MDIO v1: schematizing seismic data for AI and processing

Brian Michell, Altay Sansal, Ben Lasscock, Mark Roberts*
TGS

Summary

The MDIO format is an open-source storage solution for seismic data, designed for processing, AI training and inference, data management, and visualization. Built on top of the open-source projects Zarr, Blosc, Xarray, and TensorStore, it enables efficient access to large datasets on both local and cloud storage, reducing costs and enhancing accessibility. MDIO v1 introduces a structured dataset model that includes Metadata, Variables, Dimensions, and Coordinates, simplifying the organization and use of seismic data. It is interoperable with Python's Xarray format to ensure compatibility with scientific workflows and natively supports Dask for parallel processing. A native C++ implementation based on Google's TensorStore library offers options for integrating MDIO with existing processing systems.

We present case studies highlighting the benefits of MDIO in seismic workflows, such as rapid SEG-Y ingestion, cloud-based I/O with an example from the AWS Open Data Registry, and the use of structured metadata for pre-stack and post-stack visualization. MDIO enhances efficiency in large-scale geophysical analysis and AI applications by clearly defining the relationships in seismic data.

Introduction

The open-source MDIO format (Sansal et al., 2022) has been widely utilized to represent seismic data in digital workflows, including artificial intelligence (AI) model training, data management, and data visualization on web applications. The MDIO format is inherently designed to be completely open source, building on the cloud-native storage format Zarr (Miles 2023) and utilizing compression codecs from Blosc (Haenel, 2014). This design achieves significant cost savings for storing large seismic data libraries in the cloud. MDIO data is accessible and self-describing, enabling efficient access from both local disk and cloud storage. Previous applications of MDIO include serving pre-stack seismic data for training AI denoising models on a global dataset scale (Brusova et al., 2021; Valenciano et al., 2022), conducting salt interpretation at a basin scale (Warren et al., 2023), and providing wind resource assessment data with analytics to web applications (Sansal et al., 2023). More recently, MDIO has been crucial in training large 3D seismic foundation models (Lasscock et al. 2024, Sansal et al. 2025a, Sansal et al. 2025b), where efficient access to data from a global seismic data library has been essential.

This study illustrates how MDIO v1 enhances the conceptualization of seismic data diversity, including pre-stack data. With its native C++ implementation (TGSAI, 2025), MDIO v1 provides better integration with established seismic processing frameworks such as Madagascar (Fomel et al., 2007), further broadening its impact across the industry.

MDIO v1: A Technical Overview

The newly released MDIO v1 revision is designed to enhance the contextualization of seismic data and improve support for visualization and seismic processing. The core concept within MDIO v1 is the Dataset, which represents a collection of related Variables and additional Metadata such as coordinate reference systems (CRS). Each Variable is an array of data with clearly defined Dimensions, optional Coordinates, and additional Metadata—such as histograms and units. Coordinates contain array data that provide further context for the Variables, for example, by supplying latitude and longitude or cdp-x and cdp-y. A template for an MDIO Dataset (including its Variables, Dimensions, and Coordinates) is outlined by a JSON schema. This approach enables users to customize data structures to their needs, creating self-describing data for specialized applications. Moreover, the semantics of the MDIO Dataset are designed to interoperate seamlessly with Xarray, a Python library widely used in earth sciences for constructing self-descriptive datasets (Hoyer et al. 2022, Hoyer et al. 2025). As a demonstration of Xarray and MDIO's capability to manage data on a global scale, Figure 1 presents a snapshot of global ERA5 model wind speeds (Hersbach et al. 2020), which are part of the 6 Petabyte ERA5 open climate dataset.

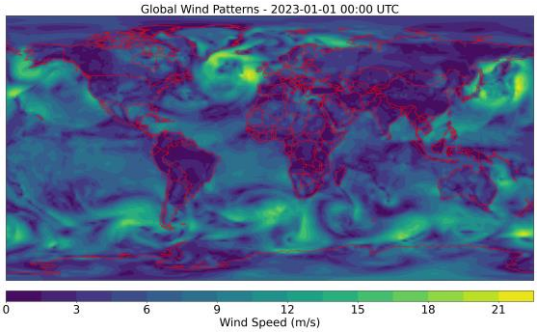

Figure 1: Global wind speed at an instant in time, accessed from the ERA-5 climate dataset hosted by Google Cloud marketplace.

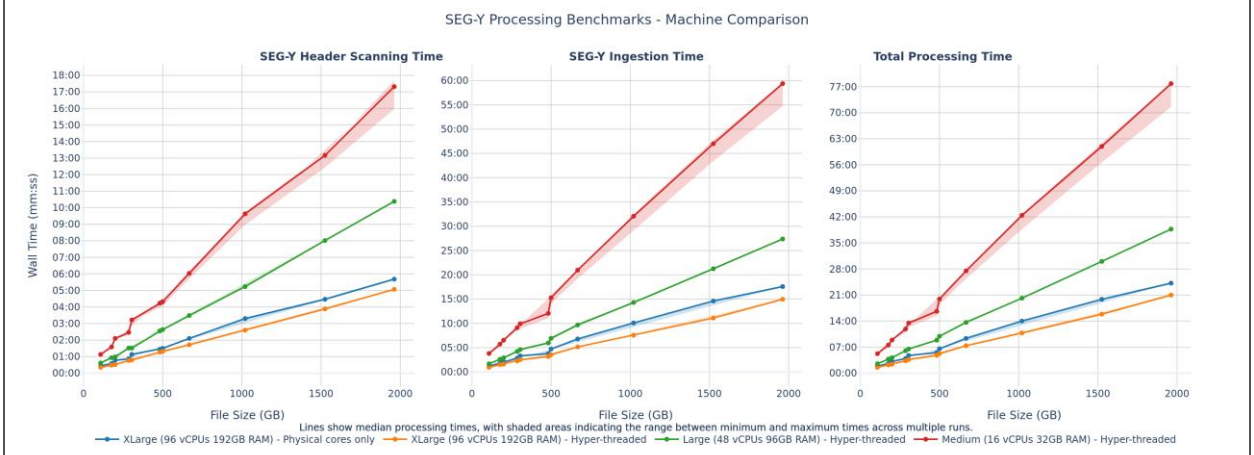

Figure 2: Benchmarks demonstrating cloud-to-cloud SEG-Y to MDIO format conversion, where both source and destination exist only in the cloud. Solid line shows mean wall time as a function of SEG-Y file size and shaded region shows total time distribution. (left) The time required to scan the SEG-Y headers, (center) the ingestion time, (right) the overall ingestion time.

For Python users, Xarray interoperability with MDIO v1 files offer built-in support for Dask (Rocklin 2015), enabling the parallel and distributed computing necessary to scale seismic processing and related applications. An open-source native C++ implementation of MDIO, built on Google's TensorStore library (TensorStore 2022, TensorStore 2025), is also available. TensorStore provides scalable multi-dimensional array storage, support for open-source compression methods (Haenel 2014), and native cloud I/O capabilities. By efficiently managing multi-dimensional data, TensorStore frees developers using MDIO from manually handling thread pools, caching, compression, and cloud-native storage, allowing them to focus on domain-specific data processing instead (Google Research 2022). Through TensorStore, MDIO v1 offers native C++ support for object storage on all major cloud providers and local file systems. Furthermore, the C++ implementation of MDIO can be integrated with existing seismic processing frameworks, such as Madagascar (Fomel et al. 2007).

From a data management perspective, MDIO provides considerable cost savings for cloud storage compared to SEG-Y. MDIO has shown an average of 31.5% lossless compression at large scales relative to SEG-Y using Blosc with Zstd codec, as demonstrated by 15 PB of seismic data stored in Google Cloud. Another significant advantage of chunked cloud storage is that each chunk is treated as an individual file in a bucket. Since most files are stored in low-cost archive storage, only the necessary chunks need to be retrieved and transferred to higher-tier storage when data or metadata are accessed—thus achieving considerable cost savings over time.

Finally, SEG-Y is the standard for exchanging seismic data in the seismic industry. Tools are available for efficiently exporting SEG-Y data to MDIO in the cloud, hybrid cloud, or on-premises. Figure 2 shows the ingestion time for converting a SEG-Y file to MDIO format, with both files stored in a cloud object store, demonstrating that ingesting to MDIO is scalable on standard cloud hardware.

Figure 3: The MDIO v1 schema associated with the Poseidon, full-stack seismic dataset.

Case Study: MDIO on AWS Open Data Registry

The Poseidon 3D Seismic Dataset, made available courtesy of ConocoPhillips and Geoscience Australia, has been published on the AWS Open Data Registry in MDIO v1 format. Figure 3 shows the dataset schema for the full-stack volume of the Poseidon dataset. An MDIO Dataset includes

Metadata, Variables, Dimensions, and Coordinates. In this case, the dataset presents seismic data and headers as Variables, while the inline, crossline, cdp-x, cdp-y, time, and trace mask (labeling live traces) function as Coordinates.

MDIO utilizes a chunked data format; in this scenario, the seismic variable is stored in isotropic 128^3 chunks (8 MB each), a chunk size optimized for both the storage medium and expected data access patterns. Due to the isotropic chunking, read times remain consistent across the inline, crossline, and time dimensions relative to their shape. Additionally, by separating coordinates from the seismic headers, tick labels can be accessed more efficiently, simplifying visualization tasks (see Figure 4).

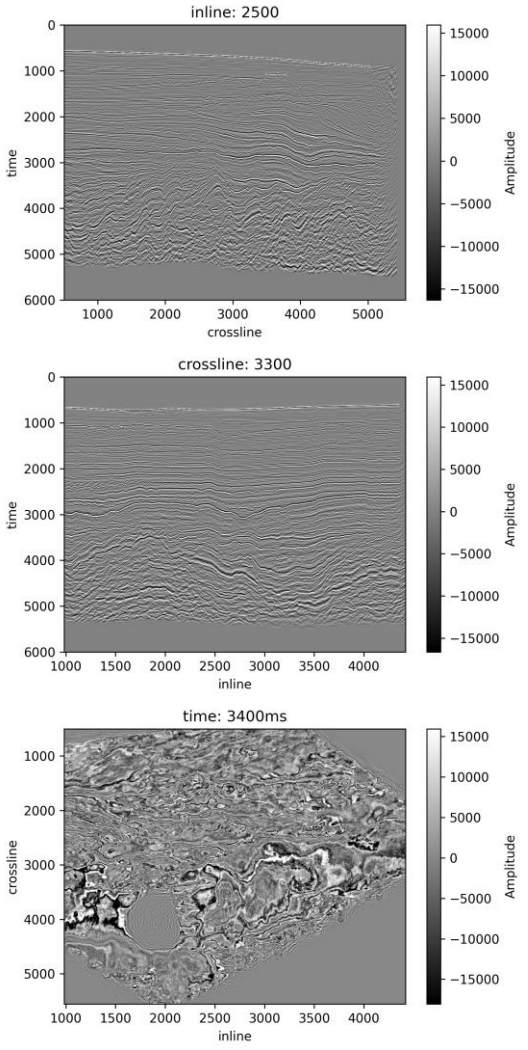

Figure 4: Shows inline, crossline and time slices extracted from the Poseidon Dataset hosted on AWS Open Data. Tick-labels read from Coordinates.

To demonstrate concurrency within the MDIO C++ library, we benchmarked the performance of reading an uncompressed 3D MDIO Variable from Google Cloud Storage to various virtual machines (VMs) in the same region. Figure 5 illustrates the throughput (in gigabytes per second: GB/s) as a function of the number of physical cores on each VM, highlighting the library's ability to scale performance by utilizing additional threads. In this benchmark, we observed that as the VMs' available memory exceeded the Variable's size, the throughput experienced diminishing returns.

The Poseidon use case highlights the significant advantages of the MDIO v1 format in performance, storage efficiency, and data accessibility. Because the MDIO chunked data format is flexible, it can adapt to support diverse access patterns—a feature particularly relevant for building seismic foundation models, where rapid access to 3D data subsets is essential (Lasscock et al. 2024, Sansal et al. 2025a, Sansal et al. 2025b). For optimal use in cloud storage, chunk sizes of approximately 8–10 MB are recommended (Sansal et al., 2022). In the Poseidon example, coordinates make the dataset self-describing and eliminate the need for non-standard headers, thereby enhancing interoperability. The MDIO Poseidon data is openly available and can be accessed via Xarray in Python or using the MDIO C++ library.

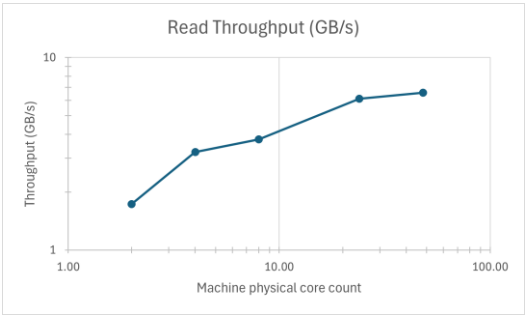

Figure 5: We benchmarked a small test volume reading from the cloud for machines with an increasing number of vCPUs and memory.

Case Study: MDIO Pre-Stack Seismic Data

Leveraging MDIO coordinates and metadata associated with seismic datasets enables more advanced workflows. Figure 6 illustrates an MDIO schema for a 3D streamer dataset, where coordinates are stored as auxiliary arrays in an object store. By employing templated MDIO schemas, each seismic data type standardizes its available metadata and provides rapid access to this information, thus ensuring

reproducible dataset descriptions and facilitating batch processing. In this example, the 3D streamer dataset's coordinates include shot and receiver X/Y locations, and gun numbers for shot gathers.

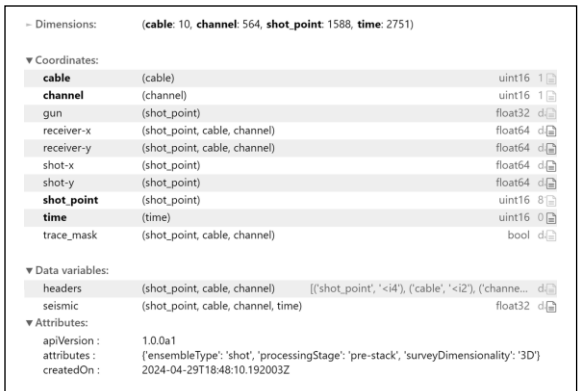

Figure 6: The MDIO v1 schema associated with a 3D streamer dataset.

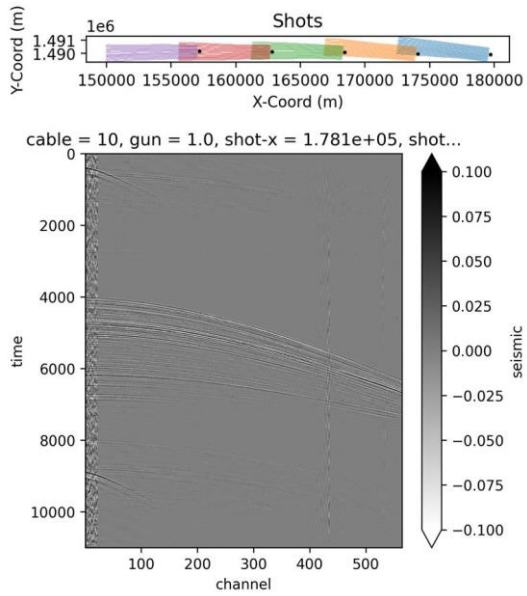

Figure 7: (Top): A Decimated streamer and shot geometry for a 3D offshore seismic dataset derived from MDIO coordinates. Bottom: A gather sliced from an MDIO dataset, showing gather for a specific cable shot, x/y combination.

By utilizing readily accessible Coordinates, users can efficiently filter data—for example, by selecting a unique *{gun, cable, channel}* combinations to produce common-channel gathers, or by *{shot_point}* to generate 3D shot gathers. These filtering capabilities and on-demand access to geographic coordinates (shot-x, shot-y, etc.) also allow for the windowing of data (needed for processes such as SRME and 5D-regularization) and for visualization and QC of the acquisition geometry without needing external information. Additionally, the filtering capabilities enable an in-memory merging of data volumes and vintages, enhancing advanced processing workflows and analytics while avoiding issues such as data duplication.

Figure 7 illustrates two data products derived from the MDIO Dataset: (top) a decimated view of streamer acquisition geometry in real-world coordinates, with shots color-coded; and (bottom) a gather for a specific cable and a receiver obtained by slicing the data using the available Coordinates *{cable, channel}*.

Conclusions

MDIO v1 offers an open, scalable, and efficient framework for managing seismic data across AI training, data management, and processing workflows. By building on established libraries such as Zarr, Blosc, Xarray, and TensorStore, MDIO supports seamless access to large-scale datasets in both local and cloud environments. Its structured design—encompassing Metadata, Variables, Dimensions, and Coordinates—facilitates interoperability with Python-based systems and parallel computing through Dask (Rocklin 2015), while the native C++ implementation effectively integrates with existing C/C++ seismic processing platforms.

Case studies, including large-scale SEG-Y ingestion and the Poseidon 3D seismic dataset, demonstrate MDIO's potential for efficient cloud-based I/O, substantial cost savings through compression, and reliable data filtering and visualization. By decoupling seismic data from header metadata and adopting chunked storage strategies, MDIO streamlines retrieval for both pre-stack and post-stack analysis, enabling reproducible and high-performance workflows that address traditional seismic exploration needs as well as AI-driven applications.