

Issues for Large-Scale Data Analysis and A Primer on ParaView and VTK

Jon Woodring

Boring stuff first:

Issues for Large-Scale Data Analysis and Visualization (from the perspective of HPC simulations)

- **I/O bandwidth and dealing with the large-scale parallel file systems**
 - This is the primary issue from my perspective – because it is the only bottleneck of any consequence
 - Compute is such a small fraction of overall time when compared to the time necessary for reads and writes
- **A secondary issue is migrating the algorithms and pipelines to massively parallel platforms without having to rewrite for every platform**
 - Thrust (i.e., data parallel programming with segmented scans – see Blelloch’s Thesis and programming on the CM-5/C*)

Strategies for I/O Bandwidth

- **In situ**
- **Staging/Co-processing**
- **Data reduction**
 - Science-based feature detection (digested data)
 - Sampling and Compression
 - Quantifying the “uncertainty” from data reduction
- **Multi-scale and streaming**
- **Distributed file systems and move compute to the data**
- **Indexing scientific data sets to subset at the I/O layer**
- **Flash/SSD technologies for random access**

How could an Interactive Analysis and Visualization Tool work with the Archives?

- **Reader module that knows how to access the databases**
- **Embed the tool at the data source as a service (Ferret for the Earth Systems Grid or ParaViewWeb for example) – but that doesn't solve data movement (i.e., inner joins)**
 - In both cases, the data access would need some support for multi-scale/streaming data subsetting for interactivity

PARAVIEW AND VTK

What is VTK (Visualization ToolKit)?

<http://www.vtk.org> <http://www.vtk.org/Wiki/VTK>

- **A demand-driven data pipeline model for analysis and visualization**
 - Sinks drive execution only to the last “dirty” (Modified) module (caveat being that there are data copies to facilitate the short-circuit termination)
 - There are other implementations of the VTK executive for push-driven, streaming, and multi-resolution execution

- **Modules (C++ classes)**

- Readers, sources, filters, mappers, views, writers

- **Programming in VTK**

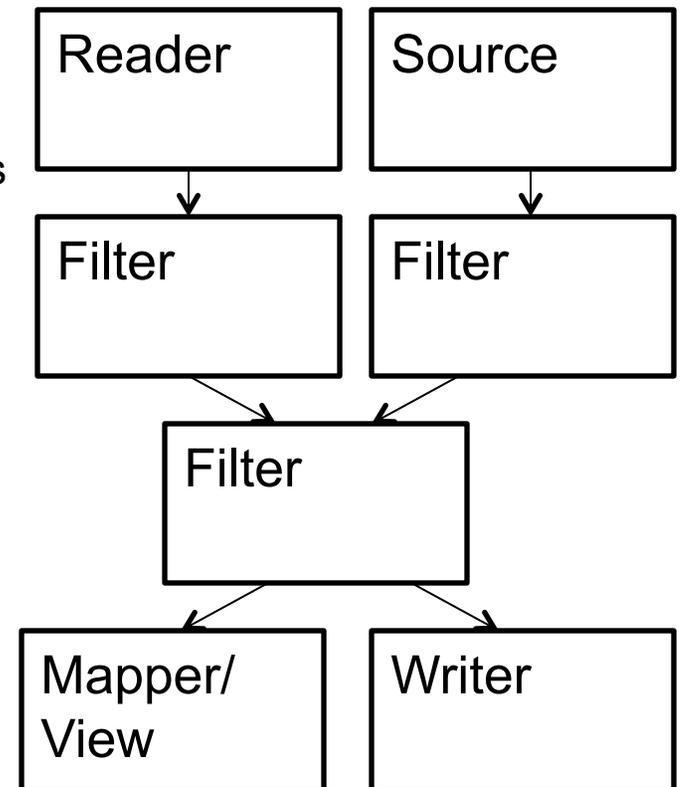
- C++, Python Programmable Filter

- **Programming with VTK**

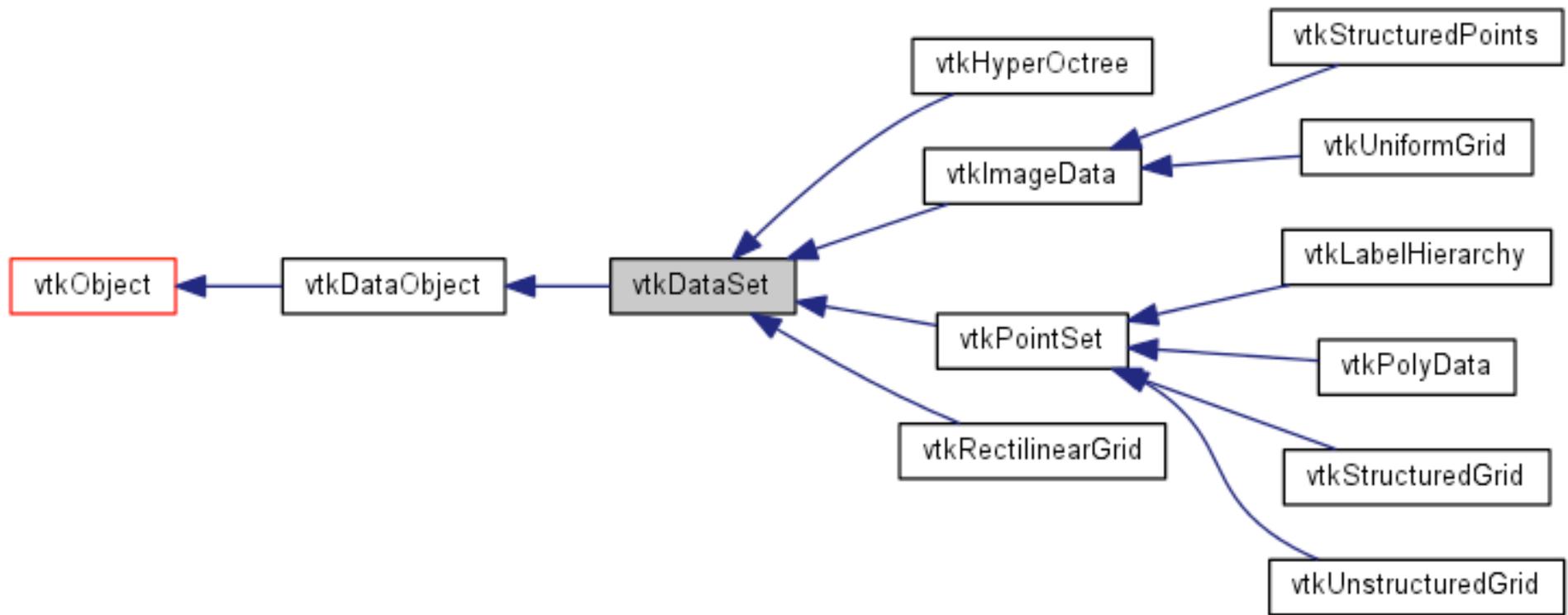
- C++, Python, Java, tcl

- **MIT/BSD style license**

- **Funded by DOE, DOD, NIH, and others**



VTK Data Types (vtkTable not shown)



- **Values are in Arrays that are Point-centered or Cell-centered values**

- Scalars, vectors, tensors, arbitrary size tuples
- `vtkAbstractArray`, `vtkDoubleArray`, `vtkFloatArray`, `vtkCharArray`, etc.

What is ParaView? <http://www.paraview.org> <http://www.paraview.org/Wiki/ParaView>

- **A parallel server framework that executes parallel VTK pipelines**
 - Many modules are trivially parallel – no extra code is needed
 - Others that require communication, just use MPI or an abstract layer to MPI
- **The GUI is *not ParaView* – it is just one of several front ends to the pvserver – just have to “talk” the ServerManager protocol**
 - Qt GUI, Custom Qt GUIs, Python (pvpython, pvbatch), ParaViewWeb, ParaView Co-Processing
- **VTK classes are exposed to ParaView by a small amount of ServerManager xml (and possibly custom Qt code for the Qt GUI)**
 - Extensible at run-time through plugins, such as the yt plugin, or Python
 - Mapper->View->Interactor are compressed to “Representation” Proxies
- **MIT/BSD style license**
- **Funded by DOE, DOD, and others**

<http://www.vtk.org/Wiki/ParaViewWeb>

DEMO OF PARAVIEW GUI AND PARAVIEWWEB

Implementing a new VTK Reader or Filter

- <http://www.vtk.org/doc/nightly/html/annotated.html>
- **A module maps to a C++ class – inherit from an abstract class (vtkUnstructuredGridAlgorithm for example)**
 - Usually only one method that needs to be implemented: RequestData
 - Sometimes also need to implement RequestInformation: Light-weight meta information processing for the pipeline (spatial extent, piece number, etc.)
- **RequestData**
 - Actual data is provided or processed
 - Get input, get output, process data, and put data into the output
- **Execution is done automatically by the pipeline (vtkExecutive) when a sink or any other module requests UpdatePipeline**
 - Execution only goes as far as the last Modified (dirty module – parameter change)
 - RequestData will be called when it needs the output of the module

How Kitware Enforces Good Style for Development (if you want to commit back to the VTK/ParaView repo)

- **Overall architecture is prescribed by a data model and execution model**
 - VTK is a demand driven pipeline based around modules and data sets
 - ParaView is the proxy pattern built on top of parallel VTK pipelines
- **Branchy Style Development with a Repository (git)**
 - Automatic lint checks before commit is allowed
 - Can only branch off the stable master
 - Merges only on unstable branch
 - Weekly review to merge commits from unstable to stable only if the commit is in the bug (new feature) database
 - Currently code reviews are optional (gerrit) but I have a feeling they will become mandatory in the future

Getting Help

- <http://www.vtk.org/cgi-bin/mailman/listinfo>
- <http://www.paraview.org/paraview/help/mailing.html>
- <http://www.vtk.org/Wiki/VTK>
- <http://www.paraview.org/Wiki/ParaView>

- **How to make things “sexy” in ParaView (for PR)**
 - Use the Manta raytracer for nice shadows and lighting on polygonal meshes
 - Convert the data to structured mesh and volume render it
 - Unfortunately the vtkGaussianSplatter isn’t exposed by default to ParaView, but I (Jon Woodring) and Katrin have an xml plugin to expose it as a plugin to ParaView – hopefully in future versions of ParaView will expose Gaussian Splating by default