The Compression Trade-off

Free Sample · One Chapter

Choosing a Codec for Ratio, Speed, and Latency

Md Nasim Sheikh

www.nasimstg.dev

About this sample

This is one complete chapter from *The Compression Trade-off*, a short, data-driven guide to choosing a lossless codec. The figures are measured by a reproducible Docker harness you can run on your own data. The full book explains how the codecs work and turns the comparison into a selection framework. The chapter below, “The Measured Comparison,” reports the head-to-head result.

3

The Measured Comparison

Rather than assert which codec is fast, this book measures a fixed set on the same data. The harness (in the `bench/` directory) compresses one corpus with each codec and level, single-threaded, and records compression ratio, compress throughput, and decompress throughput. The corpus is real source code (the Python standard library, about 10 MB); the appendix gives the exact setup.

3.1 The full picture

Table 3.1 lists every codec and level measured. Read it as three columns in tension: ratio, compress speed, and decompress speed.

Table 3.1. Measured results on a 10.4 MB source-code corpus, single-threaded (AMD Ryzen 7 5700X, Docker, 2026). Ratio is hardware-independent; throughput is machine-specific.

Codec	Level	Ratio	Compress (MB/s)	Decompress (MB/s)
LZ4	1	2.76	**328.7**	**818.8**
LZ4	9	3.84	34.8	804.3
gzip	6	4.39	40.4	249.4
gzip	9	4.44	10.7	255.6
Zstd	3	4.48	196.9	567.1
Zstd	9	5.24	50.8	549.2
Zstd	19	6.02	2.7	416.1
Brotli	5	5.16	29.0	396.5
Brotli	11	**6.17**	0.8	389.0
XZ	6	6.13	3.1	139.9
XZ	9	6.13	2.9	141.2

Figure 3.1 plots ratio against compression throughput, the primary trade-off. The vertical axis is logarithmic, because compress speed spans from under 1 to over 300 MB/s.

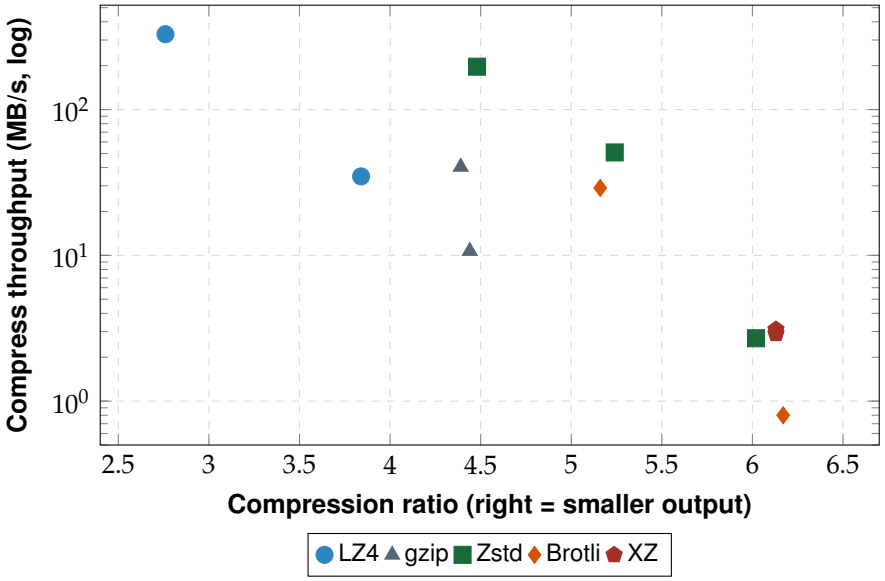

Figure 3.1. Compression ratio versus compress throughput (log scale). Up is faster, right is smaller; the useful codecs form a frontier from LZ4 (fast, low ratio) down to XZ and Brotli (slow, high ratio), with Zstd spanning the middle.

Reading Figure 3.1 and Table 3.1

The frontier: LZ4 at the top-left (329 MB/s, ratio 2.76) and XZ / Brotli-11 at the bottom-right (ratio ∼6.1, under 3 MB/s) mark the extremes. Zstd runs down the middle.

Zstd-3 dominates gzip: at ratio 4.48 it is both slightly smaller *and* nearly five times faster to compress than gzip-6 (197 vs 40 MB/s), and it decompresses more than twice as fast. gzip is Pareto-dominated here.

Decompression is a different axis: LZ4 decompresses at ∼820 MB/s and Zstd at ∼550, while XZ manages only ∼140. For read-heavy data, XZ's high ratio comes with a real read-time cost.

3.2 What the numbers mean

Three patterns hold across the data. First, the top ratios (about 6.1) are reached only at very low compression speed: Zstd-19, XZ, and Brotli-11 all sit near or below 3 MB/s. Second, most of the ratio is available cheaply: Zstd-3 captures

a 4.48 ratio at 197 MB/s, close to gzip's ratio at a fraction of the cost, so the first large gain is nearly free and the last few percent is expensive. Third, compression and decompression diverge: LZ4 and Zstd decompress several times faster than XZ, which decides read-heavy workloads regardless of ratio.

Key Insight

The single most useful result is that Zstd has largely replaced gzip on the frontier: at a similar ratio it compresses and decompresses several times faster, and it can be turned up to match XZ-class ratios when needed. The remaining reasons to reach past Zstd are a specific need for LZ4's raw speed or for the last few percent of ratio from XZ or Brotli.

These are measurements on one corpus and one machine. Ratios will shift with the data (text compresses better than already-compressed or random bytes) and throughput with the hardware. The appendix documents the setup, and the harness reproduces the table on your own data, which is the comparison that should decide a production choice.

Get the full book

The complete *Compression Trade-off* explains all five codecs, the selection framework, the practical details (dictionaries, block size, threads), and a benchmarking protocol, and ships the reproducible harness.

www.nasimstg.dev

Open-source benchmark harness included.