Integrating energy datasets: the MDIO format

Altay Sansal[1], Ben Lasscock[1] and Alejandro Valenciano[1*] detail the MDIO format and specification and review five energy data applications: wind resource assessment, seismic data management, data processing, machine learning, and seismic data visualisation.

Abstract

MDIO offers a technical solution for storing and retrieving energy data in the cloud and on-premises. As an open-source framework, it incorporates high-resolution, multi-dimensional arrays that accurately represent wind resources and seismic data for multiple applications. By utilising the Zarr format, MDIO ensures efficient chunked storage and parallel I/O operations, facilitating easy data interaction in diverse infrastructures. This paper covers MDIO's application in renewable energy (wind simulations), predictive analytics, and seismic imaging and interpretation, aiming to provide a robust technical platform for researchers navigating the evolving energy landscape.

Introduction

The energy sector is deeply anchored in data, guiding exploration, production, and asset management strategies. Numerous data formats have emerged to support diverse scientific and business needs. NetCDF4 (Unidata, 2021) and HDF5 (The HDF Group, 1997-2023) are prominent in atmospheric sciences and climate modelling domains. SEG-Y stands out for seismic data storage in exploration geophysics, while academic circles have adopted formats such as SEP, SU, and Madagascar for seismic research. Additionally, commercial ventures have brought forth and later open-sourced formats such as Data Dictionary System (DDS, freedds.org), OpenVDS (Barre et al., 2022), and OpenZGY (OSDU, 2023). We've aimed to encapsulate big binary storage formats but acknowledge the possibility of omissions and welcome reader insights on any overlooked formats for subsequent revisions.

This article details the MDIO format and specification and reviews five energy data applications: wind resource assessment, seismic data management, data processing, machine learning, and seismic data visualisation.

MDIO format

MDIO (Sansal et al., 2023) is a file format that simplifies the access of seismic and wind data through Zarr (Miles et al., 2023) arrays and JSON metadata. This format enables storing data and metadata within the same file structure, eliminating the need for a separate database. MDIO makes sharing and analysing data easier, enabling lossless and lossy data compression. Additionally, MDIO can transform irregular data into a regularised hypercube without taking up extra storage space. This is possible by integrating the Segyio library (Equinor, 2023) for parsing SEG-Y files and writing text and binary headers.

Furthermore, MDIO provides a standard API that can read, write, and perform tensor operations on data, regardless of its storage location, such as network file storage or cloud object stores. The library also includes converters for popular file formats such as SEG-Y and NetCDF4. Seismic and wind projects often have similar data patterns, including multiple datasets that share the same grid, coordinate reference system information and time/depth series of spatial data. Although the workflow depends on the specific project, MDIO can help to make it easier to manage and analyse data.

MDIO specification

In this section, we will provide a brief overview of the seismic data specification stored as MDIO. The MDIO format was developed as an extension of the Zarr protocol. Figure 1 illustrates the current specification (MDIO 0.4.2), which includes the following key features:

1. **Global attributes:** Includes information about the creation of the data set, details on the geometry, and global statistics such as minimum, maximum, root-mean-square (RMS), mean, and standard deviation values of samples.
2. **Metadata group:** This group contains text, binary, trace headers, and a live trace mask. These metadata elements hold additional information about the data, such as acquisition parameters, processing history, and quality control flags. Horizons associated with the seismic grid can also be stored together here. It should be held as individual maps (2D arrays) within the metadata group, given the current MDIO schema.
3. **Data group:** The group contains multi-dimensional arrays that users can divide into one or more chunks and store as Zarr arrays. These arrays have seismic data or any other multi-dimensional array representing other attributes. The user can choose the size of the chunks and compress the data in a lossless or lossy manner. Compression algorithms supported include Blosc, Zstd (Collet et al., 2018), LZ4 (Collet et al., 2011), and ZFP (Lindstrom, 2014).

MDIO simplifies the process for users by providing optimised default chunk dimensions for various seismic and wind data

[1] TGS
[*] Corresponding author, E-mail: Alejandro.Valenciano@tgs.com
DOI: 10.3997/1365-2397.fb2023084

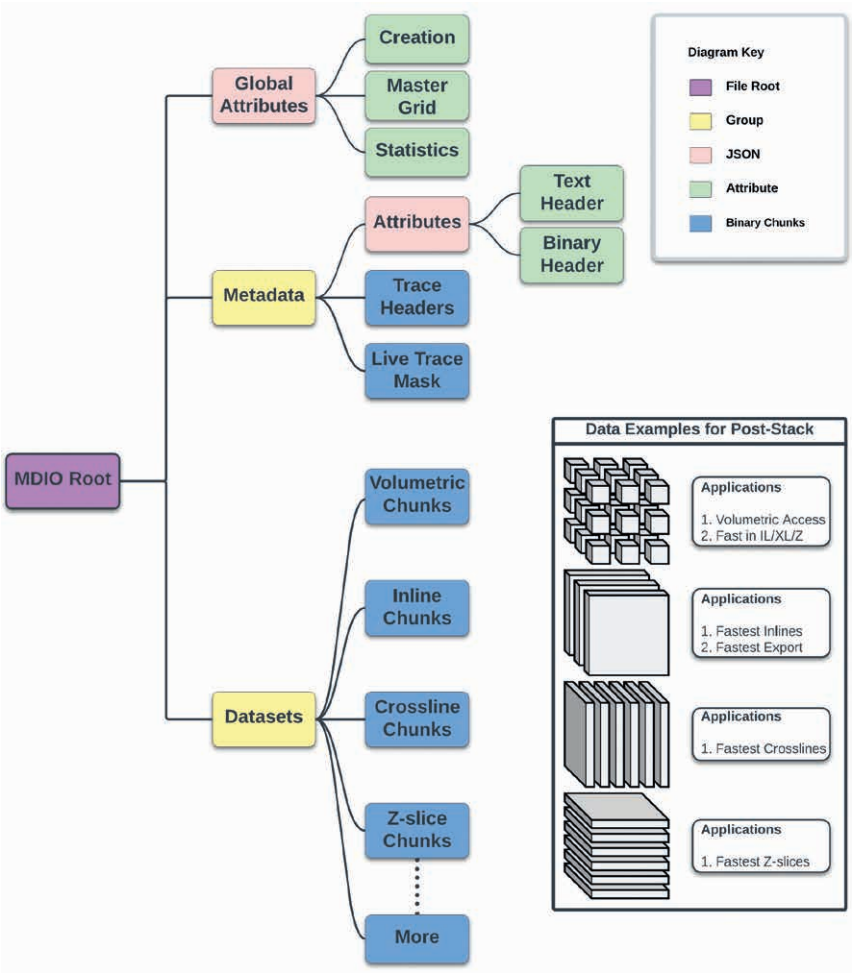

Figure 1 MDIO seismic data specification, v0.4.2. We show various access patterns for 3D seismic datasets.

types. Adjusting these settings can enhance performance for specific workflows. When using lossy compression options, it is possible to increase the chunk dimensions for transferring more data segments at once. However, maintaining precise chunk sizes can result in faster loading times when an application has bandwidth limitations. It is crucial to note that there is a trade-off between chunk sizes and the number of chunks, which can impact performance if not adjusted correctly. We have chosen the following chunk sizes based on benchmarks and heuristics to cater for three primary workloads: archival I/O performance (including data ingestion and export), the flexibility to re-chunk for specialised cases in the future, and satisfactory visualisation performance in the direction users explore data.

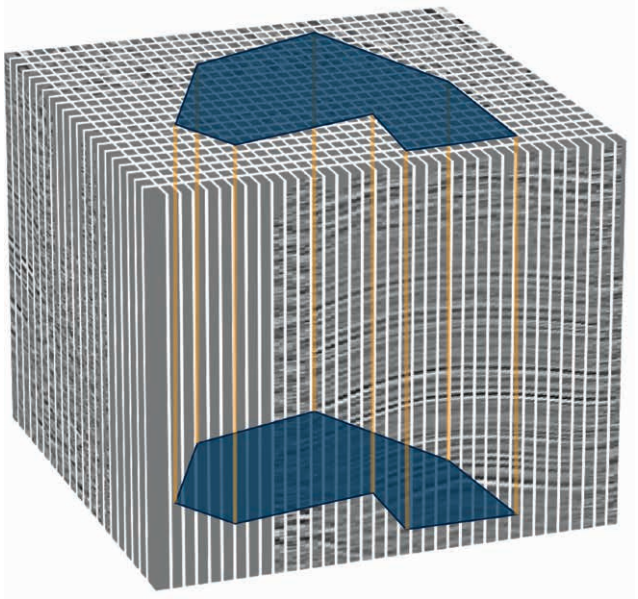

Figure 2 Sample plots of aggregated statistics queried in real-time by a web application from a cloud object store using an optimised MDIO file.

Users can install MDIO, an open-source library under the Apache 2.0 licence, through widely used channels like Conda and Pip. For more information, visit: https://mdio.dev.

Applications

Wind resource assessment

In the rapidly advancing renewable energy domain, accurately gauging the potential output of emerging wind farms is pivotal, directly influencing crucial factors such as lease bids, deployment outlays, and overall ROI. Like oil and gas exploration methodologies, wind farm assessments synergise historical data with intricate physics-based modelling. Advanced tools, such as the WRF modelling system (Skamarock et al., 2019), produce vast datasets spanning terabytes attributed to their complex spatiotemporal simulations. These datasets, whether highly detailed for specific regions or more general for broader areas are then processed to offer monthly and yearly aggregates. While computations are inherently parallel, they demand a

sophisticated data access approach due to monthly or hourly grouping requirements.

Recognising the limitations inherent in the commonly utilised NetCDF4 format, transitioning to MDIO has ushered in a new era of data I/O optimisation. Coupling this with the capabilities of Zarr and Dask (Rocklin, 2015) has resulted in a substantial boost in processing efficiency, with improvements registering at an impressive 100x speed increase. Furthermore, adopting cloud-native MDIO format on Google Cloud Storage, combined with the automation and scalability offered by Python's Dask library and Google's Cloud Composer, has allowed us to build a high-performance wind data processing pipeline. This holistic approach has not only heightened read performance but also significantly trimmed down processing durations from weeks to a matter of hours. Figure 2 shows a sample of the aggregated statistics queried in real time by a web application with an MDIO back end.

Seismic data management

Managing seismic datasets stored in the SEG-Y format can be complex. Operations centered on data management, particularly when handling petabyte-scale data, often require indexing trace data headers to support varied spatial search queries. Common reliance on relational databases for this purpose exacerbates costs and introduces operational complexities, with the inherent design of SEG-Y emphasising sequential access. This design often becomes inefficient, especially for applications like imaging and visualisation, when demands turn to accessing individual traces or header samples in cloud object stores. The industry is leaning towards cloud-based storage solutions, recognising the imperatives of cost-effectiveness and scalability.

MDIO addresses these challenges. On average, it delivers a storage cost reduction of 41% using Zstd lossless compression on petabyte-scale seismic datasets. The advantage of MDIO isn't limited to cost savings: its chunking mechanism ensures that the entire dataset needn't be decompressed for quality checks. An essential workflow involves subsetting vast data volumes via a geospatial polygon. MDIO's unique masking capability ensures that only pertinent traces and headers are retained during data retrieval or re-encoding to SEG-Y, effectively obfuscating irrelevant sections. This flexible masking also extends to time/depth data modulation. Figure 3 demonstrates this concept with the polygon and the optimal chunk layout.

Another compelling capability is seismic streaming facilitated by MDIO. This service optimises on-demand data access. Collaboratively, MDIO and Dask ensure a robust backend performance, efficiently managing requests from object stores. The advantage? Users are presented with a lightweight client library tailored to their requirements. Although alternative solutions like async-Zarr exist, the dependable Dask library, particularly MDIO's native Dask backend, offers a more established approach to distributing workloads, revealing Zarr arrays as lazy and compatible with async processes.

Seismic imaging and processing

In seismic imaging and processing, several factors determine the viability of the MDIO file format. There is a need to support both local HPC and cloud computing. A file format must support high-performance read/write of seismic data, be interoperable with existing software, and be extensible with new technological developments.

Many seismic imaging and processing platforms adopt a hybrid model of on-premises computing with scalability to a private cloud. For this model to work effectively, the seismic data format must support both scenarios seamlessly, with applications potentially needing to access data from a local file system, remote object store, or a combination of the two. MDIO provides a unified set of APIs allowing data access and manipulation from both sources. As discussed, MDIO also supports both lossy and lossless data compression, which can reduce the overall storage costs associated with data and improve the system's throughput. In algorithms like POCS (Abma and Kabir, 2006), the data can be read in memory once, efficiently, and the output could again be written to a chunk without locks, disc seeking, or using many web requests. Figure 4 demonstrates a processing application that computes windowed 3D Fourier transforms.

In summary, chunk-aligned write operations can be safely executed in parallel across multiple processes and efficiently

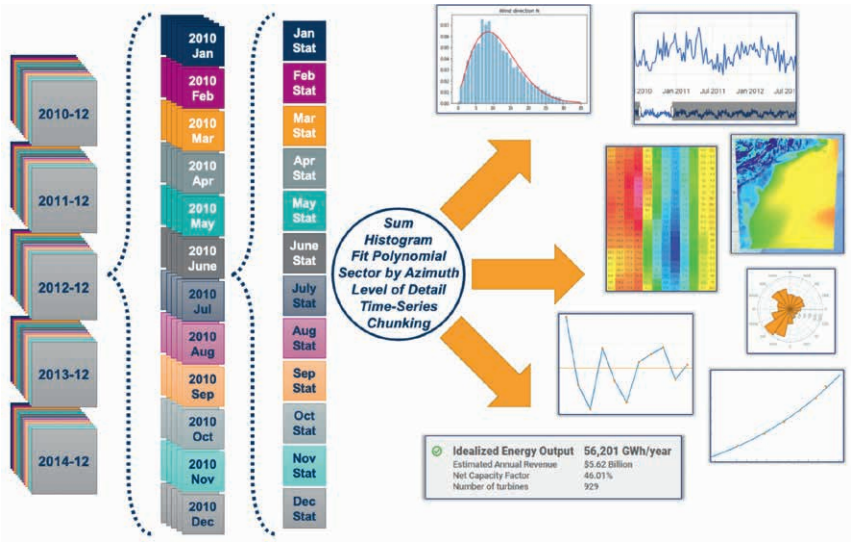

Figure 3 Concept of polygon-limited seismic data retrieval with spatial chunks.

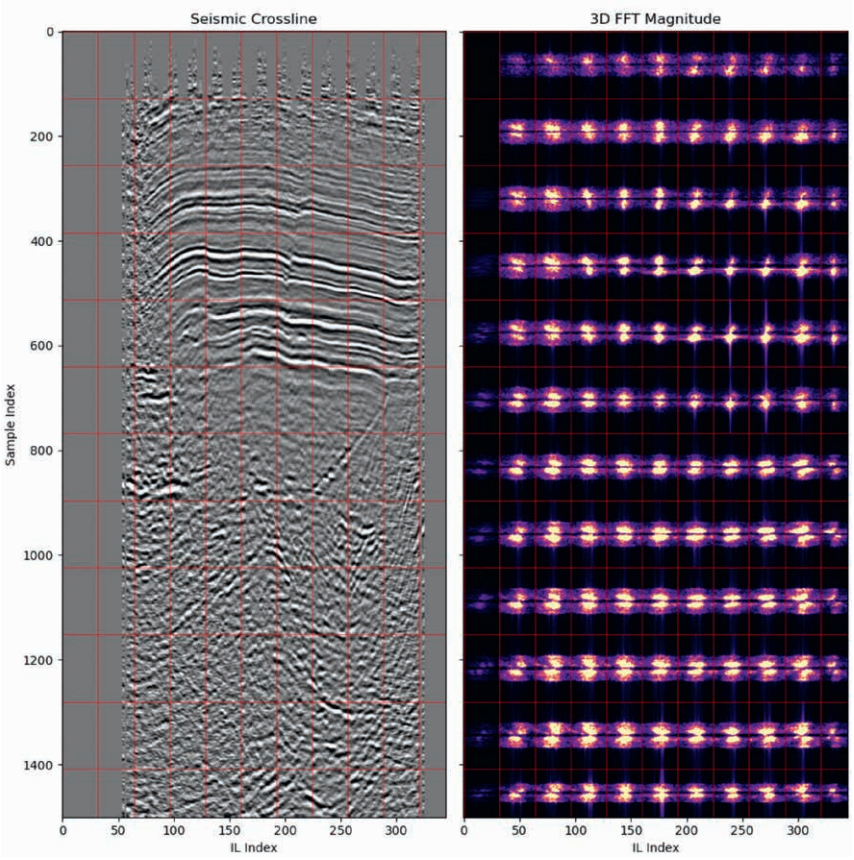

Figure 4 3D Fast Fourier Transforms are applied to 3D chunks, as illustrated on the left, with an overlay of chunk boundaries on the seismic image. The right side displays the central wavenumber in the crossline direction ($K_{crossline}$). The local coordinate axes for each chunk's resulting FFT magnitude are K_{inline} vs. frequency. Each one of the FFT processes reads either a single chunk or multiple chunks, forming a larger chunk and significantly reducing disc I/O.

orchestrated using libraries like Dask and MDIO. Nonetheless, it is crucial to recognise that despite the optimised access patterns, the I/O overhead might still be significant in specific workflows. Retaining the entire array or portions in memory could be more practical in such cases. For these situations, an in-memory variant of compressed MDIO could prove beneficial.

Cloud object stores typically support large numbers of concurrent read and write operations on objects without performance degradation but can suffer a higher latency than other file storage solutions. The Zarr format naturally supports a high level of concurrency on read and write, optimising its performance on object stores and potentially on a network file system. It is worth noting that performance can be limited by cloud service providers when thousands of machines access the same object. Arranging data in multiple chunks (preferably with randomised prefixes) further enhances parallelism (Google Cloud, 2023).

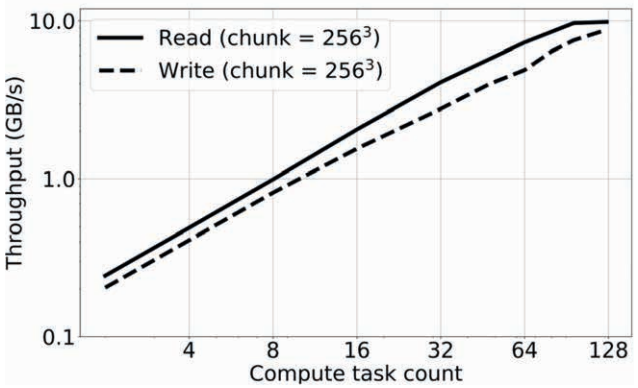

Figure 5 MDIO read/write performance as a function of the number of compute tasks.

Figure 5 shows the throughput scaling of MDIO read and write of a 3D volume as a function of compute tasks for MDIO seismic data on Google Cloud Storage (GCS). The MDIO volume is read using Dask workers deployed on a Kubernetes cluster. Throughput is calculated by measuring the time to read the MDIO volume by scaling up the number of workers. Each worker contains a thread pool with two threads and 4 GB of memory.

Manipulation of seismic data for imaging, processing, and visualisation requires the platform to efficiently support a set of tensor operations on the data. These might include reading slices (e.g., inline, crossline, and depth slices for visualisation), header manipulation, and others. By decomposing the data into chunks, I/O using the MDIO can be significantly more performant than other popular alternatives, such as SEG-Y and derivatives. For example, a header manipulation or indexing operation on an entire SEG-Y data would require at least one read for every trace in the dataset. Accessing data from an object store might require many requests (one per trace) or some transmission of trace data irrelevant to the task. By contrast, MDIO would require only one or relatively few reads to the chunked headers, which is more efficient in almost all cases.

Another example might be a read to a slice or arbitrary line through a seismic survey. At the same time, this can be efficient for formats like SEG-Y in the sort order of the data, but it becomes increasingly inefficient to slice data orthogonally to this. With MDIO, the shape of the chunks can be designed to provide consistent performance for different access patterns.

Most of the existing commercial seismic imaging and processing software is written in C, C++, and Fortran. Libraries like Tensorstore (Maitin-Shepard & Leavitt, 2022) provide C++ implementations of the Zarr protocol. MDIO is also designed to work seamlessly with popular scientific libraries and frameworks, such as NumPy and Dask. An imaging application employing Dask and Python with Zarr for 3D Marchenko imaging is discussed (Ravasi & Vasconcelos, 2021). The authors found that the features of the chunking, compression, multi-threading, and multi-processing read/write concurrency supported by Zarr were advantageous. The authors also demonstrate that the integration of Zarr with Dask and Kubernetes allowed them to create a fault-tolerant application that they could deploy to the cloud, realising potential cost savings of using 'spot-pricing' for their computing.

Machine learning

The MDIO Python API is compatible with libraries like NumPy, TensorFlow, and PyTorch, commonly used for the training and inference of machine and deep-learning models. A data-loading pipeline is critical for training deep-learning models with seismic data. Due to the seismic dataset's size, training on lines or patches is preferred to alleviate memory constraints. By using MDIO, patches can be extracted during training. This contrasts with an expensive data preprocessing stage where patches are extracted from seismic data and stored as JPG/PNG images. Because of this, there is very little additional overhead in creating a data pipeline for training compared to working natively with either NumPy data or collections of images. MDIO also carries a set of global summary statistics that can be used to standardise inputs to a machine-learning model, which is an important step in most ML training pipelines.

An example application of where MDIO was used to generate patches in the deep learning model for salt interpretation is shown in Figure 6. Utilising nine datasets from the eastern Gulf of Mexicio (GoM), a nine-fold cross-validation technique was applied to measure the generalisation performance of the ML model. This method involves using one dataset as the test set and the remaining eight for training and repeating the process for all subsets. We applied an ensemble of the nine models to predict salt on a new unseen survey in Green Canyon. The study aimed to illustrate how salt variability and morphology in the GoM can impact the ML algorithm's ability to predict salt bodies on unseen data.

Real-time visualisation

As discussed in seismic imaging and data management applications, MDIO enables real-time visualisation in web and desktop applications. Data can be streamed directly from the cloud to applications running on workstations, laptops, or mobile phones. For this use case, users typically want to be able to select and view 2D slices through large multi-dimensional seismic datasets. Common strategies for making this process performant in all dimensions include creating transposed copies of the underlying data and employing concurrent multi-threaded or multi-process reads. Data compression is vital to minimise storage costs and reduce bottlenecks in streaming data to a client application. Zarr naturally supports all these requirements (discussed by Ravasi and Vasconcelos, 2021). Integration with Dask can make concurrency more cost-effective in a cloud environment. Zarr chunking can also be configured to ensure that read performance is isotropic or optimised for a specific direction.

Minimising the transmitted data volume in all use cases is advantageous, particularly for visualisation where full precision

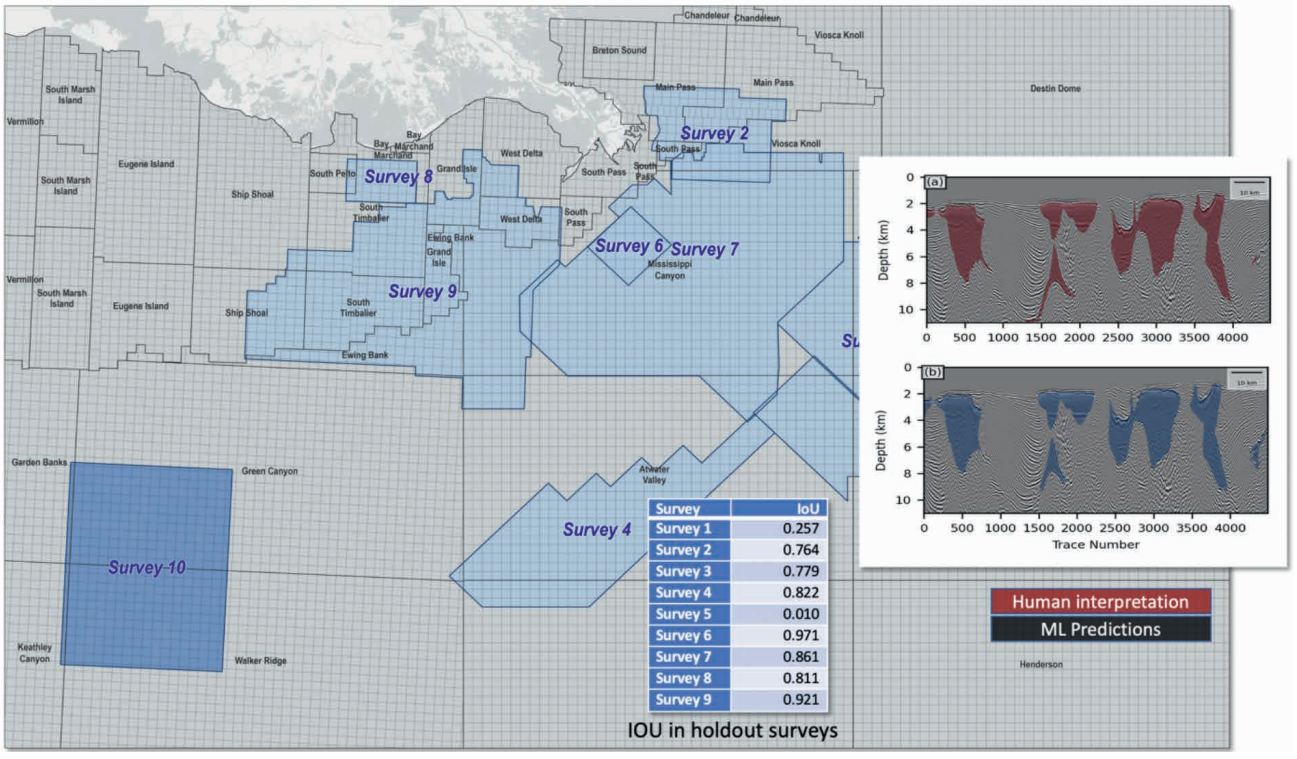

Figure 6 Training a deep-learning model for salt interpretation (from Warren et al., 2023).

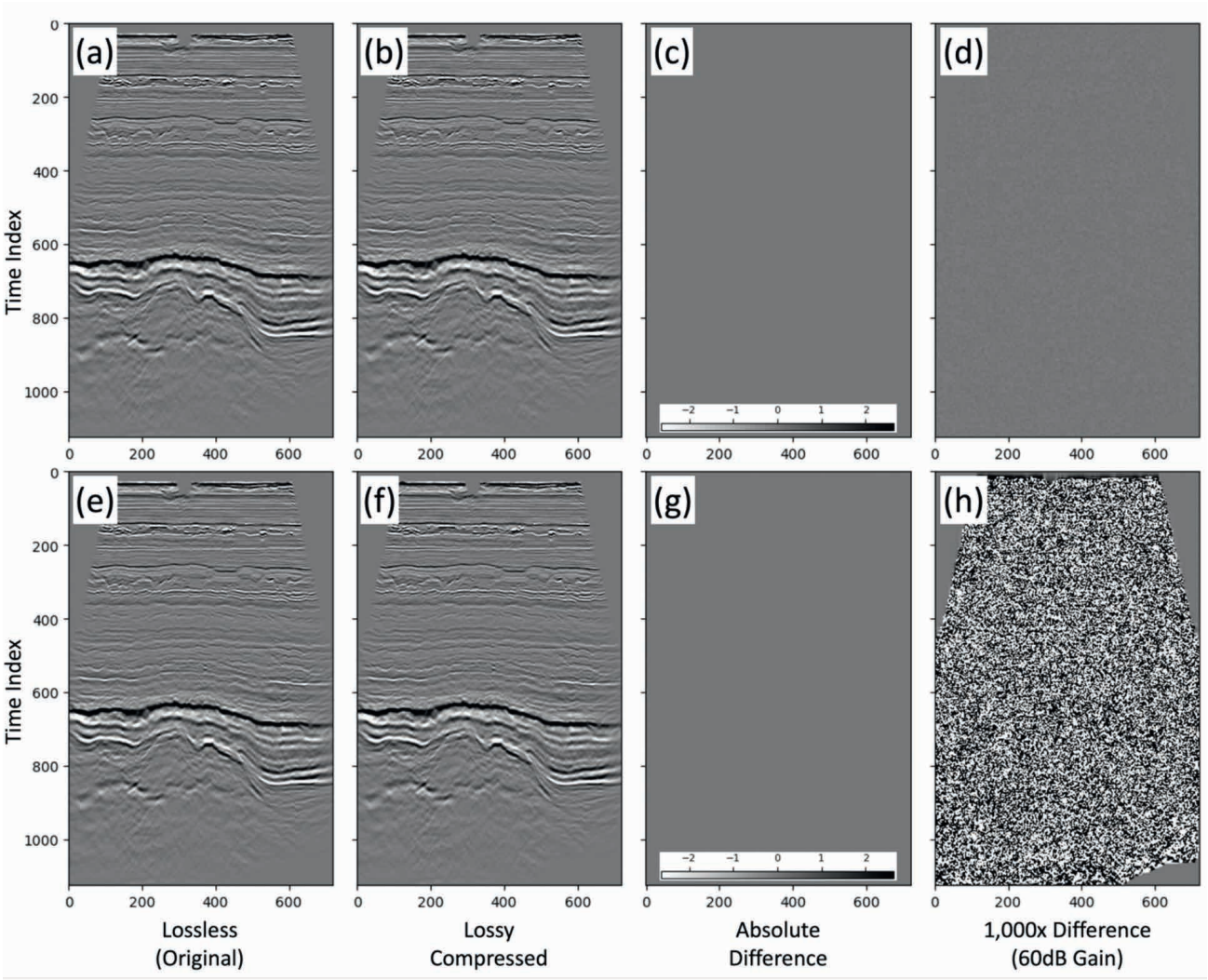

Figure 7 Images (a) and (e) show the original (float32) Volve angle stack dataset. Image (b) shows a lossy version with a 5:1 compression ratio. Image (c) shows the absolute difference between (a) and (b). Image (d) is the difference (c) gained by 1000x (60dB). Image (f) shows a 25:1 compressed version of (e). The images (g) and (h) show absolute and gained differences between (e) and (f).

float data is not generally needed. In this case, it is beneficial to use lossy compression. Figure 7 shows the result of ZFP compression on seismic data. The dataset shown is an angle-stack of the Volve survey (courtesy of data owners, published under CC By 4.0 license. Details at: https://discovervolve.com). The 5:1 compressed data is perceptually lossless, and the 60dB gained difference shows no signal leakage. The 25:1 compressed data show a minimal absolute difference, and the gained difference shows minor leakage. The data quality can be tuned to the downstream application needs and available bandwidth.

Conclusions

MDIO is a powerful and flexible tool that allows users to easily manage multi-dimensional energy data. It offers standardised storage and processing capabilities and is built as an extension of the Zarr protocol. With MDIO, data sharing, collaboration, and analysis are simplified, making it an ideal solution for researchers, practitioners, and developers in the energy sector. Optimised default chunk dimensions for various seismic and wind data types are available, addressing the challenges of data management and cloud object storage. MDIO is available under the Apache 2.0 licence and can be easily installed through popular channels such as Conda and Pip.

Acknowledgments

We thank Sathiya Namasivayam, Charles Nguyen, Mohammad Nuruzzaman, Cable Warren, and the rest of the TGS Data and Analytics team for their valuable discussions and support. We also appreciate TGS for permitting us to publish this material. Additionally, we acknowledge the organisations and companies that generously provide public datasets and open-source tools for the community's benefit.

References

Abma, R. and Kabir, N. [2006]. "3D interpolation of irregular data with a POCS algorithm," *Geophysics, **71***, E91-E97. Sansal, A., Kainkaryam, S., Lasscock, B. and Valenciano, A. [2023]. "MDIO: Open-source format for multidimensional energy data," *The Leading Edge,* **42**, 465-473.

Barré, M., Walley, S., Savajol, V., Deng, J., Rhodes, C., van Welden, A., Endresen, P., Storm-Olsen, M. and James, A. [2022]. *Implementation of cloud-ready technology designed to address longstanding data inter-*

operability and accessibility issues through the use of a comprehensive semi-automated seismic interpretation workflow. Paper presented at the SEG/AAPG International Meeting for Applied Geoscience and Energy, Houston, Texas, USA. https://doi.org/10.1190/image2022-3743148.1.

Dask Development Team [2016]. Dask: Library for dynamic task scheduling. Retrieved from https://dask.org.

Equinor [2023]. "segyio: Fast Python library for SEGY files." March 28, 2023. https://github.com/equinor/segyio.

Google Cloud [2023]. "Request rate and access distribution guidelines", accessed March 21, 2023. https://cloud.google.com/storage/docs/request-rate.

The HDF Group [1997-2023]. Hierarchical Data Format, version 5, 1997-2023. https://www.hdfgroup.org/HDF5.

Miles, A., jakirkham, Bussonnier, M., Moore, J., Papadopoulos Orfanos, D., Fulton, A., Bourbeau, J., Lee, G., Patel, Z., Bennett, D., Rocklin, M., AWA BRANDON AWA, Chopra, S., Abernathey, R., Sales de Andrade, E., Kristensen, M. R. B., Durant, M., Schut, V., Dussin, R. and Bell, R. [2023]. zarr-developers/zarr-python: v2.14.1 (v2.14.1). Zenodo. https://doi.org/10.5281/zenodo.7633154.

Lindstrom, P. [2014]."Fixed-rate compressed floating-point arrays." *IEEE Transactions on Visualization and Computer Graphics,* **20**(12), 2674-2683.

Maitin-Shepard, J. and Leavitt, L. [2022]. *TensorStore for High-Performance, Scalable Array Storage*. Retrieved from https://ai.googleblog.com/2022/09/tensorstore-for-high-performance.html.

Matteo Ravasi and Ivan Vasconcelos [2021]. An open-source framework for the implementation of large-scale integral operators with flexible, modern high-performance computing solutions: Enabling 3D Marchenko imaging by least-squares inversion, *Geophysics,* **86**, WC177-WC194.

OSDU [2023]. Seismic DDMS OpenZGY Gitlab page, https://community.opengroup.org/osdu/platform/domain-data-mgmt-services/seismic/open-zgy, accessed March 21, 2023.

Rocklin, M. [2015]. Dask: Parallel Computation with Blocked algorithms and Task Scheduling. *Proceedings of the 14th Python in Science Conference*.

Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Liu, Z., Berner, J., Wang, W., Powers, J. G., Duda, M. G., Barker, D. M. and Huang, X.-Y. [2019]. *A Description of the Advanced Research WRF Version 4*. NCAR Tech. Note NCAR/TN-556+STR, 145 pp. https://doi.org/10.5065/1dfh-6p97.

Unidata [2021]. NetCDF, version 4.8.1. Unidata, https://doi.org/10.5065/D6H70CW6.

Warren, C., Kainkaryam, S., Lasscock, B., Sansal, A., Govindarajan, S. and Valenciano, A. [2023]. "Toward generalized models for machine-learning-assisted salt interpretation in the Gulf of Mexico," *The Leading Edge* **42**, 390-396.

Yann, C. "lz4." lz4. github. io/lz4 [2011].

Yann, C. and Kucherawy, M. [2018]. Zstandard Compression and the application/zstd Media Type. No. rfc8478. 2018.

EAGE
EUROPEAN ASSOCIATION OF GEOSCIENTISTS & ENGINEERS

PETROLEUM GEOSTATISTICS 2023

TOWARDS A SUSTAINABLE ERA OF GEOENERGY

27-30 NOVEMBER 2023 - PORTO | PORTUGAL

Focus on new methods and applications in
Geostatistics for the Petroleum Industry

- Geo-modelling and Geostatistics
- Geophysics and Geomechanics
- Inverse Dynamic Modelling and UQ
- Data Science and AI
- Geo-Energy Case Studies

REGISTER TODAY AND SAVE!

VISIT **PETROLEUMGEOSTATISTICS2023.ORG**