VizSchema: metadata for visualization

Presenter: J.R. Cary[†*]

[†]Tech-X Corporation; [*]University of Colorado

Thanks to the VizSchema Team: T. Austin, A. Hakim, J. R. Cary, S. Veitzer, A. Pletzer, D.N. Smithe, M. Miah, P. Stoltz, S. Shasharina, P. Hamill, S. Kruger, D. Alexanda, P. Messmer

December 4, 2008

- Why
- How (attribute definitions)
- Details
- What (codes that have adopted)

VizSchema: getting your data into advanced (parallel) viz tools

- With the production of large amounts of data through parallel computation, we need to move to parallel visualization tools
  - Handle files that do not fit into memory
  - Process data in reasonable time
- Modern tools allow incorporation of plugins to read data
  - VisIt, ParaView
  - Develop C++ class that overrides base-class (virtual) methods for
    - Mesh and data registration
    - Return (by pointer) the mesh and data
- The rub: plugins must be able to interpret the data
  - What are the meshes (rectilinear, structured, unstructured) and their parts?
  - What datasets are there, on what meshes are they defined, and what type (nodal, zonal) are they?

Visualization Schema: provide the attributes to permit visualization

VizSchema expanding to include provenance

- Who generated the data?
- When was it generated?
- With what version of the software?

Provenance Schema: attributes to reproduce the data

VixSchema data semantics

- Variable names, descriptors

Data semantics: attributes to interpret the data

Principles of VizSchema

- Should not interfere with other schema
- Should be able to work for the many common types of data
- Should be appendable

The visualization community has classified mesh and data types

- (VTK becoming defacto standard)
- Meshes
  - Structured: uniform, rectilinear, irregular
  - Unstructured: points and cells
  - We provide some coordinate system generalizations
- Data lives on those meshes, some number of values per point (nodal) or cell (centered)
  - (okay, face and edge too…)

Our task it to add the corresponding descriptions to the data

Self-describing data formats provide the means to attach metadata

- Binary data, so small
- Tags tell type (float/double/int) and shape (dimensions)
- Can attach attributes to data to describe
  - What is represents (mesh, data)
  - Correspondents (mesh of data)
  - Units, preferred names, …
- Two main formats
  - NetCDF: limited to flat collection of arrays (?, now being generalized? New version depends on hdf5.)
  - HDF5
    - Fully hierarchical (better for complex component models)
    - Standard for parallel I/O

VizSchema defines the tags needed for interpretation of data

```
xena.cary$ h5ls -lr core-edge-exemplar_core_5.h5
/                        Group
/density                 Dataset {151, 33}
/mappedGrid              Group
/mappedGrid/points       Dataset {151, 33, 2}
/temperature_H2p1        Dataset {151, 33}
/temperature_electron    Dataset {151, 33}

xena.cary$ h5dump -A -d /density core-edge-exemplar_core_5.h5
HDF5 "core-edge-exemplar_core_5.h5" {
DATASET "/density" {
   DATATYPE  H5T_IEEE_F64LE
   DATASPACE  SIMPLE { ( 151, 33 ) / ( 151, 33 ) }
   ...
   ATTRIBUTE "vsMesh" {
      (0): "mappedGrid"
   }
   ATTRIBUTE "vsType" {
      (0): "variable"
   }
}
}
```

VizSchema allows definition of mesh type and data

```
xena.cary$ h5dump -A -g /mappedGrid core-edge-exemplar_core_5.h5
HDF5 "core-edge-exemplar_core_5.h5" {
GROUP "/mappedGrid" {
   ...
   ATTRIBUTE "vsKind" {
      (0): "structured"
   }
   ATTRIBUTE "vsType" {
      (0): "mesh"
   }
   DATASET "points" {
      DATATYPE  H5T_IEEE_F64LE
      DATASPACE  SIMPLE { ( 151, 33, 2 ) / ( 151, 33, 2 ) }
      ...
   }
}
}
```

Dimensionality

Work in progress as we define more and newer types, account for more variation (e.g., Fortran data put out with C interface) See: https://ice.txcorp.com/trac/vizschema

Companion descriptor file complicates for complex component models

- XDMF reader is a great idea
  - Define XML file that defines the meshes and data inside an HDF5 file for plotting

but

- For FACETS, every output file can have a different structure, potentially with different names, so one XML file per data file?
- VizSchema keeps data with its metadata so that interpretability is not lost

Metadata can be attached *a posteriori*

- Open file using pytables
- Open a dataset
- Attach attributes (one line of code per attribute)
- Close file

```
h5file = tables.openFile(fileName, mode='a')
dataSet = h5file.getNode("/" + dataSetName)
dataSet.attrs.vsType = "variable"
h5file.close()
```

Namespacing with *vs* to avoid clashes

VizSchema plugin for VisIt provides reference implementation, but more to do

- Reads data and meshes for uniform cartesian, structured, unstructured polygons
- More to go: tets, hexahedra, rectilinear grids, …
- Parallelism ("MDST")
- Multiblock ("MDST") for data correlation
- Time sequences of data ("MDMT")
- Retrofit to NetCDF?
- We welcome involvement: defining names, generalizing schema

VizSchema now in use by multiple computational applications

- VORPAL (OASCR winner, 2008)
- NIMROD
- FACETS
- UEDGE
- MODAVE (Climate)
- PolySwift
- Will be working with COMPASS SciDAC to bring capability to other codes
- Will be working with other fusion codes