# hrzip: Efficient Lossless Compression for High-Resolution Astronomical Data

Jian Lin*§, Kai Wu*§, Heshan Wang*, Yujun Su†, Kai Zhang‡, Bo Mao*, and Suzhen Wu*¶
*School of Informatics, Xiamen University, Xiamen, China
†Chinajey Intelligent System Co., Ltd, Ningbo, China
‡Inspur Data Technology Co., Ltd, Jinan, China
¶Corresponding Author: Suzhen Wu (suzhen@xmu.edu.cn)

***Abstract*—Ground-based telescopes are indispensable instruments for astronomical observation, requiring long-term continuous monitoring and high-resolution signal acquisition. These requirements pose considerable challenges for data storage and maintenance. Data compression is therefore critical, yet it remains difficult to achieve both high throughput and high compression ratios. Existing general-purpose compressors are not tailored for high-resolution astronomical data, while specialized astronomical compressors fail to fully exploit the redundancy inherent in such data, limiting their capacity to satisfy practical requirements for efficient storage.**

**To address these challenges, we propose hrzip, a lossless compressor designed for high-resolution astronomical data. We analyze the characteristics of high-resolution astronomical data and provide insight into the deficiencies of existing compressors when applied to such data. To address these issues, hrzip integrates three core modules: data rearrangement, value prediction, and entropy coding. Experimental results on the Five-hundred-meter Aperture Spherical radio Telescope (FAST) public datasets show that hrzip achieves a compression throughput of 610 MB/s and a compression ratio of 1.30 on floating-point datasets. hrzip also performs competitively on integer datasets, demonstrating superior overall compression performance compared with existing compressors for high-resolution astronomical data.**

***Index Terms*—Astronomical Data, Data Compression, High-Resolution Data, Encoding**

## I. Introduction

Astronomical data play an indispensable role in astronomical research, and new observational data often lead to the discovery of novel phenomena or underlying patterns. Astronomical observations explore the cosmos through ground-based telescopes, space-borne observatories, and specialized detectors. The raw signals collected by these facilities are calibrated and processed into structured datasets for scientific research. The parameters in these datasets are highly correlated, making their analysis in a high-dimensional parameter space particularly significant. Correspondingly, the volume of observational data is enormous, posing significant storage challenges.

For example, the Rubin Observatory can generate up to 20 TB of raw data per night [1]. Meanwhile, the space-based James Webb Space Telescope can downlink only 57 GB per day [2], forcing the rest to be discarded [3]. The yet-to-be-completed Square Kilometre Array (SKA) is expected to generate data at a rate exceeding 500 PB per year once fully operational [4]. According to the International Virtual Observatory Alliance [5], the volume of academic-grade astronomical data doubles every 6 to 9 months, which is a growth rate that significantly outpaces Moore's Law. If these data are compressed, storage usage can be greatly reduced [6], and redundant computational costs in scientific simulations can be avoided [7].

To contextualize these challenges, we focus on the Five-hundred-meter Aperture Spherical Radio Telescope (FAST) [8], which faces growing bottlenecks in data storage and transmission. The FAST project currently uses Zstandard (zstd), a widely adopted general-purpose compressor, in its storage system to mitigate storage pressure. However, general-purpose compressors inevitably overlook the inherent correlations in astronomical data induced by underlying physical laws, resulting in suboptimal throughput and compression ratios.

Existing astronomical data compressors suffer from fundamental limitations. First, they focus almost exclusively on image data and are ill-suited to time-series or spectral data [9]–[12]. Second, they are primarily designed for integer-valued data [13]–[15]. However, integer data represent only a fraction of astronomical datasets, thereby limiting the applicability of existing approaches. More importantly, although data volumes in this field are growing rapidly, studies on specialized compressors remain scarce. Although other scientific data compressors like zfp and fpzip can be applied to floating-point data, they fail to fully account for specific data characteristics and therefore cannot achieve optimal compression performance.

How to achieve strong compression performance on such time-series and frequency-domain data, represented by FAST data and characterized by high sampling rates rather than radio-frequency bands, remains an open challenge. Therefore, developing more specialized compressors is necessary.

In this paper, we propose hrzip, a lossless compressor specifically designed for astronomical data. Through a series of targeted optimizations, hrzip natively supports both integer and floating-point data while achieving the best overall compression performance on high-resolution astronomical data represented by FAST datasets. The main contributions of this paper are as follows:

- We focus on astronomical data generated by FAST and

§Jian Lin and Kai Wu contributed equally to this work

conduct an in-depth analysis of the differences between high-resolution astronomical data and conventional ones, with particular attention to the semantic differences across dimensions in different FAST datasets. Based on these dimensional characteristics, we propose a data rearrangement strategy that creates favorable conditions for subsequent compression.
- We propose a new dynamic floating-point processing strategy. This strategy not only preserves continuity after differencing, but also enables the subsequent compression stage to identify numerous pairs of values in astronomical data that have opposite signs but equal magnitudes. Although this method requires additional metadata, it eliminates the fundamental difference between integer and floating-point data in the subsequent compression stages, thereby providing native support for unified compression of both data types.
- We design an entropy coding scheme that combines robustness and efficiency. When the data follow a geometric distribution, its compression ratios approach the theoretical optimum of Golomb coding. Even when the data contain substantial noise, the compression performance does not degrade significantly. As a result, the encoder becomes more adaptive to data fluctuations and outliers while maintaining compression performance.
- We implemented a prototype of hrzip and conducted extensive experiments on multiple datasets from different FAST observing programs. hrzip improved compression ratios while maintaining high compression throughput by appropriately identifying and exploiting the characteristics of high-resolution astronomical data, thereby achieving the best overall performance. In addition, we carefully compared the behavior of different compressors in the experiments and analyzed the underlying reasons, providing new insights for future astronomical data compression.

The remainder of this paper is organized as follows. Section II provides the background and motivation for our work and introduces the astronomical datasets employed. Furthermore, Section III details the design of hrzip. We present the experimental setup and results in Section IV, and discuss related work in Section V. Finally, we conclude the paper in Section VI.

## II. Background & Motivation

In this section, we first present some technical background on data compression, then elucidate its potential value for high-resolution astronomical data, and finally discuss the overlooked requirements and data characteristics that existing compressors fail to address.

### A. Background

*1) Correlation Identification:* Correlation identification removes spatial, temporal, or structural redundancy from data by mapping raw data to abstract symbols or numerical representations. Real-world data inherently exhibit complex internal structures, and these regularities embedded in the data content constitute correlation redundancy. Most general-purpose lossless compressors eliminate repeated-pattern redundancy through dictionary matching, while some compressors use delta coding to directly exploit local correlations and remove redundancy through a prediction-residual mechanism. Given that these residuals usually follow a Laplace distribution or a geometric distribution and have much smaller magnitudes than the original data, coupling them with entropy coders (e.g., Golomb coding, which is mathematically optimized for geometric distributions) yields exceptional compression performance.

*2) Entropy Coding:* Entropy coding operates on abstract symbols or numerical representations and attempts to compress them according to specific rules. Entropy coding originates from information theory established by Claude Shannon in 1948, whose core insight is the fundamental connection between uncertainty and compressibility characterized through mathematical modeling. Shannon's information entropy formula $H(X) = -\sum_i p(x_i) \log_2 p(x_i)$, quantifies the average amount of information carried by each symbol of a discrete source and establishes the theoretical limit of lossless compression. As stated by Shannon's source coding theorem, for an ergodic source, the average code length $\bar{L}$, of any lossless coding scheme cannot be lower than the source entropy $H(X)$. When the source symbols follow a non-uniform distribution, the source contains statistical redundancy, which can be quantified as $1-\frac{H(X)}{\log_2 N}$, where $N$ is the size of the symbol alphabet.

By exploiting this skewed probability distribution, entropy coding employs variable-length codes to minimize the average bit-length toward the entropy limit, thereby achieving a compact representation without information loss. Specifically, entropy coding assigns short codewords to high-frequency symbols and long codewords to low-frequency symbols so as to minimize the expected code length.

*3) High-Resolution Astronomical Data:* High-resolution astronomical data acquired by radio telescopes are stored in the Flexible Image Transport System (FITS) format, which are further partitioned into multiple tiles for independent compression. FAST provides a representative example. Benefiting from the ultra-wideband reception capability of the world's largest single-dish radio telescope, FAST processes the received raw voltage data through a real-time digital signal processing system to perform channelization, polarization synthesis, and preliminary calibration, thereby generating two distinct types of standardized science data products for long-term archival storage. These two types of standardized science data products are optimized for different scientific objectives and exhibit two fundamentally different characteristics:

**Pulsar Search / Timing (PSR):** This mode is designed for millisecond pulsar searches and high-precision timing applications, and therefore preserves extremely high temporal resolution. Depending on the observation mode, the sampling interval ranges from microseconds to milliseconds. Each tile is represented as a three-dimensional array of 8-bit unsigned integers indexed by time, polarization, and frequency, with

```
XTENSION= 'BINTABLE'           / ***** Subintegration data  *****
BITPIX  =                    8 / N/A
NAXIS   =                    2 / 2-dimensional binary table
NAXIS1  =              8470604 / width of table in bytes
NAXIS2  =                  256 / Number of rows in table (NSUBINT)
PCOUNT  =                    0 / size of special data area
GCOUNT  =                    1 / one data group (required keyword)
TFIELDS =                   17 / Number of fields per row
TTYPE1  = 'TSUBINT '           / Length of subintegration
TFORM1  = '1D      '           / Double
TTYPE2  = 'OFFS_SUB'           / Offset from Start of subint centre
TFORM2  = '1D      '           / Double
......
TTYPE16 = 'DAT_SCL '           / Data scale factor for each channel
TFORM16 = '4096E   '           / NCHAN*NPOL floats
TTYPE17 = 'DATA    '           / Subint data table
TFORM17 = '4194304B'           / NBIN*NCHAN*NPOL*NSBLK int, byte(B) or bit(X)
......
TDIM17  = '(1,2048,2,1024)'    / Dimensions (NBITS or NBIN,NCHAN,NPOL,NSBLK)
TUNIT17 = 'Jy      '           / Units of subint data
EXTVER  =                    1 / auto assigned by template parser
```

Fig. 1: Metadata structure of a FAST integer dataset.

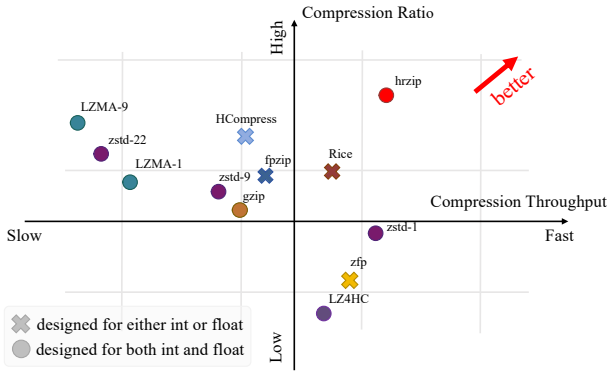


Fig. 2: Applicability of existing lossless compressors to high-resolution astronomical data.

singleton dimensions omitted. The data are dominated by a continuous noise background with random fluctuations, mainly arising from the cosmic microwave background and instrumental thermal noise. In contrast, transient foreground signals appear as high-intensity spikes of very short duration, corresponding to astronomical events of interest.

**Spectroscopy / Continuum (SPEC):** This mode prioritizes frequency resolution and is intended for spectral-line surveys and continuum imaging studies. Each tile is represented as a two-dimensional array of 32-bit IEEE 754 floating-point values, containing integrated measurements over a given time window across the frequency and polarization dimensions. The frequency resolution is typically on the order of several hundred hertz. The data vary smoothly along the frequency axis on a global scale, while spectral-line regions may contain fine spectral structures associated with either astronomical signals or noise.

Metadata for high-resolution astronomical data reside in the header of FITS files and are recorded as ASCII text in the form of keyword-value pairs, with each pair occupying 80 bytes. As exemplified by an integer-data FITS file from FAST (Fig. 1), the NAXIS fields define the number of rows and columns in the binary table, TFORM17 specifies the size and data type of the primary observational data, and TDIM17 indicates the information of dimensions of the primary observational data. Based on this information, a program can extract the logical structure of the data.

### B. Motivation

*1) Challenges in Lossless Compression of Floating-Point Data:* Existing astronomical data compressors are primarily designed for astronomical images and therefore cannot be directly applied to high-resolution data. Moreover, most of these compressors provide little or no support for floating-point data. Specifically, astronomical data processing is highly specialized and lacks unified, standardized data formats. Unlike multimedia data with regular structures, such as PNG and MP4, astronomical data comprise diverse forms, including images, spectra, and time series [16], each with distinct structural characteristics. In addition, the underlying data types vary widely, ranging from 8-bit unsigned integers to 64-bit floating-point values. As a result, existing specialized compressors are often unsuitable for high-resolution astronomical datasets, forcing floating-point datasets to rely on general-purpose or scientific data compressors instead.

Moreover, floating-point data are more commonly compressed using lossy schemes. However, not all astronomical research scenarios can tolerate information loss, especially for extreme cases where no errors are allowed and the decimal places are uncertain [17]. Once data undergo lossy compression, they do not retain their original values, which inevitably reduces the research utility and academic credibility of the data. The archival specifications of FAST explicitly require that the original observational data be preserved without any modification. This further restricts the applicability of existing compressors.

Fig. 2 illustrates the applicability of various lossless compressors to high-resolution astronomical data. The visualization indicates a lack of established lossless compression solutions specifically tailored for high-resolution astronomical data, particularly concerning floating-point formats, which is the domain that hrzip aims to address.

*2) Room for Improvement in Compression Ratios and Throughput:* We hypothesize that high-resolution astronomical data compression still has substantial room for improvement in the trade-off between compression ratio and throughput. Through targeted analysis and optimization, it will be possible to improve both in practical implementations. To validate this hypothesis, we conducted several experiments in a consistent hardware and software environment as shown in Table I, where all evaluation code was compiled with `-O3` and `-march=native` flags to enable aggressive and architecture-specific compiler optimizations.

To investigate the performance of existing compressors on high-resolution time-series and frequency-domain data collected by FAST, we selected several representative compressors: (1) gzip 1.10 [18], a widely used general-purpose compressor; (2) LZ4 1.9.3 [19], known for its exceptional speed; we also included its high-compression variant, LZ4HC; (3) LZMA 5.2.5 [20], a compressor designed for high com-

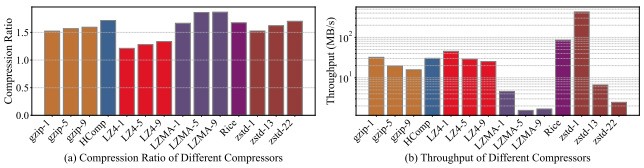


Fig. 3: Existing compressors' compression benchmark on FAST integer datasets.

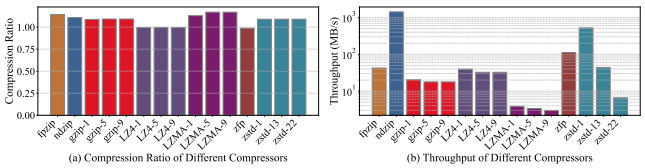


Fig. 4: Existing compressors' compression benchmark on FAST floating-point datasets.

TABLE I: Experimental Environment Configuration

| Category | Item | Specification |
|---|---|---|
| Hardware | Server Architecture | Single-node Server |
| | CPU Model | Intel Xeon Gold 5318Y |
| | Base Frequency | 2.10 GHz |
| | Cores / Threads | 24 Cores / 48 Threads |
| | Memory Configuration | 4 × 32GB DDR4-3200 RDIMM |
| | Memory Channel | Quad-Channel |
| Software | Operating System | Ubuntu 22.04.1 LTS |
| | Kernel Version | Linux 5.15.0-151-generic |
| | Compiler | g++ 11.4.0 |
| | Language Standard | ISO C++17 |
| | Compilation Flags | `-O3, -march=native` |

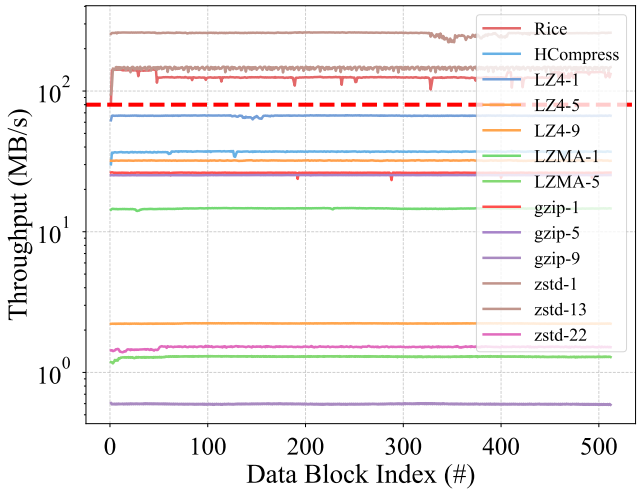


Fig. 5: Real-time compression throughput of existing compressors.

pression ratios; (4) zstd 1.4.8 [21], recognized for its excellent overall performance. These compressors are installed directly via the `apt` package manager to obtain their respective C development libraries.

Furthermore, we evaluated specialized compressors for astronomical and other scientific data. For integer data, we used Rice and HCompress, two widely used astronomical data compressors, both implemented in CFITSIO (v4.5.0). For floating-point data, we additionally evaluated ndzip [22], fpzip [23] and zfp [24] in the comparison, as they are representative compressors in scientific data compression. These specialized compressors are highly optimized for the characteristics of high-dimensional scientific data and serve as important domain-specific baselines for astronomical data compression.

Compressor parameter settings are specified as follows. For compressors with adjustable compression levels, we evaluated representative levels. Specifically, we tested the minimum compression level, which prioritizes throughput; the maximum compression level, which favors compression ratio; and an intermediate level, which balances throughput and compression ratio. For the Rice compressor, the block size was set to 64. For the HCompress compressor, we merged the second and third dimensions of each tile so that the data could be compressed as a 2D image. Before calculating the compressed size, we also removed any header and footer information from the output to obtain the raw output size.

To evaluate the compression performance and robustness of existing compressors in real-world astronomical observation scenarios, we conducted systematic benchmarking on several publicly released project datasets from FAST. The detailed specifications of the selected datasets are summarized in Table II, covering various typical observation modes. Specifically, the Dimensions field denotes the dimensional structure of the data tiles and provides the key information used by the compressor to optimize its tiling strategy.

Our tests focus on the performance of various compression algorithms under single-threaded conditions. This is because a single FITS file typically consists of thousands of mutually independent tiles, and the compression process is carried out tile-by-tile. As a result, parallelism is relatively easy to implement in clusters, while the main bottleneck lies in the compression efficiency of individual tiles.

In our experiments, we evaluated the compressors from two perspectives: compression ratio and throughput. Throughput is defined as the volume of uncompressed data processed per unit of time, representing the input bandwidth during compression and the output bandwidth during decompression. Unless otherwise stated, both compression and decompression were executed using a single thread in this study. The compression ratio $CR$ for a single data tile is defined as the ratio of the original data size to the compressed data size: $CR = \frac{S_{orig}}{S_{comp}}$, where $S_{orig}$ and $S_{comp}$ denote the sizes of the data before and after compression, respectively. A higher $CR$ indicates better compression performance. The overall compression ratio of a given dataset is defined as the geometric

TABLE II: Detailed specifications of the FAST datasets used for evaluation.

| Source Name | Backend | Observation Mode | Data Size | Dimensions | Data Type |
|---|---|---|---|---|---|
| J1713+0747 | psr(4k) | Tracking | 1.13 TB | (1024,4,4096) | uint8 |
| FRB20190107A | psr(2k) | SnapShot | 2.19 TB | (1024,2,2048) | |
| J1903+0433 | psr(2k) | SwiftCalibration | 3.10 TB | (1024,4,2048) | |
| J0534+2200 | psr(1k) | SwiftCalibration | 449 GB | (1024,4,1024) | |
| FUDS4_RA | spec (W+N) | OnTheFlyMapping | 166 GB | (65536,4) | float32 |
| GAMA15_03 | | DecDriftWithAngle | 213 GB | | |
| NSA75400 | | OnOff | 1.21 TB | | |
| NGC2841_OTF1 | | MultiBeamOTF | 173 GB | | |

mean of the compression ratios across all its data tiles, whereas the overall throughput is calculated as their arithmetic mean. This approach ensures a balanced evaluation of algorithmic performance across datasets with diverse characteristics and prevents the results from being dominated by the data features of exceptionally large files.

Before testing, we can already observe that scientific data compressors often lack compatibility with both floating-point and integer data. The Rice algorithm cannot directly compress floating-point numbers, while zfp and fpzip are not designed for integer compression. Moreover, specialized astronomical data compressors have been phased out due to their limited applicability. This already reveals the shortcomings of existing specialized compressors in this domain.

Figs. 3 and 4 demonstrate the performance of existing compressors on FAST datasets. We found that LZMA-9 achieves the highest compression ratio, but its throughput is too low for practical use. Compared to LZMA-9, other compressors prioritize higher throughput at the expense of compression ratio, exhibiting varying degrees of this trade-off. Even among general-purpose compressors, similar trade-offs exist. For example, zstd offers compression levels from 1 to 22, with lower levels producing higher throughput and lower compression ratios. And zstd-1 also achieves the highest throughput among all tested compressors. This observation suggests that the trade-off between compression ratio and throughput can still be substantially improved for high-resolution astronomical data compression.

Another noteworthy finding is that several specialized compressors, including ndzip, fpzip, Rice, and HCompress, can substantially outperform general-purpose compressors in compression performance. ndzip achieves excellent throughput mainly because it adopts lightweight prediction/differential encoding combined with efficient bit-level compression, which suggests that designing lightweight and distinctive compressing schemes can effectively improve the performance of compressor. However, this is not universally true: for example, zfp performs worse. Although exploiting the statistical properties of astronomical data can improve compression, high-resolution astronomical data also possess distinctive characteristics that require careful treatment to prevent performance degradation.

To further analyze whether existing compressors can meet practical throughput requirements, we selected an integer dataset, extracted 2 GB of contiguous data, and partitioned it into equal-sized blocks according to the requirements of each compressor. We calculated real-time throughput by measuring the time it took each compressor to compress each block. We then used the metadata in the sample's header to obtain the size and sampling duration, from which we computed the sample's throughput. The throughput of this sample was used as the throughput requirement that compressors must satisfy, represented as a dashed line in the figures.

The dataset we chose is generated from FAST observations of the pulsar J0534+2200, producing 4 MB of data every 0.05 seconds. Therefore, to achieve real-time compression, the compressor must achieve a throughput exceeding 80 MB/s.

The results are shown in Fig. 5. Most compressors did not meet the throughput requirements for this dataset, with zstd-1, zstd-13, and Rice being the only exceptions. However, these three compressors' gain in throughput comes at the cost of a substantially lower compression ratio compared to LZMA-9, suggesting that existing compressors are generally unable to meet the high bandwidth demands of astronomical data.

*3) Dimensional Variability:* In our analysis of FAST astronomical data, we observed that variability along higher dimensions is relatively smooth, whereas values along lower dimensions—stored contiguously on disk—appear more random. In fact, in instruments and simulations in other scientific fields [25]–[27], data patterns often exhibit a high degree of spatial smoothness, which means that spatially adjacent data points often have similar values [28]. To quantify how dimension ordering affects entropy, we conducted further tests. Specifically, we (1) selected two typical types of high-resolution data from the FAST alongside conventional astronomical images from the WISE [29] telescope; (2) extracted a representative sample of approximately 16 MB from each dataset, flattened and partitioned the array along each dimension into fixed-length blocks of roughly 1024 consecutive values, ensuring that the block volumes are similar across different dimensions and that the partitioning does not cross the dimensional boundaries of the data matrix; (3) computed the bit-level zero-order entropy for each block to measure its internal compressibility. Specifically, for a data type with a bit-width of $n$, if the proportion of the $i$-th bit being 1 across

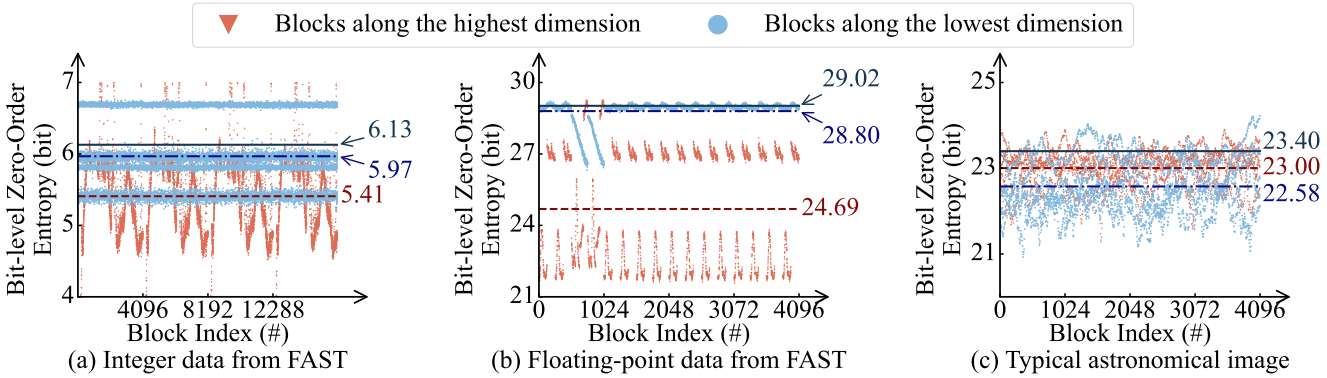


Fig. 6: Dimensional variability in astronomical datasets.

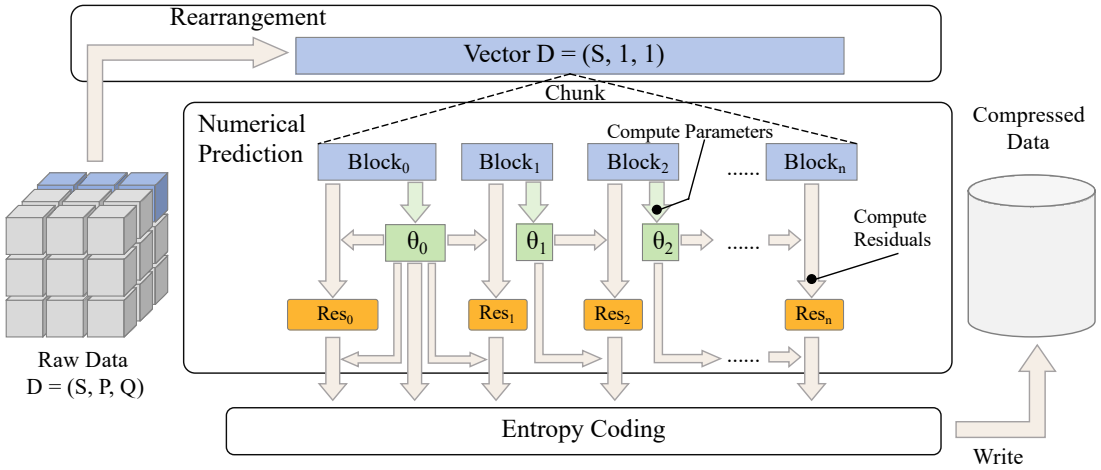


Fig. 7: The overall workflow of hrzip.

all elements in the block in binary representation is $p_i$, then the bit-level zero-order entropy of the block is defined as:

$$\sum_{i=1}^{n} H(p_i) = \sum_{i=1}^{n} -p_i \log_2 p_i - (1-p_i)\log_2(1-p_i)$$

As shown in Fig. 6, the entropy characteristics of FAST data differ from those of conventional astronomical images. In the figure, the solid lines represent the bit-level zero-order entropy calculated by treating the entire dataset as a single compression unit. In contrast, the dotted and dashed lines correspond to the statistical mean of the bit-level zero-order entropy for compression units partitioned along the dimensions with the lowest and highest rates of change, respectively. The experimental results clearly demonstrate that the compressibility of FAST data exhibits a strong dimensional correlation, being far more influenced by the dimensional structure than conventional astronomical images. For the two types of FAST data, since each dimension is fundamentally distinct, the average entropy of data blocks expanded along different dimensions also varies significantly: in integer-based data, the entropy difference between the two dimensions with the greatest disparity is approximately 10%, while in floating-point data, this difference reaches as high as 17%. Conversely, for conventional astronomical image data—which consists of two statistically similar spatial dimensions—the entropy variation caused by different partitioning directions is only about 2%, which is practically negligible, reflecting its insensitivity to dimensional selection.

We further applied this approach to high-resolution astronomical data from the Atacama Large Millimeter/submillimeter Array (ALMA) archive [30]. Specifically, we selected the ALMA observations of NGC_7319E as a representative external dataset for evaluation. The results show that the disparity reaches 8.5%, further indicating the widespread presence of dimensional variability in this type of data.

Beyond the pronounced non-uniform rates of change across different dimensions, we further discover that FAST data typically exhibit lower rates of change in higher-dimensional directions. In fact, this pattern is not unique to FAST; it is prevalent in the raw observational data from radio and other high-resolution scientific instruments, including waveform records from high-energy physics detectors and raw data from various other telescopes that remain archived but not publicly accessible. The essence of this phenomenon stems from the engineering trade-off between high-resolution multi-dimensional sampling modes and storage efficiency. To

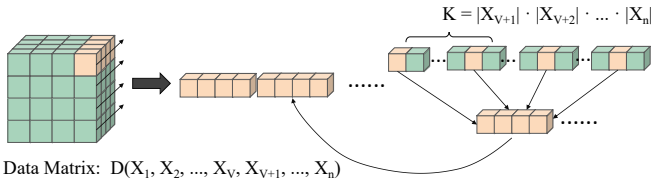


Fig. 8: The workflow of data rearrangement in hrzip.

meet massive real-time data writing requirements, systems typically store data sequentially on physical media according to the acquisition order. Consequently, data points that are adjacent in storage space often correspond to the dimensions where the signal changes most rapidly. Although this storage layout facilitates real-time recording, it does not fully exploit the redundancy in higher dimensions, thereby limiting the performance of traditional compression methods. Therefore, if dimensions with lower rates of change can be reordered to contiguous storage positions through data reorganization or dimensional transformation, the local correlation of the data can be significantly enhanced, thereby improving the compression ratio of the compressor.

However, existing astronomical data compressors are image-oriented: they neither exploit dimension permutation as a compression strategy nor are they suited for compressing data along higher-dimensional directions. When we modified Rice by incorporating this dimension permutation, it yielded only an additional reduction of about 0.1%, demonstrating that the scheme cannot capture the marginal entropy gains. We therefore set out to redesign a compressor that can achieve substantially higher ratios on high-resolution data sets like those from FAST.

## III. Design

In this section, we introduce hrzip, a compressor designed for high-entropy, high-resolution astronomical data that supports both integer and floating-point formats. By incorporating the statistical characteristics inherent in high-resolution astronomical observations, hrzip aims to deliver higher compression ratio while maintaining high throughput.

The hrzip architecture comprises three primary modules: data rearrangement, value prediction, and entropy coding. Fig. 7 illustrates the overall workflow of hrzip.

The data rearrangement module converts multi-dimensional astronomical data into a one-dimensional layout based on the dimension with the lowest data variability. Additionally, hrzip rearranges floating-point bits using the entropy of their sign bits. Like other specialized compressors, hrzip splits the data into fixed-length blocks and compresses each block independently.

The value prediction module exploits the physical locality of elements along the temporal or frequency dimension within each block. It forms residuals by subtracting the predicted values from the actual observations. Compared with the raw data, these residuals typically exhibit a smaller dynamic range, making them more amenable to entropy coding.

The entropy coding module then targets the lower variability of the residual sequence and encodes the residuals using an Exp-Golomb-like prefix code. In this way, the residual sequence is encoded into a compact bit representation by exploiting its statistical redundancy, achieving high compression ratios for large-scale data without sacrificing computational efficiency.

In addition, as both data rearrangement and entropy coding require their own metadata, hrzip incorporates a metadata management scheme based on the assumption of local smoothness in high-resolution astronomical data. By leveraging compression parameters from the preceding block to derive metadata for the current block, this scheme aims to minimize the storage and throughput overhead associated with metadata management.

### A. Data Rearrangement

*1) Dimensional Permutation:* To utilize the inherent characteristics of high-resolution astronomical data and identify more correlations, hrzip groups data along the dimension with the lowest variability, which is determined by the application layer based on FITS header metadata, and rearranges the data if necessary. The process is illustrated in Fig. 8.

Our analysis indicates that, for a given category of astronomical data, the dimension with the lowest data variability is generally consistent. For PSR data, this corresponds to the temporal dimension, as the physical characteristics of celestial radio signals remain relatively stable within short sampling windows ranging from microseconds to milliseconds. For SPEC data, this dimension is the frequency one, because the bandpass response of the receiver system varies slowly within a single integration period at a resolution of several hundred hertz. Across both data types, the dimension with the lowest variability is typically the highest logical dimension of the data structure.

Leveraging these observations, hrzip adopts a static strategy that prioritizes the highest logical dimension: during dimension reordering, the highest logical dimension is mapped to the contiguous storage layout by default, while the relative order of the remaining dimensions is preserved. This simplification avoids the computational overhead of calculating dimensional variability and mitigates the risk of dimension misidentification potentially induced by sampling noise in dynamic strategies, thereby ensuring stable compression performance. In practice, hrzip does not physically transpose the entire matrix. Instead, it computes the position of each element along the target dimension and reads them sequentially to form a vector in memory, which is equivalent to performing compression along an arbitrary dimension.

*2) Blocking:* To arrange the data more effectively, hrzip divides the vector generated in the previous step into multiple blocks. A block is the smallest independently encoded unit in hrzip. We do not use the hash-based long-distance matching technique commonly used in general-purpose compressors because identifying long-range correlations is computationally expensive, and the exploitable correlations in FAST data are

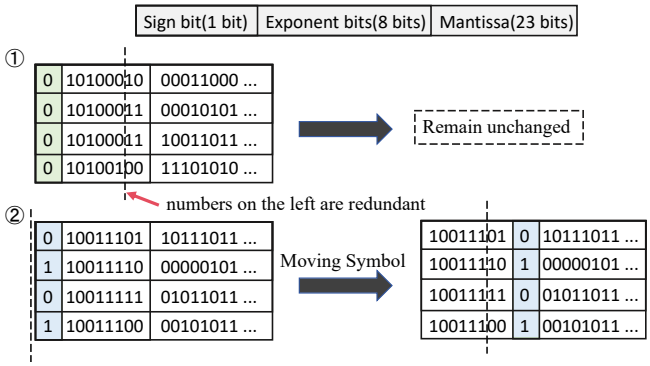


Fig. 9: Explanation of the dynamic floating-point sign placement strategy.

limited. Currently, we set the block size $B$ to 64 elements, which, as will be demonstrated in the subsequent evaluation section, provides good compression performance across datasets with different characteristics.

Meanwhile, to ensure that data blocks align with dimensional boundaries, the initial block may be slightly longer than the remaining blocks. Specifically, for a dimension with a contiguous length $L$, the initial block is configured to contain $B+(L\%B)$ elements, so that the remaining elements are divisible by $B$. This design is adopted because, as explained later, the initial block is processed differently from the subsequent blocks.

*3) Dynamic Floating-point Sign Placement:* After numerical matrices are rearranged and partitioned into blocks, we seek to enable 32-bit floating-point data to be compressed in the subsequent stages in much the same way as 32-bit integers, with minimal modification to the overall pipeline. For floating-point numbers in the IEEE 754 standard, their binary representation can be directly compressed as integers. However, their sign, exponent, and mantissa bits are stored in separate bit fields. This induces abrupt numerical changes when two floating-point values are close in magnitude but differ in sign across zero. As a result, the residuals produced by delta coding may exhibit large jumps, which severely violates the local smoothness assumption required for effective differential prediction.

Existing static solutions are not well suited to astronomical data compression. fpzip flips all bits of negative values and the sign bit of positive values. However, this method cannot capture redundancy between values with opposite signs but similar magnitudes, a pattern frequently observed in astronomical datasets. In contrast, although ndzip relocates the sign bit to the least significant bit (LSB) position, it does not explicitly exploit the intrinsic redundancy of the sign bits themselves.

We observe that when the signs of the data remain consistent, the values do not cross the zero boundary and can therefore be compressed directly as integers with favorable effectiveness. In contrast, when positive and negative signs are mixed, it becomes necessary to adopt the bit-field remapping idea, and relocate the sign bit to avoid the aforementioned issue. Motivated by this observation, hrzip dynamically selects a sign bit handling strategy based on the sign-consistency characteristics of each data block. As shown in Fig. 9, for a given floating-point data segment, all elements uniformly adopt one of the following two sign-placement strategies:

- Strategy 1: Maintaining the sign bit in its original position. When the signs within a data segment are highly consistent and values rarely cross zero, the delta coding operation can automatically eliminate the statistical redundancy of the sign bit. The occasional residual spikes caused by zero crossing appear only as sparse outliers, and the resulting entropy increase can be effectively tolerated within hrzip's robust coding framework.
- Strategy 2: Swapping the positions of the sign bit and the exponent field. When the sign bits are randomly distributed, hrzip moves the sign bits to the boundary between the exponent and mantissa fields, allowing delta coding to preserve part of the sign-related correlation. This placement is a trade-off after balancing the entropy of different bit positions: the entropy of the sign bit is typically higher than that of most exponent bits, but lower than that of the most significant mantissa bit (MSB). This layout not only isolates sign-induced discontinuities, but also preserves magnitude symmetry.

hrzip selects the sign-placement strategy by first scanning each data segment and storing the choice as a one-bit metadata flag for decompression. The decision is based on the minority-sign ratio $p$. When $p$ is small, Strategy 1 is clearly preferable; otherwise, Strategy 2 should be adopted. Therefore, we need to determine a threshold $P$ for the minority-sign ratio: use Strategy 1 when $p < P$, and Strategy 2 when $p > P$. We empirically set $P = 13\%$ based on sensitivity analysis, and the effect of this choice is examined further in the evaluation section.

### B. Data Prediction

After data rearrangement and blocking, hrzip proceeds to the data prediction stage. The core objective of this stage is to leverage the statistical correlations in the data to transform the original values into a residual sequence, thereby reducing the entropy and improving compression performance.

Astronomical data typically exhibit strong spatial and temporal correlations, meaning that neighboring samples often carry highly redundant information. Therefore, after applying predictive coding or differential transformation, the resulting residuals between adjacent samples are usually much smaller than the original values. Moreover, these residuals tend to follow a sharply peaked distribution centered around zero. This decorrelation process substantially lowers the entropy of the data, thereby enabling the subsequent entropy coding stage to achieve higher compression ratio.

Due to the extremely high sampling rate of data from FAST and the relatively small block size adopted by hrzip, the signal within a single data block can be regarded as fluctuating slightly around a constant value. In other words, the signal can be modeled as a locally stationary process comprising a

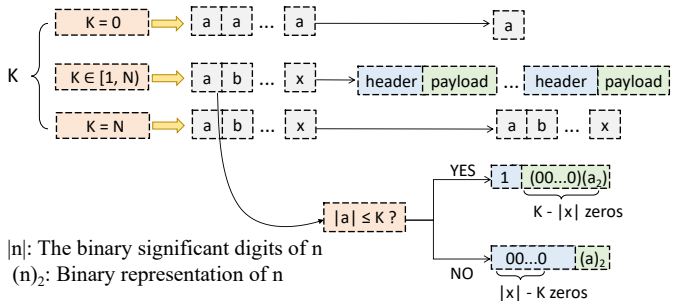


Fig. 10: Explanation of the entropy coding strategy.

constant component superimposed with low-amplitude high-frequency noise. Therefore, hrzip adopts a single predictor for the entire block. Observations show that using the median of the elements as the constant predictor provides the best prediction accuracy. However, because computing the median is relatively expensive and its high computational complexity substantially reduces throughput, we use the mean $\mu$ as an approximation instead, calculated as the average of the remaining elements after excluding one maximum and one minimum value.

For the signed residual sequence obtained after differencing, an unsigned mapping is required before Golomb coding, as Golomb codes are defined only for non-negative integers. Considering that the differencing function is invoked frequently in the compression pipeline, hrzip adopts the ZigZag mapping, which converts signed residuals into unsigned integers by assigning zero to zero, negative values to odd integers, and positive values to even integers. This matches the near-zero residual distribution of high-resolution astronomical data.

### C. Entropy Coding

hrzip uses a prefix code to encode the residual sequence. This code is similar to Exp-Golomb encoding and is referred to as M-Exp-Golomb encoding. The prefix code requires a parameter $k$. During encoding, if the effective bit-width $n$ of the current value does not exceed $k$, the value is represented by a single bit 1 followed by $k$ data bits. If the effective bit-width exceeds $k$, the encoder first writes $n-k$ leading zeros to indicate the excess length, and then appends the actual binary representation of the value.

Experience shows that when the data follow a geometric distribution, M-Exp-Golomb's performance is slightly inferior to that of Golomb coding. However, in the presence of substantial noise, it outperforms Golomb coding. This is because, although M-Exp-Golomb and Exp-Golomb share a visually similar structure based on counting leading zeros and a delimiter bit, Exp-Golomb's code length grows linearly with the value magnitude. As a result, when facing the unusually large noise values that frequently occur in astronomical data, Exp-Golomb is less robust and can easily suffer from codeword expansion and a sharp drop in coding efficiency.

Another notable advantage of this coding scheme is that its optimal parameter $k$ can be computed efficiently without making any prior assumptions about the data distribution. Specifically, for a sequence with effective bit-widths up to $N$, let $c_t$ denote the number of values whose effective bit-width is $t$. Then, the total code length is given by:

$$S_k = (k+1)\sum_{t=0}^{k} c_t + \sum_{t=k+1}^{N} (2t-k)c_t$$

To determine when increasing k shortens the encoding length, consider the condition $S_{k+1} < S_k$, which means:

$$\sum_{t=0}^{k} c_t < \sum_{t=k+2}^{N} c_t$$

Therefore, the optimal k can be determined with just one lightweight scan.

Moreover, to further improve compression performance under extreme statistical characteristics, hrzip introduces a metadata-specialized mechanism for two extreme scenarios: constant-valued data and near-random data. Under the M-Exp-Golomb coding framework, the theoretical range of the parameter $k$ is $[1, N]$, where $N$ denotes the data bit-width. However, this constraint reveals inherent limitations under two boundary conditions.

First, when all elements in a block are identical, the residuals are always zero. Even at the minimum value of $k = 1$, the scheme still assigns one bit to each element, and therefore cannot reach the theoretical zero-entropy limit.

Second, when the data approach complete randomness, the optimal parameter becomes $k = N$. In this case, standard encoding requires $N + 1$ bits to represent each $N$-bit data. This introduces a one-bit expansion per element, which is counterproductive for data compression.

To address these boundary cases, hrzip redefines the coding semantics of the parameter $k$. For near-random data, the meaning of $k = N$ is modified: when $k = N$, hrzip bypasses the entropy-coding stage and directly reads and writes block elements using their original $N$-bit representation. This not only avoids the compression degradation introduced by $(N + 1)$-bit codewords, thereby ensuring that the worst-case compression ratio approaches 1, but also significantly improves data throughput by removing the encoding computation overhead.

For constant-valued data, hrzip extends the semantics of $k = 0$. In this case, all elements are identical, and hrzip stores only a single representative sample, while the remaining elements consume no additional bits, thereby achieving the theoretical zero-entropy limit for constant-valued blocks.

The overall process is illustrated in Fig. 10.

### D. Metadata Management

Traditional approaches require a first pass to determine the optimal parameter for the current data block, followed by a second pass to encode the block using that parameter. The selected parameter is then stored as metadata and prepended to the compressed data to guide decoding.

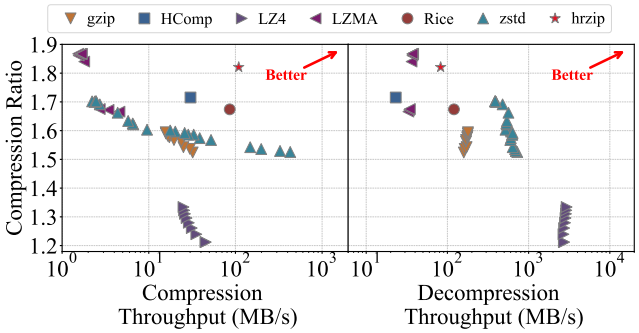


Fig. 11: Compression and decompression performance comparison on FAST integer datasets.

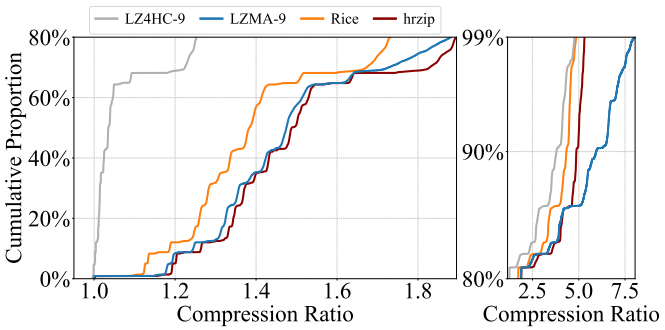


Fig. 12: Compression ratio CDF comparison on FAST integer datasets.

This approach can introduce significant metadata overhead when dealing with high-entropy data, and may even result in the compressed data being larger than the original, which defeats the purpose of compression. Therefore, hrzip introduces a novel metadata management strategy that significantly reduces metadata overhead and improves compression performance through two key innovations: parameter reuse and dynamic sign bit placement.

*1) Parameter Reuse:* The first block is processed using the conventional two-pass procedure, while each subsequent block is compressed directly using the optimal parameters derived from the previous block. We adopt this approach for two main reasons:

**Improved compression ratio:** The optimal parameter for any given block is very similar to that of the next block. Therefore, parameter reuse not only introduces negligible compression loss within each block, but also eliminates the need to store parameters for all subsequent blocks. Although the parameters account for only about 3% of the uncompressed size, their absolute storage cost remains considerable given that astronomical data are inherently difficult to compress.

**Optimized throughput:** This strategy makes it possible to compute the optimal parameter for the next block in parallel while encoding the current block. Such parallelism provides the compiler with more opportunities to hide memory access latency and branch prediction costs, thereby significantly improving the overall compression throughput of the algorithm.

*2) Adaptation to floating-points:* To support both floating-point strategies introduced in Section III-A3, we introduce an additional flag bit to indicate the placement of the sign bit. Similarly, for the first block, this flag is encoded as metadata, while for each subsequent block, the placement of the sign bit is determined by the optimal choice derived from the data of the previous block. In this way, we can support both strategies with minimal overhead, thereby better exploiting correlations and improving compression ratios.

## IV. Evaluation

In this section, we evaluate hrzip and the baseline compressors on the FAST public datasets, and analyze their overall performance and resource usage.

### A. Experimental environment and test data

To evaluate the compression performance and robustness of hrzip in real-world astronomical observation scenarios, we continued to use the compressors introduced in Section II-B2 and the datasets listed in Table II. For the reason stated in Section II-B2, we tested the performance of various compressors under single-threaded conditions. These compressors span both general-purpose and specialized data compression and the datasets cover multiple common scenarios, providing comprehensive coverage for evaluation.

The hardware and software configurations remained consistent with those listed in Table I. Regarding compression algorithm parameters, we evaluated **all available compression levels** for general-purpose compressors. For all levels of LZMA and level 12–22 of zstd (which exhibit extremely low throughput), results were extrapolated by sampling one out of every 32 tiles. The parameter settings for the Rice and HCompress remained consistent with those in Section II-B2, as did the definitions of compression ratio and throughput.

### B. Performance Testing and Analysis

Fig. 11 illustrates the compression ratios and throughput of various compressors on integer datasets. hrzip achieves a competitive trade-off between compression ratio and throughput, providing the best overall performance. Although LZMA remains the benchmark for compression ratio, reaching a peak of 1.87, the cost of this extreme performance is substantial: the compression throughput of LZMA-9 (LZMA at compression level 9) is only 1.68 MB/s, potentially constraining its use in real-time processing scenarios. In contrast, while achieving a high compression ratio of 1.82 (second only to LZMA), hrzip maintains a compression throughput of 109 MB/s. This represents an improvement in throughput of approximately 65× at the cost of only a 3% gap in the compression ratio. As a representative of modern general-purpose compressors, zstd offers a wide range of compression levels (1-22) to cover diverse application requirements; its maximum compression ratio is approximately 1.70, and its overall performance shown in the figure can be considered a superior alternative to gzip. Notably, even at its highest compression level, zstd's compression ratio remains inferior to that of hrzip, with its

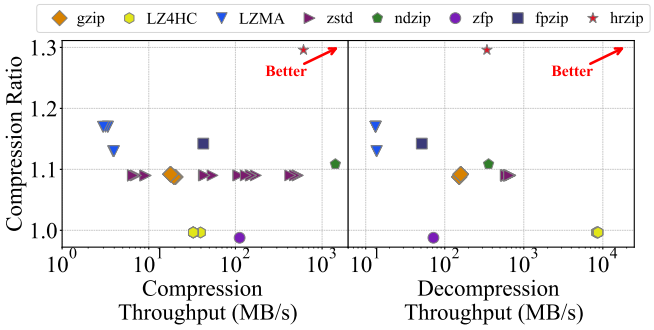


Fig. 13: Compression and decompression performance comparison on FAST floating-point datasets.

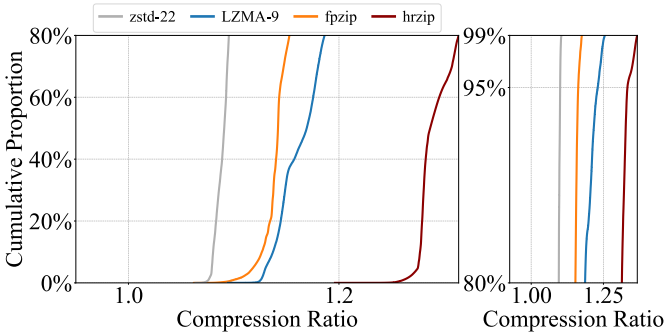


Fig. 14: Compression ratio CDF comparison on FAST floating-point datasets.

throughput dropping to 2.46 MB/s—approaching the level of LZMA-9. Meanwhile, hrzip maintains a throughput advantage of dozens of times over zstd while delivering higher compression. By incorporating domain-specific optimizations, hrzip addresses this bottleneck for astronomical data, yielding a more favorable performance profile than the evaluated general-purpose baselines.

During decompression, hrzip exhibits lower throughput compared to its compression. During compression, parameter calculation and entropy encoding can be executed concurrently; however, during decompression, the decoder must wait for the current block to be fully decoded before deriving the parameters for the subsequent block. This serial bottleneck restricts parallelization, thereby limiting CPU utilization. Nevertheless, in astronomical data processing, compression typically faces more stringent real-time constraints and higher throughput requirements than decompression. Consequently, we consider this trade-off to be acceptable within the context of the target application.

Fig. 12 further illustrates the cumulative distribution function (CDF) of the compression ratios for selected high-performance compressors across all integer dataset tiles. Except for a small fraction of low-entropy tiles which LZMA can exploit to achieve a higher overall compression ratio than hrzip in those specific cases—hrzip outperforms other general-purpose and specialized compressors on the majority of high-entropy, low-compressibility tiles. This superior performance is attributed to LZMA's lack of specialized optimization for data with high entropy, whereas hrzip is designed to handle such characteristics effectively. We also conducted the same test as in Fig. 5 on hrzip. The results show that hrzip achieves a minimum real-time compression rate of 87.2 MB/s and an average throughput of 111.4 MB/s, further validating its advantage over existing compressors in terms of compression throughput.

Fig. 13 illustrates the compression ratios and throughputs of various compressors on floating-point datasets. In the comprehensive performance evaluation using the floating-point dataset, hrzip significantly outperforms all baseline compressors in both compression ratio and throughput, establishing a distinct Pareto frontier. Specifically, with a compression ratio of 1.30 and a throughput of 610.39 MB/s, hrzip comprehensively surpasses competitors such as zstd (1.09, 523.83 MB/s) and fpzip (1.14, 42.56 MB/s). The performance of fpzip may be limited by its Lorenzo predictor [31], which utilizes neighboring values across all dimensions. As the successor to fpzip, ndzip retains the advantage of high throughput but also inherits a similar data prediction problem. Given that variability differs across dimensions, incorporating data from high-variability directions can paradoxically degrade prediction accuracy. Furthermore, we observed that zfp yields a compression ratio of less than 1.0 on floating-point datasets. This occurs because zfp relies heavily on local smoothness and transform-domain compactness. When floating-point data exhibit high entropy and weak correlation between adjacent elements, the energy of the coefficients fails to concentrate after zfp's block transformation, making bit-plane encoding ineffective at removing redundancy. When combined with the overhead of block metadata and encoding, this may ultimately result in the compressed data size exceeding the original size. In contrast, our M-Exp-Golomb encoding accounts for such high-entropy scenarios. By selectively bypassing the entropy encoding stage and directly reading/writing data blocks using the original $N$-bit width, our method prevents data expansion and ensures encoding robustness.

General-purpose compressors achieve only limited compression ratios on floating-point data. Specifically, the compression ratios of all evaluated general-purpose compressors consistently fall between 1.09 and 1.17, and increasing the compression level yields little to no improvement. For LZMA, the modest increase in compression ratio from level 3 to 4 is mainly due to the switch from fast to normal mode and the enlarged dictionary size, which enable more exhaustive match searching and better exploitation of long-range repeated patterns at the cost of lower compression speed. These observations further suggest the high entropy in FAST floating-point data. In this context, increased algorithmic complexity primarily leads to a sharp decline in throughput while delivering only marginal gains in compression ratio, indicating an unfavorable trade-off.

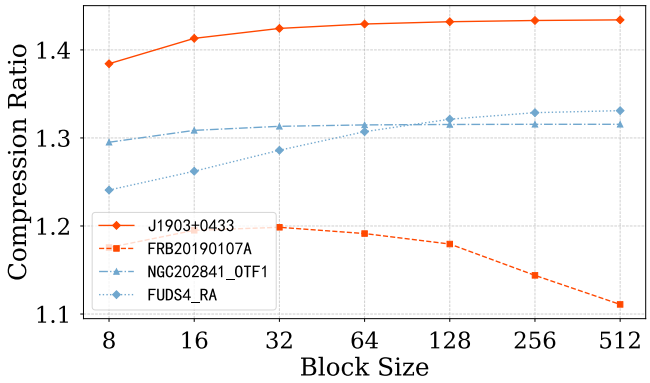


Fig. 15: Compression performance of different block sizes on representative datasets.

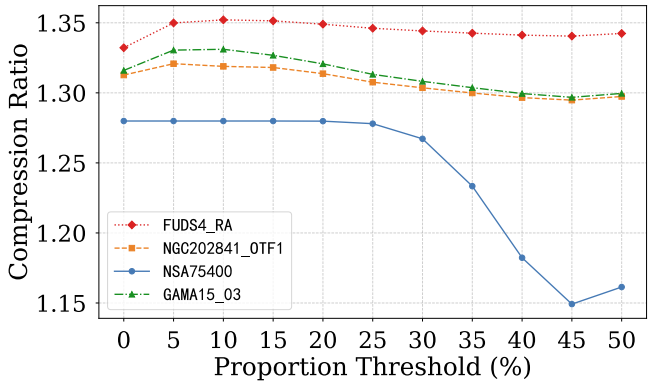


Fig. 16: Compression performance of different sign bit rearrangement thresholds on representative datasets.

Furthermore, hrzip achieves a throughput of 342 MB/s when decompressing floating-point data, significantly exceeding its performance on integer datasets. This enhancement stems from the fact that hrzip processes data on a per-value basis; since the bit-width of a floating-point value is 4× that of an integer, the throughput can theoretically reach a 4× increase. These results demonstrate that specific optimizations targeting the domain characteristics of astronomical floating-point data can effectively overcome the performance bottlenecks inherent in general-purpose compression frameworks.

Fig. 14 illustrates the CDF of compression ratios for the floating-point dataset, showing a high degree of uniformity across data tiles for each compressor. Compared to integer datasets, the distribution curves for floating-point data are more compact and concentrated. This indicates that the compression ratios of these data tiles are primarily constrained by the intrinsic entropy of the data themselves, rather than by significant performance variations resulting from the data fluctuations commonly observed in integer datasets.

### C. Sensitivity and Ablation Analysis

*1) Block Size:* Block size affects the performance of compressors. When a data block is too small, it contains too few elements to capture effective statistical features and is more susceptible to random noise, which in turn reduces the stability of subsequent processing stages. In contrast, when a data block is too large, the symbol distribution in entropy coding tends to become more uniform, thereby reducing compression ratios.

To evaluate the impact of block size, we randomly extracted 10 GB of data from each floating-point dataset, converted the data into one-dimensional vectors, and then fed them into the hrzip prototype. The results, shown in Fig. 15, are consistent with our theoretical expectations. Across datasets with different characteristics, performance under different block sizes initially improved, then stabilized, and in some cases even declined. To maintain a simple compressor design, hrzip currently partitions vectors into fixed-length blocks with $B = 64$. This setting achieves strong performance in our experiments while keeping implementation complexity manageable.

*2) Sign Bit Threshold:* hrzip's sign bit rearrangement threshold $P$ can affect the efficiency of M-Exp-Golomb encoding. If $P$ is too small, sign bits are rearranged too aggressively, which can obscure their correlations and reduce the compression ratio. In contrast, if $P$ is too large, sign bits are rearranged too conservatively, which can negatively impact data prediction, leading to a lower compression ratio.

We randomly extracted 10 GB of data from each representative dataset and varied the threshold $P$ used in hrzip for dynamic floating-point bit placement. In our experiments, $P$ ranged from 0% to 50% in increments of 5%. Here, $P = 0\%$ means that the sign bit and exponent bits are always swapped, whereas $P = 50\%$ means that the sign bit always remains in its original position. The upper bound of $P$ is 50% because $P$ represents the proportion of minority sign bits in the data, which by definition cannot exceed 50%. The results, shown in Fig. 16 demonstrate that $P = 13\%$ lies approximately within the optimal performance range, where data rearrangement effectively exploits the correlations among sign bits without excessively disrupting the statistical properties of the data, thereby achieving a higher compression ratio.

*3) Trimmed Mean vs. Median:* The data rearrangement module in hrzip uses the mean as a constant predictor. We replaced it with the median and evaluated the modified version using the same experimental setup and datasets as the performance testing. The results show that, for floating-point datasets, the compression throughput decreases by 81.6%, while the compression ratio also slightly decreases by 0.18%. This further indicates that the local variations in high-resolution astronomical data are smooth and that the difference between the median and the mean is extremely small.

*4) Parameter Reuse vs. Two-pass Encoding:* We replaced the parameter reuse policy in the metadata management module of hrzip with the classical two-pass encoding/decoding scheme, and evaluated the modified version under the same conditions as the performance testing. The two-pass scheme reduces the compression ratio by 1.21% on average, due to the additional metadata that must be stored. It also decreases compression throughput by 26.91% on average because of the extra

computational overhead. This demonstrates the effectiveness and necessity of alleviating compression bottlenecks through domain-specific trade-off designs.

## V. Related Work

In this section, we review the existing research on astronomical data compression, including both general-purpose compressors and specialized data compressors. Furthermore, we analyze their limitations in the context of high-resolution astronomical data processing.

Classical astronomical data compressors are designed for integer-based astronomical images. Among them, Rice [9] and HCompress [10] are the most widely adopted. HCompress accepts a 2D image as input, applies Haar wavelet decomposition to image blocks, and finally encodes the coefficients using a combination of quad-tree bit-plane encoding and Huffman coding. Rice, on the other hand, performs differential prediction on the data and subsequently applies Golomb coding to the resulting residuals. Given the high proportion of images in astronomical data, several studies have investigated the potential of the JPEG family for astronomical image compression. Compressor ACC [14] predicts pixel values via context-based linear regression [32], and encodes the residuals using the JPEG 2000 bit-stream. Maireles-González et al. [11] systematically evaluated the performance of the main lossless compressors in astronomical data and found that JPEG-LS achieves superior compression ratios on 16-bit integer data. In other studies, Maireles-González et al. [12] showed that substituting the 5/3 DWT of JPEG 2000 with the discrete Haar wavelet yields higher compression ratios on astronomical images. Not only that, the wavelet transform used in existing scientific lossy compressors also includes the CDF9/7 [33] wavelet in SPERR [34] and the Sym13 [35] wavelet in FAZ [36].

These compressors are limited to processing 2D integer data and focus specifically on image data, making them difficult to adapt directly to the high-resolution floating-point time-series data generated by FAST.

Existing work on floating-point data has focused predominantly on lossy compression, including the fpack/funpack tools [37] and the SZ [32], [38] algorithm family. fpack/funpack provides a compression framework for floating-point astronomical images that supports both lossless compression via Rice encoding and lossy compression through linear quantization. SZ is a floating-point lossy compressor suitable for astronomical data, with several subsequent improved versions [31], [39]–[46] that have been specifically optimized for hardware platforms such as GPUs and FPGAs. MGARD [47] overcomes the uniform grid constraints of SZ by decomposing floating-point data into a multi-grid hierarchical representation and applying quantization techniques, achieving superior compression capabilities for scientific data. Zhao et al. [48] analyzed significant flaws in the prediction methods used by SZ and adopted an innovative prediction approach incorporating cubic spline interpolation, which substantially improves the compression ratio. Polycomp [49] designs a lossy compressor that combines least-squares polynomial fitting with the Discrete Chebyshev Transform (DCT), allowing for a user-defined error ceiling to balance compression ratio and data precision.

Several lossy compression methods have also been designed specifically for the characteristics of astronomical floating-point data. SolarZip [50] targets solar EUV imaging data, employing a hybrid strategy controller to dynamically select the optimal prediction algorithm (e.g., spline interpolation, Lorenzo prediction) based on error bounds to achieve scientific fidelity at high compression ratios. Chege et al. [51] normalized visibility astronomical data across antenna-frequency or row-frequency dimensions to ensure constant noise variance, followed by non-linear quantization with dithering and bit packing. Batmunkh et al. [52] proposed neural network compression methods based on deep autoencoders and 1D convolutional autoencoders with particular attention to the imbalance between quiet-Sun and sunspot data. For the compression of astronomical light curves, Demir et al. [53] proposed fractional-order regularized wavelet compression, modeling the compression of wavelet coefficients across scales as a fractional-order relaxation evolution equation to reduce data scale without erasing weak transient signals. However, their lossy nature limits their applicability in scenarios where faint astronomical signals must be preserved.

For lossless compression of scientific data, existing general-purpose solutions primarily include fpzip, zfp(including subsequent versions zfp-r [41], cuZFP [24], [54]), and ndzip. fpzip maps floating-point numbers to monotonic integers, supports various predictors to estimate data values and compute residuals, and then employs an improved fast range coder to entropy code the leading-zero counts of these residuals. ndzip transforms the floating-point predictors of fpzip into separable integer-domain transforms and replaces entropy coding with parallel-friendly bit-matrix transposition and zero-word elimination, achieving very high throughput using modern parallel hardware architectures. zfp partitions data into independent blocks of size $4^d$, aligns intra-block exponents using a block floating-point format, and converts them into fixed-point representations. It subsequently applies a parameterized orthogonal block transform to remove spatial correlations, and finally encodes transform coefficients by significance layers using embedded bit-plane coding based on group testing to generate a fixed bitstream of user-specified length. Although zfp is primarily designed for lossy compression, near-lossless compression can be achieved through high-bitrate configurations. Despite their general applicability, these tools do not fully exploit the physical characteristics of astronomical data and thus often fail to deliver optimal compression performance on specific astronomical observational datasets.

Bitshuffle [55] is a preprocessing algorithm that performs bit-level transposition. It rearranges the bits of adjacent array elements by bit plane, collecting bits at the same bit position from different values into contiguous regions. This enhances local correlation in the data and improves compressibility for general-purpose compressors. However, the transposition process introduces additional computational overhead. In our

evaluation, this overhead reduced the overall throughput of the zstd-based compression pipeline by at least 50%, making it difficult to meet the performance requirements of the application scenario considered in this paper.

In general, prior work has predominantly focused on lossy compression for floating-point data, whereas lossless approaches have mainly advanced individual components such as prediction, decorrelation, and residual coding. Efficient lossless compression of high-resolution astronomical data, particularly floating-point observation data, remains an open challenge.

## VI. Conclusion & Future Work

In this paper, we identify the distinct characteristics of another category of astronomical data. Targeting these traits, we propose hrzip, a novel compressor specifically designed for high-resolution astronomical data. Experimental results demonstrate that by fully leveraging these data characteristics, hrzip achieves a high compression ratio while maintaining excellent performance.

Future work could further extend our system to support dynamic cluster awareness, complemented by cost comparison, to account for the trade-offs between CPU and storage resources under different deployment conditions.

A worthwhile direction for future work is to evaluate the general applicability of hrzip to different types of scientific data, particularly its suitability for simulated data rather than observational ones alone. If hrzip can achieve strong compression performance in such contexts, its potential range of applications could be significantly broadened.

## Acknowledgment

We thank the anonymous reviewers for their valuable feedback. This work is supported by the National Key Research and Development Program of China (Grant No. 2023YFB4502703) and the National Natural Science Foundation of China under Grant No. U22A2027.

This publication makes use of data products from the Five-hundred-meter Aperture Spherical radio Telescope (FAST). FAST is a Chinese national mega-science facility, operated by the National Astronomical Observatories of Chinese Academy of Sciences (NAOC).

This publication makes use of data products from the Wide-field Infrared Survey Explorer, which is a joint project of the University of California, Los Angeles, and the Jet Propulsion Laboratory/California Institute of Technology, funded by the National Aeronautics and Space Administration. This paper makes use of the following ALMA data: ADS/JAO.ALMA#2023.1.00812.S. ALMA is a partnership of ESO (representing its member states), NSF (USA) and NINS (Japan), together with NRC (Canada), NSTC and ASIAA (Taiwan), and KASI (Republic of Korea), in cooperation with the Republic of Chile. The Joint ALMA Observatory is operated by ESO, AUI/NRAO and NAOJ.

This publication makes use of data products from the Wide-field Infrared Survey Explorer, which is a joint project of the University of California, Los Angeles, and the Jet Propulsion Laboratory/California Institute of Technology, funded by the National Aeronautics and Space Administration.

## References

[1] Q. Le Boulc'h, F. Hernandez, and G. Mainetti, "The Rubin Observatory's legacy survey of space and time DP0.2 processing campaign at CC-IN2P3," *EPJ Web of Conference*, vol. 295, p. 04049, 2024.

[2] "James Webb Telescope communications," https://spectrum.ieee.org/james-webb-telescope-communications, 2022.

[3] W. Huang, J. Yang, S. Yin, H. Li, Y. Gu, Z. Liu, X. Jing, Z. Wei, S. Fu, H. Hu, G. Tan, and D. Tao, "MANS: Efficient and portable ANS encoding for multi-byte integer data on CPUs and GPUs," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. Association for Computing Machinery, 2025, pp. 1299–1314.

[4] "Reproducible science in the context of the SKA by the use of virtualization technologies," https://www.hq.eso.org/sci/meetings/2022/REPRODUCIBILITY2022/programme/JorgeBrunoMorgado.html.

[5] P. Škoda, "Massively Parallel Machine Learning in the Virtual Observatory as a Key Technology in the Era of Multi-Million Spectral Surveys," in *Astronomical Society of India Conference Series*, vol. 14, 2017, pp. 73–82.

[6] K. Zhao, S. Di, D. Perez, X. Liang, Z. Chen, and F. Cappello, "Mdz: An efficient error-bounded lossy compressor for molecular dynamics," in *IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 27–40.

[7] A. M. Gok, S. Di, Y. Alexeev, D. Tao, V. Mironov, X. Liang, and F. Cappello, "Pastri: Error-bounded lossy compression for two-electron integrals in quantum chemistry," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, 2018, pp. 1–11.

[8] "Five-Hundred-Meter Aperture Spherical Radio Telescope," https://fast.bao.ac.cn/, 2026.

[9] R. F. Rice, P.-S. Yeh, and W. Miller, "Algorithms for a very high-speed universal noiseless coding module," Jet Propulsion Laboratory, National Aeronautics and Space Administration, JPL Publication, 1991.

[10] R. L. White, M. Postman, and M. G. Lattanzi, "Compression of the guide star digitised schmidt plates," in *Digitised Optical Sky Surveys*, H. T. MacGillivray and E. B. Thomson, Eds. Dordrecht: Springer Netherlands, 1992, pp. 167–175.

[11] Ò. Maireles-González, J. Bartrina-Rapesta, M. Hernández-Cabronero, and J. Serra-Sagristà, "Analysis of lossless compressors applied to integer and floating-point astronomical data," in *2022 Data Compression Conference (DCC)*, 2022, pp. 389–398.

[12] Ò. Maireles-González, J. Bartrina-Rapesta, M. Hernández-Cabronero, and J. Serra-Sagristà, "Efficient lossless compression of integer astronomical data," *Publications of the Astronomical Society of the Pacific*, vol. 135, no. 1051, p. 094502, 2023.

[13] T. Truong, R. Sudharsan, Y. Yang, P. X. Ma, R. Yang, S. Mandt, and J. S. Bloom, "Astrocompress: A benchmark dataset for multi-purpose compression of astronomical data," *arXiv preprint arXiv:2506.08306*, 2025.

[14] P. Páta and J. Schindler, "Astronomical context coder for image compression," *Experimental Astronomy*, vol. 39, pp. 495–512, 2015.

[15] B.-Z. Wu and A. C.-W. Tang, "Lossless compression using joint predictor for astronomical images," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnação, C. T. Silva, and D. Coming, Eds. Springer Berlin Heidelberg, 2009, pp. 274–282.

[16] W. Shao, D. Fan, C. Cui, Y. Xu, S. Wei, and X. Lyu, "Deep learning-based astronomical multimodal data fusion: A comprehensive review," *Information Fusion*, vol. 130, p. 104103, 2026.

[17] Y. Shi, X. Zou, X. Chen, S. Jin, D. Tao, C. Deng, Y. Chen, and W. Xia, "Machete: An efficient lossy floating-point compressor designed for time series databases," in *2024 Data Compression Conference (DCC)*, 2024, pp. 532–541.

[18] "The gzip homepage," https://www.gzip.org, 2023.

[19] "LZ4 compression algorithm," https://github.com/lz4/lz4, 2011.

[20] I. Pavlov, "7-zip," https://www.7-zip.org, 1999.

[21] "Zstandard: Fast real-time compression algorithm," https://github.com/facebook/zstd, 2016.

[22] F. Knorr, P. Thoman, and T. Fahringer, "ndzip: A high-throughput parallel lossless compressor for scientific data," in *2021 Data Compression Conference (DCC)*, 2021, pp. 103–112.

[23] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1245–1250, 2006.

[24] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[25] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. Van Andel, "Nyx: A massively parallel amr code for computational cosmology," *The Astrophysical Journal*, vol. 765, no. 1, p. 39, 2013.

[26] J. E. Kay, C. Deser, A. Phillips, A. Mai, C. Hannay, G. Strand, J. M. Arblaster, S. C. Bates, G. Danabasoglu, J. Edwards, M. Holland, P. Kushner, J.-F. Lamarque, D. Lawrence, K. Lindsay, A. Middleton, E. Munoz, R. Neale, K. Oleson, L. Polvani, and M. Vertenstein, "The community earth system model (CESM) large ensemble project: A community resource for studying climate change in the presence of internal climate variability," *Bulletin of the American Meteorological Society*, vol. 96, no. 8, pp. 1333–1349, 2015.

[27] Z. Li, "Recent advances in earthquake monitoring II: Emergence of next-generation intelligent systems," *Earthquake Science*, vol. 34, no. 6, pp. 531–540, 2021.

[28] Y. Huang, S. Di, R. Underwood, P. Myint, M. Chu, G. Li, N. Schwarz, and F. Cappello, "lsCOMP: Efficient light source compression," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. Association for Computing Machinery, 2025, pp. 2006–2023.

[29] E. L. Wright, P. R. M. Eisenhardt, A. K. Mainzer, M. E. Ressler, R. M. Cutri, T. Jarrett, J. D. Kirkpatrick, D. Padgett, R. S. McMillan, M. Skrutskie, S. A. Stanford, M. Cohen, R. G. Walker, J. C. Mather, D. Leisawitz, T. N. Gautier, I. McLean, D. Benford, C. J. Lonsdale, A. Blain, B. Mendez, W. R. Irace, V. Duval, F. Liu, D. Royer, I. Heinrichsen, J. Howard, M. Shannon, M. Kendall, A. L. Walsh, M. Larsen, J. G. Cardon, S. Schick, M. Schwalm, M. Abid, B. Fabinsky, L. Naes, and C.-W. Tsai, "The wide-field infrared survey explorer (wise): Mission description and initial on-orbit performance," *The Astronomical Journal*, vol. 140, no. 6, p. 1868, 2010.

[30] "ALMA Observatory," https://www.almaobservatory.org/en/home/, 2021.

[31] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak, "Out-of-core compression and decompression of large n-dimensional scalar fields." *Computer Graphics Forum*, vol. 22, pp. 343–348, 2003.

[32] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2017, pp. 1129–1139.

[33] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 45, no. 5, pp. 485–560, 1992.

[34] S. Li, P. Lindstrom, and J. Clyne, "Lossy scientific data compression with SPERR," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, 2023, pp. 1007–1017.

[35] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 41, no. 7, pp. 909–996, 1988.

[36] J. Liu, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "FAZ: A flexible auto-tuned modular error-bounded compression framework for scientific data," in *Proceedings of the 37th ACM International Conference on Supercomputing (ICS)*. Association for Computing Machinery, 2023, pp. 1–13.

[37] W. D. Pence, R. L. White, and R. Seaman, "Optimal compression of floating-point astronomical images without significant loss of information," *Publications of the Astronomical Society of the Pacific*, vol. 122, no. 895, p. 1065, aug 2010.

[38] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016, pp. 730–739.

[39] J. Tian, S. Di, K. Zhao, C. Rivera, M. H. Fulp, R. Underwood, S. Jin, X. Liang, J. Calhoun, D. Tao, and F. Cappello, "cuSZ: An efficient gpu-based error-bounded lossy compression framework for scientific data," in *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*. Association for Computing Machinery, 2020, pp. 3–15.

[40] Y. Huang, S. Di, X. Yu, G. Li, and F. Cappello, "cuSZp: An ultra-fast gpu error-bounded lossy compression framework with optimized end-to-end performance," in *SC23: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2023, pp. 1–14.

[41] V. A. P. Magri and P. Lindstrom, "A general framework for progressive data compression and retrieval," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, pp. 1358–1368, 2024.

[42] X. Wu, Q. Gong, J. Chen, Q. Liu, N. Podhorszki, X. Liang, and S. Klasky, "Error-controlled progressive retrieval of scientific data under derivable quantities of interest," in *SC24: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2024.

[43] J. Liu, J. Tian, S. Wu, S. Di, B. Zhang, R. Underwood, Y. Huang, J. Huang, K. Zhao, G. Li, D. Tao, Z. Chen, and F. Cappello, "CUSZ-i: High-ratio scientific lossy compression on GPUs with optimized multi-level interpolation," in *SC24: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2024, pp. 1–15.

[44] S. Wu, J. Pan, J. Liu, J. Tian, Z. Qiu, J. Huang, K. Zhao, X. Liang, S. Di, Z. Chen, and F. Cappello, "Boosting scientific error-bounded lossy compression through optimized synergistic lossy-lossless orchestration," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. Association for Computing Machinery, 2025, pp. 2024–2037.

[45] S. Song, Y. Huang, P. Jiang, X. Yu, W. Zheng, S. Di, Q. Cao, Y. Feng, Z. Xie, and F. Cappello, "CereSZ: Enabling and scaling error-bounded lossy compression on Cerebras CS-2," in *Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing*. Association for Computing Machinery, 2024, pp. 309–321.

[46] J. Tian, S. Di, C. Zhang, X. Liang, S. Jin, D. Cheng, D. Tao, and F. Cappello, "waveSZ: a hardware-algorithm co-design of efficient lossy compression for scientific data," in *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. Association for Computing Machinery, 2020, pp. 74–88.

[47] Q. Gong, J. Chen, B. Whitney, X. Liang, V. Reshniak, T. Banerjee, J. Lee, A. Rangarajan, L. Wan, N. Vidal, Q. Liu, A. Gainaru, N. Podhorszki, R. Archibald, S. Ranka, and S. Klasky, "Mgard: A multigrid framework for high-performance, error-controlled data compression and refactoring," *SoftwareX*, vol. 24, p. 101590, 2023.

[48] K. Zhao, S. Di, M. Dmitriev, T.-L. D. Tonellot, Z. Chen, and F. Cappello, "Optimizing error-bounded lossy compression for scientific data by dynamic spline interpolation," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021, pp. 1643–1654.

[49] M. Tomasi, "Polycomp: Efficient and configurable compression of astronomical timelines," *Astronomy and Computing*, vol. 16, pp. 88–98, 2016.

[50] Z. Liu, S. Tan, A. Warmuth, F. Schuller, Y. Hong, W. Huang, Y. Gu, B. Zhu, G. Tan, and D. Tao, "SolarZip: An efficient and adaptive compression framework for Solar EUV imaging data: Application to Solar Orbiter/EUI data," *Astronomy & Astrophysics*, vol. 702, p. A160, 2025.

[51] J. K. Chege, L. V. E. Koopmans, A. R. Offringa, B. K. Gehlot, S. A. Brackenhoff, E. Ceccotti, S. Ghosh, C. Höfer, F. G. Mertens, M. Mevius, and S. Munshi, "The impact of lossy data compression on the power spectrum of the high-redshift 21 cm signal with lofar," *Astronomy & Astrophysics*, vol. 692, p. A211, 2024.

[52] J. Batmunkh, Y. Iida, T. Oba, and H. Iijima, "Compression method for solar polarization spectra collected from Hinode SOT/SP observations," *Astronomy and Computing*, vol. 51, p. 100929, 2025.

[53] T. Demir and A. Koçyiğit, "Atangana-baleanu regularized wavelet compression for astronomical time-series," *arXiv preprint arXiv:2512.13916*, 2025.

[54] "cuzfp github repository," https://github.com/LLNL/zfp/tree/develop/src/cuda_zfp, 2022.

[55] "Bitshuffle," https://github.com/kiyo-masui/bitshuffle, 2015.