A Quick Intro to using HDF5

- **What is HDF5 (Hierarchical Data Format)?**

- **FUTILS: A Module of F90 routines built on HDF5**
  - **Diagnostic files for ORB5 (first motivation)**
  - **Restart files (parallel IO with MPI-IO)**
  - **Simple to use (specialized for some types of simulations similar to ORB5)**

- **Access/Visualization of HDF5 files:**
  - **NCSA tools: h5ls, h5dump, hdfview**
  - **2d visualization: Matlab, Python, …**
  - **Others tools: OpenDX, VTK, Paraview, Chombo, …**

What is HDF5?

- File format for storing scientific data
  - To store and organize all kinds of data
  - To share data , to port files from one platform to another

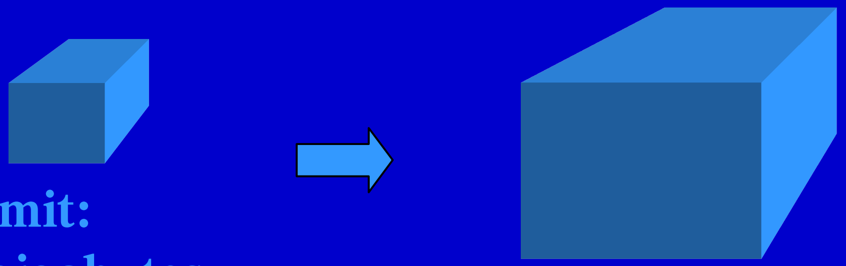
- Software for accessing scientific data
  - Flexible I/O library (parallel, remote, etc.)
  - Efficient storage
  - Available on almost all platforms
  - C, F90, C++ , Java APIs
  - Tools (HDFView, utilities)

Storage Capacity

- Store large objects
- Store large numbers of objects

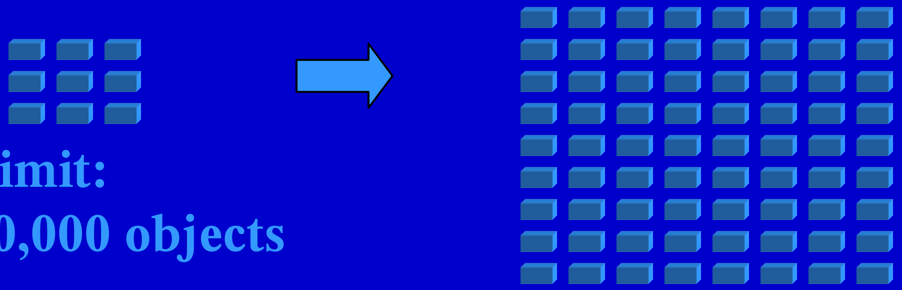
HDF4

HDF5



Limit:
2 gigabytes

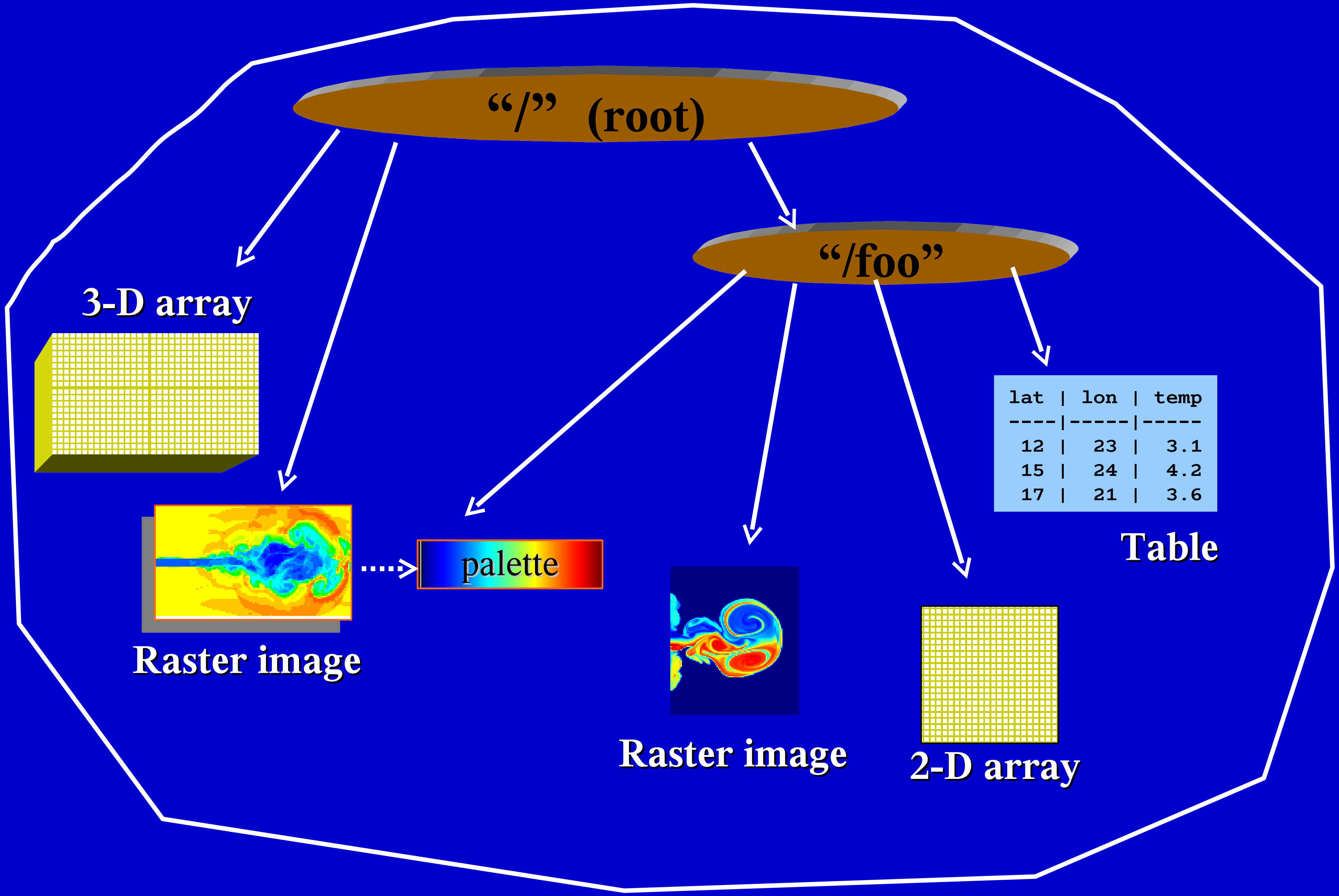
no limit



Limit:
20,000 objects

no limit

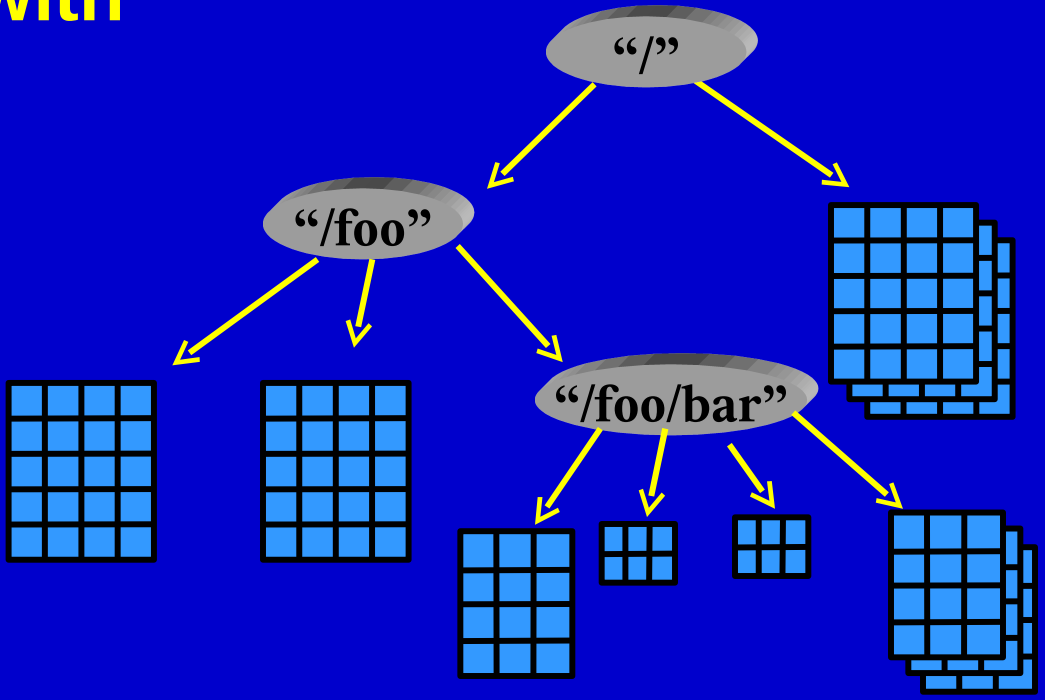
HDF5 file = groups + datasets



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

Groups

- Group structure for organizing the file
- Every file starts with a root group
- Like directories in file system
- Groups have attributes



Dataset Components

Metadata

Dataspace

Rank	Dimensions
3	Dim_1 = 4
	Dim_2 = 5
	Dim_3 = 7

Datatype

IEEE 32-bit float

Storage info

Extendable

Compressed

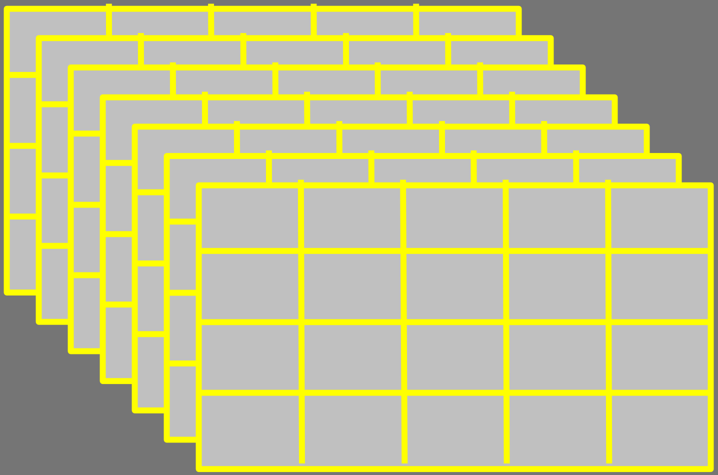
Attributes

Time = 32.4

Pressure = 987

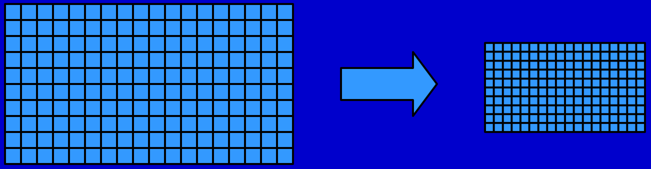
Temp = 56

Data



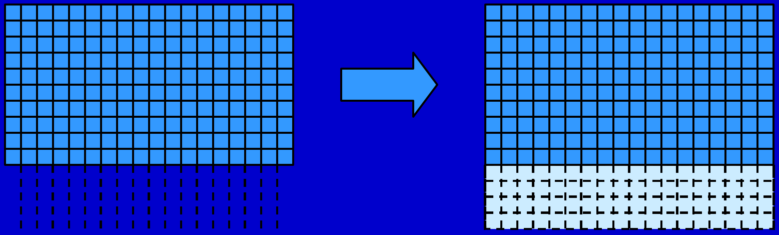
Special Storage Options

- **compressed**



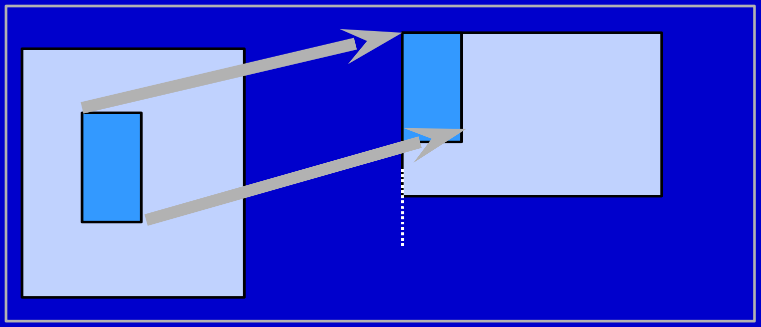
Improves storage efficiency

- **extendable**

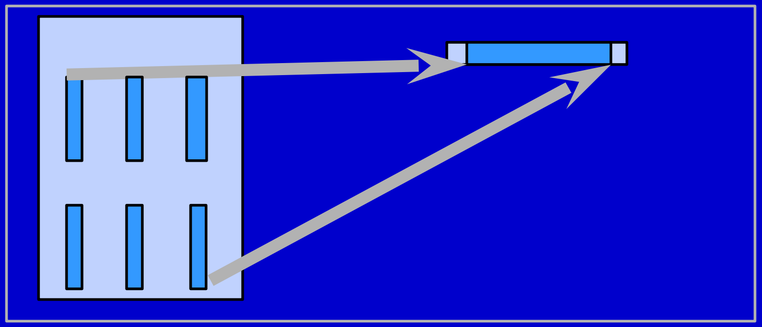


Arrays can be extended individually

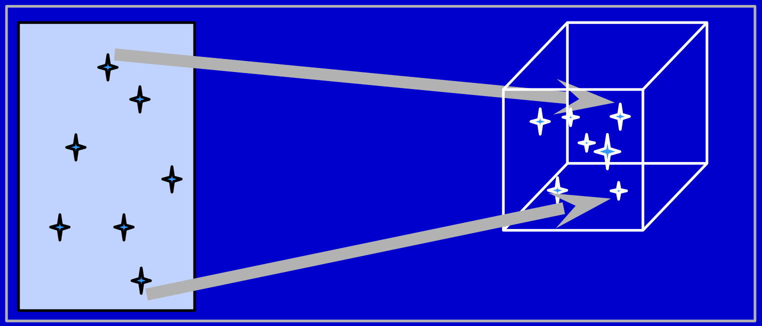
Sample Mappings between File Dataspaces and Memory Dataspaces



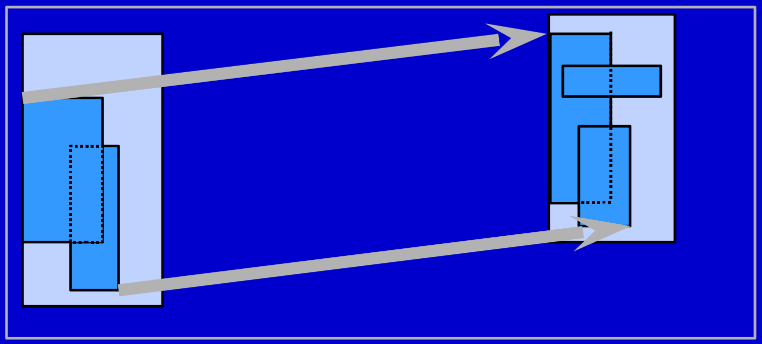
(a) Hyperslab from a 2D array to the corner of a smaller 2D array



(b) 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) Union of hyperslabs in file to union of hyperslabs in memory.

Module FUTILS

- **Double precision real (64 bits) arrays in *memory***
- **32 bit (default) or 64 bit IEEE Float Little Endian on *file***
- **Optional Compression (GZIP)**
- **Fixed dimension arrays**
  - **ARRAY(Nx, Ny, Nz)**
- **Extendable array with unlimited last dimension (for the time)**
  - **ARRAY(Nx, Ny, 1:∞)**
- **Real/Integer attributes attached to groups and datasets**

Subroutines of FUTILS

- Files
  - creatf, openf, closef
- Groups
  - creatg
- Datasets
  - putarr: array(nx, ny, nz) fixed dim.
  - creatd, append: array(nx, ny, …) unlimited last dim.
- Attributes
  - attachg, attachd

Others Tools

- HDF5 distribution: h5ls, h5dump

- Graphical browser: hdfview

- Matlab: hdf5read
  - phi = hdf5read(‘demo.h5’, ‘/profile_2d/phi’)

- Python: pytables + matplotlib (matlab clone)
  - f = openFile(‘demo.h5’, mode=‘r’)
  - phi_id = f.getNode(‘/profile_2d’, ‘phi’)
  - phi = phi_id[:, ::2]

Parallel HDF5

- Use MPI-IO (MPI2 standard) => Portable
  - MPICH2: on top of ROM-IO (NFS, UFS and PVFS2)
  - BG/L MPI (based on MPICH2): GPFS
- All processors can access to the same file.
- Efficiency for a Parallel File System:
  - PVFS2 in PLEIADES2
  - GPFS in BG/L
- Very easy to use with parallel FUTILS

Parallel FUTILS (actual version)

- **1D parallel partition**
  - Only 1 dimension is partitioned (can be any)
  - Examples:
    - ARRAY(Nx, Ny/P) partitioned by columns
    - ARRAY(Nx/P, Ny) partitioned by rows
    - ARRAY(Nx, Ny/P, Nz)

- **User Interface**
  - Same subroutines as in the serial FUTILS
  - *Optional* argument `mpicomm` in CREATF/OPENF to define the MPI Communicator.
  - *Optional* argument `pardim` in PUTARR to indicate which dimension is partitioned.

Conclusions

- Is useable, at least for DIAGNOSTICS in ORB5

- TODO list:
  - Read datasets: RESTART
  - Benchmarking parallel IO
  - Tables: store the input NAMELIST

- Need feedbacks from users: bugs and new features

- SVN repositories:
  - http://crppsvn.epfl.ch/repos/private_tmt/futils/tags/release-1.0/ (serial)
  - http://crppsvn.epfl.ch/repos/private_tmt/futils/tags/release-2.0/ (parallel)