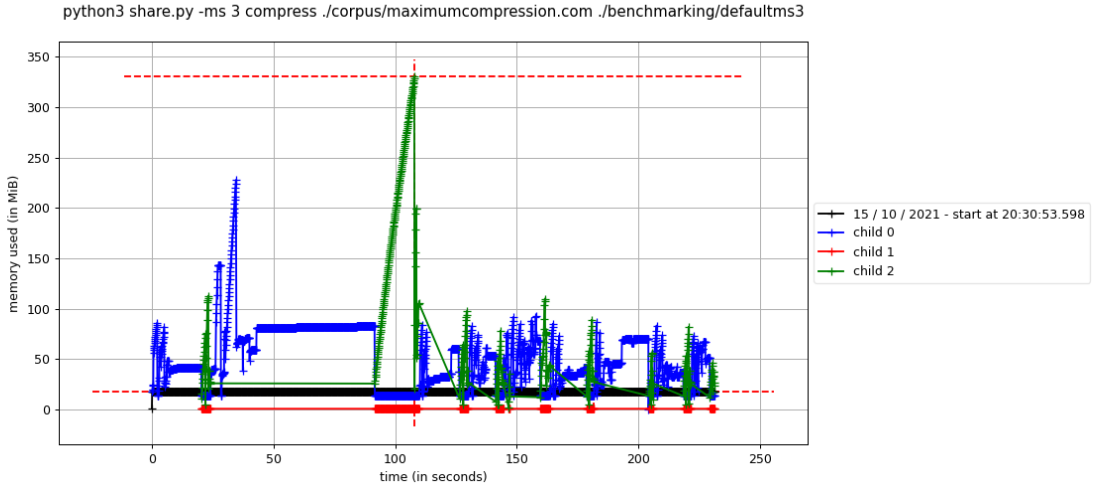
Memory Saver

When just starting all the compressors for a single file at the same time, I encountered an out of memory error. When taking out the strong compressors to run synchronosuly, Then I thought, since some systems might have limited RAM available; someone else encountering an our of memory error, I'd provide an option so decrease the number of parallel started compressors which should decrease RAM requirements. So i provide the memory saver option -ms . It simply takes all these compressors, but only starts half of them together, and waits for them to finish, then starts the other half. On the next level, it will half it again, so start the first quarter of compressors together, and so fort. The highest level lets the application run in synchronous mode, so no parallelisation at all. Of course this will affect compression and decompression times (as we can see in the graphics in level 3 with compression time)






python3 share.py -ms 2 -v compress ./corpus/maximumcompression.com ./benchmarking/defaultms2
memory used (in MiB)
time (in seconds)
15 / 10 / 2021 - start at 20:25:09.886
child 0
child 1
child 2
child 3
child 4
child 5
child 6
child 7
child 8
child 9
child 10
child 11


python3 share.py -ms 3 compress ./corpus/maximumcompression.com ./benchmarking/defaultms3
memory used (in MiB)
time (in seconds)
15 / 10 / 2021 - start at 20:30:53.598
child 0
child 1
child 2

Random Notes compressor selection

http://www.mattmahoney.net/dc/10gb.html (22.sept.2021) 10GB compression benchmark, we are looking for fast decompression (low extract number) while having good compression (low size number) this if for 10GB, which might be edge case, or maybe not, depending on user base for this program. System 4 has most benchmarks. In high compression tiers, decompression is way to slow for the purpose of this program. Acceptable entries, extraction time < 1000, sorting by filesize (compression) and then only listing entries that are faster decompressing for that sorting:

Size	Compress	Extract	Program with options
2893742274	1756*	838*	pcompress 3.-l14 -s60
2933199750	9520	412*	packet 1.9 -mx -h8 -b5 -r -s
2954727653	3568	328*	packet 1.2 -r -mx -b512 -h4
3112349650	8227	315*	lza 0.82b -mx9 -h7 -b7 -r -s
3129833887575	315	lza 0.80	-mx9 -h7 -b7 -r -s
3131331323	641	302*	lza_x64 0.70b -mx9 -h7 -b7 -r -s
3161399085	4746	296*	lza_x64 0.63 -mx5 -h8 -b7 -r -s
3186833109	2669	189*	pcompress 2.4 -G -L -P -c lzmaMt -
365186259491*	171*	pcompress 2.4	-G -L -P -c zlib -

http://www.mattmahoney.net/dc/silesia.html Silesia Open Source Compression Benchmark, maybe closer to use case. DOes not incluse speed, so this lists worth for this program is limited. Only information we extract is when speed does not matter, raw compression only, cmix seems to be the winner, even more so when combined with precomp. Keep in mind cmix has huge memory requirements, otherwise it simply cannot run. The information we can extract is powerful compression program names (sorted again after filesize and new compression algorithm/program name)

Total size	program
28437634	precomp v0.4.7 -cn | cmix v18
30088099	paq8pxd_v58 -15
35336837	paq8l -8
35457761	fp8_v4 -8
36603712	precomp v0.4.4 -cn | zpaq 7.05 -method 7
37809279	tangelo 1.0
40068926	mcm 0.82 -9 -max
...	
211938580	Uncompressed

We cannot evaluate the practicality of this list. If decompression takes a long time, it is impractical for the purpose of this program. But we can add a max compression button, which might feature a program of this list

Results sorted on Compression ratio.

Pos	Program	Switches used	TAR	Q6600	Compressed Size (bytes)	Compression Ratio (%)	Comp time (sec)	Decomp time (sec)	Efficiency lower is better
001	PAQ8px	-7	N	Y	62443738	80.26	23427	22019	45446
002	PAQ8P	-7	N	Y	62683172	80.19	17976	16653	35562
003	WinRK 3.1.2	Maximum	N	Y	64294243	79.68	14265	13985	34692
004	PAQ8JC	-6 (dir /oen)	N	N	66191139	79.08	18908	16212	53237
005	PAQAR 4.5	-7e (dir /oen/b)	N	N	68435715	78.37	21322	21410	83103
006	WinRK 3.1.2	High	N	Y	69003458	78.19	1311	980	4745
007	PAsQDa 4.4	-7e	N	N	69109187	78.15	21101	21173	88594
008	NanoZip 0.09a	-cc -m800m	N	Y	69857803	77.92	464	438	2054
009	UDA 0.301	(none)	Y	Y	71478767	77.41	3369	3412	18487
010	DURILCA 0.5	-m800 -t1	N	Y	72175578	77.19	738	657	4109
011	Slim 0.23d	-m800 -o64	N	Y	73664683	76.71	1574	1529	10783
012	WinUDA 0.291	mode-3	N	Y	73735574	76.69	3663	3691	25756
013	Slim 0.23d	(none)	N	Y	73781972	76.68	1555	1486	10706
014	KGB 1.1 (PAQ6)	(very good)	N	Y	75341375	76.18	7830	7889	65796
015	NanoZip 0.09a	-cO	N	Y	75720696	76.06	208	45.3	1106
016	ZPAQ 2.05	max.cfg	N	Y	76328559	75.87	1107	1111	10359
017	CCMx 1.30c	6	Y	Y	77079949	75.64	130	126	1300
018	DURILCA 0.5	(none)	N	Y	77604453	75.47	947	951	10213
019	PPMonstr J	-m800 -o32	N	Y	77699686	75.44	587	593	6417
020	CCM 1.30c	6	Y	Y	77926922	75.37	107	103	1171
021	UHARC 0.6b	-mx -md32768 -mm+	N	Y	78344876	75.24	278	234	2991
022	FreeARC 0.666	-mx	N	Y	78406220	75.22	83.9	33.5	691
023	PPMonstr J	(none)	N	Y	78443862	75.20	568	573	6739
024	CCM 1.30c	(none)	Y	Y	78598980	75.15	102	98	1202
025	LPAQ8	8	Y	Y	78704186	75.12	504	507	6147
026	WinRK 3.1.2	Normal	N	Y	78882890	75.07	157	33.5	1181
027	PIMPLE2	(none)	Y	N	79628157	74.83	2195	2207	29654
028	WinUDA 0.291	mode-0	N	Y	79710669	74.80	1178	1180	16031
029	NanoZip 0.09a	(none)	N	Y	79905525	74.74	58.4	14.1	504
030	COMPRESSIA 1.0b	max sol dict 15 Mb	N	N	80698340	74.49	803	99999	764699
031	FreeARC 0.666	(none)	N	Y	81234951	74.32	24.1	11.8	289
032	DURILCA light 0.5	-m800 -t1	N	Y	81666474	74.19	155	76	1951
033	EPM r9	-m800	Y	Y	81934909	74.10	980	988	17126
034	CMM4 0.2b	26	Y	Y	82435458	73.94	302	307	5603
035	EPM r9	(none)	Y	Y	82727852	73.85	903	904	17172
036	Ocamyd 1.66 test1	-m8 -s0	Y	N	82880385	73.80	4666	4664	90177
037	BIT 0.7	(none)	Y	Y	83321888	73.66	305	298	6121
038	SQUEEZ 5.63	Ultra 32Mb M E Solid	N	Y	83565058	73.59	181	29.6	2196
039	Quark 0.95r	-l7 -d25	Y	N	83673567	73.55	2095	38	22513
040	RZM 0.07h	(none)	Y	Y	84030374	73.44	244	19.9	2898
041	SBC 0.970 rev3	-m3 -b62	N	Y	84736176	73.21	102	40	1686
042	Quark 0.95r	(none)	Y	N	84741314	73.21	702	37	8782
043	Ultra7z Opt 0.05	(none)	N	Y	84932335	73.15	406	40.2	5416
044	UHARC 0.6b	(none)	N	Y	84954506	73.15	209	33	2945
045	SBC 0.970 rev3	-b9	N	Y	85740265	72.90	60	33	1235
046	FreeARC 0.666	-m3	N	Y	85905882	72.85	13.6	10.5	326
047	ASH 07	-m800 -o60	Y	Y	86225632	72.74	3641	3639	102003
048	SBC 0.970 rev3	(none)	N	Y	86245288	72.74	57	33	1264
049	WINZIP 14	Zipx Best Method	N	Y	86276462	72.73	95	27.5	1726
050	WinRAR 4.1b3	Best solid	N	Y	86658210	72.61	48.7	22.5	1047
051	KGB 1.1 (PAQ6)	(normal)	N	Y	86878458	72.54	1133	1157	34498
052	7-Zip 9.25a	Ultra LZMA2 64Mb 3thr	N	Y	87198046	72.44	91.1	7.8	1544

```
score_X = POWER(2; ((size_X / size_TOP) - 1) / 0,1) * time_X

with  score_X     efficiency score for a certain compressor X
      time_X      time elapsed by compressor X (comp + decomp time)
      size_X      archive size achieved with compressor X
      size_TOP    archive size by top archiver (smallest benchmark result)
```

$$score_x = 2^{\left(\frac{\left(\left(\frac{size_x}{size_{top}}\right)-1\right)}{0,1}\right)} . time_x$$

```
"0,1" represents 10% and power of 2 ensures that for each 10% worse results (compared
with top) the time is doubled, so any archiver (except top compressor) will get a
penalty on time. The score of top compressor is always equal to its time value.
```

404 Not Found

Results sorted on compressor efficiency

Pos	Program	Switches used	TAR	Q6600	Compressed	Compression	Comp time	Decomp time	Efficiency
					Size (bytes)	Ratio (%)	(sec)	(sec)	lower is better
000	TESTSET		N	Y	316355757	0.00	0	0	0
001	FreeARC 0.666	(none)	N	Y	81234951	74.32	24.1	11.8	289
002	FreeARC 0.666	-m3	N	Y	85905882	72.85	13.6	10.5	326
003	FreeARC 0.666	-m2	N	Y	91650028	71.03	6.9	7.8	376
004	NanoZip 0.09a	(none)	N	Y	79905525	74.74	58.4	14.1	504
005	NanoZip 0.09a	-cDp	N	Y	90638348	71.35	20.7	9.1	681
006	FreeARC 0.666	-mx	N	Y	78406220	75.22	83.9	33.5	691
007	WinRAR 4.1b3	Normal solid	N	Y	90361866	71.44	36.2	4.7	907
008	WinRAR 4.1b3	Normal	N	Y	91995231	70.92	29.7	4.6	912
009	7-Zip 9.25a	Norm LZMA2 16Mb ws=32 4thr	N	Y	88580416	72.00	49.2	7.1	1024
010	WinRAR 4.1b3	Best solid	N	Y	86658210	72.61	48.7	22.5	1047
011	NanoZip 0.09a	-cO	N	Y	75720696	76.06	208	45.3	1106
012	CCM 1.30c	6	Y	Y	77926922	75.37	107	103	1171
013	WinRK 3.1.2	Normal	N	Y	78882890	75.07	157	33.5	1181
014	FlashZIP 0.99b8	-m1	N	Y	94690080	70.07	20.2	12.9	1187
015	CCM 1.30c	(none)	Y	Y	78598980	75.15	102	98	1202
016	SBC 0.970 rev3	-b9	N	Y	85740265	72.90	60	33	1235
017	SBC 0.970 rev3	(none)	N	Y	86245288	72.74	57	33	1264
018	WinRK 3.1.2	Fastest	N	Y	90821411	71.29	30.9	23.7	1274
019	CCMx 1.30c	6	Y	Y	77079949	75.64	130	126	1300
020	CSC 3.2a6	-m3	Y	Y	93295468	70.51	35.8	9.2	1382
021	7-Zip 9.25a	Ultra LZMA2 64Mb 3thr	N	Y	87198046	72.44	91.1	7.8	1544
022	SBC 0.970 rev3	-m3 -b62	N	Y	84736176	73.21	102	40	1686
023	WINZIP 14	Zipx Best Method	N	Y	86276462	72.73	95	27.5	1726
024	WinACE 2.69	Norm solid ace20	N	Y	89742715	71.63	74	12.3	1787
025	DURILCA light 0.5	-m800 -t1	N	Y	81666474	74.19	155	76	1951
026	NanoZip 0.09a	-cc -m800m	N	Y	69857803	77.92	464	438	2054
027	StuffIt 14.0	Auto filetype L16 M30 BP E	N	Y	90125515	71.51	30.1	67	2098
028	WinACE 2.69	Max 4096Kb solid ace20	N	Y	88485215	72.03	108	12.3	2166
029	ACE 2.6	-m5 -d4096	N	Y	88486031	72.03	109	12.9	2195

Sep 22 2021 - 11:30 vorm.
Revision 8.2 (Web Version)
Test Machine Specifications:
* Intel Core i7 (Sandy Bridge)
* 16 GB RAM (DDR-3 @ 1600 MHz), SSD (SATA-3)

info@squeezechart.com
squeezechart@gmx.de

Squeeze Chart 2018

© **practical chart**

TAR is used when the program cannot compress all files of a testset in a single archive
or when recurse through subdirectories does not work properly.
tools slower than 25.000 sec are not tested on this chart.

http://ftp.sac.sk/pub/pc/pack/
http://mattmahoney.net/dc/text.html
http://ftp.elf.stuba.sk/pub/pc/pack/
http://ftp.sac.sk/sac/pack/
ftp://ftp.sac.sk/pub/sac/pack

	THE TEST DATASET'S BEST Bits Per Byte value ⇒ BEST ratio ⇒ UNCOMPRESSED SIZE	test set type	% weighted average	encoding time in seconds	decoding time in seconds	APP 1.6669 20.8364% 327'811'342	AUDIO 3.1864 39.8296% 606'681'890	CAMERA RAW 6.1819 77.2734% 577'835'051	GUTENBERG 2.9956 37.4448% 709'240'453	INSTALLER 6.4956 81.1945% 609'139'052	MOBILE 7.1224 89.0296% 492'103'406	PGM / PPM 3.3127 41.4085% 633'482'445	SOURCES 1.2217 15.2718% 200'001'544	XML 1.1320 14.1494% 1'000'000'000	usual file extension	architecture OS bit depth	uses GPU	cores used	when released / developed	where developed (country)	algorithms used	program age in years
GUI	NanoZip 0.09 -cc -m2g (2011) Sami Runsas		###	6'687	6'653	69'699'825	344'830'935	454'082'848	273'648'068	572'487'400	478'618'331	273'648'831	31'322'350	149'750'763	.nz	win64	no	4	4-Nov-2011	Finland	Fast CM + filter	9.88
GUI	NanoZip 0.08 -cc -m2g (2010) Sami Runsas		###	6'933	6'738	69'728'614	344'830'061	454'178'452	271'181'957	572'480'489	478'680'399	273'691'671	31'341'330	150'005'854	.nz	win64	no	1	3-Jun-2010	Finland	Fast CM + filter	11.30
•	MCM 0.83 -x11 (2015) Mathieu Chartier ☺	TAR	###	4'467	5'405	70'631'322	396'424'812	455'006'834	274'694'277	574'227'512	481'705'736	289'906'777	30'543'764	141'494'143	.mcm	win64	no	1	5-Apr-2015	Canada	Fast CM + LZP	6.46
•	MCM 0.83 -x9 (2015) Mathieu Chartier ☺	TAR	###	4'049		71'336'984	396'424'812	455'203'653	275'813'829	576'448'230	481'809'823	290'436'539	30'650'778	145'800'236	.mxm	win64	no	1	5-Apr-2015	Canada	Fast CM + LZP	6.46
•	ZCM 0.93 -m8 -r -s (2015) Francesco Nania		###	2'996	2'646	74'369'138	373'127'042	461'946'290	284'122'457	575'669'657	483'167'730	263'150'037	32'644'513	159'466'123	.zcm	win64	no	1	12-May-2015	Italy	Fast CM + filter	6.36
•	ZCM 0.88 -m7 -r -s (2013) Francesco Nania		###	2'673	2'345	75'418'609	374'148'733	459'900'610	286'929'481	574'442'060	483'725'771	264'529'873	33'321'514	163'973'392	.zcm	win32	no	1	21-Jun-2013	Italy	Fast CM + filter	8.25
•	ZCM 0.92 -m7 -r -s (2014) Francesco Nania		###	2'323	1'977	75'359'696	374'153'977	466'996'270	287'686'468	575'595'855	483'925'057	266'039'219	33'334'683	163'733'292	.zcm	win64	no	1	3-Jul-2014	Italy	Fast CM + filter	7.22
GUI	EcoZip 16.02 (2016) Stephan Busch et al. ☺		###	3'135	2'114	77'302'000	411'356'510	454'493'800	279'408'950	576'215'426	471'125'939	301'976'409	33'252'588	143'282'521	.7ze	win64	no	8	5-Aug-2016	Germany	CSC 3.2, LIBBSC 3.1.0, LZHAM 1.1, LZMA/II, MCM 0.83, packMP3, PPMd + delta	5.13
•	ZCM 0.70d -m7 -r -s (2012) Francesco Nania		###	2'538		75'621'740	386'194'956	463'205'174	285'838'300	574'059'514	484'204'751	268'775'465	33'915'597	166'349'880	.zcm	win32	no	1	16-Oct-2012	Italy	Fast CM + filter	8.93
•	MCM 0.82 -9 -max (2015) Mathieu Chartier ☺	TAR	###	5'237		71'393'199	439'235'978	454'294'944	280'953'977	576'659'878	481'499'898	290'020'188	30'863'921	151'001'565	.mcm	win64	no	1	16-Feb-2015	Canada	Fast CM + LZP	6.60
•	RAZOR Archiver 1.01 -d 1023m Christian Martelock		###	12'842	298	73'326'392	347'971'806	487'002'929	293'796'218	571'877'430	484'616'521	266'714'145	32'742'019	166'929'016	.rz	win64	no	2	8-Sep-2017	Germany	Dedup + ROLZ + filter	4.04
GUI	Precomp 0.4.3 + FreeARC 0.67 ultra+exp B. Ziganshin ☺		###	5'261	772	68'304'203	351'107'079	489'483'982	266'960'540	578'428'557	463'570'395	303'459'585	34'744'226	184'970'713	.arc	win32	no	1	12-Dec-2012	Russia	Dedup. + LZMA + PPMd + GRZip + TTA	8.78
•	MCM 0.81 -9 (2015) Mathieu Chartier ☺	TAR	###	5'073		72'212'430	440'703'353	455'697'995	281'556'418	577'178'690	481'923'105	290'909'227	31'205'269	152'663'150	.mcm	win64	no	1	10-Feb-2015	Canada	Fast CM + LZP	6.62
GUI	NanoZip 0.09 -cO -m2g ('11) Sami Runsas		###	2'791	1'284	77'381'984	346'114'516	469'040'251	302'565'445	572'140'371	484'239'749	276'707'145	35'424'470	172'140'357	.nz	win64	no	4	4-Nov-2011	Finland	BWT + LZT + filter	9.88
GUI	PowerArchiver 17.01.04 .pa opt. strong extreme ConeXware, Inc.		###	2'292	1'183	73'023'304	414'536'042	474'678'493	274'663'420	556'347'267	438'416'279	301'657'338	35'519'216	176'591'802	.pa	win64	no	4	15-Oct-2017	United States	BWT, Lepton, LZMA/II, packMP3, PLZMA4, PPMd_sh, Reflate, ZSTD + filter	3.94
GUI	PowerArchiver 17.00.92 .pa opt. strong extreme ConeXware, Inc.		###	2'540		72'746'947	414'536'042	472'823'130	271'337'397	558'244'601	438'117'471	301'657'338	35'369'853	176'591'803	.pa	win64	no	4	25-Jul-2017	United States	BWT, Lepton, LZMA/II, packMP3, PLZMA4, PPMd_sh, Reflate, ZSTD + filter	4.16
GUI	WinRK 3.1.2 (2006) Malcolm Taylor		###	7'416	2'596	76'614'559	343'812'879	485'395'149	286'131'614	573'646'514	481'541'757	266'877'734	34'882'268	181'306'077	.rk	win64	no	4	1-Mar-2006	New Zealand	ROLZ-3 + PPM	15.56
•	CCMx 1.30c opt. 7 Christian Martelock	TAR	###	3'043	3'406	78'398'914	391'606'412	466'979'717	289'869'520	577'169'132	484'047'738	296'599'146	32'892'892	167'908'122	.ccmx	win32	no	1	24-Apr-2008	Germany	Fast CM + filter	13.41
•	MCM 0.8 -9 (2015) Mathieu Chartier ☺	TAR	###	5'289		74'963'812	459'549'468	456'703'885	281'086'733	577'411'076	482'079'510	293'388'393	31'488'791	154'292'837	.mcm	win64	no	1	5-Feb-2015	Canada	Fast CM + LZP	6.63
•	ZPAQ 6.56c -method 57 Matt Mahoney ☺		###	8'766		73'831'697	424'982'356	446'512'625	284'259'827	576'699'643	480'748'568	286'358'642	31'900'064	161'578'265	.zpaq	win64	no	8	19-Nov-2014	United States	Dedup. + Fast CM + filter	6.84
GUI	EcoZip 16.02 (2016) Stephan Busch et al. ☺		###	1'572	1'073	78'881'496	411'430'983	458'949'277	292'369'020	577'040'723	471'642'213	301'976'409	35'369'789	167'869'219	.7ze	win64	no	8	9-Jul-2016	Germany	BZIP2, CSC 3.2, Deflate64, LIBBSC 3.1.0, LZMA/II, packMP3, PPMd + delta	5.20
•	ZPAQ 6.33 -method 77 ('13) Matt Mahoney ☺		###	11'750		71'858'850	425'204'026	447'512'299	283'252'643	576'622'979	480'937'821	286'204'166	31'381'272	157'683'624	.zpaq	win64	no	8	21-Jun-2013	United States	Dedup. + Fast CM + filter	8.25
GUI	WinRK 3.1.2 (ROLZ2) (2006) Malcolm Taylor		###	7'068	3'065	79'404'688	345'763'924	497'270'881	286'859'957	586'211'755	484'970'039	266'988'020	35'221'720	182'165'440	.rk	win64	no	4	1-Mar-2006	New Zealand	ROLZ-3 + PPM	15.56
•	MCM 0.3 -9 (2013) Mathieu Chartier	TAR	###	4'783		82'529'921	450'157'372	458'055'272	283'140'276	577'002'867	482'161'433	294'329'466	32'210'270	156'605'126	.mcm	win32	no	1	17-Jun-2013	Canada	Fast CM	8.26
GUI	NanoZip 0.09 -co -m2g (2011) Sami Runsas		###	476		81'589'626	347'884'435	491'875'984	299'055'780	579'100'338	485'878'580	277'080'042	38'903'203	185'674'185	.nz	win64	no	4	4-Nov-2011	Finland	BWT + LZT + filter	9.88
GUI	FreeARC 0.67 (2012) -mx -ld1600m Bulat Ziganshin ☺		###	2'091		80'135'168	351'107'079	501'293'683	302'192'145	578'708'723	490'303'204	303'462'080	34'886'587	184'182'322	.arc	win32	no	2	12-Dec-2012	Uzbekistan	Dedup./REP + DICT + LZP + LZMA, PPMd, GRZip + MM/TTA + filter	8.78
GUI	FreeARC'Next 0.11 -mx (2016) Bulat Ziganshin		###	1'505		78'239'941	351'107'743	501'327'715	302'570'735	578'745'488	490'302'518	303'379'251	35'705'949	187'382'529	.arc	win64	no	2	8-Oct-2016	Uzbekistan	Dedup./REP + DICT + LZP + LZMA, PPMd, GRZip + MM/TTA, ZSTD + filter	4.96
•	ZPAQ 7.05 -method 53 ('15) Matt Mahoney ☺		###	7'378		77'195'169	426'521'556	448'763'836	289'825'962	581'593'331	481'480'893	287'923'024	32'964'064	177'034'166	.zpaq	win64	no	8	16-Dec-2013	United States	Dedup. + Fast CM + filter	7.77
•	KANZI 1.3 -l 5 -b 1G -t none (2018) Frederic Langlet et al. ☺		###	6'446		76'379'017	452'607'709	459'081'733	284'117'733	579'331'569	483'071'113	295'452'725	32'427'433	171'161'753	.knz	win64	no	1	15-Jan-2018	United States	TPAQ (based on Tangelo) (Predictive Arithmetic Coding)	3.69
•	LPAQ9m -9 (2009) Matt Mahoney et al. ☺	TAR	###	11'152	11'947	82'315'624	483'211'343	466'721'795	283'643'675	574'158'259	483'441'042	298'879'799	31'224'332	149'353'612	.lpaq9m	win32	no	1	20-Feb-2009	USA & Russia	Fast CM	12.59
•	Packet 1.9 -mx -s -b5 -h9 Francesco Nania		###	4'444	0	84'121'705	360'488'298	484'818'589	312'475'409	574'613'752	486'077'898	281'058'602	37'071'470	194'869'716	.packet	win64	no	1	19-Aug-2016	Italy	LZ77 + Huffman + filter	5.09
•	LPAQ7e -9 (2007) Matt Mahoney et al. ☺	TAR	###	11'071	11'481	82'483'738	504'932'570	464'634'575	278'351'488	574'136'654	481'952'693	298'708'197	31'143'250	150'172'517	.lpaq7e	win32	no	1	31-Oct-2007	USA & Russia	Fast CM + E8E9	13.89
•	PAQ9a -9 (2007) Matt Mahoney et al. ☺	TAR	###	15'702	16'591	79'385'282	454'055'569	457'915'837	280'356'591	574'254'311	481'186'044	294'809'748	31'283'840	159'137'846	.paq9a	win32	no	1	31-Dec-2007	United States	LZP + Fast CM	13.73
•	CMM4 0.2b 96 (2008) Christopher Mattern ☺	TAR	###	5'534	4'843	84'362'230	510'874'018	468'665'303	286'575'129	578'462'992	483'475'436	303'327'118	32'582'554	164'930'949	.cmm4	win32	no	1	4-Jun-2008	Germany	CM	13.30
•	LPAQ4e -9 (2007) Matt Mahoney et al. ☺	TAR	###	10'865	11'715	83'789'117	507'404'490	461'784'004	278'771'170	574'253'251	480'992'119	296'982'199	31'636'700	153'118'456	.lpaq4	win32	no	1	1-Oct-2007	USA & Russia	Fast CM + filter	13.98
GUI	Precomp 0.4.3 + 7-ZIP 9.3.0 ultra 128MB	TAR	###	6'491		78'762'255	441'043'625	487'151'173	267'535'706	576'683'197	460'935'592	337'122'045	35'800'534	180'044'575	.7z	win32	no	1	2-Jan-2009	Russia	LZMA2lc4:128; PPMd:1024 o10 + delta:3	12.72
GUI	7-ZIP 9.3.8 ultra128:fb273:bt4:lc4 ☺		###	2'830	1'639	79'500'816	441'035'970	493'043'375	304'617'390	577'098'703	487'440'321	322'709'493	36'126'394	179'406'622	.7z	win64	no	3	3-Jan-2015	Russia	LZMA2:128, PPMd:1024 o14, delta:3/delta:4	6.72
GUI	7-ZIP 17.02 ultra128:fb273:bt4:lc3 yx=9 ☺		###	3'019		79'379'134	429'887'231	493'043'375	304'617'390	577'095'176	487'465'775	325'808'955	36'040'119	179'406'622	.7z	win64	no	3	28-Aug-2017	Russia	LZMA2:128, PPMd:1024 o14, delta:3/delta:4	4.07
•	Tangelo 2.3 rev 0116 J. Ondrus & F. Langlet ☺	TAR	###	8'456		84'179'726	473'432'412	462'450'295	291'544'773	579'019'843	483'857'831	295'031'539	32'862'855	174'635'480	.tan	win32	no	1	12-Jan-2016	zech Republic & US	Fast CM	5.69
•	RINGS 2.5 -s -r -m8 (2015) Francesco Nania		###	1'746		94'651'975	400'606'351	465'919'810	306'984'877	578'879'059	483'774'152	289'985'205	42'923'147	165'537'369	.rings	win64	no	1	6-Jun-2015	Italy	BWT + ARI	6.29
•	BIT 0.7 LWCX -P=5 Osman Turan	TAR	###	6'284		85'276'158	510'306'657	468'307'527	287'061'074	578'113'802	483'390'665	303'742'288	32'915'199	168'718'324	.bit	win32	no	1	4-Jan-2009	Turkiye	CM	12.72
GUI	UHARC 0.6b -mx/-m3 -mm+ Uwe Herklotz		###	8'159		80'232'277	395'293'930	477'253'880	301'703'146	576'204'183	485'825'646	292'075'855	35'198'987	203'974'516	.uha	win32	no	1	6-Feb-2005	Germany	PPM 50 MB + LZ + ARI 291 MB	16.63
•	Packet 1.2 -h5 -b512 -mx Francesco Nania		###	3'295		84'169'777	408'497'885	480'645'014	307'814'708	575'919'246	485'575'078	283'982'876	38'063'205	204'700'171	.packet	win64	no	1	20-Jul-2015	Italy	LZ77 + Fast CM	6.17
•	SBC 0.970r3 -m3 -b63 -os Sami Makinen		###	2'250		97'303'748	352'299'035	473'450'475	304'275'096	582'698'288	485'944'367	304'403'688	38'774'613	194'610'863	.sbc	win64	no	1	11-Mar-2008	Finland	BWT + ARI	13.53
•	LPAQ1 -9 (2007) Matt Mahoney et al. ☺	TAR	###	11'332	12'095	84'488'523	516'330'193	467'019'940	281'496'506	574'106'066	481'430'206	304'463'339	31'930'129	157'883'859	.lpaq1	win32	no	1	24-Jul-2007	United States	Fast CM	14.16
•	Tangelo 2.0 (2013) Jan Ondrus ☺	TAR	###	19'933		78'906'065	451'394'450	458'817'126	288'474'516	576'488'437	481'671'721	291'703'582	31'314'571	164'118'727	.tan	win32	no	1	6-Jul-2013	Czech Republic	Fast CM	8.21
•	Tangelo 2.4 (2013) Jan Ondrus ☺	TAR	###	3'947		90'970'960	522'955'272	475'340'089	272'133'582	577'858'048	459'121'106	318'791'327	33'876'178	174'391'732	.tan	win32	no	1	17-Aug-2013	Czech Republic	Fast CM	8.10
•	Pcompress 3.2 -l14 -s60m Moinak Ghosh ☺		###	971		81'284'737	541'499'208	500'927'702	265'583'853	581'033'256	471'724'887	316'530'910	36'971'719	181'000'873	.pz	linux64	no	8	23-Sep-2015	India	Dedup. + BZIP2 / LZMA / PPMd / packJPG + filter	6.00
•	Tangelo 2.3 (2013) Jan Ondrus ☺	TAR	###	3'665		89'309'602	516'077'838	474'196'801	290'752'195	577'801'308	484'812'825	312'002'309	33'346'539	172'846'758	.tan	win32	no	1	22-Jul-2013	Czech Republic	Fast CM	8.17
GUI	UHARC 0.4 -mx/-m3 -mm+ Uwe Herklotz		###	5'590		83'847'245	397'613'629	478'646'282	304'729'847	582'549'377	485'987'428	293'432'096	36'934'509	212'232'106	.uha	win32	no	1	28-Dec-2001	Germany	LZP / ALZ / PPM	19.73
GUI	FLASHZIP 1.1.3 -r -s -mx3 -k7 -b1024 (2013)		###	1'667		88'961'837	384'619'601	496'852'845	319'261'810	578'923'326	486'803'989	288'721'818	38'740'607	210'064'678	.flz	win32	no	1	5-Aug-2013	Italy	ROLZ + ARI	8.13
GUI	FLASHZIP 1.1.0 -r -s -mx3 -k7 -b1024 (2012)		###	1'697		88'961'837	384'633'518	497'660'006	319'268'847	578'923'326	486'803'989	288'721'818	38'740'447	210'064'678	.flz	win32	no	1	3-Dec-2012	Italy	ROLZ + ARI	8.80
•	Masked Compression 2.5.1 ultral3 (2016)		###	12'254		79'157'936	336'453'399	498'911'675	308'950'030	568'159'664	466'309'927	362'708'737	35'123'891	190'596'205	.bin	win32	no	2	13-Aug-2016	South Africa & India	FreeARC 0.67 + BMF 2.01 / XZ / OFR 4.910b / Precomp 0.4.6 / Reflate / ZSTD 0.8	5.11
GUI	7-ZIP 9.3.8 LZMA2 + delta Igor Pavlov ☺		###	3'217		79'500'816	441'035'970	498'111'402	318'117'697	577'098'703	487'440'321	322'709'493	36'126'394	197'414'870	.7z	win64	no	3	3-Jan-2015	Russia	LZMA2:128, BCJ2, delta:3/delta:4	6.72
•	MSC 0.0.6.4 (CLS) (2013) ProFrager		###	5'066		89'425'541	349'671'145	490'618'385	317'088'663	579'947'707	474'364'661	358'748'909	35'959'247	193'760'673	.msc	win32	no	2	26-Aug-2013	Russia	FreeARC 0.67 + BMF 2.01 + LZMA + TAK 2.3.0	8.07
•	RZM 0.07e (2008) Christian Martelock	TAR	###	5'095		80'746'777	470'056'236	478'724'569	307'929'578	576'714'890	484'783'957	320'968'722	34'837'417	193'711'323	.rzm	win32	no	1	16-Mar-2008	Germany	64 MB sliding window ROLZ	13.52
•	Packet 1.1 -h4 -b512 -mx Francesco Nania		###	3'046		85'566'387	400'864'727	481'667'787	313'282'760	574'046'330	485'433'113	297'838'031	38'728'339	215'148'334	.packet	win32	no	1	7-Dec-2013	Italy	LZ77 + Fast CM	7.79
•	Masked Compression 2.5.1 ultral2 (2016)		###	12'503		79'509'013	336'453'384	498'916'706	308'350'577	574'773'201	466'647'197	362'721'070	35'703'675	190'595'031	.bin	win32	no	2	13-Aug-2016	South Africa & India	FreeARC 0.67 + BMF 2.01 / XZ / OFR 4.910b / Precomp 0.4.6 / Reflate / ZSTD 0.8	5.11
•	Pcompress 3.0 -l14 -s60m Moinak Ghosh		###	854		85'505'721	542'717'830	493'511'557	267'948'151	579'602'074	473'825'655	333'182'817	37'383'155	178'865'586	.pz	linux	no	8	15-Jan-2014	India	Dedup. + BZIP2 / LZMA / PPMd / packJPG + filter	7.69
GUI	7-ZIP 17.02 ultra128:fb273:bt4:lc4:yx9 ☺		###	2'864		79'294'915	469'539'846	493'043'410	304'617'572	577'104'610	487'440'328	363'105'844	36'138'527	179'406'676	.7z	win64	no	3	28-Aug-2017	Russia	LZMA2:128, PPMd:1024 o14, delta:3/delta:4	4.07
•	Masked Compression 2.5.1 ultral1 (2016)		###	8'032		87'231'425	336'453'399	499'042'151	311'193'882	574'410'464	471'846'411	362'790'156	35'587'415	196'192'263	.bin	win32	no	2	13-Aug-2016	South Africa & India	FreeARC 0.67 + BMF 2.01 / LZMA / OFR 4.910b / Precomp 0.4.6 / Reflate / ZSTD 0.	5.11
•	BSC 3.1.0 -b1024r -m0 -e2 Ilya Grebnov ☺	TAR	###	1'069	650	106'062'384	413'460'698	504'212'594	294'567'520	585'458'040	490'817'414	324'642'920	38'190'522	167'869'306	.bsc	win64	CUDA	8	8-Jul-2012	Russia	BWT + libdifsufsort + QLFC + CM + filter	9.21
GUI	STUFFIT 14.0.1.27 (2010) bit-perfect SmithMicro Corp.		###	2'679		95'432'555	403'289'660	484'760'105	271'777'543	585'557'564	452'056'163	344'467'525	39'655'270	200'157'791	.sitx	win64	no	8	8-Feb-2010	United States	LZARI 5:28 / PPM 16:30	11.62
•	BSC 3.1.0 -b1024rsca -m0 -e2 Ilya Grebnov ☺	TAR	###	1'319		105'153'385	413'450'026	504'454'068	294'567'520	585'458'040	490'817'414	324'577'856	38'261'586	172'050'326	.bsc	win64	CUDA	8	8-Jul-2012	Russia	BWT + libdifsufsort + QLFC + CM + filter	9.21
•	CsArc 3.3 -m5 -d512 -t4 -r Siyuan Fu ☺		###	1'743		85'356'727	413'445'335	485'263'850	315'378'301	578'137'544	486'880'107	332'166'309	39'083'025	201'410'740	.csa	win64	no	4	21-Mar-2015	United States	LZ77 + ARI + filters	6.50
GUI	Squeez 5.63 / SpeedCmd (32MB ultra) auto		###	7'325		85'752'271	361'509'947	489'920'850	298'886'419	577'911'802	483'275'656	355'828'299	40'097'449	185'335'396	.sqx	win64	no	1	11-May-2005	Germany	LZR / PPM (o8, 768 MB) + filter	16.36
GUI	WinRar 4.20 solid best force PPMd Eugene Roshal		###	3'285	2'183	92'959'368	408'872'338	487'351'765	305'582'289	592'737'914	490'590'092	319'664'733	38'238'952	192'946'122	.rar	win64	no	1	9-Jun-2012	Russia	PPM-II o9-128M + filter	9.29
•	Blizzard 0.24 300m ('08) Christian Martelock	TAR	###			98'816'831	502'115'581	492'083'648	293'705'230	582'751'906	485'502'585	306'869'116	38'309'760	171'545'984	.bliz	win32	no	1	30-Jun-2008	Germany	BWT + filter	13.23
GUI	CTXf 0.75 pre-b1 -me -b Nikita Lesnikov		###			101'856'759	407'913'955	491'460'126	308'122'202	590'693'341	489'051'945	311'183'180	37'513'145	198'221'614	.cxf	win32	no	1	27-Sep-2003	Belarus	PPM-II + filter	17.99
•	BSC 3.1.0 -b1024rsca -m0 -H28 -M255 Ilya Grebnov ☺	TAR	###	1'288		108'172'356	412'185'160	507'255'944	300'897'766	581'534'528	484'920'043	324'387'750	38'202'132	172'066'806	.bsc	win64	CUDA	8	8-Jul-2012	Russia	LZP + BWT + libdifsufsort + QLFC + CM + filter	9.21
•	Blizzard 0.24 30m ('08) Christian Martelock	TAR	###			97'991'972	526'871'036	467'663'215	297'266'236	583'452'991	483'739'655	300'975'553	38'031'300	186'228'420	.bliz	win32	no	1	30-Jun-2008	Germany	BWT + filter	13.23
•	REFLATE v1l (2016) Eugene Shelwien	TAR	###	7'290		76'843'311	534'770'341	486'053'545	306'603'503	###	474'174'492	351'459'454	34'465'685	197'061'497	.ref	win32	no	1	21-Sep-2016	Ukraine	ZLIB recompressor + PLZMA	5.00
•	Pcompress 3.1 -l14 -s60m Moinak Ghosh ☺		###	973		82'423'242	541'971'178	500'900'409	265'573'919	581'048'799	471'895'396	365'395'869	37'022'849	189'550'950	.pz	linux64	no	8	5-Feb-2015	India	Dedup. + BZIP2 / LZMA / PPMd / packJPG + filter	6.63

Summary (brief results)

Program	Ver	Arguments	St.	Size	%	C.Time	D.Time	Rating	c/d
nanozip	0.09a (64)	-m.5g -cc	*****	623 643 550	18.9	**4904.45**	**4792.47**	290	528/46
nanozip	0.09a (64)	-m.5g -cc -p4	*****	657 437 806	19.9	**1353.09**	**1327.78**	719	1314/113
nanozip	0.09a (64)	-m.5g -cO	½****	657 919 225	19.9	1813.44	**436.77**	852	975/342
nanozip	0.09a (64)	-m.5g -cO -p2	*****	676 508 285	20.5	**1045.05**	**270.89**	**1185**	1376/449
zcm	0.50ai	-m5	****	692 818 251	21.0	1603.69	1570.61	410	748/65
freearc	0.666	-mx -lc500m	½****	706 908 234	21.4	1178.65	**224.89**	793	870/386
freearc	0.666	-m9d -lc500m	****	719 121 524	21.8	1210.20	**119.14**	731	740/**635**
lpaq	8	7	½**	728 206 074	22.1	5069.85	5113.64	86	160/13
ccm	1.30c	x 5	½**	729 932 323	22.1	1650.23	1587.67	266	481/42
ppmonstr	Jr1	-m600 -o16	½*	734 521 831	22.3	9663.08	9750.66	42	78/7
ccm	1.30c	5	***	735 303 098	22.3	1332.17	1270.86	312	562/50
nanozip	0.09a (64)	-m.5g -co	*****	735 353 979	22.3	**456.03**	**90.87**	**1483**	1640/**695**
uharc	0.6b	-mdp -mx	½*	747 273 011	22.6	3182.80	2664.78	121	206/21
mcomp	2.00	-M256M -mf3	**	760 902 054	23.1	6828.70	398.77	84	82/119
freearc	0.666	-m4	½****	761 362 930	23.1	**324.67**	94.50	**1449**	1725/**501**
cmm4	0.1e	75	½	766 629 366	23.2	3382.41	3436.73	84	156/13
rzm	0.07h		**	771 921 245	23.4	3173.41	217.20	159	157/194
uharc	0.6b	-mdp -m3	*	774 962 651	23.5	3519.69	367.72	134	137/111
7-zip	9.12b (64)	-mx -md48m	½***	779 659 779	23.6	1148.17	103.81	396	398/372
uharc	0.6b	-mdp -m2	½*	780 467 656	23.7	3051.99	371.22	144	148/103
uharc	0.6b	-mdp -m1	*	791 477 304	24.0	2437.19	374.00	155	164/91
csc	3.2f	-m3 -d320	***	801 503 240	24.3	619.56	114.58	530	579/264
7-zip	9.12b (64)		½**	805 116 998	24.4	891.84	103.83	375	386/280
7-zip	9.12b (64)	-m0=lzma2 -mx -md15m	½**	807 289 820	24.5	637.72	104.56	491	527/272
flashzip	0.99d1	-m9 -b5	*	820 705 242	24.9	1673.48	156.09	172	173/157
freearc	0.666	-m3	½****	825 660 724	25.0	**130.56**	92.03	**1336**	**2099**/252
nanozip	0.09a (64)	-m.5g -cDP	½*****	830 163 327	25.2	236.17	**33.68**	1048	1104/**654**
csc	3.2f	-m2 -d320	**	840 353 904	25.5	451.81	110.08	449	515/179
yzx	0.11	-m9 -b5 -h4	*	843 113 485	25.5	1275.97	156.58	171	177/122
nanozip	0.09a (64)	-m.5g -cDp	****	843 802 174	25.6	223.11	33.81	946	1004/**560**
7-zip	9.12b (64)	-mx -md4m	½*	845 209 016	25.6	838.47	108.41	253	263/172
yzx	0.11	-m7 -b5 -h4	½	845 259 618	25.6	972.50	156.78	212	227/119
csc	3.2f	-m1 -d320	½*	846 399 651	25.6	320.52	112.17	546	679/164
xwrt	3.2	-l6 -b32 -m32		857 480 793	26.0	2305.62	128.76	86	83/126
4x4	0.2a	lzma:32mht4:mc16	½***	857 568 612	26.0	427.50	**29.91**	456	450/**543**
xwrt	3.2	-l6 +d -l65535 -m32		857 988 198	26.0	2303.92	129.11	85	83/125
sbc	0.970r3 (64)	-m3 -b63		858 047 074	26.0	1186.89	487.84	124	161/33
yzx	0.11	-m8 -b5 -h4		859 631 823	26.0	1110.55	158.47	161	169/100
yzx	0.11	-m6 -b5 -h4	½	861 731 092	26.1	856.88	158.51	196	214/98
sbc	0.970r3 (64)	-m3 -b9 -of		867 122 163	26.3	968.28	475.26	130	179/31
nanozip	0.09a (64)	-m.5g -cD	*****	882 175 527	26.7	**95.80**	**25.37**	**1309**	**1527/487**
nanozip	0.09a (64)	-m.5g -cdP	****	884 252 106	26.8	146.81	27.92	887	974/433
7-zip	9.12b (64)	-mx1 -md48m	*	893 029 360	27.1	681.97	104.86	179	190/104
winace	2.6	-d4096 -m5		897 631 771	27.2	1447.73	249.72	79	85/42
winace	2.6	-d4096 -m4		898 403 943	27.2	1296.70	249.62	86	94/41
lzturbo	0.95	-p0 -59	**	899 105 522	27.2	4370.11	26.18	30	28/391
winrar	4.20b3 (64)	-m5 -mc14:128t	½	902 235 561	27.3	492.16	313.37	158	238/32
nanozip	0.09a (64)	-m.5g -cdp	½***	905 169 033	27.4	125.92	28.93	793	900/331
winace	2.6	-d4096 -m2		906 749 204	27.5	967.53	250.66	99	115/38
flashzip	0.99d1	-m4 -b5		908 452 921	27.5	543.88	159.28	168	201/58
winace	2.6	-d4096 -m1		909 079 588	27.5	905.14	249.89	102	120/37
blizzard	0.24b	c 100000000		912 765 871	27.7	804.64	704.69	75	129/12
m1	x2-0.6	6 enwik7.txt		919 992 547	27.9	1869.35	1918.86	28	51/4
winrar	4.20b3 (64)	-m5	½**	920 048 131	27.9	314.27	53.09	283	306/153
winrar	4.20b3 (64)	-m4	**	920 933 096	27.9	282.78	53.20	307	336/151
bsc	2.4.5 (64)	-b110t -m3 -psrl	*	921 301 595	27.9	414.94	173.81	174	228/46
winrar	4.20b3 (64)	-m3	½*	923 642 139	28.0	235.41	53.20	347	392/147
csc	3.2f	-m2 -d4	½	925 219 556	28.0	475.58	119.06	165	191/64
freearc	0.666	-m2	½**	927 615 799	28.1	**73.91**	75.70	640	1194/99
squeez	5.62	-m5		930 059 173	28.2	1801.87	305.76	44	48/24
winace	2.6			930 558 933	28.2	951.75	251.66	77	90/29
sbc	0.970r3 (64)	-m2 -b19		934 502 337	28.3	674.20	379.19	84	121/18
nanozip	0.09a (64)	-m.5g -cd -p1	***	934 736 671	28.3	117.00	34.00	586	697/203
ppmd	sh8	/m590 /o16 /r		936 667 966	28.4	3111.94	3175.34	14	26/2
sbc	0.970r3 (64)	-m1 -b63 -of -ad		936 774 698	28.4	866.16	545.69	61	92/12
sbc	0.970r3 (64)			940 673 600	28.5	615.16	365.50	84	124/18
ppmd	sh8	/m590 /o16		942 394 965	28.6	1263.86	1314.33	32	59/5

World Compression Challenge

file compression benchmark

It is a reference for the latest information about the performance of lossless data compression software. The test takes place on many file types so as not to benefit a specific compressor and to verify the actual ability to adapt to any type of data. OS: Windows 10 , CPU: AMD Ryzen™ 3700x (8 core– 16 thread). Test By Nania Francesco A. (Italy)

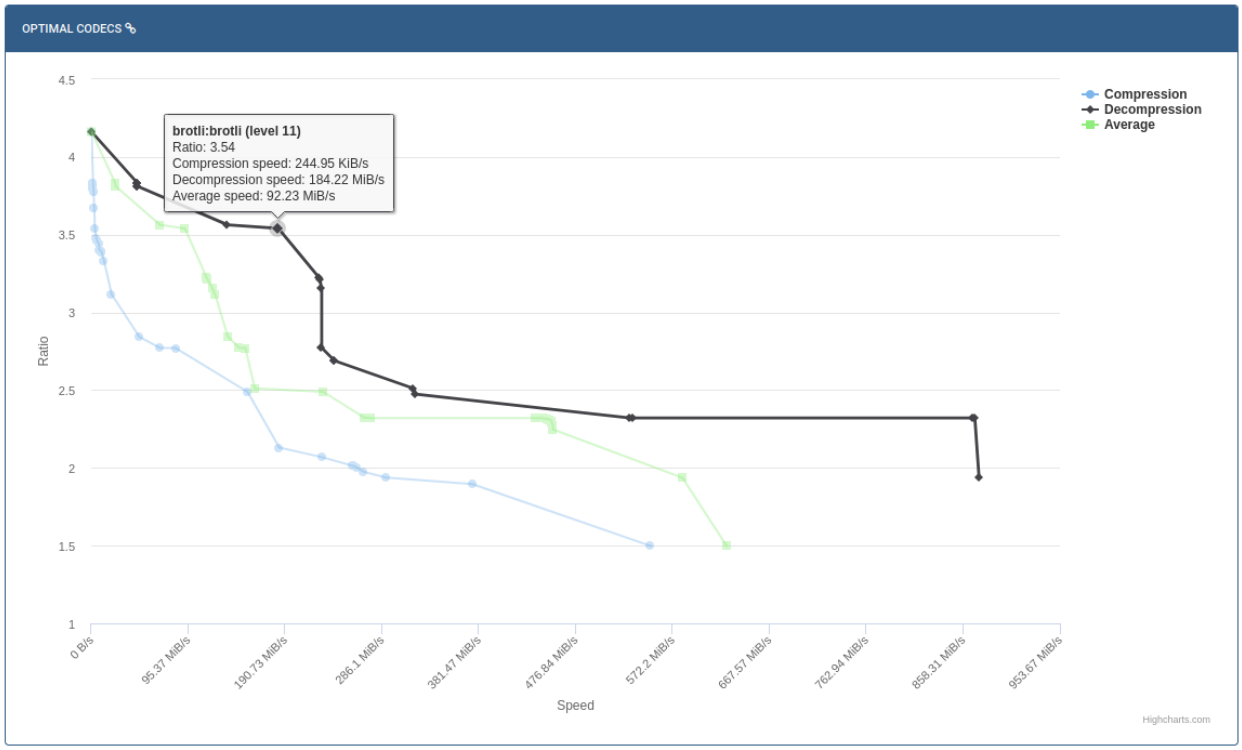
APP1 APP2 GAME1 GAME2 BMP ISO PDF RAW BIN TXT MKV WAV MP3 JPG

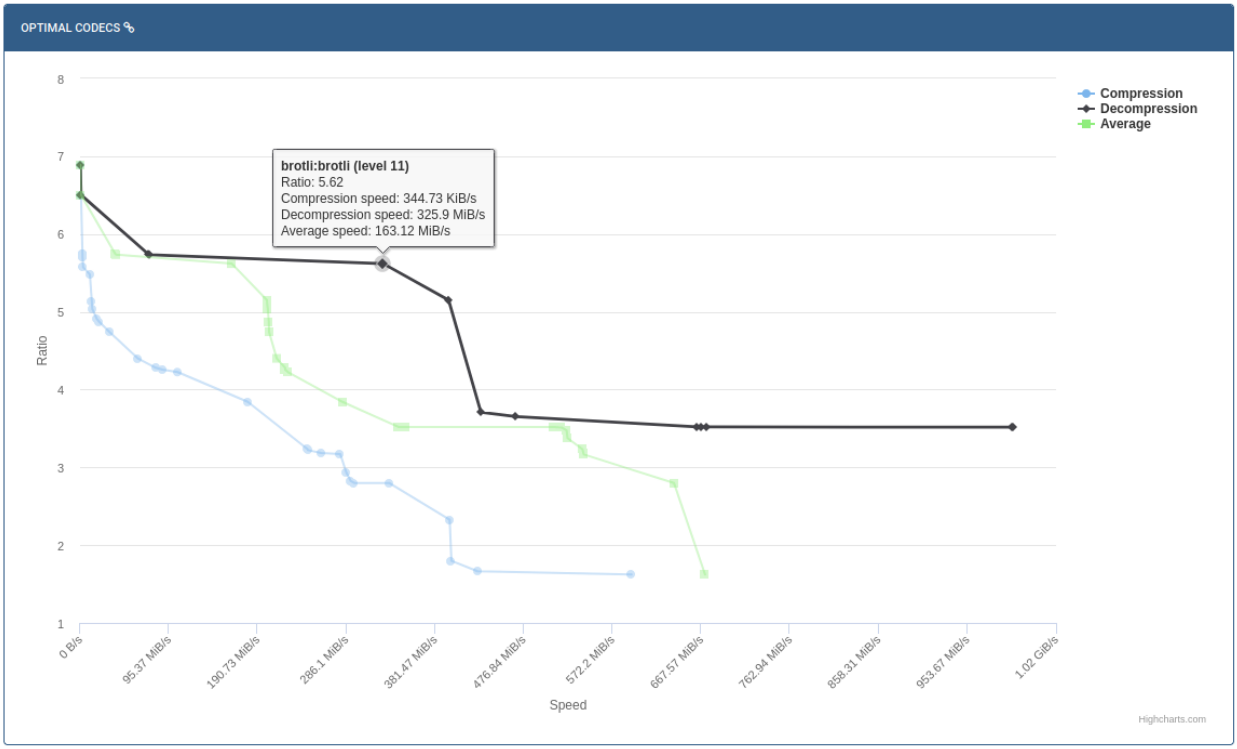
SUMMARY TEST

SUMMARY TEST

Rank	Archiver / Compressor	SIZE	TIME C.	TIME D.	C.EFF.	D.EFF.	C/D EFF.	OPTIONS
1	Nanozip v.0.09a Sami Runsas ,FIN [ARC]	571470427	158.02	161.33	27915	28021	27968	a -cc -r -p8 -m1g
2	ZCM v.0.93- Nania Francesco A.,ITA [ARC]	573057223	290.50	261.63	32218	31294	31756	a -r -s -m6
3	Razor v.1.03.7[Christian Martelock,GER]	582717404	1198.00	25.63	61645	24129	42887	a -r -d 512M
4	Nanozip v.0.09a Sami Runsas ,FIN [ARC]	584894598	116.89	58.47	27136	25267	26201	a -cO -r -p8 -m1g
5	Packet v.1.9 Nania Francesco A.,ITA [ARC	587113416	384.73	20.99	35796	24156	29976	a -r -s -mx -b5 -h6
6	ZCM v.0.93- Nania Francesco A.,ITA [ARC]	589484836	122.72	102.03	27506	26844	27175	a -r -s -m5 -t4
7	Nanozip v.0.09a Sami Runsas ,FIN [ARC]	591086553	79.95	29.58	26202	24590	25396	a -co -r -p8 -m1g
8	ZCM v.0.93- Nania Francesco A.,ITA [ARC]	593176250	80.31	75.50	26297	26143	26220	a -r -s -m2 -t8
9	Packet v.1.9 Nania Francesco A.,ITA [ARC	594521726	186.28	21.81	29742	24479	27110	a -r -s -m0 -b5 -h6
10	Precomp v.0.47 (Christian Schneider|GER)	602561640	157.69	84.66	29149	26812	27980	-cl
11	ARC v.0.67a Bulat Ziganshin,UZB [ARC]	604443942	216.55	34.76	31107	25290	28198	a -r -m9x -lc2000 -mt8
12	7ZIP v.19.00 -Igor Pavlov,RUS-[ARC]	613919737	168.78	13.06	29958	24975	27466	a -mx9 -mmt=8 -md=256m -mfb=273
13	ARC v.0.67a Bulat Ziganshin,UZB [ARC]	619409644	198.47	52.03	31127	26441	28784	a -r -m9 -sGROUPING -mt8
14	CSARC v.3.3 - Siyuan Fu, CHI -[ARC]	620500220	193.44	21.42	31010	25505	28257	a -m5 -d256m -r -t8
15	DGC v.1.10 Shin-ichi TSURUTA,JAP [ARC]	622344064	220.48	125.86	31949	28921	30435	a
16	ARC v.0.67a Bulat Ziganshin,UZB [ARC]	623349348	204.64	37.95	31482	26148	28815	a -r -m9x -sGROUPING -mt8
17	ARC v.0.67a Bulat Ziganshin,UZB [ARC]	623349348	208.06	38.66	31592	26171	28881	a -r -m8x -sGROUPING -mt8

Uharc is windows, but can be run with wine https://bbs.archlinux.org/viewtopic.php?id=26320
https://www.7-zip.org/download.html has console versions for linux, windows, mac
http://www.mattmahoney.net/dc/zpaq.html windows executables and make files for linux and mac
https://github.com/moinakg/pcompress
http://moinakg.github.io/pcompress/
https://encode.su/threads/2829-RAZOR-strong-LZ-based-archiver
There is no x86 or linux executable.
You need to run it with wine on linux. Wine rz.exe
PACKET
http://heartofcomp.altervista.org/Packet.htm
LZA
http://heartofcomp.altervista.org/LZA.htm
pcompress
https://code.google.com/archive/p/pcompress/





Compare different datasets and keep the machine the same. Brotli 11 in these benchmarks always seems to be very fast while compression ratio wise being on a similar level with lzma and others. Of course zpaq aways higher but slow decompression speeds, we dont want

downloaders to be frustrated waiting way longer for the decompression than the download itself, otherwise they will not use it.
https://github.com/google/brotli
Lzturbo 39 seems to be very fast decompressing. Like twice the speed of brotli. Also faster compression, making it more enjoyable for users. Its for windows and linux.
Zstd seems also to be in a similar league, faster than brotli, worse than lzturbo

Brotli vs Zstandard benchmark results tables

Out-of-the-box compression and extraction times in seconds, size of resulting archive in MB, the lower the better for all columns.

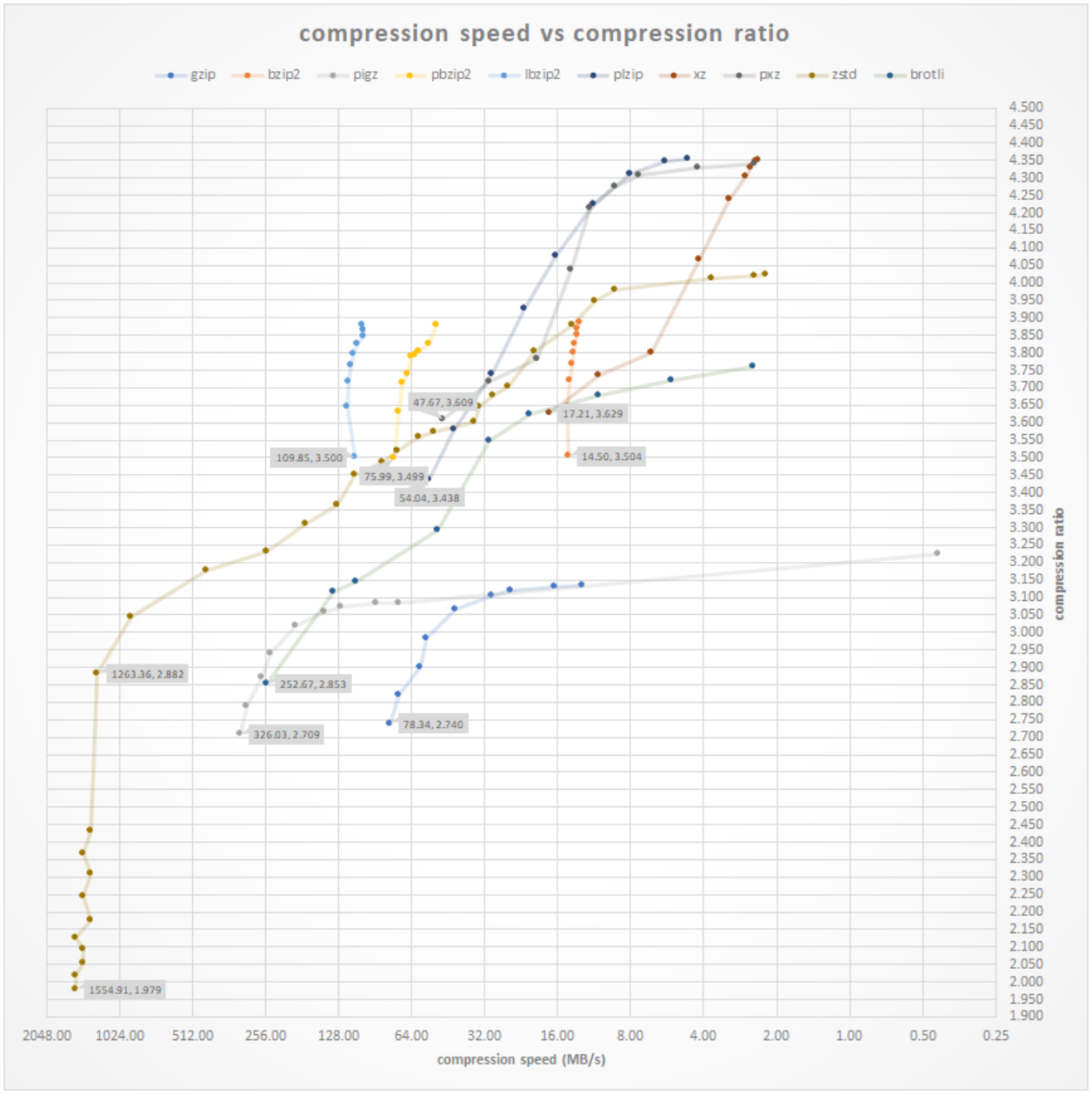
	Brotli			Zstandard			
Compression level	**Compression time (sec)**	**Extraction time (sec)**	**Archive size (MB)**	**Compression -T0 (sec)**	**Compression single thread -T1 (sec)**	**Extraction time (sec)**	**Archive size (MB)**
Minimum Brotli -0 / Zstd -1	5.4	5.0	473	1.9	4.6	2.6	451
Default Brotli -3 / Zstd -3	15.0	4.5	408	3.6	7.5	3.1	398
High Brotli -5 / Zstd -9	46.9	4.6	353	38.4	60.0	3.2	355
Very high Brotli -7 / Zstd -15	175.0	4.7	336	157.0	461.0	3.3	330
Maximum Brotli -9 / Zstd -19	677.0	5.7	322	359.0	1030.0	4.0	302

Fixed 128 MB window size for both algorithms:

	Brotli --large_window=27			Zstandard -T1 --long=27		
Compression level	**Compression time (sec)**	**Extraction time (sec)**	**Archive size (MB)**	**Compression -T0 (sec)**	**Extraction time (sec)**	**Archive size (MB)**
Minimum Brotli -0 / Zstd -1	5.7	5.0	473	8.7	2.8	407
Default Brotli -3 / Zstd -3	16.8	4.1	373	12.3	3.1	362
High Brotli -5 / Zstd -9	52.3	4.2	322	63.0	3.2	321
Very high Brotli -7 / Zstd -15	162.0	4.4	307	461.0	3.7	295
Maximum Brotli -9 / Zstd -19	721.0	5.4	293	1037.0	3.5	273

For reference, results of traditional compressors group, at default compression settings:

Compression format	**Compression time (sec)**	**Extraction time (sec)**	**Archive size (MB)**
ZIP WinRar, default	15.6	6.0	407
RAR WinRar, default	100.0	4.0	318
7Z PeaZip, default	241	3.2	293


compression speed vs compression ratio
gzip
bzip2
pigz
pbzip2
lbzip2
plzip
xz
pxz
zstd
brotli
47.67, 3.609
17.21, 3.629
109.85, 3.500
14.50, 3.504
75.99, 3.499
54.04, 3.438
1263.36, 2.882
252.67, 2.853
326.03, 2.709
78.34, 2.740
1554.91, 1.979
compression ratio
compression speed (MB/s)
2048.00
1024.00
512.00
256.00
128.00
64.00
32.00
16.00
8.00
4.00
2.00
1.00
0.50
0.25

193 lines (192 sloc) | 14.6 KB

Raw Blame

The following results are obtained with `lzbench 1.8` with the `-t16,16 -eall` options using 1 core of Intel Core i7-8700K, Ubuntu 18.04.3 64-bit, and clang 9.0.1 with "silesia.tar" which contains tarred files from Silesia compression corpus. The results are sorted by ratio. The results in **bold** show pareto frontier on compression ratio.

Compressor name	Compress.	Decompress.	Compr. size	Ratio
lzlib 1.11 -9	**1.82 MB/s**	**76 MB/s**	48296889	22.79
fastlzma2 1.0.1 -10	**3.99 MB/s**	**105 MB/s**	48666065	22.96
lzma 19.00 -9	2.66 MB/s	**107 MB/s**	48707450	22.98
xz 5.2.4 -9	2.62 MB/s	88 MB/s	48745306	23.00
fastlzma2 1.0.1 -8	**5.18 MB/s**	103 MB/s	49126740	23.18
xz 5.2.4 -6	2.95 MB/s	89 MB/s	49195929	23.21
lzma 19.00 -5	3.28 MB/s	103 MB/s	49710307	23.45
lzlib 1.11 -6	2.82 MB/s	74 MB/s	49777495	23.49
csc 2016-10-13 -5	3.86 MB/s	77 MB/s	49801577	23.50
brotli 2019-10-01 -11	0.63 MB/s	**451 MB/s**	50412404	23.79
fastlzma2 1.0.1 -5	**7.44 MB/s**	103 MB/s	51209571	24.16
zstd 1.4.3 -22	2.28 MB/s	**865 MB/s**	52738312	24.88
tornado 0.6a -16	2.15 MB/s	207 MB/s	53257046	25.13
csc 2016-10-13 -3	**9.38 MB/s**	71 MB/s	53477914	25.23
zstd 1.4.3 -18	3.58 MB/s	**912 MB/s**	53690572	25.33
fastlzma2 1.0.1 -3	**11 MB/s**	94 MB/s	54023837	25.49
bzip2 1.0.8 -9	**15 MB/s**	41 MB/s	54572811	25.75
lzham 1.0 -d26 -1	2.98 MB/s	340 MB/s	54740589	25.83
tornado 0.6a -13	6.94 MB/s	202 MB/s	55614072	26.24
bzip2 1.0.8 -5	**16 MB/s**	44 MB/s	55724395	26.29
xz 5.2.4 -3	6.76 MB/s	84 MB/s	55745125	26.30
csc 2016-10-13 -1	**21 MB/s**	73 MB/s	56201092	26.52
lzlib 1.11 -3	6.81 MB/s	69 MB/s	56320674	26.57
brotli 2019-10-01 -8	10 MB/s	533 MB/s	57140168	26.96
zstd 1.4.3 -15	7.12 MB/s	**1024 MB/s**	57167422	26.97
lzma 19.00 -4	14 MB/s	95 MB/s	57201645	26.99
tornado 0.6a -10	5.73 MB/s	192 MB/s	57588241	27.17
lzma 19.00 -2	**25 MB/s**	91 MB/s	58867911	27.77

Lzma 19.00 -9 decompression as fast as fastlzma2 and probably more accessible?

2893742274 1756* 838* 4 pcompress 3.-l14 -s60

2933199750 9520 412* 4 packet 1.9 -mx -h8 -b5 -r -s