2025 European HDF5 Users Group Meeting

Monday 26 May 2025 - Wednesday 28 May 2025

Book of Abstracts

Contents

1

Enhancing HDF5 with Multi-Threading, Sparse Data Storage, and Encryption: Preliminary Results and Demonstrations

Author: Elena Pourmal[1]

Co-author: John Mainzer [1]

[1] *Lifeboat, LLC*

Corresponding Authors: epourmal@gmail.com, john.mainzer@lifeboat.llc

Over the past few years, Lifeboat LLC has been focused on advancing the capabilities of HDF5 by incorporating multi-threaded support, enhancing the storage of sparse and variable-length data, and implementing robust encryption mechanisms for data stored within HDF5 files. These improvements are aimed at optimizing performance, increasing flexibility, and strengthening data security.
In our presentation, we will share the preliminary results of our work on multi-threading and sparse data storage. Specifically, we will showcase the progress made in enabling efficient multi-threaded I/O operations and managing sparse datasets more effectively. Additionally, we will demonstrate the newly introduced encryption feature for securing sensitive data within HDF5, ensuring both confidentiality and integrity.
We look forward to engaging with the community on these developments and receiving valuable feedback to further refine these features.

Key words: HDF5 multi-threading, sparse data, variable-size data, data encryption

May we record your session?:

Yes

2

Writing HDF5 in Pure Java: Practical Guide

Author: Igor Khokhriakov[1]

[1] *FS-SC (Scientific computing)*

Corresponding Author: igor.khokhriakov@desy.de

This contribution presents our experience using a pure Java implementation of the HDF5 file format to support metadata collection at the P05 beamline at Hereon, DESY. Since 2016, we have relied on Java-based solutions to generate HDF5 files for hundreds of experiments with minimal maintenance overhead.

We will provide a practical overview of how to use the Java HDF5 library effectively in the context of beamline operations, covering:
–Creating and organizing datasets and groups
–Handling compound datatypes and attributes
–Performance considerations and memory management
–Integration with control systems such as Tango and TINE

This talk is intended for developers and beamline scientists interested in building robust data workflows using Java, without relying on native code bindings. The approach also supports long-term maintainability and ease of integration into existing Java-based infrastructures.

May we record your session?:

Yes

3

Innovative Data Acquisition Solutions with HSDS at MAX IV Synchrotron

Authors: Felix Engelmann[1]; Zdenek Matej[2]

Co-authors: Jason Brudvik [3]; Michele Cascella [2]

[1] *Institute for Cybersecurity & Digital Trust (ICDT), The Ohio State University*

[2] *MAX IV Laboratory, Lund University*

[3] *European Spallation Source*

Corresponding Authors: fe-research@nlogn.org, michele.cascella@maxiv.lu.se, jason.brudvik@ess.eu, zdenek.matej@maxiv.lu.se

MAX IV is an accelerator-based light source located in southern Sweden. It continuously operates 16 experimental stations using X-rays for material and life sciences. User data analysis primarily occurs on a small edge HPC, while automatic scientific data processing, from around 100 data sources, runs in an edge-cloud environment. These pipelines provide rapid feedback on data acquisition and analysis to control workstations and users' laptops at the facility or accessed remotely. The vast majority of raw experimental data is stored in the HDF5 format.

Data analysis and visualization at synchrotron light sources is inherently a distributed computing problem (HPC, edge, users'terminals). Traditionally, we have used distributed file systems together with HDF5, which motivated the development of VDS and SWMR. The HDF group introduced HSDS over half a decade ago, and POSIX HSDS storage was implemented shortly after.

This allowed us at MAX IV to start HSDS microservices for our experimental stations, even without object storage. We routinely use HSDS to store intermediate data for our distributed data reduction pipelines, particularly for streaming tomography. This offers several advantages, including no dependencies on distributed file systems and straightforward data access from users'terminals. We can also utilize the existing HDF5 tools, such as the silx viewer and other standard software like h5pyd.

In many cases, the intermediate HDF5 data do not require persistent storage. Therefore, we developed an extension for dranspose –the distributed data analysis pipelines framework, which allows in-memory Python dictionary data to be exposed with the HSDS/h5pyd interface.

Keywords: data acquisition, distributed applications, hybrid environments, memory resident datasets, HSDS, synchrotron, tomography

May we record your session?:

Yes

4

Accelerating Data Compression in HDF5 through Parallel Filter Processing

Author: Frederick Neu[None]

Corresponding Author: frederick.neu@studium.uni-hamburg.de

Modern science and engineering creates and accumulates huge amounts of data which is persisted through tools like HDF5 in order to be available for further analysis, display and many other operations. Increasing efficiency in this data processing is critical for nowadays growing data quantities, not only for saving time, but also to efficiently use available resources.
This thesis aimed to provide a working prototype and analysis on parallel application of data filters within the HDF5 environment with special emphasis on HDF5 registered filters, such as LZ4 compression.
This prototype is embedded into the HDF5 framework, can be freely accessed as any other provided function and is available ready to use after building the project, while maintaining standard library behavior for all other use cases. Generally implementation is based on an all purpose thread pool with variable functionality based on registered callback function and can therefore be used in later versions of development. Analysis is based on comparison of standard library performance and prototype, based on identical example datasets, as well as statical analysis of provided program code. Furthermore CPU utilization and I/O performance are evaluated.
Results suggest a great potential for a fully implemented design including most capabilities of the stock HDF5 library. Near full CPU utilisation is shown with little to no wait for I/O completion and therefore cutting down runtime by quite an extensive amount.
This prototype, related work and analysis show the great improvement possible by adjusting the existing framework to a multithreaded solution while still maintaining full standard behaviour.

May we record your session?:

Yes

5

High Performance Storage for HPC: Proven HDF5 Scalability and Consistency on Pure Storage FlashBlade

Author: Michael Kaspars[1]

Co-author: Bikash Roy Choudhury [1]

[1] *Pure Storage*

Corresponding Authors: broychoudhury@purestorage.com, mkaspers@purestorage.com

The integration of the HDF5 high-level IO library, encompassing Single Writer Multiple Reader (SWMR) and parallel mode (pHDF5), with Pure Storage FlashBlade using NFS, modern Linux Kernels, and Networks, offers a robust and high-performance solution. This combination significantly reduces Total Cost of Ownership (TCO) and technical debt compared to traditional parallel file systems.

HDF5, particularly in SWMR configuration, ensures data consistency during concurrent read and write operations. Pure Storage FlashBlade's NFS provides a high-performance storage solution, ideal for High-Performance Computing (HPC) environments, facilitating efficient access and modification of large datasets. Enhancements within HDF5, such as superblocks, further bolster data consistency. FlashBlade's POSIX-compliant data access guarantees atomicity during read and write operations to direct flash memory modules, a crucial requirement for SWMR operations to prevent inconsistencies.

Pure Storage FlashBlade delivers exceptional throughput, low latency, and supports GPU Direct Storage with NFS over RDMA, essential for HPC. Multi-iteration write and read tests have demonstrated robust throughput and consistency, validating FlashBlade's capability to meet HDF5 demands in both SWMR and parallel configurations. Parallel HDF5 leverages MPI-IO, which supports NFS backends, treating it as a unified file system. This integration ensures parallel processes operate

efficiently on shared datasets, with MPI layer compatibility enabling effective data handling across multiple nodes.

With optimized HDF5 settings, SWMR mode operates without exclusive or shared locks, allowing a single writer to handle write operations while multiple readers access data concurrently without conflicts. Comprehensive tests have validated FlashBlade's ability to run HDF5 in both SWMR and parallel modes, ensuring atomic writes and reads. Performance benchmarking using HDF5-specific tools has confirmed high throughput and data consistency across multiple nodes via NFS.

In conclusion, the synergy of HDF5's robust data handling with the high performance and consistency of Pure Storage FlashBlade using NFS creates a powerful solution for managing large, complex datasets in HPC environments.

May we record your session?:

Yes

6

Seamless Integration of Blosc2 and HDF5 for High-Performance Data Compression

Author: Francesc Alted[1]

[1] *Blosc project*

Corresponding Author: francesc@blosc.org

HDF5 is the de facto standard for storing large volumes of binary data in files. Blosc2, an award-winning high-performance library, excels at compressing binary data in memory. Both are widely used, making their integration natural. This talk will cover using Blosc2 as an HDF5 filter and HDF5 as a Blosc2 backend.

We will outline the current state of the Blosc2 plugin for HDF5 (https://github.com/Blosc/HDF5-Blosc2) and provide instructions on its usage. Additionally, we will introduce the b2h5py package (https://github.com/Blosc/b2h5py), which bypasses the slow HDF5 filter pipeline to achieve high performance. Sparse datasets will be used to demonstrate Blosc2's performance on HDF5.

Finally, we will discuss various codecs available in Blosc2, with a focus on the Grok codec (https://github.com/GrokImag which efficiently compresses data in the JPEG2000 format. We will also touch on the enhancements made to BTune (https://ironarray.io/btune), a Blosc2 plugin, to support lossy compression and automatically select the best codec/filter.

This work has been carried out as part of the LEAPS-INNOV program (https://leaps-innov.eu/), which strives to build a European ecosystem for photon sciences. The integration of Blosc2 and HDF5 ensures efficient storage and retrieval of large datasets, a critical factor for the program's success.

Compression #HighPerformance #HDF5 #Blosc2 #JPEG200 #LEAPS-INNOV

May we record your session?:

Yes

7

The keys to the management of time-evolution signals on HDF5 files in ITER

Author: Rodrigo Castro[1]

Co-authors: Yury Makushok [2]; Lana Abadie [3]

[1] *CIEMAT*

[2] *INDRA*

[3] *ITER*

Corresponding Author: rodrigo.castro@ciemat.es

ITER (International Thermonuclear Experimental Reactor) is the largest international experiment in the field of generating nuclear fusion energy by magnetic confinement. ITER's objective is to operate in modes that come as close as possible to the conditions of a commercial fusion reactor, which implies long pulses and systems running continuously.
From the point of view of the data acquisition and control system, ITER, on the one hand, requires an enormous number of signals (due to its size and experimental nature), many of them with a very high sampling rate (due to the highly chaotic nature of the plasma under extreme conditions), and, on the other hand, it requires a data storage and management system compatible with long pulse and continuous data acquisition. These requirements have driven the design and implementation of ITER's data storage and management system, where time evolving signals and their real time processing have absorbed the great majority of development effort.
From the very beginning, ITER chose HDF5. This presentation describes the technical factors that have proved decisive in using HDF5 as the core technology for storing and managing all of ITER' s time evolving signals, as well as for the real time handling of their data. It details the different challenges that have been addressed throughout the project, together with a detailed description of the solutions that have been adopted at the design and implementation level.

May we record your session?:

Yes

8

Synergizing Science and Data: IOwarp for Scalable Workflow Management

Author: Scot Breitenfeld[1]

Co-authors: Anthony Kougkas [2]; Gerd Heber [1]; Jacob Hochhalter [3]; Vivek Srikumar [3]; Xian-He Sun [2]

[1] *The HDF Group*

[2] *Illinois Institute of Technology*

[3] *University of Utah*

Corresponding Author: brtnfld@hdfgroup.org

Scientific workflows are evolving rapidly, demanding the seamless integration of simulation, experiments, analytics, and AI. This evolution is placing immense pressure on traditional data management systems. To address these challenges, we present IOwarp, a new initiative focused on building a comprehensive data management platform. IOwarp aims to streamline complex scientific workflows by providing scalable and efficient data access, movement, transformation, and sharing capabilities. We will discuss the core architecture of IOwarp and will highlight the complementary roles of IOwarp in modern scientific data management. We welcome feedback and collaboration to further develop and refine these tools for the scientific community.

May we record your session?:

Yes

9

Using HDF5 to Store Scientific Data at the European XFEL

Author: Jose Vazquez-Garcia[1]

Co-authors: Alessandro Silenzi [1]; Dennis Goeries [1]; Djelloul Boukhelef [1]; Gero Flucke [1]; Krzysztof Wrona [1]; Luca Gelisio [1]; Sergey Esenov [1]; Steffen Hauf [1]

[1] *European XFEL*

Corresponding Authors: krzysztof.wrona@xfel.eu, luca.gelisio@xfel.eu, djelloul.boukhelef@xfel.eu, serguei.essenov@xfel.eu, steffen.hauf@xfel.eu, dennis.goeries@xfel.eu, gero.flucke@xfel.eu, jose.vazquez@xfel.eu, alessandro.silenzi@xfel.eu

The **European XFEL** is an X-ray laser research facility that produces extremely short and intense X-ray flashes, enabling investigations across a wide range of fields—from the structure of matter to the dynamic evolution of molecular systems. A typical experiment can generate petabytes of data within a day, originating from diverse detectors and in multiple formats. Managing this high-volume, heterogeneous data, along with metadata, in real- or near real-time poses unique challenges.

HDF5 provides a robust solution to these requirements. Its ability to define both data and its structure enables consistent storage across various data sources. HDF5 also supports parallel, real-time writing, by distributing experimental data across multiple files, and linking them coherently. Features such as sparse chunking and compression further optimize storage, which is critical given the facility's data output.

In this talk, I will introduce **karaboHDF5**, a library that enables real-time, space-efficient storage of experimental data, while ensuring the data is well-structured and easy to access for analysis and processing. Developed by the European XFEL on top of the HDF5 core library, it is tailored to the facility's demanding performance and usability requirements, and fully leverages the capabilities of HDF5.

Keywords: HDF5, real-time data acquisition, high-volume data, heterogeneous data, X-ray data acquisition.

May we record your session?:

Yes

10

HSDS v1.0 –Performance Features

Author: John Readey[1]

[1] *The HDF Group*

Corresponding Author: jreadey@hdfgroup.org

HSDS (Highly Scalable Data Service) is a REST-based service that provides read/write access to HDF5 data stores –using object storage or posix. By using a combination of multi-processing and asynchronous IO, HSDS can achieve remarkable performance when accessing very large datasets. On the other hand, performance lagged for clients invoking a series of smaller requests (reading or writing a small dataset selection, creating an attribute or a new link). As typical in client-server

architectures, the per-request latency is quite high compared to in-process operations (e.g. making a call to the HDF5 library). To address this, the next release of HSDS and h5pyd (the HSDS client library for Python) will look to improve performance by a combination of read-ahead logic and combining write operations into a single request. At the same time, clients can continue using the same h5py-api to achieve better performance without needing to make any code changes.

May we record your session?:

Yes

11

HDF: Powering the Future of Data - What's Now, What's New, What's Next!

Author: Scot Breitenfeld[1]

[1] *The HDF Group*

Corresponding Author: brtnfld@hdfgroup.org

This talk will provide an overview of the current state of the HDF5 software ecosystem, highlighting recent advancements, key components, and ongoing challenges. We will explore the future directions of the various tools and libraries that empower researchers and developers to manage HDF5-related workflows efficiently. Additionally, the talk will outline potential future initiatives for the HDF5 ecosystem to ensure its continued relevance and evolution in the ever-expanding realm of data-intensive applications.

May we record your session?:

Yes

12

An Open HDF5-Based Format for Industrial Inspection Data with Experimental Integration of the Onion VFD for File Versioning

Author: Baptiste Gauthier[1]

[1] *Evident Scientific*

Corresponding Author: baptiste.gauthier@evidentscientific.com

The NDE File Format (.nde), developed by Evident, is an open, extensible data format tailored for the non-destructive evaluation (NDE) and testing (NDT) industry. Built upon the HDF5 container and augmented with JSON-based metadata, it offers a platform-independent solution for storing inspection data, primarily for ultrasonic modality. By adopting an open format, .nde files can be accessed and analyzed without proprietary software, promoting integration with third-party tools and advanced analytics platforms.

To address the challenges of data versioning and traceability inherent in NDT workflows, we explore the application of the Onion Virtual File Driver (VFD), introduced in HDF5 version 1.13.2. The Onion VFD enables in-file revision management by layering modifications atop the original dataset,

allowing users to access and revert to previous versions without duplicating entire files. This approach ensures data integrity and auditability, essential for compliance with industry standards and regulations.

This presentation demonstrates how the integration of the .nde format with the Onion VFD addresses real-world data management challenges in industrial inspection environments. We discuss the technical implementation, benefits observed in data handling and version control, and potential implications for broader adoption in the NDT industry.

May we record your session?:

Yes

13

Shiver Me Timbers: The Design of Sharded Storage for HDF5

Author: Quincey Koziol[1]

[1] *NVIDIA*

Corresponding Author: qkoziol@nvidia.com

As object storage becomes even more prevalent, HDF5's underlying storage format needs to updated to match the interface that cloud and on-prem object systems provide. This talk will present a design overview of a mapping of the HDF5 data model onto S3-compatible storage systems. An outline of the planned VOL connector implementation and projected performance goals will be part of the talk.

May we record your session?:

Yes

14

Race Ya! An Accelerator-Native I/O Pipeline for HDF5

Author: Quincey Koziol[1]

[1] *NVIDIA*

Corresponding Author: qkoziol@nvidia.com

GPUs and similar accelerators have become the dominant compute platform for STEM applications, from finance to space flight, and beyond. However, HDF5 continues to execute exclusively on the Host CPU of GPU nodes. This talk will present a design overview of moving the I/O pipeline filters, datatype conversions and other transforms from the CPU to the GPU, including how to perform I/O directly from GPU memory. Implementation details of the revised transform pipeline will be included, along with projected performance benefits.

May we record your session?:

Yes

15

HDF5 Viewer Roundtable

Authors: Axel Bocciarelli[1]; Gerd Heber[2]; John Readey[2]; Thomas VINCENT[1]

[1] *ESRF*

[2] *The HDF Group*

Corresponding Authors: gheber@hdfgroup.org, jreadey@hdfgroup.org

A very common need when presented with an HDF5 file (especially for non-programmers or those new to HDF5), is some way to "see"the contents. Happily, there are a variety of viewers for HDF5 data sources available: HDFView, H5Web (myhdf5), HDF Compass, etc. However, it's not obvious how these compare or when one or another might be preferable for a particular application. In this session, we'll have a short review of HDFView, H5Web, and HDF Compass, covering their history, features, intended use, and future roadmap. We will conclude with an open discussion of what future steps would best serve the needs of the HDF5 community.

May we record your session?:

Yes

16

HDF5: The most versatile container for sharing scientific and engineering data

Author: Gerd Heber[1]

[1] *The HDF Group*

Corresponding Author: gheber@hdfgroup.org

When people hear this affirmation, a common reaction is, "When, HDF Group, when?" In her book *How to Make Sense of Any Mess*, Abby Covert reminds us to be careful and not fall in love with our plans or ideas but with the effects we can have when we communicate clearly. While I would reject the notion that "a mess" aptly describes the current state of HDF5, I want to use this short presentation to clearly communicate our plans and ideas vis-à-vis HDF5 and the community.

May we record your session?:

Yes

17

Zarr access via NetCDF-C

Author: Manuel Reis[1]

[1] *Deutsches Klimarechenzentrum - DKRZ*

Corresponding Author: reis@dkrz.de

With the growing adoption of the Zarr data format for scalable and cloud-optimized storage, NetCDF has introduced an interface to support Zarr access. This integration enables a broader range of scientific software, beyond the Python ecosystem, to interact with Zarr datasets through the familiar NetCDF API. In this presentation, we will discuss the current state of the NetCDF-Zarr implementation, explore opportunities for optimizing I/O patterns, and highlight desirable directions for future support of remote data access features.

May we record your session?:

Yes

18

Down to the bytes: can we simplify alternative access to HDF5?

Author: Thomas Kluyver[1]

[1] *Eur.XFEL (European XFEL)*

Corresponding Author: thomas.kluyver@xfel.eu

HDF5 is an enormously powerful and flexible file format. There are many different ways to use it, and it's difficult to provide one API that works efficiently for all the possible use cases. However, the complexity of the on-disk file format is a high barrier to alternative implementations, so with a few heroic exceptions, most code reading & writing HDF5 does so through the canonical C implementation, including its bindings in various other languages.

In this open-ended talk, I aim to give an overview of a range of different ways to access HDF5 files, bypassing part or all of libhdf5. This will include practical techniques to speed up access in specific circumstances, projects building additional indexes around HDF5 files, and alternative implementations of the file format. I'll conclude with some ideas on how we might lower the barrier and make it easier to implement optimised HDF5 file access for specific scenarios.

May we record your session?:

Yes

19

Revisiting HDF5 SWMR and file versioning features

Author: Elena Pourmal[1]

Co-author: Scot Breitenfeld [2]

[1] *Lifeboat, LLC*

[2] *The HDF Group*

Corresponding Authors: epourmal@gmail.com, brtnfld@hdfgroup.org

In this talk, we will present an overview and demonstration of two HDF5 features implemented via virtual file drivers (VFDs), both currently in a prototype stage: the full single-writer/multiple-reader capability (VFD SWMR) and HDF5 versioning (also known as the Onion VFD).
The VFD SWMR feature enables file modifications during writing and provides guarantees on the maximum latency before new data becomes available to readers. The Onion VFD allows tracking of

modifications to an HDF5 file between open/close calls while preserving—or providing access to—the file's state prior to a given set of changes.
Although the Onion VFD has been released and is available in current maintenance versions, a more robust implementation is still in development. In this presentation, we aim to gather feedback from the HDF5 user community on both VFDs and to share our roadmap for developing production-ready versions of these features.

May we record your session?:

Yes

20

Using a HDF5 File as a Zarr v3 Shard

Author: Mark Kittisopikul[1]

[1] *Howard Hughes Medical Institute*

Corresponding Author: kittisopikulm@janelia.hhmi.org

Version 3 of the Zarr specification includes a sharding codec that allows for chunks to contain small inner chunks. The format of the resulting binary file format of shards is reminiscent of a HDF5 file. Both HDF5 files and Zarr v3 shards may contain compressed chunks. Furthermore, the Zarr v3 shard specification is similar to the Fixed Array Data Block structure within a HDF5 file. Additionally, the sharding concept is similar to a subset of the HDF5 virutal dataset feature. I will discuss how a standard HDF5 file could be used a Zarr v3 shard and how a set of such files could be used as a Zarr array or assembled into a HDF5 virtual dataset. Finally, I will discuss potential cooperation between Zarr and HDF5 and alternatives to my approach.

As a bonus topic, time permitting, I could discuss how a single file could simultaneously act as a valid TIFF file, HDF5 file, and Zarr v3 shard, taking advantage of cloud optimization of each of these formats.

May we record your session?:

Yes

22

h5pydantic

Author: Clinton Roy[1]

[1] *ANSTO*

Corresponding Author: clinton.roy@gmail.com

h5pydantic is a Pydantic based library aimed at making it easier for scientists to organise their HDF5 files, by writing Python models of their experiments. The library is similar to an Object Relational Mapper (ORM), but instead of targeting a relational database, it targets HDF.

The library is inspired by the need of the Australian Synchrotron during the commissioning of our new beamlines under the BRIGHT program. The library is currently under development.

May we record your session?:

Yes

23

Keynote: From Data Blocks to Code Snippets: Boosting HDF5 with GitHub Copilot

Author: Julia Kordick[1]

[1] *Microsoft*

Corresponding Author: julia.kordick@microsoft.com

Working with HDF5 often means navigating large datasets, verbose APIs, and boilerplate-heavy code. In this demo-heavy session, we'll explore how AI-assisted coding tools—specifically GitHub Copilot—can accelerate common HDF5 workflows across C, C++, and Python. From auto-generating read/write boilerplate, to documenting complex structures, to scaffolding tests and data conversion routines, Copilot is more than autocomplete—it's your new lab assistant.

We'll walk through examples, showing where Copilot shines, where it struggles, and how to make the most of it in a scientific or engineering context.

Whether you're wrangling metadata, porting legacy scripts, or just tired of writing H5Fopen() for the hundredth time—this session will show you how AI can actually help.

24

Streamlining HDF5's AI Workloads Benchmarking

Corresponding Author: dlyaver.djebarov@rwth-aachen.de

Rapid adoption of artificial intelligence (AI) in scientific computing requires new tools to evaluate I/O performance effectively. HDF5 is one of the data formats frequently used not only in HPC but also in modern AI applications. However, existing benchmarks are insufficient to address the current challenges posed by AI workloads. This talk introduces an extension to the existing HDF5 benchmark, called h5bench, by incorporating the workload characteristics from the MLPerf Storage - DLIO benchmark. This extension allows users to test AI workloads together with traditional HPC benchmarks in the same context without the complexities of installing various machine learning libraries. Our experimental analysis demonstrates that the extension can replicate the existing I/O patterns with easily customizable configurations to perform various scaling tests.