HDF5 Wins 2002 R&D 100 Award

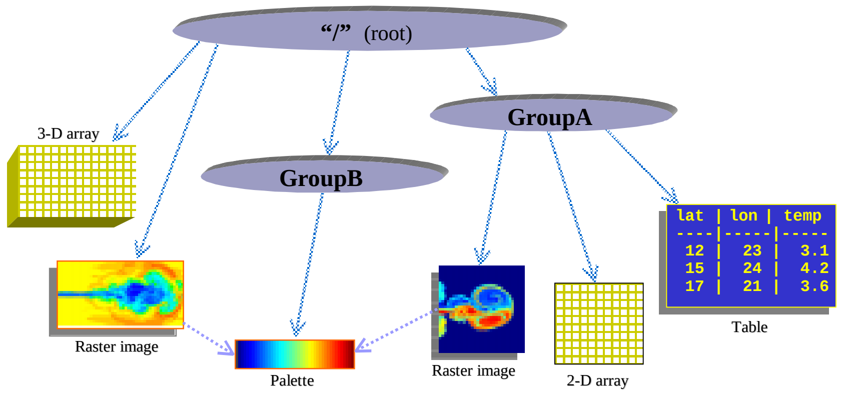
HDF5 – Hierarchical Data Format 5
National Center for Supercomputing Applications (NCSA)
at the University of Illinois at Urbana-Champaign (UIUC)

By the mid-1990s, it was apparent that the data generated by and supporting many scientific research programs had outgrown the capacities of then-current data-handling software. The HDF Group at NCSA, builders of the original HDF, then set out to design a new data format and software library that would meet the growing needs of scientific research, facilitate scientific collaboration, and take advantage of the ever-increasing capacities of computing systems.

The HDF Team at NCSA

The result of that effort is HDF5, a completely new file format and software library for data storage, management, exchange, and archiving of large and complex scientific, engineering, and other data. HDF5 includes

- a generalized and highly adaptable data model,
- a completely portable file format, so that a file can be written on any system and read on any other,
- support for datasets as large or complex as any research or development project requires, and
- a library that can run on virtually any scientific research computing system, including massively parallel systems.



Sample HDF5 file with groups to provide structure, datasets, raster images, a palette

Among other benefits, we believe that the combination of features designed and built into HDF5, and described and elaborated in the accompanying materials, enable a greater degree of collaboration than any other format or library available today.

Table of Contents

What is HDF5? And What are Its Primary Functions?

HDF5 is a data format and an associated software library designed to store, access, manage, exchange, and archive diverse, complex data in continuously evolving heterogeneous computing and storage environments. HDF5 is extensively used with scientific research, engineering development, and other data.

HDF5 supports any type of data suitable for digital storage, regardless of its origin or size. For example, petabytes of remote sensing data received from satellites, terabytes of computational results from weather or nuclear testing models, and megabytes of high-resolution MRI brain scans are stored in HDF5 files along with additional information necessary for efficient data exchange, data processing, visualization, and archiving. The HDF5 format and library provide a powerful means of organizing and accessing data in a manner that allows scientists to share, process, and manipulate data in today's heterogeneous and quickly-evolving high-performance computational environment, including the emerging computational GRIDs.

There are many data formats and software libraries that have been used by scientists and industry to store and share data in particular scientific fields since early 1960s. For example, the FITS format and library is used by astronomers to store and process celestial data from optical and radio telescopes. The PDB format from the Lawrence Livermore National Laboratory (LLNL) is used by physicists to efficiently handle data in heterogeneous computational environments. NASA Earth scientists archive remote-sensing data in HDF4 files. The NetCDF file format and library were created by NCAR to facilitate data storage and exchange in atmospheric research and modeling. The TIFF and GIF file formats are successfully used to store a wide variety of images.

Most currently available data formats were created to store data that can be easily described by conventional data structures such as multidimensional arrays of numbers, tables or records, and images. Most of the libraries also addressed the issues of efficient data access and storage, and file portability. But today many of them cannot easily address the challenges of new computing systems and architectures, such as tremendous data volume (i.e., terabytes or petabytes of data, where most data-handling software is limited to 2 gigabyte files), complex data structures such as irregular meshes, highly diverse datatypes, heterogeneous computational environments, parallel data access and processing, the diversity of physical file storage media, and varying notions of the file itself. The rigid data models of most current file formats become an obstacle in using them in multidisciplinary science. HDF5 was designed and implemented to address these current challenges and to be ready to face future developments.

The combination of the features listed below make HDF5 a unique and "breakthrough" technology:

Unlimited size, extensibility, and portability

- HDF5 does not limit the size of files or the size or number of objects in a file.
- The HDF5 format and library are both extensible and designed to evolve gracefully with the articulation of new demands.
- HDF5 functionality and data is portable across virtually all computing platforms used in scientific research and is distributed with C, C++, Java, and Fortran90 programming interfaces.

General data model

- HDF5 has a very simple but versatile data model. HDF5 is compatible with all of the competing formats discussed in Item 10b in that those data models can be expressed in terms of HDF5. For example, the HDF5 team has developed a netCDF prototype on top of HDF5 (see http://hdf.ncsa.uiuc.edu/HDF5/papers/netcdfh5.html).
- Through its grouping and linking mechanisms, the HDF5 data model enables complex data relationships and dependencies.
- HDF5 accommodates the inclusion of many common types of meta data and arbitrary types and quantities of user-defined meta data.

Flexible, efficient I/O

- HDF5, through its virtual file layer (VFL), offers extremely flexible storage and data transfer capabilities by means of special-purpose file configurations and powerful I/O mechanisms, including standard I/O, parallel I/O, and network I/O.
- An application writer can add additional drivers to implement customized data storage or transport.
- The parallel I/O driver for HDF5 makes it possible to write data in parallel directly to HDF, resulting in improved access times on parallel systems.

Flexible data storage

- HDF5 employs various data compression, data extensibility, and chunking strategies to enhance data access, management, and storage efficiency.
- HDF5 provides for external storage of raw data, often saving disk space and allowing raw data to be shared among HDF5 files and/or applications.

Unlimited variety of datatypes

- HDF5 either offers or enables the creation of a virtually unlimited variety of datatypes and imposes no limit on the complexity of a user-defined datatype.
- Any datatype can be stored in an HDF5 file and shared among other objects in the file, providing a powerful and efficient mechanism for describing data.
- Datatype storage includes all relevant information, such as endianness, size, and architecture (e.g., IEEE, STD, MIPS).

Data transformation and complex subsetting

- HDF5 enables datatype and spatial transformation during I/O operations.
- HDF5 data I/O functions can operate on selected subsets of the data.

How Does HDF5 Compare with Other Data Format Products?

List of Competitors

There are many data formats and libraries. When considering competitive issues, we have looked for formats and libraries that

- are portable,
- are freely distributed,
- are well supported,
- have been designed for use with scientific data, and
- are successfully used with scientific data applications.

We exclude formats such as GIF and JPEG as they do not meet all of the above criteria. While they are widely used data formats, they store only images and, in the case of the GIF format, can incur licensing requirements.

Applying these criteria, HDF5's competitors are the NetCDF, HDF4, PDB, FITS I/O, OpenDX, and TIFF file formats and supporting libraries.[1] These file formats are portable between computers of different architectures and are widely used by the scientific and engineering community for data storage, management, archiving, and exchange. In addition to the above criteria, they all are based on similar data models, are *de facto* standards in their respective fields, and the first four provide high performance I/O libraries along with the file format. HDF5 is compared with each of these formats and libraries in the Competitive Matrix of Item 10b.

The NetCDF (http://www.unidata.ucar.edu/packages/netcdf/index.html) data format and library were developed by the Unidata Program Center in the late 1980s to provide atmospheric scientists with a portable file format and I/O library to facilitate data exchange. NetCDF is used now beyond the atmospheric sciences. It has C, C++, Fortran, Perl, and Java programming interfaces, is supported on most platforms, and is distributed free of charge. The NetCDF data model is a multidimensional array of basic type elements.

The HDF4 (http://hdf.ncsa.uiuc.edu) data format and library were developed by NCSA, at the University of Illinois at Urbana-Champaign, in the late 1980s to facilitate data exchange between NCSA scientists. It very quickly became widely used in many scientific and engineering fields. In 1993, NASA chose NCSA's HDF format to be the standard file format for storing data from the Earth Observing System (EOS), which is the data gathering system of sensors (mainly satellites) supporting the Global Change Research Program. It has C, Fortran, and Java interfaces, is supported on most platforms, and is distributed free of charge. The HDF4 data model includes multidimensional arrays of basic type elements, annotations (text), tables, raster images, and grouping structures.

The PDB (http://pact.llnl.gov/PACT_Docs/pdb/pdb.html) format and library was developed at Lawrence Livermore National Laboratory (LLNL), a U. S. Department of Energy laboratory, to provide scientists with file management routines for storing and retrieving binary data in portable format. It has C and Fortran interfaces, is supported on most platforms, and is distributed free of charge. The PDB data model is a multidimensional array of structures (records).

The FITS (http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/fits_libraries.html) portable data format and I/O library were developed in the early 1960s at NASA Goddard Space Flight Center for storing and analyzing astronomical data sets. It has C and Fortran interfaces, is supported on most platforms, and is distributed free of charge. The FITS data model includes a multidimensional array of basic types, ASCII and binary tables, and a grouping mechanism.

[1] Another popular format that meets these criteria is NASA's CDF format. Since CDF is similar in most respects to netCDF, we omit it from this comparison.

Under OpenDX, we are referring to the underlying data model used by the IBM Data Explorer Visualization Software (http://www.research.ibm.com/dx/index.html). It has a very rich data model based on the notions of field, array and group objects and is used to represent different kinds of meshes and data defined on those meshes. OpenDX uses ASCII files to describe relations between objects and ASCII, binary (not portable), NetCDF, or HDF4 files for problem size data. We chose OpenDX as an example of a data model and file format that represents a class of widely used visualization applications such as AVS, IDL, and EnSight.

TIFF (http://partners.adobe.com/asn/developer/pdfs/tn/TIFF6.pdf) is a portable tag-based file format for storing and exchanging raster images. It is designed to accommodate changes and enhancements as new imaging needs arise. While the TIFF format is clearly defined, there is no publicly-available, standard TIFF library.

Competitive matrix

The following table, the Competitive Matrix, discusses features that can be directly compared across the libraries and file formats listed in Item 10a.

HDF5:	Hierarchical Data Format 5, data format and library from NCSA (the nominee)
HDF4:	Hierarchical Data Format, data format and library from NCSA
netCDF:	Network Common Data Form, data format and library from UCAR
PDB:	Portable Data Format, data format and library from LLNL
FITS:	Flexible Image Transport System, data format and library from NASA-GSFC
OpenDX:	Data Explorer Visualization Software, data model and file format from IBM
TIFF:	Tag Image File Format, data format from Adobe Systems

The Competitive Matrix consists of two major sections. The first section, Object Features, compares features of objects common to all formats. The second, Library Features, compares library features that are common among data-handling libraries.

As mentioned above, this table is structured around the features that can be directly compared across the relevant libraries and formats. The discussion in Item 10c, which is structured around the key features of HDF5 that we believe make it a winning candidate for the R&D 100 Award, draws further comparisons.

Competitive Matrix[1]

Code: B: **B**reakthrough capabilities enable features that have not previously been easily available to general application developers.
U: **U**nique capabilities are capabilities that are not available in the other libraries.
S: **S**ignificant capabilities provide significant improvement over other libraries.

	Code	Feature	HDF5	HDF4	netCDF	PDB	FITS	OpenDX	TIFF
	Object Features:								
1	**Files**								
1.1	S	Parallel file access	Yes, on any system with MPI I/O	No	(In development)	Yes, on SMP machines	No	Yes, on SMP machines	No
1.2	BUS	Storage media for the file	File system, memory, network Any future device may be added by using VFL	File system only	File system only	File system only	File system only	File system only	File system only
1.3	S	Grouping mechanism with information retained in file	Yes	Yes	No	Yes	Yes	No	No
2	**Scalability of objects**								
2.1	US	Number of objects	Unlimited	Limited	Limited	Limited	Limited	Limited	Limited
2.2	US	Maximum size of objects/files	Limited only by computer or file system capacity	2^{31}-1 bytes	2^{31}-1 bytes	2^{31}-1 bytes	2^{31}-1 or 2^{63}-1 bytes [2]	2^{31}-1 bytes	2^{31}-1 bytes
3	**Object storage**								
3.1	S	External storage of raw data	Yes	Yes	None	None	None	Yes	None
3.2	S	Chunking storage	Yes	Yes	None	None	None	None	Yes
3.3	BUS	Ability to tune chunked I/O in library	Yes	No	None	None	None	None	None
3.4	US	Arrays can be extended in any direction	Yes	No	No	No	No	No	No
4	**Metadata handling capabilities**								
4.1	BUS	Ability to store metadata separately from raw data	Flexible	Limited	No	No	No	Limited	No
4.2	S	Support for complex metadata structures	Yes	No	No	Yes	No	No	No
5	**Datatypes**								
5.1	S	Simple integer and floating point datatypes	Integer and floating point types of any size or precision	Integer: 8, 16, 32, 64-bit signed and unsigned Float: 32, 64-bit	Integer: 8, 16, 32-bit signed Float: 32, 64-bit	Integer: 8, 16, 32, 64-bit signed Float: 32, 64-bit	Integer: 8, 16, 32, 64-bit signed and unsigned Float: 32, 64-bit	Integer: 8, 16, 32, 64-bit signed and unsigned Float: 32, 64-bit	Integer: 8, 16, 32-bit unsigned Float: 32, 64-bit IEEE format
		Other simple datatypes	Variable-length array String Object reference		String	Pointer String	String Logical Complex	String Complex	Rational: represented by 2 integers (machine dependent)
5.2	S	Compound datatypes (record structures)	Any amount of complexity, including types nested to any number of levels	One level, as part of a table structure	Not available	User-defined structures as defined in C language	One level, as part of a table structure	Not available	Not available
5.3		Full datatype definition stored in file	Yes	No	No	Yes	No	No	Not applicable
5.4	US	Shared datatypes	Yes	Not available	Not available	Not available	Not available	Not available	Not available
5.5	S	User-defined datatypes, such as 13-bit integers or 128-bit floats	Yes	Not available	None	Yes	None	Not available	None
5.6	S+	Allowable elements of arrays	Any HDF5 type	Any HDF4 simple type	Any netCDF simple type	Any PDB type	Any FITS simple type	Any DX simple type	Any TIFF simple type

[1] Comparisons are based on available format and library distributions as of 1 January 2002.
[2] FITS files are not portable between 32-bit and 64-bit systems.

	Code	Feature	HDF5	HDF4	netCDF	PDB	FITS	OpenDX	TIFF
6	***Library Features:*** **Performance tuning mechanisms**								
6.1	US	Chunking I/O	Yes	No	No	No	No	No	Not applicable
6.2	US	Hyperslab selection I/O	Yes	No	No	No	No	No	Not applicable
6.3	US	Conversion I/O	Yes	No	No	No	No	No	Not applicable
6.4	US	Storage in general [1]	Tunable B-tree ratios, group-to-object relationships	No	No	No	No	No	Not applicable
6.5	US	Separation of small metadata I/O requests and large problem-sized I/O requests	User configurable	No	No	No	No	No	Not applicable
7	**Partial I/O capabilities**								
7.1	S	Subsetting of compound datatypes by field	Subsetting of compound datatype members at any level	Subsetting by fields in Vdatas	Not available	Not available	Read/write columns Table manipulation (insert/delete rows and columns)	Not available	Not applicable
7.2	US	Subsetting of arrays	Hyperslabs, union of hyperslabs, point selections. Set operations on the hyperslabs are designed in.	Simple hyperslab subsetting	Simple hyperslab subsetting	None	Simple hyperslab subsetting	Not available	Not applicable
7.3	S	Speed of access to objects	Fast	Slow	Moderate	Moderate	Slow	Unknown	Not applicable
8	**Data transform during I/O**								
8.1	S	Ability to convert data when reading or writing	Converts any type to any other of the same class (e.g. integer-to-integer). Other conversions can be easily added.	Converts any type to any other of the same class (e.g. integer-to-integer).	Converts types of the same class, but only if they are the same size.	Supports datatype conversion. Other conversions can be easily added.	Supports implicit datatype conversion during read/write operations.	Not available	Not applicable
8.2	S	Data compression and other conversions	GZIP compression is built in. Applications can add other compression methods or conversions.	GZIP, JPEG, Adaptive Huffman compression	None	None	None	None	JPEG, PackBits compression, Modified Huffman, LZW compression

[1] E.g., the number of entries in a group

Summary of improvements over competitive products and technologies

The HDF5 file format and library provide a unique synthesis of flexibility and advanced features that make it a breakthrough in data storage, archiving, and management. In this section, we provide comprehensive descriptions and comparisons of those features that are unique or represent significant improvement over available libraries. Cross-references link this discussion to the features discussed in the competitive matrix of Item 10b.

- Data model and file structures
- Data volume (size and number of objects)
- Datatypes
- Virtual file layer, VFL
- Meta data-handling capabilities
- Improvements to I/O
- Data storage (compression, chunking, extendibility of objects, and external storage)
- Data transformation and partial I/O

Data model and file structures

HDF5 is based on a simple, yet powerful and flexible data model. The data model consists of two primary types of objects: datasets and groups.

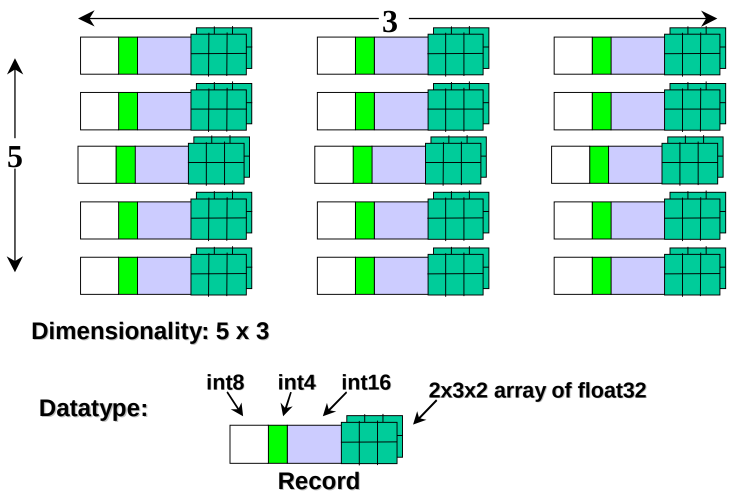


Figure 10.1: Sample HDF5 dataset – an array of records. This dataset consists of a 5x3 array of records. Each record contains four fields: the first field is an 8-byte integer, the second is a 4-byte integer, the third is a 16-byte integer, and the fourth, the last field, is a 2x3x3 array of 32-bit floats.

Datasets are multidimensional arrays of simple or compound HDF5 datatypes. HDF5 simple datatypes are similar to the C or Fortran integer, float, and character types; HDF5 compound datatypes can be compared to C or Fortran structures, or records, and can be nested. Vector or tensor data fields, or inventory records can be conveniently stored in an HDF5 dataset. As indicated in Competitive Matrix in Item 10b, only PDB has a similar feature; the difference is described in the Partial I/O section of that table.

HDF5 groups and links allow the creation of complex data dependencies reflecting the nature and/or intended usage of the stored data. The grouping mechanism combines related objects together; the linking mechanism, which is similar to the hard and soft links of UNIX environments, allows the sharing of objects between different groups. HDF4, PDB and FITS also provide

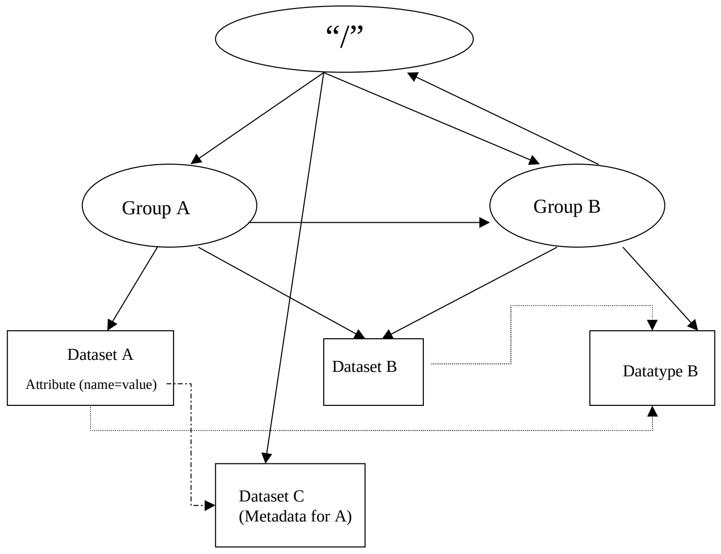


Figure 10.2: Sample HDF5 file structure illustrating the main concepts of the HDF5 data model. The file contains two groups, "A" and "B". Group "B" is a member of group "A". Both groups are members of the root group, "/", which serves as an entry to the file structure graph. The root group itself is a member of group "B". Dataset "B" is a member of groups "A" and "B". Datasets "A" and "B" share the same datatype "B" that is stored in the file under group "B". The attribute of the dataset "A" points to another dataset, "C", which may be used as meta data of "A".

grouping mechanisms, but only HDF5 has the general dependencies structure constituted as a directed graph with a designated entry point.

Figure 10.2 provides an example of the HDF5 file structure and illustrates the main concepts of the HDF5 data model. Each HDF5 object may have associated metadata stored in the file in the form of simple attributes (name=value pair). "value" may be a pointer to another HDF5 object stored in the file allowing sharable attributes that have the problem size data. This is a significant feature that is critical to the management, understanding, and reuse of scientific data.

The HDF5 data model is compatible with the data models of other formats, and it can also accommodate such complex structures as the irregular grids used in solid or fluid mechanics simulations.

Data models of the data formats listed in Item 10A can all be easily expressed in terms of the HDF5 data model. The HDF5 group created a netCDF prototype based on the HDF5 (see http://hdf.ncsa.uiuc.edu/HDF5/papers/netcdfh5.html). Other examples of using HDF5 to accommodate complex structures, such as the irregular grids used in fluid mechanics, include HDF-EOS5 (see http://hdfeos.gsfc.nasa.gov/hdfeos/SoftwareDist.html), EnSight (see ftp://ftp.ncsa.uiuc.edu/HDF/HDF5/contrib/h5_ensight/), and SAF (no public reference currently available).

Data volume (size and number of objects)

Unlike many existing data management systems, HDF5 does not limit the size or number of objects that can be stored in an HDF5 file. HDF5 can manage data on architectures with different memory and storage architectures, including those supporting 64-bit and 128-bit address spaces. Furthermore, files created on one architecture can be accessed on another architecture, including being supported by the corresponding operating systems, without any special application code. This ability to scale across architectures increases the portability of data stored in HDF5 and ensures that the data will be available in the future as computer architectures evolve.

Datatypes

HDF5 has a rich collection of datatypes. An HDF5 datatype is a collection of properties providing a complete definition of the characteristics of the data involved and is stored in the file so that it is available whenever and wherever the data is accessed. HDF5 predefined datatypes are similar to the types used in C and Fortran.   HDF5 also defines complex datatypes such as strings, arrays, object pointers, integers of user-defined length, floats of user-defined precision, and compound datatypes similar to C structures or SQL records. There is no limitation to the complexity of a datatype and elements of a compound datatype may themselves be of a compound datatype.

Libraries listed in Item 10A achieve file portability by storing data in big-endian format. To make HDF5 files portable, the HDF5 library stores datatype characteristics such as size, bit order, precision, and architecture in the file. This feature provides great flexibility in the use of the library, enabling datatype conversions, for example; and it may be used to decrease data post-processing time. For example, a powerful machine that is generating the raw data can write the file in the native format of a slower target machine of a different architecture so that the file can be efficiently read on that target system.

Any datatype can be stored in an HDF5 file and shared among several objects in the file; this capability provides a powerful and efficient mechanism for describing and storing data.

Virtual file layer , VFL

The HDF5 library incorporates a *virtual file layer, or VFL,* that allows applications to specify particular file storage media such as network, memory, or remote file systems; to specify different file systems on the same machine; or to specify special-purpose I/O mechanisms such as streaming I/O, MPI I/O, and buffered I/O. The VFL enables alternative I/O mechanisms on the application level, even providing public APIs so that application developers can write new drivers and plug them into HDF5 Library. This feature is available only in HDF5.

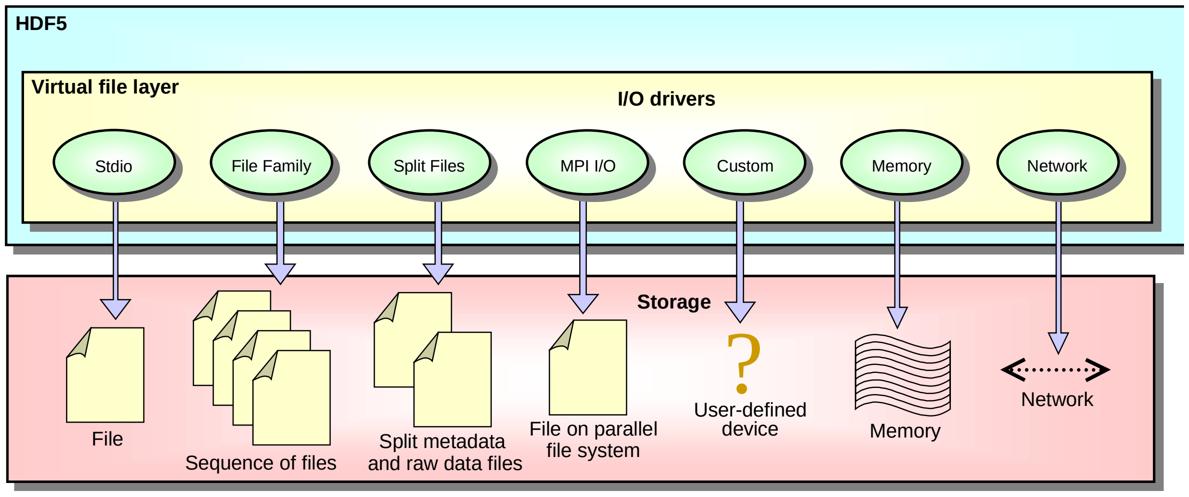


Figure 10.3: The HDF5 virtual file layer (VFL). Through the use of the VFL, an HDF5 file can be stored as a conventional UNIX file or as multiple files (to overcome a 32-bit system limit on file size); it can be stored as two or more files containing separated meta data and raw data; it can be stored as multiple files on a parallel file system; it can be saved in memory or sent over the network; or it can be handled by another non-standard, user-supplied driver.

Figure 10.3 offers a conceptual view of the VFL and illustrates the diversity of the storage media available for an HDF5 file.   New I/O drivers can be added to the VFL in a standard way as needed.

Optional separation of metadata and raw data

HDF5 is unique in its capability to configurably separate meta data from raw data and in making this configurability available to the application level. At the simplest level, HDF5's split driver separates meta data and raw data by creating a separate file for each. As illustrated in Figure 10.4, these files are treated as one logical HDF5 file at the application level. To meet more sophisticated requirements, the multi driver can separate the five types of meta data into up to five separate physical files. With the raw data file, this creates up to six physical files that are treated as one logical HDF5 file at the application level.

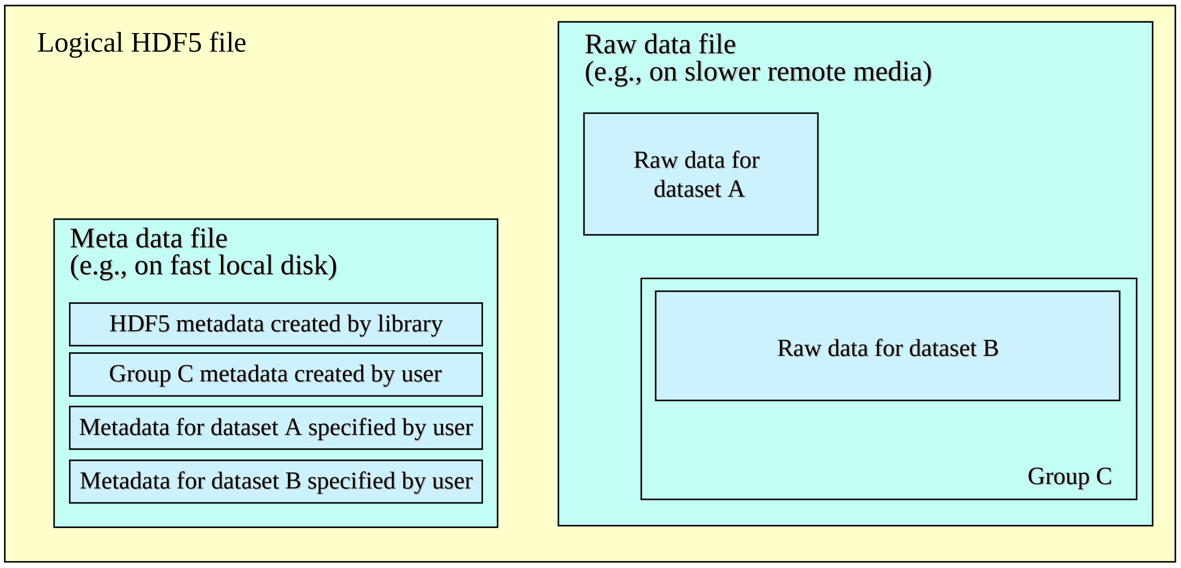

Logical HDF5 file
Raw data file
(e.g., on slower remote media)
Raw data for dataset A
Meta data file
(e.g., on fast local disk)
HDF5 metadata created by library
Group C metadata created by user
Metadata for dataset A specified by user
Metadata for dataset B specified by user
Raw data for dataset B
Group C

Figure 10.4: A logical HDF5 file written by the split driver. This file is stored on media as two physical files, one containing meta data and one containing raw data. The meta data might be stored on a file system tuned for many small reads and writes while the raw data file is stored on a file system tuned for large datasets.

The physical files may reside on the same file system or on different file systems as long as all the file systems are available on the machine. Since meta data often requires small I/O requests, this feature can be especially important for performance in massively parallel environments, where small I/O requests may cause tremendous slowing of I/O performance.

High I/O performance

In many cases, the use of data management software requires a trade-off in reduced I/O performance. The HDF5 library requires no such sacrifice, delivering nearly all of the available performance.

Benchmarks we have run showed significant I/O performance advantages for the HDF5 sequential library in general. For the parallel library, usage of the split driver allows performance to reach the same level as that achieved by the underlying MPI I/O (see http://hdf.ncsa.uiuc.edu/HDF5/papers/SC2001/SC01_tutorial/Session-IV.pdf).

Figures 10.5 and 10.6 illustrate benchmark results of read and write operations on a fixed-size contiguous dataset in a sequential IRIX environment. HDF5 performance is compared with that of FITSIO, HDF4, and netCDF; the read buffer varies from 64 to 512 megabytes. The performance shown is measured in megabytes/second.

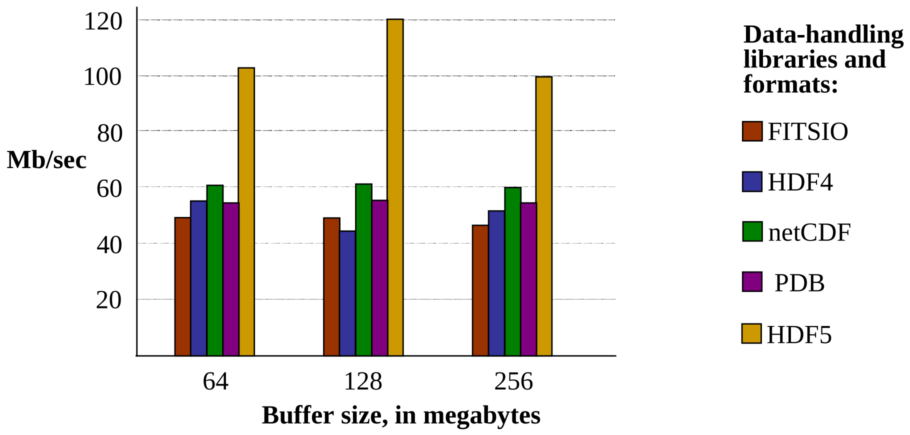


Figure 10.5: Sequential reading benchmarks. Reading a contiguous dataset on an Irix system. Times reported include all required operations: opening the file and dataset, reading the dataset, and closing the dataset and file.

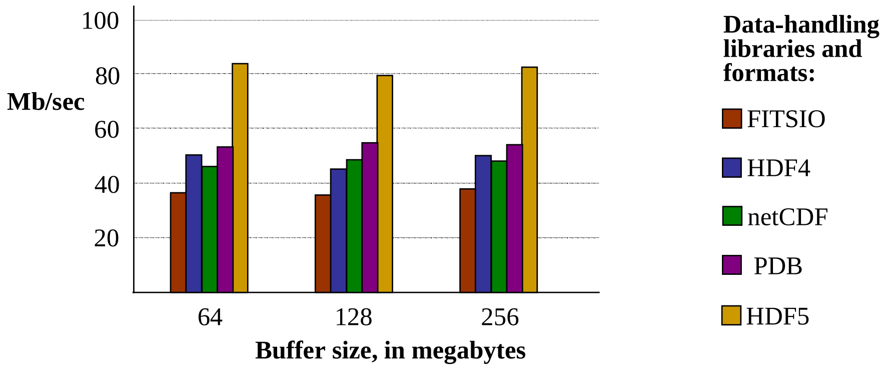


Figure 10.6: Sequential writing benchmarks. Writing a contiguous dataset on an Irix system. Times reported include all required operations: creating the file and dataset, writing the dataset, and closing the dataset and file.

Figure 10.7 illustrates the benchmark results of write operations in a parallel environment. Note that the HDF5 library with the split file driver achieves nearly 100% of the throughput capacity of the underlying MPI I/O software.

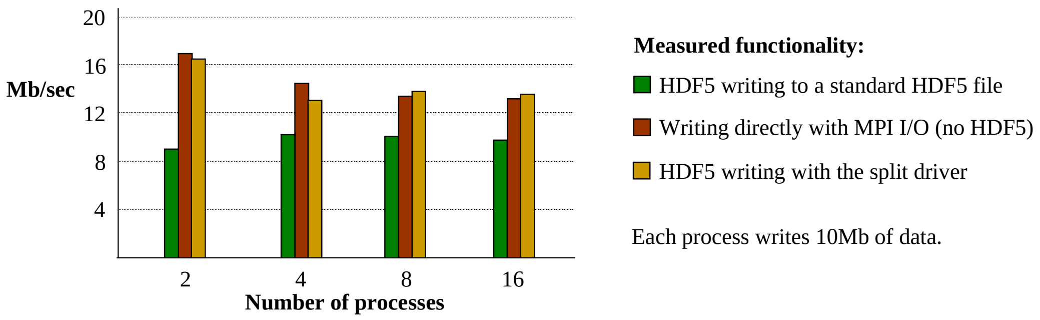


Figure 10.7: Parallel writing benchmarks on Tflops (at SNL). This chart compares the following operations: HDF5 writing to a standard file, the underlying MPI I/O write operation, and HDF5 writing to a split file. The number of processors varies from 2 to 16. The results are shown in megabytes per second. Each process writes 10 Mb of data, so the 2-process test writes a 20 MB file, the 4-processor test a 40 Mb file, and the 16-process test a 160 Mb file.

Data storage (compression, chunking, extendibility of objects, and external storage)

HDF5 employs several strategies, such as compression, chunking (tiling), object extendibility, external storage of raw data for the dataset, and raw and meta data separation, to enhance the efficiency of data access, management, and storage.

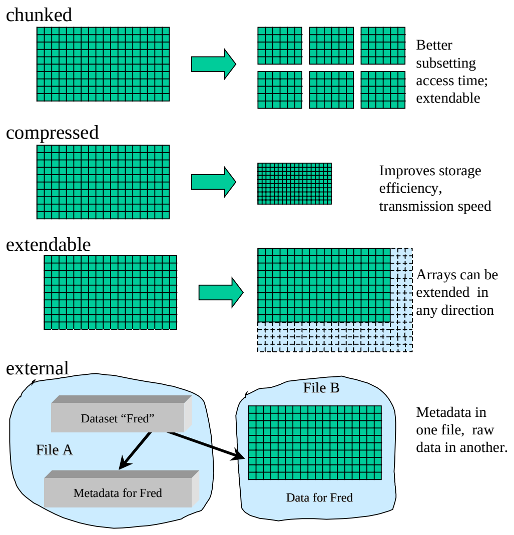

chunked
Better subsetting access time; extendable
compressed
Improves storage efficiency, transmission speed
extendable
Arrays can be extended in any direction
external
Dataset “Fred”
File A
Metadata for Fred
File B
Data for Fred
Metadata in one file, raw data in another.

Figure 10.8: HDF5 special storage options. HDF5 Datasets can be broken into multiple chunks to provide efficient data access, compressed to reduce file space and speed data transfer, extended at will and in any dimension to provide maximum flexibility as datasets grow and provide for data that is of an unknown size, or stored externally to take advantage of the characteristics of different file systems.

HDF5 uses the standard GNU zlib library (http://www.gnu.org/directory/zlib.html) to store compressed data; if this method is not effective for a particular type of data, the user can plug in his own more appropriate compression or filtering method to transform data during I/O operations. Other libraries do not provide this option to application developers.

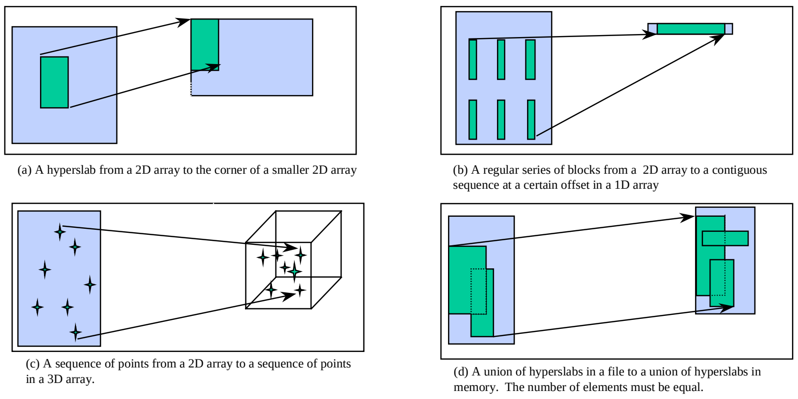
HDF5 does not require all data to be written at once; datasets may be extended later if necessary. Furthermore, HDF5 datasets may be extended along any dimension. HDF4 and netCDF allow datasets to be extended only along one dimension, while other libraries do not have this feature at all.

Since HDF5 is designed to store very large datasets, a chunking mechanism is provided for efficient partial I/O access to the data. External storage of raw data can save disk space and allows raw data to be shared among HDF5 files and/or applications.

The advantages of separating meta data and raw data into different physical files is discussed above, in the section “Optional separation of meta data and raw data.”

Data transformation and partial I/O

HDF5 provides powerful mechanisms for spatial and datatype transformations during I/O operations.



(a) A hyperslab from a 2D array to the corner of a smaller 2D array

(b) A regular series of blocks from a 2D array to a contiguous sequence at a certain offset in a 1D array

(c) A sequence of points from a 2D array to a sequence of points in a 3D array.

(d) A union of hyperslabs in a file to a union of hyperslabs in memory. The number of elements must be equal.

Figure 10.9: HDF5 spatial subsetting operations. HDF5 provides several means of taking a spatial subset of a dataset. The selection can be as a set of individual points (c), as a simple hyperslab (a), or as a compound hyperslab (b or d). A hyperslab or compound hyperslab can be mapped to a hyperslab of a different shape (d).

Spatial transformations define the complex selection of elements and/or modify the shape of the array that participates in I/O. Set operations such as union or difference are used to create a non-regular shaped spatial selection, then I/0 operations are performed on the selected region. The shape of the array can also be changed during I/O operations. For example, a subset of a 3-dimensional array can be read or written to a subset of a 1-dimensional array. Figure 10.9 illustrates several examples of spatial transformation. HDF4, FITS and netCDF provide selections

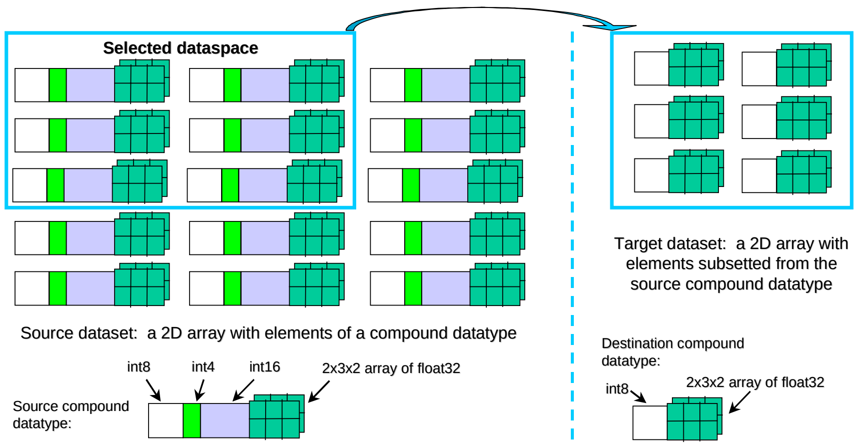

Selected dataspace
Target dataset: a 2D array with elements subsetted from the source compound datatype
Source dataset: a 2D array with elements of a compound datatype
Destination compound datatype:
int8
int4
int16
2x3x2 array of float32
Source compound datatype:
int8
2x3x2 array of float32

Figure 10.10: Spatial transformation with datatype transformation during a read operation. The *spatial* transformation in this operation is the selection of the six elements in the left upper corner of the original dataset. The simultaneous *datatype* transformation is the selection of only the 8-byte integer and 2x3x2 float array fields of each record. Both are subsetting operations. The result of the read operation is a 3x2 array of an HDF5 compound datatype that has an 8-byte integer field and a 2x3x2 float array field.

of contiguous and regularly spaced elements similar to examples (a) and (b) in Figure 10.9. Selections of irregular shapes are available only in HDF5.

Datatype transformations change the datatype of the element during I/O operations. For example, floats can be transformed to other floating point datatypes; integers can be transformed to other integer datatypes; and selected fields of a compound datatype can be mapped to the fields of a different compound datatype. Any combination of the compound datatype members can be selected and can participate in I/O operations. This feature, available only in HDF5, is a generalization of the HDF4 and FITS capability to subset tables by field.

Figure 10.10 illustrates a transformation operation that includes both spatial and datatype transformations.

Primary applications of HDF5

Primary uses of HDF5

The primary applications of HDF5 are in data storage, management, exchange, and archiving. HDF5 brings particular value to projects with large quantities of data, often including many different types of data. Examples of such uses include geophysical and environmental research applications, mathematical modeling, major visualization packages, and commercial real-time sensor applications such as are used on a manufacturing line. Most of this data is scientific research data, but there are also visualization, engineering, manufacturing, and business applications.

Major software applications employing HDF5

Major research applications built on top of or using the HDF5 library include the following:

SAF, LibSheaf, and CDMlib comprise a suite of US Department of Energy (DOE) software libraries central to some of DOE's largest research efforts. All of these libraries use HDF5 as their data I/O layer.

In these systems, HDF5 is used to support highly complex data models. Furthermore, these applications require extremely high-performance parallel I/O; HDF5 has satisfied these I/O performance requirements. Like the HDF5-EOS system, these DOE applications also generate huge data volumes.

As part of DOE's Advanced Simulation and Computing Initiative (ASCI), SAF is being developed at Sandia National Laboratory (SNL) and Lawrence Livermore National Laboratory (LLNL), LibSheaf is being developed by Limit Point Systems, and CDMlib was developed at Los Alamos National Laboratory (LANL). ASCI's goal is "to shift…from nuclear test-based methods to computational-based methods of ensuring the safety, reliability, and performance of our nuclear weapons stockpile." The ASCI Data Models and Formats group (ASCI DMF) has worked closely with us in developing HDF5 to address the data management issues of this historically significant program and to replace divergent storage formats with a single unified underlying format. This involves not only supporting ASCI data requirements in the new HDF5 format, but also implementing the HDF5 IO library on the multi-teraflop ASCI machines, four of the six largest and fastest computing systems in the world.[1]

(No public website is currently available for these DOE applications.)

SILO (http://www.llnl.gov/bdiv/meshtv/manuals.html), from Lawrence Livermore National Laboratory (LLNL), is a high-level, portable scientific data library designed to address difficult data-handling issues such as those imposed by the use of incompatible data formats and libraries. While the Silo/PDB driver is most frequently used, a Silo/HDF5 driver was written when it became evident that the 2 GB file size limit was becoming a problem. In addition to overcoming that limit, the Silo/HDF5 driver includes the following improvements: no data is stored in ASCII format, thus avoiding round-off problems and HDF5's compound datatype feature is available. Further, performance of the following tasks is improved with the Silo/HDF5 driver: opening a file to read a few items, reading scalar data of struct type (objects), and reading large vector data with either byte-order conversions or conversions between floating point and double precision floating point numbers.

Aura (http://eos-chem.gsfc.nasa.gov/), a NASA Earth Observing System mission to study the Earth's

[1] Four such machines have been built for the ASCI program: ASCI White, ASCI Red, ASCI Blue Pacific, and ASCI Blue Mountain. According to statistics compiled at www.top500.org, they comprise four of the six fastest computers in the world.

ozone, air quality and climate, has elected to use HDF5 as its standard data format when the mission flies in 2003. Like other EOS missions (which use HDF4), this system will generate as much as a terabyte of data per day, and will include many different kinds of data from four instruments aboard the satellite. NASA is developing an HDF5-based software package called HDF5-EOS, which is specifically designed for the storage and manipulation of earth-science data. Other applications that read HDF5-EOS files can be found at http://hdf.ncsa.uiuc.edu/hdf5eoss.html.

HL HDF (ftp://ftp.ncsa.uiuc.edu/HDF/HDF5/contrib/hl-hdf5/README.html), from the Swedish Meteorolgical and Hydrological Institute (SMHI, http://www.smhi.se/), focuses on selected HDF5 functionality and makes it available to users at a high level of abstraction, facilitating the management of virtually any type of scientific data. This interface is available in a C version and a Python version, called PyHL.

HL-HDF is an example of a growing body of high-level interfaces to HDF5 that provide more abstractable access to the HDF5 library and file format. Providing such access “at a high level of abstraction” saves application developers a great deal of time as they do not have to address the low-level details of the native HDF5 library and format. High-level interfaces make well-understood reasonable assumptions and manage the calls to the HDF5 library accordingly.

Major commercial applications built on top of or using the HDF5 library include the following:

IDL and IDL-HDF5: The Interactive Data Language (IDL), from Research Systems, Inc. (RSI, http://www.rsinc.com/), is an interactive, high-level language that enables in-depth analysis, decision-making, and diagnosis through data visualization. The current version of IDL reads and writes HDF4 files; RSI is currently developing an HDF5 interface for IDL.

In the meantime, Photon Research Associates, Inc. (http://www.photon.com/) has developed IDL-HDF5, an application programming interface, to access HDF5 files and the library through IDL.

EnSight, from CEI (http://www.ceintl.com/products.html), is a software package for analyzing, visualizing and communicating high-end scientific and engineering datasets CEI is working with NCSA to develop an HDF5-based data format and I/O library that will provide the scalable performance demanded by its user community.

AVS (http://www.avs.com/) is one of the first developers of visualization technology for the scientific, engineering, and technical industries. HDF5 is a natural solution for AVS-based applications that must deal with very large datasets, and as a result software readers and writers have already been implemented to make HDF5 accessible to AVS.

Other applications of HDF5

Other uses for HDF5

While the primary uses of HDF5 are in data storage, management, exchange, and archiving, the library and file format encompass sufficient flexibility that they are used also in image processing applications, in situations requiring simple database functionality, and as an object store for application variables. The grid and mesh capabilities could also be used to support CAD/CAM applications, though we are not aware of such an application at this time.

A sampling of other software applications employing HDF5

Research software

The following examples illustrate the breadth of HDF5's penetration into a wide variety of scientific research communities.

The *Globus Project*, centered at Argonne National Laboratory (ANL) and actively supported by NCSA, ANL, and several partners, is a major development effort focussed on the fundamental technologies required to deploy computational grids. HDF5 is used to support data I/O requirements.

NeXus, from Argonne National Laboratory (ANL, http:www.neutron.anl.gov/nexus/), provides a standard data format for the x-ray and neutron scattering community. As stated on the NeXus website, NeXus was developed to meet the expressed needs of scientists and computer programmers working in neutron and synchrotron facilities around the world for a common data format. As instrumentation has become more complex and data visualization more challenging, individual scientists, or even institutions, have found it difficult to keep up with new developments. A common data format was sought to facilitate the exchange of experimental results ideas about how to analyze them. The NeXus developers selected HDF4 for the original underlying NeXus data format because of the flexibility HDF provides in organizing the instrument descriptions contained in NeXus files (http://www.neutron.anl.gov/nexus/NeXus_intro.html#Criteria). The next generation of NeXus, currently in development, is based on HDF5 (http://www.neutron.anl.gov/nexus/NeXus_2001.html and http://www.hmi.de/projects/ess/vitess/UFilgesVITWSH.pdf).[1]

FLASH, a product of the ASCI/Alliances Center for Astrophysical Thermonuclear Flashes at the University of Chicago, aids in the study of thermonuclear flashes on the surfaces of compact stars. The FLASH code, which is built on Parallel HDF5, is a modular, adaptive, parallel simulation code capable of handling compressible flow problems in astrophysical environments.

The Chombo package, from Lawrence Berkeley National Laboratory (LBNL, http://seesar.lbl.gov/anag/chombo/index.html), provides a set of tools for implementing finite difference methods for the solution of partial differential equations on block-structured adaptively refined rectangular grids. The package includes both elliptic and time-dependent modules. Chombo uses HDF5 to support computing on parallel platforms and to provide standardized self-describing file formats.

ChomboVis, also from LBNL, is built on top of the Visualization Toolkit (VTK) and provides a simple graphical user interface for interacting with 2- and 3-dimensional adaptive mesh refinement (AMR) datasets.

[1] "Introduction to the NeXus Data Format," http://www.neutron.anl.gov/nexus/NeXus_intro.html. Argonne National Laboratory.

Commercial software

HDF Explorer, from Space Research, Inc. (http://www.space-research.pt/), is a data visualization program that read HDF4 and HDF5 files. HDF Explorer reads all HDF4 and HDF5 datatypes and supports image generation from both scalar and vector data, easy browsing through 3-dimensional datasets, and data exporting facilities. Data is explored in a tree-like interface; datasets are displayed in a grid window; and all of this is provided in an easy-to-use yet powerful graphical interface. HDF Explorer is available for Windows NT/95. A free version, known as HDF Explorer (Basic), provides an HDF4 and HDF5 reader.

EarthScan 2000, from EarthScan Network, Inc., is an agricultural remote sensing tool that delivers year-round farm management solutions. EarthScan provides remotely-sensed imagery via the World Wide Web. Satellite and aerial imagery is ingested into an object-relations database repository based on HDF5, the IDL-HDF5 interface (see above), and Microsoft SQL Server 7. HDF5 was selected for its optimized hyperslab extraction capabilities.

And the biggest surprise of the year, 2001 – special effects for the movies

Early in 2001, the HDF5 Help Desk started getting email messages from a programmer in New Zealand. His questions were not of the ordinary sort and our technical support staff eventually asked just how he was using HDF5. All he was free to say at the time was that he was working on graphical special effects. It later developed that he was with the Weta Workshop and was using HDF5 to generate atmospheric effects, the smoke, wind, clouds, and other weather effects, for *The Lord of the Rings* film trilogy.

This episode provides an excellent example of the versatility inherent in the HDF5 design.

Representative List of Technical Fields in which HDF5 is Used

Technical Fields as Indicated in Selected HDF5 Download Registrations
15 October 2001 through 22 February 2002

Aerospace
Agricultural research
Air traffic control
Aircraft emissions database
Applied mathematics
Astrophysics
Astrophysics / supernovae
Atmospheric chemistry
Atmospheric physics
Bioengineering
CEM Simulation
Climatology / hydrology
Computational fluid dynamics
Computational physics
Computational physics / education
Computational physics and computational astrophysics
Computer modeling
Computer science
Data processing
Earth observation / atmospheric science
Earth science
Environment
Fast searching, sorting and retrieval
Fluid mechanics
GIS
Geodetic Science
Geology
Gravitational physics
Hydrology
Information technology
Magnetic mass spectrometer development
Marine biology / ecology
Materials science
Meteorological data products
Meteorology
Microscopy
Molecular biology
Nano device simulation
Neutron scattering
Ocean color
Ocean remote sensing
Optics / optoelectronics
Petroleum engineering
Photonic band gap studies
Photonic crystals
Photonics
Post-fire erosion analysis
Protein crystallography, molecular modeling
Protostellar accretion discs
Remote sensing
SAR processing
Satellite / weather radar remote sensing
Satellite oceanography
Semiconductor process simulation
Software engineering, distributed systems
Space geodesy
Space physics
Surface water flow and sediment transport
Theoretical chemistry
Visualization
Volcanology
Water resources management
X-ray physics

HDF5 Summary

With HDF5, we have designed and developed a portable data format and library that for the first time addresses simultaneously several challenges of today's data storage, management, exchange, and archiving needs. This includes such issues as growing volumes of data, strong demand for high performance I/O, diversity of computational environments and storage media, and growing complexity of data.

Unlike other scientific formats and libraries, HDF5 does not limit the sizes of files or the number and sizes of objects in the files. HDF5 files of any size may be created on 32-bit systems and be processed on 64-bit systems and vice versa. The combination of the simple, powerful and flexible data model, and the rich unlimited collection of HDF5 datatypes, allows HDF5 to easily and efficiently accommodate data of high complexity.

Unlike other libraries, the HDF5 virtual file layer (VFL) is designed and implemented to support different types of I/O, file systems, and storage media. HDF5 provides a flexible I/O pipeline that can be fully customized by HDF5 application developers. The library has many tuning knobs to achieve maximum I/O performance. We were the first to implement a scientific data format and library that works in parallel computing environments using MPI I/O.

Though HDF5 is the newest of the major scientific data management libraries, it has already been adopted by scientists in several scientific research communities, several major research projects rely on it, and several major software applications support it. The HDF5 group is committed to supporting this quickly growing number of HDF5 library users and HDF5-based applications.