Cornell University
Center for Advanced Computing

Data Formats for Visualization and Interoperability

Steve Lantz
Senior Research Associate

How will you store your data?

- Raw binary is compact but not portable
  - "Unformatted," machine-specific representation
  - Byte-order issues: big endian (IBM) vs. little endian (Intel)
- Formatted text is portable but not compact
  - Need to know all the details of formatting just to read the data
  - 1 byte of ASCII text stores only a single decimal digit (~3 bits)
  - Most of the fat can be knocked out by compression (gzip, bzip, etc.)
  - However, compression is impractical and slow for large files
- Need to consider how data will ultimately be used
  - Are you trying to ensure future portability?
  - Will your favored analysis tools be able to read the data?
  - What storage constraints are there?

Issues beyond the scope of this talk…

- Provenance
  - The record of the origin or source of data
  - The history of the ownership or location of data
  - Purpose: to confirm the time and place of, and perhaps the person responsible for, the creation, production or discovery of the data
- Curation
  - Collecting, cataloging, organizing, and preserving data
- Ontology
  - Assigning types and properties to data objects
  - Determining relationships among data objects
  - Associating concepts and meanings with data (semantics)
- Portable data formats can and do address some of these issues…

Portable data formats: the HDF5 technology suite

- Versatile data model that can represent very complex data objects and a wide variety of metadata
- Completely portable file format with no limit on the number or size of data objects in the collection
- Free and open software library that runs on a range of platforms, from laptops to massively parallel systems, and implements a high-level API with C, C++, Fortran 90, and Java interfaces
- Rich set of integrated performance features that allow for optimizations of access time and storage space
- Tools and applications for managing, manipulating, viewing, and analyzing the data in the collection

Source: www.hdfgroup.org/hdf5

Features lending flexibility to HDF5

- *Datatype definitions* include information such as byte order (endian) and fully describe how the data is stored, insuring portability
- *Virtual file layer* provides extremely flexible storage and transfer capabilities: Standard (Posix), Parallel, and Network I/O file drivers
- *Compression and chunking* are employed to improve access, management, and storage efficiency
- *External raw storage* allows raw data to be shared among HDF5 files and/or applications
- *Datatype transformation* can be performed during I/O operations
- *Complex subsetting* reduces transferred data volume and improves access speed during I/O operations

Source: www.hdfgroup.org/hdf5

Portable data formats: netCDF

- NetCDF (network Common Data Form) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data
- The free and open netCDF distribution contains the C/C++/F77/F90 libraries, plus the netCDF utilities ncgen and ncdump
- NetCDF for Java is also available (standalone)
- Many other interfaces to netCDF data exist: MATLAB, Objective-C, Perl, Python, R, Ruby, Tcl/Tk
- There is a well-developed suite of software tools for manipulating or displaying netCDF data

Source: http://www.unidata.ucar.edu/software/netcdf

Properties of NetCDF data

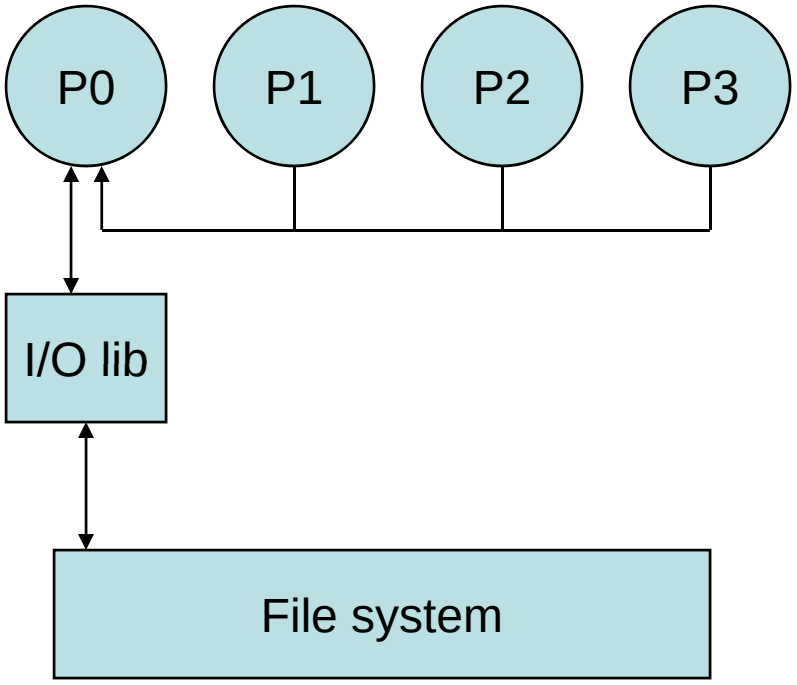
- *Self-Describing*. A netCDF file includes information about the data it contains
- *Portable*. A netCDF file can be accessed by computers with different ways of storing integers, characters, and floats
- *Direct-access*. A small subset of a large dataset may be accessed without first reading through all the preceding data
- *Appendable*. Data may be appended to a properly structured netCDF file without copying the dataset or redefining its structure
- *Shareable*. One writer and multiple readers may simultaneously access the same netCDF file
- *Archivable*. NetCDF will always be backwards compatible

Source: http://www.unidata.ucar.edu/software/netcdf

Advantages of netCDF

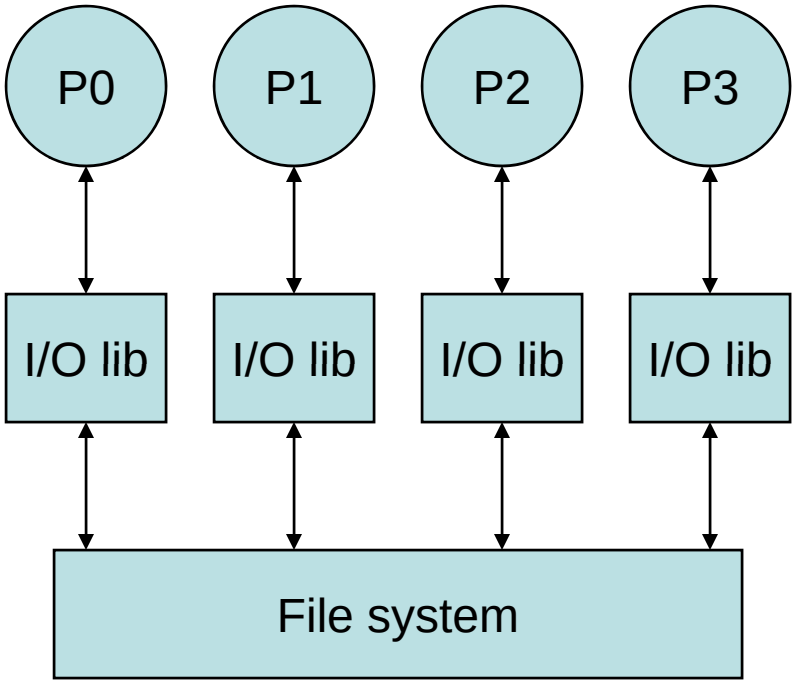
- NetCDF has been around longer, especially in the climate, weather, atmosphere, and ocean research communities (source is UCAR)
- NetCDF has nice associated tools, especially for geo-gridded data
  - *Panoply* (http://www.giss.nasa.gov/tools/panoply/) focuses on the presentation of geo-gridded data. It is written in Java and is platform independent. The feature set overlaps with ncBrowse and ncview.
  - *Ferret* (http://ferret.wrc.noaa.gov/Ferret/) offers a Mathematica-like approach to analysis. New variables and expressions may be defined interactively; calculations may be applied over arbitrarily shaped regions; geophysical formatting is built in.
  - *Parallel-NetCDF* (http://trac.mcs.anl.gov/projects/parallel-netcdf/) is built upon MPI-IO to distribute file reads and writes efficiently among multiple processors.

Path from serial to parallel I/O – part 1



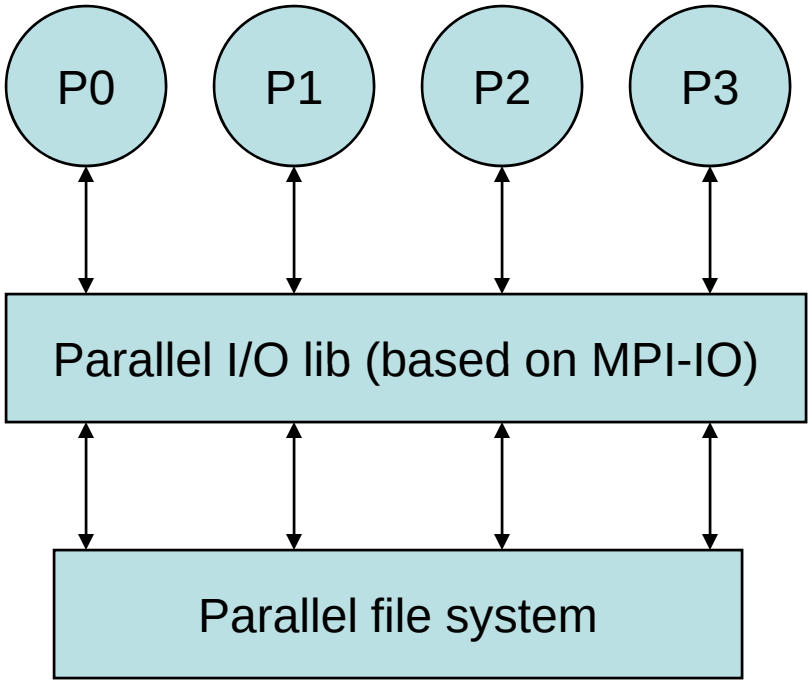
- P0 may become bottleneck
- System memory may be exceeded on P0

Path from serial to parallel I/O – part 2



- Possible to achieve good performance
- May require post-processing
- More work for applications, programmers

Path from serial to parallel I/O – part 3



- Main point: either HDF5 or netCDF can take the place of the parallel I/O library
- Variant: only P1 and P2 act as parallel writers; they gather data from P0 and P3 respectively (chunking)

Data formats for visualization: ParaView

- **ParaView** .pvd
- **VTK** .vtp .vti .vtr .vts .vtu .vtm
- **Parallel (partitioned) VTK** .pvtp .pvti .pvtr .pvts .pvtu
- **Legacy VTK** .vtk .pvtk
- **EnSight** .case .sos
- **MOVIE.BYU** .g
- **XDMF** .xmf
- **PLOT3D** .xy .q
- **SpyPlot** .spcth
- **AVS UCD** .inp
- **UNC Meta Image** .mha .mhd
- **HDF5** .h5 – image data only
- **Digital Elevation Model** .dem
- **Exodus Finite Element** .g .e .ex2 .ex2v2 .exo .gen .exoII
- **SAF, LS-Dyna**
- **Facet, PNG, VRML**
- **Stanford PLY polygonal** .ply
- **Protein Data Bank** .pdb
- **XMol** .xyz
- **Stereo Lithography** .stl
- **Gaussian Cube** .cub
- **Raw (Binary)**

Data formats for **visualization**: ParaView

- **ParaView** .pvd
- **VTK** .vtp .vti .vtr .vts .vtu .vtm
- **Parallel (partitioned) VTK** .pvtp .pvti .pvtr .pvts .pvtu
- **Legacy VTK** .vtk .pvtk
- **EnSight** .case .sos
- **MOVIE.BYU** .g
- **XDMF** .xmf
- **PLOT3D** .xy .q
- **SpyPlot** .spcth
- **AVS UCD** .inp
- **UNC Meta Image** .mha .mhd
- **HDF5** .h5 – image data only
- **Digital Elevation Model** .dem
- **Exodus Finite Element** .g .e .ex2 .ex2v2 .exo .gen .exoII
- **SAF, LS-Dyna**
- **Facet, PNG, VRML** .wrl
- **Stanford PLY polygonal** .ply
- **Protein Data Bank** .pdb
- **XMol** .xyz
- **Stereo Lithography** .stl
- **Gaussian Cube** .cub
- **Raw (Binary)**

Data formats for **interoperability**: ParaView

- **ParaView** .pvd
- **VTK** .vtp .vti .vtr .vts .vtu .vtm
- **Parallel (partitioned) VTK** .pvtp .pvti .pvtr .pvts .pvtu
- **Legacy VTK** .vtk .pvtk
- **EnSight** .case .sos
- **MOVIE.BYU** .g
- **XDMF** .xmf
- **PLOT3D** .xy .q
- **SpyPlot** .spcth
- **AVS UCD** .inp
- **UNC Meta Image** .mha .mhd
- **HDF5** .h5 – image data only
- **Digital Elevation Model** .dem
- **Exodus Finite Element** .g .e .ex2 .ex2v2 .exo .gen .exoII
- **SAF, LS-Dyna**
- **Facet, PNG, VRML**
- **Stanford PLY polygonal** .ply
- **Protein Data Bank** .pdb
- **XMol** .xyz
- **Stereo Lithography** .stl
- **Gaussian Cube** .cub
- **Raw (Binary)**

XDMF: eXtensible Data Model and Format

- Standardized method to exchange scientific data between High Performance Computing codes and tools
- Distinction is made between Light data and Heavy data
  - Light data: the description of the data
  - Heavy data: the values themselves
- Heavy data movement needs to be kept to a minimum
- Due to the different nature of heavy and light data, they are stored using separate mechanisms
  - Light data is stored using XML
  - Heavy data is typically stored using HDF5

Data formats for visualization: VisIt

- In contrast to ParaView, most of the formats readable by VisIt are output by applications (ANSYS, FLUENT, FLASH, ZeusMP, etc.)
- Several general-purpose formats for data exchange can also be read: BOV (brick of values), CGNS, FITS, H5part, ITAPS/TSTT, netCDF (with guesswork), Silo, XDMF
- What about HDF5?
  - Specifications like Silo, H5part, and XDMF are built on top of HDF5
  - HDF5 is so flexible and extensible that it can be hard to parse
  - In the future, even netCDF-4 will be a layer built on top of HDF5(!)
- Recommended formats in the VisIt documentation: Silo, CGNS, netCDF, VTK (simplest)
- You can get a 220-page manual on “Getting Your Data into VisIt”(!)

Case study: Stanford LES code

- Acknowledgement to Konstantin Kemenov of Cornell University, for providing recent Ranger results using this code plus ParaView…
- In the Large Eddy Simulation (LES) code, snapshots of the time evolution are periodically dumped, unformatted, in a given layout
- The dump files contain dimensions, vector and scalar fields, etc.
- Konstantin's first attempt was to translate the dump files into Plot3D format using a simple F95 code: failed
- Second attempt was to translate files into Legacy VTK: succeeded, but many tedious details had to be to understood (e.g., big endian)
- F95 code written for Linux/ifort did not work for gfortran or on a Mac
- Demo 1: compare Konstantin's code to comparable code for XDMF
- Demo 2: show Konstantin's ParaView visualizations

Conclusions

- There are three competing goals for how to store your HPC data
  - Compact and efficient I/O
  - Portability of data to other platforms
  - Intelligibility of data format to your preferred analysis tools
- No single “magic bullet” covers all three completely
- There are several ways to compromise if you want to use ParaView or VisIt for doing visualization and analysis
  - Have your code directly write XDMF or Silo (HDF5 + additional metadata) using provided libraries
  - Write a script or program in your favorite language to translate the data from an efficient format you like into a format that the tools understand
  - Write filters that can plug directly into VisIt or ParaView