COLLEGE OF ENGINEERING
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES
UNIVERSITY OF MICHIGAN

Lecture 19
Data Library Formats

Prof. Brendan Kochunas
11/11/2019

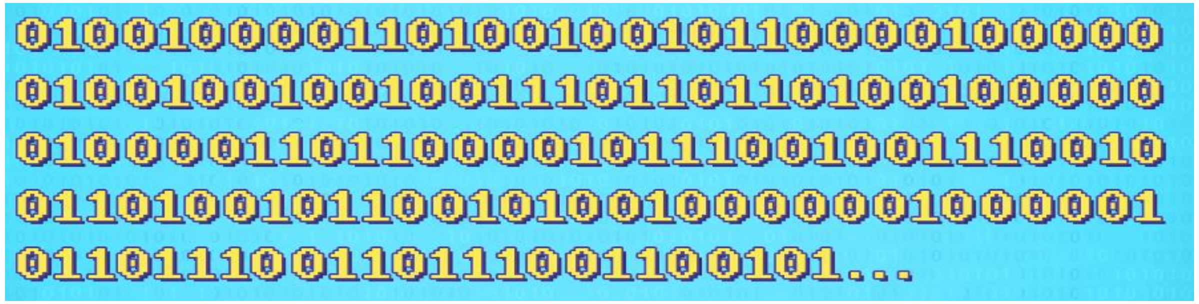
NERS 590-004

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

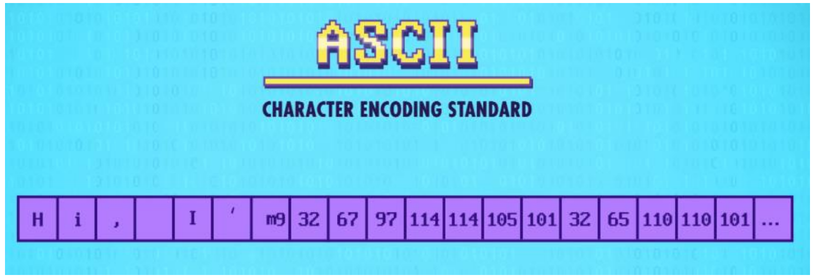
Today’s Learning Objectives

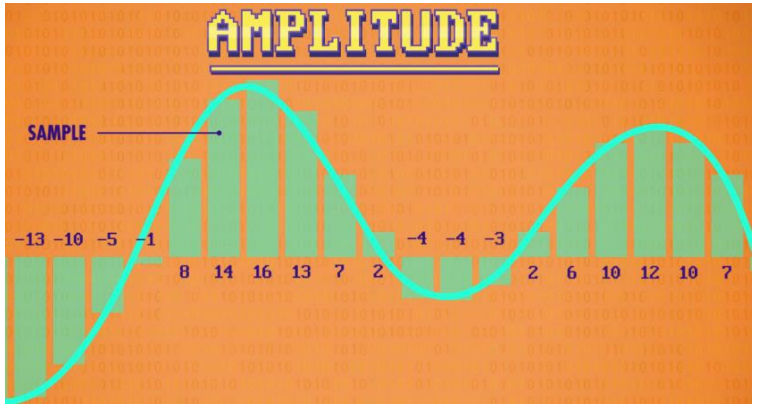
- Understand what formats are out there
- What formats are better for certain things
- Where to go get more information
- Basic information about each format
  - How they are structured
  - How they are written and read
  - What languages having bindings to the API
  - Where to get the libraries

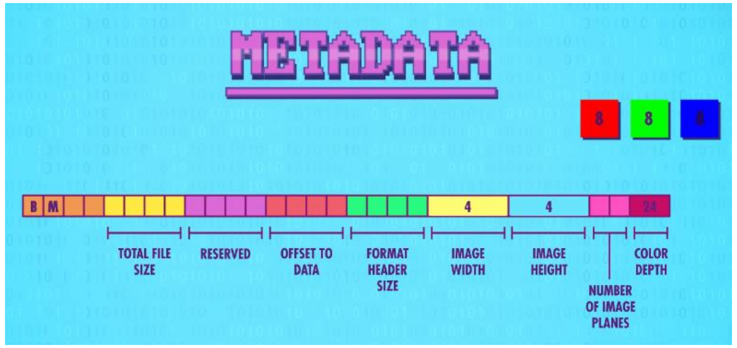
What is a “file” and how are they stored?



```
01001000011010010010110000100000
01001001001001110110110100100000
01000011011000010111001001110010
01101001011001010010000001000001
011011100110111001100101...
```







Ref: https://www.youtube.com/watch?v=KN8YgJnShPM

Why should I care about file formats?

- In your research you will spend a lot of time generating data.
    - You will probably also spend a decent amount of time analyzing the data.
- Doing data analysis is often effort intensive
    - Your analysis is likely unique or has some unique aspects
    - Its probable there is not an analysis tool that already exists that does exactly what you need
        - You're lucky if there is!
    - Complete general data analysis tools kind of exist, but take some effort to use
- Data analysis should not be harder than it needs to be
    - There are requirements to using existing tools
        - Typically this is the format of your data
- Having your data in the "right" format can make your life (as an analyst) easier

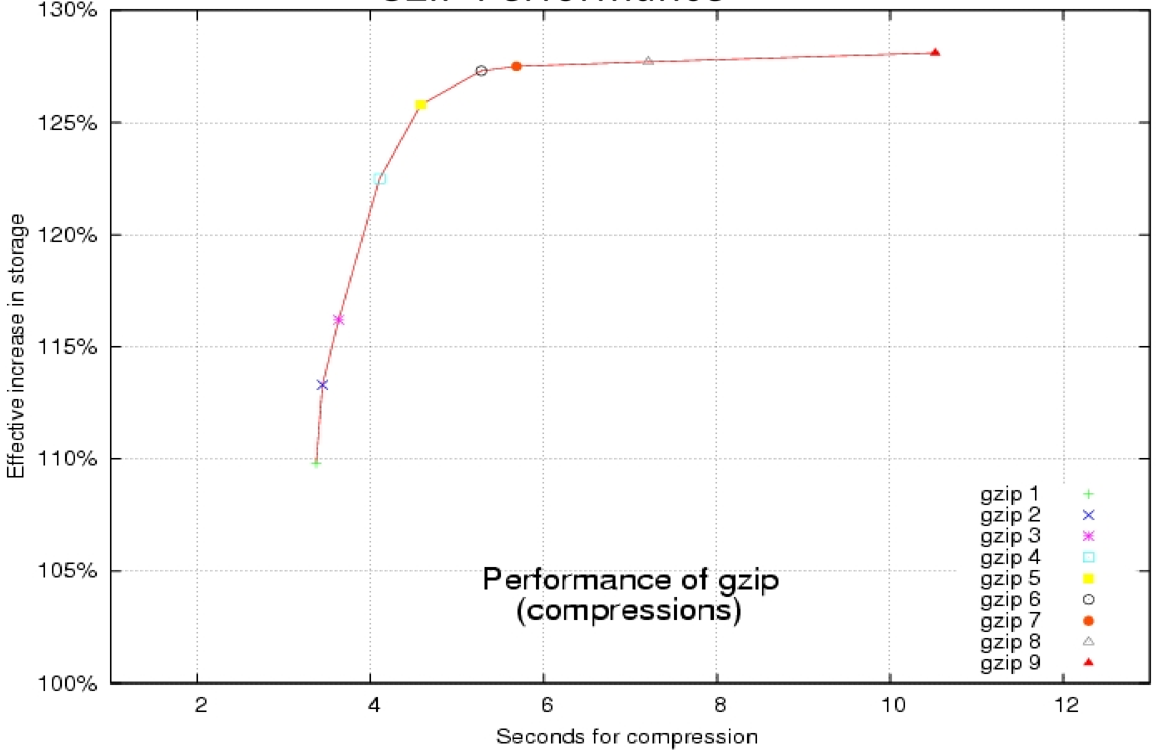
What do we desire from a “good” file format?

- Should be portable
- Should be easy to inspect
- Should be as small as possible
- Easy to describe our data sets
- Easy to extract data into other programs for analysis
- Data can be written or read (relatively) fast
- Longevity and archive-ability
- Preserve numerical precision (binary values in memory)

Example: how data compression affects performance

- A quick tangent on compression
- Several open algorithms for compression
  - zip
  - gzip (DEFLATE)
  - SuperZip
  - bzip2 (better zip 2)

GZIP Performance



COLLEGE OF ENGINEERING
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES
UNIVERSITY OF MICHIGAN

Data Libraries

- A collection of numerical and/or geospatial data sets for use in research.
- Include all the good parts of different file formats.
- Abstract away details of file layouts
- Provide standard, portable file formats
- include metadata (data about data, headers) describing contents

Formats

Outline

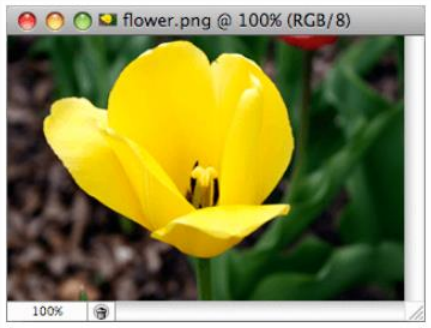
- Plain Text Formats
  - XML
  - XDF
  - JSON
  - YAML
- Binary Formats
  - netCDF/PnetCDF
  - HDF5
  - SILO

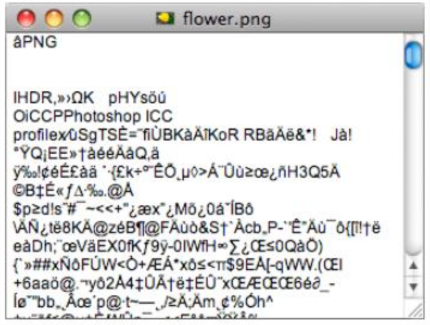
File Formats: Main Types

They encode data differently

Plain Text

- Only contain textual data.
- Less likely to become corrupted.
- A typical plain text file contains several lines of text, each followed by an end-of-line character.
- Uses a simple, standard format.





```
âPNG


IHDR,»›ΩK   pHYsöû
OiCCPPhotoshop ICC
profilex⁄ûSgTSÈ=˜ﬁÙBKàÄîKoR RBäÄë&*!   Jà!
*ŸQ¡EE»†àéèÄâQ,ä
ÿ‰!¢éÉ£àä ˙·{£k+°˜ÊÕ¸µ◊>Á¨Ûù≥œ¿ñH3Q5Ä
©B‡É«ƒ∆·‰.@Å
$p≥d!s˜#¯~<<+"¿æx"¿Mõ¿0á˜ÎBô
\ÄÑ¿të8KÄ@zéB¶@FÄùò&S†˙Äcb¸P-˙Ê˜Äù¯ô{[ﬂ!†ë
eàDh;˙œVäEX0fKƒ9ÿ-0IWfiH∞∑¿Œ≤0QàÖ)
{˙»##xÑôFÚW<Ò+ÆÁ*xô≤<π$9EÅ[-qWW.(ŒI
+6aaö@.¬yô2Å4‡ÛÄ†ë‡ÉÛ¨xŒÆŒŒ6é∂_-
Íø˜"bb„¸Åœ˙p@·t~—¸/≥Ä;Äm¸¢%Óh^
```

Binary

- Typically contain a sequence of bytes, or ordered groupings of eight bits.
- May include multiple types of data in the same file, such as image, video and audio.
- Data can be interpreted by supporting programs, but will not show up “readable” in a text editor.
- Often contain headers, which identifies the file’s contents.

File Formats: Main Types

Plain Text

- Easily readable
- Archivable
- Portable
- Does not preserve precision/binary values well
- Relatively large
- Good compression

Binary

- Not readable
- Less portable
- Preserves *exact* values
- Usually smaller
- Faster access
- Usually not well compressed
- Less archivable

Plain Text Formats

XML, XDF, JSON, YAML

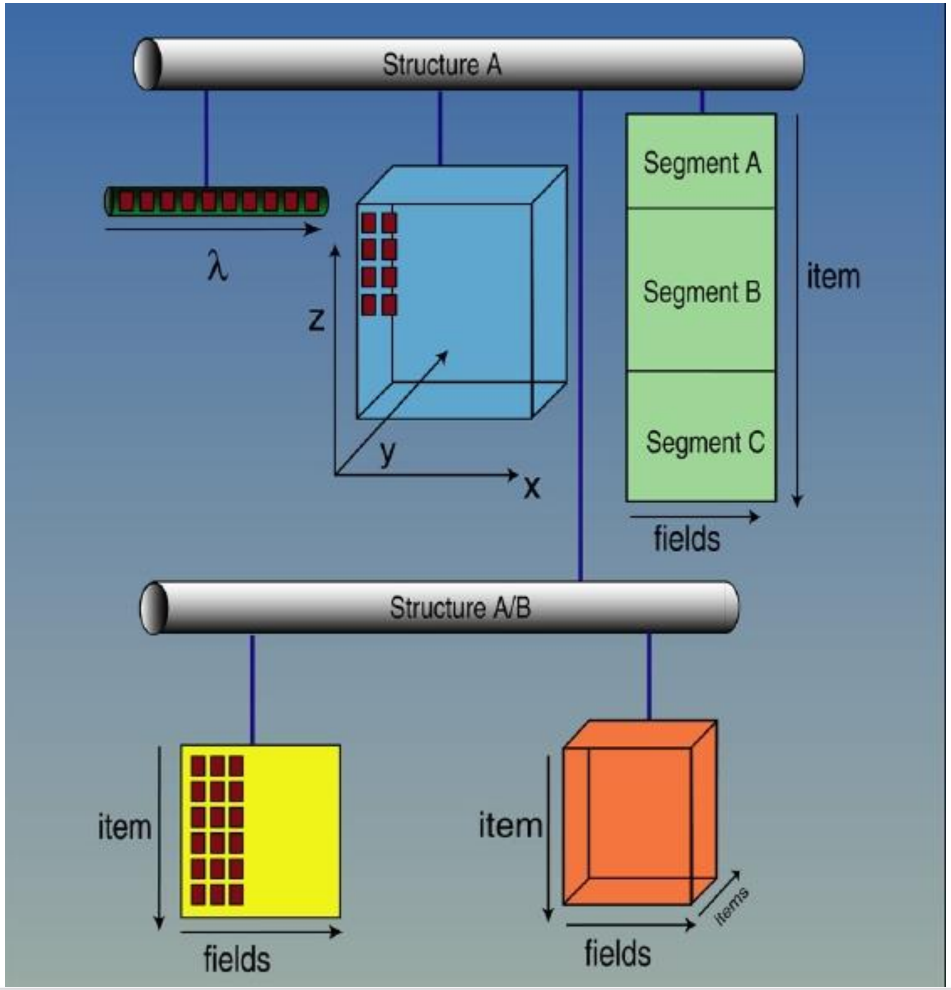
eXtensible Markup Language (XML)

- Originally developed with the world wide web
  - Not scientists in mind
  - Similar to HTML
  - Designed to store and transport data
- Structure built around tags
  - each tag needs a beginning and end marker
  - tags have attributes and contents
  - no predefined tags
- Lots of libraries and tools available for XML
- Some binary implementations, but nothing standardized
- Plain text, but not always easily readable by a person

```
<person>
 <firstName>John</firstName>
 <lastName>Smith</lastName>
 <age>25</age>
 <address>
   <streetAddress>21 2nd Street</streetAddress>
   <city>New York</city>
   <state>NY</state>
   <postalCode>10021</postalCode>
 </address>
 <phoneNumber>
   <type>home</type>
   <number>212 555-1234</number>
 </phoneNumber>
 <phoneNumber>
   <type>fax</type>
   <number>646 555-4567</number>
 </phoneNumber>
</person>
```

eXtensible Data Format (XDF)

- General Science data format
- Maintained by Astronomical Data Center
- http://archive.astro.umd.edu/XDF/
- Perl and Java API's
  - Can probably use xml libraries for compiled languages
- Not widely used outside astrophysics



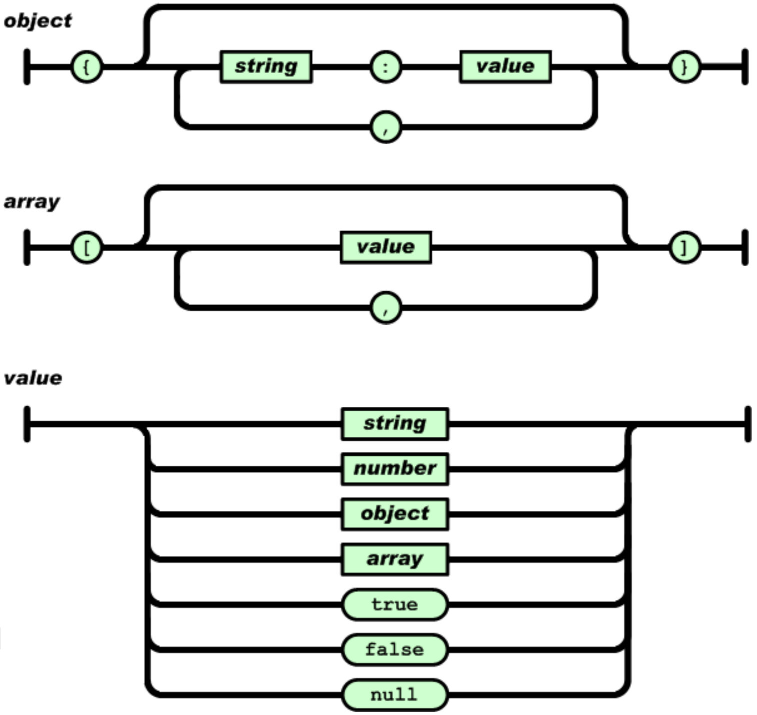
eXtensible Data Model and Format (XDMF)

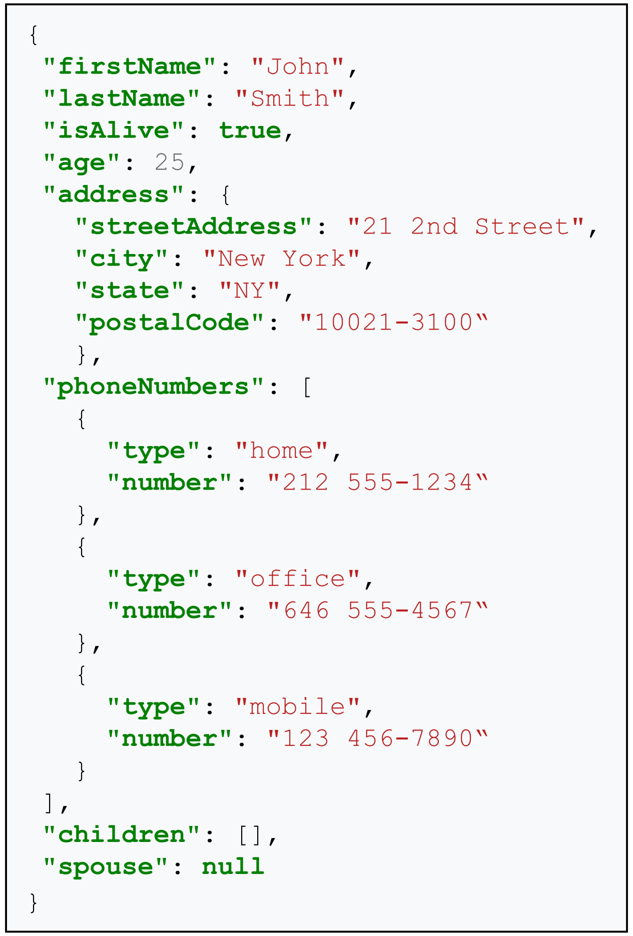
- Standardized method to exchange data between HPC codes and tools
- Categorizes data by two main attributes: size and function
- In addition to raw values, data can refer to format (rank and dimensions of an array), or model
- Three major components
    - elements
    - entities
    - processing
- http://www.xdmf.org/index.php/XDMF_Model_and_Format

```
<DataItem ItemType="HyperSlab"
          Dimensions="25 50 75 3"
          Type="HyperSlab">
    <DataItem Dimensions="3 4"
              Format="XML">
        0 0 0 0
        2 2 2 1
        25 50 75 3
    </DataItem>
    <DataItem
        Name="Points"
        Dimensions="100 200 300 3"
        Format="HDF">
        MyData.h5:/XYZ
    </DataItem>
</DataItem>
```

JavaScript Object Notation (JSON)

- Used to transmit data objects consisting of key-value pairs
- Developed for web applications
  - Not scientific computing
  - Needed for server-to-browser communication
- Allows one to “serialize” an object (like a C-struct)





```
{
 "firstName": "John",
 "lastName": "Smith",
 "isAlive": true,
 "age": 25,
 "address": {
   "streetAddress": "21 2nd Street",
   "city": "New York",
   "state": "NY",
   "postalCode": "10021-3100“
   },
 "phoneNumbers": [
   {
     "type": "home",
     "number": "212 555-1234“
   },
   {
     "type": "office",
     "number": "646 555-4567“
   },
   {
     "type": "mobile",
     "number": "123 456-7890“
   }
 ],
 "children": [],
 "spouse": null
}
```

Yet Another Markup Language (YAML)

YAML Ain’t Markup Language

- Competes for same applications as XML
    - simplifies notation
- Superset of JSON
- Lots of libraries interfaces to languages
    - C/C++, Ruby, Java, Python, Perl, and others

```
firstName: John
lastName: Smith
age: 25
address:
 streetAddress: 21 2nd Street
 city: New York
 state: NY
 postalCode: '10021‛
phoneNumber:
 - type: home
   number: 212 555-1234
 - type: fax
   number: 646 555-4567
 gender:
   type: male
```

Comparison

```
<person>
 <firstName>John</firstName>
 <lastName>Smith</lastName>
 <age>25</age>
 <address>
   <streetAddress>21 2nd Street</streetAddress>
   <city>New York</city>
   <state>NY</state>
   <postalCode>10021</postalCode>
 </address>
 <phoneNumber>
   <type>home</type>
   <number>212 555-1234</number>
 </phoneNumber>
 <phoneNumber>
   <type>fax</type>
   <number>646 555-4567</number>
 </phoneNumber>
</person>
```

```
{
 "firstName": "John",
 "lastName": "Smith",
 "isAlive": true,
 "age": 25,
 "address": {
   "streetAddress": "21 2nd Street",
   "city": "New York",
   "state": "NY",
   "postalCode": "10021-3100”
   },
 "phoneNumbers": [
   {
     "type": "home",
     "number": "212 555-1234”
   },
   {
     "type": "office",
     "number": "646 555-4567”
   },
   {
     "type": "mobile",
     "number": "123 456-7890”
   }
 ],
 "children": [],
 "spouse": null
}
```

```
firstName: John
lastName: Smith
age: 25
address:
 streetAddress: 21 2nd Street
 city: New York
 state: NY
 postalCode: '10021’
phoneNumber:
 - type: home
   number: 212 555-1234
 - type: fax
   number: 646 555-4567
 gender:
   type: male
```

Binary Formats

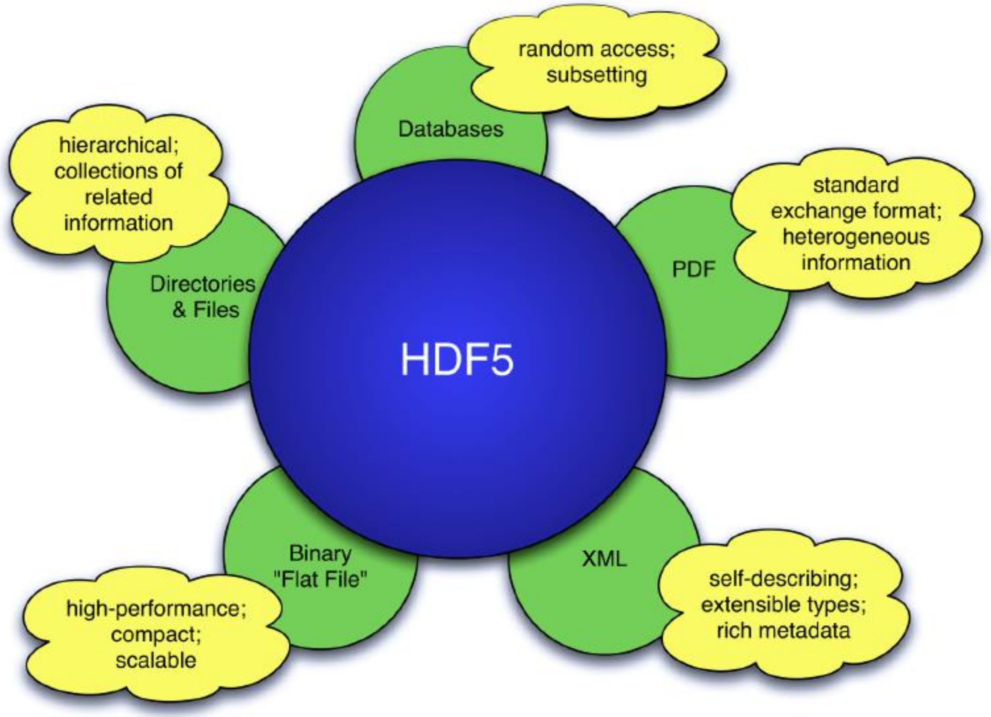
HDF5, netCDF, and SILO

Hierarchical Data Format, version 5 (HDF5)

What is it?

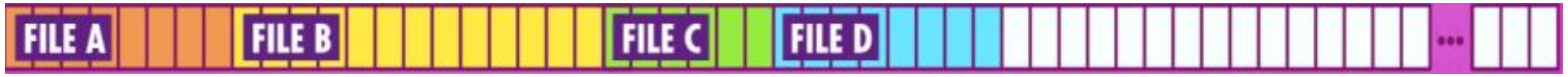
- Open file format
- Open source software
- Designed for:
  - high volume and/or complex data
  - every size/type of system (portable)
  - efficient storage and I/O
  - support long-term data preservation
- Fundamentally array based
- https://www.hdfgroup.org/

HDF5 is like…

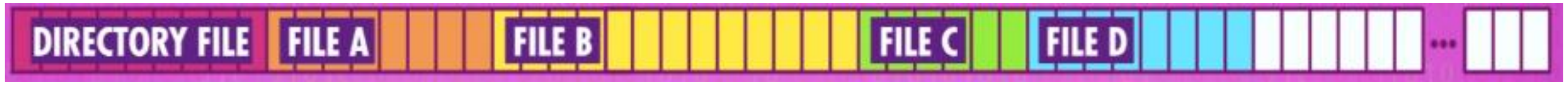

random access; subsetting
Databases
hierarchical; collections of related information
Directories & Files
standard exchange format; heterogeneous information
PDF
HDF5
Binary "Flat File"
high-performance; compact; scalable
XML
self-describing; extensible types; rich metadata

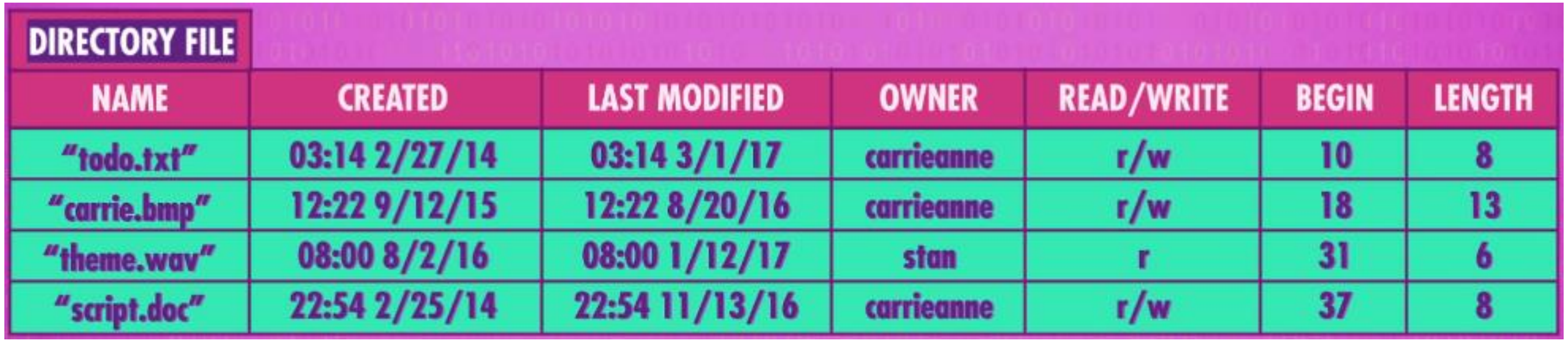
Background: hierarchical file system

- To store more than one file at a time



- To tell the computer where the files begin and end
- You need directory file – kept right at the front of storage, location 0

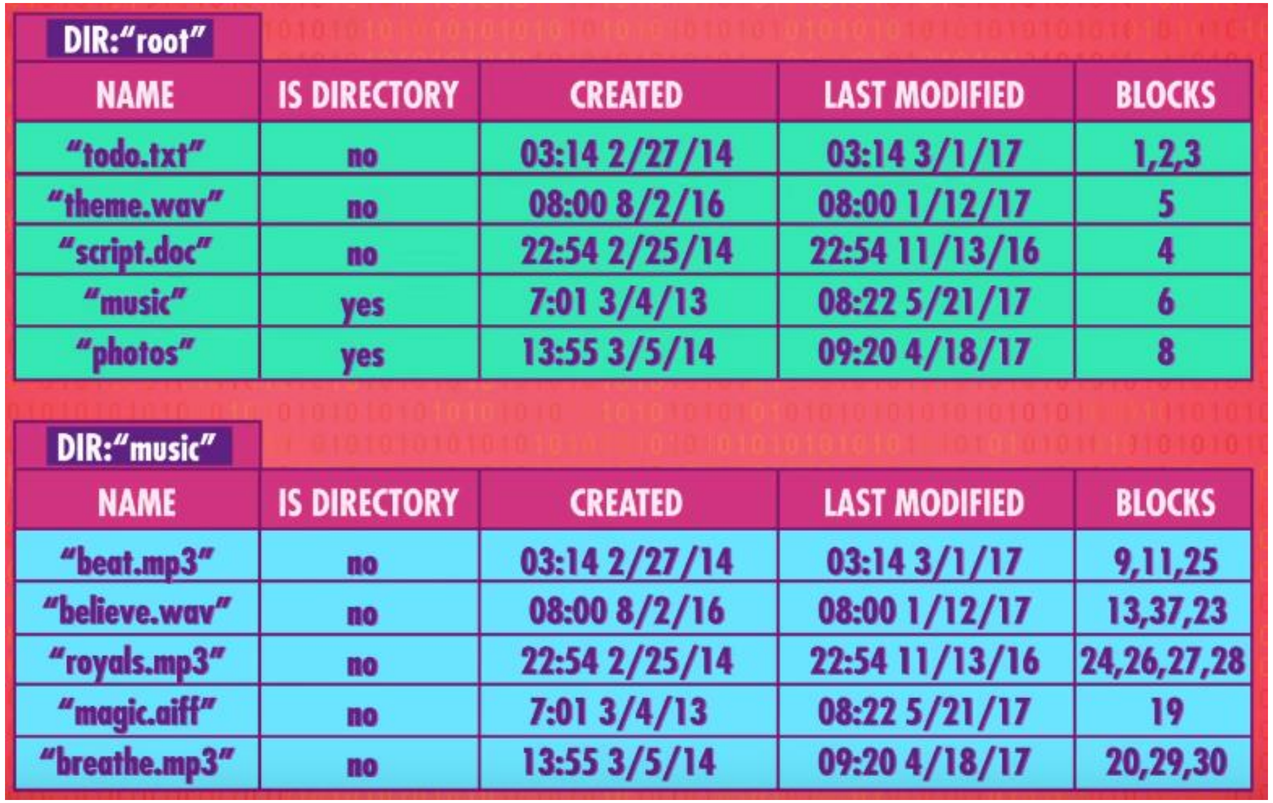




DIRECTORY FILE

NAME	CREATED	LAST MODIFIED	OWNER	READ/WRITE	BEGIN	LENGTH
“todo.txt”	03:14 2/27/14	03:14 3/1/17	carrieanne	r/w	10	8
“carrie.bmp”	12:22 9/12/15	12:22 8/20/16	carrieanne	r/w	18	13
“theme.wav”	08:00 8/2/16	08:00 1/12/17	stan	r	31	6
“script.doc”	22:54 2/25/14	22:54 11/13/16	carrieanne	r/w	37	8

Hierarchical file system



DIR:"root"

NAME	IS DIRECTORY	CREATED	LAST MODIFIED	BLOCKS
"todo.txt"	no	03:14 2/27/14	03:14 3/1/17	1,2,3
"theme.wav"	no	08:00 8/2/16	08:00 1/12/17	5
"script.doc"	no	22:54 2/25/14	22:54 11/13/16	4
"music"	yes	7:01 3/4/13	08:22 5/21/17	6
"photos"	yes	13:55 3/5/14	09:20 4/18/17	8

DIR:"music"

NAME	IS DIRECTORY	CREATED	LAST MODIFIED	BLOCKS
"beat.mp3"	no	03:14 2/27/14	03:14 3/1/17	9,11,25
"believe.wav"	no	08:00 8/2/16	08:00 1/12/17	13,37,23
"royals.mp3"	no	22:54 2/25/14	22:54 11/13/16	24,26,27,28
"magic.aiff"	no	7:01 3/4/13	08:22 5/21/17	19
"breathe.mp3"	no	13:55 3/5/14	09:20 4/18/17	20,29,30

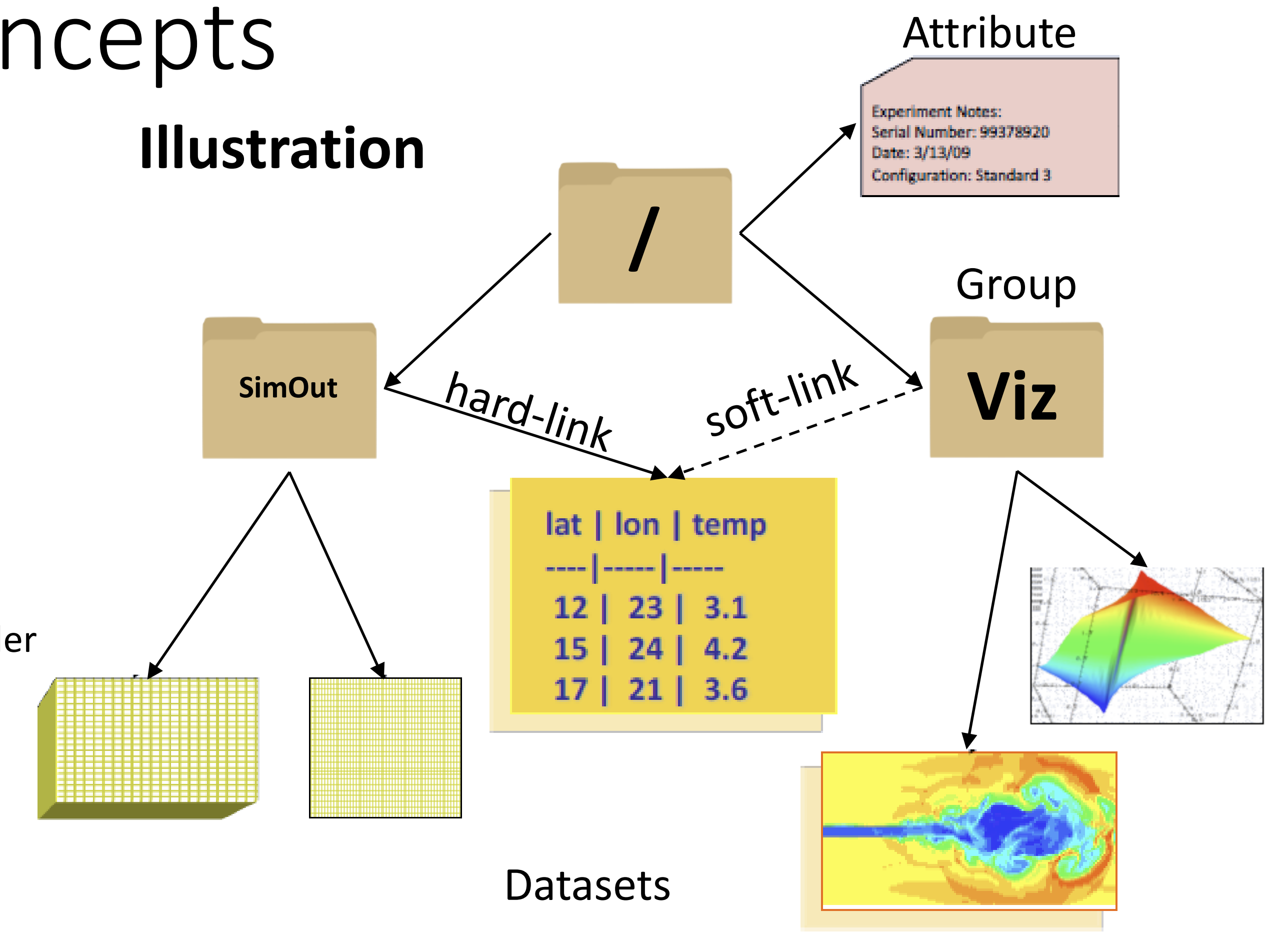
"A file contains a file system": HDF5

HDF Data Models & Concepts

Data Model

- Datatype
  - Like a basic variable (e.g. an int)
  - Describes a single data element
- Dataspace
  - Describes how data elements are laid out.
  - e.g. is this an array? what are dimensions
- Dataset
  - Organize and contain data elements (of a single data type with a dataspace)
- **Group:** Collection of datasets and other groups (like a folder on a file system)
- **Attribute:** Contain metadata and can be associated with other model "objects"
- **File:** Collection of groups & datasets
- **Link:** Like a shortcut

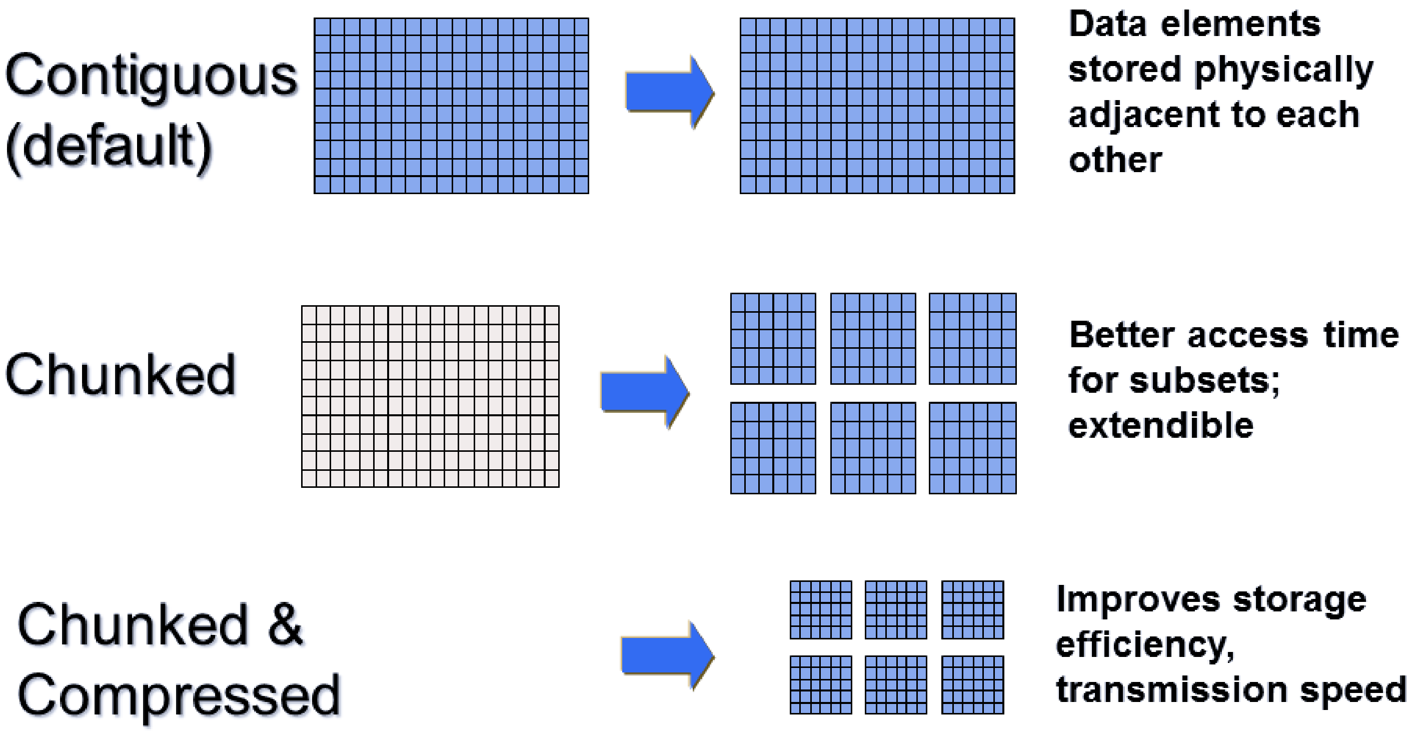
Illustration



Experiment Notes:
Serial Number: 99378920
Date: 3/13/09
Configuration: Standard 3

```
lat | lon | temp
----|-----|-----
 12 |  23 |  3.1
 15 |  24 |  4.2
 17 |  21 |  3.6
```

Dataspace options



Contiguous (default)

Data elements stored physically adjacent to each other

Chunked

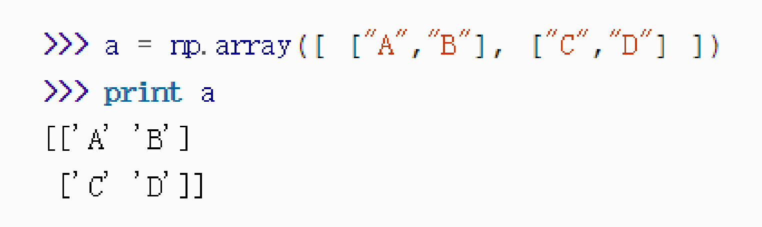
Better access time for subsets; extendible

Chunked & Compressed

Improves storage efficiency, transmission speed

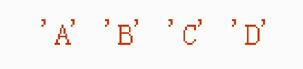
How chunking and compression can help you

- Tied up in the details of how data is arranged on disk
- Think about how multidimensional arrays are actually handled



```
>>> a = np.array([ ["A","B"], ["C","D"] ])
>>> print a
[['A' 'B']
 ['C' 'D']]
```

Mathematically, 2D object



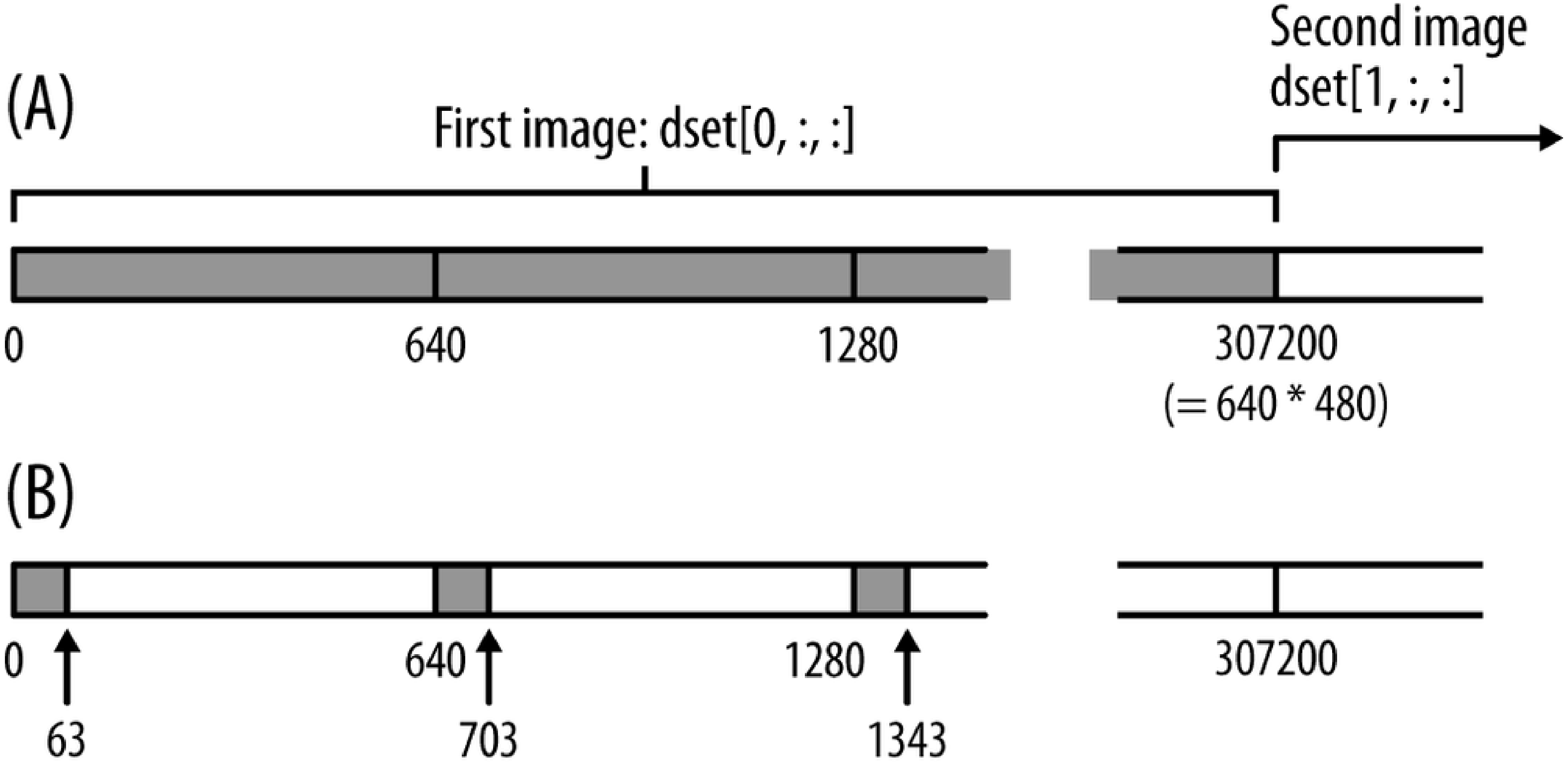
```
'A' 'B' 'C' 'D'
```

Stored in 1D buffer

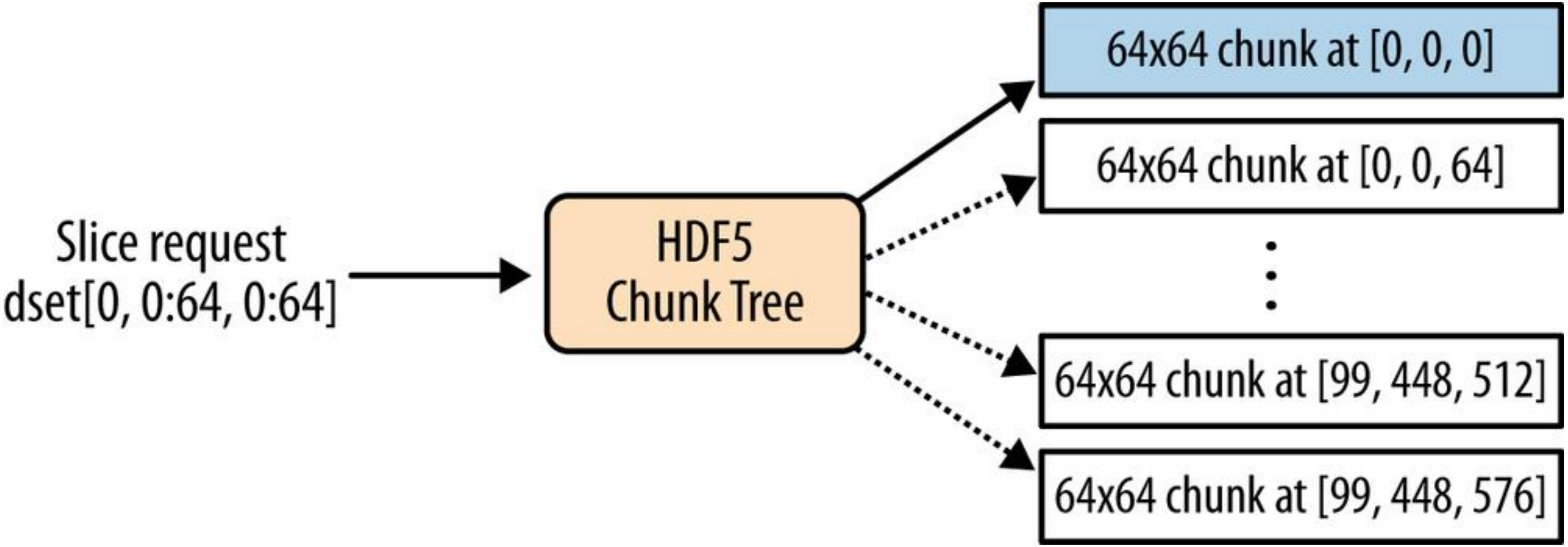
- The first rule (really, the only one) for dealing with data on disk, *locality*: reads are generally faster when the data being accessed is all stored together.

Storage mechanism should match your access pattern

- Consider as an example a dataset containing one hundred 640×480 grayscale images.
- Let’s say the shape of the dataset is (100, 480, 640)

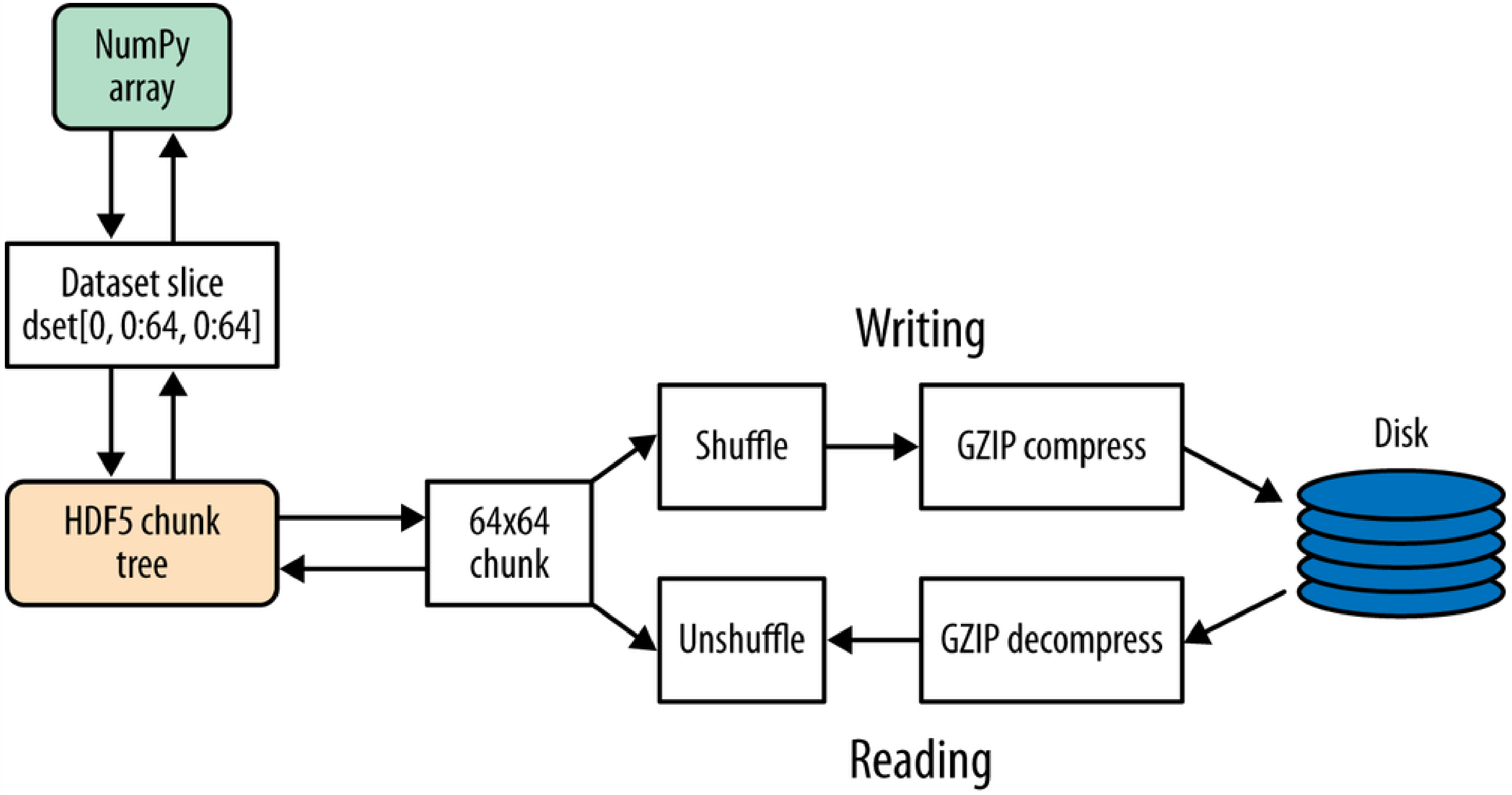


HDF chunk tree


Slice request
dset[0, 0:64, 0:64]
HDF5
Chunk Tree
64x64 chunk at [0, 0, 0]
64x64 chunk at [0, 0, 64]
⋮
64x64 chunk at [99, 448, 512]
64x64 chunk at [99, 448, 576]

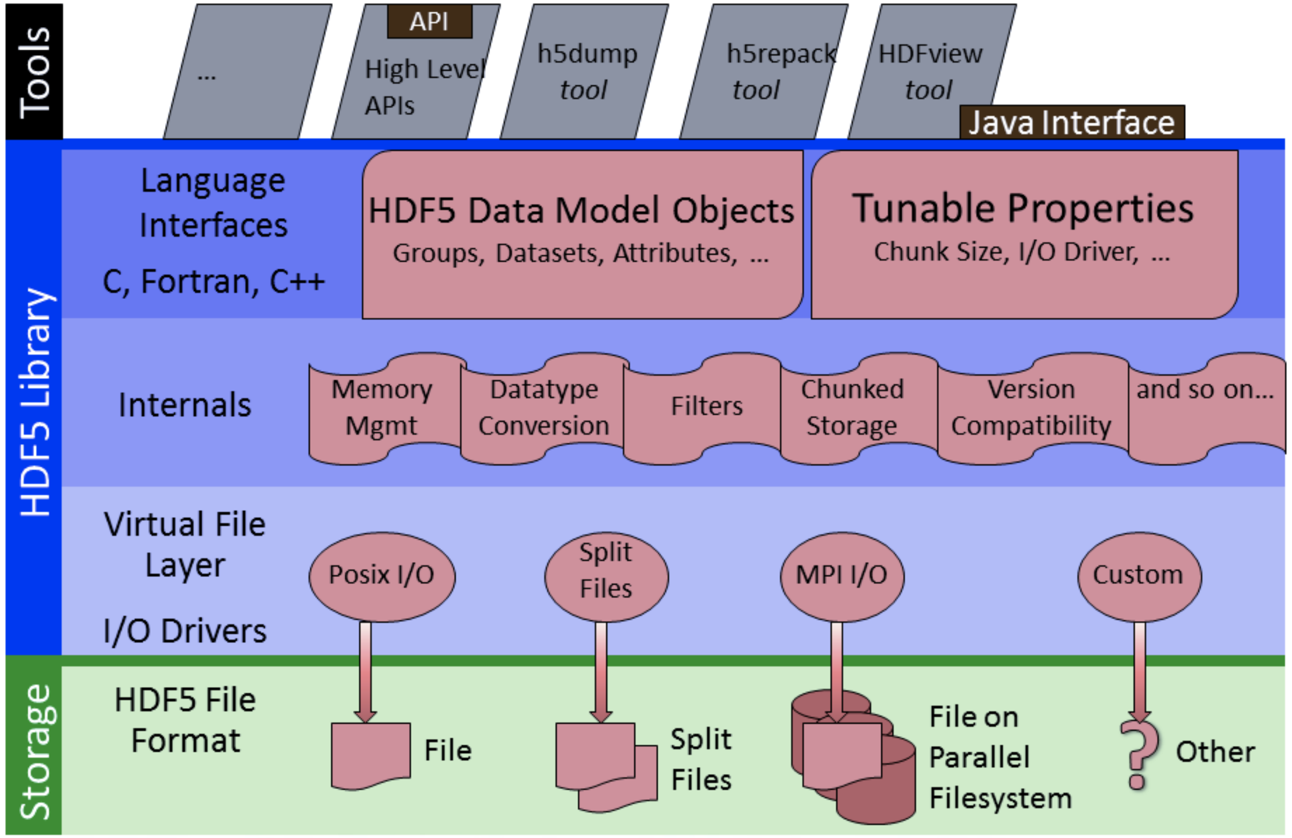
Users can choose the chunk size that they preferred or use auto chunk.

The filter pipeline



- Each *filter* is free to do anything it wants to the data in the chunk: compress it, checksum it, add metadata, etc.
- When the file is read, each filter is run in “reverse” mode to reconstruct the original data.
- You have to specify your filters when the dataset is created.

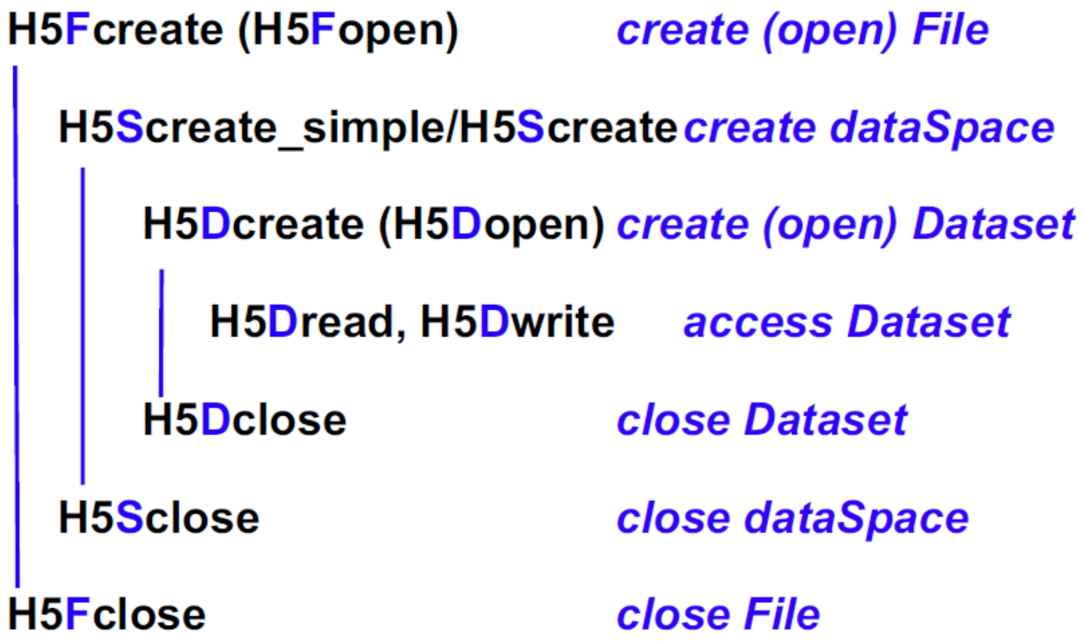
HDF5 Software Layers & Storage



The HDF5 API

- HDF5 library implemented in C
  - Native API is C interface
  - Library also provides Fortran and C++ interfaces
- Other interfaces readily available
  - Python: h5py
  - Java: JHI5
  - MATLAB
  - Microsoft .NET
- Not all C interfaces are exposed in other langugе API’s

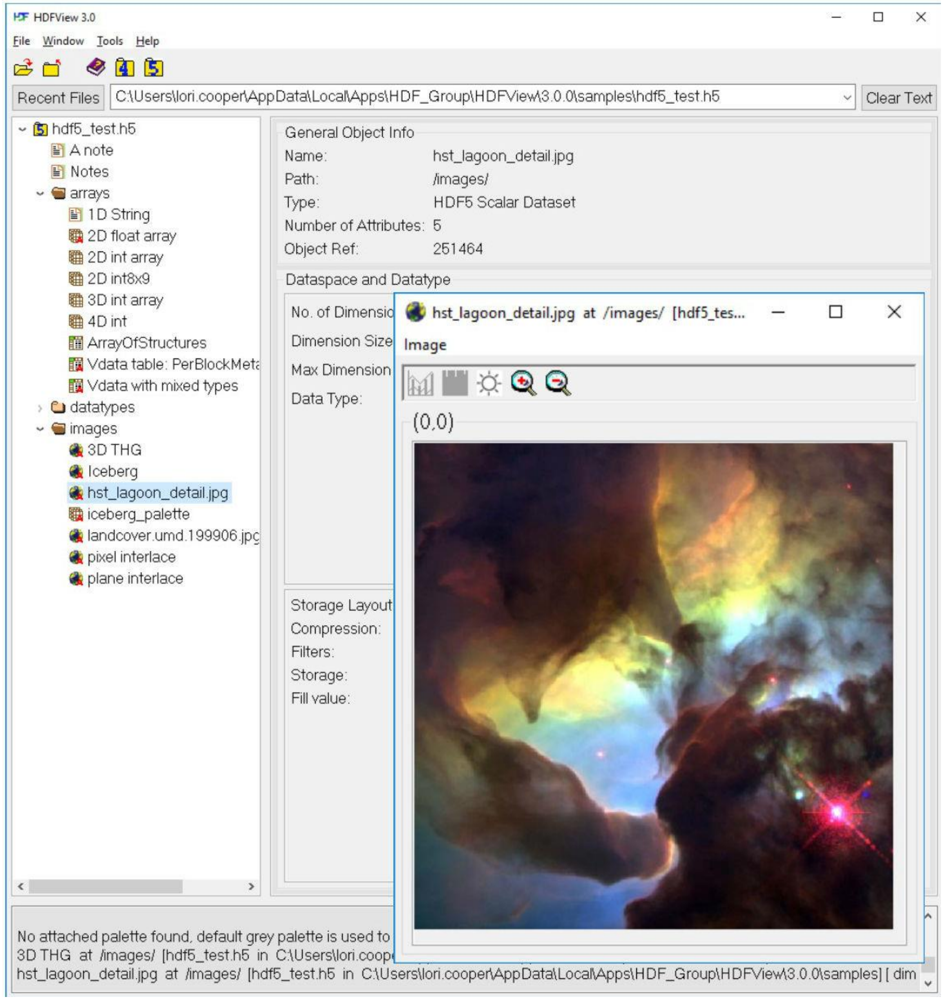
Basics calls for creating and writing/reading data to/from file



H5Fcreate (H5Fopen)	*create (open) File*
H5Screate_simple/H5Screate	*create dataSpace*
H5Dcreate (H5Dopen)	*create (open) Dataset*
H5Dread, H5Dwrite	*access Dataset*
H5Dclose	*close Dataset*
H5Sclose	*close dataSpace*
H5Fclose	*close File*

HDF5 Tools and Software

- HDFView
  - Java program for viewing files
- Compiler wrappers
  - h5cc, h5c++, h5fc
- Command line tools
  - h5dump – output file to text
  - h5repack – compress and change data layouts
  - h5ls – list contents of files
  - h5copy – copy parts of files to other files



C:\Users\lori.cooper\AppData\Local\Apps\HDF_Group\HDFView\3.0.0\samples\hdf5_test.h5

Name: hst_lagoon_detail.jpg
Path: /images/
Type: HDF5 Scalar Dataset
Number of Attributes: 5
Object Ref: 251464

```
No attached palette found, default grey palette is used to
3D THG  at  /images/  [hdf5_test.h5  in  C:\Users\lori.coop
hst_lagoon_detail.jpg  at  /images/  [hdf5_test.h5  in  C:\Users\lori.cooper\AppData\Local\Apps\HDF_Group\HDFView\3.0.0\samples] [ dim
```

Network Common Data Format (netCDF)

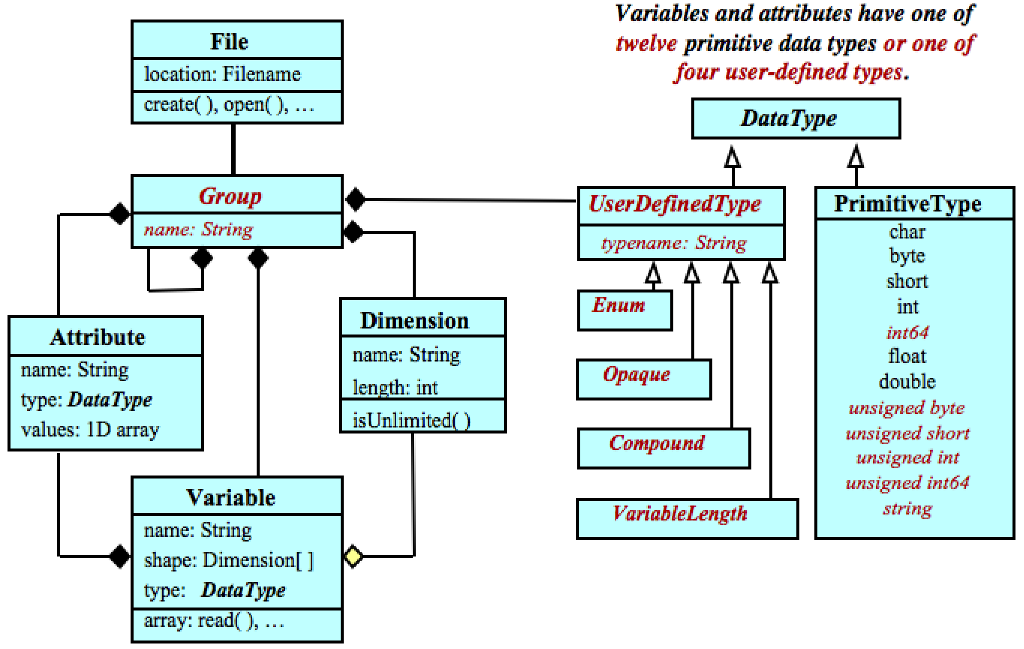
- Purpose: create, access, and share array oriented data
  - Self-describing
  - Portable
- Written in C
  - Native API is C
  - Library has API's for Fortran, C++
  - Also a Java and Python API
- Interoperable with HDF5 v1.8.x series
  - Links to HDF5 "on back end"
- https://www.unidata.ucar.edu/software/netcdf/

- Parallel-netCDF derivative also exists
  - https://trac.mcs.anl.gov/projects/parallel-netcdf

netCDF Data Model

- Very similar to HDF5 data model
- Hierarchical with groups
- User-definable types
- Support for attributes/metadata
- Dimension info about logical layout of data

UML Diagram of NetCDF Datamodel

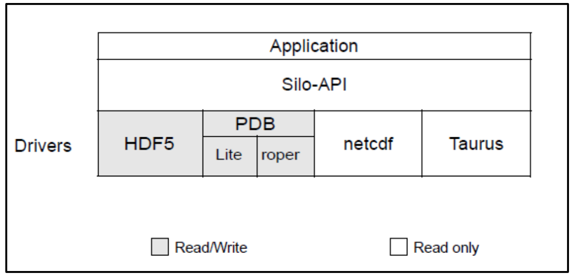


Variables and attributes have one of twelve primitive data types or one of four user-defined types.

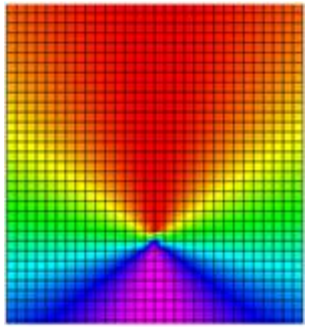
A file has a top-level unnamed group. Each group may contain one or more named subgroups, user-defined types, variables, dimensions, and attributes. Variables also have attributes. Variables may share dimensions, indicating a common grid. One or more dimensions may be of unlimited length.

SILO

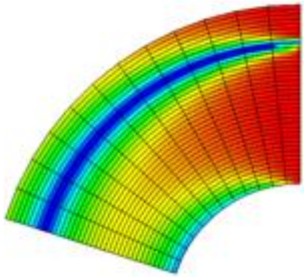
- Mesh and Field library that builds on top of HDF5 and other I/O libraries

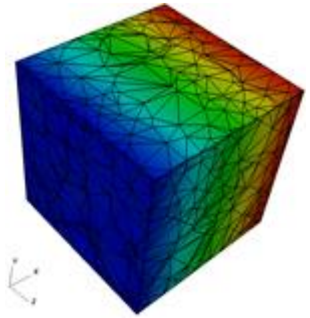


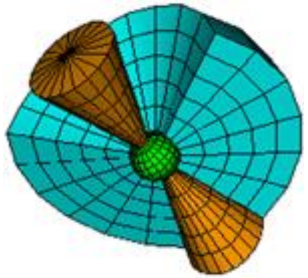
- Primarily processed by visualization tools Paraview and VisIt
- API in C and Fortran
  - experimental support for JSON
- https://wci.llnl.gov/simulation/computer-codes/silo


Structured Rectilinear

Constructive Solid Geometry


Curvilinear


Arbitrary Polyhedral


Unstructured