Zstandard

Star 4,838

Zstandard is a real-time compression algorithm, providing high compression ratios. It offers a very wide range of compression / speed trade-off, while being backed by a very fast decoder (see benchmarks below). It also offers a special mode for small data, called dictionary compression, and can create dictionaries from any sample set. Zstandard library is provided as open source software using a BSD license.

Benchmarks

For comparison, several fast compression algorithms were tested and compared on a Linux Mint Debian edition server (`Linux version 4.8.0-1-amd64`), with a Core i7-6700K CPU @ 4.0GHz, using lzbench (https://github.com/inikep/lzbench), an open-source in-memory benchmark by @inikep compiled with gcc 6.3.0, on the Silesia compression corpus (http://sun.aei.polsl.pl/~sdeor/index.php?page=silesia).

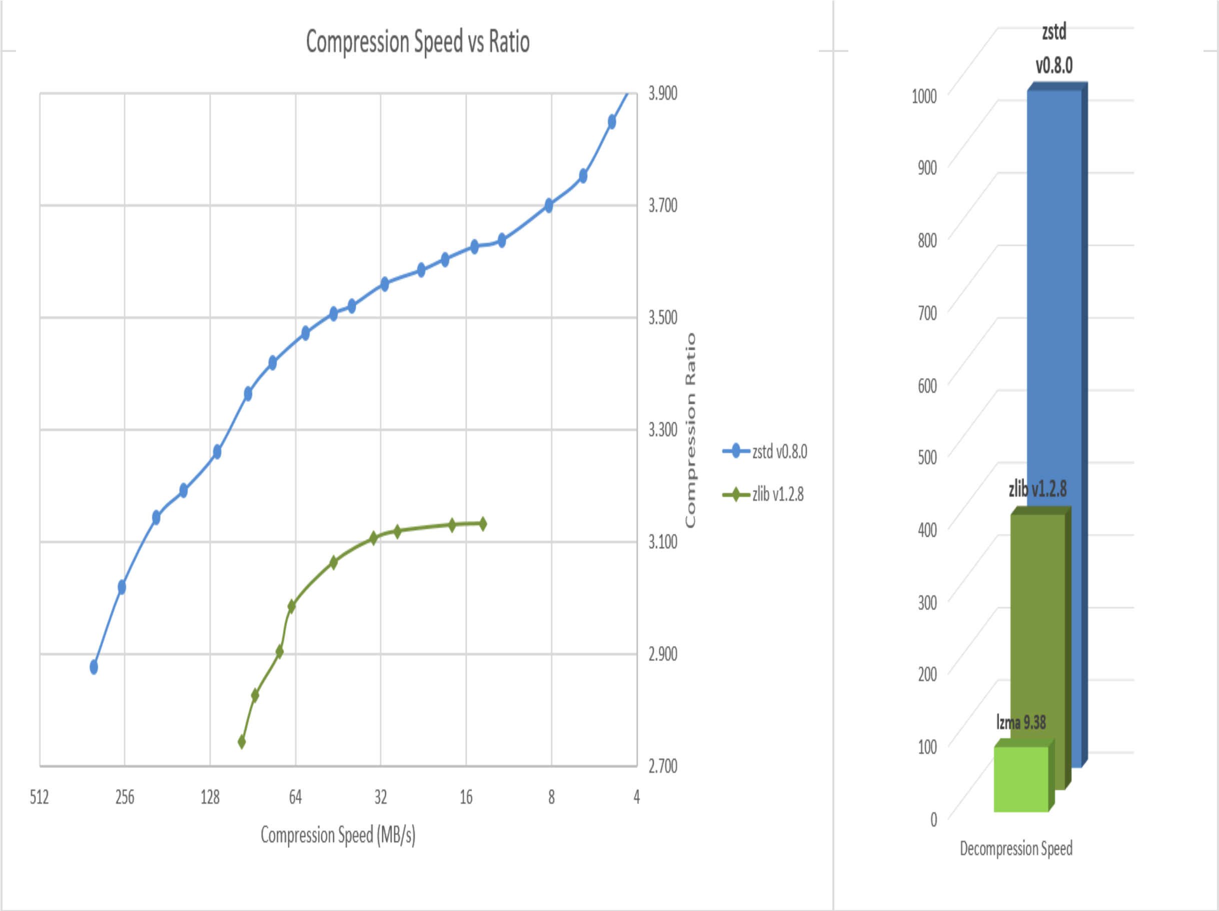
Compressor name	Ratio	Compression	Decompress.
zstd 1.1.3 -1	2.877	430 MB/s	1110 MB/s
zlib 1.2.8 -1	2.743	110 MB/s	400 MB/s
brotli 0.5.2 -0	2.708	400 MB/s	430 MB/s
quicklz 1.5.0 -1	2.238	550 MB/s	710 MB/s

Compressor name	Ratio	Compression	Decompress.
lzo1x 2.09 -1	2.108	650 MB/s	830 MB/s
lz4 1.7.5	2.101	720 MB/s	3600 MB/s
snappy 1.1.3	2.091	500 MB/s	1650 MB/s
lzf 3.6 -1	2.077	400 MB/s	860 MB/s

Zstd can trade compression speed for stronger compression ratios. It is configurable by small increment. Decompression speed is preserved and remain roughly the same at all settings, a property shared by most LZ compression algorithms, such as zlib (http://www.zlib.net/) or lzma.

The following tests were run on a Core i7-3930K CPU @ 4.5GHz, using lzbench (https://github.com/inikep/lzbench), an open-source in-memory benchmark by @inikep compiled with gcc 5.2.1, on the Silesia compression corpus (http://sun.aei.polsl.pl/~sdeor/index.php?page=silesia).

Compression Speed vs Ratio	Decompression Speed



Several algorithms can produce higher compression ratio but at slower speed, falling outside of the graph. For a larger picture including very slow modes, click on this link (https://raw.githubusercontent.com/facebook/zstd/master/doc/images/DCspeed5.png) .

The case for Small Data compression

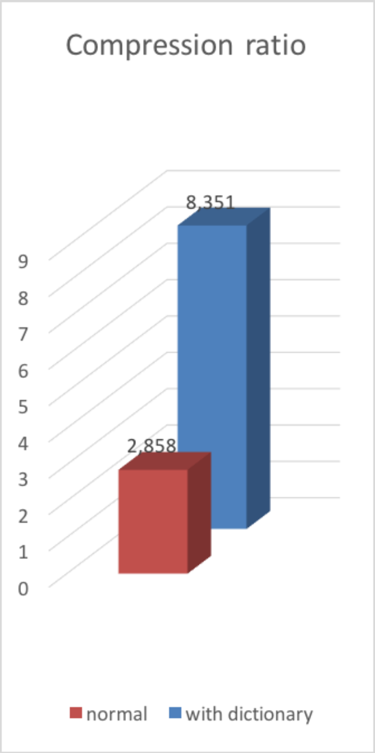
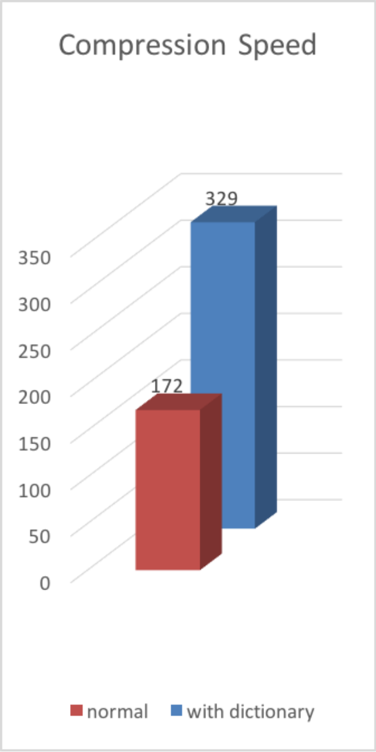
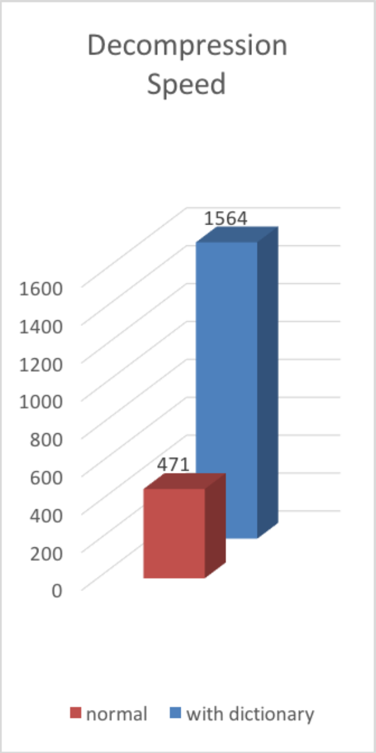
Previous charts provide results applicable to typical file and stream scenarios (several MB). Small data comes with different perspectives.

The smaller the amount of data to compress, the more difficult it is to compress. This problem is common to all compression algorithms, and reason is, compression algorithms learn from past data how to compress future data. But at the beginning of a new data set, there is no "past" to build upon.

To solve this situation, Zstd offers a **training mode**, which can be used to tune the algorithm for a

To solve this situation, Zstd offers a **training mode**, which can be used to tune the algorithm for a selected type of data. Training Zstandard is achieved by provide it with a few samples (one file per sample). The result of this training is stored in a file called "dictionary", which must be loaded before compression and decompression. Using this dictionary, the compression ratio achievable on small data improves dramatically.

The following example uses the `github-users` sample set (https://github.com/facebook/zstd/releases/tag/v1.1.3), created from github public API (https://developer.github.com/v3/users/#get-all-users). It consists of roughly 10K records weighting about 1KB each.

Compression Ratio	Compression Speed	Decompression Speed
 	 	

These compression gains are achieved while simultaneously providing *faster* compression and decompression speeds.

Training works if there is some correlation in a family of small data samples. The more data-specific a dictionary is, the more efficient it is (there is no *universal dictionary*). Hence, deploying one dictionary per type of data will provide the greatest benefits. Dictionary gains are mostly effective in the first few KB. Then, the compression algorithm will gradually use previously decoded content to better compress the rest of the file.

Dictionary compression How To :

1) Create the dictionary

```
zstd --train FullPathToTrainingSet/* -o dictionaryName
```

2) Compress with dictionary

```
zstd -D dictionaryName FILE
```

3) Decompress with dictionary

```
zstd -D dictionaryName  --decompress FILE.zst
```

A rich API set :

Zstandard API is designed with learning curve in mind. At the top, you'll find simple methods, using trivial arguments and behavior. Then, at each new paragraph, the API introduces new concepts and parameters, giving gradually more control for advanced usages.

You can learn more about Zstandard API by reading its documentation (http://facebook.github.io/zstd/zstd_manual.html).

Bindings for other languages

Should you need Zstandard in another language than reference C (https://github.com/facebook/zstd), here is a list of known bindings and their authors :

Language	**Author**	**URL**
Java	Luben Karavelov	https://github.com/luben/zstd-jni (https://github.com/luben/zstd-jni)

Language	**Author**	**URL**

Rust	Alexandre Bury	https://crates.io/crates/zstd (https://crates.io/crates/zstd)
Python	Gregory Szorc	https://pypi.python.org/pypi/zstandard (https://pypi.python.org/pypi/zstandard)
C#	SKB Kontur	https://github.com/skbkontur/ZstdNet (https://github.com/skbkontur/ZstdNet)
Node.js streams	albertdb	https://www.npmjs.com/package/node-zstandard (https://www.npmjs.com/package/node-zstandard)
Node.js buffers	Zwb	https://www.npmjs.com/package/node-zstd (https://www.npmjs.com/package/node-zstd)
PHP	Kamijo	https://github.com/kjdev/php-ext-zstd (https://github.com/kjdev/php-ext-zstd)
Perl	Jiro Nishiguchi	https://metacpan.org/release/Compress-Zstd (https://metacpan.org/release/Compress-Zstd)
Swift	Anatoli Peredera	https://github.com/omniprog/SwiftZSTD (https://github.com/omniprog/SwiftZSTD)
Go	Vianney Tran	https://github.com/DataDog/zstd (https://github.com/DataDog/zstd)
Ruby	Jarred Holman	https://github.com/jarredholman/ruby-zstd (https://github.com/jarredholman/ruby-zstd)
D	Masahiro Nakagawa	https://code.dlang.org/packages/zstd (https://code.dlang.org/packages/zstd)
Ada	John Marino	https://github.com/jrmarino/zstd-ada (https://github.com/jrmarino/zstd-ada)
Lua	Soojin Nam	https://github.com/sjnam/lua-resty-zstd (https://github.com/sjnam/lua-resty-zstd)

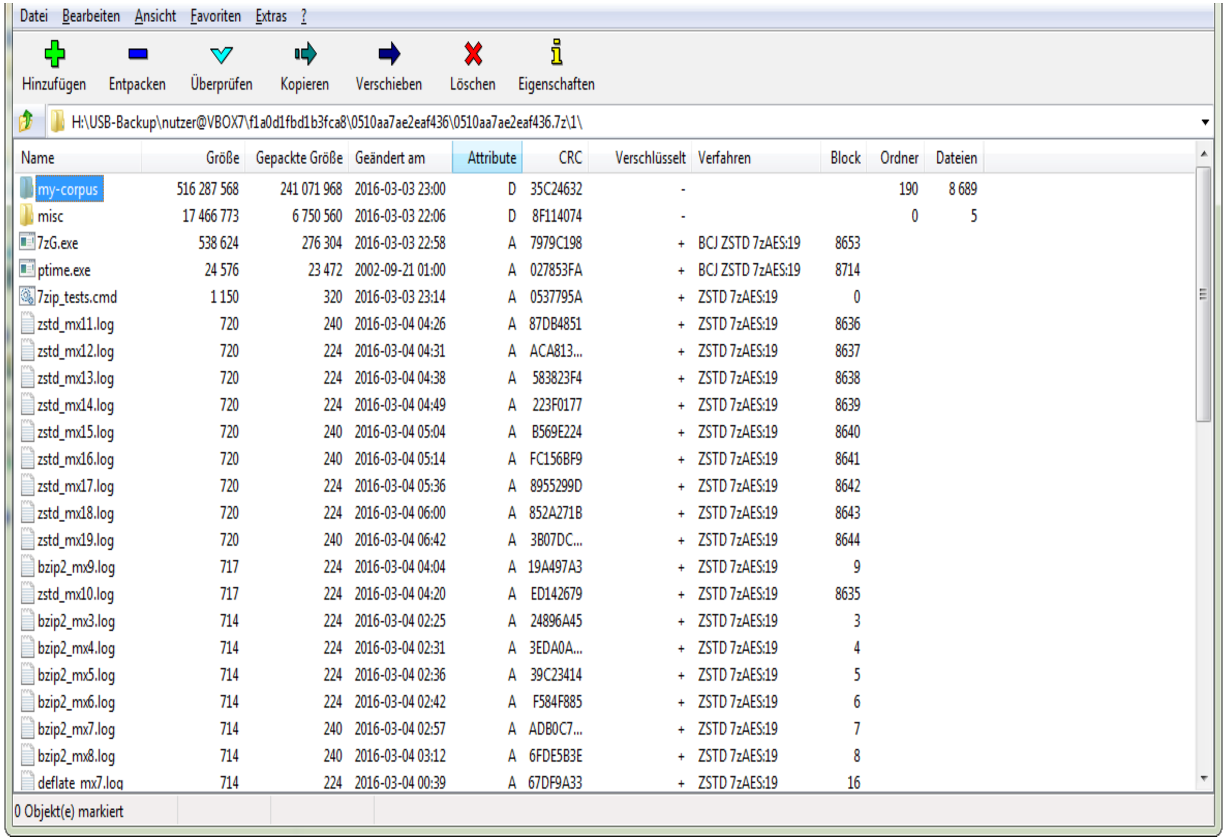
Language	Author	URL

Haskell	Bryan O'Sullivan	https://github.com/facebookexperimental/hs-zstd (https://github.com/facebookexperimental/hs-zstd)
Erlang	Yuki Ito	https://hex.pm/packages/zstd (https://hex.pm/packages/zstd)
Visual Basic 6	Tanner_H	http://www.vbforums.com/showthread.php?840413-Compression-in-VB6-modern-solutions (http://www.vbforums.com/showthread.php?840413-Compression-in-VB6-modern-solutions)
OCaml	ygrek	https://opam.ocaml.org/packages/zstd/ (https://opam.ocaml.org/packages/zstd/)
Delphi	Razor12911	http://encode.ru/threads/2119-Zstandard?p=49075&viewfull=1#post49075 (http://encode.ru/threads/2119-Zstandard?p=49075&viewfull=1#post49075)

Graphical User Interface

Tino Reichardt has developed and hosts a version of 7-zip archive manager with Zstandard (https://mcmilk.de/projects/7-Zip-ZStd/).

H:\USB-Backup\nutzer@VBOX7\f1a0d1fbd1b3fca8\0510aa7ae2eaf436\0510aa7ae2eaf436.7z\1\



H:\USB-Backup\nutzer@VBOX7\f1a0d1fbd1b3fca8\0510aa7ae2eaf436\0510aa7ae2eaf436.7z\1\

Name	Größe	Gepackte Größe	Geändert am	Attribute	CRC	Verschlüsselt	Verfahren	Block	Ordner	Dateien
my-corpus	516 287 568	241 071 968	2016-03-03 23:00	D	35C24632	-			190	8 689
misc	17 466 773	6 750 560	2016-03-03 22:06	D	8F114074	-			0	5
7zG.exe	538 624	276 304	2016-03-03 22:58	A	7979C198	+	BCJ ZSTD 7zAES:19	8653		
ptime.exe	24 576	23 472	2002-09-21 01:00	A	027853FA	+	BCJ ZSTD 7zAES:19	8714		
7zip_tests.cmd	1 150	320	2016-03-03 23:14	A	0537795A	+	ZSTD 7zAES:19	0		
zstd_mx11.log	720	240	2016-03-04 04:26	A	87DB4851	+	ZSTD 7zAES:19	8636		
zstd_mx12.log	720	224	2016-03-04 04:31	A	ACA813...	+	ZSTD 7zAES:19	8637		
zstd_mx13.log	720	224	2016-03-04 04:38	A	583823F4	+	ZSTD 7zAES:19	8638		
zstd_mx14.log	720	224	2016-03-04 04:49	A	223F0177	+	ZSTD 7zAES:19	8639		
zstd_mx15.log	720	240	2016-03-04 05:04	A	B569E224	+	ZSTD 7zAES:19	8640		
zstd_mx16.log	720	240	2016-03-04 05:14	A	FC156BF9	+	ZSTD 7zAES:19	8641		
zstd_mx17.log	720	224	2016-03-04 05:36	A	8955299D	+	ZSTD 7zAES:19	8642		
zstd_mx18.log	720	224	2016-03-04 06:00	A	852A271B	+	ZSTD 7zAES:19	8643		
zstd_mx19.log	720	240	2016-03-04 06:42	A	3B07DC...	+	ZSTD 7zAES:19	8644		
bzip2_mx9.log	717	224	2016-03-04 04:04	A	19A497A3	+	ZSTD 7zAES:19	9		
zstd_mx10.log	717	224	2016-03-04 04:20	A	ED142679	+	ZSTD 7zAES:19	8635		
bzip2_mx3.log	714	224	2016-03-04 02:25	A	24896A45	+	ZSTD 7zAES:19	3		
bzip2_mx4.log	714	224	2016-03-04 02:31	A	3EDA0A...	+	ZSTD 7zAES:19	4		
bzip2_mx5.log	714	224	2016-03-04 02:36	A	39C23414	+	ZSTD 7zAES:19	5		
bzip2_mx6.log	714	224	2016-03-04 02:42	A	F584F885	+	ZSTD 7zAES:19	6		
bzip2_mx7.log	714	240	2016-03-04 02:57	A	ADB0C7...	+	ZSTD 7zAES:19	7		
bzip2_mx8.log	714	240	2016-03-04 03:12	A	6FDE5B3E	+	ZSTD 7zAES:19	8		
deflate_mx7.log	714	224	2016-03-04 00:39	A	67DF9A33	+	ZSTD 7zAES:19	16		

0 Objekt(e) markiert

Zstandard is used by :

Cloud solutions

amazon (https://aws.amazon.com/redshift/) Redshift (https://aws.amazon.com/aboutaws/whats-new/2017/01/amazon-redshift-now-supports-the-zstandard-high-data-compression-encoding-and-two-new-aggregate-functions/)

(https://www.mercurial-scm.org/) Mercurial (http://blog.deveo.com/whats-new-in-mercurial-hg-4-1/)

(https://docs.taskcluster.net/) TaskCluster (https://github.com/taskcluster/taskcluster-worker/pull/114)

(http://ceph.com/) Ceph (https://github.com/ceph/ceph/tree/master/src/compressor/zstd)

Databases

(https://hadoop.apache.org/) Hadoop (https://issues.apache.org/jira/browse/HADOOP-13578)

(http://rocksdb.org/) RocksDB (https://twitter.com/rocksdb/status/771387757306388480)

(http://www.wiredtiger.com/) WiredTiger (https://github.com/wiredtiger/wiredtiger/tree/master/ext/compressors/zstd)

(https://redis.io/) Redis (https://github.com/chadnickbok/redis-zstd-module)

(https://clickhouse.yandex/) Clickhouse

(https://github.com/yandex/ClickHouse/tree/master/contrib/libzstd)

(http://groonga.org/) Groonga (http://groonga.org/en/blog/2016/11/29/groonga-6.1.1.html)

(https://tarantool.org/) Tarantool (https://github.com/tarantool/tarantool/blob/1.8/cmake/BuildZSTD.cmake)

(https://github.com/XeLabs/tokudb) TokuDB (https://github.com/XeLabs/tokudb/commit/7ba55b2ee6aaac0980daa0a6b269a5c551da4ba6)

Serialization

(http://www.fstpackage.org/) FST (https://github.com/fstpackage/fst/tree/master/src/ZSTD)

(http://www.blosc.org/) Blosc (http://www.blosc.org/blog/zstd-has-just-landed-in-blosc.html)

(http://bcolz.blosc.org/en/latest/) bcolz (https://github.com/Blosc/bcolz/tree/master/c-blosc/internal-complibs/zstd-1.1.2)

(https://pypi.python.org/pypi/mrcz) mrcz (https://github.com/em-MRCZ/c-mrcz)

(http://www.well.ox.ac.uk/~gav/bgen_format/bgen_format.html) bgen

(http://www.well.ox.ac.uk/~gav/bgen_format/bgen_format.html)

(https://developer.mozilla.org/en-US/docs/Mozilla/Gecko) Gecko (https://bug635044.bugzilla.mozilla.org/show_bug.cgi?id=1316183)

(https://support.hdfgroup.org/HDF5) HDF5 (https://github.com/aparamon/HDF5Plugin-Zstandard)

(https://github.com/Sereal/Sereal) Sereal (https://github.com/Sereal/Sereal/tree/master/Perl/shared/zstd)

Network

(https://code.facebook.com/projects/1410559149202582/fbthrift/) fbthrift (https://github.com/facebook/fbthrift/commit/ec42813f0ced737617d4614900ef3a96c1f3d17f)

(https://code.facebook.com/projects/676603015770415/proxygen/) proxygen (https://github.com/facebook/proxygen/blob/master/proxygen/lib/utils/ZstdStreamDecompressor.h)

(https://code.facebook.com/projects/470946523057396/mcrouter/) mcrouter (https://github.com/facebook/mcrouter/blob/master/mcrouter/lib/ZstdCompressionCodec.h)

(https://rspamd.com/) Rspamd (https://rspamd.com/announce/2016/11/21/rspamd-1.4.0.html)

Other