Characterizing Lossless GPU Data Compression Across AMD CDNA and RDNA Architectures

Cristiano Künas[1(✉)], Gabriel Freytag[1], Jean Luca Bez[2], Thiago Araújo[1], and Philippe Navaux[1]

[1] Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil
{cakunas,gfreytag,tsaraujo,navaux}@inf.ufrgs.br
[2] Lawrence Berkeley National Laboratory, Berkeley, CA, USA
jlbez@lbl.gov

Abstract. Data movement remains a major bottleneck in high-performance computing workflows, making GPU-accelerated compression an increasingly important optimization. While NVIDIA platforms benefit from mature compression libraries such as nvCOMP, the performance of practical lossless compression on AMD GPUs remains undercharacterized. This paper addresses that gap by porting LZ4, Snappy, and Cascaded from CUDA to HIP and evaluating them across four AMD GPUs spanning three CDNA generations (MI50, MI210, MI300X) and RDNA 3 (RX 7900 XT), using both synthetic datasets and seismic simulation data. Our study makes three contributions. First, we provide a cross-generational characterization of lossless GPU compression on recent AMD accelerators. Second, we present a functional CUDA-to-HIP port of three widely used algorithms, establishing an open baseline for future AMD-specific optimization. Third, we analyze how architectural differences shape compression and decompression behavior across algorithms and datasets. The results show that the MI300X achieves up to 11× higher decompression throughput than the MI50, while RDNA 3 is competitive for compression workloads characterized by irregular memory accesses. We also show that transfers dominate end-to-end execution time on all evaluated platforms, indicating that the main benefits of GPU compression are most likely to emerge in GPU-resident workflows. For realistic seismic floating-point data, lossless compression ratios remain modest ($1.03 - 1.04\times$), suggesting that error-bounded lossy compression is a promising direction for future work.

Keywords: GPU Computing · Data Compression · LZ4 · AMD · HIP · ROCm · High-Performance Computing

1 Introduction

Modern scientific computing faces an increasingly critical challenge: the volume of data generated by simulations is growing faster than our ability to move it

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16767, pp. 357–374, 2027.
https://doi.org/10.1007/978-3-032-30491-9_23

efficiently. High-performance computing (HPC) applications like seismic simulation, climate modeling, and fluid dynamics produce datasets ranging from gigabytes to petabytes, making data movement the dominant bottleneck in many workflows [8]. This I/O wall has motivated significant interest in lossless data compression [10,16,24,26], which is already a critical component in storage systems, from parallel file systems in HPC clusters to cloud-based object stores, where it reduces both capacity costs and transfer overhead.

GPU-accelerated compression has emerged as an attractive solution, leveraging the parallel architecture of graphics processors to achieve throughputs that can exceed storage system bandwidths. On NVIDIA platforms, the nvCOMP library [26] has become the standard, offering highly optimized implementations of algorithms such as LZ4 [10], Snappy [16], and Cascaded [24] that achieve high compression throughput on modern GPUs. This mature ecosystem has enabled researchers to integrate high-performance compression into their CUDA-based workflows seamlessly.

However, the HPC landscape is becoming increasingly heterogeneous. AMD GPUs now power several of the world's most powerful supercomputers, including El Capitan at Lawrence Livermore National Laboratory and Frontier at Oak Ridge National Laboratory, the two highest-ranked systems on the TOP500 list [28], both powered by AMD hardware. This shift creates an urgent need for portable, high-performance software tools capable of running efficiently on GPUs. Unfortunately, a significant **tool gap** exists, while the CUDA ecosystem benefits from years of mature compression libraries, AMD users lack equivalent tools. Existing community efforts provide basic functionality but have not received the same level of optimization, leaving researchers with AMD hardware at a disadvantage for data-intensive workflows.

This paper addresses this gap through three contributions. First, we provide a systematic cross-generational characterization of lossless compression performance on AMD GPUs, including CDNA (Compute DNA) and RDNA (Radeon DNA) architectures, covering both synthetic and seismic datasets. Second, we present a functional CUDA-to-HIP port of three practical algorithms (LZ4, Snappy, and Cascaded), derived from nvCOMP 2.2, thereby establishing an open AMD-compatible baseline for future research and optimization. Finally, we analyze how architectural features such as cache hierarchy, memory bandwidth, and wavefront behavior shape compression and decompression throughput, exposing distinct performance across AMD GPU families.

Our evaluation spans four AMD GPUs (MI50, MI210, MI300X, and RX 7900 XT), using synthetic inputs and realistic TTI seismic wavefields. The results show that decompression throughput scales strongly across generations (MI300X achieves up to 11× the MI50 baseline), that CDNA and RDNA architectures exhibit complementary strengths (RDNA 3 is competitive for compression with irregular access patterns, while CDNA 3 dominates decompression), and that transfer overhead dominates end-to-end time on all platforms, indicating that kernel-level gains require integration into GPU-resident workflows.

The remainder of this paper is organized as follows. Section 2 reviews related work and positions our study within the broader GPU-compression literature. Section 3 provides background on GPU-based compression and motivates the need for AMD-compatible tools. Section 4 describes our porting approach and implementation. Section 5 details the experimental methodology. Section 6 presents the cross-generational performance analysis. Section 7 discusses the implications of the results, limitations of the current study, and directions for future work. Section 8 concludes the paper.

2 Related Work

Data compression has become a central technique for mitigating the widening gap between computational capability and storage or communication performance in HPC systems. Earlier I/O and resilience work already established this motivation: *Hello ADIOS* [22] discussed the growing importance of data-management efficiency in leadership-class systems, while *VeloC* [23] showed that checkpointing frameworks increasingly benefit from techniques that reduce data volume and hide I/O costs. These studies make clear why fast accelerator-side compression is valuable, but they do not characterize how such techniques behave on modern AMD GPUs.

A substantial body of research has explored GPU compression for scientific data, especially in the error-bounded lossy setting. *cuSZ* [27] demonstrated one of the first efficient CUDA-based implementations of SZ on GPUs, and later work such as *cuSZp* [18], *cuSZp2* [17], and *FZ-GPU* [29] pushed the design space toward higher throughput, stronger kernel fusion, and improved compression ratio on NVIDIA devices. Collectively, these works show that GPU compression is now a mature optimization target. However, their implementations and evaluations are overwhelmingly CUDA-centric, so they provide limited guidance for users targeting AMD accelerators.

Lossless GPU compression has likewise advanced, again largely within the NVIDIA ecosystem. *GPULZ* [30] optimized LZSS-style compression for multi-byte data on modern GPUs, while *ndzip-gpu* [19] targeted efficient lossless compression of scientific floating-point arrays. More recently, *FCBench* emphasized that compressor behavior depends strongly on both dataset characteristics and the runtime bottlenecks exposed by a given platform [9]. This line of work is relevant to our study because it highlights that throughput and compression ratio are not intrinsic properties of an algorithm alone, but emerge from the interaction between algorithm, data, compiler toolchain, and hardware architecture.

A smaller but important set of works has started to address portability and broader cross-platform evaluation. *HPDR* argues for portable high-performance data reduction across heterogeneous systems [8]. These efforts move the field beyond single-platform peak results. Still, they do not provide the cross-generational AMD-centric baseline needed by practitioners deploying on AMD-based systems such as El Capitan or Frontier, nor do they focus on the practical question of how CUDA-originated compression kernels behave after translation to HIP.

A particularly relevant recent study is Burtchell and Burtscher (2025), which characterizes parallel data-compression performance across compilers and GPUs [6]. Both papers adopt a measurement-driven perspective, but differ in focus: whereas that work examines compression behavior across compiler and platform combinations broadly, ours specifically targets the AMD ecosystem, emphasizing CUDA-to-HIP portability and cross-generational behavior across CDNA and RDNA architectures.

Our work is positioned at the intersection of compression research, performance portability, and AMD software support. While `nvCOMP` offers a mature GPU-compression stack on NVIDIA [26], `hipCOMP-core` remains an early-access preview on AMD [4]. In contrast to prior work that primarily proposes new CUDA-based compressors or evaluates on NVIDIA-centric platforms, we provide a quantitative, cross-generational characterization on AMD hardware and report on an initial CUDA-to-HIP port, establishing a baseline for the remaining software and optimization gap.

3 Background and Motivation

3.1 GPU-Based Data Compression

In scientific computing, lossless compression is often preferred as it guarantees exact data reconstruction. Among lossless algorithms, LZ4 [10] offers a favorable balance between speed and ratio, Snappy [16] prioritizes minimal latency, and Cascaded compression applies multiple encoding stages (delta + RLE (Run-Length Encoding) + bit-packing) for higher ratios on structured data. All three can be parallelized across GPU threads, with their implementation details described in Sect. 4.3.

NVIDIA's nvCOMP library [26] provides optimized CUDA implementations of these and other algorithms, achieving throughputs that can saturate the memory bandwidth of high-end GPUs. Version 2.2, which serves as the codebase for our HIP (Heterogeneous-Interface for Portability) port, remains widely deployed. However, nvCOMP is tightly coupled to the CUDA ecosystem, limiting its applicability to other platforms.

3.2 The AMD GPU Landscape

AMD has made substantial inroads in high-performance computing, with its GPUs now powering some of the world's fastest supercomputers. This success is built on two distinct GPU architectures optimized for different markets.

The CDNA architecture is geared towards data center and HPC workloads, with an emphasis on high computational throughput, high-bandwidth memory (HBM), and features specifically designed for scientific computing. The evolution from CDNA (MI50/MI100) to CDNA 2 (MI210/MI250X) and finally to CDNA 3 (MI300X) brought substantial improvements in both raw performance and memory bandwidth. The MI300X, in particular, features 304 compute units

and 192 GB of HBM3 memory with over 5 TB/s of bandwidth, making it well-suited for memory-intensive workloads like data compression.

RDNA targets gaming and consumer applications but has found use in cost-sensitive HPC deployments. RDNA 3, exemplified by the RX 7900 XT, introduces a chiplet-based design with separate compute and memory dies, achieving competitive compute performance at lower power consumption. While not designed for datacenter use, RDNA GPUs offer an interesting point of comparison for understanding how architectural choices affect compression performance. Table 1 summarizes the key specifications of the GPUs evaluated in this work.

Table 1. AMD GPU specifications evaluated in this study.

GPU	Arch.	CUs	Memory	BW (GB/s)	TDP (W)
MI50	CDNA	60	16 GB HBM2	1,024	300
MI210	CDNA 2	104	64 GB HBM2e	1,638	300
MI300X	CDNA 3	304	192 GB HBM3	5,300	750
RX 7900 XT	RDNA 3	84	20 GB GDDR6	800	315

AMD's software ecosystem centers on ROCm (Radeon Open Compute), an open-source platform that provides tools for GPU computing. The HIP programming model enables developers to write code that targets both AMD and NVIDIA GPUs with minimal modifications, thereby facilitating portability. However, while HIP simplifies porting CUDA code, achieving optimal performance on AMD hardware often requires architecture-specific optimizations that go beyond simple API translation.

3.3 The Compression Tool Gap

Despite AMD's growing presence in HPC, the availability of optimized GPU compression tools has not kept pace with the hardware. The CUDA ecosystem benefits from nvCOMP—a vendor-supported, production-ready library available since 2020, offering architecture-tuned implementations of LZ4, Snappy, Cascaded, zstd, ANS, and others. In contrast, the HIP/ROCm ecosystem relies on community-driven efforts such as hipCOMP, which appeared in 2022 and remains experimental, with limited algorithm coverage (LZ4 and Snappy), direct-port optimization, and sparse documentation.

This asymmetry creates practical challenges for the growing number of HPC systems built on AMD GPUs. Scientists working on AMD-equipped systems—including El Capitan, Frontier, and others—cannot directly benefit from the mature CUDA compression ecosystem and must either port code themselves, use suboptimal tools, or forgo GPU-accelerated compression entirely. This situation motivates our work: by characterizing the current state of compression on AMD GPUs and providing an open implementation, we aim to bridge this gap and contribute to a vendor-agnostic compression ecosystem for HPC.

4 GPU Compression Library for AMD Platforms

This section describes the implementation of a GPU-accelerated lossless compression library targeting AMD GPUs via the HIP programming model. Our goal was to create a functional and portable implementation suitable for cross-architecture performance characterization, prioritizing correctness and cross-platform compatibility over architecture-specific optimization.

4.1 Porting Approach

Our implementation is based on the open-source nvCOMP 2.2 codebase [26], which provides a batched compression API that processes multiple independent data chunks in parallel—a design well-suited to GPU architectures. We chose this codebase as a starting point because it contains well-structured, GPU-optimized implementations of the three target algorithms (LZ4, Snappy, Cascaded) and its batched API model maps naturally to the HIP programming model. Our porting methodology follows the standard CUDA-to-HIP translation workflow documented in the HIP programming guide [3] and involves three phases.

Phase 1: API Translation. We used AMD's `hipify-perl` tool [3] to perform the initial mechanical translation of CUDA source code to HIP. The tool handled approximately 90% of the API surface automatically; the remaining manual modifications involved replacing CUB [25] device primitives with hipCUB equivalents, adapting warp-level intrinsics, and resolving header conflicts between CUDA and HIP include paths.

Phase 2: Build System Adaptation. We restructured the CMake build to detect the HIP compiler (`hipcc`) and target specific GPU architectures via their GFX identifiers (e.g., `gfx906` for MI50, `gfx942` for MI300X, `gfx1100` for RX 7900 XT). A build-time option controls whether the library compiles with 32-thread or 64-thread wavefront assumptions (Sect. 4.2).

Phase 3: Correctness Validation. We validated the port using a correctness check: for every test input, we verified that `decompress(compress(data)) == data` with bit-exact equality across all chunk sizes, data types, and algorithms. This round-trip validation is a standard technique for verifying lossless compression implementations and ensures that no data corruption was introduced during the porting process.

4.2 Wavefront Size Adaptation

NVIDIA GPUs use 32-thread *warps*, while AMD CDNA GPUs default to 64-thread *wavefronts* [2]; RDNA architectures use 32-thread wavefronts natively [3]. This difference affects compression kernels that rely on warp-level primitives (ballot, shuffle, reduction). To not rewrite all warp-level logic, we adopted a compile-time strategy, defining a `warpsize` constant and a `lane_mask_t` type selected at build time (64-bit masks with `warpsize = 64` for CDNA targets, and

32-bit masks with `warpsize = 32` for RDNA 3), preserving correctness across architectures while exploiting the wider wavefront when available. However, it does not address deeper optimization opportunities, such as restructuring thread cooperation patterns to better match AMD's memory hierarchy, which we leave for future work.

4.3 Chunked Compression Model

All three algorithms use a *batched chunked* compression model [26], where input data is divided into fixed-size chunks compressed independently in parallel (Fig. 1). Each chunk is assigned to one or more thread blocks, enabling thousands to be processed concurrently. The algorithms differ in their thread-to-chunk mapping. **LZ4** assigns one warp per chunk with a private hash table (64 KB) for match-finding. **Snappy** uses two cooperating warps (match-finding + encoding). And **Cascaded** applies a multi-stage pipeline (delta + RLE + bit-packing) with 128 threads per 4 KB partition. These different parallelization strategies lead to distinct performance characteristics across architectures, as analyzed in Sect. 6.

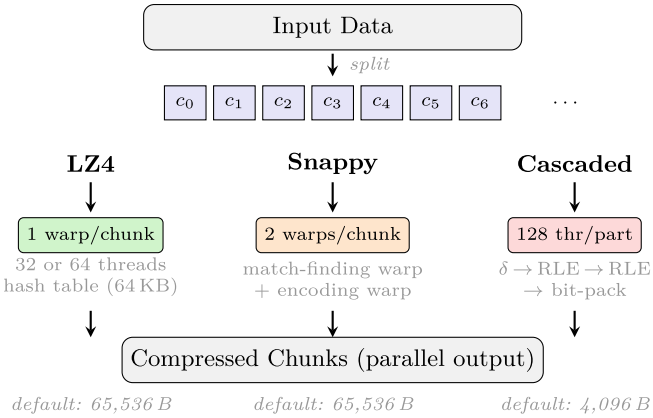

Fig. 1. Batched chunked compression model. Input data is split into independent chunks processed in parallel. Each algorithm assigns different thread configurations per chunk: LZ4 uses a single warp, Snappy uses two cooperating warps, and Cascaded uses 128 threads per 4 KB partition with a multi-stage encoding pipeline.

4.4 Benchmark Suite and Reproducibility

To ensure reproducibility across diverse GPU platforms, we containerized the complete build and execution environment using Singularity [20], the standard container runtime in HPC due to its rootless execution model and native GPU

passthrough. Our container image is based on Ubuntu 22.04 with ROCm 7.0.1 and accepts a GPU architecture argument at build time, ensuring that the same software stack is used across all platforms while targeting architecture-specific binaries.

We developed a benchmark suite that compiles alongside the compression library, with one executable per algorithm. Throughput is measured using GPU-side event timing (`hipEvent`), which captures only kernel execution time and excludes host-device data transfer overhead [7,12]. Throughput is reported in GB/s of *uncompressed* data:

$$\text{Throughput (GB/s)} = \frac{\text{uncompressed size (bytes)}}{t_{\text{kernel}} \times 10^9} \tag{1}$$

where t_{kernel} is the mean kernel execution time in seconds over all timed iterations. An automation layer orchestrates all algorithm×dataset×chunk-size combinations, collecting results into structured CSV files.

5 Experimental Methodology

This section details the experimental setup, dataset design, and measurement protocol used to evaluate compression performance across AMD GPU generations. Our methodology is designed to enable reproducible comparisons and to isolate GPU-level performance from system-level variability.

5.1 Hardware Platforms

We evaluated compression performance on four AMD GPUs spanning three architectural generations, as detailed in Table 1. The MI50 (CDNA) serves as our baseline, representing first-generation compute-focused AMD GPUs with 16 GB of HBM2 memory and 1,024 GB/s bandwidth. The MI210 represents CDNA 2, the architecture widely deployed in systems such as LUMI [11] at CSC Finland. The MI300X is AMD's current flagship datacenter accelerator based on CDNA 3, featuring 192 GB of HBM3 memory with over 5.3 TB/s of aggregate bandwidth. The RX 7900 XT provides a consumer-grade RDNA 3 comparison point, featuring a chiplet-based design with 20 GB GDDR6 and AMD's Infinity Cache.

Computing Environments. The MI50, MI210, and MI300X experiments were conducted using the Grid'5000 testbed [5], a large-scale distributed infrastructure for experimental research in computer science. The RX 7900 XT experiments were performed at the PCAD[1] laboratory at UFRGS. On all platforms, experiment execution used Singularity containers with identical ROCm 7.0.1 userspace (Sect. 4.4), differing only in the GPU architecture target used at build time.

[1] https://pcad.inf.ufrgs.br/.

5.2 Benchmark Datasets

We designed four dataset categories spanning the compressibility spectrum (Fig. 2), combining synthetic patterns with real scientific data [7,27].

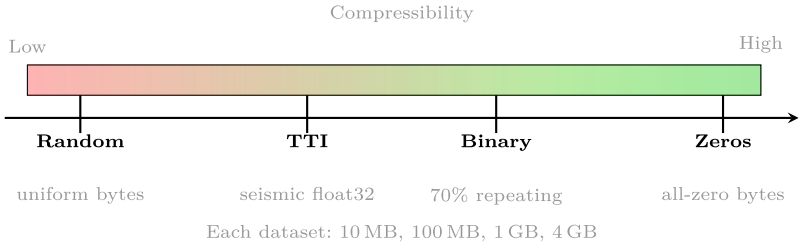

Fig. 2. Benchmark dataset compressibility spectrum.

Three **synthetic datasets** provide controlled evaluation. *Zeros* (all-zero bytes, maximum compressibility), *Random* (uniform bytes from `/dev/urandom`, incompressible), and *Binary* (70% repeating sequences, 30% random, moderate compressibility).

TTI (Tilted Transversely Isotropic) Wavefield Data provides a realistic benchmark from computational geophysics [1,14]. The dataset was extracted from a 3D seismic wave-propagation simulation on a 448^3 grid with 201 timesteps, producing 67.3 GB of single-precision floating-point data in RSF format [15]. We extract timesteps from the middle of the simulation (starting at timestep 100) to avoid early timesteps dominated by near-zero initialization values that would inflate compression ratios.

Each dataset category was evaluated at four sizes: Small (10 MB), Medium (100 MB), Large (1 GB), and XLarge (4 GB), yielding 4 data types $\times$ 4 sizes = 16 test files and 48 benchmark configurations per GPU.

5.3 Measurement Protocol

Our measurement protocol follows established GPU benchmarking practices [7,12] and is designed to minimize measurement noise while capturing representative performance. Throughput is measured using GPU-side event timing (`hipEvent`), isolating kernel execution from host-device transfers, as detailed in Sect. 4.4.

Each benchmark configuration executes $w = 3$ warmup iterations followed by $n = 10$ timed iterations, allowing the GPU clock to stabilize, JIT compilation to complete, and memory caches to reach steady state. We report the arithmetic mean over timed iterations. All experiments use the default chunk sizes described in Sect. 4.3.

We report three primary metrics: *compression throughput* (GB/s), computed as uncompressed data size divided by compression kernel time (Eq. 1); *decompression throughput* (GB/s), using decompression kernel time; and *compression ratio*, the original size divided by compressed size (higher values indicate better compression). Cross-generation comparisons use *speedup* relative to the MI50 baseline, calculated as the ratio of throughput on the target GPU to throughput on the MI50 for the same algorithm, dataset, and size.

6 Results: Cross-Generation Analysis

This section presents our experimental evaluation of LZ4, Snappy, and Cascaded compression across AMD GPU generations. We analyze throughput across algorithms and data types, execution time composition, compression ratios, scaling behavior, and cross-generation speedup.

6.1 Compression and Decompression Throughput

Figure 3 presents compression and decompression throughput for the XLarge (4 GB) dataset, the most representative size, across all three algorithms, four data types, and four GPUs.

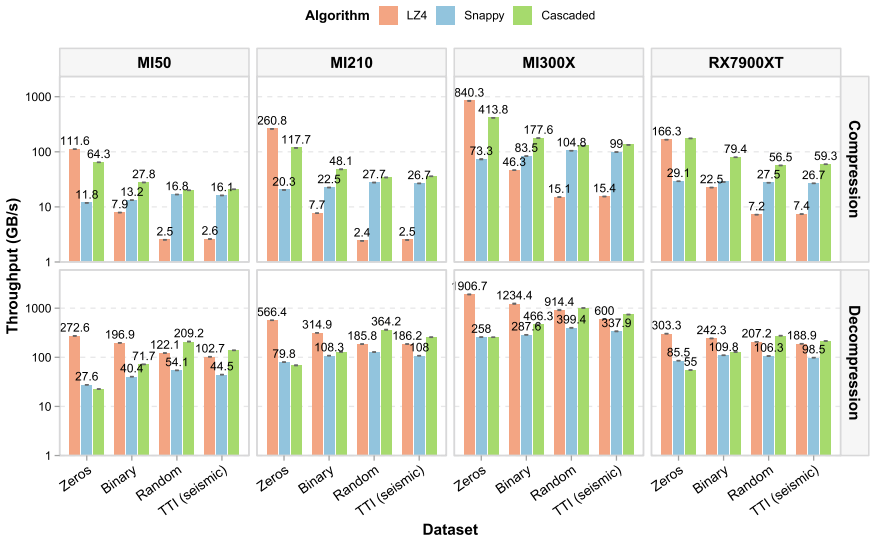

Fig. 3. Compression (top) and decompression (bottom) throughput for the 4 GB dataset across GPUs, algorithms, and data types. Higher is better.

Two fundamental patterns emerge. First, **data compressibility dramatically affects LZ4 throughput**. On the MI300X, LZ4 reaches 840 GB/s compressing zeros but only 15 GB/s for TTI seismic data, a difference exceeding 50×. This sensitivity arises because highly compressible data yields very

small outputs, so the kernel spends less time on hash-table match searches. Cascaded, by contrast, maintains remarkably stable throughput across data types (414 GB/s for zeros vs. 134 GB/s for TTI on the MI300X), due to its fixed delta+RLE+bitpacking pipeline executing the same computational stages regardless of data content.

Second, **compression and decompression favor different architectures**. For compression, the consumer-grade RX 7900 XT matches or exceeds the datacenter MI300X across most datasets. We attribute this to RDNA 3's 96 MB Infinity Cache providing low-latency access to the hash tables (up to 64 KB per warp) that dominate compression's random access pattern, combined with higher clock speeds (up to 2.5 GHz vs. 2.1 GHz). For decompression, the MI300X dominates, reaching 1907 GB/s on zeros. Its 5.3 TB/s HBM3 bandwidth fully serves decompression's sequential streaming pattern of reading compressed data and writing large contiguous output blocks.

The MI210 (CDNA 2) occupies the expected intermediate position between the MI50 and MI300X. For decompression, the MI210 achieves 1.5–3.0× speedup over the MI50 across algorithms, consistent with its 1.6× higher HBM bandwidth (1,638 GB/s vs. 1,024 GB/s). For compression, gains are more modest (1.7× for Snappy and Cascaded on TTI data), while LZ4 compression shows no improvement over the MI50, with both achieving ∼2.5 GB/s for TTI. This confirms that LZ4 compression is bound by hash-table latency rather than memory bandwidth.

6.2 Execution Time Breakdown

While throughput metrics isolate kernel performance, practical deployments must also transfer data between host and device. Figure 4 decomposes total execution time into four phases for the TTI dataset at 4 GB, namely host-to-device (H2D) transfer, compression, decompression, and device-to-host (D2H) transfer.

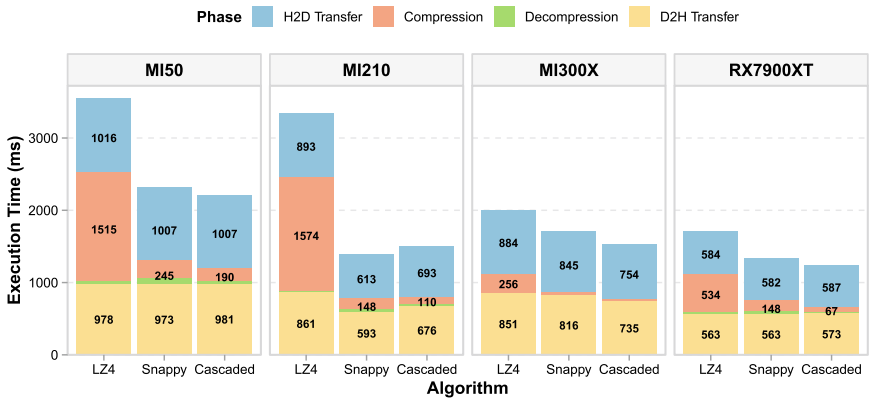

Fig. 4. Execution time breakdown for TTI seismic data (4 GB). Transfer phases (H2D and D2H) dominate total time on all GPUs, especially for Cascaded and Snappy.

Data transfer dominates total execution time across all GPUs. On the MI50, LZ4 compression still represents a significant fraction of total time (1515 ms, ∼43%), but even here, H2D and D2H transfers account for the majority. On the MI300X, faster kernels shift the balance further, with LZ4 compression dropping to 256 ms (∼13% of total), while transfers consume the remaining time. For Cascaded on the RX 7900 XT, compression takes only 67 ms, under 5% of total execution.

This has a practical implication. Optimizing compression kernels alone yields diminishing returns without addressing transfer overhead. Integration approaches that overlap compression with transfers, or that operate directly on GPU-resident data, such as compressing simulation output before writing to storage, would better exploit the kernel-level performance gains.

6.3 Compression Ratio

Table 2 presents compression ratios across algorithms and data types for the 4 GB dataset. Ratios are GPU-independent, depending only on the algorithm and data content.

Table 2. Compression ratio by algorithm and dataset (4 GB input). Ratios are GPU-independent.

Algorithm	TTI (seismic)	Binary	Random	Zeros
LZ4	1.04	3.26	1.00	245
Snappy	1.03	2.96	1.00	21.3
Cascaded	1.03	3.04	1.00	92.0

LZ4 achieves the highest ratios for highly compressible data (245× for zeros), but offers low compression for TTI seismic data (1.04×) and random data (1.00×). Snappy exhibits similar trends with lower peak ratios (21.3× for zeros).

For the TTI scientific dataset, our use case, all three algorithms achieve near-identical compression (1.03–1.04×), indicating that single-precision seismic wavefield data contains limited byte-level redundancy exploitable by lossless methods. This result suggests that, for this class of seismic floating-point data, lossless compression is best viewed as a baseline rather than as a high-impact data-reduction solution, motivating future investigation of error-bounded lossy approaches.

6.4 Throughput Scaling

Figure 5 shows how compression throughput scales with input size for the TTI dataset, the most relevant workload for our target application.

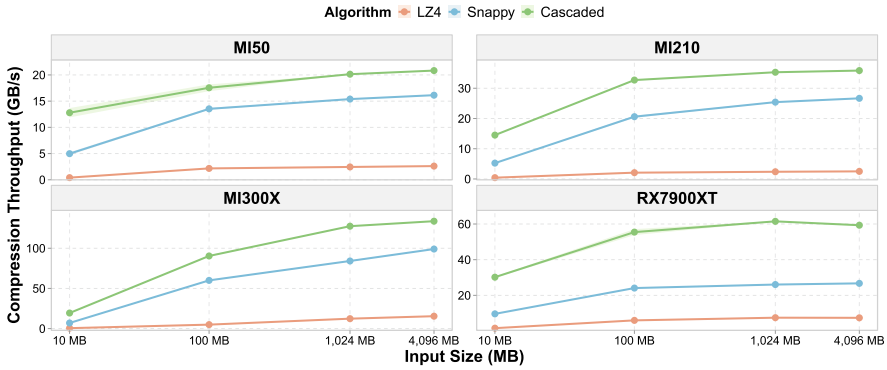

Fig. 5. Compression throughput scaling with input size for TTI seismic data across GPU generations. Throughput increases with input size as parallelism is better utilized.

All algorithms exhibit increasing throughput from 10 MB to 4 GB, reflecting improved GPU utilization at larger input sizes. At 10 MB, kernel launch overhead and low parallelism limit performance across all GPUs. By 1 GB, most configurations approach peak throughput, with only borderline gains at 4 GB.

On the MI300X, Cascaded and Snappy show steep scaling from 100 MB to 1 GB, reflecting the GPU's need for more data to saturate its 304 compute units. The MI50, with 60 compute units, saturates earlier and shows flatter scaling. The MI210, with 104 compute units, exhibits intermediate scaling behavior, saturating at approximately 1 GB input size. LZ4 shows the flattest scaling across all GPUs for TTI data, consistent with its hash-table-bound compression being less sensitive to increased data parallelism.

6.5 Cross-Generation Speedup

Figures 6 and 7 summarize compression and decompression speedup relative to the MI50 baseline across all datasets and input sizes.

The speedup results highlight a fundamental **compression/decompression asymmetry** across architectures. For compression (Fig. 6), the MI300X achieves up to 8× speedup for Snappy and Cascaded at large input sizes, while LZ4 gains are more modest (6×), limited by hash-table latency that does not scale with bandwidth. The RX 7900 XT shows consistent compression speedups across algorithms and sizes (up to 4×), benefiting from its Infinity Cache and higher clock frequency.

For decompression (Fig. 7), the MI300X reaches up to 11× speedup, with LZ4 and Snappy benefiting most from HBM3 bandwidth. The RX 7900 XT's decompression speedup is more modest (up to 3×), constrained by its 800 GB/s GDDR6 bandwidth.

The MI210 demonstrates moderate generational gains, achieving up to 2.4× decompression speedup and 1.7× compression speedup over the MI50 for Snappy

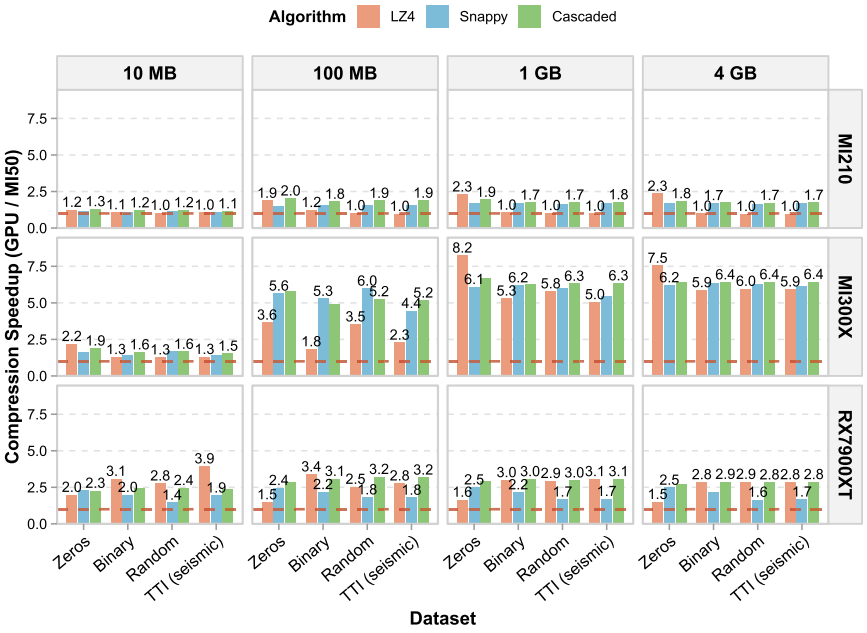

Fig. 6. Speedup relative to MI50 baseline for compression across datasets and input sizes. The dashed line marks MI50 baseline (1×).

and Cascaded on TTI data. The LZ4 compression on the MI210 shows virtually no improvement over the MI50 (∼2.5 GB/s on both), while decompression improves by 1.8×. These gains align with the MI210's 1.6× higher memory bandwidth, suggesting near-linear scaling with bandwidth for decompression-bound workloads.

This asymmetry has practical implications for hardware selection. Applications that primarily decompress data (e.g., analysis pipelines reading compressed datasets) benefit from HBM-equipped CDNA hardware. At the same time, workloads dominated by compression may find RDNA hardware competitive or benefit from architecture-specific kernel optimizations on CDNA.

7 Discussion

The compression-decompression asymmetry reported in Sect. 6 reflects a fundamental difference in memory access patterns. Compression relies on small, random hash-table reads that benefit from RDNA 3's Infinity Cache, while decompression follows a sequential streaming pattern that favors raw HBM bandwidth. The MI210's decompression scaling closely tracks its bandwidth advantage, whereas LZ4 compression remains latency-bound across all CDNA generations. These findings suggest that hardware selection should consider the

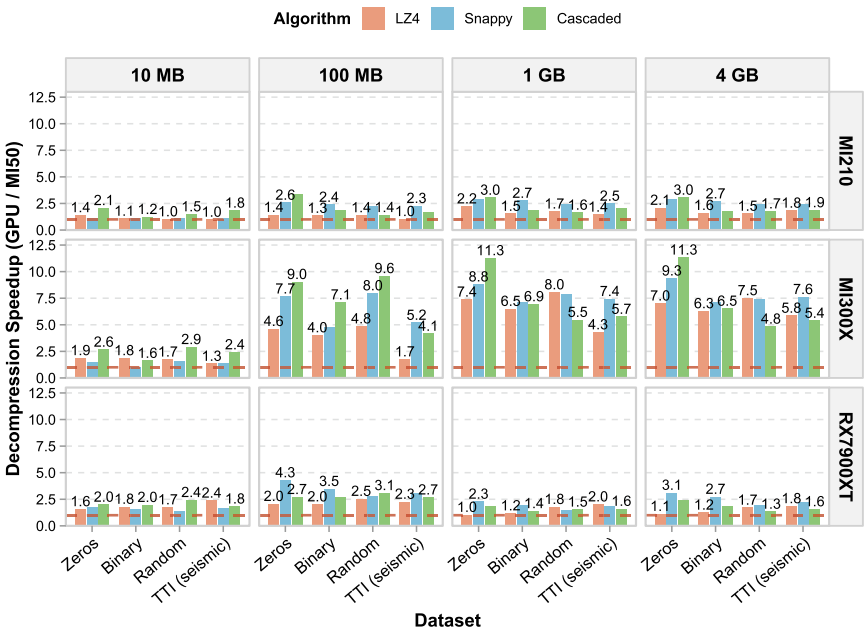

Fig. 7. Speedup relative to MI50 baseline for decompression across datasets and input sizes. The dashed line marks MI50 baseline (1×).

compression-to-decompression ratio of the target workload, not just peak bandwidth or compute specifications.

Despite the current optimization headroom, GPU compression on AMD platforms already provides tangible value in I/O-bound workflows. For instance, writing a 4 GB checkpoint to a parallel file system at 10 GB/s takes 0.4 s without compression. For highly compressible data such as sparse matrices or partially converged fields (compression ratios of 3× or higher, as observed for the binary dataset), the reduction in I/O volume can significantly reduce checkpoint time. For decompression-dominated workflows, the MI300X's throughput (Fig. 3) means that storage bandwidth is almost always the bottleneck, so compression ratio translates directly to end-to-end speedup.

Our characterization also reveals concrete optimization opportunities. A CDNA-optimized LZ4 kernel could redesign the current one-warp-per-chunk model to exploit full 64-thread wavefronts, and move the 64 KB hash table into the Local Data Share (LDS), replacing global-memory lookups with single-cycle LDS reads and directly addressing the latency bottleneck identified in Sect. 6. For floating-point scientific data, byte-shuffling or XOR-based pre-conditioning could expose redundancy invisible to general-purpose compressors, potentially improving the low lossless ratios observed for seismic data (Table 2). Where application constraints permit bounded error, error-bounded lossy compressors such as SZ [13] and ZFP [21] routinely achieve 10–100× ratios on similar data [27],

making their integration a natural next step. Finally, a unified cross-vendor API that compiles to optimized backends on both AMD and NVIDIA platforms would broaden the practical impact of this work.

This study has limitations. The HIP implementation prioritizes portability and correctness over AMD-specific tuning, so results represent a practical baseline rather than peak achievable performance. Throughput measurements isolate kernel execution time, with transfer costs analyzed separately. The evaluation covers three lossless algorithms with fixed chunking, and additional algorithms and tuning configurations remain for future work. Finally, compression behavior is inherently data-dependent, so that different scientific workloads may exhibit different trade-offs.

8 Conclusion

This paper presented a cross-generational characterization of lossless GPU data compression across AMD CDNA and RDNA architectures, evaluating LZ4, Snappy, and Cascaded on four GPUs spanning the MI50, MI210, MI300X, and RX 7900 XT. Beyond presenting raw benchmark results, the study examined how architectural differences shape compression and decompression behavior and established an AMD-compatible baseline through a functional CUDA-to-HIP port.

The results show a clear compression–decompression asymmetry across architectures. RDNA 3 is competitive for compression workloads with irregular memory-access behavior, whereas CDNA 3 strongly favors decompression through substantially higher HBM bandwidth, with the MI300X reaching up to 11× the MI50 baseline. At the same time, the execution-time breakdown shows that PCIe transfer overhead dominates end-to-end time on all evaluated platforms, indicating that the full benefits of GPU compression are most likely to arise when compression is integrated into GPU-resident workflows rather than used as an isolated kernel optimization.

For realistic seismic floating-point data, all evaluated lossless methods achieve only modest compression ratios (1.03–1.04×). This does not weaken the relevance of the study; rather, it clarifies the role of lossless compression as a portability and systems baseline and points to a more promising next step for this application domain: error-bounded lossy compression. Future work will therefore focus on AMD-specific kernel optimizations, broader algorithm coverage, and integration with portable data-reduction pipelines for heterogeneous HPC environments.

Acknowledgments. This study was partially supported by Petrobras grant no. 2020/00182-5, by CNPq/MCTI/FNDCT - Universal under grants 408755/2025-3, and by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. This effort was also partially supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research (ASCR) under contract number DE-AC02-05CH11231 with LBNL.

Some experiments in this work used the PCAD infrastructure, https://pcad.inf.ufrgs.br/, at INF/UFRGS. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

1. Alkhalifah, T.: An acoustic wave equation for anisotropic media. Geophysics **65**(4), 1239–1250 (2000)
2. AMD: CDNA 2 architecture white paper. Technical report, Advanced Micro Devices (2021). https://www.amd.com/content/dam/amd/en/documents/instinct-business-docs/white-papers/amd-cdna2-white-paper.pdf
3. AMD: HIP: Heterogeneous-compute interface for portability (2025). https://github.com/ROCm/HIP. Accessed 19 Feb 2026
4. AMD ROCm: hipCOMP-core: HIP port of compression algorithms (2025). https://github.com/ROCm/hipCOMP-core. Accessed 20 Feb 2026
5. Balouek, D., et al.: Adding virtualization capabilities to the Grid'5000 testbed. In: Ivanov, I.I., van Sinderen, M., Leymann, F., Shan, T. (eds.) CLOSER 2012. CCIS, vol. 367, pp. 3–20. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-04519-1_1
6. Burtchell, B.A., Burtscher, M.: Characterizing the performance of parallel data-compression algorithms across compilers and GPUs. In: Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 269–278. ACM (2025). https://doi.org/10.1145/3731599.3767369
7. Che, S., et al.: Rodinia: a benchmark suite for heterogeneous computing. In: IEEE International Symposium on Workload Characterization (IISWC), pp. 44–54 (2009). https://doi.org/10.1109/IISWC.2009.5306797
8. Chen, J., et al.: HPDR: High-performance portable scientific data reduction framework. In: 2025 IEEE IPDPS, pp. 1104–1116. IEEE (2025). https://doi.org/10.1109/IPDPS64566.2025.00101
9. Chen, X., et al.: Fcbench: cross-domain benchmarking of lossless compression for floating-point data. Proc. VLDB Endowment **17**(6), 1418–1431 (2024). https://doi.org/10.14778/3648160.3648180
10. Collet, Y.: LZ4 - extremely fast compression (2025). https://lz4.github.io/lz4/. Accessed 20 Feb 2026
11. CSC – IT Center for Science: LUMI supercomputer (2025). https://www.lumi-supercomputer.eu/. Accessed 25 Feb 2025
12. Danalis, A., et al.: The scalable heterogeneous computing (SHOC) benchmark suite. In: Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU), pp. 63–74 (2010). https://doi.org/10.1145/1735688.1735702
13. Di, S., Cappello, F.: Fast error-bounded lossy HPC data compression with SZ. In: IEEE IPDPS, pp. 730–739 (2016). https://doi.org/10.1109/IPDPS.2016.11
14. Fletcher, R.P., Du, X., Fowler, P.J.: Reverse time migration in tilted transversely isotropic (TTI) media. Geophysics **74**(6), WCA179–WCA187 (2009)
15. Fomel, S., Sava, P., Vlad, I., Liu, Y., Bashkardin, V.: Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments (2013)

16. Google: Snappy: a fast compressor/decompressor (2025). https://github.com/google/snappy. Accessed 20 Feb 2026
17. Huang, Y., Di, S., Li, G., Cappello, F.: cuSZp2: a GPU lossy compressor with extreme throughput and optimized compression ratio. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE (2024). https://doi.org/10.1109/SC41406.2024.00021
18. Huang, Y., Di, S., Yu, X., Li, G., Cappello, F.: cuSZp: an ultra-fast GPU error-bounded lossy compression framework with optimized end-to-end performance. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–13 (2023)
19. Knorr, F., Thoman, P., Fahringer, T.: ndzip-GPU: efficient lossless compression of scientific floating-point data on GPUs. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–14. ACM (2021). https://doi.org/10.1145/3458817.3476224
20. Kurtzer, G.M., Sochat, V., Bauer, M.W.: Singularity: scientific containers for mobility of compute. PLoS ONE **12**(5), e0177459 (2017). https://doi.org/10.1371/journal.pone.0177459
21. Lindstrom, P.: Fixed-rate compressed floating-point arrays. IEEE TVCG **20**(12), 2674–2683 (2014). https://doi.org/10.1109/TVCG.2014.2346458
22. Liu, Q., et al.: Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks. Concurrency Comput. Pract. Experience **26**(7), 1453–1473 (2014). https://doi.org/10.1002/cpe.3125
23. Nicolae, B., Moody, A., Gonsiorowski, E., Mohror, K., Cappello, F.: VeloC: towards high performance adaptive asynchronous checkpointing at large scale. In: IEEE IPDPS, pp. 911–920 (2019). https://doi.org/10.1109/IPDPS.2019.00099
24. NVIDIA: Cascaded compression — nvCOMP documentation (2024). https://docs.nvidia.com/cuda/nvcomp/cascaded.html. Accessed 19 Mar 2025
25. NVIDIA Corporation: CUB: Cooperative primitives for CUDA C++ (2025). https://nvidia.github.io/cccl/cub/. Accessed 22 Feb 2026
26. NVIDIA Corporation: nvcomp: High-speed data compression library for nvidia GPUs (2025). https://developer.nvidia.com/nvcomp. Accessed 20 Feb 2026
27. Tian, J., et al.: cuSZ: an efficient GPU-based error-bounded lossy compression framework for scientific data. In: ACM PACT, pp. 3–15 (2020). https://doi.org/10.1145/3410463.3414624
28. TOP500: TOP500 list – November 2025. https://top500.org/lists/top500/2025/11/ (2025), 66th edn, published November, 2025. Accessed 20 Mar 2026
29. Zhang, B., et al.: FZ-GPU: a fast and high-ratio lossy compressor for scientific computing applications on GPUs. In: Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing, pp. 129–142. ACM (2023). https://doi.org/10.1145/3588195.3592994
30. Zhang, B., et al.: GPULZ: Optimizing LZSS lossless compression for multi-byte data on modern GPUs. In: Proceedings of the 37th ACM International Conference on Supercomputing, pp. 348–359. ACM (2023). https://doi.org/10.1145/3577193.3593706