Real-time and Post-hoc Compression for Data from Distributed Acoustic Sensing

Bin Dong[a], Alex Popescu[b], Verónica Rodríguez Tribaldos[a], Suren Byna[a], Jonathan Ajo-Franklin[a,c], Kesheng Wu[a] and the Imperial Valley Dark Fiber Team

[a]*Lawrence Berkeley National Laboratory, Berkeley, CA, USA.*
[b]*University of California, Berkeley, CA, USA.*
[c]*Rice University, Houston, TX, USA*

ARTICLE INFO

Keywords:
Distributed Acoustic Sensing
DAS
Data Compression
Lossless
Real-time
Post-hoc
Parallel
HDF5
ZSTD
TurboPFOR
ZigZag

ABSTRACT

Distributed Acoustic Sensing (DAS) is an emerging sensing technology that records the strain-rate along fiber optic cables at high spatial and temporal resolution. This technique is becoming a popular tool in seismology, hydrology, and other subsurface monitoring applications. However, due to the large coverage (10's of km) and high density of measurements (1 m spacing at 100's of Hz), a DAS installation could produce terabytes of data records per day. Because many of DAS devices are deployed in remote locations, this large data size poses significant challenges to its transfer and storage. In this paper, we explore lossless compression methods to reduce the storage requirement in both real-time and post-hoc scenarios. We propose a two-stage compression method to improve the compression ratio and the compression speed. This two-stage compression method could reduce the storage requirement by 40%, which is 20% more than other lossless methods, such as ZSTD. We demonstrate that the compression method could complete its operation well before the DAS device needs to output the next file, making it suitable for real-time DAS acquisition. We also implement a parallel compression method for a post-hoc scenario and demonstrate that our method could effectively utilize a parallel computer. With 256 CPU cores, our parallel compression method achieves the speed of 26GB/seconds.

CRediT authorship contribution statement

Bin Dong: Conceptualization, Data curation, Methodology, Software, Writing-original draft. **Alex Popescu:** Investigation, Writing-review & editing. **Verónica Rodríguez Tribaldos:** Data curation, Validation, Writing-review & editing. **Suren Byna:** Methodology, Writing-review & editing. **Jonathan Ajo-Franklin:** Conceptualization, Data curation, Validation, Funding acquisition,Writing-review & editing, Project administration. **Kesheng Wu:** Conceptualization, Validation, Funding acquisition, Writing-review & editing, Project administration. **the Imperial Valley Dark Fiber Team:** Validation, Project administration.

1. Introduction

Distributed Acoustic Sensing (DAS) systems record strain-rate caused by ground motion along optical-fiber cables that are deployed in surface trenches, boreholes, or the seafloor (Hartog, 2017; Ajo-Franklin et al., 2017; Lindsey et al., 2020). These DAS data records can be used to detect earthquakes, image subsurface structures, and enable a range of geophysical applications (Xing et al., 2018; Shiloh et al., 2018). DAS systems can record strain-rate on optical-fibers up to 10s of kilometers in length and sample strain-rate at high spatial (e.g., 1 m) and temporal (up to 10s of kHz) resolution, generating dataset of significant size (> 1TB per day) (Paitz et al., 2021; Ajo-Franklin et al., 2017). Because DAS deployments are often at remote locations with limited network accesses, this large size poses significant challenges for both local and archival storage as well as telemetry and data distribution.

ORCID(s): 0000-0002-0725-0833 (B. Dong)

As an illustration of these challenges, here we describe a DAS deployment, currently operating in the Imperial Valley in Southern California (Verónica and Jonathan, 2021). The DAS Interrogator Unit (IU) used in the experiment is deployed at an isolated telecommunication in-line amplification (ILA) hut near Calipatria, CA. The geoscience project team, however, is entirely remote, distributed among Berkeley (CA), Houston (TX), Livermore (CA), and San Diego (CA). The IU produces a datastream of approximately 597 GB/day or 4.2 TB/week, which is locally recorded on a RAID array connected to the IU. The RAID array has a capacity for about 5 months of recording and requires three annual trips for data exchange followed by a laborious multi-institution transfer. These visits are expensive and this setting and delays the analysis of the data. Additionally, acquisition parameters had to be chosen to reduce the number of field trips thus the team could not exploit the IUs optimal performance.

This particular deployment and many other DAS efforts share a common challenge in data acquisition, that is their data acquisition rates significantly exceed their network connection speed, which requires the data to be stored on local disks that have to be periodically replaced. In this work, we explore compression methods to reduce the storage requirements for DAS data and therefore reduce field visits and potentially offer opportunities to optimize both local storage and telemetry (Foukas et al., 2017). In particular, this paper considers two scenarios for DAS data compression:

- a real-time "edge compression" scenario, where the DAS data is continuously produced from DAS interrogator units and saved in an attached computer disk in a field deployment (Rodríguez Tribaldos et al., 2020). In this scenario, the compression time is bounded by the cycle of DAS sampling.
- a post-hoc scenario, where terabytes or even exabytes of DAS data have been acquired in previous experiments and can be compressed to reduce storage space (Zhan, 2019; Verdon et al., 2020; Feigl, 2016; Ajo-Franklin et al., 2017). Although there is no time limit in this case, finishing compression quickly is still beneficial, for example to reduce the time period where both compressed and uncompressed data files are needed.

DAS is still a relatively new technology for geophysical applications, and the community has not yet achieved a consensus on what data elements are valuable and which ones can be discarded. Thus, we will consider lossless compression methods in this work. Our study targets the raw optical phase data recorded as short integers. The datasets used here were recorded using a Silixa iDAS V.2 interrogator unit, a system in wide academic and commercial use. Scaling raw optical phase recorded in integer formats to floating point representation is a common operation in data analysis (Feigl, 1969, 2016). This scaled version can be compressed too (Zhao et al., 2020; Diffenderfer et al., 2019), but we will focus on the integer version of the raw data as we explore compression approaches without the need for data pre-processing. To the best of the authors knowledge, this is the first work to explore a lossless compression method for raw DAS data. We have identified the following challenges towards this goal:

- DAS data has several unusual features. First, the raw data recorded by DAS interrogator units is stored as short integers, which is already relatively compact. Second, the DAS data exhibits normal distribution with both negative and positive values. However, many lossless integer compression methods only work on positive values (Powturbo, 2021). This is because the commonly used binary representation of a negative integer sets the leading bits to 1, and most encoding schemes require more space to represent 1s than 0s (Lemire and Boytsov, 2015; Trotman, 2014).

- Geoscientists have established software tools, such as DASSA (Dong et al., 2020), MATLAB (Higham and Higham, 2016), and HDF5 (The HDF Group, 2010), to analyze and store the DAS data. However, most compression methods have their own software stack (Lemire and Boytsov, 2015; Trotman, 2014). Thus, it is critical to seamlessly integrate compression methods into these existing DAS software without significant modification.
- Existing DAS data accumulated from past experiments are large. Most existing compression methods only support thread based parallelization, which uses only a single computing node (Trotman, 2014). To compress large-scale DAS data quickly in a post-hoc scenario, a parallel compression method across nodes is needed.

To address these challenges, we have studied the characteristics of DAS data and used them to develop a lossless DAS data compression method. Our goal is to find a generic data compression method that could compress DAS data with high compression ratio and fast compression speed. We then develop algorithms to customize this generic data compression method to work in a real-time scenario and a post-hoc scenario. In summary, our contributions include:

- We propose a two-stage DAS data compression method, consisting of a pre-processing stage and an encoding stage. The pre-processing stage transforms the raw DAS data to better fit the encoding method, and the encoding stage finds a compact representation for the pre-processed data.
- Through extensive experimentation, we find a combination of ZigZag as pre-processor (Google, 2021) and TurboPFOR (Powturbo, 2021) as the encoder producing the highest compression ratios.
- For the real-time scenario, we developed a new filter using the HDF5 library (Dong et al., 2020) so that DAS data could be seamlessly outputted as compressed HDF5 files.
- For parallel use cases, we develop a parallel implementation that supports different parallelization strategies.

We demonstrate the effectiveness of our methods on three real DAS data sets from the Imperial Valley Dark Fiber Experiment (Ajo-Franklin et al., in press), the Fiber Optic Sacramento Seismic Experiment (FOSSA) (Ajo-Franklin et al., 2019) and the Monterey Bay Experiment (Lindsey et al., 2019a). Our method reduce the DAS data size by up to 40%. Comparing with state-of-the-art compression method ZSTD (Collet and Kucherawy, 2021), our method saves as much as 21% extra space. The real-time DAS data compression takes around two seconds on a single CPU core to process a data file representing 1 minute of DAS recording, which means the compression method finishes before the IU produces the next data file, thus allowing real-time compression. Tests with our parallel implementation show that it scales nearly linearly up to 256 CPU cores and achieves $26GB/second$ with 256 cores.

The paper is organized as follows. Section 2 introduces related work. Section 3 describes our DAS data compression method. Section 4 reports performance evaluation. Section 5 concludes the paper with future work.

2. Related Work

Data compression methods are extensively documented in the literature (Jayasankar et al., 2021; Smith, 2010; Holtz, 1993; Mittal and Vetter, 2016). DAS data has its unique characteristics, such as short integer type. Thus, here we use data type to classify existing compression methods into three categories: universal compression method, integer compression method, and DAS-specific compression method.

Universal compression method: LZ77 (Ziv and Lempel, 1977) is a universal and lossless method for data compression. The general idea for LZ77 is to find repeated items in the data and let all these items refer to the first one with a length-offset pair encoding. A sliding window is maintained to track how far in the data the repeated items are looked for. LZ77 has inspired many data compression methods, such as LZ78, LZW, LZSS, LZMA, GIF in PNG, and DEFLATE in ZIP (Kinsner and Greenfield, 1991; Pylak, 1990; Liaw, 1995; Deutsch, 1996). Huffman code is another lossless compression method (Huffman, 1952) which uses optimal prefix code based on a frequency-sorted binary tree. Similarly, the information theory based compression method, entropy coding, can encode data items by using the number of bits that is inversely proportional to its probability (MacKay, 2002). Zstandard (or ZSTD) (Collet and Kucherawy, 2021) is one of the latest lossless compression methods with advantages from LZ77, entropy encoding, and Huffman coding. Thus, we take ZSTD as a representative method of universal compression methods in our performance evaluation to compress DAS data.

Integer compression method: There are a number of techniques optimized for compressing integer data with higher compression ratio or speed. For example, differential coding could convert a series of integers to have more repeated data items to compress. Also, considering the type of data provides the opportunity to view data with large granularity (e.g., 4-byte integers or 2-byte short integers) rather than one byte. The latest compression methods for integer compression methods include TurboPFor (Powturbo, 2021) and FastPFor (Lemire and Boytsov, 2015). FastPFor does not support compressing 2-byte short integers. TurboPFor supports 2-byte short integers, but it only accepts unsigned short integers. As mentioned before, most raw DAS data are stored as short integers with both negative and positive values. In this work, we extend the TurboPFor to work on DAS data and also to improve its compression ratio.

DAS-specific compression method: Convolutional neural networks (CNN)(Liehr et al., 2020), principal component analysis (PCA)(Ibrahim et al., 2020), Curvelet (Qin et al., 2017), and other methods (Meng et al., 2019; Soto et al., 2016) have been applied to reduce noise in DAS datasets (Liehr et al., 2020). These de-noising methods can be regarded as lossy compression methods designed to reduce the cost or improve quality of specific analysis procedures. In this paper, we explore a lossless compression method that would benefit all types of analyses.

Table 1
A summary of the DAS datasets used in this paper. The "Measure Length" is the length of the optical-fiber cable and the "Spatial Resolution" defines the length of the optical fiber which can be treated as one channel (sensor). Hence, $\frac{Measure\ Length}{Spatial\ Resolution}$ defines how many channels (sensors) per experiment. The "one minute" within the "Output Interval" column specifies that the DAS system outputs a data file per minute. Each output from the one minute interval is a separate file. Our compression method treats each file as the smallest granularity in compression. Each file consists of a 2D array whose size is $[Frequency \times 60\ ,\ \frac{Measure\ Length}{Spatial\ Resolution}\ ]$(i.e., [rows, columns]). The total number of files and the total size of the data are listed in the column "File Count/Total Size". These files are collected between the "Start Time" and the "End Time" (UTC). The format for the time is "month/day/year hour:minute:second".

	Measure Length(m)	Spatial Resolution (m)	Frequency (Hz)	Output Interval	File Count / Total Size	Start Time	End Time
Sacramento	23296	2	500	One Minute	7200/4.6T	01/02/18 16:17:29	01/07/18 16:16:29
ImpValley	27648	4	500	One Minute	7200/2.7T	11/10/20 23:18:85	11/15/20 23:46:31
MBARI	39168	4	500	One Minute	2859/0.8TB	06/22/19 00:10:34	06/22/19 23:59:34

3. Two-stage DAS Data Compression Method

In this section, we introduce the characteristics of the raw DAS data, the two-stage compression method, and its implementation for both real-time and post-hoc scenarios.

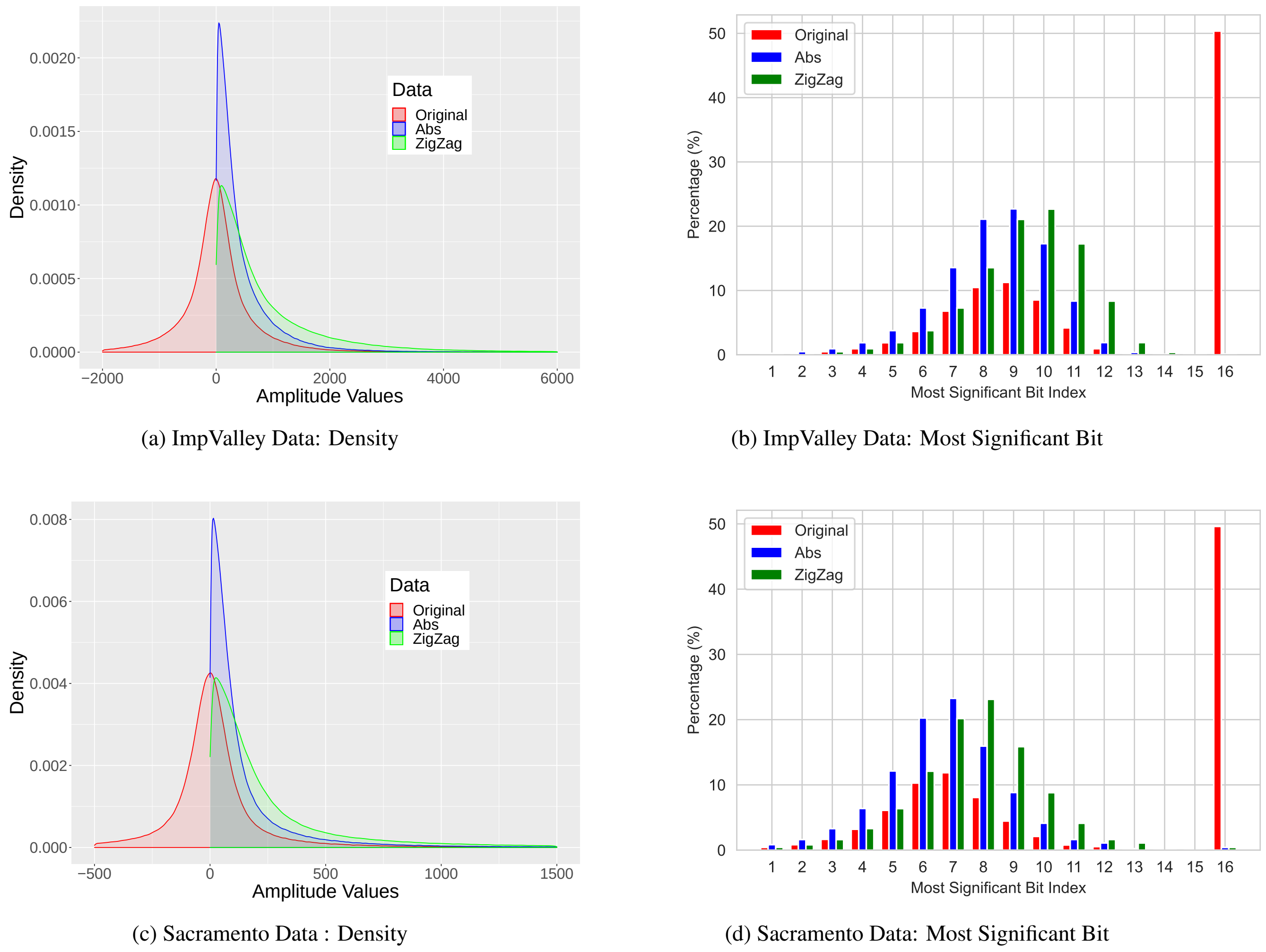
3.1. Characteristics of DAS Data

Three DAS datasets are used in our work and their basic information is summarized in Table 1. The first dataset, referred to as "Sacramento", is from an experiment conducted in the Sacramento Valley, CA (Ajo-Franklin et al., 2019). The second one referred as "ImpValley", comes from an ongoing experiment in the Imperial Valley, CA (Ajo-Franklin et al., in press) . The third one (referred as "MBARI") comes from the Monterey Accelerated Research System (MARS) in Monterey Bay, California (Lindsey et al., 2019a). The statistical properties of these datasets are shown in red in Fig. 1 ("Original" legend). From the large number of data files, only two randomly selected files from "Sacramento" dataset and "ImpValley" dataset are shown here. The "MBARI" dataset and other files "Sacramento" and "ImpValley" in have similar statistical properties. From these figures, we can draw the following conclusions:

- **Density** describes the histogram of values in the data. For the original data, both the Sacramento data and Imperial Valley data show a normal distribution with both negative and positive values.
- **Most significant bit (MSB)** of a value refers to the left-most bit that is 1 in its binary representation[1]. An example of the MSB is presented in Fig. 2. About 50% of the original data have the MSB at index 16. Except these negative values, the MSB of positive values shows normal distribution with center at 9 and 7 for Imperial Valley and Sacramento, respectively.

.

[1]Two's complement: `https://en.wikipedia.org/wiki/Two%27s_complement`



(a) ImpValley Data: Density (b) ImpValley Data: Most Significant Bit

(c) Sacramento Data : Density (d) Sacramento Data: Most Significant Bit

Figure 1: Statistical properties of the DAS data from the Sacramento and Imperial Valley experiments. The data file from the Sacramento dataset is 01/04/2018 23:35:31 and from the Imperial Valley dataset is 11/12/2020 18:41:32. To easily compare the density plots shown in the figure, we drop the long tail parts of the distribution but only focus on the majority parts of the data (See Appendix 3 for the full plot of the density). We observe the same distribution in the MBARI data.

3.2. Two-stage DAS Data Compression Method

The two-stage DAS compression method includes a pre-processing stage and an encoding stage, as shown in Fig. 3. The pre-processing stage converts DAS values into proper shapes to fit the subsequent encoding better. The encoding stage uses short data representation to store the converted DAS data and therefore reduces its size. We will present how to find proper methods for the pre-processing and the encoding of DAS data.

3.2.1. DAS Data Pre-processing Methods

Since the DAS data examined has around 50% negative values and most integer specific compression methods only work on positive values, our specific goal for the pre-processing is to convert negative values into positive ones without

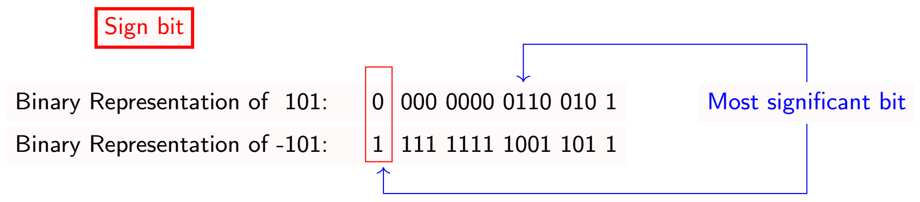


Figure 2: Binary representation and most significant bit of a 101 and -101 with the signed magnitude method. Note that two's complement format is used here to present the -101.

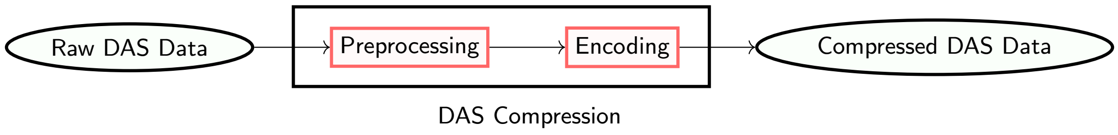


Figure 3: Overview of the two-stage compression method for the DAS data

loss of information. In our pre-processing stage, we consider two methods:

- **Abs** method converts negative values into positive ones by obtaining their absolute values. During this conversion, the sign of negative values is lost. Thus, a separate bitmap is created to record the sign of each value. After applying the **Abs** method on DAS data, the normal distribution of DAS datasets is converted into a skewed distribution, which gives the compression method more chance to find repeated values. This is shown by the plot marked in the blue color in Fig 1a and Fig.1c. For the most significant (MSB) bit after applying the **Abs**, the negative values with 16 MSB bit index (as the highest red bar in Fig. 1b and Fig. 1d) are transformed into positive ones with small MSB bit index. Therefore, this transform can help the encoding stage to find shorter representation to compress DAS data.
- **ZigZag** method converts negative values to positives and also appends the sign bit to the beginning of binary representation (Google, 2021). For the short typed DAS data, the ZigZag can be expressed as $(v \ll 1) \wedge (v \gg 15)$, where v is the value and the $\ll$, $\gg$ and $\wedge$ are all bit operators. The $v \ll 1$ shifts the value to the left by 1 bit to get the magnitude of the value[2]. The $v \gg 15$ shifts the value to the right by 15 bits to get the sign bit. The $\wedge$ is the bitwise exclusive "OR" operator to merge the magnitude and the sign bit. To convert the value back, the operator is $(v \gg 1) \wedge (-(v\&1))$, where & is the bitwise "AND" operator. After applying the ZigZag method on the DAS data, the results have the similar distribution as the one from the **Abs** method above, as shown in Fig. 1b and Fig. 1d. Compared to the centers from the **Abs** method (9 for ImpValley and 7 for Sacramento), the centers of the ZigZag method are larger (10 for ImpValley and 9 for Sacramento). This is because ZigZag stores the sign bit at the least significant bit but the **Abs** method stores the sign bit with an extra bitmap.

[2]Also, note that $v \ll 1$ is removing the sign bit and in most language standards, For negative value, the behavior of $\ll$ may be undefined or implementation-dependent, even though most compilers are doing arithmetic shifts. Details can be found `https://en.cppreference.com/w/cpp/language/operator_arithmetic`

3.2.2. DAS Data Encoding Methods

In our two-stage DAS data compression method, most existing algorithms can be used to encode the pre-processed DAS data. Here, we only consider TurboPFor (Powturbo, 2021) as an example to introduce how the encoding works. TurboPFor gives the highest encoding efficiency and the fastest compression speed for the DAS data in our experiments, as shown later in Section 4. The general idea behind TurboPFor is the bit-packing technology, which selects as few bits as possible to store a group of data. In TurboPFor, data series are split into small groups, i.e., 256 short integers per group. Then, the minimum number of bits for each group is determined to store all its integers inside. TurboPFor records the minimum number of bits and the type of a group at the beginning of each block. To work on data with various inputs, TurboPFor may support two types of groups: the constant group and the bit-packing group. The constant group stores the data with the same values. The bit-packing group may use bitmap and variable-type to store non-constant values. Theoretically, TurboPFor only scans data once and it can finish the encoding quickly. Further details of the bit-packing can be found in the paper (Powturbo, 2021). Fig. 4 gives an example of how bit-packing encoding works for a few example data points. When applied to real DAS data, as shown in the previous Fig. 1, the bit-packing method can drop the zero, which are converted from 1 from during ZigZag, after the MSB. With the help the pre-processing stage, the sign bit of negative values is removed from the left-most bit and therefore can improve the chances to find zero after the MSB.

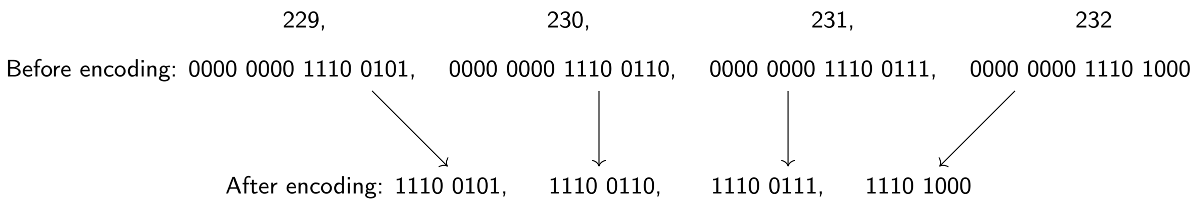

229,
230,
231,
232
Before encoding: 0000 0000 1110 0101, 0000 0000 1110 0110, 0000 0000 1110 0111, 0000 0000 1110 1000
After encoding: 1110 0101, 1110 0110, 1110 0111, 1110 1000

Figure 4: An example of encoding of 229, 230, 231, 232 with bit-packing. Before the encoding, each value is stored as a 16-bit short integer type. After the encoding, each value is stored as 8 bits, saving 50% of storage space. In TurboPFor, the type of group and number of integers are stored too. The commas are just added for visualization purposes.

3.3. Real-time DAS Data Compression

Based on the two-stage DAS data compression method above, we develop an in-situ implementation to support real-time DAS data compression. Most DAS systems are already using or transitioning to use the HDF5 format for storage (Feigl, 1969, 2016; The HDF Group, 2010). Our implementation uses the HDF5 filter plugin mechanism, which allows for DAS compression to happen on the data write path from memory to disk, as shown in Fig. 5. Most compression methods have their own file formats for the compressed data (Powturbo, 2021), but our implementation can avoid significant changes to the existing software stack used to store DAS data in the field and read back for analysis. Also, it avoids generating an extra copy of the data during compression.

HDF5 filter plugin mechanism requires the chunk size as a whole unit to apply compression. However, our experiments find chunk size has negligible impact on compression of DAS data (see Appendix 1). Therefore, we set the chunk size to be the whole array size. After compression, the whole HDF5 file can be transferred as a single unit without decompression. If users want to read back for analysis, they can reuse the same DAS data analysis software which already uses H5Dread to read the data. The HDF5 library can automatically find the decompression library and decompress the data. The implementation of the HDF5 filter plugin can be found in the code availability section.

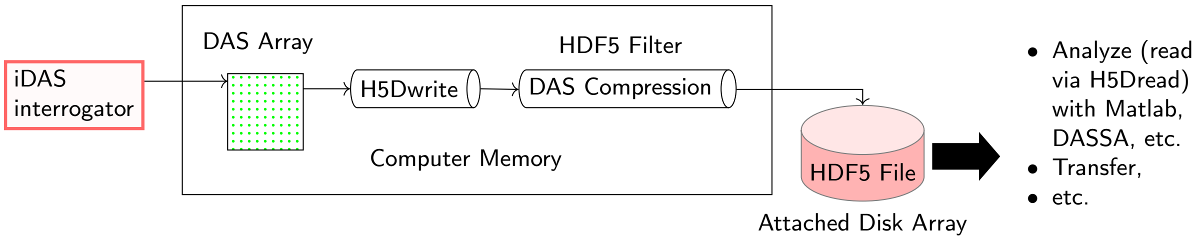


Figure 5: Overview of the real-time DAS Compression with the HDF5 filter. It receives each data, an Array, from iDAS interrogator and stores the data in computer memory. Then, it calls H5Dwrite function from HDF5 library to writes data to disk. Before the actually data is written, the DAS compression method is applied to reduce the size of data. Once data is written to the disk, it is stored as HDF5 file too. The compressed HDF5 file can be read back in the same way as before compression for analysis via H5Dread, transfer or other operations. When the compressed data in HDF5 are read via H5Dread for analysis, the HDF5 library can automatically find proper decompression library to extract data (assumed that the decompression library are set correctly in the system.).

3.4. Post-hoc DAS Data Compression

In this section, we present the method to parallelize the two-stage DAS data compression method to repack the DAS data. We use the DASSA framework (Dong et al., 2020) as the driver during the parallelization. Specifically, the DASSA framework supports various user-defined function(UDF)s with the help of FasTensor (Dong et al., 2021). We use the UDF to read the DAS data. When the data is written to disk, the compression occurs. The UDF from FasTensor can run in parallel, and therefore the compression implementation on top of it can function in parallel. During the compression, different `H5Dwrite` calls associated with their DAS compression method are assigned to different computing processes on the same compute node or striped across multiple nodes. This utilizes their aggregate computing power and I/O bandwidth. Based on this general idea, we support three different parallelization patterns:

- N:N pattern (shown in Fig. 6 (a)) compresses each input DAS data file separately, where N ($N \in \mathbb{Z}$) is the number of DAS files to compress. Each computing process compresses a file and multiple processes work concurrently. At most N computing processes can be used to archive maximum performance. When the number of computing processes is smaller than N, each process can compress multiple files one by one.
- N:M pattern ($0 < M < N$, $M \in \mathbb{Z}$) concatenates the data first and compresses each concatenated data as a single file. In the example by Fig. 6 (b), two DAS arrays are concatenated; it then calls `H5Dwrite` function and the

DAS compression method. Different concatenated files can be compressed concurrently.

- S:E pattern specifies a start file index (i.e., S) and an end file index (i.e., E) to compress a subset of the whole dataset. ($0 \leq S < E < N$, S and E are zero-based). In the example in Fig. 6 (c), two files (S=1, E= 2) are selected to be compressed separately. These selected files can be merged too before compression.

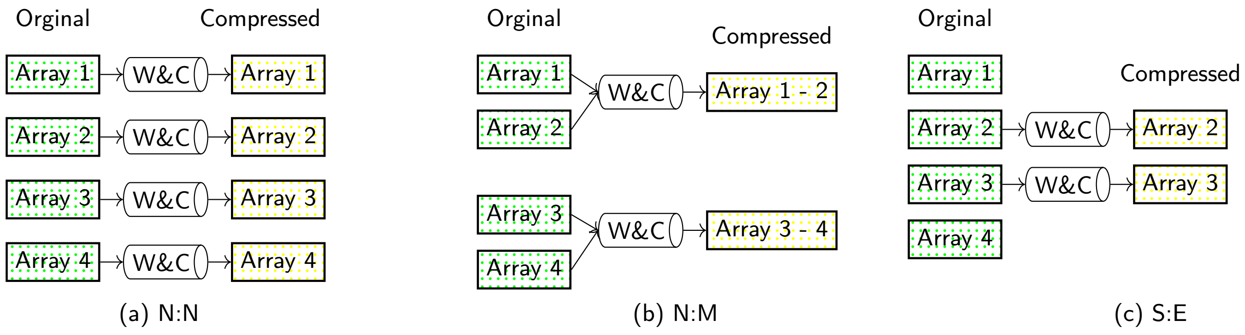


Figure 6: Overview of different supported patterns in the parallel DAS data compression. We demonstrate it with four example files, which are denoted with Array 1, Array 2, Array 3, and Array 4. The W&C in the cylinder refers to `H5Dwrite` call and Compressing, which can be concurrently executed on the path of writing data from memory to disk.

4. Experimental Results

Our experiments utilized the Cori HPC system at NERSC[3]. Details of the DAS data are presented in the previous Table 1. The metric to measure the efficiency of the compression is the compression ratio R: $R = \frac{S_{original}}{S_{compressed}}$, where $S_{original}$ is the size of the original data and $S_{compressed}$ the size of the compressed data. Larger values of R correspond to better compression methods. The $1 - \frac{1}{R}$ gives the percentage of data size reduced by the compression. We also measure and report the compute overhead (i.e., seconds) to compress the data. Although the overhead of decompression could be another factor to consider, the DAS data are more sensitive to the compression overhead, especially in the real-time scenario where limited computing power is available. Our experiments find that this set of compression methods have very similar overheads during compression and decompression (see Appendix 2).

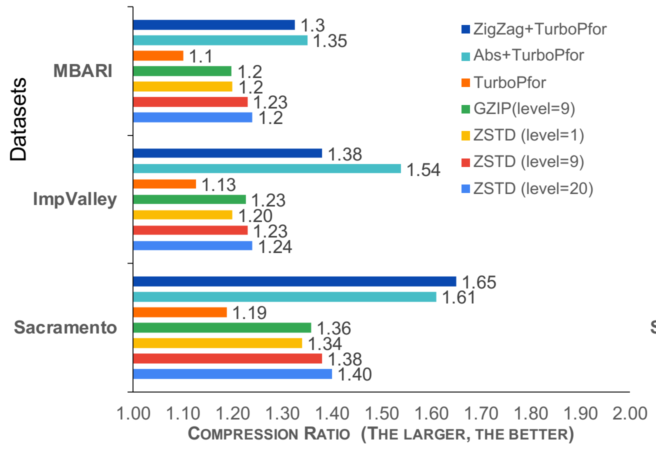
4.1. Comparison of Different Compression Methods

In this experiment, we compress our DAS data with TurboPFor, Abs+TurboPFor, ZigZag+TurboPFor, ZSTD[4] and GZIP[5]. TurboPFor means that we compress the DAS data directly with TurboPFor by treating the negative values as positive values. Abs+TurboPFor and ZigZag+TurboPFor are our two-stage compression methods, which pre-process the data with the **Abs** and **ZigZag** method, respectively, and then use TurboPFor to encode the data. ZSTD has different compression levels. Lower levels give fast compression speed and higher levels give best compression ratio. Our tests

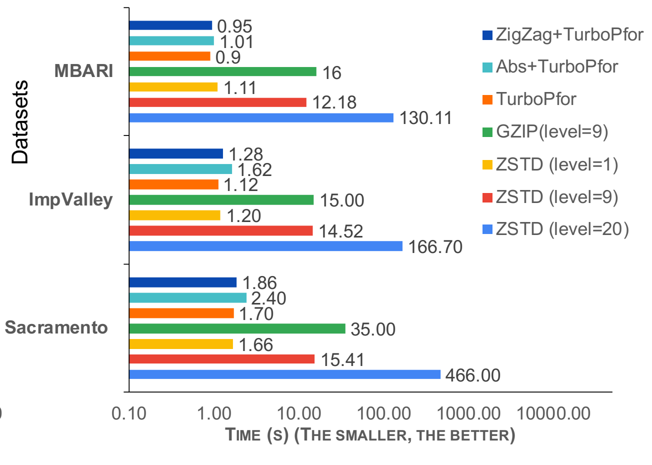
[3] `https://www.nersc.gov/`

[4] ZSTD is obtained at `http://facebook.github.io/zstd/` and it can be used as a HDF5 plugin via HDF5Plugin-Zstandard `https://github.com/aparamon/HDF5Plugin-Zstandard`

[5] GZIP is the H5Z_FILTER_DEFLATE filter avaiable in HDF5



(a)



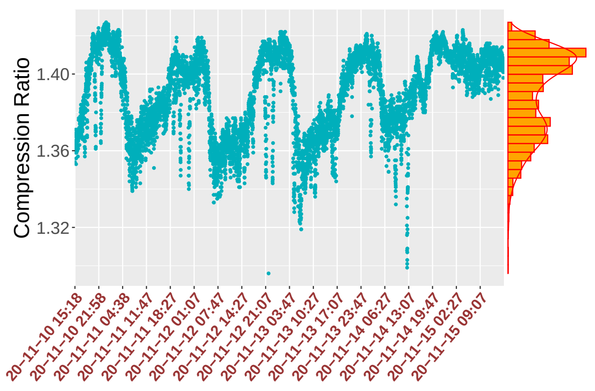
(b)

Figure 7: Evaluation of Different Compression Methods

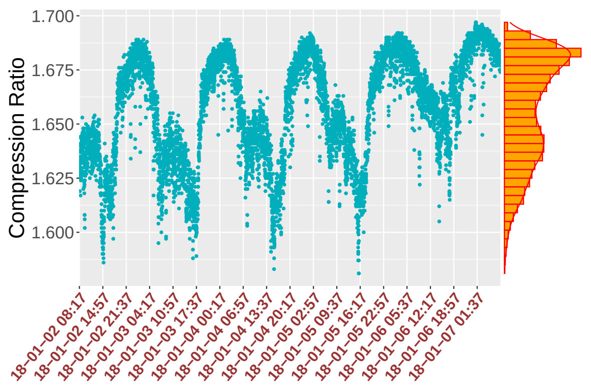
choose level 1, level 9 and level 20 for ZSTD. GZIP has multiple levels too, but our tests find similar results among these levels. Therefore, we only report level 9 for GZIP. We compress each file separately and report the average compression ratio and time here. Details of compression ratio for each file can be found in the following subsection.

As shown by Fig. 7a, TurboPFor yields the lowest compression ratio by compressing the DAS data directly. DAS data has both negative values and positive values. The negative values have a sign-bit as the left-most bit. This sign-bit reduces the chance for TurboPFor to find short bit-packing encoding to reduce its size. By contrast, by pre-processing the DAS data with either **Abs** or **ZigZag**, the compression ratio improves significantly, giving the best compression. Our results suggest that these two methods can reduce the DAS data size by at most 40%. This is consistent with our previous analysis which showed that **Abs** and **ZigZag** can store the sign-bit in separate bitmap or in lower bit and therefore can increase the chance to find small bit-packing representation. The performance of GZIP is very close the ZSTD, which has small variance among different compression levels. These universal compression methods perform better than TurboPFor but still have lower compression ratios than Abs+TurboPFor and ZigZag+TurboPFor. Compared with GZIP and ZSTD, Abs+TurboPFor and ZigZag+TurboPFor can save at least 21% extra space for the Sacramento data and at least 14% extra space for the Imperial Valley data on average.

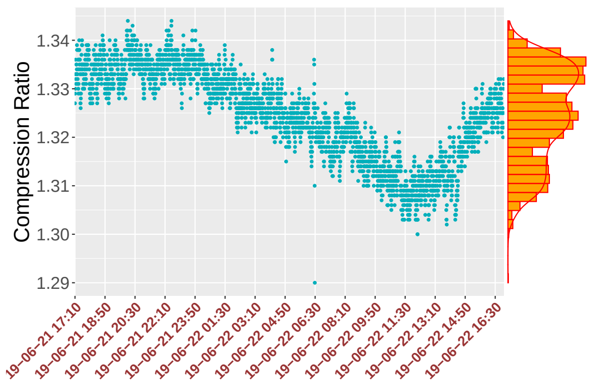
The time to compress DAS data has high variance, as shown by Fig. 7b. ZSTD (level=20) takes the longest time to compress a single file: 566 seconds for the Sacramento data, 166 seconds for the the Imperial Valley data and 130 seconds for the MBARI data. As we stated before, the DAS compression method may work online to process data from the interrogator unit. Given the output interval of the DAS data, there is a one minute upper bound for compression time. ZSTD (level=20) takes more than one minute and therefore cannot be used for real-time DAS data compression. In our tests, the average cost for writing the data is around 2 seconds. Theoretically, all other compression methods can be used to perform online DAS compression. Amongst them, ZSTD (level=1), TurboPFor, Abs+TurboPFor and ZigZag+TurboPFor have very close overhead. Combined with the analysis of the compression ratio presented before, ZigZag+TurboPFor is the best candidate for online DAS compression because of its high compression ratio and low overhead. As a result, we will focus on ZigZag+TurboPFor in following sections.



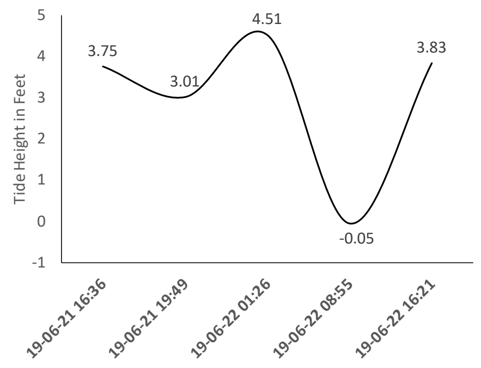
(a) ImpValley Data



(b) Sacramento Data



(c) MBARI Data

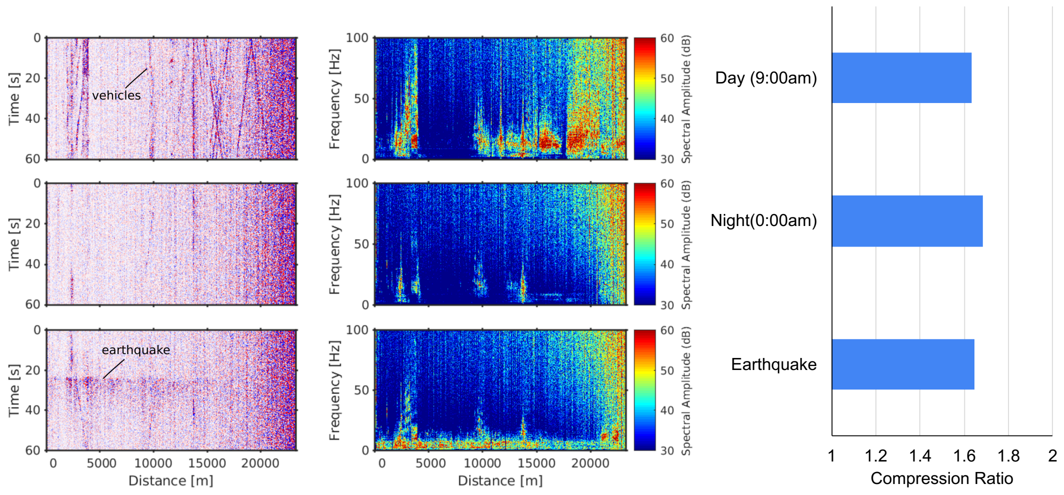


(d) Tide height for the period of the MBARI Data

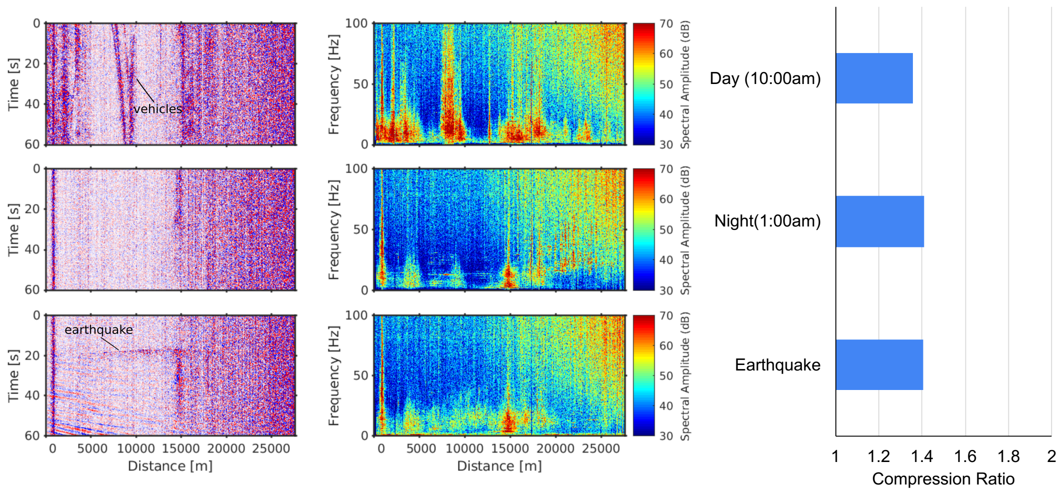
Figure 8: Compression ratio for the ImpValley data (a), the Sacramento data (b) and the MBARI data (c) with ZigZag+TurboPFor. The horizontal axis is the local time (Pacific Standard Time), which is converted from the UTC time zone. We also include tide height in (d) to explain the possible cause for the compression variance in the MBARI data.

4.2. Compression Ratio for the Whole Dataset

The compression ratio for the whole DAS data is reported in Fig. 8, where each point represents compression ratio for a file and the x-axis is timestamp (Pacific Standard time). The Sacramento data has a higher compression ratio than the ImpValley data. One possible reason is that the Sacramento data is collected from rural area with less activities around the optical fiber cable and hence less broadband ambient noise. Also, the Sacramento data has a smaller absolute magnitude than the ImpValley data. Both the Sacramento data and the ImpValley data have clear day-night patterns: the data collected at the nighttime has higher compression ratio than the data collected at the daytime. This is likely



(a) Sacramento data on the date 01/04/2018. The one minute data at day time is sampled at 9:00am local time with traffic noises. The one minute data at night time is obtained at 0:00am with only only noise. The bottom one minute data contains a earthquake M 4.4 - 2km SE of Berkeley, 01/04/2018 10:39:37 (UTC).



(b) ImpValley Data on the date 11/13/2020. The one minute data at day time is sampled at 10:00am local time with traffic noises. The one minute data at night time is obtained at 1:00am with only only noise. The bottom one minute data contains a earthquake M 5.3 - 33 km SE of Mina, Nevada, 11/13/2020 09:13:51 (UTC).

Figure 9: Raw data (left), spectrum plot (middle) and compression ratio (right) for the Sacramento and ImpValley. Each row data on the right. The top row is the one minute data at day time with traffic noises. The middle row is the one minute data at night time with only only noise. The bottom row is the one minute data with earthquake wave.

due to higher levels of seismic noise during the daytime, linked to anthropogenic activity. Peak compression ratios are achieved at midnight when only low levels of anthropogenic noise are present. In Fig. 9, we present the raw data, spectrum component, and compression ratio for the Sacramento data and the ImpValley data, which are consistent with our analysis. For the summary histogram on the right margin, there is no clear distribution observed in the plot. The compression ratio has long tails towards the smaller values, which represents the low compression ratio during the daytime. Compression ratio for the MBARI is presented and analyzed in Fig. 8c. Clearly, the MBARI data has different pattens from the previous two data sets. One major reason is the MBARI is collected from fiber under the seawater. The noise from MBARI are impacted by water depth and water movement. There seems to be a correlation between lower compression ratio and low tide, as shown in Fig. 8d. At low tide, the spread of the values seems slightly larger (see Fig. 14 in Appendix 4). Previous analysis (Lindsey et al., 2019b) show small differences in the frequency content of the data at high tide, compared with low tide. These differences could be an explanation for this very small difference in compression ratio. A further analysis are needed to confirm the correlation between the compression ratio and tide or other factors.

4.3. Scalability of DAS Compression Methods

Fig 10 reports test results from compressing our DAS data with the ZigZag+TurboPFor using different number of computing processes. The performance measurement is the speed: $\frac{Total\ Data\ Size\ (GB)}{Time\ to\ read,\ compress\ and\ write\ the\ data\ (Sec)}$. Each process is equal to a physical CPU core. Take the 256 CPU processes and Sacramento data (/w 7200 files) as an example. We run these tests with N:N pattern, where 256 files are compressed concurrently and each file is compressed by one CPU process. For all 2880 files, each CPU process compresses at most 29 files. These experimental results clearly show that our compression method can scale linearly from 8 CPU processes to 256 CPU processes. For the 256 CPU processes, the speed is 20GB/second for the Sacramento data and 26GB/second for the ImpValley data. The I/O cost to access the Sacramento data, whose size is larger than that of the ImpValley data, make the scaling test results show high variance.

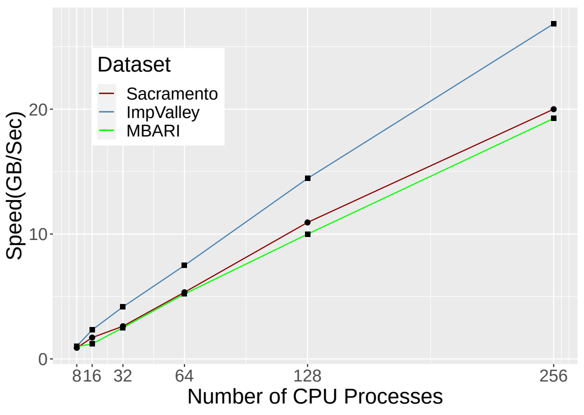


Figure 10: Scalability of Compression Method

5. Conclusions

The volumes of Distributed Acoustic Sensing (DAS) data are very large (TB/day). Efficient compression of DAS data can significantly reduce the cost of storage and transfer. This paper focuses on compressing raw DAS data as directly acquired by the DAS interrogator unit, where optical phase is represented as short integer and has both negative and positive values. We develop a two-stage compression method to convert negative values into positive ones and use a bit-packing method to encode them. We also develop an in-situ method for HDF5 file format and a parallelization method to compress DAS data in a real-time and post-hoc manner, respectively. Experimental results show that the two-stage compression method can save 40% of the size of the DAS dataset, which is considerable given the 0.5 PB/yr acquisition streams generated by single DAS arrays. For an installation that require disk retrieval every five months, this compression could extend the retrieval time to seven months. Our method also scales linearly as we use more compute nodes, delivering $26GB/seconds$ peak compression performance. Future work includes research and development of float point and lossy compression methods such as SZ and ZFP (Zhao et al., 2020; Diffenderfer et al., 2019; Kingma et al., 2019) for DAS data.

6. Acknowledgments

The Imperial Valley Dark Fiber Team includes Avinash Nayak, Patrick Dobson, Feng Cheng, Michelle Robertson, Inder Monga, Cody Rotermund, Robert Mellors, Yucheng Shang, Benxin Chi, Emily Maher, Todd Wood, Christina Morency, Eric Matzel, Elisabet Metcalfe, Lindsay Morse, and Dennise Templeton. We thank anonymous reviewers, the Associate Editor, and Editor for carefully reviewing this study and for their comments and suggestions.

This effort was supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research (ASCR) and Office of Energy Efficiency and Renewable Energy (EERE), Office of Technology Development, Geothermal Technologies Office, under contract number DE-AC02-05CH11231 with LBNL. This research used resources of the National Energy Research Scientific Computing Center (NERSC).

Computer Code Availability

- The source code used in this paper has two parts and both are open-sourced under a modified BSD license:
  - H5TuorbPFOR: The two-stage DAS data compression method is implemented as a HDF5 plugin. The H5TuorbPFOR source code and installation steps can be accessed at the public repository:
    `https://github.com/dbinlbl/H5TurboPFor`.
    Within the repository, we provide example Jupyter Notebook and C/C++ code to demonstrate how to use the compression method.
  - DASSA: The parallel DAS data compression method is implemented in DASSA. The DASSA source code can be accessed at the public repository:
    `https://bitbucket.org/dbin_sdm/dassa/`
- Program language: C/C++
- Operating System: Linux/MacOS/Unix
- Software required: H5TuorbPFOR needs TuorbPFOR and HDF5; DASSA needs H5TuorbPFOR.

Appendix 1: Impact of the subset size and merge size on compression ratio.

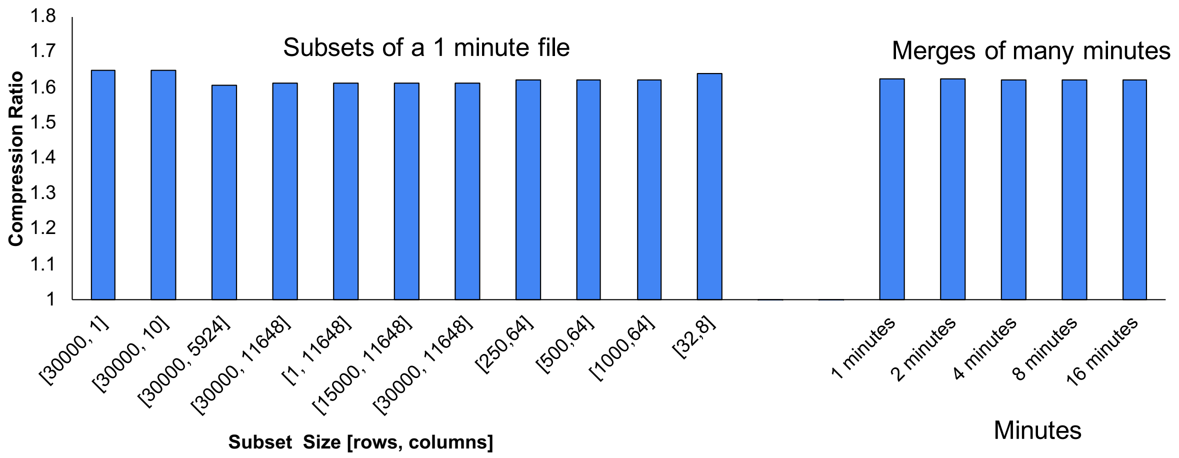


Figure 11: Measuring the impact of different subset sizes (left) amd merging sizes (right) on compression ratio for the Sacramento data. We measured the compression ratio for the ZigZag+TurboPFor method. The subset sizes tested include wide variations, from compressing 1 time series per time, i.e., [30000, 1], to compressing all channels at one time, [1, 11648]. It suggests that compressing along time series yields a small performance advantage. We also test the size by merging different 1 minutes files together as input of compression. But, in call cases, the chunk size have almost ignorable impact on the compression ratio. The same pattern is found on the ImpValley data.

341

342 ## Appendix 2: Time for DAS data compression and decompression

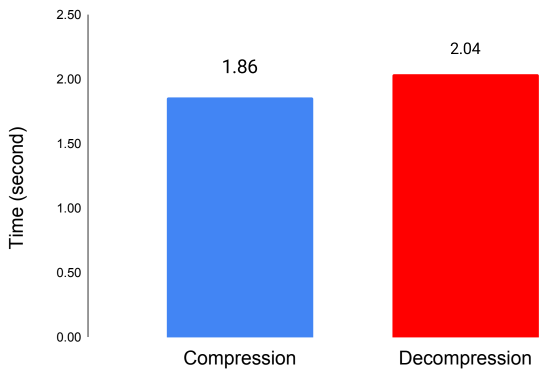
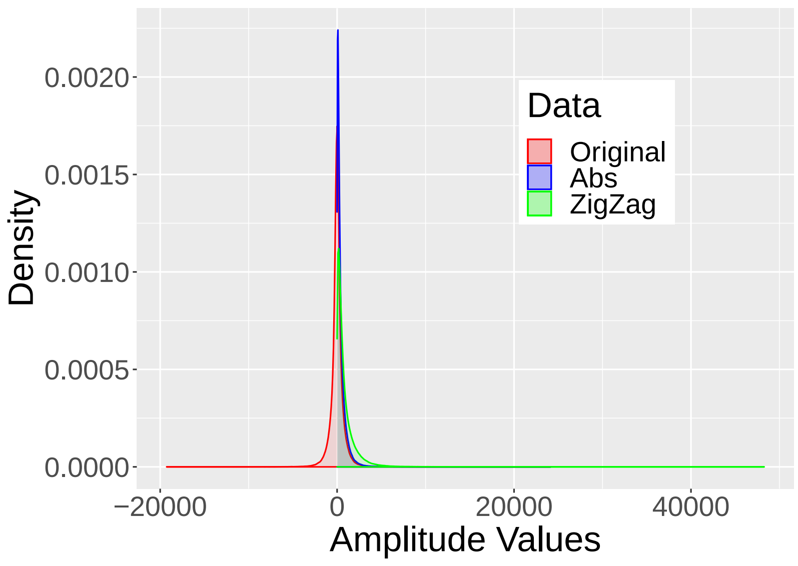
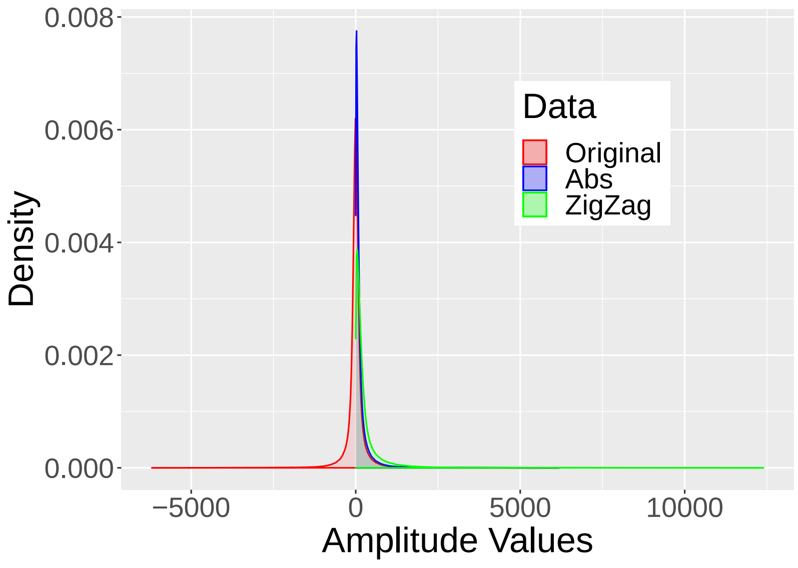


Figure 12: The average overhead of compressing and decompressing a file from the Sacramento data. We measured the compression and decompression time for the ZigZag+TurboPFor method. The compression and decompression have very close performance. The same pattern is found on the ImpValley data and other compression methods.

343 **Appendix 3: Full density plot for the DAS data from Sacramento and ImpValley.**



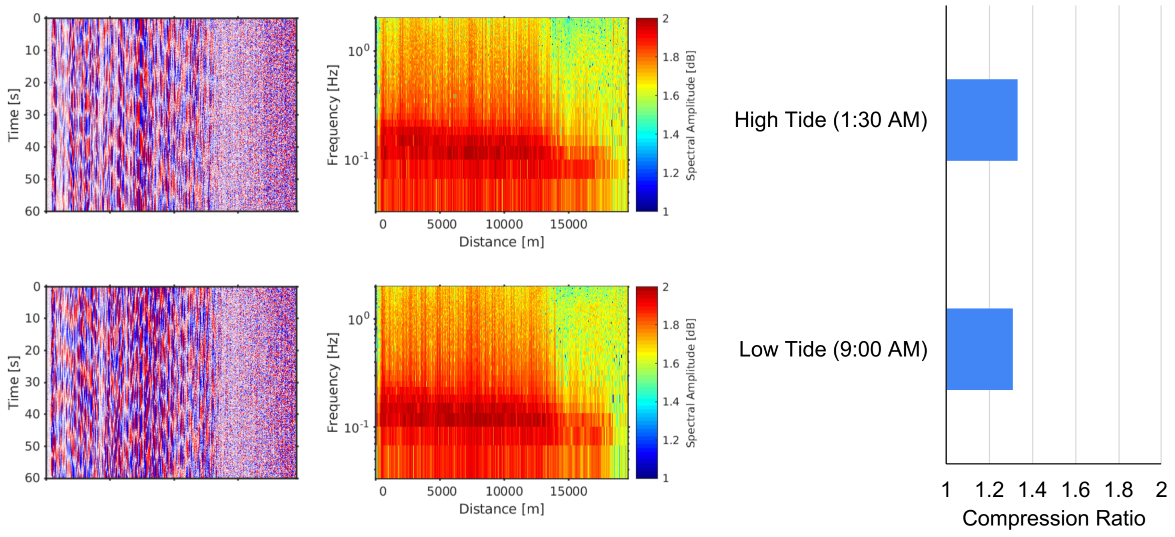
(a) ImpValley Data: Density



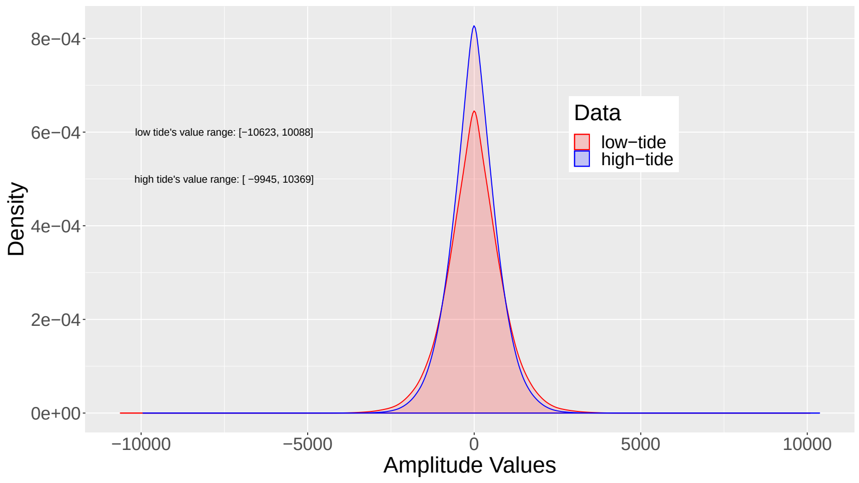
(b) Sacramento Data : Density

Figure 13: Full statistical density plot of DAS data from Sacramento and ImpValley. The data file from Sacramento dataset is 01/04/2018 23:35:31 and from the ImpValley data set is 11/12/2020 18:41:32.

344 **Appendix 4: Spectral and compression ratio for MBARI data.**



(a) Raw data, spectrum and compression ratio for data at high tide and low tide.



(b) Density of amplitude values for data at low tide and high tide.

Figure 14: Raw, spectrum, compression ratio, and density of amplitude for two files from MBARI datasets. The low tide file is the one minute data at local 9:00am (UTC time is: 06/22/2019 16:00:04); the high tide file is the one minute data at local 1:30am (UTC time is: 06/22/2019 08:30:04).

References

Ajo-Franklin, J., Dou, S., Daley, T., Freifeld, B., Robertson, M., Ulrich, C., Wood, T., Eckblaw, I., Lindsey, N., Martin, E., et al., 2017. Time-lapse surface wave monitoring of permafrost thaw using distributed acoustic sensing and a permanent automated seismic source, in: SEG Technical Program Expanded Abstracts 2017. Society of Exploration Geophysicists.

Ajo-Franklin, J.B., , Rodríguez Tribaldos, V., Nayak, A., Cheng, F., Mellors, R., Chi, B., Wood, T., Robertson, M., Rotermund, C., Matzel, E., Templeton, D.C., Morency, C., Wu, K., Dong, B., Dobson, P., in press. The Imperial Valley Dark Fiber Project: Towards Seismic Studies Using DAS and Telecom Infrastructure for Geothermal Applications. Seismological Research Letters Data Mine .

Ajo-Franklin, J.B., Dou, S., Lindsey, N.J., Monga, I., Tracy, C., Robertson, M., Rodriguez Tribaldos, V., Ulrich, C., Freifeld, B., Daley, T., Li, X.,

2019. Distributed Acoustic Sensing Using Dark Fiber for Near-Surface Characterization and Broadband Seismic Event Detection. Scientific Reports 9. doi:|https://doi.org/10.1038/s41598-018-36675-8.

Collet, Y., Kucherawy, M.S., 2021. Zstandard compression and the 'application/zstd' media type. RFC 8878, 1–45. URL: https://doi.org/10.17487/RFC8878, doi:10.17487/RFC8878.

Deutsch, L.P., 1996. RFC 1951: DEFLATE compressed data format specification version 1.3. URL: ftp://ftp.internic.net/rfc/rfc1951.txt;http://www.ietf.org/rfc/rfc1951;http://www.math.utah.edu/pub/rfc/rfc1951.txt. status: INFORMATIONAL.

Diffenderfer, J.D., Fox, A.L., Hittinger, J.A., Sanders, G.D., Lindstrom, P.G., 2019. Error analysis of zfp compression for floating-point data. SIAM Journal on Scientific Computing 41. doi:10.1137/18M1168832.

Dong, B., Tribaldos, V.R., Xing, X., Byna, S., Ajo-Franklin, J., Wu, K., 2020. Dassa: Parallel das data storage and analysis for subsurface event detection, in: 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 254–263. doi:10.1109/IPDPS47924.2020.00035.

Dong, B., Wu, K., Byna, S., 2021. User-Defined Tensor Data Analysis, 2. Springer Briefs in Computer Science, Springer.

Feigl, K., 1969. Porotomo natural laboratory horizontal and vertical distributed acoustic sensing data doi:10.15121/1721671.

Feigl, K., 2016. Brady's geothermal field das earthquake data doi:10.15121/1334285.

Foukas, X., Patounas, G., Elmokashfi, A., Marina, M.K., 2017. Network slicing in 5g: Survey and challenges. Comm. Mag. 55, 94–100. URL: https://doi.org/10.1109/MCOM.2017.1600951, doi:10.1109/MCOM.2017.1600951.

Google, 2021. Encoding in Protocol-buffers. https://developers.google.com/protocol-buffers/docs/encoding. [Online; accessed 7-Nov-2021].

Hartog, A.H., 2017. An introduction to distributed optical fibre sensors. CRC press.

Higham, D.J., Higham, N.J., 2016. MATLAB guide. volume 150. Siam.

Holtz, K., 1993. The evolution of lossless data compression techniques, in: Proceedings of WESCON '93, pp. 140–145. doi:10.1109/WESCON.1993.488424.

Huffman, D.A., 1952. A method for the construction of minimum-redundancy codes. Proceedings of the IRE 40, 1098–1101. doi:10.1109/JRPROC.1952.273898.

Ibrahim, A.D.A., Lin, S., Xiong, J., Jiang, J., Fu, Y., Wang, Z., 2020. Integrated principal component analysis denoising technique for phase-sensitive optical time domain reflectometry vibration detection. Appl. Opt. 59, 669–675. URL: http://www.osapublishing.org/ao/abstract.cfm?URI=ao-59-3-669, doi:10.1364/AO.59.000669.

Jayasankar, U., Thirumal, V., Ponnurangam, D., 2021. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. Journal of King Saud University - Computer and Information Sciences 33, 119–140. URL: https://www.sciencedirect.com/science/article/pii/S1319157818301101, doi:https://doi.org/10.1016/j.jksuci.2018.05.006.

Kingma, F.H., Abbeel, P., Ho, J., 2019. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables, in: International Conference on Machine Learning.

Kinsner, W., Greenfield, R., 1991. The lempel-ziv-welch (lzw) data compression algorithm for packet radio, in: [Proceedings] WESCANEX '91, pp. 225–229. doi:10.1109/WESCAN.1991.160551.

Lemire, D., Boytsov, L., 2015. Decoding billions of integers per second through vectorization. Software: Practice and Experience 45, 1–29. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2203, doi:https://doi.org/10.1002/spe.2203, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2203.

Liaw, W.M., 1995. Reading GIF files. Dr. Dobb's Journal 20, 56, 58, 60, 103, 106–107.

Liehr, S., Borchardt, C., Münzenberger, S., 2020. Long-distance fiber optic vibration sensing using convolutional neural networks as real-time denoisers. Opt. Express 28, 39311–39325. URL: http://www.osapublishing.org/oe/abstract.cfm?URI=oe-28-26-39311, doi:10.1364/OE.402789.

Lindsey, N., Dawe, C., Ajo-Franklin, J., 2019a. Photonic seismology in monterey bay: Dark fiber DAS illuminates offshore faults and coastal ocean dynamics URL: https://doi.org/10.31223%2Fosf.io%2F7bf92, doi:10.31223/osf.io/7bf92.

Lindsey, N.J., Dawe, T.C., Ajo-Franklin, J.B., 2019b. Illuminating seafloor faults and ocean dynamics with dark fiber distributed acoustic sensing. Science 366, 1103–1107. URL: https://www.science.org/doi/abs/10.1126/science.aay5881, doi:10.1126/science.aay5881, arXiv:https://www.science.org/doi/pdf/10.1126/science.aay5881.

Lindsey, N.J., Rademacher, H., Ajo-Franklin, J.B., 2020. On the broadband instrument response of fiber-optic DAS arrays. Journal of Geophysical Research: Solid Earth 125, e2019JB018145.

MacKay, D.J.C., 2002. Information Theory, Inference; Learning Algorithms. Cambridge University Press, USA.

Meng, Y., Zha, J., Liu, Y., 2019. Intensifying the SNR of BOTDA using adaptive constrained least squares filtering. Optics Communications 437, 219–225. doi:10.1016/j.optcom.2018.12.073.

Mittal, S., Vetter, J.S., 2016. A survey of architectural approaches for data compression in cache and main memory systems. IEEE Trans. Parallel Distributed Syst. 27, 1524–1536.

Paitz, P., Edme, P., Gräff, D., Walter, F., Doetsch, J., Chalari, A., Schmelzbach, C., Fichtner, A., 2021. Empirical investigations of the instrument response for distributed acoustic sensing (DAS) across 17 octaves. Bulletin of the Seismological Society of America 111, 1–10.

Powturbo, 2021. Turbopfor-integer-compression. https://github.com/powturbo/TurboPFor-Integer-Compression.

Pylak, P., 1990. Efficient modification of LZSS compression algorithm. Annales UMCS Informatica, section AI , 61–72URL: http://www.annales.umcs.lublin.pl/AI/2003/07.pdf.

Qin, Z., Chen, H., Chang, J., 2017. Detection performance improvement of distributed vibration sensor based on curvelet denoising method. Sensors 17. URL: https://www.mdpi.com/1424-8220/17/6/1380, doi:10.3390/s17061380.

Rodríguez Tribaldos, V., Lindsey, N.J., Dou, S., Ulrich, C., Robertson, M., Dong, B., Dumont, V., Wu, K., Monga, I., Tracy, C., Ajo-Franklin, J.B., 2020. Combining Ambient Noise and Distributed Acoustic Sensing (DAS) Deployed on Dark Fiber Networks for High-resolution Imaging at the Basin Scale, in: AGU Fall Meeting Abstracts, pp. S023–04.

Shiloh, L., Eyal, A., Giryes, R., 2018. Deep learning approach for processing fiber-optic das seismic data, in: 26th International Conference on Optical Fiber Sensors, Optical Society of America. p. ThE22. URL: http://www.osapublishing.org/abstract.cfm?URI=OFS-2018-ThE22, doi:10.1364/OFS.2018.ThE22.

Smith, C.A., 2010. A survey of various data compression techniques.

Soto, M.A., Ramírez, J.A., Thévenaz, L., 2016. Intensifying the response of distributed optical fibre sensors using 2d and 3d image restoration. Nature Communications 7, 1–11. URL: https://EconPapers.repec.org/RePEc:nat:natcom:v:7:y:2016:i:1:d:10.1038_ncomms10870.

The HDF Group, 2010. HDF5 User Guide.

Trotman, A., 2014. Compression, simd, and postings lists, in: Proceedings of the 2014 Australasian Document Computing Symposium, Association for Computing Machinery, New York, NY, USA. p. 50–57. URL: https://doi.org/10.1145/2682862.2682870, doi:10.1145/2682862.2682870.

Verdon, J.P., Horne, S.A., Clarke, A., Stork, A.L., Baird, A.F., Kendall, J.M., 2020. Microseismic monitoring using a fiber-optic distributed acoustic sensor array. GEOPHYSICS 85, KS89–KS99. URL: https://doi.org/10.1190/geo2019-0752.1, doi:10.1190/geo2019-0752.1, arXiv:https://doi.org/10.1190/geo2019-0752.1.

Verónica, R.T., Jonathan, A.F., 2021. Aquifer monitoring using ambient seismic noise recorded with distributed acoustic sensing (das) deployed on dark fiber. Journal of Geophysical Research: Solid Earth 126, e2020JB021004. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020JB021004, doi:https://doi.org/10.1029/2020JB021004, arXiv:https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2020JB021004. e2020JB021004 2020JB021004.

Xing, X., Dong, B., Ajo-Franklin, J., Wu, K., 2018. Automated parallel data processing engine with application to large-scale feature extraction, in: MLHPC 2018, pp. 37–46. doi:10.1109/MLHPC.2018.8638638.

Zhan, Z., 2019. Distributed Acoustic Sensing Turns Fiber-Optic Cables into Sensitive Seismic Antennas. Seismological Research Letters 91, 1–15. URL: https://doi.org/10.1785/0220190112, doi:10.1785/0220190112, arXiv:https://pubs.geoscienceworld.org/ssa/srl/article-pdf/91/1/1/4912248/srl-2019112.1.pdf.

Zhao, K., Di, S., Liang, X., Li, S., Tao, D., Chen, Z., Cappello, F., 2020. Significantly improving lossy compression for hpc datasets with second-order prediction and parameter optimization, in: Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing, Association for Computing Machinery, New York, NY, USA. p. 89–100. URL: https://doi.org/10.1145/3369583.3392688, doi:10.1145/3369583.3392688.

Ziv, J., Lempel, A., 1977. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory 23, 337–343. doi:10.1109/TIT.1977.1055714.