ANS LEGACY · ADOPTION AND IMPACT STUDY · VERSION 2.2 · 3 SEPTEMBER 2026

# Entropy coding at planetary scale

Where Asymmetric Numeral Systems are used, and what they save — installed base, users, data volumes and savings 2014–2026, extrapolated to 2030

**≈6.1B**
machines with an ANS decoder in 2026 (via Zstandard's FSE stage alone; Apple LZFSE 2.5B, JPEG XL 5.2B overlap this set)

**≈4.5B people**
whose data is coded with ANS through zstd-based services, browsers and operating systems

**≈188 EB/yr**
ANS-coded output written or transmitted in 2026 and charged once (range 80–571); the same year carries ≈224 EB of encoder and ≈639 EB of decoder operations; 93 % through zstd

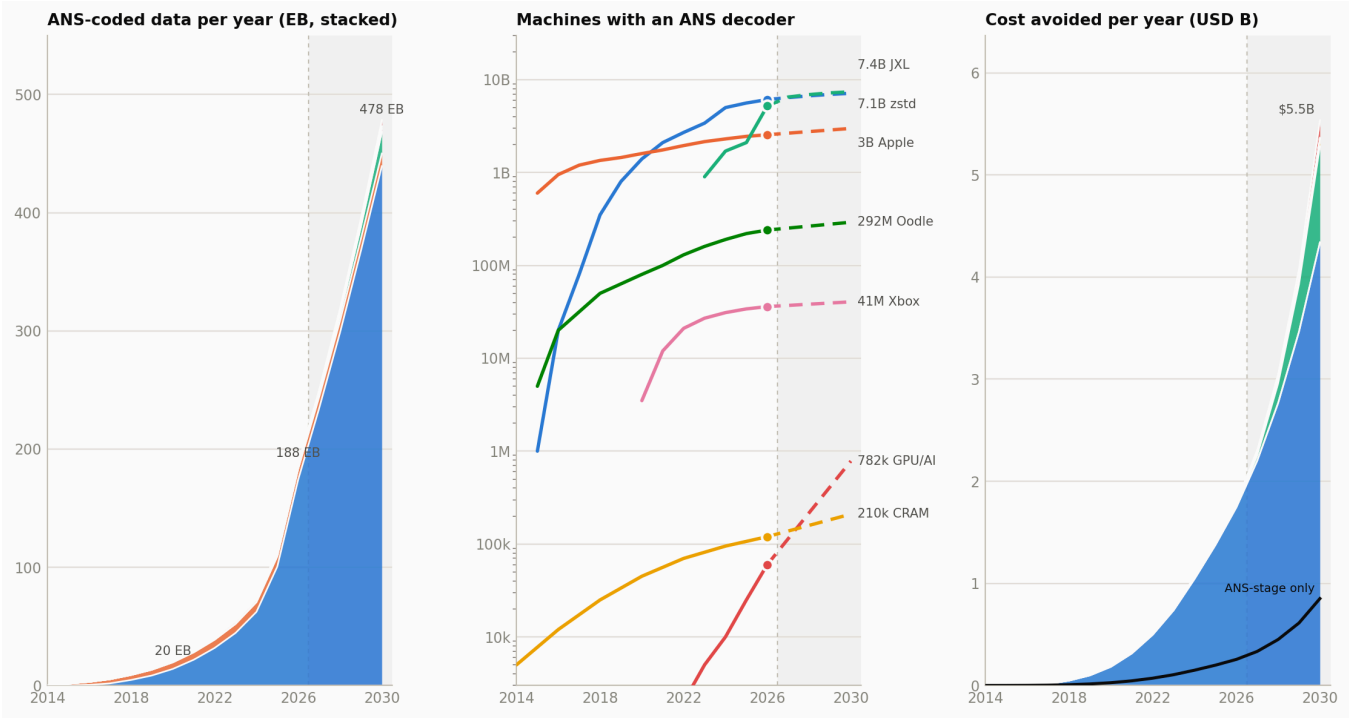
**≈20 EB/yr**
bytes avoided in 2026 relative to the codecs displaced; 3.5 EB/yr attributable to the ANS entropy stage itself

**$1.8B · $4.6B**
cost avoided in 2026 on two lenses: public/auditable, charging one stored copy (range $0.5B–$7.3B), and infrastructure-inclusive, charging replicas and backups; strict ANS-stage attribution $255M

**$6.2B**
cumulative cost avoided 2014–2026 (central); ≈78 EB of storage and transfer avoided

**≈478 EB/yr**
ANS-coded data in 2030 (central extrapolation; range 200–1,452), with $5.5B/yr avoided and $21.2B cumulative

**≈5.2B devices**
carry a JPEG XL decoder by end-2026: Chrome 155 ships it enabled by default (stable 6 Oct 2026), Firefox 157 on 29 Sep, Safari since 17. Xbox BCPack stays the only documented rANS decoder in consumer silicon (~36M consoles)




Left: compressed bytes coded with ANS-based formats per year, stacked by software (thin line: high scenario). Centre: installed base of machines carrying a decoder (log scale; lines overlap). Right: cost avoided per year, whole-codec (stacked) and ANS-stage only (black). Shaded: extrapolation 2027–2030.

**2013–14**
FSE (tANS) and rANS published; CRAM 3.0 adopts rANS

**2015–16**
Apple ships LZFSE to every iOS/OS X device; zstd 1.0; Oodle LZNA/BitKnit

**2017–20**
Linux 4.14, Draco, Kafka, Fedora/Arch packages, OpenZFS; Xbox Series X|S with hardware rANS (BCPack)

**2021–23**
CRAM 3.1 SIMD rANS; ALICE O² rANS at CERN; Ubuntu .deb; Windows 11; Safari 17 decodes JPEG XL

**2024–25**
Chrome 123 and Firefox 126 zstd; Cloudflare default; Steam to zstd; Python 3.14; JPEG AI standard; samtools defaults to CRAM 3.1

**2026**
Safari 26.3 zstd; DirectStorage 1.4 adds zstd to PC games (Mar); Apple LZRAVEN (rANS); Firefox 157 (29 Sep) and Chrome 155 (6 Oct) ship JPEG XL by default; NVIDIA recommends gANS for checkpoints

**Global**
Zstandard (FSE/tANS + Huffman): Linux, Android, Windows 11, all browsers incl. Safari 26.3, Steam, Meta, Cloudflare, Kafka, RocksDB, Parquet/Iceberg, PostgreSQL, Python 3.14, Discord's gateway, Fedora's Btrfs root

**Platform**
Apple LZFSE (2.5 B devices, 2015–) and LZRAVEN (rANS, OS 27 OTA); Oodle LZNA/BitKnit (rANS) and rarely-used tANS modes across ~240 M gaming machines

**Domain**
JPEG XL (rANS; Safari 17+, Firefox 157, Chrome 155 default-on), CRAM 3.x genomics (rANS Nx16), Xbox BCPack (hardware rANS), DirectStorage 1.4 zstd on PC, Draco 3D, ALICE O² at CERN (350-server rANS stage)

**Emerging AI**
nvCOMP gANS checkpoint compression (NVIDIA-recommended), token-native agent stores (ANS 3.3× over UTF-8), UCCL-Zip collectives, ANS-GEMM inference, JPEG AI, RAS accelerator

**Not ANS**
Brotli, AV1, GDeflate (the older PC DirectStorage path; 1.4 adds optional zstd), PS5 Kraken decoder, ZipNN, DFloat11, Cloudflare Unweight, Blackwell decompression engine, hipCOMP-core

Prepared for Jarosław Duda (Jagiellonian University). Compiled with Claude (Anthropic) from ~200 primary sources; incorporates an independent Sol/Kimi study, two rounds of review, and operator evidence from Yann Collet (zstd) and Jon Sneyers (JPEG XL) reported in Section 7.5. The central case is a **public-evidence floor**, not a best estimate: Zstandard's author states that several figures here are "way below reality" because infrastructure numbers are not published (Section 7.5). Evidence classes (Measured use / Default-on / Optional capability / Modeled / Research) follow each application. Model (`model.py`), yearly CSV tables and figures are delivered with this document; 2027–2030 values are extrapolations.

## Contents



Colour key used throughout: Zstandard (FSE/tANS) Apple LZFSE/LZRAVEN JPEG XL CRAM Xbox BCPack Oodle ANS modes Draco GPU/AI ANS ALICE O². Confidence tags on factual claims: confirmed primary source read; likely secondary or partial confirmation; estimate modelled or unverified. Evidence classes on applications: Measured use operator-published bytes or telemetry; Default-on verifiable default in a shipped product; Optional capability shipped and selectable, adoption unknown; Modeled plausible but not independently auditable; Research paper or prototype, no production assumption; Private undisclosed, excluded from totals.

# 1. Summary of findings

Twelve years after the 2013 ANS preprint, ANS entropy coding ships in the Linux kernel, in every major browser, on every Apple device, in the Xbox Series storage pipeline (in silicon), in the standard file format for aligned genomic reads, in NVIDIA's recommended GPU codec for model checkpoints and in the real-time data-reduction chain of CERN's ALICE experiment. Most of that reach arrives indirectly, through Zstandard's FSE stage; the largest *direct* ANS deployments are Apple's LZFSE (2.5 billion devices), Xbox Series BCPack (~36 million consoles, hardware rANS) and JPEG XL (2.5 billion devices with a decoder, with Firefox default-on on 29 September 2026 and Chrome's Intent to Ship approved but unscheduled).

### Which ANS-based compressors are actually widely used

Ranked by present-day reach, with the evidence class each claim rests on (details and sources in Section 3):

1. **Zstandard** (FSE = tANS for the literal-length, match-length and offset streams; Huffman for literals) is by a wide margin the most deployed ANS carrier. It is shipped in the Linux kernel since 4.14 (2017) and selectable for kernel images, modules, initramfs, SquashFS and Btrfs (the upstream default image compressor remains gzip); it is the package format of Fedora, Arch and Ubuntu; it is in OpenZFS 2.0, Kafka, RocksDB, MongoDB, PostgreSQL 15/16, Parquet/ORC/Iceberg/Databricks defaults, Windows 11 (Oct 2023), Chrome 123 and Firefox 126 (2024), Cloudflare's default (2024/25), Steam content delivery (2025), Python 3.14's standard library (2025), Safari 26.3 (Feb 2026), SQL Server 2025 and .NET 11. Web Almanac data shows zstd rising from 2.7 % of CDN-served responses (2024) to 12 % (2025). Meta reports that, before its 2025 OpenZL release, zstd made up "the vast majority of compression use at Meta". Default-on for most carriers; byte volumes Modeled.
2. **Apple LZFSE** (tANS) has shipped in every iOS/macOS device since 2015, is the default of the Compression framework and Apple Archive, and is used for the iOS kernel cache and APFS transparent compression; Apple's installed base passed 2.5 billion active devices in January 2026. In OS 27 (2026) Apple adds **LZMESH** (Huffman, replacing LZFSE for general use) and **LZRAVEN** (an adaptive 8-way interleaved rANS coder per third-party reverse engineering, replacing LZMA and used for OTA payloads). Default-on; OTA algorithm mix Modeled.
3. **JPEG XL** (rANS in libjxl at every effort except the lowest lossless setting) decodes on all Apple devices since Safari 17 / iOS 17 (Sept 2023) and on Windows 11 via Microsoft's extension (2025); it is written natively by the iPhone 16 Pro camera (ProRAW-JXL), DNG 1.7, DICOM (Supplement 232) and Adobe/Affinity/GIMP tooling. Chromium re-added JPEG XL (Rust jxl-rs) behind a flag in Chrome 145 (Feb 2026) and filed an *Intent to Ship* on 24 Aug 2026 with three approvals but no milestone; Mozilla ships default-on in Firefox 157 on 29 September 2026. Default-on decoders; web bytes Modeled scenario.
4. **CRAM 3.0/3.1** (rANS since 2014; SIMD rANS Nx16 since 2021) is the storage format of the European Nucleotide Archive, EGA, UK Biobank's 500k genomes, All of Us, Genomics England, Broad (archival), Sanger, Illumina DRAGEN and NVIDIA Parabricks output; samtools has written CRAM 3.1 by default since May 2025. Default-on; archive sizes Measured use, CRAM share modelled.
5. **Microsoft Xbox Series X|S "BCPack"** — the answer to the "DirectX" question — is, per Microsoft's GDK documentation, "a custom entropy coder designed specifically for BCn data … compressed using a rANS algorithm", implemented in the console's hardware decompression block since November 2020. PC DirectStorage's GDeflate is Deflate/Huffman-based and is *not* ANS. Default-on hardware; title share Modeled.
6. **Google Draco** (rANS) is the standard glTF geometry compressor (KHR_draco_mesh_compression, 2018), bundled in three.js, Babylon.js, Cesium, Unity and Blender; `draco3d` saw 18.2 million npm downloads in the 30 days to 29 Aug 2026. Byte volumes are small, reach is large. Default-on in glTF pipelines.
7. **GPU/AI**: NVIDIA nvCOMP has shipped a GPU rANS codec since v2.2 (Feb 2022), rebranded "gANS", with float16 and (2026) float8 modes; NVIDIA's April 2026 blog recommends it as the default for GPU-resident checkpoint compression (1.46–1.48× on BF16 weights at ~530 GB/s vs 1.27× for zstd at ~16 GB/s; a 70 B-parameter job writes 782 GB per checkpoint, 1.13 PB/month). Meta's dietGPU (rANS, 2022) was archived in June 2026; NeuZip uses ANS via nvCOMP. Beyond checkpoints, 2025–26 research puts ANS into token-native storage for agents and RAG (3.3× over UTF-8 with a static unigram ANS coder), into GPU collectives (UCCL-Zip), into inference kernels that multiply directly on ANS-coded weights (ANS-GEMM, ISCA 2026), into dynamic 3D-Gaussian streaming (GS-NFS) and into a hardware rANS accelerator design (RAS). The popular weight compressors ZipNN, DFloat11 and Cloudflare's Unweight use Huffman, not ANS. Production AI volume is unknown; the model carries an illustrative 0.1 EB/yr (range 0.01–0.4). Optional capability / Research.
8. **Oodle** needs a correction to the common assumption: LZNA (2015) and BitKnit (2015/16) are rANS codecs, but Kraken, Mermaid and Leviathan are Huffman-based; tANS was added as an optional mode in 2018 and, per Fabian Giesen, is "rarely used". The PS5's hardware Kraken decoder (reported to implement the pre-2018 Oodle 2.2 feature set) is therefore not an ANS deployment. Optional capability.
9. **Specialist and niche deployments**: CERN's **ALICE O²** uses a custom rANS coder as the final lossless stage of its Run-3 online data reduction (design point 300 → 90 GB/s on a 350-server GPU farm; 423 PB of compressed raw data stored by Feb 2025) Measured use; the **LOLZ** and (probably) **RAZOR** archivers — adaptive rANS + LZ — are the standard compressors of the game-"repack" scene (excluded from totals; Modeled, very wide range); **JPEG AI** (ISO/IEC 6048-1:2025) standardised ANS in a learned image codec but has no deployment yet Research.

### Trends and extrapolation

Measured in compressed bytes, ANS-coded data grew from about 20 EB/yr in 2020 to 74 EB/yr in 2024 and an estimated 188 EB/yr in 2026 (range 80–571) — a step change driven by browsers and Cloudflare turning on HTTP zstd, Steam re-encoding its 100 EB/yr of game delivery from LZMA to zstd, and data-platform defaults (Databricks, Athena/Iceberg, ClickHouse Cloud, Elasticsearch) switching to zstd. The central extrapolation reaches 478 EB/yr by 2030 (range 200–1,452); JPEG XL adds ~14 EB/yr by 2030 if Chrome ships default-on in 2027, and GPU-side ANS a few EB/yr. Cumulative bytes avoided relative to the codecs displaced reach ≈78 EB by end-2026 and ≈241 EB by 2030 (whole-codec), of which ≈14 EB and ≈43 EB are attributable to the ANS entropy stage itself. The corresponding cost avoided — transfer priced once, stored savings charged for as long as they remain resident — is $1.8B in 2026 (range $0.5B–$7.3B), ≈$6.2B cumulative since 2014 and ≈$21.2B by 2030 (central; the high-retention scenario of version 1.1 gives $2.9B and $35.2B).

**How to read the savings numbers.** Two attributions are reported throughout. *Whole-codec* savings compare each ANS-based format with the format it displaced in practice (zstd vs gzip/Snappy/LZ4, CRAM vs BAM, JPEG XL vs JPEG, BCPack vs zlib). *ANS-stage* savings isolate the ratio gain of tANS/rANS over the Huffman coder each codec would otherwise have used — typically 2–5 % of output for LZ codecs, ~15 % for CRAM, ~5 % for JPEG XL, ~20 % for GPU float codecs. The truth lies between the two: ANS did not create zstd's LZ77 improvements, but it did let Collet, Bainville, Bonfield and the JPEG XL team ship arithmetic-coding-class ratios at Huffman-class speed, which is why these codecs were adopted at all. Speed and energy effects (Section 4.6) are not monetised.

**What changed in version 2.0.** After comparison with an independent Sol/Kimi study and Sol's review of version 1.1 (Section 7), the model now separates durable from transient storage (retention 80 % vs 10 % per year instead of a flat 85 %), treats Steam's zstd share as a modelled scenario (40 % of 2026 bytes instead of 70 %), splits data-platform volumes into durable and transient components with evidence labels, lowers the Apple and GPU/AI volumes to their better-supported levels, excludes the repack scene from totals, and reports the previous assumptions as an explicit high scenario. The 2026 cost figure moved from $2.5 B to $1.8B; the reach figures did not change.

**Version 2.1** (3 September 2026) adds a verification of Kimi's follow-up proposals (Section 7.4): JPEG AI's normative "me-tANS" coder and ByteDance's filings around it, Huawei's and Inspur's ANS/FSE coder patents, zstd defaults in PolarDB IMCI and OceanBase, and a clarification that server zstd accelerators (Kunpeng KAEzstd, Intel QAT) offload only the LZ77 stage. No headline number changed.

**Version 2.2** (3 September 2026) responds to two operator comments — Yann Collet, the author of Zstandard, writing that "several of these numbers are way below reality ... infrastructure numbers do not come public", and Jon Sneyers reporting that Cloudinary alone serves 0.5–1 billion JPEG XL images a day — and to a verification addendum built on them (Section 7.5). The first consequence is a change of framing: the central case in this report is a *public-evidence floor*, not a best estimate of reality, and for the components no operator publishes the truth is more likely to lie in the upper half of the stated range. Three things changed. The report now separates *economically countable output* from *encoder* and *decoder operations*, because one number was doing three jobs; the money model charges replicas and backups in a second, infrastructure-inclusive lens alongside the auditable single-copy one; and JPEG XL's Chrome milestone is no longer a guess — ChromeStatus carries milestone 155, stable 6 October 2026. Four missing carriers were added (Discord's gateway, Fedora's Btrfs root, Box transfers, Cloudflare's cache-transcoding prototype), DirectStorage 1.4's optional zstd path was added, four over-strong "default" labels were downgraded, and one challenged claim was re-checked and upheld: Elasticsearch 8.16/9.0 really does use zstd for `best_compression` (PR #112665, merged 13 September 2024, ≈12 % less storage than DEFLATE), while OpenSearch offers zstd codecs opt-in beside its LZ4 default.

## 2. Scope, definitions and method

### 2.1 What counts as "ANS-based"

A compressor is counted if its entropy-coding stage uses any member of the ANS family — tANS/FSE (tabled), rANS (range), or uABS/rABS (binary) — for a material share of its output. Four tiers are distinguished:

- **Direct ANS codecs**: the ANS coder is the codec's own entropy stage (LZFSE, LZRAVEN, CRAM rANS, JPEG XL, Draco, BCPack, Oodle LZNA/BitKnit, nvCOMP gANS, dietGPU, ALICE O², LOLZ, Brunsli, PIK, JPEG AI).
- **Indirect via Zstandard**: zstd uses FSE (tANS) for the three sequence streams and for Huffman-table headers, and Huffman (huff0) for literals (RFC 8878). Everything that embeds zstd — the Linux kernel, browsers, databases, Steam — is therefore an ANS carrier. This tier dominates the byte counts.
- **Optional or rare modes**: codecs where ANS is available but seldom selected (Oodle Kraken/Mermaid/Leviathan tANS mode; JPEG XL's prefix-code fallback is the inverse case). Counted only for the share estimated to use the ANS path.
- **Specialist, niche and emerging**: ALICE O² (counted), game-repack archivers (described, excluded from totals), and the 2025–26 AI applications (described, illustrative volume only).

Codecs commonly assumed to use ANS but which do not (AV1/AVIF, Brotli, HEIC/HEVC, GDeflate, Kraken's default modes, ZipNN, DFloat11, Cloudflare Unweight, NVIDIA's Blackwell decompression engine, AMD hipCOMP-core) were checked and are listed in Section 3.13 so they are not double-counted. Apache Arrow is a special case: its in-memory format is codec-independent, and Arrow IPC offers optional LZ4 or zstd buffer compression (default uncompressed) — an optional zstd carrier, with no volume assigned.

### 2.2 Evidence classes

Following the review of version 1.1, every application row carries one of six evidence labels, and the byte totals are built only from the first four:

| Label | Meaning | Examples in this report |
|---|---|---|
| Measured use | Operator-published byte volumes or codec telemetry. | Steam's 100 EB/yr total (not its zstd share); Web Almanac's zstd response share; ENA/UK Biobank/ALICE stored volumes; Apple's 2.5 B active devices; NVIDIA's checkpoint case study. |
| Default-on | Verifiable default in a shipped product or standard. | zstd in Fedora/Arch/Ubuntu packages, Databricks, Iceberg, ClickHouse Cloud, RNTuple, Zarr, bazel-remote; LZFSE in Apple Archive; CRAM 3.1 in samtools 1.22; JPEG XL decoders in Safari and Firefox 157; BCPack in the Xbox Series SoC. |
| Optional capability | Shipped and selectable; adoption share unknown. | zstd in Kafka, Parquet, MongoDB, Kubernetes OCI layers, restic, GDAL; Oodle tANS mode; nvCOMP gANS; Arrow IPC zstd. |
| Modeled | Plausible, anchored on public statements or proxies, not independently auditable. | zstd share of Steam bytes; data-platform byte volumes; Apple OTA payload volumes; Xbox title share; Oodle ANS-mode share; GPU/AI volume. |
| Research | Paper, prototype or standard without deployment; no production volume assumed. | dietGPU, NeuZip, ANS-GEMM, UCCL-Zip, token-native storage, GS-NFS, RAS, JPEG AI, PILC/OSOA, MANS. |
| Private | Rumoured or undisclosed internal use; excluded from totals. | Huawei, Tencent, Baidu and other hyperscalers beyond downstream zstd use (their patent filings are reviewed in Sections 3.12 and 7.4); possible proprietary codecs at Google or Amazon. |

### 2.3 The quantity panel: seven metrics that are routinely confused

Version 2.2 adds this panel after Yann Collet observed (encode.su, 3 September 2026, quoted in Section 7.5) that the totals in version 2.1 looked low for zstd. The most likely cause was not a single missing deployment but a single number doing three jobs. A compressed object is encoded, compacted, replicated, cached, backed up and read many times; counting only the surviving object understates machine work, while counting every operation as a fresh saving overstates money. The report therefore reports the following separately, and every financial figure is tied to exactly one of them.

| Quantity | What it counts | Where it is used |
|---|---|---|
| **Capability** (machines) | Devices whose OS, browser or firmware contains a decoder. An upper bound on exposure, not a measure of use. | Figure 2, Table A1. |
| **People reached** | People whose data passes through the codec, directly or via a backend. | Figure 3, Table A2. |
| **Economically countable output** | Compressed bytes written or transmitted once and charged once. This is the volume series of Sections 4.3–4.5 and the only quantity that feeds the money model. | Figures 4–5, Tables A3–A7. |
| **Encoder-output operations** | The same logical object re-encoded by LSM compaction, replication, cache fills and re-uploads. Machine work, not new stored bytes. | Figure 9A. |
| **Decoder / read operations** | Every read of a compressed byte. The largest number in the report and the one most often quoted loosely; it is never converted into a saving. | Figure 9A. |
| **Charged copies** (replication) | How many stored copies an avoided byte is avoided in: 1 for the auditable floor, 2–3 where three-way replication, erasure coding, backups and cross-region copies apply. | Figure 9B, Appendix B. |
| **Avoided bytes and dollars** | Counterfactual minus actual, on output only, with whole-codec and strict ANS-stage attributions kept apart throughout (Section 2.5). | Figures 6–8. |

Two consequences are worth stating plainly. Kernel or library *inclusion* raises capability, not volume: material bytes come from enabled consumers — Btrfs/F2FS/EROFS/SquashFS/zram, initramfs and package choices, applications — not from the presence of `lib/zstd` in a source tree. And a codec that is available in a configuration parameter is not a codec that is enabled by default; Section 3 labels the two differently, and version 2.2 downgrades four labels that failed that test.

### 2.4 Metrics

| Metric | Definition used | Main evidence |
|---|---|---|
| Machines | Active physical devices (phones, tablets, PCs, servers, consoles, GPUs) whose operating system, browser or firmware contains a decoder for the software, at year end. A device carrying several ANS decoders is counted once per software; the per-software lines overlap and must not be summed. | Apple active-device disclosures; console shipments; StatCounter shares; kernel/browser release dates; caniuse reach. |
| Users | People whose data is compressed or decompressed by the software, directly or through a backend they use (e.g. Meta's storage since 2016). For research software (CRAM, nvCOMP) the count is practitioners. | Meta daily active people (3.58 B, Q4 2025), ITU (6.0 B online), browser shares, Apple device-per-user ratios, bioconda downloads, UK Biobank researcher counts. |
| Data volume | Compressed bytes per year *written to storage or transmitted over a network* in an ANS-coded format (the independent Sol/Kimi study uses logical input bytes instead, typically 2–4× larger). Decode volume is larger and is not the metric. In-memory uses (zram, zswap, RPC payloads) are excluded. | Steam's 100 EB/yr; Web Almanac shares × estimated web text traffic; archive sizes (ENA 62.8 PB, UK Biobank, All of Us, ALICE 423 PB); Apple update cadence × installed base; console download estimates. |
| Resident footprint | Average stored bytes attributable to a year's writes that are still resident: durable tables are retained 80 %/yr, archives 95–97 %/yr, logs/streams/caches 10 %/yr, training checkpoints 5 %/yr. Storage cost is charged on this footprint, not on annual writes. | Retention policies of the platforms concerned; NVIDIA checkpoint cadence; archive mandates. |
| Savings (bytes) | Counterfactual size − actual size, for the data volume above. Two attributions: whole-codec and ANS-stage (Section 2.5). | Published benchmarks: Cloudflare, Meta, Kafka, MongoDB/Percona, Elasticsearch, htslib CRAM, Cloudinary/WebKit JPEG XL, Xbox Wire, NVIDIA. |
| Savings (USD) | Transferred bytes saved × provider-side transfer cost ($2–8/TB) + resident footprint saved × storage cost ($30–100/TB-year). End-user bandwidth, CPU time and energy are not monetised. | Public cloud list prices (S3 standard ≈ $276/TB-yr; archive tiers $12–48/TB-yr), NIH SRA cloud costs (~$180/TB-yr), CDN wholesale pricing. |

### 2.5 Counterfactuals and attribution

- **Whole-codec**: the ANS-based format vs the format it demonstrably displaced (gzip/Snappy/LZ4 → zstd; BAM → CRAM; JPEG → JPEG XL; zlib → BCPack; LZMA → zstd at Steam, where the byte saving is ≈0 and the benefit is speed). This overstates ANS's own contribution because LZ77 improvements, dictionaries, column separation (CRAM) and quantisation (Draco) are included.
- **ANS-stage**: the ratio gain of the ANS coder over the Huffman coder the same codec would otherwise have used — the counterfactual the industry chose when it did not use ANS (Brotli, Oodle Kraken, Apple LZMESH, ZipNN, DFloat11, Unweight). Collet's FSE benchmarks show the gap is negligible on near-uniform 8-bit alphabets (≤1 %) and large on skewed small alphabets (Proba80: +38 %); zstd routes exactly the skewed small-alphabet streams to FSE. The model uses 2–3 % of output for zstd/LZFSE, 15 % for CRAM and ALICE, 5 % for JPEG XL and Draco, 8 % for BCPack and 20 % for GPU float codecs. This understates ANS's contribution because it ignores that speed parity with Huffman is what made adoption possible.

### 2.6 Scenarios, sources and confidence

Two scenarios are computed from the same model. The **central** scenario uses the assumptions in Appendix B. The **high** scenario reproduces version 1.1 (Steam 70 % zstd, larger data-platform and Apple volumes, 85 % retention for all stored data, faster JPEG XL and GPU ramps) and is shown as a dashed line in the figures. Component-level uncertainty ranges (×0.2–×4) are combined additively into the bands. Six parallel literature searches (~200 web searches, ~600 page fetches), a comparison with an independent study and an external review (Section 7) underlie the evidence. The encode.su thread itself could not be fetched (the site blocks automated clients), so Section 6 lists candidate additions without knowing the thread's exact current contents. Numbers for 2026 are full-year estimates from data available to 2 September 2026; 2027–2030 are extrapolations and are shaded in every figure. The model (`model.py`), yearly CSV tables and figures are delivered alongside this document.

## 3. Adoption evidence, application by application

Figure 1 places the milestones on one timeline; the subsections give the evidence behind each dot. Full URLs are in Section 9, grouped by subsection.

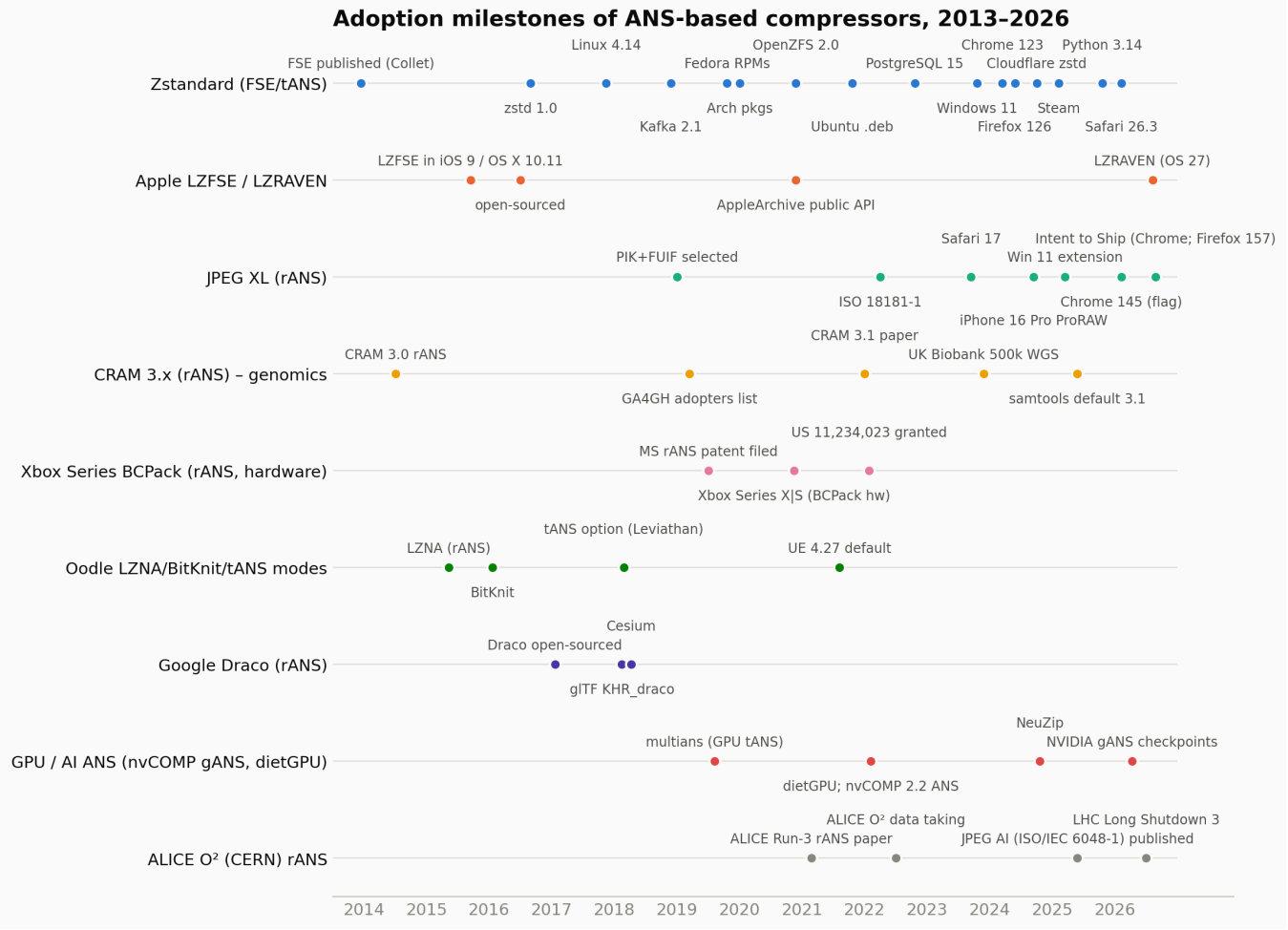



**Figure 1.** Adoption milestones of ANS-based compressors, 2013–2026. Each row is one software family; dots mark dated releases, standards or platform decisions documented in Section 3.

### 3.1 Zstandard (Facebook/Meta; FSE = tANS)

Yann Collet published FSE in December 2013 as a multiplication-free tANS variant benchmarked against zlib's Huffman coder, folded it into zhuff, and then into Zstandard, whose 1.0 release (31 Aug 2016) claimed 10–15 % smaller output than zlib at equal speed and 3–5× faster compression at equal ratio. RFC 8878 fixes the format: FSE codes the literal-length, match-length and offset code streams and the Huffman weight table; literals use Huffman. Collet's stated reason for keeping Huffman for literals is that byte literals are near-uniform (~0.4 % per symbol) so fractional bits gain little, whereas the sequence codes are small, skewed alphabets where FSE's advantage is large.

Table 3.1. Zstandard adoption milestones (all confirmed from release notes or vendor documentation unless marked).

| Date | Milestone | Source |
|---|---|---|
| 2016-09 / 2017-01 | RocksDB 4.12 kZSTD; Amazon Redshift ZSTD column encoding | RocksDB HISTORY; AWS What's New |
| 2017-10 | Apache Parquet adds ZSTD codec (format 2.4) | parquet-format PR #70 |
| 2017-11 | Linux 4.14: zstd library, Btrfs and SquashFS zstd ("already in production at Facebook") | kernelnewbies; Phoronix |
| 2018-03 → 2021-10 | Ubuntu 18.04 dpkg/apt zstd support; Ubuntu 21.10 .deb packages and initramfs zstd by default | ubuntu-devel; Phoronix; Launchpad #1931725 |
| 2018-11 | Apache Kafka 2.1 (KIP-110): zstd 4.28× vs Snappy 2.5× on Shopify production data | Kafka KIP-110 |
| 2018-12 | Meta: "managed compression" 50 % better ratio than replaced methods; caches hold up to 40 % more data; backups −16/−27 % | engineering.fb.com |
| 2019-08 / 2019-10 | MongoDB 4.2 zstd (WiredTiger + network); MySQL 8.0.18 zstd connection compression | Percona; MySQL release notes |
| 2019-10 / 2020-01 | Fedora 31 switches all RPMs xz → zstd (install 8 s → 2 s); Arch Linux repositories xz → zstd (+0.8 % size, ~13× faster decompression) | Fedora Changes; archlinux.org |
| 2020-10 / 2020-11 | Linux 5.9 zstd kernel image & initramfs (Facebook: 12 s → 3 s); OpenZFS 2.0 zstd | kernelnewbies; LWN; OpenZFS release |
| 2021-04 / 2021-08 | Fedora 34 Btrfs `compress=zstd:1` by default; Blender .blend gzip → zstd | Fedora Changes; bf-committers |
| 2021-05 / 2023-02 | containerd 1.5 and Docker 23 zstd OCI layers | containerd, Docker release notes |
| 2022-10 / 2023-09 | PostgreSQL 15 (WAL, pg_basebackup) and 16 (pg_dump) zstd | PostgreSQL release notes |

| Date | Milestone | Source |
|---|---|---|
| 2023-09 / 2024-11 / 2024-12 | Iceberg 1.4 default Parquet codec zstd (partially effective, see §8); Databricks Runtime 16 default zstd for Delta; Trino Delta writer default zstd | iceberg PR #8593; Databricks docs; trino PR |
| 2023-10 | Windows 11 File Explorer (libarchive) opens .tar.zst/.tzst | BleepingComputer (KB5031455) |
| 2024-03 / 2024-05 | Chrome 123 and Firefox 126 ship `Content-Encoding: zstd`; Meta reports positive user metrics; Firefox notes zstd "heavily used on sites such as Facebook" | Chrome releases; blink-dev; Firefox notes |
| 2024-09 | Cloudflare: zstd for all plans; measured 2.86:1 vs gzip 2.56:1 (11.3 % smaller at comparable speed); Free plan compresses with zstd by default, while Pro and Business default to Brotli and Enterprise to gzip | Cloudflare blog; Cloudflare docs |
| 2024-09 | OpenSearch 2.9+ (zstd and zstd_no_dict codecs, 26–32 % smaller indices than the LZ4 default; Amazon OpenSearch Serverless offers the same, opt-in). Elasticsearch 8.16/9.0: zstd for `best_compression` (≈12 % less storage, ≈14 % more indexing throughput vs DEFLATE) | elasticsearch PR #112665 |
| 2025-02 → 2025-05 | Steam client gains zstd depot chunks ("VSZa" vs LZMA "VZa"; first live chunk observed 9 May 2025) — new and updated chunks only, existing depots are not re-encoded; Steam delivered 80 EB in 2024 and 100 EB in 2025 (274 PB/day) | SteamKit #1503/#1504; PC Gamer / Valve Year in Review |
| 2025-10 | Python 3.14 adds `compression.zstd` (PEP 784); Meta's OpenZL paper: zstd was "the vast majority of compression use at Meta" | Python docs; arXiv 2510.03203 |
| 2025 / 2026 | SQL Server 2025 ZSTD backup compression; .NET 11 `System.IO.Compression` Zstandard; Android 17 zram recompression names zstd as the secondary algorithm, but `mmd.zram.recompression.enabled` defaults to `false` and first-pass reclaim uses lz4, so this is an opt-in capability rather than a default | Microsoft Learn; dotnet/runtime #112075; source.android.com |
| 2026-01 | Web Almanac 2025: zstd = 12 % of CDN-served mobile responses (Cloudflare 15 %, Google CDN 10 %), up from 2.72 % in the 2024 edition | almanac.httparchive.org |
| 2026-02 | Safari 26.3 ships zstd Content-Encoding (decoder in the OS networking stack of iOS/macOS 26.3) | WebKit blog |
| 2026-09 | Cloudflare begins transcoding uncompressed cached text to zstd level 3 ("petabytes of effective cache capacity") | Cloudflare blog |

**Reach indicators.** caniuse puts browser support for zstd content-encoding at 84.4 % of global page views (Jul 2026). The PyPI `zstandard` package is downloaded ~190 million times a month (2.5 billion cumulative); npm's `@mongodb-js/zstd` 7.3 million and `fzstd` 3.2 million a month. facebook/zstd has 27.7k GitHub stars. No vendor publishes bytes-per-day compressed with zstd; the volume model in Section 4 therefore triangulates from Steam's disclosure, Web Almanac shares and data-platform defaults. estimate

**Further zstd carriers identified in review** (all confirmed from documentation; volumes unmeasured): CERN's ROOT **RNTuple** uses zstd level 5 by default (format v1 in ROOT 6.34, production API targeted for 6.36; ATLAS has published plans to adopt RNTuple as its main event-data storage) Default-on ; Oxford Nanopore's **VBZ** codec (StreamVByte + zstd) compresses raw nanopore signal in FAST5/POD5, >30 % smaller and 5–10× faster than gzip Default-on ; **Zarr** v3 uses Zstd as its default codec, so cloud-native climate, microscopy and astronomy arrays default to zstd Default-on ; **bazel-remote** stores build-cache blobs zstd-compressed by default and **GitHub Actions** caches are zstd tarballs on hosted runners Default-on ; the **OCI image specification** defines a `tar+zstd` layer media type Optional capability ; **restic** repository format v2 zstd-compresses data and tree blobs Optional capability ; **GDAL** GeoTIFF supports ZSTD and LERC_ZSTD (default uncompressed) Optional capability ; **Apache Arrow IPC** supports optional LZ4 or zstd buffer compression (default uncompressed) Optional capability . These enter the model as a small "scientific/HPC" component and within the transient data-platform component; none has a published byte count.

**Chinese data platforms** (added in v2.1 after a follow-up review): Alibaba Cloud's PolarDB for MySQL uses zstd as the default codec of its In-Memory Column Index (`imci_default_codec = ZSTD`; the alternative is LZ4) Default-on , and OceanBase (Ant Group) offers zlib, Snappy, zstd and LZ4, with zstd plus encoding as the configuration its own compression write-up calls default for tables (ratio ≈4.6; its log-archive option lists zstd_1.3.8 beside an LZ4 default) Default-on likely ; ByteDance's Volcano Engine LAS data lake on the Lance format is reported to use zstd for point-cloud data (≈70 % reduction; LanceDB case study, text not retrievable here) Optional capability likely . All three are carriers of FSE-coded bytes at hyperscale, but none publishes volumes; they are covered by the global data-platform components in Section 4.3 and are not added on top (Section 7.4).

**Carriers added in version 2.2** (all confirmed from primary documentation). **Discord** replaced zlib with streaming Zstandard on its gateway in April 2024: the compression ratio rose from 6 to almost 10, the median payload fell from 270 to 166 bytes and the gateway's total WebSocket traffic dropped by nearly 40 % (with Passive Sessions V2), for a service whose concurrent connections number in the millions — every message delivered to every connected client is now FSE-coded Default-on . **Fedora** has mounted its Btrfs root with `compress=zstd:1` by default since Fedora 34 (2021); Fedora's own analysis measured ~40 % of disk saved for ~9 % more system time, so on those installations every file written passes through FSE Default-on . **Box** enabled ZSTD for customer file transfers for all customers, citing browser support after the 2023 RFC ratification Default-on . **Cloudflare** prototyped cache transcoding (1 September 2026): uncompressed text objects — 22.3 % of cached bytes — recompressed with zstd level 3 shrank by 2.834× on average across a million-request test on ten cache servers, implying petabytes of effective capacity, though this is an experiment and not a fleet-wide default Research . These four were missing from version 2.1 and are part of the answer to Collet's observation (Section 7.5).

### 3.2 Apple: LZFSE (2015) and LZRAVEN (2026)

LZFSE ("Lempel-Ziv Finite State Entropy") was announced at WWDC 2015 and shipped in iOS 9 and OS X 10.11 as the recommended algorithm of the Compression framework; Apple's documentation describes it as LZ77 with "a variant of tabled Asymmetric Numeral Systems". It was open-sourced in 2016 (BSD-3). Confirmed uses: the Compression framework default; Apple Archive (.aar, public API since macOS 11, LZFSE recommended, no zstd option); APFS/HFS+ transparent file compression types 11 and 12; the iOS kernel cache (LZFSE "bvx2" compression on recent releases); Apple Disk Images (ULFO). OTA update payloads have used Apple Archive containers since iOS 14 (inner algorithm not independently dumped — LZFSE is the container's default) likely . Xcode .xip archives use LZMA, not LZFSE.

In OS 27 (2026) Apple's Compression documentation adds two algorithms: **LZMESH**, "LZ77 with length-limited Huffman coding … designed to replace both COMPRESSION_LZFSE and COMPRESSION_ZLIB", and **LZRAVEN**, "designed to replace COMPRESSION_LZMA" with "modern entropy coding, adaptive context-sensitive modeling" and vectorised multi-symbol decoding, three times faster than LZMA on Apple silicon. A reverse-engineered, conformance-tested implementation (liblzraven) describes LZRAVEN as an 8-way interleaved adaptive rANS coder with AV1/Daala-style 16-symbol adaptive CDFs, and OS 27 OTA payloads switched from pbzx (LZMA) to a pbzm (LZRAVEN) container. Apple does not name the coder, so the rANS attribution is likely . The net effect is that Apple moved its *fast* tier from tANS to Huffman while moving its *high-ratio* tier from LZMA's range coder to rANS; both tiers ship to the entire installed base, which Apple reported as more than 1.0 B active devices (Jan 2016), 1.4 B (2019), 2.0 B (2023), 2.2 B (2024), 2.35 B (2025) and 2.5 B (Jan 2026).

### 3.3 JPEG XL (ISO/IEC 18181; rANS)

JPEG XL's entropy coding is LZ77-enabled and can use either ANS or prefix (Huffman) codes; libjxl uses prefix codes only at its lowest lossless effort (e1) and ANS everywhere else — lossy encodes use ANS at every effort — with a default effort of 7, so essentially all JPEG XL files in the wild are rANS-coded. Its lineage — Brunsli (2015–2019), PIK (2017–2019), WebP 2 — is ANS throughout.

Table 3.2. JPEG XL milestones.

| Date | Milestone | Source |
|---|---|---|
| 2021-10 / 2022-03 | ISO/IEC 18181-2 (file format) and 18181-1 (codestream) published | ISO; jpegxl.info |
| 2022-10 → 2023-02 | Chromium deprecates and removes its experimental JXL support ("not enough interest from the entire ecosystem") | The Register; Wikipedia |
| 2023-06 / 2023-09 | Apple ships JPEG XL in Safari 17 on iOS 17, iPadOS 17, watchOS 10 and macOS (system-wide ImageIO decoding on iOS 17/macOS 14 per 9to5Mac/Wikipedia) | WebKit blog |
| 2023-06 / 2024-01 | Adobe DNG 1.7 adds JPEG XL compression; Samsung Galaxy S24 Expert RAW writes DNG 1.7/JXL | Wikipedia DNG; Cloudinary |
| 2024-04/05 | DICOM Supplement 232: three JPEG XL transfer syntaxes for medical imaging (DICOM 2024d) | NEMA; dicomstandard.org |
| 2024-09 | iPhone 16 Pro captures ProRAW as JPEG-XL lossy/lossless inside DNG (48 MP lossless ≈46 MB vs ≈75 MB previously) | 9to5Mac; PetaPixel |
| 2025-03 | Microsoft "JPEG XL Image Extension" for Windows 11 24H2; Photoshop 26.8, GIMP 3.0, Paint.NET, GNOME 45, KDE support | Microsoft Store; Wikipedia |
| 2025-11 | PDF Association names JPEG XL the "preferred solution" for HDR images in PDF; Chromium (Rick Byers) invites a memory-safe decoder contribution | The Register; issues.chromium.org |
| 2025-12 → 2026-02 | jxl-rs (Rust) lands in Chromium; Chrome 145 stable ships JXL behind `enable-jxl-image-format` | Chromium commits; Phoronix |
| 2026-06 | Firefox 152: JXL available as a Firefox Labs toggle (default-on in Nightly); Google OSS blog "Journey to JPEG XL"; first commercial JXL encoder IP core (Shikino/CAST) | Mozilla dev-platform; Google OSS blog |
| 2026-08-24 | Mozilla Intent to Ship: default-on in Firefox 157 (scheduled 29 September 2026). Chromium Intent to Ship (`image/jxl`, all Blink platforms, "will ship enabled for all users", using the memory-safe jxl-rs decoder) with three LGTMs the same day | dev-platform; blink-dev |
| 2026-09 (verified 3 Sep) | **ChromeStatus now carries a shipping milestone: 155 on desktop, Android, WebView and iOS.** Chromium Dash puts M155 stable on **6 October 2026** (beta 16 September). With Firefox 157 a week earlier and Safari since 17, JPEG XL becomes decodable in every major browser inside one month | chromestatus API; chromiumdash |

**Version 2.2 correction.** Version 2.1 modelled Chrome as an assumed mid-2027 event because the Intent to Ship named no milestone. That is now out of date: the feature entry carries milestone 155 on all four Blink platforms, and M155 reaches stable on 6 October 2026, so the model brings browser reach and the byte ramp forward by roughly a year (decoders ≈5.2B devices at end-2026; 0.1 EB of JXL output in 2026 rising to 18 EB in 2030). Capability is still only an upper bound on content: encoders, CDN defaults and authoring tools lag browsers, and a browser can still delay or roll back a launch. Operator telemetry supplied by Jon Sneyers (Cloudinary, JPEG XL Discord, September 2026) gives the first request-side anchor: Cloudinary alone is already serving 0.5–1 billion JXL images a day, with 3–4 billion a day forecast by year-end. At 183–365 billion responses a year and 80–250 kB per delivered image that is ≈0.015–0.091 EB/yr of delivered JXL bytes — which is why the request count, not the byte count, is the headline number for 2026, and why responses must not be read as unique encodes. On 2 September 2026, before the Chrome and Firefox releases, caniuse reported 15.0 % global "partial" support (Safari) and 0 % default-on Chromium; StatCounter gives Chrome 69.4 %, Safari 15.8 %, Firefox 3.0 % of page views. HTTP Archive has never measured JXL's web share because its Chromium crawler did not accept the format. Camera-side, ~2 trillion photos are taken a year (94 % on smartphones); only the iPhone 16 Pro ProRAW path and Samsung Expert RAW write JXL today. Size benefits: ≈20 % for lossless recompression of existing JPEGs (Apple, Brunsli), ≈50 % for lossy at equal quality (Sneyers et al. 2025), 20 % smaller than libavif at high quality (Cloudinary, 2024; note Cloudinary co-authored the format).

### 3.4 CRAM (genomics; rANS since 2014)

CRAM 3.0 (2014) introduced James Bonfield's rANS 4×8 codec (order-0 and order-1, four interleaved states) — the first production use of ANS in a widely used data format. CRAM 3.1 (draft 2019; Bonfield, *Bioinformatics* 2022) added rANS Nx16 with 32-way interleaving for SSE4.1/AVX2/AVX-512/NEON, RLE and bit-packing transforms, an adaptive arithmetic coder, and specialised quality and read-name codecs; htscodecs 1.0 (Feb 2021) packages these as a standalone library, and HTSlib/SAMtools 1.22 (30 May 2025) made CRAM 3.1 the default output. Adopters named by GA4GH (2019) and confirmed since: EMBL-EBI ENA and EGA ("CRAM:BAM ≈ 2:1, more than a million CRAM files" already in 2018), Wellcome Sanger (unaligned CRAM in place of FASTQ), Genomics England, Broad Institute (archival), H3Africa, Illumina (DRAGEN `--output-format CRAM`), Sweden's NGI, Australian Genomics, UK Biobank (500k whole genomes released as CRAM on the DNAnexus RAP, Nov 2023), All of Us (245k genomes submitted as CRAM), 1000 Genomes/IGSR, JBrowse 2 (cram-js), htsjdk/GATK/Picard, Rust noodles, Genozip; NVIDIA's Parabricks GPU genomics suite bundles htscodecs (its third-party notice credits the rANS code derived from Giesen's ryg_rans), a commercial distribution path for CRAM's rANS.

Table 3.3. CRAM compression benchmarks (htslib.org, updated April 2025; 10 M alignments each).

| Dataset | BAM | CRAM 3.0 | CRAM 3.1 | 3.1 vs BAM | 3.1 vs 3.0 |
|---|---|---|---|---|---|
| Illumina NovaSeq | 515 MB | 207 MB | 176 MB (archive 158 MB) | −65.8 % | −15.0 % |
| Illumina HiSeq 2500 | 1,546 MB | 880 MB | 852 MB (archive 775 MB) | −44.9 % | −3.2 % (archive −10.7 %) |
| PacBio Revio | 4,832 MB | — | 1,288 MB | −73.3 % | — |
| Oxford Nanopore | 2,110 MB | — | 1,327 MB | −37.1 % | — |

Independent confirmation: the Emirati Genome Program (2022) converted ~10 PB of BAM to CRAM 3.1 with a 58 % footprint reduction and projected US$1.2 M/yr savings; Sanger states 40–50 % over BAM. Archive scale: ENA held 62.8 PB (2024 paper), growing 30–40 %/yr and projecting 150 PB within five years; UK Biobank's 200k genomes were "five petabytes" (2021) and its whole dataset ~30 PB (2023); All of Us expected 4.75 PB in 2023; Genomics England stored 21 PB (2020); NCBI SRA exceeded 36 PB in 2020 but stores its own normalised format rather than CRAM. Community reach: bioconda downloads of samtools 9.4 M, htslib 8.8 M, pysam 11 M (all-time, Sept 2026); ~30,000 approved UK Biobank researchers.

### 3.5 Microsoft: Xbox Series BCPack, DirectStorage and the rANS patent

> **The "DirectX" question resolved.** Microsoft's public GDK documentation for DirectStorage on Xbox Series X|S states that the console's hardware decompression block supports three modes — DEFLATE (RFC 1950), **BCPack** and Swizzle — and that "BCPack is a custom entropy coder designed specifically for BCn data ... color endpoints are separated from palette indices ... and compressed using a rANS algorithm." Xbox Wire (2020) describes the block as delivering 4.8 GB/s effective from 2.4 GB/s raw, work Microsoft equated to more than four Zen 2 CPU cores. It is the only publicly documented rANS decoder in consumer silicon that this research found. confirmed
>
> The PC side was different until this year: DirectStorage 1.1 (Nov 2022) GPU decompression uses **GDeflate**, a bit-swizzled Deflate (LZ77 + Huffman) designed by NVIDIA (IETF draft-uralsky-gdeflate) — not ANS. **That changed on 11 March 2026**, when Microsoft published the **DirectStorage 1.4** public preview, which "adds Zstd to our multi-tier decompression framework with support for CPU and GPU decompression" and ships the Game Asset Conditioning Library (GACL), lossless and lossy pre-transforms that Microsoft says improve zstd ratios by up to 50 % and that DirectStorage reverses at load time; AMD, Intel, NVIDIA and Qualcomm are described as adding hardware-specific zstd decompression optimisations later in 2026. Because zstd's sequence streams are FSE-coded, this puts a tANS decoder into the PC asset-streaming path for the first time — on the GPU as well as the CPU. It is a preview, it requires explicit authoring, and no shipped title is known to use it yet, so version 2.2 records it as an Optional capability with no volume and treats it as one of the larger 2030 swing factors (Section 5.2). confirmed Windows otherwise carries ANS indirectly, via zstd in File Explorer (2023), SQL Server 2025 backups and .NET 11.

Microsoft's US patent 11,234,023 ("Features of range asymmetric number system encoding and decoding", filed 28 Jun 2019, granted 25 Jan 2022 after two USPTO rejections and an After-Final re-filing) claims a two-phase hardware rANS decoder; a continuation (US 2022/0109891) targets "special-purpose codec hardware ... RANS decoder modules" for "video, images, audio, texture for graphics", and the family is granted in Europe (EP3991303B1), China (CN114073009B) and Korea (KR102922298B1). The timing and the "texture for graphics" language are consistent with BCPack, but no Microsoft document links the two explicitly likely. Microsoft discloses no Xbox Series unit counts; third-party estimates of roughly 30–36 million consoles by 2025–26 are used in Section 4 estimate.

### 3.6 Oodle (RAD Game Tools → Epic): a correction

Oodle is often listed wholesale as an ANS deployment. The record from its authors is narrower. **LZNA** (May 2015) "uses rANS instead of arithmetic coding" with adaptive nibble models; **BitKnit** (2015/16) is a two-state interleaved rANS coder (Giesen, "rANS in practice"). **Kraken** (April 2016), Mermaid and Selkie shipped with Huffman and raw streams only; Giesen: "originally we meant to use tANS in the original Kraken release back in 2016, but the shipping version of Kraken doesn't contain it; it was later added as a new entropy coding variant in 2018 along with Leviathan, but is rarely used." Selkie never entropy-codes. Consequently the PlayStation 5's hardware Kraken decoder (Cerny, March 2020: 5.5 GB/s in, 8–9 GB/s typical, 22 GB/s peak; PS5 sell-in 95.3 M units by June 2026) is primarily a Huffman/LZ decoder; an encode.su discussion of the PS5 announcement (thread 3364, cited by the independent Sol/Kimi report) states that the hardware implements the Oodle 2.2-era Kraken feature set, which predates the tANS option — consistent with a 2016–17 silicon design freeze, though we could not read the thread directly likely. Oodle Data has been the default pak compression in Unreal Engine since 4.27 (Aug 2021; Kraken is "the usual default" in UE5), and RAD claimed "close to 25,000 games" at the 2021 acquisition, so Oodle's *installed* footprint is hundreds of millions of machines — but only a small fraction of Oodle-compressed bytes go through an ANS path. Section 4 models that fraction at ~3 % estimate.

### 3.7 Google Draco (3D geometry; rANS)

Draco (open-sourced January 2017) codes quantised attributes and Edgebreaker connectivity symbols with rANS (its `ans.h` cites arXiv 1311.2540 and derives from libvpx's experimental ANS). Khronos ratified `KHR_draco_mesh_compression` for glTF 2.0 in February 2018 ("up to 12×" on sample models), Cesium adopted it for 3D Tiles (2018–19), and it is bundled in three.js (`DRACOLoader`, used by `GLTFLoader`), Babylon.js, glTF-Transform, Unity ("Draco for Unity" 5.4, glTFast), Blender's glTF exporter and Pixar USD plugins. Reach indicators: `draco3d` 18.2 M npm downloads per month, `draco3dgltf` 0.6 M, google/draco 7.4k stars. Size gains are 75–95 % versus raw glTF geometry, but a 2021 analysis found quantisation + ZIP within 15 % of Draco, indicating that most of the gain is quantisation and connectivity coding rather than the entropy coder. Byte volumes are tiny (Section 4); reach is large. Google Maps' Photorealistic 3D Tiles, Sketchfab and e-commerce viewers are widely reported but not verified users likely.

### 3.8 GPU and AI/LLM: production and emerging uses

**nvCOMP.** NVIDIA added GPU-accelerated ANS in nvCOMP 2.2.0 (7 Feb 2022), improved throughput in 2.3/2.4, added a float16 mode, renamed the codec "gANS" ("entropy encoder optimized for ML/floating point data"), doubled throughput in 5.2 and added a float8 (E4M3) mode with sub-chunking in 5.3 (2026); nvCOMPDx exposes ANS decode inside user kernels. In April 2026 NVIDIA's developer blog ("Cut checkpoint costs with ~30 lines of Python and nvCOMP") recommends gANS as the default for GPU-resident checkpoint (de)compression: 1.18× on a full checkpoint vs 1.14× for zstd; 1.46–1.48× on BF16 weights vs 1.27–1.28× for zstd; ~530 GB/s vs ~16 GB/s, on H200 and Blackwell. The Blackwell hardware Decompression Engine accelerates Snappy, LZ4 and Deflate — not ANS. RAPIDS cuDF uses nvCOMP for Snappy/zstd only. PyPI `nvidia-nvcomp-cu12`: ~28k downloads/month.

**Meta dietGPU** (Jeff Johnson, early 2022): byte-oriented rANS and a float codec that compresses exponents (bf16 ≈0.67×, fp16 ≈0.85×) at 250–600 GB/s on A100, aimed at collective communications in distributed training; described as an "early alpha", never documented in production, and archived (read-only) on 14 June 2026. A HIP port (hipANS) exists.

**Research and adjacent systems.** multians (2019) was the first massively parallel GPU tANS decoder; Recoil (ICPP 2023) decodes rANS from arbitrary split points at 11+ GB/s on CPU and 90+ GB/s on GPU. NeuZip (Hao et al., 2024) losslessly compresses float exponents with ANS (via nvCOMP, per the DFloat11 authors) during training. A June 2026 preprint ("Approaching Shannon Bound with Lossless LLM Weight Compression", NUS/HUST/ETH) extends dietGPU's rANS kernels to tile-granular decoding inside tensor-core GEMMs; an open KVTC implementation offers rANS as one of its KV-cache back-ends; torch_ans provides a CUDA rANS extension for PyTorch. In HPC, MANS (Chinese Academy of Sciences, SC25) ports ANS to multi-byte integer data on CPUs, NVIDIA and AMD GPUs and ships an HDF5 filter (H5Z-MANS). AMD's hipCOMP-core, by contrast, lists ANS as "not supported (NVIDIA proprietary)". In learned image compression, InterDigital's CompressAI (rANS via ryg_rans; 11k PyPI downloads/month), Bamler's constriction (ANS + range coding; 4k/month), and the bits-back line (BB-ANS, Bit-Swap, HiLLoC, craystack) are standard tooling. On the data side, several LLM pre-training corpora are distributed as zstd (The Pile `.jsonl.zst`, SlimPajama), though RedPajama-v2 and Common Crawl use gzip and FineWeb's Parquet codec is undocumented.

#### Emerging AI applications, 2025–26 Research unless stated

The review of version 1.1 asked for these to be tracked as three independent curves — storage (checkpoints), communication (collectives) and token/KV/prompt stores — because their economics differ. The evidence today:

| Application | What the source reports | Evidence class |
|---|---|---|
| GPU checkpoint compression (nvCOMP gANS) | NVIDIA, 9 Apr 2026: a 70 B AdamW checkpoint is 782 GB; every 30 minutes → 1.13 PB/month per job; gANS 1.18× on the full checkpoint (zstd 1.14×), 1.46–1.48× on BF16 weights (zstd 1.27–1.28×) at ~530 GB/s vs ~16 GB/s; checkpoint wait 156 s → 133 s; modelled savings "up to $40,000 every month" for a 405 B model on 128 Blackwell GPUs. Storage and retention assumptions are NVIDIA's. | Optional capability ; fleet volume unknown |
| Token-native storage for agents and RAG (arXiv 2608.02376, Aug 2026) | Store text as the model's BPE token IDs rather than UTF-8: uint16 packing alone is 2.25× smaller than UTF-8 on English; a static unigram ANS coder (via constriction) reaches 3.30×, versus 1.95× for gzip-9 and 1.97× for zstd-19 on the same text. Because BPE numbers tokens by merge order, re-ranking IDs by frequency lets a plain streamvbyte codec recover most of the ANS ratio at ~7× faster decode; the paper asks AI labs to publish frequency-ranked vocabularies. Relevant to persistent agent memory, retrieval stores and prompt caches. | Research |
| UCCL-Zip — compression inside GPU collectives (arXiv 2604.17172, 2026) | GPU rANS on the exponent field with locally built frequency tables inside NCCL-style kernels; up to 47.5 % faster RL weight synchronisation and single-digit-percent end-to-end inference latency gains. | Research |
| ANS-GEMM — inference on ANS-coded weights (arXiv 2606.15789; ISCA 2026) | Weights kept as tile-local rANS streams and decoded inside the GEMM (dietGPU-derived kernels); integrated with SGLang; larger feasible batches (Qwen-14B 47 → 75, Mixtral-176B 20 → 95) and 1.2–1.6× throughput where memory bandwidth binds. FP8/FP4 weights leave far less lossless headroom than BF16. | Research |
| Edge–cloud split inference (arXiv 2511.11664, 2025) | Sub-millisecond GPU rANS coding of intermediate DNN features to cut transmission latency and bandwidth in split execution. | Research |
| RAS — hardware rANS accelerator (arXiv 2511.04684, Oct 2025) | Bit-exact rANS accelerator for neural lossless compression; RTL simulation reports 121× encode and 71× decode speed-ups over a Python baseline; no silicon or area figures yet. | Research |
| GS-NFS — dynamic Gaussian-splat streaming (arXiv 2606.05650, USC, 2026) | Bandwidth-adaptive streaming of dynamic Gaussian splats and point clouds; positions are compressed with the ANS coder from NVIDIA's nvCOMP library, attributes with RLGR. Extends ANS into neural rendering and telepresence. | Research |
| NeuZip; KVTC; MANS/H5Z-MANS; torch_ans | Lossless exponent compression during training; rANS as a KV-cache back-end; ANS for multi-byte scientific integers with an HDF5 filter; a PyTorch CUDA rANS extension. | Research |

For the model, only checkpoint compression carries a volume (0.1 EB/yr in 2026, illustrative; range 0.01–0.4), because it is the only path with a vendor recommendation and a quantified case; communication and token-store bytes are set to zero in 2026 and discussed as 2030 options in Section 5. FP8 and FP4 weight formats leave much less lossless headroom than BF16, so growth in model traffic does not translate proportionally into ANS savings.

**What is not ANS in this space** matters for the estimate: ZipNN (IBM, 2024) "uses only ... Huffman codes" and reports rejecting FSE because it was 0–2 % better at "at times over 2×" the cost; DFloat11 (2025) and Huff-LLM are Huffman; ZipLLM uses XOR-delta coding; Cloudflare's Unweight (Apr 2026) Huffman-codes exponent bytes for up to 22 % smaller model files; DeepMind's "Language Modeling Is Compression" and Bellard's ts_zip use arithmetic coding; Hugging Face's Xet storage uses LZ4/BG4; Apple Core ML compression is lossy quantisation. GPU/AI is thus a genuine but small ANS niche whose growth hinges on whether NVIDIA's gANS recommendation is taken up for checkpointing and on the 2026 weight-compression results reaching inference stacks.

### 3.9 ALICE O² at CERN: rANS in a particle-physics data pipeline

The ALICE experiment's Run-3 online–offline (O²) system uses a custom rANS entropy coder as the final component of its lossless data-reduction chain. The 2021 design paper ("Fast Entropy Coding for ALICE Run 3", arXiv 2102.09649) sets the target of compressing the post-reduction stream from about 300 GB/s to about 90 GB/s, codes detector arrays with symbols of up to 25 bits using static, per-array frequency tables (with an escape mechanism for unseen values), and motivates rANS by the throughput needed on the Event Processing Node farm — 350 servers, each with eight AMD GPUs (CERN Courier, 2023). The compressed time frames written to storage amount to 423 PB of raw data accumulated in Run 3 by February 2025 (EPJC, 2026). Data taking stopped when the LHC entered Long Shutdown 3 on 29 June 2026, with operation scheduled to resume around 2030, so the ALICE series in Section 4 peaks in 2024–25, falls to reprocessing-only volumes in 2027–29 and returns in 2030. This deployment was absent from the first version of this report and was identified through the independent Sol/Kimi study. confirmed (design, farm, stored volume) / estimate (annual bytes through the rANS stage, which ALICE does not publish)

### 3.10 Game-repack compressors: LOLZ and RAZOR

A second deployment invisible to a product-name census is the game "repack" scene. **LOLZ** (ProFrager, 2017–18) is described by its author as "a compressor built on adaptive rANS" with LZ parsing; **RAZOR** (Christian Martelock, 2017) is a closed ROLZ archiver whose reconstructed source is reported on encode.su to use rANS (not independently verified here). Public repacking guides show recipes such as `xtool:zlib + srep:m3f + lolz`, and the largest repack site recorded about 39.6 million visits in the three months to June 2026 (Similarweb), i.e. a few million people downloading tens of gigabytes each. Under this report's definition — bytes transmitted in an ANS-coded format — that could be of order 0.1–3 EB/yr, but visits are a weak proxy (repeat visits, aborted and cached downloads), so following the review the estimate is *excluded from all totals* and kept here as a qualitative finding. The activity is copyright-infringing; it is recorded here as evidence of where adaptive rANS is used at scale, not valued, and its savings relative to LZMA-class packaging are near zero (adaptive rANS matches a range coder's ratio; the gain is speed and LOLZ's modelling). likely

### 3.11 JPEG AI and learned compression

**JPEG AI** (ISO/IEC 6048-1:2025 | ITU-T T.840.1), the JPEG committee's learned image-coding standard, replaced its range coder with an ANS coder at the committee's 99th meeting (May 2023) and was published in 2025; contributors include Huawei, ByteDance and Tencent researchers. ByteDance's PCT filings on JPEG AI codestream markers and on storing JPEG AI images in HEIF files (WO2025155739A1, priority January 2024; WO2025259668A1, priority June 2024) state that the codestream segments are "parsed with Meta-Tabled Asymmetric Numeral Systems (me-tANS) entropy coder" — confirming the name of the standard's normative coder and showing that ByteDance holds signalling and file-format IP around it (the filings concern markers and containers, not the coder itself) confirmed. No decoder ships in a browser or camera yet, so it carries no volume in Section 4 — it is the second ISO image standard to adopt ANS after JPEG XL. Research learned-compression systems that use ANS were listed in Section 3.8 (CompressAI, constriction, BB-ANS/Bit-Swap/HiLLoC); Huawei's PILC (CVPR 2022) and OSOA papers describe GPU-oriented lossless image coding with a modified ANS stage at ~200 MB/s on a V100, and Alibaba/DAMO's ANS-LIC (2025) is a parallel ANS hardware design for learned codecs — all research, none deployed. confirmed (standards) / likely (research details)

### 3.12 Other implementations and deployments

**Media lineage.** Google Brunsli (rANS; ~22 % lossless JPEG repacking; became JPEG XL's JPEG-recompression mode), PIK (rANS), WebP 2 (ANS; experimental, "will not be released as an image format"), and the 2016–17 rANS experiment in libaom that was removed before AV1 1.0.

**General-purpose compressors.** lzturbo / TurboANX (Hamid Buzidi, since 2014), Turbo-Range-Coder (adaptive-CDF rANS with SSE/AVX2/NEON/RISC-V RVV), Kanzi (Java/Go/C++, ANS0/ANS1 from ryg_rans), Dropbox DivANS (Rust, 2018), zhuff, Lizard (uses huff0 — Huffman — despite depending on the FSE library).

**Pure re-implementations of zstd's FSE** — klauspost/compress (Go), ruzstd (Rust), ZstdSharp (C#), aircompressor (Java, used by Trino), fzstd (JS), Zig `std.compress.zstd` — plus AMD/Xilinx Vitis zstd FPGA kernels.

**Hardware.** Xbox Series BCPack (rANS ASIC); Microsoft US 11,234,023 and family; Najmabadi et al. (Stuttgart, 2018) FPGA tANS decoder vs canonical Huffman; a 2024 IJET low-cost FPGA tANS encoder; Vitis zstd kernels (FSE decode in HLS).

**Patent literature (Chinese vendors, v2.1).** Huawei has a pending family on a division- and modulo-free ANS encoder/decoder (EP4637150A1 / HK40128054A, priority 9 January 2023) — a coder design with no application domain stated in the text — and application filings that put ANS in the claims: CN118202389A (learned point-cloud compression, claim 21: "asymmetric numeral system (ANS) entropy decoding", priority October 2021) and CN119497858A (neural-network weight compression, claim 9: ANS encoding scheme, priority July 2022); SenseTime's CN114363615B and CN116437087A mention ANS only as one admissible entropy coder beside arithmetic coding; Inspur (Suzhou) holds CN113572479B (granted December 2021) on generating FSE tables "for FSE compression and decompression realised by hardware under the zstd specification"; ByteDance's JPEG AI filings are described in Section 3.11. None of these documents a shipped product, so they carry no volume, but they show that the ANS coder itself is now the subject of vendor IP in China confirmed (documents read) / estimate (deployment).

**Genomics beyond CRAM.** htscodecs JavaScript port, cram-js (WASM), htsjdk (Java rANS 4×8 and Nx16), noodles (Rust), Genozip (wraps htscodecs rANS), fqzcomp5, PetaGene (patented ANS-based genomic coding).

**Libraries by language.** Rust: constriction, rans-rs, ryg-rans-rs (2026), ans (bits-back, 2026), zune-entropy (tANS), webgraph-ans-rs, jxl-oxide, jxl-rs. Python: constriction, craystack, Stanford Compression Library, py-rans, YAECL. Go: kanzi-go, klauspost fse. Java: Kanzi, libANS. JS/WASM: htscodecs, Draco decoder, fzstd. C/C++: FSE, ryg_rans, rans-tricks, hypersonic-rANS (AVX-512, 2023), Duda's AsymmetricNumeralSystemsToolkit.

**New codecs 2025–26.** Falcon (rANS audio codec, Rust), mzc (tANS archiver), @dotprotocol/compression (npm), marc (C++20), KVTC rANS mode.

### 3.13 Frequently mistaken for ANS (checked, not counted)

| Item | What it actually uses |
|---|---|
| Brotli | Huffman (RFC 7932) with context modelling |
| AV1 / AVIF | Daala-derived multi-symbol adaptive arithmetic coder; the rANS experiment was removed before 1.0 |
| HEIC / HEVC / H.264 / VVC | CABAC |
| GDeflate (DirectStorage PC, RTX IO) | Deflate/Huffman, bit-swizzled for 32-way SIMD |
| Oodle Kraken / Mermaid / Selkie (default modes) | Multi-stream Huffman; Selkie uncompressed streams; tANS optional since 2018 and rarely used |
| PS5 hardware decompressor | zlib + Oodle Kraken (Huffman-class) |
| NVIDIA Blackwell Decompression Engine | LZ4, Snappy, Deflate |
| ZipNN, DFloat11, Huff-LLM, Cloudflare Unweight | Huffman on float exponents (ZipNN optionally plain zstd) |
| ZipServ; TDT (tensor transform); AMD hipCOMP-core | ZipServ: fixed-length bitmap codec; TDT: reversible transform whose back-end may or may not be ANS; hipCOMP-core lists ANS as "not supported (NVIDIA proprietary)" |
| Lizard (LZ5) | huff0 Huffman (a component of the FSE library, not tANS) |
| LZ4, Snappy, LZO, LZVN, LZBITMAP, Apple LZMESH | No entropy coder / Huffman |
| Hugging Face Xet, Vortex, FastLanes | Chunk dedup, LZ4/BG4, lightweight encodings — no ANS |
| Apache Arrow | Codec-independent memory format; Arrow IPC offers *optional* LZ4 or zstd buffer compression (default uncompressed) — an optional zstd carrier, not a false positive and not counted |
| Huawei Kunpeng KAEzstd (kZIP engine); Intel QAT zstd plugin | Hardware performs only the LZ77 match stage of zstd ("lz77_zstd" algorithm; compression only, decompression in software — Huawei KAE documentation; Intel's plugin is an *external sequence producer*); FSE and Huffman coding stay in the zstd software library, so these are not ANS-in-silicon |
| Xiaomi point-cloud (G-PCC-style) patents | Context-adaptive arithmetic coding; ANS appears only in boilerplate ("may be any other type of entropy coders like asymmetric numeral systems", US12579696B2) — not evidence of ANS use |
| Tencent neural image compression (US12225205B2 and the TNC codec line) | Arithmetic coding — a negative data point for Tencent ANS use beyond downstream zstd |
| TensorFlow Compression, L3C, torchac, NNCodec | Range / arithmetic coding |
| CCSDS 121.0-B-3 (space) | Rice adaptive coding |

### 3.14 Design parameters of deployed ANS coders

The deployed coders span the whole ANS design space — tabled and range variants, static and adaptive probabilities, 4-bit to 25-bit alphabets. The table records what the public specifications and source code state; proprietary details are marked.

| Codec | Variant | Alphabet(s) coded with ANS | Probability model | Table / state size |
|---|---|---|---|---|
| Zstandard | tANS (FSE) | Literal-length codes (36), match-length codes (53), offset codes (window-dependent, ≥22 recommended); literals themselves are Huffman (max code length 11) | Per-block transmitted table, predefined table, repeat of previous table, or RLE | Accuracy log ≤9 (LL, ML), ≤8 (offsets); defaults 6/6/5 bits; Huffman weights FSE-coded with accuracy log ≤6 (RFC 8878) |
| Apple LZFSE | tANS (FSE) | L (20), M (20), D (64) and literals (256) | Per-block normalised histograms | 64 / 64 / 256 / 1024 states (lzfse_internal.h) |
| Apple LZRAVEN | rANS, 8-way interleaved (RE) | 16-symbol adaptive CDFs, Daala/AV1-style | Symbol-adaptive contexts | 8 interleaved states (liblzraven; Apple does not document) |
| JPEG XL | rANS (or prefix codes) | Up to 256 symbols per histogram; HybridUint splits large values into token + raw bits | Clustered histograms per context, fixed within a stream; meta-adaptive context trees | 12-bit tables (4096), 32-bit state (libjxl ans_params.h) |
| Draco | rANS | Integer symbol alphabet sized to the observed unique symbols; large values as raw bits | Per-stream encoded histogram | Precision 12–20 bits chosen from the unique-symbol bit length (rans_symbol_encoder.h) |
| CRAM 3.0 / 3.1 | rANS 4×8; rANS Nx16 (N = 4 or 32) | Bytes (256), order-0 or order-1 (previous byte as context); RLE/bit-pack/stripe transforms in 3.1 | Static per-block frequency tables | 8-bit vs 16-bit renormalisation; 12-bit (order-0) / 10–12-bit (order-1) frequency totals; 32 interleaved states for SIMD (CRAMcodecs) |
| Oodle LZNA / BitKnit | rANS, adaptive | LZNA: nibble (16-symbol) models; BitKnit: model-dependent | Symbol-adaptive (deferred-summation) models | 32-bit state, 16-bit renormalisation; two interleaved states in BitKnit (Giesen) |
| Oodle Kraken family tANS mode | tANS (optional) | Byte streams (≤256) | Static per-block | Rarely selected; Huffman modes dominate (Giesen) |
| Xbox BCPack | rANS (hardware) | BCn colour endpoints and palette indices, coded separately | Proprietary | Proprietary (Microsoft GDK; US 11,234,023 describes a two-phase decoder) |
| nvCOMP gANS / dietGPU | rANS (GPU, batched) | Bytes; float16/bf16/float8 exponent streams (effective support often far below 256) | Per-chunk histograms; dietGPU 9–11-bit probabilities | dietGPU: 32-bit state, many interleaved streams per warp; nvCOMP proprietary |
| ALICE O² | rANS | Detector arrays with 4–25-bit value ranges (sparse alphabets up to $2^{25}$) | Static per-array tables reused across time frames, stored in the conditions database; escape symbol | Not published in detail (arXiv 2102.09649) |
| LOLZ (RAZOR probable) | rANS, adaptive | LZ tokens with adaptive models (author statement; DXT texture models documented, LZ-token internals closed) | Symbol-adaptive contexts | Not published |
| Learned codecs (CompressAI, JPEG AI) | rANS; JPEG AI ANS | Quantised latents; model-dependent tails/escapes | Model-conditioned CDFs (a different CDF per symbol from a neural prior) | CompressAI: ryg_rans 32-bit; JPEG AI: per ISO/IEC 6048-1 |

Three regimes recur: predefined or per-block static tables (zstd, LZFSE, JPEG XL, Draco, CRAM, ALICE), symbol-adaptive models (LZNA, BitKnit, LZRAVEN, LOLZ) and model-conditioned CDFs (learned codecs). tANS dominates the static regime because a table lookup replaces the multiply; rANS dominates wherever probabilities change per symbol or where many streams must be interleaved for SIMD/GPU throughput.

### 3.15 Deployment census with evidence labels

| Application | ANS role | Domain | Evidence class | Counted in totals as |
|---|---|---|---|---|
| Zstandard | FSE/tANS sequences + Huffman literals | OS, browsers, data platforms, games, web | Default-on (carriers); bytes Modeled | 10 components, 174 EB (2026) |
| Apple LZFSE / LZRAVEN | tANS; rANS (OS 27) | Apple OS updates, archives, file system | Default-on; OTA mix Modeled | 9 EB (2026), transfer |
| JPEG XL | rANS | Browsers, cameras, DNG, DICOM | Default-on decoders; bytes Modeled | 0.2 EB (2026); Chrome scenario from 2027 |
| CRAM 3.x | rANS 4×8 / Nx16 | Genomic alignments | Default-on; volumes Measured use (archives) | 75 PB (2026), storage |
| Xbox Series BCPack | rANS in hardware | Console texture streaming | Default-on hardware; share Modeled | 2.5 EB (2026), transfer |
| Oodle (ANS modes) | rANS (LZNA/BitKnit); tANS option | Games (UE default pak codec) | Optional capability | 1.5 EB (2026), transfer |
| Google Draco | rANS | glTF / 3D web | Optional capability (optional glTF extension) | 6 PB (2026), transfer |
| nvCOMP gANS / dietGPU / NeuZip | rANS (GPU) | Checkpoints, weights | Optional capability; volume unknown | 0.1 EB (2026), illustrative |
| ALICE O² | rANS, static tables | Particle-physics data reduction | Measured use (423 PB stored) | 0.08 EB (2026), storage |
| LOLZ / RAZOR repacks | adaptive rANS | Game-repack distribution | Modeled, weak proxy | excluded from totals |
| DirectStorage 1.4 (PC) | zstd FSE/tANS, CPU and GPU decode | PC game asset streaming | Optional capability (public preview, Mar 2026) | not counted (2030 swing factor) |
| Discord gateway | zstd streaming (FSE) | Real-time messaging | Default-on since Apr 2024 | in zstd messaging component |
| Fedora Btrfs root; OpenZFS zstd | zstd (FSE) | Local and server filesystems | Default-on (Fedora); volumes Modeled | 5.0 EB (2026), storage |
| JPEG AI; PIK; Brunsli; WebP 2 | ANS / rANS | Image standards and precursors | Research / superseded | not counted |
| Token-native storage; UCCL-Zip; ANS-GEMM; GS-NFS; RAS; MANS; PILC/OSOA; ANS-LIC | rANS / ANS | AI, HPC, neural rendering, hardware | Research | not counted (2030 options in Section 5) |
| Huawei, Tencent, Baidu, other hyperscalers (patent literature: Section 3.12 and 7.4) | — | — | Private | excluded |

## 4. Quantitative estimates: machines, users, data volumes, savings

All series are yearly, 2014–2030; 2027–2030 are extrapolations (shaded). The eight software families plus ALICE O² are coloured consistently. Every input is listed in Appendix B with its evidence class and the yearly values (Tables A1–A7) in Appendix A; the model is a short Python script delivered with this report, and a second, *high* scenario (the version-1.1 assumptions) is shown as a dashed line wherever it differs materially. Reach figures (machines, users) are the best-anchored quantities; byte volumes are order-of-magnitude estimates; dollar figures depend on the retention convention explained in Section 4.5.

### 4.1 Machines carrying an ANS decoder

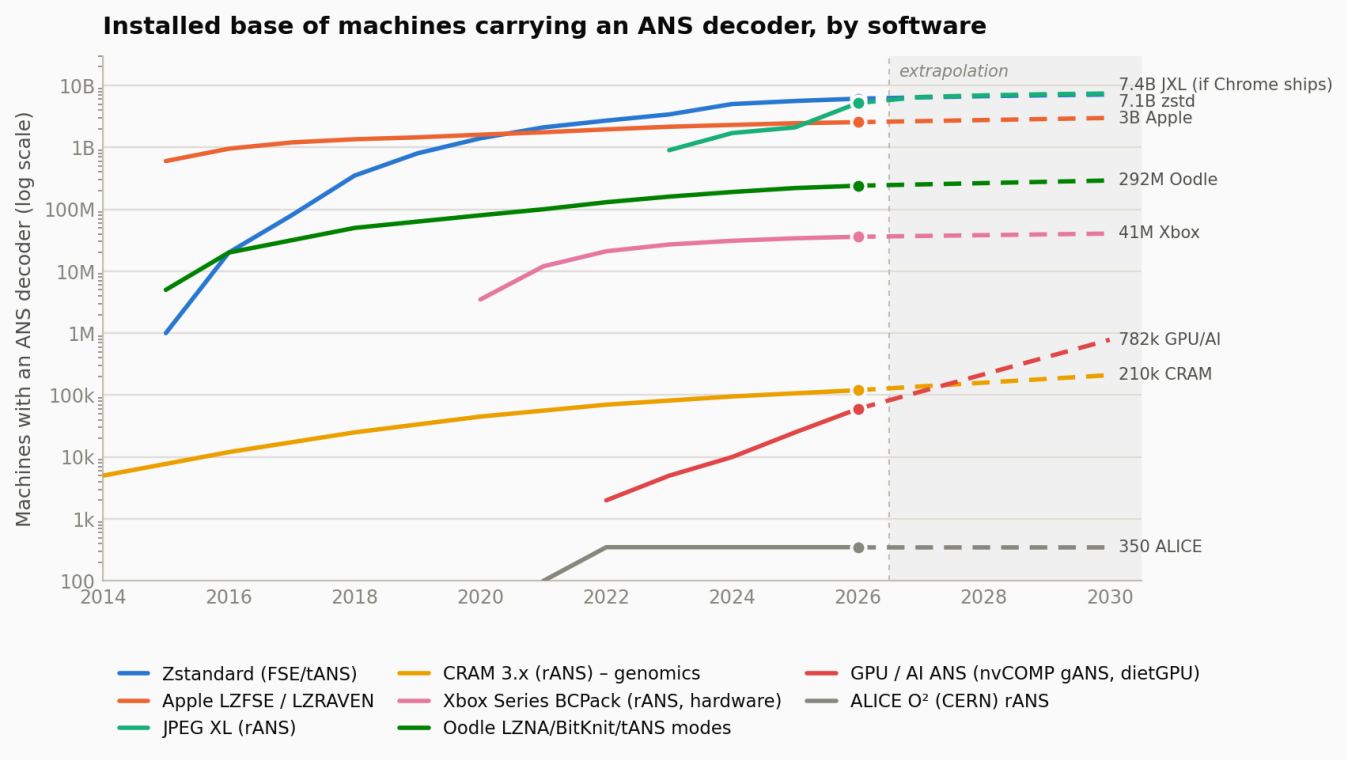



**Figure 2.** Installed base of machines carrying a decoder for each ANS-based software (log scale). Lines overlap — an iPhone carries LZFSE, a JPEG XL decoder and, since Safari 26.3, a zstd decoder — so they must not be summed. Solid: anchored; dashed: extrapolated. JPEG XL's 2027 rise assumes Chrome ships default-on in 2027 (scenario). Capability is not use: a decoder present on a device says nothing about how many bytes it decodes.

The zstd line rises from tens of millions of Linux servers (2016–17) through Android devices launching on 4.14+ kernels (2018–2020), Windows 11 (2023), Chrome and Firefox (2024) and finally Safari/iOS 26.3 (2026), at which point essentially every active phone, PC and server — about 6.1B devices — contains an FSE decoder. Apple's LZFSE line tracks Apple's disclosed active-device base (≈2.55 B on iOS 9+/OS X 10.11+ in 2026). JPEG XL decoders reach ≈2.5 B devices in 2026 (Apple, Windows extension, Firefox 157 on 29 September) and would exceed 4.5 B in 2027 with Chrome. The Xbox line (≈36 M consoles) is small in count but unique as hardware rANS. CRAM's ≈120k compute nodes, the GPU/AI line (≈60k GPUs in 2026, illustrative) and ALICE's 350 servers are orders of magnitude smaller and are shown to fix the scale. Draco decoders are fetched per page visit and are not an installed base; Draco appears in the user chart instead.

### 4.2 Users

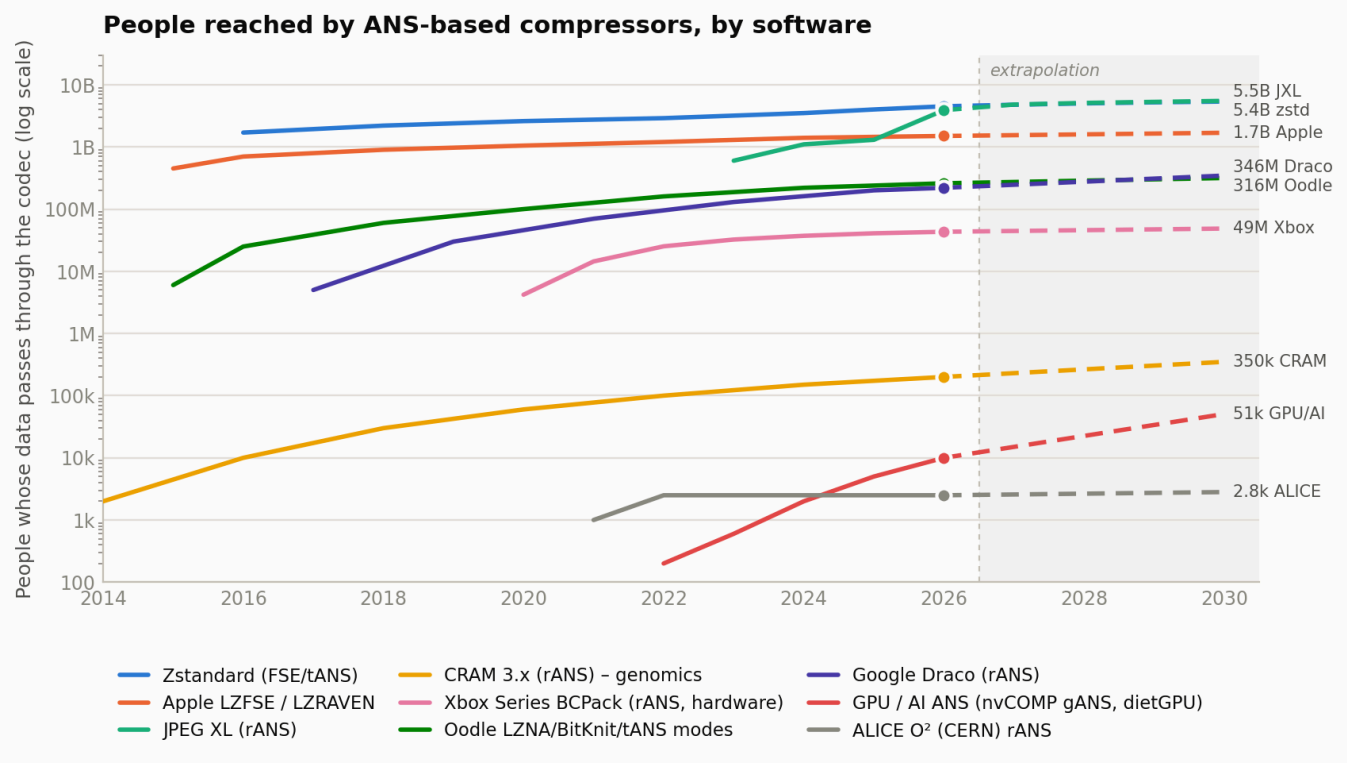



**Figure 3.** People whose data is coded by each software, directly or through a backend they use (log scale). zstd's user count starts with Meta's user base (backend zstd from 2016) and rises with browser and CDN support; CRAM and GPU/AI count practitioners rather than the people whose data is stored.

By 2026 roughly 4.5B people — most of the world's 6.0 B internet users, bounded below by Meta's 3.58 B daily active people — receive or store zstd-coded data through Meta's services, Cloudflare's default compression, Steam, Linux/Android systems and their browsers. Apple's users (≈1.5 B people behind 2.5 B devices) receive LZFSE-coded software updates several times a year. JPEG XL's users are today Apple's iOS 17+ users plus Firefox users from late September; the central scenario adds Chrome's ~3 B users in 2027. Oodle-compressed games reach ~260 M players, but only the ANS-mode share is ANS-coded. CRAM's ~200k practitioners, the ~10k engineers using GPU ANS and ALICE's ~2,500 collaborators complete the picture.

### 4.3 Data volume coded with ANS

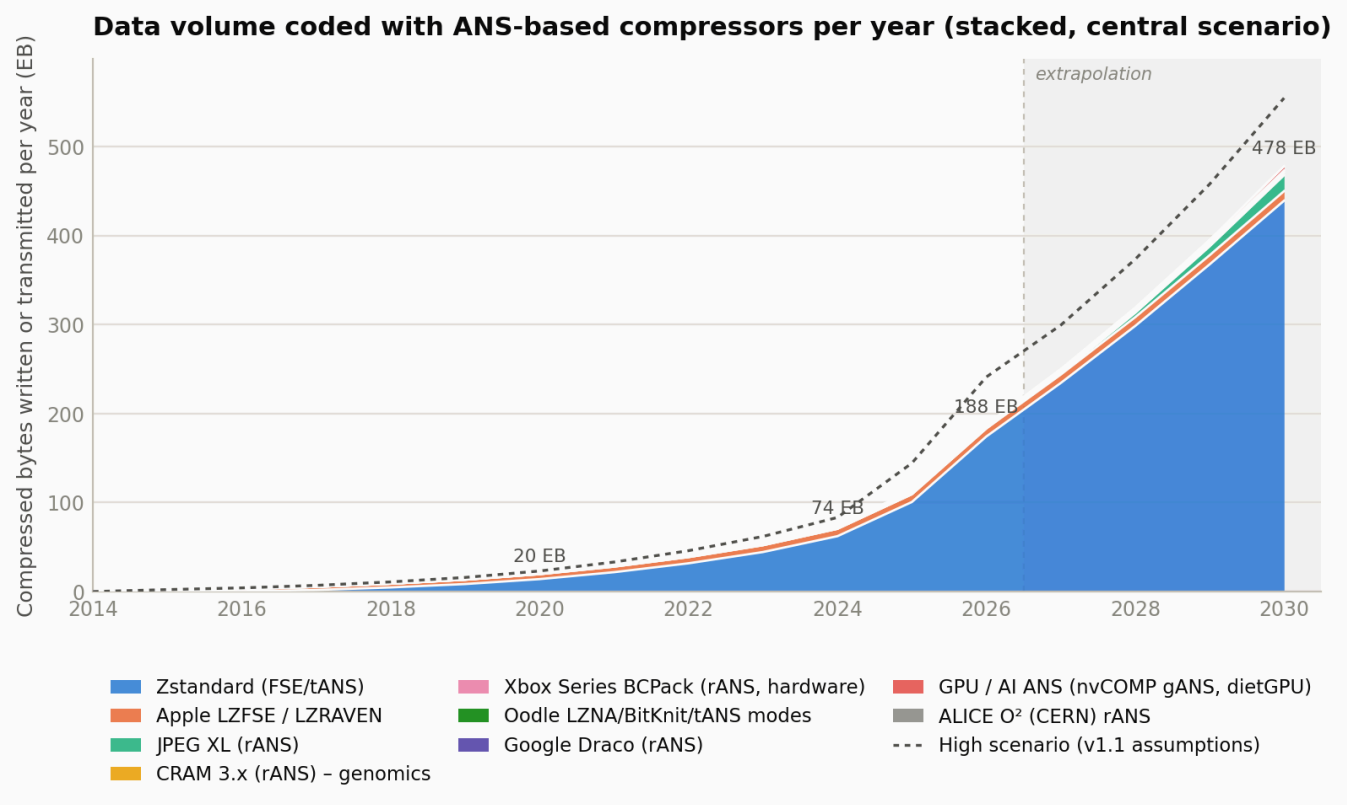



**Figure 4.** Compressed bytes written to storage or transmitted per year in ANS-coded formats, stacked by software (EB = $10^{18}$ bytes), central scenario; the dashed line is the high scenario (version-1.1 assumptions). The total's uncertainty range is ×0.37 to ×2.3 in 2026.

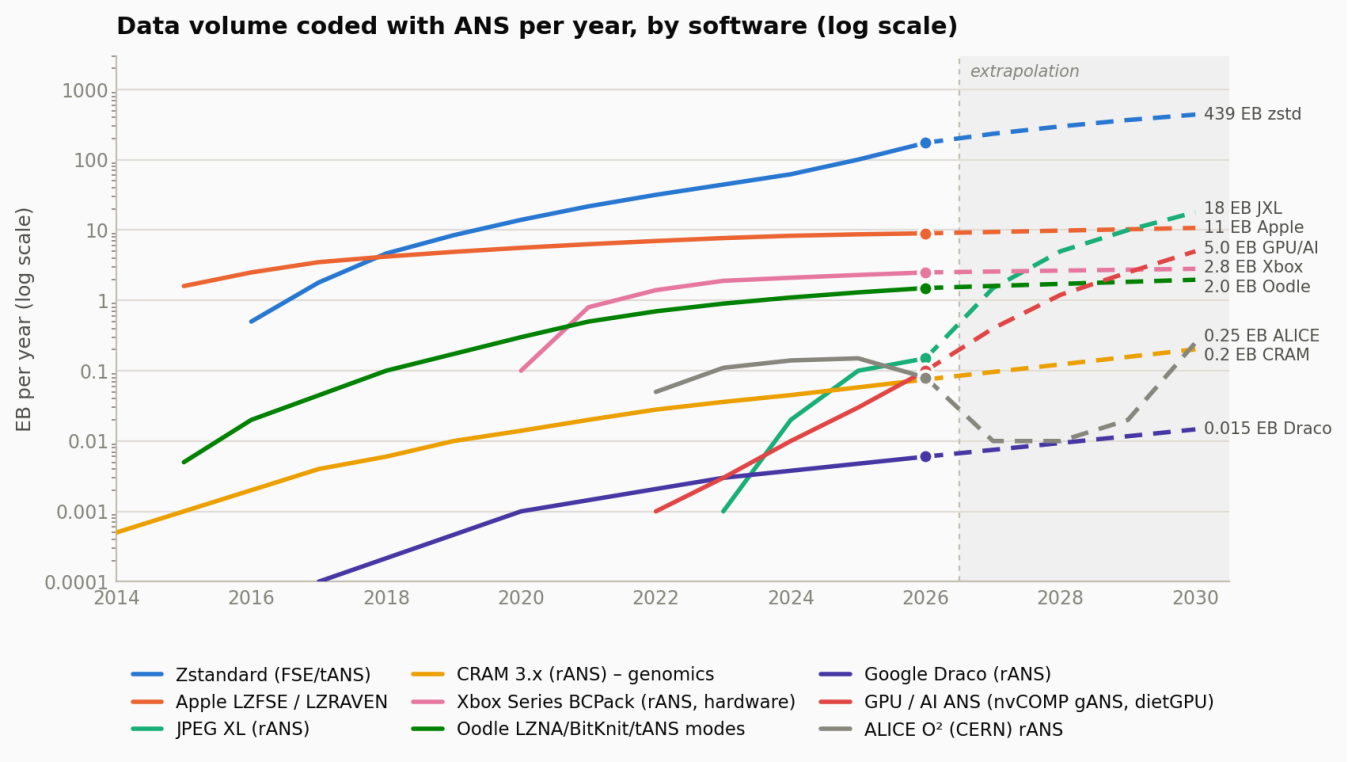



**Figure 5.** The same volumes on a log scale, so that the smaller families are visible. Draco's 15 PB/yr and CRAM's 200 PB/yr (2030) are roughly 4½ and 3½ orders of magnitude below zstd; ALICE's line drops during LHC Long Shutdown 3.

Zstandard accounts for ~93 % of ANS-coded bytes in 2026 (≈174 EB of 188 EB). Figure 6 breaks it down into eight components, each carrying an evidence label in Appendix B. **Steam** is the best-measured flow (Valve: 80 EB in 2024, 100 EB in 2025) but its zstd share is modelled: the client gained zstd chunks in February 2025, the first live chunk was seen in May 2025, and only new and updated chunks use zstd, so the central scenario puts 15 % of 2025 and 40 % of 2026 delivered bytes on zstd (≈55 EB), rising to ~78 % by 2030 as depots churn; the high scenario's 70 % in 2026 is retained as an upper case. **HTTP zstd** is anchored on Web Almanac shares (2.7 % → 12 % of CDN responses) applied to an estimated 150–300 EB/yr of compressible web text. **Data platforms** are split into a durable component (analytics tables and database storage, where Databricks, Iceberg, ClickHouse Cloud and Elasticsearch now default to zstd while Parquet/Spark still default to Snappy) and a transient component (streams, logs, caches, compactions, backups and CI caches such as GitHub Actions and bazel-remote); both are modelled from product defaults, not telemetry. **Meta** is split the same way, anchored on its "vast majority of compression use" statement and exabyte-scale warehouse. A small **scientific/HPC** component covers ROOT RNTuple, Zarr, HDF5/NetCDF filters, Oxford Nanopore VBZ and GDAL; OS and package distribution is small.

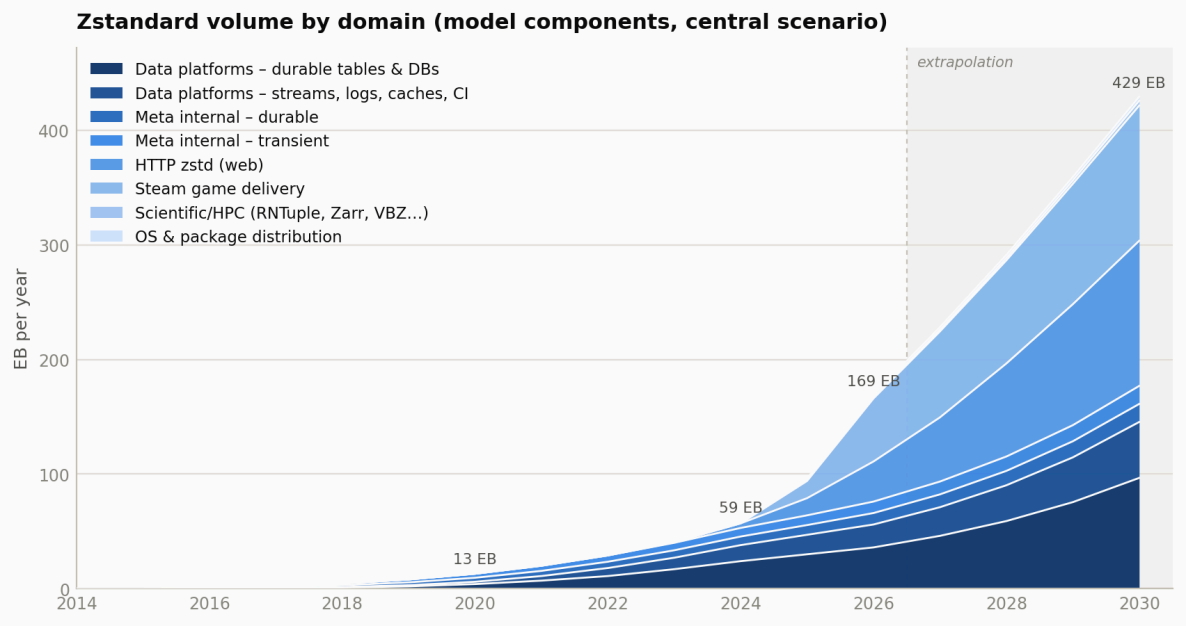



**Figure 6.** Zstandard's modelled volume by domain (central scenario). Steam is measured in total but modelled in share; HTTP is anchored on Web Almanac shares; storage components are order-of-magnitude estimates from product defaults.

Apple's ≈9 EB/yr is dominated by software updates (≈2.5 B devices × ~4.5 GB/yr of update payloads × ~80 % updating), delivered as Apple Archive payloads (LZFSE default) until OS 26 and as LZRAVEN payloads from OS 27; because the algorithm mix of shipped payloads is not public, this is a modelled quantity with a range of ×0.2–×1.6. Xbox BCPack's ≈2.5 EB/yr assumes ~36 M consoles downloading ~250 GB/yr, 55 % BCn texture data, half of titles using the BCPack path. CRAM's 75 PB/yr in 2026 is consistent with ENA/EGA growth (~15–20 PB/yr), UK Biobank, All of Us, Genomics England and clinical DRAGEN output. JPEG XL is still ≈0.2 EB/yr (ProRAW, DNG, DICOM, Safari-only web delivery). GPU/AI ANS is *unknown*: the model carries an illustrative 0.1 EB/yr (range 0.01–0.4) — NVIDIA's own example job writes 1.13 PB/month, so 0.1 EB/yr corresponds to roughly eight such

jobs running continuously. ALICE's compressed raw data was ~0.14 EB/yr in 2024–25 and falls to near zero during Long Shutdown 3. The game-repack scene is excluded from totals.

### 4.3.1 Three denominators: output, encoder operations, decoder operations

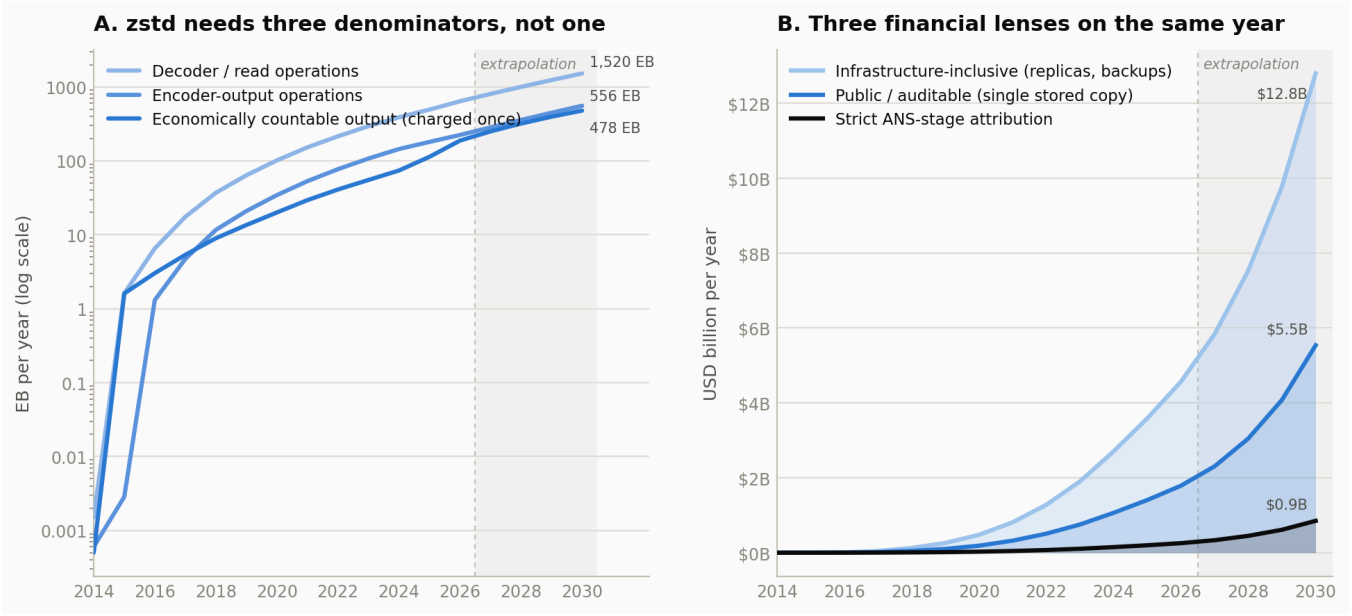



Figure 9. Left (A): the same fleet measured three ways. Only the lowest line — output charged once — feeds the money model; the two upper lines are machine work. Right (B): the same year seen through three financial lenses (Section 4.5). Shaded: extrapolation 2027–2030.

Applying the per-component multipliers of Appendix B (compaction, replication and cache fills for encoder operations; reads per written byte for decoder operations) to the same volume series gives, for 2026, ≈188 EB of economically countable output, ≈224 EB of encoder-output operations and ≈639 EB of decoder/read operations; by 2030 the three are ≈478, ≈556 and ≈1,520 EB. The multipliers are coarse — a factor of two either way is easy to argue, and warehouse read amplification in particular could be far higher — so the operations curves should be read as order-of-magnitude statements about machine work, not as measurements. Their purpose is to stop a single number from carrying three meanings: the decoder figure is the one that answers "how much data does ANS touch", the output figure is the only one that may be turned into avoided bytes, and the encoder figure sits between them.

This distinction also explains why the report's totals can look low to an operator. Someone who sees every compaction, replica, cache fill and read inside a large fleet is watching the upper curves; a public model that charges each byte once is describing the lowest one. Both can be right at the same time (Section 7.5).

## 4.4 Savings in bytes

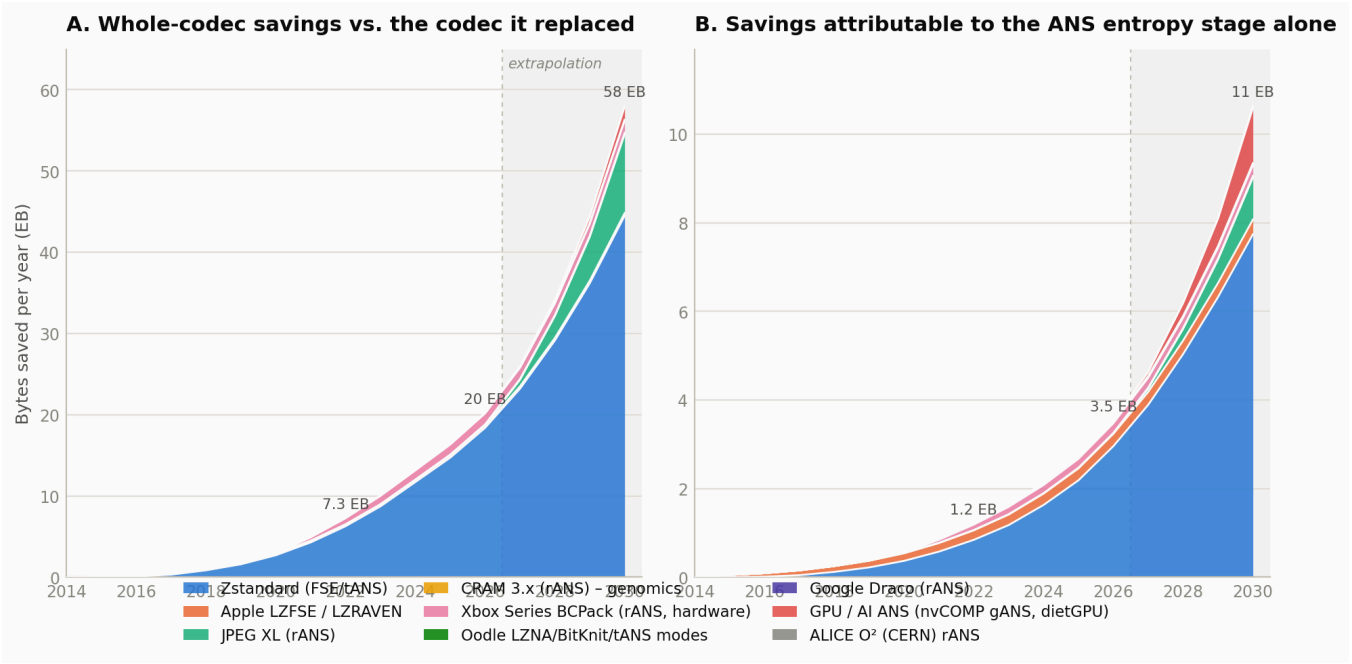



**Figure 7.** Bytes avoided per year. Panel A compares each ANS-based format with the format it displaced (whole-codec); panel B isolates the ratio advantage of the ANS coder over the Huffman coder the same codec would otherwise have used. Steam and package distribution contribute nothing to A (zstd ≈ LZMA/xz in size; the gain there is speed).

Whole-codec savings reach ≈20 EB in 2026 — about 10 % of the ANS-coded volume — because the dominant migrations (gzip/Snappy/LZ4 → zstd, BAM → CRAM, zlib → BCPack) save 15–50 % while the LZMA → zstd migration at Steam, the largest single flow, saves nothing in bytes. ANS-stage savings are ≈3.5 EB in 2026, roughly a fifth of the whole-codec figure, rising to ≈11 EB in 2030 as GPU float codecs (where the ANS stage is most of the gain) and JPEG XL grow. Cumulatively, ≈78 EB (whole-codec; range 23–308) and ≈14 EB (ANS-stage) have been avoided since 2014.

### 4.5 Financial savings

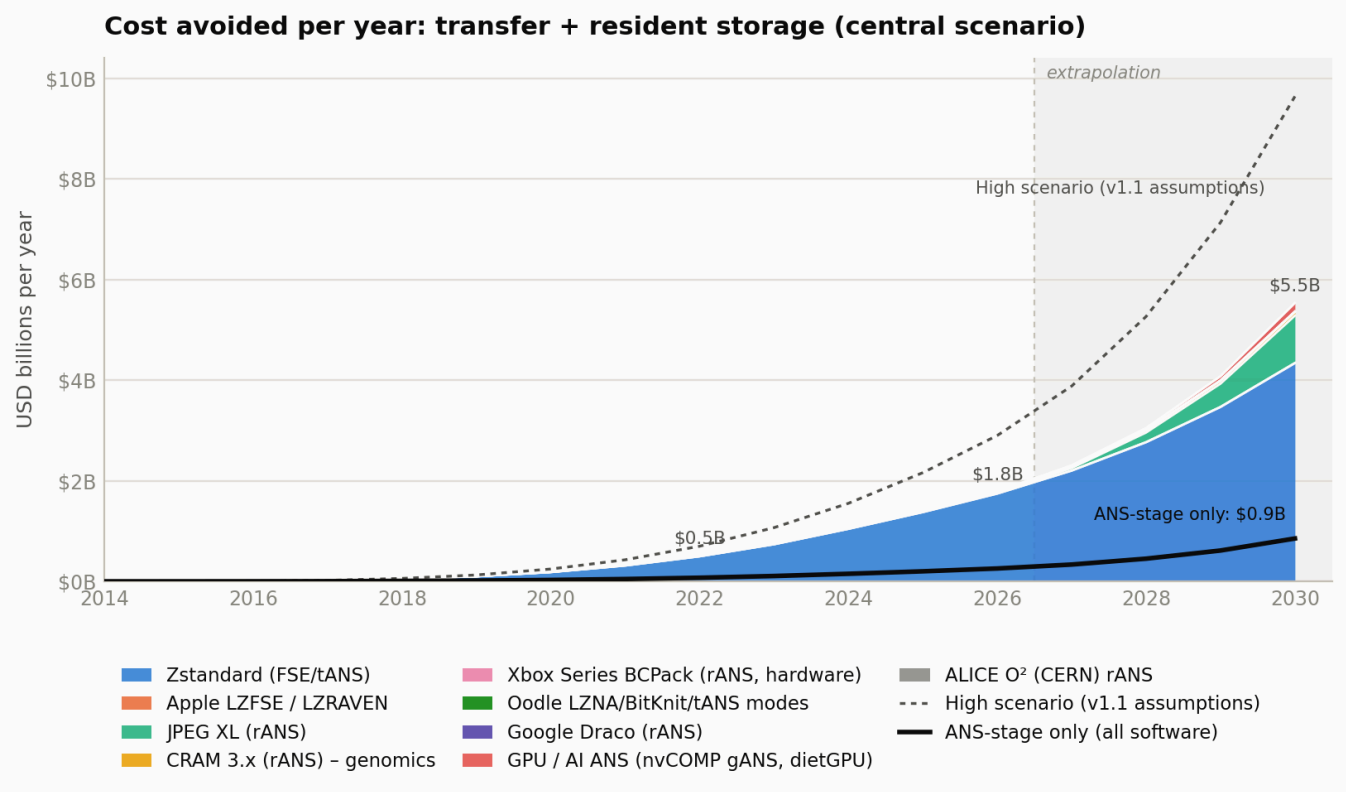



**Figure 8.** Cost avoided per year (USD, central scenario): transferred savings priced once at $2–8/TB, stored savings charged on the *resident* footprint at $30–100 per TB-year with retention set per component (80 %/yr for durable tables, 10 %/yr for logs and caches, 5 %/yr for checkpoints, 95–97 %/yr for archives). Dashed: high scenario (flat 85 % retention, larger volumes). Black: ANS-stage attribution.

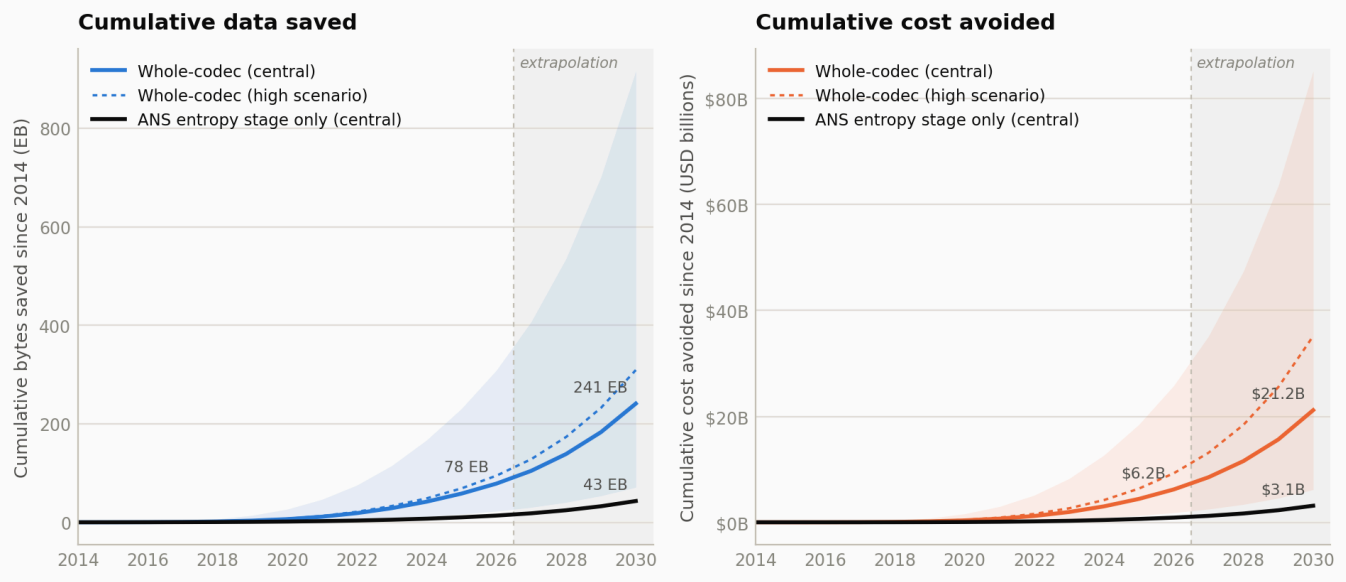



**Figure 9.** Cumulative bytes and cost avoided since 2014, central and high scenarios, with the uncertainty band from the component ranges in Appendix B.

The central estimate is $1.8B avoided in 2026 (range $0.5B–$7.3B), ≈$6.2B cumulative since 2014 (range $1.8B–$25.8B), and $5.5B/yr by 2030 with ≈$21.2B cumulative. About four fifths of it is storage cost avoided on durable data in data platforms and at Meta; transfer savings (web, Apple, consoles) are one-off and priced at a few dollars per terabyte, and transient stores (logs, caches, checkpoints) contribute little because their saved bytes are resident for weeks, not years. The resident saved footprint behind the 2026 figure is ≈35 EB. The ANS-stage attribution is $255M in 2026 (range $76M–$1,032M) and $851M in 2030.

**Three lenses, reported side by side (v2.2).** A single dollar figure invites false precision, so the model now reports three: a **public/auditable** case that charges avoided bytes against one stored copy ($1.8B/yr in 2026, range $0.5B–$7.3B); an **infrastructure-inclusive** case that charges the copies an avoided byte is actually avoided in — three-way replication or erasure coding plus backups and cross-region copies for durable platform data, two for transient and cache tiers ($4.6B/yr); and **strict ANS-stage attribution**, which credits only the entropy coder ($255M/yr). By 2030 the same three are $5.5B, $12.8B and $851M per year, and cumulatively since 2014 $6.2B, $15.8B and $0.9B to 2026. The replication term was absent from versions 1.x–2.1 and is a genuine understatement in those editions: avoided bytes are avoided in every copy. All three are gross resource costs avoided, not profit, and none of them nets out the CPU and energy cost of running the codec.

**Three readings of the same model.** The *central* figure above ($1.8B/yr in 2026) treats Steam's zstd share, data-platform volumes and Apple's OTA payloads as modelled quantities and separates durable from transient retention. The *high* scenario ($2.9B/yr; version 1.1) keeps the larger volumes and a flat 85 % retention. A *measured-only* reading — counting only flows with operator-published volumes or default-on evidence and charging one year of storage per avoided byte — lands near the low end of the range, $0.3–0.8 B/yr, which is where Sol's review of version 1.1 placed a defensible public headline. The ANS-stage attribution spans the same three readings: ≈$0.1 B, $255M and $410M. For orientation, global spending on storage hardware and cloud storage is of order $100 B a year, and zstd's addressable share — structured data that would otherwise be gzip/Snappy/LZ4-compressed — is a few percent of stored bytes, so a figure between several hundred million and a few billion dollars a year is the right order of magnitude.

### 4.6 Speed and energy: what the byte counts miss

The dollar figures deliberately exclude the effect that motivated FSE in the first place. Collet's design goal was arithmetic-coding-class ratio at Huffman-class speed using "only additions, table lookups, and shifts"; his benchmarks put FSE at 325/440 MB/s (compress/decompress) per core against an adaptive range coder's ~100–200 MB/s, and huff0 at 600/1350 MB/s. Every migration in this report was justified by speed as much as size: Steam's move from LZMA to zstd buys no bytes at all but roughly an order of magnitude in decode throughput on 100 EB/yr; Fedora's RPM switch cut decompression of its corpus from ~96 s to 7–8 s; Arch cited ~13× faster decompression; Facebook's initramfs boot went from 12 s to 3 s; CRAM 3.1's SIMD rANS decodes at multiple GB/s per core; NVIDIA's gANS runs at ~530 GB/s where zstd manages ~16 GB/s on the same GPU and cuts checkpoint wait by 15 %; Apple states LZRAVEN decodes three times faster than LZMA and chose it for OTA payloads on 2.5 B devices, where decode time is battery time; ALICE's rANS stage exists because 90 GB/s had to be sustained on a 350-server farm.

A rough scale for the compute involved: if the ≈188 EB/yr of ANS-coded data were decoded an average of five times each and an ANS decoder saves ~5 ns per byte relative to an adaptive range coder at the same ratio, the avoided CPU time is of order $10^5$ core-years per year — a few megawatts of continuous server load, or several tens of millions of dollars a year at cloud prices. This is a Fermi estimate, not a measurement; it is small next to the storage savings, and it compares ANS with arithmetic coding rather than with Huffman — against Huffman the speed is roughly a draw and the benefit is the ratio counted in Section 4.4. The real value of the speed is the adoption it unlocked: no browser, console, kernel or archive would have shipped a range coder in these roles. ANS's economic contribution is therefore bounded below by the ANS-stage byte savings (≈$255M/yr in 2026) and above by the whole-codec savings (≈$1.8B/yr central, $2.9B high), with the truth depending on how much of zstd's, CRAM's and JPEG XL's adoption one credits to the speed–ratio point that ANS made available.

# 5. Extrapolation to 2030: drivers, scenarios and risks

The dashed segments in Figures 2–9 are not curve fits; each series is carried forward with a stated growth rate or scenario (Appendix B) chosen from the drivers below. The central path takes ANS-coded volume from ≈188 EB/yr (2026) to ≈478 EB/yr (2030, range 200–1,452) and annual whole-codec cost avoided from $1.8B to $5.5B; the high scenario reaches 555 EB/yr and $9.6B.

## 5.1 Drivers already in motion

- **HTTP zstd becomes a default, not an option.** With Safari 26.3 (Feb 2026) every major browser decodes zstd; Cloudflare compresses Free-plan traffic with zstd by default and is transcoding its cache to zstd; Google's CDN serves ~10 % zstd. Web Almanac shares went 2.7 % → 12 % in one year. The model assumes 60 %, 45 %, 30 % and 20 % annual growth of zstd web bytes in 2027–2030, reaching roughly a third to a half of compressible web text. Brotli remains the incumbent for static assets (≈7 % smaller than zstd at high levels), so zstd's web share saturates below 100 %.
- **Data-platform defaults.** Databricks (2024), Athena/Iceberg tables, ClickHouse Cloud, Elasticsearch best_compression, Trino/Delta, MongoDB and Kafka now default to or recommend zstd; Parquet-with-zstd is becoming the lake standard, although Spark/Parquet's own default is still Snappy. Structured-data writes grow 25–30 %/yr; the model carries the durable component at 28 %/yr and the transient one at 25 %/yr. A downside risk is Meta's OpenZL (Oct 2025), a format-aware framework that may displace generic zstd inside Meta; its back-end codecs reportedly include zstd and the FSE/Huffman entropy coders, so ANS-coded bytes would not necessarily fall, but attribution would blur.
- **Game delivery.** Steam's 100 EB/yr re-encodes to zstd only as depots are uploaded or updated; after twenty months of migration the model now assumes 50 % of delivered bytes in 2026 (was 40 %) rising to ~80 % by 2030 with 8 %/yr traffic growth, with the range widened to ×0.7–×1.9. On PC, DirectStorage 1.4's optional zstd path (March 2026 preview, CPU and GPU decode, with GPU-vendor optimisations promised for later in 2026) could add a second, larger game-asset channel: if it becomes the default authoring choice for new titles the way GDeflate did for DirectStorage 1.1, tens of EB a year of PC asset delivery and load-time decoding would become FSE-coded by 2030. Nothing ships with it today, so it carries no volume here. Unreal Engine's Oodle default keeps the Oodle installed base growing, but only its ANS modes count.
- **Apple.** OS 27 moves OTA payloads to LZRAVEN (rANS); LZFSE persists in APFS compression and legacy paths; LZMESH (Huffman) takes over general-purpose use. Apple's byte total stays ANS-coded and grows with the installed base (~4.5 %/yr).
- **Scientific and HPC.** ROOT RNTuple (zstd-5 default) is the planned event-data format for ATLAS and other LHC experiments in Run 4, Zarr v3 defaults to zstd, and Oxford Nanopore's VBZ codes all raw nanopore signal; the small scientific component grows 40 %/yr. ALICE's rANS stage is idle during Long Shutdown 3 (June 2026 – ~2030) and returns with Run 4.
- **Genomics.** ENA projects 150 PB within five years; UK Biobank, All of Us, Our Future Health, national programmes and clinical sequencing keep CRAM writes growing ~28 %/yr; CRAM 3.1 became the samtools default only in May 2025, so 3.0 → 3.1 migration adds a further 7–16 % saving on new writes.

## 5.2 The swing factors

**JPEG XL has shipped.** This was version 2.1's largest open question and it is now settled: ChromeStatus carries milestone 155 for `image/jxl` on desktop, Android, WebView and iOS, and Chromium Dash puts M155 stable on 6 October 2026 — a week after Firefox 157. Browser capability therefore arrives at the end of 2026 rather than in mid-2027, and the central scenario now ramps web image traffic from that point — 1.5, 5, 10 and 18 EB/yr of JXL in 2027–2030 (≈0.4 %, 1.5 %, 3 % and 6 % of web image bytes; WebP took a decade to reach 12 % of image requests and AVIF is at ~1 % four years after universal support, but JPEG XL's lossless JPEG recompression lets CDNs transcode without quality decisions) — saving ~35 % against JPEG. The remaining uncertainty is no longer capability but content: encoders, CDN pipelines and authoring tools lag browsers by years, and WebP took a decade to reach 12 % of image requests. Cloudinary's operator counts (Section 3.3) show the demand side is already moving — 0.5–1 billion JXL responses a day, forecast to 3–4 billion by year-end — but responses are not unique encodes. If content migration stalls despite universal decoders, JXL stays near 1 EB/yr and the 2030 total falls by roughly 17 EB. Camera adoption beyond ProRAW (a default HEIF → JXL switch by any major phone vendor) is an upside not included: ~2 trillion photos a year at ~3 MB each is ~6 ZB/yr of source material.

**AI: three curves, not one.** (i) *Checkpoint and weight storage* — NVIDIA recommends gANS for GPU-resident checkpoint compression; if a handful of large labs adopt it, the central path reaches 5 EB/yr by 2030 (range ×0.1–×4) with ~25 % saving against uncompressed or zstd-coded checkpoints. (ii) *Communication* — UCCL-Zip-style rANS inside collectives compresses bytes that are never stored; volumes could be large (RL weight synchronisation moves full model copies many times a day) but produce no storage saving and are not modelled. (iii) *Token, KV and prompt stores* — token-native storage with ANS at 3.3× over UTF-8 targets agent memory and retrieval corpora whose size is unknown today; also not modelled. Two cautions apply to all three: FP8 and FP4 weights leave far less lossless headroom than BF16, and Huffman-based alternatives (DFloat11, ZipNN, Unweight) compete on decode simplicity. dietGPU was archived in June 2026; hardware decompression engines (Blackwell, Xbox DEFLATE mode) still favour LZ4/Snappy/Deflate over ANS.

## 5.3 Sensitivity

**What would move the numbers most, in the direction Collet's comment points.** Three of the model's terms are bounded below by public evidence and above by nothing: the private data-platform components (no operator publishes bytes), the replication factor (1 in the auditable case, 2–3 in the infrastructure case, and higher again if cross-region copies and disaster-recovery sites are counted), and read amplification. Taking the upper end of all three at once — which no single source supports and none excludes — would put 2026 near the top of the published range rather than at its centre. Version 2.2 publishes the range, the operations curves and the replication term instead, so that a reader with operator knowledge can place themselves inside it.

The 2030 totals are dominated by four assumptions, in order: the durable data-platform component (28 %/yr growth, 15 % saving fraction, 80 % retention; halving any of them cuts the dollar figure by a quarter to a half), the storage unit cost ($60/TB-yr; the plausible range $30–150 moves the total by ×0.5–×2.5), Steam's zstd share (each ten points of share is ≈15 EB/yr of volume in 2030, though no dollars, since the saving versus LZMA is speed), and JPEG XL content migration now that decoders are universal (±18 EB/yr by 2030). The ANS-stage attribution is most sensitive to the zstd sequence-stream saving fraction (2.5 %; defensible range 1–5 %) and to GPU float codecs' share. Chinese zstd-default platforms (Section 3.1) are an unquantified upside inside these ranges. The uncertainty bands combine per-component ranges and should be read as plausible envelopes, not confidence intervals.

Two structural risks could bend the curves downward: a successor to zstd that drops FSE for Huffman (as Apple did with LZMESH and Oodle did with Kraken — both for decode speed on wide-SIMD hardware), and hardware decompression engines standardising on Deflate/LZ4 rather than ANS. Four could bend them upward: LZRAVEN-style adaptive rANS as the pattern for the next high-ratio codecs, JPEG XL as a camera default, DirectStorage 1.4's zstd path as the PC authoring default, and ANS-GEMM-style inference on compressed weights in production serving.

## 6. Suggested additions to the encode.su implementation list

The thread itself could not be read from the research environment (encode.su rejects automated clients and archive mirrors were unreachable), so the list below is a set of *candidates* from 2018–2026 for you to check against the current post. Items are grouped by kind; each has a year, the ANS variant, and a URL in Section 9 (group "Catalogue"). Confidence: confirmed = source code or vendor documentation read; likely = secondary source.

### 6.1 Production deployments (highest value for the list)

| Item | Year | Variant | Why it matters |
|---|---|---|---|
| Microsoft Xbox Series X\|S **BCPack** hardware decoder confirmed | 2020 | rANS (ASIC) | Microsoft GDK: BCn texture entropy coder "compressed using a rANS algorithm"; hardware block in every Series X\|S. Related: US 11,234,023 (2022) and EP/CN/KR family. |
| Apple **LZRAVEN** (OS 27) likely | 2026 | adaptive rANS, 8-way interleaved | Apple's LZMA replacement; used for OTA payloads (pbzm). Third-party RE: liblzraven. Apple also introduced LZMESH (Huffman) to replace LZFSE — worth noting as a move away from tANS for the fast tier. |
| Apple **JPEG XL** decoder (ImageIO/Safari 17); iPhone 16 Pro ProRAW-JXL capture confirmed | 2023 / 2024 | rANS | Newest Apple ANS deployment after LZFSE; first mainstream camera writing ANS-coded photos. |
| **jxl-rs** (Google/libjxl org, Rust) in Chromium 145+ and Firefox 152+; **jxl-oxide** (Rust) confirmed | 2023–26 | rANS | Memory-safe JPEG XL decoders; Chromium and Mozilla Intents to Ship (Aug 2026). |
| Microsoft JPEG XL Image Extension (Windows 11); DICOM Supplement 232; DNG 1.7 confirmed | 2023–25 | rANS (JXL) | JPEG XL reaching Windows, medical imaging and raw-photo containers. |
| NVIDIA **nvCOMP gANS** (float16/float8 modes), **nvCOMPDx** device API; NVIDIA checkpoint-compression guidance confirmed | 2022–26 | rANS (GPU) | ~530 GB/s; recommended default for GPU-resident checkpoint compression (Apr 2026). |
| **Microsoft DirectStorage 1.4** (public preview, 11 Mar 2026): optional zstd with CPU and GPU decompression + GACL asset conditioning confirmed | 2026 | tANS (FSE) on GPU | First ANS path in the PC asset-streaming stack; GPU-vendor optimisations promised for later in 2026. |
| **Discord gateway** streaming zstd (Apr 2024) confirmed | 2024 | tANS (FSE) | zlib → zstd streaming; ratio 6 → 10, payload 270 → 166 bytes, ~40 % less WebSocket traffic at consumer scale. |
| **Fedora Btrfs `compress=zstd:1` by default** (F34, 2021); OpenZFS zstd datasets confirmed | 2021 | tANS (FSE) | Filesystem-wide transparent compression on a mainstream desktop distribution; ~40 % of disk saved in Fedora's own analysis. |
| **Box** ZSTD file transfers for all customers; **Cloudflare** cache transcoding prototype (zstd-3, 2.834×) confirmed | 2025–26 | tANS (FSE) | Enterprise content delivery and CDN cache capacity. |
| Safari 26.3 zstd; Steam depot zstd; Python 3.14 `compression.zstd`; SQL Server 2025; .NET 11; Android 17 zram zstd; Windows 11 libarchive .zst confirmed | 2023–26 | tANS via zstd | The 2023–26 wave of zstd (FSE) carriers, if the list tracks indirect deployments. |
| Scientific/HPC zstd carriers: ROOT RNTuple (zstd-5 default), Zarr v3 (Zstd default), Oxford Nanopore VBZ (StreamVByte + zstd), bazel-remote (zstd default), GitHub Actions cache, OCI `tar+zstd` layers, restic v2, GDAL ZSTD/LERC_ZSTD, Arrow IPC zstd; Chinese platforms: PolarDB for MySQL IMCI (`imci_default_codec = ZSTD`), OceanBase (zstd in default configuration) confirmed | 2018–25 | tANS via zstd | Default-on or optional zstd in science, build and backup tooling; no byte counts published. |
| CRAM 3.1 rANS **Nx16** (32-way SIMD) as samtools default; htsjdk (Java), cram-js (WASM), noodles (Rust), Genozip confirmed | 2021–25 | rANS | SIMD rANS in the genomics default path; independent language ports. |
| Google Draco in Unity ("Draco for Unity" 5.x), Blender glTF exporter, Cesium 3D Tiles 1.1 confirmed | 2018–26 | rANS | 18 M npm downloads/month for the JS decoder. |
| Dropbox **DivANS** confirmed | 2018 | rANS | Rust; adaptive CDFs; Dropbox tech blog. |
| Google **WebP 2** (libwebp2) confirmed | 2020– | ANS | Experimental; "Asymmetric Numeral System coder" in `src/utils/ans.h`. |
| CERN **ALICE O²** Run-3 rANS entropy stage confirmed | 2021– | rANS, static per-array tables | Production physics deployment: 300 → 90 GB/s design point on a 350-server GPU farm; 423 PB of compressed raw data by Feb 2025 (arXiv 2102.09649; CERN Courier; EPJC 2026). |
| **JPEG AI** (ISO/IEC 6048-1:2025 \| ITU-T T.840.1) — normative "me-tANS" (meta-tabled ANS) entropy coder, named as such in ByteDance's WO2025155739A1 / WO2025259668A1 confirmed | 2023 decision; 2025 published | ANS (range coder replaced, May 2023) | Second ISO image standard with ANS; no shipping decoders yet. |
| **LOLZ** (ProFrager) and, probably, **RAZOR** (Martelock) likely | 2017–18 | adaptive rANS + LZ/ROLZ | Standard compressors of the game-repack scene (recipes like xtool + srep + lolz); RAZOR's rANS rests on reconstructed source discussed on encode.su. |
| NVIDIA **Parabricks** bundles htscodecs confirmed | 2020s | rANS (CRAM) | Commercial GPU genomics suite; third-party notice credits the ryg_rans-derived code. |

### 6.2 GPU / parallel decoding and hardware

| Item | Year | Variant | Notes |
|---|---|---|---|
| multians (Weißenberger & Schmidt, ICPP 2019) | 2019 | tANS (CUDA) | First massively parallel decoding of unpartitioned ANS streams. |
| Meta dietGPU (archived Jun 2026); hipANS (ROCm port) | 2022 | rANS (GPU) | 250–600 GB/s on A100; float exponent codec. |
| Recoil (Lin et al., ICPP 2023) | 2023 | rANS | Decoder-adaptive parallel splits; 11+ GB/s CPU, 90+ GB/s GPU. |
| hypersonic-rANS (C. Stiller) | 2023 | rANS32/64 AVX-512 | 3.3 GB/s single-thread, 18 GB/s multi-thread. |
| Turbo-Range-Coder (powturbo) | 2020– | adaptive-CDF rANS | SSE/AVX2/NEON/RISC-V RVV. |
| "Approaching Shannon Bound with Lossless LLM Weight Compression" (NUS/HUST/ETH, arXiv 2606.15789) | 2026 | rANS | Tile-granular rANS decode inside tensor-core GEMM. |
| rANS split-computing feature compression (Kyungpook NU, arXiv 2511.11664) | 2025 | rANS (CUDA) | Sub-ms DNN intermediate-feature coding. |
| Najmabadi et al., FPGA tANS decoder vs canonical Huffman (JSPS 2018); IJET 2024 low-cost FPGA tANS encoder; AMD/Xilinx Vitis zstd kernels (FSE decode) | 2018–24 | tANS (FPGA) | Hardware implementations beyond consoles. |
| Patent literature on the coder itself: Huawei EP4637150A1 / HK40128054A (division- and modulo-free ANS encoder/decoder, priority Jan 2023); Inspur CN113572479B (FSE table generation for hardware zstd FSE, granted Dec 2021); "A FPGA-based FSE Accelerator with Dynamic Table for ZSTD compression" (IEEE, 2025) | 2021–25 | ANS / tANS (FSE) hardware designs | Coder-level IP and FPGA work in China; no shipped product documented. Note that shipped server zstd accelerators (Huawei Kunpeng KAEzstd, Intel QAT) offload only the LZ77 stage — FSE stays in software. |
| MANS / H5Z-MANS (Chinese Academy of Sciences, SC25) | 2025 | ANS for multi-byte integers | CPU, CUDA and ROCm back-ends; HDF5 filter plugin. |
| UCCL-Zip (UC Davis / Harvard / AWS / Berkeley) | 2026 | rANS (GPU, in NCCL kernels) | Compresses exponent fields inside collectives; up to 47.5 % faster RL weight sync. |
| torch_ans | 2025–26 | rANS (C++/CUDA) | PyTorch extension replacing range coders in learned-compression stacks. |
| Fast Compressed Segmentation Volumes (arXiv 2308.16619) | 2023 | rANS, static tables | GPU-friendly scientific volume visualisation. |
| RAS — bit-exact rANS accelerator for neural lossless compression (arXiv 2511.04684) | 2025 | rANS (RTL) | 121× encode / 71× decode vs a Python baseline in RTL simulation; hardware track to watch. |
| GS-NFS — bandwidth-adaptive streaming of dynamic Gaussian splats (arXiv 2606.05650, USC) | 2026 | rANS (nvCOMP) | Positions coded with nvCOMP's ANS; ANS entering neural rendering and telepresence. |
| Not ANS, for the record: AMD hipCOMP-core (ANS "not supported"), NVIDIA Blackwell DE | 2025–26 | — | Frequently assumed to carry ANS; they do not. |

### 6.3 Machine learning and learned compression

| Item | Year | Variant | Notes |
|---|---|---|---|
| BB-ANS (Townsend, Bird, Barber; ICLR 2019); Bit-Swap (Kingma et al.); HiLLoC; craystack; multiset compression (FAIR); McBits | 2019–21 | rANS (bits-back) | The bits-back-with-ANS research line. |
| CompressAI (InterDigital) | 2020 | rANS (ryg_rans) | De-facto learned-image-compression toolkit; ~11k PyPI downloads/month. |
| constriction (Bamler; Rust + Python + WASM) | 2021– | ANS + range + chain | arXiv 2201.01741; ~4k PyPI downloads/month. |
| YAECL (T. Xu) | 2022 | rANS + AC | pybind11 entropy-coding library for neural compression. |
| NeuZip (Borealis AI / U. Alberta; Hao et al., 2024) | 2024 | ANS (nvCOMP) | Lossless exponent compression during training. |
| Token-native storage (K. Shivendu, arXiv 2608.02376) | 2026 | static unigram ANS (constriction) | 3.30× over UTF-8 on English text stored as BPE token IDs; recommends frequency-ranked vocabularies so streamvbyte recovers most of the gain at ~7× faster decode. |
| NVIDIA gANS checkpoint compression guidance (developer blog, Apr 2026) | 2026 | rANS (GPU) | Vendor recommendation with a quantified case (782 GB checkpoints, 1.13 PB/month, ~530 GB/s). |
| KVTC open implementation (rANS back-end) | 2026 | rANS | KV-cache compression with best-of(zlib, lzma, rANS). |
| Huawei PILC (CVPR 2022) and OSOA (2021); Alibaba/DAMO ANS-LIC (2025) | 2021–25 | modified ANS / bb-ANS / ANS hardware | Corporate research on learned lossless image coding and ANS hardware for learned codecs; no deployment evidence. |
| Not ANS, for the "confused with" note: ZipNN, DFloat11, Huff-LLM, Cloudflare Unweight (Huffman); ZipServ (bitmaps); TDT (transform; back-end dependent); TensorFlow Compression (range coder) | 2024–26 | — | ZipNN's paper explicitly benchmarks and rejects FSE. |

### 6.4 Language libraries and new codecs

| Item | Year | Variant | Notes |
|---|---|---|---|
| Rust: rans-rs (m4tx), ryg-rans-sys, ryg-rans-rs workspace (2026), ans crate with bits-back (2026), zune-entropy (tANS), webgraph-ans-rs (graph compression), mzc (tANS archiver, 2026) | 2021–26 | rANS / tANS | Rust ecosystem largely absent from older lists. |

| Item | Year | Variant | Notes |
|---|---|---|---|
| Pure zstd/FSE re-implementations: klauspost/compress (Go), ruzstd (Rust), ZstdSharp (C#), aircompressor (Java), fzstd (JS), Zig std | 2018–23 | tANS (FSE) | Independent FSE decoders in six languages. |
| Kanzi (Java/Go/C++, F. Langlet) | 2013– | rANS (ANS0/ANS1) | Ports of ryg_rans into a full compressor. |
| Python: craystack, py-rans / Python-rANSCoder, Stanford Compression Library (rANS + tANS), Townsend's rANS tutorial (arXiv 2001.09186) | 2019–22 | rANS / tANS | Teaching and research implementations. |
| Go: nathanhack/asymmetricnumeralsystems; Java: LiMium/libANS, tomfran ANS-Graph-compression | 2018–24 | rANS / ABS | |
| Falcon audio codec (pbtechlab, Rust); @dotprotocol/compression (npm); marc (C++20) | 2025–26 | rANS | New codecs claiming rANS entropy stages. |
| Historical: fpaqa/fpaqb/fpaqc (Mahoney, 2007–08, first ABS compressors); rans-tricks (loxxous, 2018); FSC (Massimino, 2015) | 2007–18 | ABS / rANS / tANS | If not already present. |

Items that are probably already on the list (FSE, zhuff, zstd, LZFSE, lzturbo/TurboANX, Oodle LZNA/BitKnit, CRAM rANS, Draco, PIK, Brunsli, JPEG XL, ryg_rans, cbloom's posts, Duda's toolkit) are omitted here; the full catalogue with URLs is Appendix C.

# 7. Comparison with the independent Sol/Kimi study and response to the reviews

Two external inputs shaped this version. First, an independent study of the same question ("ANS adoption and economic impact", Sol, reconciled with a Kimi Agent analysis, 2 September 2026) was produced without contact with this work and with different conventions. Second, Sol reviewed version 1.1 of this report against primary sources. This section records agreement, disagreement and what changed.

## 7.1 Where the two studies agree

- **Ranking and tiers.** Both find Zstandard the only global-scale ANS carrier; Apple LZFSE and Oodle platform-scale; JPEG XL, Draco, BCPack and CRAM domain-scale; nvCOMP gANS the only commercial AI path with a quantified case; dietGPU, NeuZip and fused ANS–GEMM inference research-stage.
- **The corrections.** Both separate Xbox BCPack (rANS) from PC DirectStorage GDeflate (Huffman); both exclude the PS5 hardware Kraken decoder from ANS execution; both treat DivANS as a Dropbox experiment (130,000 sampled chunks, none persisted); both list ZipNN and DFloat11 as Huffman. This report's BCPack evidence is stronger (Microsoft's GDK documentation states "compressed using a rANS algorithm"; the other study relies on the patent).
- **Reach.** zstd people ≈4.3 B (theirs) vs ≈4.5 B (ours); LZFSE endpoints 2.3 B vs 2.55 B; BCPack ≈35 M consoles in both; CRAM ≈150–200 k practitioners in both.
- **The central caveat.** Both insist that whole-codec savings cannot be attributed to ANS alone and report an "ANS-only" lens an order of magnitude below the system lens.

## 7.2 Where they differ, and how version 2.0 treats each point

| Item (2026) | Sol/Kimi base | This report, v2.0 central | Reason and treatment |
|---|---|---|---|
| Unit of "volume" | Logical input bytes | Compressed bytes written or transmitted | Definitions differ by the compression ratio (≈2–4× for zstd-class data, ≈10× Draco, ≈3× CRAM). Where the ratio is applied the Draco, CRAM and GPU/AI volumes agree within a factor of two. |
| zstd endpoints | 8.0 B incl. logical machines | 6.1 B physical devices | Definitional; physical devices kept because the Apple, console and browser anchors are physical. |
| zstd volume | 100 EB logical (≈35–40 EB compressed) | 174 EB compressed [×0.4–×2] | The other study does not itemise **Steam**. Version 1.1 put Steam at 75 EB (70 % zstd); the review pointed out that only new and updated chunks are zstd and no share is measured, so v2.0 uses 40 % (≈55 EB) as central and keeps 70 % as the high scenario. Data-platform volumes were also lowered (100 → 56 EB across durable and transient components) and labelled as modelled. |
| zstd avoided fraction | 8 % vs mixed predecessors | 15 % (storage), 5 % (web), 0 % (Steam) | Predecessor mix: Kafka (Snappy → zstd, 2.5× → 4.3×), MongoDB (−49 % vs −28 %) and Elasticsearch (−12 % vs DEFLATE) support 15–30 % for storage systems; Cloudflare's 11.3 % vs gzip supports the web figure. |
| LZFSE volume | 7 EB logical | 9 EB compressed [×0.2–×1.6] | This report anchors on OS update payloads; the review noted that the OTA algorithm mix is not public, so the figure is labelled modelled and lowered from 13 EB. |
| Oodle | 330 M endpoints; 6 EB; 18 % avoided | 240 M; 1.5 EB (ANS modes only); 2 % | The other study counts the software-Oodle ecosystem as "ANS-capable"; this report counts only the estimated tANS/rANS-mode share, following Giesen's "rarely used". |
| JPEG XL 2026 endpoints | 1.2 B | 2.5 B | Apple's installed base on iOS 17+/macOS 14+ alone is ≈2.2 B; kept. Firefox (29 Sept 2026) and Chrome (no milestone) are now modelled separately, as the review asked. |
| JPEG XL 2030 volume | 5 EB logical | 14 EB compressed (was 40 in v1.0, 20 in v1.1) | Chrome-default scenarios in both; this report's ramp was moderated twice, guided by WebP's and AVIF's adoption curves. |
| GPU/AI | 50 k GPUs; 0.3 EB; 2030: 600 k, 6 EB | 60 k GPUs; 0.1 EB illustrative [0.01–0.4]; 2030: 5 EB | Same anchor (NVIDIA's case). The review asked for current volume to be marked unknown; v2.0 does so and adds the emerging-application table (Section 3.8). |
| ALICE $O^2$ | 350 servers; 0.15 EB logical; 0.105 avoided | 350 servers; ≈0.14 EB/yr compressed 2024–25, 0.08 in 2026 | An omission in v1.0, added in v1.1. Anchored here on the 423 PB stored and the LHC schedule. |
| LOLZ / RAZOR | 0.02 EB encoding, excluded | Excluded (0.1–3 EB/yr plausible) | v1.1 carried 0.9 EB/yr of transmitted bytes from site-visit proxies; the review judged the proxy too weak, and v2.0 excludes it from totals. |
| Financial 2026 | A (ANS-only) $20–50 M; B1 ≈$0.10 B; B2 ≈$0.56 B | Whole-codec $1.8B [$0.5B–$7.3B]; ANS-stage $255M; high scenario $2.9B | Version 1.1's $2.5 B compounded all saved bytes at 85 % retention. The review's central objection — that transfer flows, logs, compactions and checkpoints are not retained storage — is accepted: v2.0 charges storage on a resident footprint with retention set per component (80 % durable, 10 % transient, 5 % checkpoints), which halves the figure; the remaining gap to B2 is volume (Steam, platforms) and price ($60 vs ≈$120/TB-yr). The reviewer's "safer public headline" of $0.3–0.8 B corresponds to counting only measured or default-on flows with one-year charging; it sits at the low end of this report's range. |
| Financial 2030 | B2 ≈$1.9 B; C (AI upside) $0.5–3 B | Whole-codec $5.5B; ANS-stage $851M; high $9.6B | Same causes; the AI "enabling upside" lens is discussed qualitatively in Sections 4.6 and 5.2. |

## 7.3 The review of version 1.1 and what changed in version 2.0

| Review point | Response in v2.0 |
|---|---|
| "zstd is the default compressor of the Linux kernel" is incorrect; upstream defaults to gzip. | Accepted; corrected everywhere to "shipped in the kernel and selectable for images, modules, initramfs, SquashFS and Btrfs" (init/Kconfig: `default KERNEL_GZIP`). |
| 70 % of Steam's 100 EB on zstd is unsupported; treat as aggressive scenario. | Accepted; central 40 % (only new/updated chunks are zstd, first live chunk May 2025), 70 % kept as high scenario, range ×0.5–×1.8. Labelled "modelled from measured total". |
| Data-platform 70 EB rests on product support, not telemetry; split by platform, default and retention. | Accepted; split into durable (36 EB, 80 % retention) and transient (20 EB, 10 % retention) components with evidence labels; Meta split likewise; scientific/HPC carriers (RNTuple, Zarr, VBZ) added as a separate small component. |
| Apple 13 EB is a device-count extrapolation; move to plausible/high scenario. | Accepted in part: central lowered to 9 EB with range ×0.2–×1.6 and "modelled" label; 13 EB kept as high scenario. |
| GPU/AI 0.4 EB should be "unknown", 0.01–0.4 illustrative. | Accepted; central 0.1 EB illustrative, range 0.01–0.4, machines lowered to 60 k. |

| Review point | Response in v2.0 |
|---|---|
| Repacks 0.9 EB rests on a weak proxy; exclude from totals. | Accepted; excluded, retained as qualitative finding. |
| Apache Arrow is not a false positive: IPC supports optional zstd. | Accepted; reclassified as optional zstd carrier with no volume. |
| GS-NFS exists and uses nvCOMP ANS. | Accepted; the earlier "could not be located" was a fetch failure. Verified (USC; positions coded with nvCOMP ANS); the specific "0.3 ms / 2×" figures were not found in the text and are not quoted. |
| JPEG XL: model Firefox separately from Chrome; move Chrome bytes only after a stable milestone. | Accepted; Firefox 157 (29 Sept 2026) adds decoders, Chrome assumed mid-2027 as a scenario with a slower ramp (14 EB in 2030). |
| Annual writes are not retained storage; track logical bytes, compressed bytes, resident footprint and transferred bytes separately. | Accepted; the model now reports compressed bytes, resident saved footprint and transfer separately, with per-component retention; a logical-byte view is obtainable from the compression ratios in Section 7.2. |
| Emerging AI applications deserve their own section: checkpointing, token-native storage, UCCL-Zip, ANS-GEMM, split inference, RAS, GS-NFS; track three AI curves; FP8/FP4 headroom. | Accepted; Section 3.8 gained an emerging-applications table with verified figures (token-native storage: 2.25× uint16, 3.30× ANS, ~7× faster streamvbyte decode; RAS: 121×/71× RTL speed-ups), Section 5.2 tracks storage, communication and token-store curves and notes the FP8/FP4 caveat. |
| Missing zstd carriers: ROOT RNTuple, VBZ, GitHub Actions cache, bazel-remote, Zarr, OCI layers, restic, GDAL. | Accepted; all verified against documentation and added (Section 3.1) with evidence labels; RNTuple's zstd-5 default and VBZ's role in nanopore raw signal are the most significant. |
| Adopt evidence labels (Measured use / Default-on / Optional / Modeled / Research / Private). | Accepted; labels defined in Section 2.2, applied in the census (3.15), the component table (Appendix B) and throughout. |
| Publish model, CSV, uncertainty ranges; shortened URLs; tagged PDF. | Model, CSV tables (central and high scenarios), figures and the HTML edition are delivered with the PDF; shortened URLs were expanded; the PDF remains untagged (a limitation noted in Section 8). |
| Token-native storage: "up to 5.9× on Hindi" (as quoted in the review). | Not confirmed: the paper's public versions give 3.30× on English and ≈3.2× on Hindi with ANS; only the confirmed figures are used. |

### 7.4 Follow-up from the Kimi review (v2.1): patent literature, Chinese vendors and hardware

After version 2.0 was delivered, Kimi (the agent behind the other study) reviewed it and sent a "Version 2.1 revisions" pack (3 September 2026) proposing corrections, additions from a patent sweep of ~20 Chinese technology companies, and a re-centering of the data-platform volumes. Each proposal was checked against the primary document before being accepted, amended or declined; the outcome is below. The headline numbers are unchanged: the verified additions are qualitative (IP, carriers, hardware clarifications) and none supplies a measured volume.

| Proposal | Verification | Treatment in v2.1 |
|---|---|---|
| ByteDance holds patents on JPEG AI's ANS entropy stage (WO2025259668A1, WO2025155739A1, "Meta-Tabled Asymmetric Numeral" coding). | Both filings read. They concern JPEG AI codestream markers/substream types and HEIF storage; they state that codestream segments are "parsed with Meta-Tabled Asymmetric Numeral Systems (me-tANS) entropy coder". They are not patents on the coder. | Accepted in amended form: Section 3.11 now names me-tANS and records ByteDance's signalling/file-format IP around it; "holds patents on the coder" is not claimed. |
| Xiaomi: nine ANS patents on point-cloud geometry entropy coding (US12579696B2, US12531992B2, US12417555B2 granted). | All three read. Xiaomi's claims are for context-adaptive arithmetic coding of G-PCC-style geometry; US12579696B2 mentions ANS once as a generic alternative ("may be any other type of entropy coders like asymmetric numeral systems"); the other two do not mention ANS at all. | Declined as ANS evidence; recorded in Section 3.13 as boilerplate. |
| Huawei "optical ANS": EP4637150A1 / HK40128054A and US 12,676,782 on ANS-based probabilistic shaping in coherent optical transmission. | EP4637150A1 / HK40128054A (priority 9 Jan 2023) are real and describe a division- and modulo-free ANS encoder/decoder, with no optical, shaping or distribution-matcher content in the text. "US 12,676,782" and its quoted title do not resolve on Google Patents or in a web search. Two other Huawei filings from the pack's "15" do put ANS in the claims and were verified: CN118202389A (point-cloud coding) and CN119497858A (NN-weight compression). | The Huawei ANS-coder family and the two claim-level filings are added (Sections 3.12, 6.2, 9); the optical distribution-matcher application is not, pending a citable document. ANS-based distribution matching is a known research topic, but no Huawei product use is documented. |
| Second "ANS in silicon": Huawei Kunpeng 920 KAEzstd implements zstd's LZ77 + FSE pipeline in hardware; change the "1 in silicon" headline to 2. | Huawei's KAE documentation: KAEzstd "uses Kunpeng hardware acceleration to implement the lz77_zstd algorithm", supports compression only ("decompression uses software"), and no dictionary mode; the FSE/Huffman stage runs in the zstd library. Intel's QAT zstd plugin works the same way (external sequence producer). | Declined; the headline stands. Section 3.13 gains a row explaining that server zstd accelerators offload the LZ77 stage only. Inspur's CN113572479B (verified) shows FSE hardware being designed, but no shipped part is documented. |
| Inspur CN113572479B (FSE table generation); SenseTime's two ANS filings. | Inspur verified: "suitable for FSE compression and decompression realised by hardware under the zstd specification", granted 21 Dec 2021. SenseTime's CN114363615B and CN116437087A (from the pack) verified: both mention ANS only as one admissible entropy coder ("such as ANS … or arithmetic coding"). The neural-compression patent US12225205B2 is Tencent's and uses arithmetic coding. | Inspur added (3.12, 6.2); SenseTime noted as optional-mention only (3.12); the Tencent patent is recorded as a negative data point (3.13). |
| Tencent's TNC neural codec uses arithmetic coding; no Tencent ANS patents. | Consistent with US12225205B2 ("arithmetic decoding"); TNC itself was not inspected. | Accepted as `likely`; supports keeping Tencent under `Private`. |

| Proposal | Verification | Treatment in v2.1 |
|---|---|---|
| Re-center data-platform volumes (+17 % durable, +10 % transient) for Chinese platforms under-measured by Western telemetry (OceanBase, PolarDB-IMCI zstd default, ByteDance Lance lakes), lifting 2026 volume to ≈180 EB and cost avoided to ≈$1.65 B. | PolarDB IMCI's `imci_default_codec = ZSTD` verified; OceanBase's zstd default configuration for tables likely (its own write-up; the pack notes LZ4 as the log-archive default); the Lance/Volcano Engine "70 %" figure could not be read from the case study. No volume is published for any of them. The v2.0 platform components are modelled from global structured-data growth with an assumed zstd share, not from Western telemetry, so a regional uplift would double count unless the share assumption is shown to exclude China — which it does not. | Carriers added to Section 3.1 with Default-on labels; central values unchanged. The proposal sits well inside the published range (63–391 EB; $0.5–4.0 B) and is noted as an upside in Section 5.3. |
| Nits: LOLZ "nibble-style models" over-specifies; RAZOR's "✔/~" in Appendix C should be "~". | Agreed. | Both corrected (3.14, Appendix C). |
| Five entries for the encode.su list: Kunpeng KAEzstd; Inspur CN113572479B; ByteDance JPEG AI filings; Xiaomi point-cloud patents; Huawei optical ANS. | As above: two verified as stated (Inspur; ByteDance, with the amended reading), one verified but not ANS-in-silicon (Kunpeng), one not ANS evidence (Xiaomi), one unverified (optical). | Section 6 gains the Inspur and Huawei coder patents, the JPEG AI me-tANS naming, the FPGA FSE accelerator and the Chinese zstd carriers; Kunpeng and Intel QAT appear as an LZ77-only clarification. |
| Negative checks reported in the pack: Huawei Cloud CDN and Alibaba Cloud CDN offer gzip/Brotli only; Tencent Cloud zstd unconfirmed; Huawei HZU/HZBC storage compression undisclosed. | Not independently re-checked here. | Consistent with v2.0: no uplift to HTTP zstd; HZU/HZBC stays Private. |

### 7.5 Operator evidence: Yann Collet, Jon Sneyers, and the verification addendum (v2.2)

Version 2.1 was read by two people who build the codecs it describes, and both replied in public. Their statements were relayed by the client (encode.su blocks automated clients and the Discord channel requires an account), so they are recorded here as *operator statements*; everything derived from them was then verified independently against primary sources.

> **Yann Collet** (author of Zstandard and FSE; encode.su, 3 September 2026): "*Several of these numbers are way below reality. I do not blame the AI though, infrastructure numbers do not come public.*"
>
> **Jon Sneyers** (JPEG XL co-author, Cloudinary; JPEG XL Discord, September 2026): "*for Cloudinary: we are currently serving between 500m and 1b JXL images per day; I anticipate that by the end of the year (assuming Chrome starts rolling out enabled-by-default), it will probably be between 3 and 4 billion JXL images per day.*"

The two carry different evidentiary weight and version 2.2 treats them differently. Sneyers gives a count with a clear time basis and an explicit condition — and that condition, Chrome rolling out enabled-by-default, is now met (milestone 155, stable 6 October 2026) — so it can anchor a series. Collet's is the more important statement and the harder one to act on: it says *several* figures, not one, are *way* below reality, and it names the reason — the numbers that would settle the question are not published. That justifies restructuring the model, widening its ranges and hunting for missing carriers; it does not license inventing a byte counter for companies that publish none. The honest consequence, stated plainly here and in Section 8: **this report's central case is a public-evidence floor, not a best estimate of reality.** Where a component is unauditable — private warehouses, caches, replicas, RPC, internal CDN traffic — the truth is more likely to sit in the upper half of the published range than at its centre, and an operator's own view will sit higher still. Version 2.2 therefore publishes the operations curves and the replication term so that a reader who knows a fleet can locate it inside the range, rather than moving the central line to a number no source supports.

| Point | What verification found | Treatment in v2.2 |
|---|---|---|
| **Collet:** the zstd totals are too low. | Accepted as directionally right. Meta's own OpenZL paper states in Section 7 that "prior to OpenZL, Zstandard made up the vast majority of compression use at Meta", and names Nimble, Scribe and PyTorch checkpoints among its deployments — breadth confirmed, bytes not published. Notably OpenZL's own codec set is "Constant, Huffman, FSE, Bitpack", so the migration keeps a tANS coder in Meta's stack rather than removing one. | Structural rather than cosmetic: the report now separates economically countable output from encoder and decoder operations (Sections 2.3, 4.3.1), adds the replication term to the money model, widens the Meta and data-platform ranges to ×0.3–×5.0 and ×0.3–×4.0, and adds four carriers that were simply missing (below). 2026 output moves from 172 to ≈188 EB, with ≈224 EB of encoder and ≈639 EB of decoder operations alongside it. |
| **Sneyers:** Cloudinary serves 0.5–1 B JXL images/day, 3–4 B/day forecast. | Not independently verifiable (Discord), but consistent with Cloudinary's published scale and its role as a JPEG XL co-author. The byte implication is modest: 183–365 billion responses a year at 80–250 kB is ≈0.015–0.091 EB/yr. | Added as an operator anchor in Section 3.3, and used to make the point that requests, delivered bytes and unique encodes are three different quantities. It raises the reach narrative sharply and the byte total only slightly (0.1 EB in 2026). |
| **Chrome JPEG XL is outdated in v2.1.** | Confirmed and consequential. The ChromeStatus feature entry now carries shipping milestone **155** on desktop, Android, WebView and iOS; the Intent to Ship says "will ship enabled for all users"; Chromium Dash puts M155 stable on **6 October 2026**. Version 2.1's "assumed mid-2027" was wrong. | Corrected everywhere. JPEG XL decoders reach ≈5.2B devices by end-2026 and the byte ramp moves forward a year, to 18 EB in 2030. |
| **Publish three financial numbers, not one.** | Accepted; the single figure was doing the work of three. | Section 4.5 now reports public/auditable $1.8B, infrastructure-inclusive $4.6B and strict ANS-stage $255M for 2026, and Figure 9B plots all three. |
| **Several "default" labels are too strong:** Android zram, Zarr v3, Draco, Cloudflare. | All four confirmed as overstatements. `mmd.zram.recompression.enabled` defaults to `false` and first-pass reclaim uses lz4; the Zarr v3 core spec lists Blosc, Bytes, CRC32C, Gzip, Sharding and Transpose — zstd is zarr-python's implementation default, not a spec mandate; KHR_draco_mesh_compression is an optional glTF extension; Cloudflare's zstd default applies to the Free plan, with Brotli on Pro/Business and gzip on Enterprise. | All four downgraded in Sections 3.1, 3.7 and 3.15 and in Appendix B. The CRAM wording is likewise narrowed to "the default CRAM version when CRAM output is selected". |
| **DirectStorage 1.4 adds optional zstd.** | Confirmed: Microsoft's DirectX blog, 11 March 2026, public preview, "CPU and GPU decompression", plus the GACL conditioning library (up to 50 % better ratios) and promised GPU-vendor optimisations later in 2026. | Added to Section 3.5 as a correction to "the PC path is Huffman-only", to the census as an `Optional capability` with no volume, and to Section 5.2 as a swing factor. |
| **Missing carriers.** | Four found and verified, all absent from v2.1: Discord's gateway (zlib → streaming zstd, April 2024, ratio 6 → 10, ~40 % less WebSocket traffic); Fedora's Btrfs root mounted `compress=zstd:1` by default since Fedora 34, measured at ~40 % of disk saved; Box's ZSTD transfers for all customers; and Cloudflare's cache-transcoding prototype (1 Sep 2026, zstd-3, 2.834× on the 22.3 % of cached bytes that arrive uncompressed). | Discord and Fedora enter the model as two new components (0.1 and 5.0 EB in 2026); Box and the Cloudflare prototype are recorded in Section 3.1 without volume. |
| **Elasticsearch's zstd claim was queried** during this round. | Re-checked and upheld against the primary source: elastic/elasticsearch PR #112665 (merged 13 September 2024, shipped in 8.16.0 and 9.0.0) removed the feature flag so that zstd is used when `index.codec` is `best_compression`, for "lower storage usage (upto ~12%) and higher indexing throughput (upto 14%) compared to DEFLATE". Separately, *OpenSearch* 2.9+ and Amazon OpenSearch Serverless offer `zstd`/`zstd_no_dict` as opt-in codecs beside an LZ4 default (26–32 % smaller indices in AWS's benchmark). | Both now stated, with their different statuses: Elasticsearch a shipped non-default setting, OpenSearch an opt-in codec. |
| **"Twelve years after the 2013 ANS preprint."** | Imprecise: the modern preprint is arXiv:1311.2540 (2013), the journal version 2014, and the underlying ideas are older. | Reworded to date from the 2013 preprint. |

| Point | What verification found | Treatment in v2.2 |
|---|---|---|
| **Sol's revised central values** (≈343 EB output, 685 EB encoder ops, 4.8 ZB decoder ops; $1.8 B / $4.5 B / $0.38 B). | Same structure, independently parameterised. This model's own multipliers give ≈188 EB, ≈224 EB and ≈639 EB, and the money lenses land at $1.8B, $4.6B and $255M — within a factor of two on volume and remarkably close on money, from different assumptions. | Not copied. Where the two differ, this report keeps the more conservative operations multipliers and says so; Sol's higher decoder figure is a reasonable upper case for read-heavy private fleets. |

One recommendation is only partly met. The addendum notes that the delivered package should include the model, the CSV tables and the figures; those were delivered with version 2.0 and 2.1 and are delivered again here, but the PDF is still untagged and its chart notes are small — both remain open (Section 8).

Read together, the two studies and the review bracket the answer: ANS-coded traffic and storage in 2026 is between roughly 40 and 400 EB of compressed data a year depending on how Steam and data-platform defaults are counted; the associated cost avoided is between ≈$0.3 B and ≈$2.5 B a year with $1.8B as this report's central value, and the part attributable to the ANS coder itself between ≈$0.03 B and ≈$0.4 B. All three agree that the reach — six billion machines, four to five billion people — is no longer in doubt.

## 8. Limitations and open questions

1. **No vendor publishes bytes compressed with zstd.** The zstd volume is triangulated from one hard number (Steam: 100 EB/yr total — but its zstd share is modelled, since only new and updated depot chunks use zstd), one share statistic (Web Almanac: 12 % of CDN responses) and product defaults (Databricks, Iceberg, ClickHouse, RNTuple, Zarr). The durable data-platform component could plausibly be a third or 2.5× the central value; it drives most of the dollar total. A sample of Steam depot manifests and codec telemetry from one or two platform vendors would settle the two largest unknowns.
2. **Apple OTA inner algorithm.** AppleArchive is confirmed as the OTA container since iOS 14 and LZFSE is its default, but no source dumps the per-block algorithm field of shipped payloads; the 9 EB/yr is therefore modelled (range 2–14 EB). LZRAVEN's rANS design is a third-party reverse-engineering result validated against OS 27 betas; Apple's documentation does not name the coder.
3. **Xbox BCPack usage.** Microsoft confirms the hardware rANS mode but publishes neither console counts nor the share of titles using BCPack; both are third-party estimates. The link between BCPack and Microsoft's rANS patent is inferred.
4. **Oodle's tANS share** ("rarely used") is qualitative; the 3 % figure is a placeholder. The PS5 hardware decoder is reported to implement the pre-tANS Oodle 2.2 feature set (source not directly readable here).
5. **JPEG XL's future** hinges on Chrome shipping default-on; the Intent to Ship is approved but unscheduled. HTTP Archive has never measured JXL share because its crawler is Chromium.
6. **GPU/AI adoption** is unknown: NVIDIA's recommendation and case study establish capability and unit economics, not fleet volume. The 0.1 EB/yr is illustrative (range 0.01–0.4); communication and token-store applications carry no volume. dietGPU was never confirmed in production.
7. **Counterfactual fractions** for the ANS stage (2.5 % for zstd, 3 % for LZFSE, 15 % for CRAM, 5 % for JPEG XL, 8 % for BCPack, 20 % for GPU float codecs) come from published FSE-vs-Huffman micro-benchmarks and codec-version deltas, not from ablations of the shipped codecs. A direct experiment — re-encoding zstd sequence streams with Huffman on a large corpus — would tighten the most important number in Section 4.4.
8. **Unit costs** ($40–100/TB-year storage; $2–8/TB transfer) are blended list/wholesale figures; hyperscalers' internal costs are lower and consumer bandwidth costs far higher. Speed and energy effects are not monetised beyond the Fermi estimate in Section 4.6.
9. **ALICE** is anchored on the 423 PB stored, not on bytes through the rANS stage, and depends on the LHC schedule (Long Shutdown 3 from June 2026). The **repack scene** is described but excluded from totals: site visits are a weak proxy for downloaded bytes.
10. **Retention convention.** The dollar figures charge storage on the resident saved footprint, with retention set per component (80 %/yr durable tables, 10 %/yr logs and caches, 5 %/yr checkpoints, 95–97 %/yr archives). This is the biggest single modelling choice: the flat 85 % of version 1.1 gives $2.9B instead of $1.8B for 2026, and charging one year only gives roughly half the central figure. The model tracks resident footprint explicitly so that readers can substitute their own retention assumptions.
11. **Overlap.** Machine and user counts overlap across software (a Mac carries LZFSE, a JPEG XL decoder and zstd); byte volumes do not overlap materially (each byte is coded once, by one codec).
12. **The central case is a floor, not a best estimate.** Every component that rests on private infrastructure — data-platform and Meta volumes, caches, replicas, RPC, internal CDN traffic — is anchored on public evidence that does not exist at the necessary resolution. Yann Collet, who wrote the codec, states that several of the figures are "way below reality" and identifies the cause as infrastructure numbers not being public (Section 7.5). This report accepts that judgement, and readers should treat the central line accordingly: it is the number that can be defended from public sources, not the number an operator would recognise.
13. **Operations multipliers are coarse.** The encoder- and decoder-operation curves (Section 4.3.1) rest on per-component multipliers chosen from how the systems are known to work — LSM compaction, replication, cache fills, read patterns — not on measurements. A factor of two either way is easy to argue and warehouse read amplification could be much larger; Sol's independent addendum puts 2026 decoder operations near 4.8 ZB against this model's ≈639 EB.
14. **Two key statements could not be retrieved directly.** Yann Collet's encode.su post and Jon Sneyers's Discord message were supplied by the client; encode.su blocks automated clients and the Discord channel requires an account. Everything derived from them was verified independently, but the statements themselves are recorded as user-supplied.
15. **Accessibility and reproducibility.** The PDF is produced from HTML and is not a tagged PDF, and several chart notes and legends are small; the HTML edition, `model.py`, the yearly CSV tables and the figures are delivered alongside it so every number can be regenerated. Uncertainty ranges are multiplicative envelopes, not distributions.
16. **Not researched or not verifiable** in this pass: Samsung and automotive uses (Huawei's, ByteDance's and Inspur's patent filings are now covered in Sections 3.12 and 7.4, but their internal deployments remain undisclosed); Nintendo system software (Oodle lists Switch/Switch 2 as supported platforms, but no ANS use is documented); Sketchfab/Google Maps Draco use; Genomics England's current CRAM share; NCBI SRA's post-2021 size; Firefox monthly active users; the exact current contents of the encode.su thread.

Where a number in this report matters for a decision or a publication, the recommended next step is to replace the modelled anchor with a direct measurement — for zstd, a request to Meta, Cloudflare or Valve for aggregate volumes; for CRAM, ENA/EGA's format breakdown; for BCPack, Microsoft's GDK team; for JPEG XL, Chromium's new `JXLImage` use counter once the flag is on by default.

# 9. Sources

Grouped by the section they support. All were fetched between 1 and 2 September 2026 unless the item is a dated standard or paper.

### 3.1 Zstandard

- FSE announcement (Dec 2013): fastcompression.blogspot.com/2013/12/finite-state-entropy-new-breed-of.html
- Huffman revisited (Jul 2015): fastcompression.blogspot.com/2015/07/huffman-revisited-part-1.html
- FiniteStateEntropy README benchmarks: github.com/Cyan4973/FiniteStateEntropy
- zstd 1.0 blog: engineering.fb.com/2016/08/31/core-infra/smaller-and-faster-data-compression-with-zstandard/
- Meta zstd at scale (Dec 2018): engineering.fb.com/2018/12/19/core-infra/zstandard/
- RFC 8878: datatracker.ietf.org/doc/html/rfc8878
- zstd README/releases: github.com/facebook/zstd
- RocksDB HISTORY: raw.githubusercontent.com/facebook/rocksdb/v5.18.3/HISTORY.md
- Redshift ZSTD (Jan 2017): aws.amazon.com/about-aws/whats-new/2017/01/amazon-redshift-now-supports-the-zstandard-high-data-compression-encoding-and-two-new-aggregate-functions/
- Parquet ZSTD: github.com/apache/parquet-format/pull/70
- Linux 4.14/4.15/4.18/5.7/5.9/5.13/6.10: kernelnewbies.org; lwn.net/Articles/827605/
- OpenZFS 2.0: github.com/openzfs/zfs/releases/tag/zfs-2.0.0
- Fedora RPM zstd: fedoraproject.org/wiki/Changes/Switch_RPMs_to_zstd_compression; Btrfs compression: fedoraproject.org/wiki/Changes/BtrfsTransparentCompression
- Arch Linux: archlinux.org/news/now-using-zstandard-instead-of-xz-for-package-compression/
- Ubuntu: lists.ubuntu.com/archives/ubuntu-devel/2018-March/040211.html; phoronix.com/news/Ubuntu-21.10-Zstd-Debs; bugs.launchpad.net/bugs/1931725
- Kafka KIP-110: cwiki.apache.org/confluence/display/KAFKA/KIP-110%3A+Add+Codec+for+ZStandard+Compression
- MongoDB/Percona: percona.com/blog/compression-methods-in-mongodb-snappy-vs-zstd/
- MySQL 8.0.18: dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-18.html
- PostgreSQL 15/16: postgresql.org/docs/release/15.0/, /16.0/
- Cassandra 4.0 compression docs: cassandra.apache.org/doc/4.0/cassandra/operating/compression.html
- Elasticsearch zstd: github.com/elastic/elasticsearch/pull/112665
- Iceberg default zstd: github.com/apache/iceberg/pull/8593; issue #15236
- Databricks table properties: docs.databricks.com/aws/en/delta/table-properties
- Trino Delta zstd: github.com/trinodb/trino/pull/17426
- Athena compression: docs.aws.amazon.com/athena/latest/ug/compression-formats.html
- ClickHouse codecs: clickhouse.com/docs/reference/statements/create/table/codec
- containerd 1.5; Docker 23.0 release notes
- Blender zstd: archive.blender.org/lists/bf-committers/2021-August/051124.html
- Windows 11 libarchive: bleepingcomputer.com/news/microsoft/windows-11-adds-support-for-11-file-archives-including-7-zip-and-rar/
- Chrome zstd: groups.google.com/a/chromium.org/g/blink-dev/c/GFFOLCF12a4; chromereleases.googleblog.com/2024/03/stable-channel-update-for-desktop_19.html
- Firefox 126 notes: firefox.com/en-US/firefox/126.0/releasenotes/
- Safari 26.3: webkit.org/blog/17798/webkit-features-for-safari-26-3/
- Cloudflare: blog.cloudflare.com/new-standards/; blog.cloudflare.com/cache-transcoding/; developers.cloudflare.com/speed/optimization/content/compression/
- Web Almanac CDN 2024, 2025: almanac.httparchive.org/en/2024/cdn; /en/2025/cdn
- caniuse zstd: caniuse.com/mdn-http_headers_content-encoding_zstd
- StatCounter browser/OS share (Aug 2026): gs.statcounter.com
- Steam zstd: github.com/SteamRE/SteamKit/issues/1503, /1504; Steam 100 EB: pcgamer.com/gaming-industry/valve-says-steam-users-downloaded-100-exabytes-of-games-in-2025-and-are-averaging-274-petabytes-of-installs-and-updates-every-day/
- Python 3.14: docs.python.org/3/whatsnew/3.14.html
- OpenZL: arxiv.org/pdf/2510.03203v2
- SQL Server 2025 backup compression: learn.microsoft.com/en-us/sql/relational-databases/backup-restore/backup-compression-sql-server
- .NET 11 Zstandard: github.com/dotnet/runtime/issues/112075
- Android mmd / VABC: source.android.com/docs/core/perf/mmd; source.android.com/docs/core/ota/virtual_ab/implement
- PyPI/npm stats: pypistats.org/packages/zstandard; pepy.tech/projects/zstandard; api.npmjs.org

### 3.2 Apple

- Compression framework & algorithms: developer.apple.com/documentation/compression (compression_lzfse, compression_lzmesh, compression_lzraven, compression_algorithm)
- Apple Archive compression: developer.apple.com/documentation/applearchive/archivecompression
- lzfse source: github.com/lzfse/lzfse; InfoQ (Jul 2016): infoq.com/news/2016/07/apple-lzfse-lossless-opensource
- decmpfs types: pkg.go.dev/github.com/blacktop/go-apfs/types; github.com/Siguza/fscmp; github.com/saagarjha/unxip
- Kernel cache bvx2: github.com/sbingner/kerneldec/issues/1
- OTA payloads: theapplewiki.com/wiki/OTA_Updates; jhftss.github.io/The-Nightmare-of-Apple-OTA-Update/
- LZRAVEN RE: github.com/anat0m1a/liblzraven; github.com/blacktop/ipsw/issues/1247
- Installed base: apple.com/newsroom (Jan 2016, Feb 2023, Feb 2024, Jan 2025, Jan 2026 results); macrumors.com/2019/01/29/…; sec.gov 8-K Ex. 99.1 (Q1 FY26)

### 3.3 JPEG XL and media

- ISO/IEC 18181-1/-2/-3/-4: iso.org; history: jpegxl.info/about/history.html
- Sneyers, Alakuijala et al. (2025): arxiv.org/html/2506.05987v2
- libjxl effort ladder: github.com/libjxl/libjxl/blob/main/doc/encode_effort.md; encode.h
- Chromium removal (2022): theregister.com/2022/10/31/jpeg_xl_axed_chrome/
- WebKit Safari 17: webkit.org/blog/14445/webkit-features-in-safari-17-0/
- iPhone 16 Pro JXL: 9to5mac.com/2024/09/13/…; petapixel.com/2024/09/18/…
- DNG 1.7: en.wikipedia.org/wiki/Digital_Negative; Samsung: cloudinary.com/blog/samsung-now-supports-dng-1-7…
- DICOM Sup 232: dicom.nema.org/medical/dicom/supps/LB/sup232_lb_JPEGXL.pdf
- Windows extension: apps.microsoft.com/detail/9mzprth5c0tb; ghacks.net/2025/03/03/…
- PDF Association: theregister.com/2025/11/10/another_chance_for_jpeg_xl/
- Chromium reversal & jxl-rs: devclass.com/2025/11/24/…; issues.chromium.org/issues/462919304; github.com/chromium/chromium/commit/2b547e7…; phoronix.com/news/Chrome-145-Released
- Intents to Ship (24 Aug 2026): groups.google.com/a/chromium.org/g/blink-dev/c/-gDojQbDPRI; chromestatus.com/feature/5114042131808256; groups.google.com/a/mozilla.org/g/dev-platform/c/3YMV4MS34KA; bugzilla.mozilla.org/show_bug.cgi?id=2065096; phoronix.com/news/Firefox-JPEG-XL-2026-Plans
- Google OSS blog (Jun 2026): opensource.googleblog.com/2026/06/journey-to-jpeg-xl-how-open-source-experiments-shaped-the-future-of-image-coding.html
- caniuse.com/jpegxl; Web Almanac Media 2022/2024
- Cloudinary benchmarks: cloudinary.com/blog/jpeg-xl-and-the-pareto-front; cloudinary.com/blog/jpeg-xl-how-it-started-how-its-going
- Photos per year: photutorial.com/photos-statistics/; petapixel.com/2025/06/18/…
- Brunsli: github.com/google/brunsli; PIK: github.com/google/pik; WebP2: chromium.googlesource.com/codecs/libwebp2/
- AV1 coder: arxiv.org/pdf/2008.06091

### 3.4 CRAM

- Bonfield, CRAM 3.1 (Bioinformatics 2022): academic.oup.com/bioinformatics/article/38/6/1497/6499262; preprint biorxiv.org/content/10.1101/2021.09.15.460485v1
- Specs: samtools.github.io/hts-specs/CRAMv3.pdf; CRAMcodecs.pdf
- htscodecs NEWS: github.com/samtools/htscodecs; HTSlib 1.22: github.com/samtools/htslib/releases/tag/1.22
- Benchmarks: htslib.org/benchmarks/CRAM.html; sanger.ac.uk/tool/cram/
- GA4GH (2019): ga4gh.org/news_item/cram-compression-for-genomics/; guest post "seven myths"
- Bonfield, "Dispelling the CRAM myths" (2018): datageekdom.blogspot.com/2018/09/dispelling-cram-myths.html
- Emirati Genome Program: biorxiv.org/content/10.1101/2022.12.21.521516v1
- UK Biobank WGS: community.ukbiobank.ac.uk (500k WGS FAQs; Nov 2023 release post); sanger.ac.uk news (200k, 2021); genomeweb.com
- All of Us (Nature 2024): nature.com/articles/s41586-023-06957-x
- ENA 2023/2024/2025 NAR papers: academic.oup.com/nar (52/D1/D92; 53/D1/D49; 54/D1/D120); EGA: 50/D1/D980
- Genomics England (Weka case study 2020); NIH SRA Data WG report (2021); ncbiinsights.ncbi.nlm.nih.gov/2020/06/30/sra-rfi/
- DRAGEN CRAM: knowledge.illumina.com/…/000007021; GATK formats: gatk.broadinstitute.org/hc/en-us/articles/360035890791
- bioconda download counts: img.shields.io/conda/dn/bioconda/{samtools,htslib,pysam}.json

### 3.5 Microsoft

- GDK DirectStorage overview (BCPack rANS): learn.microsoft.com/en-us/gaming/gdk/_content/gc/system/overviews/directstorage/directstorage-overview
- Xbox Velocity Architecture (Jul 2020): news.xbox.com/en-us/2020/07/14/a-closer-look-at-xbox-velocity-architecture/
- DirectStorage 1.1 / GDeflate: devblogs.microsoft.com/directx/directstorage-1-1-now-available/; ietf.org/archive/id/draft-uralsky-gdeflate-00.html; github.com/microsoft/DirectStorage (GDeflate README)
- Patents: patents.google.com/patent/US11234023B2; US20220109891A1; WO2020263438A1 (family); theregister.com/2021/03/13/microsoft_ans_patent/; theregister.com/2022/02/17/microsoft_ans_patent/; wiki.endsoftwarepatents.org/wiki/Asymmetric_numeral_systems

### 3.6 Oodle / Sony / Epic

- Oodle history & platforms: radgametools.com/oodlehist.htm; radgametools.com/oodle.htm
- LZNA: cbloomrants.blogspot.com/2015/05/05-09-15-oodle-lzna.html; rANS in practice: fgiesen.wordpress.com/2015/12/21/rans-in-practice/
- Entropy coding in Oodle Data: fgiesen.wordpress.com/2021/07/09/…the-big-picture/; …/2021/08/30/…huffman-coding/; …/2023/10/29/…6-stream-huffman-decoders/; misconceptions: …/2024/08/08/oodle-kraken-etc-misconceptions/
- UE 4.27 release notes; UE Oodle Data docs: dev.epicgames.com/documentation/unreal-engine/oodle-data
- Epic acquires RAD (Jan 2021): epicgames.com/site/en-US/news/epic-acquires-rad-game-tools

- Road to PS5 (Cerny): developer-tech.com/news/2020/mar/18/…; PS5 units: vgchartz.com/article/467765/…; simulationdaily.com (Q1 FY2026); Nintendo/Xbox: sqmagazine.co.uk/console-market-share-statistics/
- Unreal share of Steam sales: creativebloq.com/3d/video-game-design/unreal-engine-dominates…

### 3.7 Draco

- opensource.googleblog.com/2017/01/introducing-draco-compression-for-3d.html; google.github.io/draco/spec/; github.com/google/draco (ans.h)
- Khronos KHR_draco_mesh_compression (Feb 2018) press release and spec; khronos.org/blog/gltf-now-and-next
- Cesium: cesium.com/blog/2018/04/09/draco-compression/; …/2019/02/26/draco-point-clouds/
- three.js DRACOLoader docs; gltf-transform.dev; Unity Draco package docs; Blender glTF manual
- npm downloads: api.npmjs.org/downloads/point/last-month/draco3d (and draco3dgltf, gltf-pipeline)
- Leadwerks Draco analysis (2021); glTF PR #1702 (meshopt comparison)

### 3.8 GPU and AI

- nvCOMP release notes/API: docs.nvidia.com/cuda/nvcomp/release_notes.html; c_api.html; gdeflate.html; nvcompdx
- NVIDIA checkpoint blog (Apr 2026): developer.nvidia.com/blog/cut-checkpoint-costs-with-about-30-lines-of-python-and-nvidia-nvcomp/
- Blackwell decompression engine: developer.nvidia.com/blog/speeding-up-data-decompression-with-nvcomp-and-the-nvidia-blackwell-decompression-engine/
- RAPIDS cuDF I/O: docs.rapids.ai/api/cudf/stable/user_guide/io/io/
- dietGPU: github.com/facebookresearch/dietgpu; hipANS: github.com/PAA-NCIC/hipANS
- multians: github.com/weissenberger/multians; Recoil: arxiv.org/abs/2306.12141
- NeuZip: arxiv.org/abs/2410.20650; DFloat11: arxiv.org/abs/2504.11651; ZipNN: arxiv.org/html/2411.05239v2; Huff-LLM: arxiv.org/html/2502.00922v1; ZipLLM: arxiv.org/abs/2505.06252
- Shannon-bound LLM weight compression: arxiv.org/abs/2606.15789; split-computing rANS: arxiv.org/abs/2511.11664; KVTC: github.com/OnlyTerp/kvtc
- DeepMind "Language Modeling Is Compression": arxiv.org/abs/2309.10668; ts_zip: bellard.org/ts_zip/
- CompressAI: github.com/InterDigitalInc/CompressAI; constriction: github.com/bamler-lab/constriction (arXiv 2201.01741); BB-ANS: arxiv.org/abs/1901.04866; craystack: github.com/j-towns/craystack
- Datasets: github.com/EleutherAI/the-pile; huggingface.co (SlimPajama, RedPajama-V2, Dolma, FineWeb READMEs); commoncrawl.org/get-started; Hugging Face Xet blogs
- PyPI stats: pypistats.org/packages/{nvidia-nvcomp-cu12,compressai,constriction,zipnn}

### 3.9–3.11 ALICE O², repack compressors, JPEG AI, and Section 7 comparison

- ALICE "Fast Entropy Coding for ALICE Run 3": arxiv.org/abs/2102.09649
- CERN Courier, "ALICE ups its game for sustainable computing" (350 EPN servers, 8 AMD GPUs each): cerncourier.com/a/alice-ups-its-game-for-sustainable-computing/
- ALICE Run-3 raw data 423 PB (EPJC 2026): link.springer.com/article/10.1140/epjc/s10052-026-15721-0
- CERN, LHC enters Long Shutdown 3 (29 Jun 2026): home.cern/cern-bids-farewell-to-the-lhc-and-enters-long-shutdown-3/
- LOLZ author page (ProFrager): krinkels.org/resources/lolz.264/; repack recipes: fileforums.com/archive/index.php/t-99554-p-8.html; RAZOR reconstructed source thread: encode.su/threads/3281-…; repack-site traffic: similarweb.com/website/fitgirl-repacks.site/
- PS5 Kraken hardware / Oodle 2.2 feature set (as cited by the Sol/Kimi report): encode.su/threads/3364-sony-PS5-comes-with-an-inbuild-quot-Kraken-compression-quot
- JPEG AI: jpeg.org/jpegai/; press release 11 May 2023 (range coder replaced by ANS): jpeg.org/items/20230511_press.html; overview paper: arxiv.org/abs/2510.13867
- Huawei PILC (CVPR 2022): openaccess.thecvf.com/content/CVPR2022/html/Kang_PILC_…; OSOA: arxiv.org/abs/2111.01662; Alibaba/DAMO ANS-LIC: ieeexplore.ieee.org/document/10992335
- NVIDIA Parabricks htscodecs notice: docs.nvidia.com/clara/parabricks/about-parabricks/third-party-software-notices/htscodecs
- MANS / H5Z-MANS: github.com/hpdps-group/MANS; UCCL-Zip: arxiv.org/abs/2604.17172; torch_ans: github.com/worldlife123/torch_ans; Fast Compressed Segmentation Volumes: arxiv.org/abs/2308.16619; AMD hipCOMP-core (ANS not supported): github.com/ROCm/hipCOMP-core
- Cloudflare Unweight (Huffman): blog.cloudflare.com/unweight-tensor-compression/; ZipServ: arxiv.org/abs/2603.17435; TDT: arxiv.org/abs/2506.18062
- Dropbox DivANS blog (130,000 chunks, none persisted): dropbox.tech/infrastructure/building-better-compression-together-with-divans
- RAD Granny history (BitKnit file mode, Jul 2015): radgametools.com/granny/history.html
- Meta Q4 2025 results (3.58 B daily active people): investor.atmeta.com; ITU Facts and Figures 2025 (6.0 B online): itu.int/en/ITU-D/Statistics/Pages/StatisticsUpdate/December2025.aspx
- Firefox release calendar (Firefox 157: 29 Sep 2026): whattrainisitnow.com/calendar/; Mozilla Hacks, Intent to ship JPEG XL (Aug 2026)
- Sol/Kimi, "ANS adoption and economic impact", 2 Sep 2026 (document supplied by J. Duda)

### 3.8 emerging AI and further zstd carriers (added in v2.0 after review)

- Token-native storage: arxiv.org/abs/2608.02376 (K. Shivendu, 3 Aug 2026); author's write-up: kshivendu.dev/blog/token-storage
- RAS rANS accelerator: arxiv.org/abs/2511.04684; GS-NFS: arxiv.org/abs/2606.05650 (html v1); UCCL-Zip: arxiv.org/abs/2604.17172; ANS-GEMM / Shannon bound: arxiv.org/abs/2606.15789; ISCA 2026 acceptance: systems.ethz.ch/news-and-events/news/2026/03/paper-on-llm-weight-compression-accepted-at-isca-2026.html; split inference: arxiv.org/abs/2511.11664
- NVIDIA checkpoint compression: developer.nvidia.com/blog/cut-checkpoint-costs-with-about-30-lines-of-python-and-nvidia-nvcomp/
- ROOT I/O compression (RNTuple zstd-5 default): root.cern/manual/io/; RNTuple binary format v1: root.cern/blog/rntuple-binary-format/; ATLAS adoption: cds.cern.ch/record/2924123
- Oxford Nanopore VBZ: github.com/nanoporetech/vbz_compression
- bazel-remote (zstd default): github.com/buchgr/bazel-remote; GitHub Actions cache: github.com/actions/cache; Zarr default codec: zarr.readthedocs.io/en/stable/user-guide/arrays/; OCI media types: github.com/opencontainers/image-spec/blob/main/media-types.md; restic format v2: restic.readthedocs.io/en/stable/100_references.html; GDAL GeoTIFF: gdal.org/en/stable/drivers/raster/gtiff.html; Arrow IPC compression: arrow.apache.org/docs/format/Columnar.html#compression
- Linux kernel compression default (KERNEL_GZIP): github.com/torvalds/linux/blob/master/init/Kconfig
- Sol, review of version 1.1 of this report (2 Sep 2026; document supplied by J. Duda)

### Section 7.4 — patent literature, Chinese platforms and hardware (added in v2.1)

- patents.google.com/patent/WO2025259668A1/en (ByteDance, "JPEG AI substream types and markers", priority 10 Jun 2024, published 18 Dec 2025; "parsed with Meta-Tabled Asymmetric Numeral Systems (me-tANS) entropy coder"); patents.google.com/patent/WO2025155739A1/en (ByteDance, JPEG AI images in HEIF, priority 17 Jan 2024, published 24 Jul 2025)
- patents.google.com/patent/EP4637150A1/en and patents.google.com/patent/HK40128054A/en (Huawei, "Data encoding method, data decoding method, and related apparatuses", priority 9 Jan 2023, EP publication 22 Oct 2025 — ANS-based encoder/decoder without division or modulo)
- patents.google.com/patent/CN118202389A/en (Huawei, learned point-cloud compression; claim 21 ANS entropy decoding; priority 28 Oct 2021); patents.google.com/patent/CN119497858A/en (Huawei, neural-network compression; claim 9 ANS; priority 29 Jul 2022); patents.google.com/patent/CN114363615B/en and CN116437087A (SenseTime; ANS as one admissible entropy coder)
- patents.google.com/patent/CN113572479B/en (Suzhou Inspur, "Method and system for generating finite state entropy coding table", filed 22 Sep 2021, granted 21 Dec 2021)
- patents.google.com/patent/US12579696B2/en, US12531992B2, US12417555B2 (Beijing Xiaomi Mobile Software; point-cloud geometry entropy coding — arithmetic coding, ANS mentioned once as a generic alternative)
- patents.google.com/patent/US12225205B2/en (Tencent America, "Block-wise entropy coding method in neural image compression", granted 11 Feb 2025 — arithmetic decoding)
- github.com/kunpengcompute/KAE (KAEZstd: "使用鲲鹏硬加速模块实现lz77_zstd算法"; compression only) and hikunpeng.com/document/detail/en/kunpengaccel/kae/kae/README_EN.md; github.com/kunpengcompute/KAEzip (gzip/zlib only); hikunpeng.com MySQL KAEZstd page-compression whitepaper; community.intel.com — Intel QAT Zstandard plugin as external sequence producer
- alibabacloud.com/help/en/polardb/polardb-for-mysql/user-guide/set-a-compression-algorithm ("ZSTD is the default algorithm (imci_default_codec = ZSTD)"); oceanbase.github.io/docs/blogs/showcases/compression-ratio (zlib/Snappy/zstd/LZ4; default configuration with zstd, ratio ≈4.6); lancedb.com/blog/volcano-engine-autonomous-driving-data-lake-solution (Volcano Engine LAS on Lance; compression figures quoted from the Kimi pack, page text not retrievable); Kimi Agent, "Entropy coding at planetary scale — Version 2.1 revisions" (3 Sep 2026, supplied by the client)
- ieeexplore.ieee.org/document/11044267 ("A FPGA-based FSE Accelerator with Dynamic Table for ZSTD compression", 2025; abstract not retrievable from the research environment — title only)

### Section 7.5 — operator evidence and v2.2 verifications

- chromestatus.com/feature/5114042131808256 and chromestatus.com/api/v0/features/5114042131808256 (JPEG XL decoding, shipping milestone 155 on desktop/Android/WebView/iOS, read 3 Sep 2026); chromiumdash.appspot.com/fetch_milestone_schedule?mstone=155 (M155 stable 6 Oct 2026, beta 16 Sep); blink-dev Intent to Ship, 24 Aug 2026 ("will ship enabled for all users", jxl-rs)
- Yann Collet, encode.su thread 2078, post 88913 (3 Sep 2026) — supplied by the client; the site rejects automated clients, so the statement is recorded as user-supplied
- Jon Sneyers, JPEG XL Discord (Sep 2026) — supplied by the client: Cloudinary serving 0.5–1 B JXL images/day, 3–4 B/day forecast by year-end
- arxiv.org/abs/2510.03203 (Meta, OpenZL): §7 "prior to OpenZL, Zstandard made up the vast majority of compression use at Meta"; §A.1 codecs "Constant, Huffman, FSE, Bitpack"
- devblogs.microsoft.com/directx/directstorage-1-4-release-adds-support-for-zstandard/ (11 Mar 2026, public preview, CPU and GPU decode, GACL)
- discord.com/blog/how-discord-reduced-websocket-traffic-by-40-percent (Apr 2024 rollout; ratio 6 → 10; 270 → 166 bytes)
- fedoraproject.org/wiki/Changes/BtrfsTransparentCompression and fedoraproject.org/wiki/Changes/BtrfsByDefault/CompressionLevelAnalysis (Fedora 34, `compress=zstd:1`, ~40 % saved, ~9 % more system time)
- developers.cloudflare.com/speed/optimization/content/compression/ (Free = Zstandard, Pro/Business = Brotli, Enterprise = Gzip); blog.cloudflare.com/cache-transcoding/ (1 Sep 2026; zstd-3; 2.834×; 22.3 % of cached bytes; prototype)
- source.android.com/docs/core/perf/mmd (`mmd.zram.recompression.algorithm = zstd`, `mmd.zram.recompression.enabled = false`); zarr-specs.readthedocs.io/en/latest/v3/codecs/ (core codecs: Blosc, Bytes, CRC32C, Gzip, Sharding, Transpose)
- docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-zstd-compression.html (LZ4 default; zstd/zstd_no_dict opt-in; 26–32 % smaller); support.box.com/hc/en-us/articles/33465731870227 (ZSTD transfers for all customers)
- Sol, "ANS adoption report verification addendum" (3 Sep 2026, supplied by the client)

### 3.12 Catalogue (Appendix C)

- en.wikipedia.org/wiki/Asymmetric_numeral_systems; github.com/topics/{asymmetric-numeral-systems,rans,tans,entropy-coding}
- mattmahoney.net/dc/ (fpaq); github.com/rygorous/ryg_rans; cbloomrants.blogspot.com/2014/02/02-18-14-understanding-ans-12.html; github.com/JarekDuda/AsymmetricNumeralSystemsToolkit
- sites.google.com/site/powturbo/entropy-coder; github.com/powturbo/Turbo-Range-Coder; github.com/flanglet/kanzi-cpp; github.com/inikep/lizard; github.com/dropbox/divans
- github.com/tirr-c/jxl-oxide; github.com/libjxl/jxl-rs; github.com/pbtechlab/Falcon; crates.io/crates/mzc
- github.com/samtools/htsjdk; github.com/GMOD/cram-js; github.com/zaeleus/noodles; github.com/divonlan/genozip; npmjs.com/package/@jkbonfield/htscodecs
- github.com/rainerzufalldererste/hypersonic-rANS; docs.amd.com (Vitis Data Compression Library); dl.acm.org/doi/10.1007/s11265-018-1421-4; ijet.pl (2024 FPGA tANS encoder)
- Rust/Python/Go/Java/JS libraries: github.com/m4tx/rans-rs; github.com/infinityabundance/ryg-rans-rs; github.com/arclabs561/ans; github.com/etemesi254/zune-entropy; github.com/ciminilorenzo/webgraph-ans-rs; github.com/KillingSpark/zstd-rs; github.com/klauspost/compress; github.com/oleg-st/ZstdSharp; github.com/airlift/aircompressor; github.com/101arrowz/fzstd; github.com/nathanhack/asymmetricnumeralsystems; github.com/LiMium/libANS; github.com/bits-back/bits-back; github.com/fhkingma/bitswap; github.com/kedartatwawadi/stanford_compression_library; github.com/FGlazov/Python-rANSCoder; github.com/tongdaxu/YAECL-Yet-Another-Entropy-Coding-Library

# Appendix A. Yearly tables

Central estimates. Years marked * are extrapolations. Machines and users overlap across rows and are not summed; volumes and savings are summed.

Table A1. Machines carrying an ANS decoder (installed base, end of year). * = extrapolation.

| | 2015 | 2016 | 2018 | 2020 | 2022 | 2024 | 2025 | 2026 | 2027* | 2028* | 2030* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| zstd | 1.0M | 20M | 350M | 1.4B | 2.7B | 5.0B | 5.6B | 6.1B | 6.4B | 6.7B | 7.1B |
| Apple LZFSE/LZRAVEN | 600M | 950M | 1.4B | 1.6B | 1.9B | 2.3B | 2.5B | 2.5B | 2.7B | 2.8B | 3.0B |
| JPEG XL | – | – | – | – | – | 1.7B | 2.1B | 5.2B | 6.5B | 6.9B | 7.4B |
| CRAM | 7.7k | 12k | 25k | 45k | 70k | 95k | 107k | 120k | 138k | 159k | 210k |
| Xbox BCPack | – | – | – | 3.5M | 21M | 31M | 34M | 36M | 37M | 38M | 41M |
| Oodle ANS modes | 5.0M | 20M | 50M | 80M | 130M | 190M | 220M | 240M | 252M | 265M | 292M |
| Draco | – | – | 12M | 46M | 95M | 161M | 200M | 220M | 246M | 276M | 346M |
| GPU/AI ANS | – | – | – | – | 2.0k | 10k | 25k | 60k | 114k | 217k | 782k |
| ALICE O² | – | – | – | – | 350 | 350 | 350 | 350 | 350 | 350 | 350 |

Table A2. People whose data is coded by each compressor (directly or via a backend). * = extrapolation.

| | 2015 | 2016 | 2018 | 2020 | 2022 | 2024 | 2025 | 2026 | 2027* | 2028* | 2030* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| zstd | – | 1.7B | 2.2B | 2.6B | 2.9B | 3.5B | 4.0B | 4.5B | 4.8B | 5.0B | 5.4B |
| Apple LZFSE/LZRAVEN | 450M | 700M | 900M | 1.1B | 1.2B | 1.4B | 1.4B | 1.5B | 1.5B | 1.6B | 1.7B |
| JPEG XL | – | – | – | – | – | 1.1B | 1.3B | 3.9B | 4.8B | 5.1B | 5.5B |
| CRAM | 4.5k | 10k | 30k | 60k | 100k | 150k | 173k | 200k | 230k | 264k | 350k |
| Xbox BCPack | – | – | – | 4.2M | 25M | 37M | 41M | 43M | 44M | 46M | 49M |
| Oodle ANS modes | 6.0M | 25M | 60M | 100M | 160M | 220M | 239M | 260M | 273M | 287M | 316M |
| Draco | – | – | 12M | 46M | 95M | 161M | 200M | 220M | 246M | 276M | 346M |
| GPU/AI ANS | – | – | – | – | 200 | 2.0k | 5.0k | 10k | 15k | 22k | 51k |
| ALICE O² | – | – | – | – | 2.5k | 2.5k | 2.5k | 2.5k | 2.6k | 2.7k | 2.8k |

Table A3. Compressed data written or transmitted per year in ANS-coded formats (EB), central scenario. * = extrapolation.

| | 2015 | 2016 | 2018 | 2020 | 2022 | 2024 | 2025 | 2026 | 2027* | 2028* | 2030* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| zstd | – | 0.500 | 4.65 | 14.0 | 31.7 | 62.2 | 101 | 174 | 234 | 298 | 439 |
| Apple LZFSE/LZRAVEN | 1.60 | 2.50 | 4.20 | 5.60 | 7.00 | 8.30 | 8.70 | 9.00 | 9.40 | 9.83 | 10.7 |
| JPEG XL | – | – | – | – | – | 0.020 | 0.100 | 0.150 | 1.50 | 5.00 | 18.0 |
| CRAM | 0.0010 | 0.0020 | 0.0060 | 0.014 | 0.028 | 0.045 | 0.058 | 0.075 | 0.096 | 0.123 | 0.201 |
| Xbox BCPack | – | – | – | 0.100 | 1.40 | 2.10 | 2.30 | 2.50 | 2.58 | 2.65 | 2.81 |
| Oodle ANS modes | 0.0050 | 0.020 | 0.100 | 0.300 | 0.700 | 1.10 | 1.30 | 1.50 | 1.60 | 1.72 | 1.97 |
| Draco | – | – | 0.0002 | 0.0010 | 0.0021 | 0.0038 | 0.0048 | 0.0060 | 0.0075 | 0.0094 | 0.015 |
| GPU/AI ANS | – | – | – | – | 0.0010 | 0.010 | 0.030 | 0.100 | 0.400 | 1.20 | 5.00 |
| ALICE O² | – | – | – | – | 0.050 | 0.140 | 0.150 | 0.080 | 0.010 | 0.010 | 0.250 |
| **Total** | **1.61** | **3.02** | **8.96** | **20.0** | **40.9** | **73.9** | **113** | **188** | **250** | **319** | **478** |

Table A4. Whole-codec savings per year vs. the codec replaced (EB). * = extrapolation.

| | 2015 | 2016 | 2018 | 2020 | 2022 | 2024 | 2025 | 2026 | 2027* | 2028* | 2030* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| zstd | – | 0.088 | 0.850 | 2.70 | 6.27 | 11.7 | 14.7 | 18.4 | 23.2 | 29.1 | 44.5 |
| Apple LZFSE/LZRAVEN | 0.049 | 0.077 | 0.130 | 0.173 | 0.216 | 0.257 | 0.269 | 0.278 | 0.291 | 0.304 | 0.332 |
| JPEG XL | – | – | – | – | – | 0.011 | 0.054 | 0.081 | 0.808 | 2.69 | 9.69 |
| CRAM | 0.0010 | 0.0020 | 0.0060 | 0.014 | 0.028 | 0.045 | 0.058 | 0.075 | 0.096 | 0.123 | 0.201 |
| Xbox BCPack | – | – | – | 0.054 | 0.754 | 1.13 | 1.24 | 1.35 | 1.39 | 1.43 | 1.52 |
| Oodle ANS modes | 0.0001 | 0.0004 | 0.0020 | 0.0061 | 0.014 | 0.022 | 0.027 | 0.031 | 0.033 | 0.035 | 0.040 |
| Draco | – | – | 0.0003 | 0.0015 | 0.0031 | 0.0057 | 0.0071 | 0.0090 | 0.011 | 0.014 | 0.022 |
| GPU/AI ANS | – | – | – | – | 0.0003 | 0.0033 | 0.010 | 0.033 | 0.133 | 0.400 | 1.67 |
| ALICE O² | – | – | – | – | 0.0088 | 0.025 | 0.026 | 0.014 | 0.0018 | 0.0018 | 0.044 |
| **Total** | **0.051** | **0.168** | **0.988** | **2.95** | **7.30** | **13.2** | **16.4** | **20.3** | **26.0** | **34.1** | **58.0** |

Table A5. Savings attributable to the ANS entropy stage alone (EB per year). * = extrapolation.

| | 2015 | 2016 | 2018 | 2020 | 2022 | 2024 | 2025 | 2026 | 2027* | 2028* | 2030* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| zstd | – | 0.013 | 0.121 | 0.369 | 0.843 | 1.63 | 2.19 | 2.96 | 3.89 | 5.04 | 7.74 |
| Apple LZFSE/LZRAVEN | 0.049 | 0.077 | 0.130 | 0.173 | 0.216 | 0.257 | 0.269 | 0.278 | 0.291 | 0.304 | 0.332 |
| JPEG XL | – | – | – | – | – | 0.0011 | 0.0053 | 0.0079 | 0.079 | 0.263 | 0.947 |
| CRAM | 0.0002 | 0.0004 | 0.0011 | 0.0025 | 0.0049 | 0.0079 | 0.010 | 0.013 | 0.017 | 0.022 | 0.036 |
| Xbox BCPack | – | – | – | 0.0087 | 0.122 | 0.183 | 0.200 | 0.217 | 0.224 | 0.231 | 0.245 |
| Oodle ANS modes | 0.0001 | 0.0004 | 0.0020 | 0.0061 | 0.014 | 0.022 | 0.027 | 0.031 | 0.033 | 0.035 | 0.040 |
| Draco | – | – | <0.0001 | <0.0001 | 0.0001 | 0.0002 | 0.0003 | 0.0003 | 0.0004 | 0.0005 | 0.0008 |
| GPU/AI ANS | – | – | – | – | 0.0003 | 0.0025 | 0.0075 | 0.025 | 0.100 | 0.300 | 1.25 |
| ALICE O² | – | – | – | – | 0.0088 | 0.025 | 0.026 | 0.014 | 0.0018 | 0.0018 | 0.044 |
| **Total** | **0.050** | **0.091** | **0.254** | **0.559** | **1.21** | **2.13** | **2.73** | **3.54** | **4.64** | **6.19** | **10.6** |

Table A6. Cost avoided per year, whole-codec (USD, central). * = extrapolation.

| | 2015 | 2016 | 2018 | 2020 | 2022 | 2024 | 2025 | 2026 | 2027* | 2028* | 2030* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| zstd | – | $4M | $48M | $183M | $496M | $1.0B | $1.4B | $1.7B | $2.2B | $2.8B | $4.3B |
| Apple LZFSE/LZRAVEN | $247k | $387k | $649k | $866k | $1M | $1M | $1M | $1M | $1M | $2M | $2M |
| JPEG XL | – | – | – | – | – | $607k | $3M | $7M | $50M | $190M | $941M |
| CRAM | $148k | $340k | $1M | $4M | $8M | $15M | $20M | $27M | $35M | $45M | $76M |
| Xbox BCPack | – | – | – | $162k | $2M | $3M | $4M | $4M | $4M | $4M | $5M |
| Oodle ANS modes | $306 | $1k | $6k | $18k | $43k | $67k | $80k | $92k | $98k | $105k | $120k |
| Draco | – | – | $3k | $12k | $25k | $45k | $57k | $72k | $90k | $112k | $176k |
| GPU/AI ANS | – | – | – | – | $33k | $338k | $1M | $3M | $14M | $41M | $171M |
| ALICE O² | – | – | – | – | $265k | $2M | $2M | $3M | $3M | $3M | $4M |
| **Total** | **$395k** | **$4M** | **$50M** | **$187M** | **$508M** | **$1.1B** | **$1.4B** | **$1.8B** | **$2.3B** | **$3.1B** | **$5.5B** |

Table A7. Cost avoided per year attributable to the ANS entropy stage alone (USD, central). * = extrapolation.

| | 2015 | 2016 | 2018 | 2020 | 2022 | 2024 | 2025 | 2026 | 2027* | 2028* | 2030* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| zstd | – | $513k | $7M | $25M | $68M | $143M | $190M | $242M | $307M | $388M | $611M |
| Apple LZFSE/LZRAVEN | $247k | $387k | $649k | $866k | $1M | $1M | $1M | $1M | $1M | $2M | $2M |
| JPEG XL | – | – | – | – | – | $59k | $337k | $726k | $5M | $19M | $92M |
| CRAM | $26k | $60k | $227k | $620k | $1M | $3M | $4M | $5M | $6M | $8M | $13M |
| Xbox BCPack | – | – | – | $26k | $365k | $548k | $600k | $652k | $672k | $692k | $734k |
| Oodle ANS modes | $306 | $1k | $6k | $18k | $43k | $67k | $80k | $92k | $98k | $105k | $120k |
| Draco | – | – | $91 | $421 | $876 | $2k | $2k | $3k | $3k | $4k | $6k |
| GPU/AI ANS | – | – | – | – | $25k | $254k | $763k | $3M | $10M | $31M | $128M |
| ALICE O² | – | – | – | – | $265k | $2M | $2M | $3M | $3M | $3M | $4M |
| **Total** | **$274k** | **$961k** | **$8M** | **$27M** | **$71M** | **$150M** | **$199M** | **$255M** | **$334M** | **$450M** | **$851M** |

## Appendix B. Model assumptions

The model (delivered as `model.py` with the report) represents each software family as one or more components, each with an evidence class. For each component the compressed volume series is anchored on the evidence in Section 3 and interpolated geometrically between anchor years; after the last anchor it grows at the stated rate. Savings are computed as counterfactual size minus actual size, where counterfactual size = volume ÷ (1 − s) for saving fraction s. Stored savings accumulate into a resident footprint $R_t = R_{t-1}\cdot(\text{retention}) + \text{saved}_t$ and are charged per TB-year; transferred savings are charged once per TB. Ranges are multiplicative bounds on each component's volume and are combined additively for the totals' bands. The high scenario (ANS_SCENARIO=high) restores the version-1.1 assumptions: Steam ~75 % zstd in 2026, data-platform 70 EB and Meta 20 EB with 85 % retention, Apple 13 EB, GPU/AI 0.4 EB, JPEG XL 26 EB in 2030.

**Added in version 2.2.** Each component also carries three multipliers, listed in `model.py` as 0PS: *enc*, encoder-output operations per unit of counted output (3.0 for durable platform data, from LSM compaction plus replicas; 0.02 for a Steam depot chunk encoded once and delivered to many users; 0.001 for an Apple OS build); *dec*, decoder operations per unit of output (8–10 for warehouse tables, 6 for compressed filesystems, 1 for delivery paths, 0.3 for checkpoints, which are written far more often than read); and *repl*, the number of stored copies an avoided byte is avoided in (1 for the auditable case; 3 for durable platform and Meta components, 2 for transient and CRAM, 1.5 for scientific archives). The operations multipliers feed Figure 9A only and never the money model; *repl* feeds the infrastructure-inclusive lens of Figure 9B. All three are judgement calls about how the systems work, not measurements, and are the least defensible numbers in the model.

Table B1. Model components (central scenario): volumes, saving fractions, cost basis, retention, evidence class and uncertainty ranges.

| Component | EB/yr 2026 | EB/yr 2030* | Whole-codec saving | ANS-stage saving | Cost basis | Range | Evidence class | Counterfactual |
|---|---|---|---|---|---|---|---|---|
| Steam game delivery: share of chunks re-encoded LZMA -> zstd since Feb-May 2025 (total volume measured by Valve; zstd share modelled) | 55.0 | 118 | 0% | 0.0% | transfer, $2/TB | ×0.7–×1.9 | Modeled from measured total | LZMA (same size; benefit is decode speed) |
| HTTP Content-Encoding: zstd (Meta, Cloudflare, Google CDN...) | 35.0 | 127 | 5% | 2.0% | transfer, $8/TB | ×0.4–×2.5 | Modeled from measured share (Web Almanac) | gzip (Cloudflare: zstd 11.3 % smaller) or Brotli (~7 % smaller than zstd) |
| Data platforms, durable: analytics tables and database storage (Parquet/Delta/Iceberg/ORC, RocksDB-family DBs, MongoDB, Cassandra, ClickHouse, Elasticsearch/OpenSearch, Redshift) | 36.0 | 96.6 | 15% | 2.5% | storage, $60/TB-yr, retained 80%/yr | ×0.3–×4.0 | Default-on in several platforms; share modelled | gzip / Snappy / LZ4 defaults (Kafka 4.28x vs 2.5x; MongoDB -49 % vs -28 %; Elasticsearch 8.16+ best_compression zstd -12 % vs DEFLATE; OpenSearch zstd 26-32 % vs LZ4) |
| Data platforms, transient: streams, logs, caches, compactions, backups, CI/build caches (Kafka, log stores, GitHub Actions cache, Bazel remote cache, restic, OCI layers) | 20.0 | 48.8 | 15% | 2.5% | storage, $60/TB-yr, retained 10%/yr | ×0.3–×4.0 | Optional production capability; share modelled | Snappy / LZ4 / gzip |
| Meta internal, durable (warehouse tables, blob storage) | 10.0 | 15.7 | 15% | 2.5% | storage, $40/TB-yr, retained 80%/yr | ×0.3–×5.0 | Modeled/internal claim (OpenZL paper: 'Prior to OpenZL, Zstandard made up the vast majority of compression use at Meta') | zlib / LZ4 / Snappy previously used at Meta (2018: 'managed compression' 50 % better) |
| Meta internal, transient (logs, caches, message queues, RPC payloads written to disk) | 10.0 | 15.7 | 15% | 2.5% | storage, $40/TB-yr, retained 10%/yr | ×0.3–×5.0 | Modeled/internal claim | as above |
| Scientific & HPC data: ROOT RNTuple (zstd-5 default), Zarr (zstd via zarr-python default, not a spec mandate), HDF5/NetCDF filters, Oxford Nanopore VBZ (POD5), GDAL ZSTD | 1.10 | 4.23 | 20% | 2.5% | storage, $30/TB-yr, retained 90%/yr | ×0.3–×3 | Default-on (RNTuple, Zarr, VBZ); volumes modelled | zlib / LZ4 / gzip (VBZ >30 % smaller than gzip; RNTuple zstd vs zlib-1) |
| OS & package distribution (Fedora/Arch/Ubuntu packages, Android OTA, initramfs) | 2.00 | 3.50 | 3% | 2.0% | transfer, $5/TB | ×0.3–×3 | Default-on (distros); volumes modelled | xz (same size) or gzip (zstd ~15 % smaller) |
| Local and server filesystem / memory compression: Btrfs compress=zstd (Fedora default since F34, ~40 % saved), OpenZFS zstd datasets, SquashFS/EROFS images, Android zram recompression (opt-in) | 5.00 | 9.69 | 35% | 4.0% | storage, $30/TB-yr, retained 75%/yr | ×0.2–×4.0 | Default-on (Fedora Btrfs); volumes modelled | uncompressed filesystem (Fedora measured ~40 % saved at zstd:1) or gzip/LZ4 datasets |

| Component | EB/yr 2026 | EB/yr 2030* | Whole-codec saving | ANS-stage saving | Cost basis | Range | Evidence class | Counterfactual |
|---|---|---|---|---|---|---|---|---|
| Real-time messaging and RPC streams: Discord gateway (zstd streaming since Apr 2024, ~40 % less WebSocket traffic), gRPC/WebSocket payloads, mobile sync | 0.150 | 0.311 | 40% | 10.0% | transfer, $2/TB | ×0.3–×4.0 | Default-on (Discord gateway); volumes modelled | zlib (Discord: stream ratio 6 -> 10, payload 270 -> 166 bytes) |
| iOS/macOS software updates (Apple Archive, LZFSE default; LZRAVEN from OS 27) + system file compression | 9.00 | 10.7 | 3% | 3.0% | transfer, $5/TB | ×0.2–×1.6 | Modeled (OTA algorithm mix not public) | zlib / LZMA (LZFSE ~ zlib-5 ratio at 2-3x speed; LZRAVEN ~ LZMA ratio at 3x speed) - value is speed/energy |
| JPEG XL images (Safari 17+, Firefox 157, Chrome 155 shipping enabled-by-default from 6 Oct 2026; iPhone 16 Pro ProRAW, DNG 1.7, DICOM; Cloudinary serving 0.5-1 B JXL responses/day) | 0.150 | 18.0 | 35% | 5.0% | 50% web transfer $8/TB, 50% photo storage $100/TB-yr | ×0.3–×3.3 | Default-on decoders (Safari, Firefox 157, Chrome 155); web bytes scenario | JPEG (JXL ~20 % smaller lossless recompression, ~50 % smaller lossy at equal quality) |
| Aligned sequencing reads stored as CRAM 3.0/3.1 (ENA/EGA, UK Biobank, All of Us, Genomics England, Broad, Sanger, DRAGEN, Parabricks) | 0.075 | 0.201 | 50% | 15.0% | storage, $100/TB-yr, retained 95%/yr | ×0.5–×2 | Default-on (samtools 1.22 writes CRAM 3.1 when CRAM output is selected); archive sizes measured, CRAM share modelled | BAM (BGZF/Deflate): CRAM 3.1 45-66 % smaller; rANS/Nx16 ~15 % over gzip-based CRAM 2 |
| BCn texture data in Xbox Series X\|S titles decoded by the hardware BCPack (rANS) block | 2.50 | 2.81 | 35% | 8.0% | transfer, $3/TB | ×0.3–×2 | Modeled (hardware confirmed; title share unknown) | zlib on BCn textures |
| Share of Oodle-compressed game data using rANS/tANS modes (LZNA, BitKnit, Leviathan/Kraken tANS option) | 1.50 | 1.97 | 2% | 2.0% | transfer, $3/TB | ×0.2–×3 | Optional production capability ('rarely used') | Oodle's own Huffman modes |
| Draco-compressed glTF / 3D Tiles / point clouds delivered on the web and in apps | 0.0060 | 0.015 | 60% | 5.0% | transfer, $8/TB | ×0.3–×3 | Optional capability (KHR_draco_mesh_compression is an optional glTF extension); volume modelled | gzip-compressed glTF |
| Checkpoints and weights compressed with GPU rANS (nvCOMP gANS, dietGPU, NeuZip) - production volume unknown; illustrative | 0.100 | 5.00 | 25% | 20.0% | storage, $100/TB-yr, retained 5%/yr | ×0.1–×4 | Optional production capability; volume unknown (illustrative) | uncompressed BF16 (gANS 1.46x) or zstd (1.27x) on weights |
| ALICE O² (CERN) Run-3 rANS stage output (compressed time frames to storage; LHC Long Shutdown 3 from mid-2026 to ~2030) | 0.080 | 0.250 | 15% | 15.0% | storage, $30/TB-yr, retained 97%/yr | ×0.5–×2 | Measured stored volume (423 PB); stage bytes modelled | Huffman/zlib-class coder on the same detector symbol arrays |

### Installed-base and user anchors

- **zstd machines**: Linux servers and Android devices on kernel ≥4.14 (2018–20), Windows 11 (Oct 2023), Chrome 123/Firefox 126 incl. Android Chrome (2024), Steam client (2025), Safari/iOS 26.3 (Feb 2026); saturating near the global active device base (~7–8 B) with 3–5 %/yr growth after 2026.
- **Apple**: Apple-disclosed active devices (1.0 B Jan 2016 → 2.5 B Jan 2026) × share on iOS 9+/OS X 10.11+; users ≈ devices ÷ 1.7; +4 %/yr devices, +3 %/yr users after 2026.
- **JPEG XL**: Apple devices on iOS 17+/macOS 14+ (60 % of the base by end-2023, ~95 % by 2026) + Windows 11 extension + Firefox 157; central scenario adds Chromium default-on in 2027 (StatCounter Chrome share 69 %).
- **CRAM**: compute nodes running htslib-family software ≈ 5k (2014) → 120k (2026); practitioners 2k → 200k (bioconda 9–11 M downloads per package; ~30k UK Biobank researchers); +15 %/yr.
- **Xbox**: third-party Series X|S estimates 3.5 M (2020) → 36 M (2026); users = 1.2 × consoles; +3 %/yr.
- **Oodle**: PS5 units (Sony: 95.3 M by Jun 2026) plus PCs/consoles running UE 4.27+/UE5 titles (Unreal ≈31 % of Steam unit sales in 2024); +5 %/yr.
- **Draco**: yearly reach of Draco-compressed web/app 3D content, 5 M (2017) → 220 M (2026); +12 %/yr (low confidence).
- **GPU/AI**: datacenter GPUs using nvCOMP ANS/dietGPU-class codecs, 2k (2022) → 60k (2026), illustrative; engineers 200 → 10k; +90 %/+50 %/yr (low confidence).
- **ALICE O²**: EPN farm 350 servers (2022–); people = ALICE collaboration (~2.5k).

### Volume anchors (compressed bytes per year)

- **Steam**: Valve 80 EB (2024), 100 EB (2025), assumed +8 %/yr; zstd share of delivered bytes 15 % (2025), 50 % (2026), 63/72/79/80 % (2027–30) — only new and updated chunks are zstd, but migration has now run for twenty months. Range widened to ×0.7–×1.9; high scenario 30 % and ~75 %.
- **HTTP zstd**: compressible web text ≈150–300 EB/yr (≈$1.8\times10^{14}$ page loads × ~0.8 MB compressed text, plus API traffic); zstd share from Web Almanac (2.7 % CDN 2024, 12 % 2025) and Cloudflare/Google/Meta defaults → 4, 15, 35 EB in 2024–26.
- **Data platforms**: structured/semi-structured enterprise writes ≈200–400 EB/yr compressed (2024–26), zstd share rising to ~20–25 %, split into durable tables/databases (24, 30, 36 EB in 2024–26; retention 80 %/yr) and transient streams/logs/caches/CI (14, 17, 20 EB; retention 10 %/yr); Meta separately 15, 17, 20 EB split 50/50 durable/transient. Scientific/HPC (RNTuple, Zarr, VBZ, HDF5, GDAL) 1.1 EB in 2026, +40 %/yr.
- **Apple updates**: devices × ~4.5 GB/yr of update payloads × 80 % updating → 9 EB (2026), range ×0.2–×1.6 (OTA algorithm mix not public). High scenario 13 EB.
- **Xbox BCPack**: consoles × ~250 GB/yr downloads × 55 % BCn texture share × 50 % of titles using BCPack.
- **CRAM**: ENA/EGA growth (~15–20 PB/yr), UK Biobank (500k WGS ≈ 12–15 PB), All of Us (4.75 PB), Genomics England, clinical DRAGEN output → 45 PB (2024), 75 PB (2026); +28 %/yr.
- **JPEG XL**: ProRAW/DNG/DICOM/Safari web ≈0.1 EB (2026), consistent with Cloudinary's 0.5–1 B responses/day at 80–250 kB. Decoders become universal in Q4 2026 (Firefox 157 on 29 Sep, Chrome 155 stable 6 Oct), so the content ramp starts a year earlier than in v2.1 → 1.5, 5, 10, 18 EB (2027–30) ≈ 0.7–8 % of web image bytes (~180–220 EB/yr). High scenario 3, 9, 16, 26 EB.

- **Local filesystems** (new in v2.2): Btrfs `compress=zstd:1` is Fedora's default since F34 (~40 % of disk saved in Fedora's own analysis), plus OpenZFS zstd datasets, SquashFS/EROFS images and opt-in Android zram recompression → 5.0 EB (2026) of compressed writes, ×0.2–×4, retention 75 %/yr, self-hosted storage at $30/TB-yr.
- **Messaging and RPC** (new in v2.2): Discord's gateway moved from zlib to streaming zstd in April 2024 (ratio 6 → 10, ~40 % less WebSocket traffic); with comparable gRPC/WebSocket paths → 0.1 EB (2026), ×0.3–×4. Bytes are small; operations and people reached are not.
- **Oodle ANS modes**: ~3 % of ≈40–60 EB/yr Oodle-compressed game delivery (PS5 + UE titles).
- **GPU/AI**: checkpoint/weight compression with gANS — volume unknown; illustrative 0.1 EB (2026, range 0.01–0.4) → 5 EB (2030); retention 5 %/yr; NVIDIA's example job = 1.13 PB/month. Communication (UCCL-Zip) and token-store bytes not modelled.
- **ALICE O²**: compressed raw data ≈423 PB over 2022–Feb 2025 → ≈0.14 EB/yr at full duty cycle; ≈0.08 EB in 2026 (data taking ended 29 June 2026); reprocessing only during Long Shutdown 3; Run 4 from ~2030. Savings vs a Huffman/zlib-class coder 15 % (assumed); storage $30/TB-yr.
- **Game repacks (LOLZ/RAZOR)**: excluded from totals (0.1–3 EB/yr plausible; visits are a weak proxy).

## Appendix C. Catalogue of ANS implementations (condensed)

Compiled from source repositories, vendor documentation and registries; ✔ = primary source read, ~ = secondary source. URLs in Section 9 ("Catalogue").

| Name | Author / org | Year | Variant | Platform | Use / notability | |
|---|---|---|---|---|---|---|
| fpaqa/fpaqb/fpaqc | M. Mahoney | 2007–08 | ABS/uABS | C++ | First compressors using Duda's coder | ✔ |
| FSE | Y. Collet | 2013 | tANS | C | Canonical tANS; basis of zstd and LZFSE | ✔ |
| zhuff | Y. Collet | 2013–14 | tANS | C | First LZ+FSE compressor | ✔ |
| ryg_rans | F. Giesen (RAD) | 2014 | rANS, SIMD interleaved | C/C++ | Canonical rANS; ported everywhere | ✔ |
| Understanding ANS series | C. Bloom (RAD) | 2014 | tANS/rANS | blog+code | Analysis feeding Oodle LZNA | ✔ |
| AsymmetricNumeralSystemsToolkit | J. Duda | 2014 | tANS | C++ | Reference toolkit; tuned spread (2021) | ✔ |
| TurboANX / lzturbo | H. Buzidi | 2014– | tANS | C | Pareto-front compressor family | ✔ |
| rans_static → CRAM 3.0 | J. Bonfield | 2014 | rANS 4×8 | C | Genomics standard | ✔ |
| Zstandard | Collet / Meta | 2015–16 | tANS + Huffman | C, RFC 8878 | Most deployed ANS carrier | ✔ |
| LZFSE | Apple | 2015 | tANS | C | iOS/macOS default compression | ✔ |
| Oodle LZNA; BitKnit | RAD | 2015–16 | rANS adaptive | C/C++ | Games | ✔ |
| Oodle Kraken/Mermaid/Leviathan tANS mode | RAD | 2018 | tANS (optional) | C/C++ | Rarely selected | ✔ |
| FSC | P. Massimino | 2015 | tANS | C | Independent tANS | ~ |
| Draco | Google | 2017 | rANS | C++/JS/WASM | glTF geometry compression | ✔ |
| PIK; Brunsli; JPEG XL (libjxl) | Google / JPEG | 2017–22 | rANS | C++ | Image codecs; ISO/IEC 18181 | ✔ |
| WebP 2 | Google | 2020– | ANS | C++ | Experimental | ✔ |
| VP10 / early AV1 ANS experiment | Google | 2016–17 | rANS/uABS | C | Removed before AV1 1.0 | ~ |
| DivANS | Dropbox | 2018 | rANS | Rust | Cold-storage research compressor | ✔ |
| Kanzi | F. Langlet | 2013– | rANS | Java/Go/C++ | ANS0/ANS1 entropy coders | ✔ |
| rans-tricks | L. Marsh | 2018 | rANS | C++ | Branchless, adaptive | ✔ |
| Xbox Series BCPack | Microsoft | 2020 | rANS (ASIC) | Xbox Series X\|S | Hardware texture decoder | ✔ |
| US 11,234,023 + family | Microsoft | 2019–22 | rANS | patent | Two-phase hardware rANS decoder | ✔ |
| Turbo-Range-Coder | powturbo | 2020 | adaptive rANS | C | SIMD incl. RISC-V RVV | ✔ |
| htscodecs (rANS Nx16), JS port, htsjdk, cram-js, noodles, Genozip, fqzcomp5 | Bonfield; Broad; GMOD; Macias; Lan | 2018–25 | rANS | C/JS/Java/Rust | CRAM 3.1 ecosystem | ✔ |
| PetaGene | PetaGene Ltd | 2016– | ANS-based (patented) | commercial | Genomic compression | ~ |
| multians | Weißenberger & Schmidt | 2019 | tANS (GPU) | CUDA | Parallel ANS decoding | ✔ |
| dietGPU; hipANS | Meta; PAA-NCIC | 2022 | rANS (GPU) | CUDA/HIP | Float codec; archived 2026 | ✔ |
| nvCOMP ANS / gANS; nvCOMPDx | NVIDIA | 2022–26 | rANS (GPU) | CUDA | float16/float8 modes; checkpoints | ✔ |
| Recoil | Lin et al. | 2023 | rANS | CPU/GPU | Parallel decoding with split points | ✔ |
| hypersonic-rANS | C. Stiller | 2023 | rANS AVX-512 | C++ | 18 GB/s multi-thread | ✔ |
| Vitis Data Compression (zstd kernels) | AMD/Xilinx | 2020–21 | tANS (FPGA) | Alveo | FSE decode in HLS | ✔ |
| FPGA tANS decoder (Najmabadi et al.); IJET 2024 encoder | Univ. Stuttgart; PL | 2018; 2024 | tANS (FPGA) | hardware | Academic hardware | ~ |
| BB-ANS; Bit-Swap; HiLLoC; craystack; multiset compression; McBits | UCL; Berkeley; FAIR | 2019–21 | rANS (bits-back) | Python | Latent-variable lossless compression | ✔ |
| CompressAI | InterDigital | 2020 | rANS | PyTorch | Learned image compression | ✔ |
| constriction | R. Bamler | 2021 | ANS/range/chain | Rust/Python | Research library | ✔ |
| YAECL | T. Xu | 2022 | rANS + AC | C++/Python | Neural compression | ✔ |
| NeuZip | Borealis AI | 2024 | ANS (nvCOMP) | PyTorch | Training memory reduction | ✔ |
| Shannon-bound LLM weight compression; rANS split computing; KVTC | NUS/ETH; KNU; OSS | 2025–26 | rANS (GPU) | CUDA | Inference-side ANS | ✔ |
| jxl-oxide; jxl-rs | tirr-c; libjxl org | 2023–26 | rANS | Rust | JPEG XL decoders (Firefox, Chromium) | ✔ |
| LZRAVEN | Apple | 2026 | adaptive rANS (RE) | OS 27 | LZMA replacement; OTA payloads | ~ |
| Rust: rans-rs, ryg-rans-sys, ryg-rans-rs, ans, zune-entropy, webgraph-ans-rs, mzc | various | 2021–26 | rANS/tANS | Rust | Libraries and archiver | ✔ |
| Python: py-rans, Stanford Compression Library, rANS tutorial | Glazov; Tatwawadi; Townsend | 2020–22 | rANS/tANS | Python | Teaching/research | ✔ |

| Name | Author / org | Year | Variant | Platform | Use / notability | |
|---|---|---|---|---|---|---|
| Go/Java/JS: kanzi-go, klauspost fse, nathanhack ANS, libANS, ANS-Graph-compression, fzstd, htscodecs JS | various | 2017–24 | rANS/tANS | Go/Java/JS | Libraries | ✔ |
| Zstd re-implementations: ruzstd, ZstdSharp, aircompressor, Zig std | various | 2017–23 | tANS (FSE) | Rust/C#/Java/Zig | Independent FSE decoders | ✔ |
| ALICE O² rANS (CERN) | ALICE collaboration | 2021– | rANS, static tables | C++ (O²) | Run-3 online data reduction, 350-server EPN farm | ✔ |
| LOLZ; RAZOR | ProFrager; C. Martelock | 2017–18 | adaptive rANS + LZ/ROLZ | Windows CLI | Game-repack pipelines (LOLZ: author statement; RAZOR's rANS reported on encode.su, not verified) | ~ |
| JPEG AI (ISO/IEC 6048-1:2025) | JPEG committee | 2025 | ANS | standard | Learned image coding; range coder replaced by ANS (2023) | ✔ |
| MANS / H5Z-MANS; UCCL-Zip; torch_ans; Fast Compressed Segmentation Volumes | CAS; UC Davis et al.; worldlife123; — | 2023–26 | ANS / rANS (GPU) | CUDA/ROCm/HDF5 | HPC integers, collectives, PyTorch extension, scientific volumes | ✔ |
| Token-native storage; RAS accelerator; GS-NFS | K. Shivendu; Qin, Fan, Yan; USC | 2025–26 | static unigram ANS; rANS (RTL); rANS (nvCOMP) | research | Agent/RAG stores (3.3× over UTF-8); hardware rANS; Gaussian-splat streaming | ✔ |
| Huawei PILC / OSOA; Alibaba ANS-LIC | Huawei Noah's Ark; Alibaba DAMO | 2021–25 | modified ANS; bb-ANS; ANS hardware | research | Learned lossless image coding; no deployment | ~ |
| Falcon audio codec; @dotprotocol/compression; marc | pbtechlab; DOT; Masa-tam | 2025–26 | rANS / ANS | Rust/JS/C++ | New codecs | ✔/~ |

Report generated 2 September 2026. Model, figures and this document were produced with Python (pandas, matplotlib) and Chromium; all inputs are in `model.py`.