Big Data Analysis with ParaView

Going beyond the built in server
Session 1: Introductory Lecture

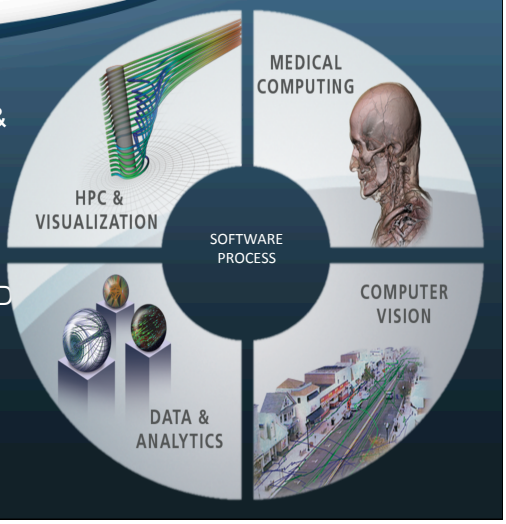
See also: http://vimeo.com/63567497

Agenda

- Session 1:
  - What the heck is Kitware?
  - What is ParaView?
  - Fundamentals of parallel ParaView
  - Three getting started exercises mixed in
- Session 2: (evening)
  - Deeper understanding of ParaView's various features
  - 1:1 help

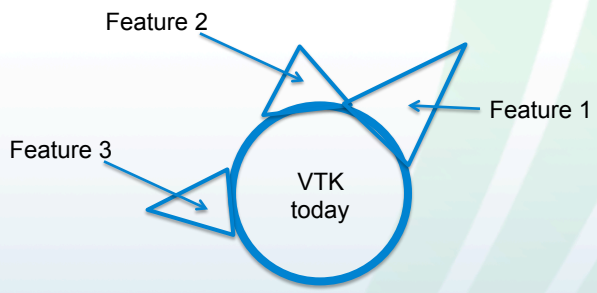
Kitware

- Collaborative software R&D: algorithms & applications, image & data analysis, support & training
- Industry, government, academia
- Best known for open source toolkits and applications
- 126 employees: ⅓ masters, ⅓ PhD
- Founded in 1998; $28M revenue 2011



VTK today

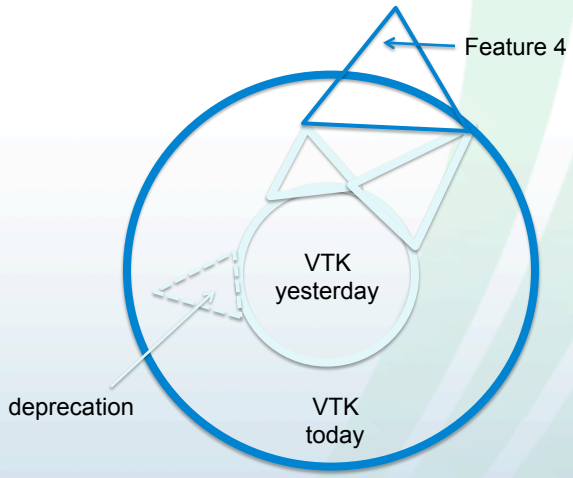
Kitware



Features come from:

- open source external developers and/or
- Kitware consulting development
  - Why go through Kitware?
    Most familiar with code, save development time

Kitware



The development frontier *is* Kitware's business model

Kitware

We grow Open Source Solutions

- **No licensing costs; proven in products**
- **Funding & contributions from around the world**
- **VTK**—the Visualization Toolkit
- **ParaView**—Large data visualization application
- **ITK**—Insight image analysis Toolkit
- **CMake**—cross-platform build system
  - CDash, CTest, CPack, software process tools
- **OpenView** / **Tangelo**—Informatics and infovis
- **Kiwi & VES**—Mobile / GLES rendering
- **IGSTK**, Lesion Sizing Toolkit, **CTK**, **vxl**, **Open Chemistry Project**, **VolView**, **tubeTk**, and more…

What is ParaView?

An application and architecture for display and analysis of **massive** scientific datasets.

- Front end to VTK based visualization
- Distributed Memory Parallel features turned on
- Client/Server architecture lets it runs on variety of platforms
  - from netbooks
  - to the largest machines in the world
- Support for tile display and parallel rendering
- Level of detail techniques keep it interactive on huge data

Kitware

History (http://www.paraview.org/Wiki/ParaView_Release_Notes

- 1999 LANL/Kitware project (via ASCI Views)
  - Build an end user tool from VTK
  - Make VTK scale
  - October 2002 first public release, version 0.6
- 2002-2005 Versions 0.6 through 2.6
  - Continued growth under DOE Tri Labs, Army Research Lab and various other partnerships
- September 2005 ParaQ project started
  - Sandia, Kitware and CSimSoft
  - Make ParaView easier to use
  - Add quantitative analysis
  - May 2007 version 3.0 released
- Continuing to evolve
  - 4.0 released Jun 2013 – releases ~6 months

Kitware

What to expect from parallel ParaView?

- Amdahl's Law

$$Speedup(CPUs) = \frac{1}{Serial + \frac{Parallel}{CPUs}}$$

aka Strong scaling

If data size is fixed, don't expect great scalability.

More processors != faster

- Gustafson's Law

$$Speedup(Machines) = Machines - Serial * (Machines - 1)$$

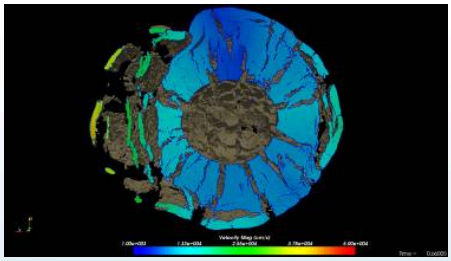
aka Weak scaling

As data size grows, you must have more resources.

More disk and memory = higher resolution possible

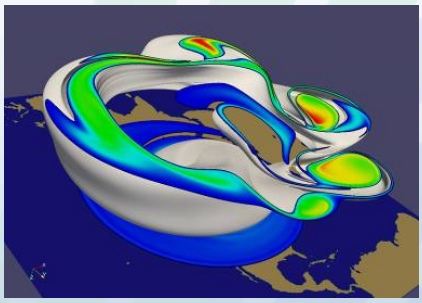
Kitware

ParaView is for Extremely Large Data



1 billion cell asteroid detonation simulation

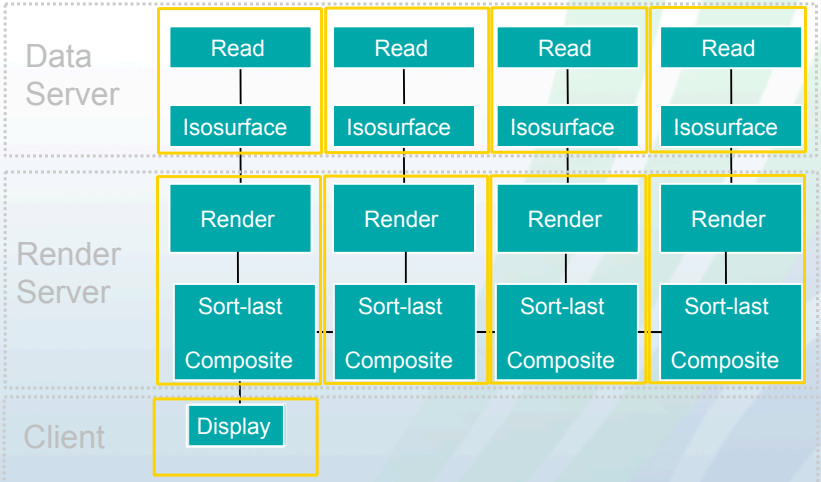
½ billion cell weather simulation



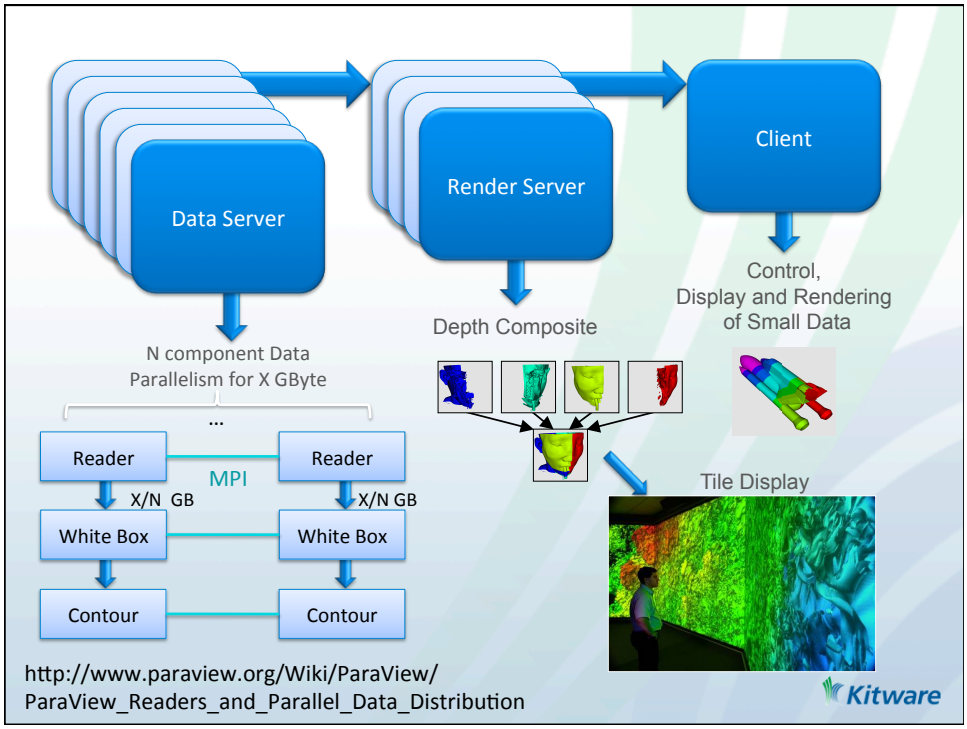
source: Sandia National Lab

Kitware

Data Parallel Pipeline



Kitware



ParaView's running modes

Builtin aka Standalone aka Serial	DS RS Client	all components within one process (client may be GUI or pvpython) "paraview" || "pvpython"
Combined Server	DS RS Client	data processing and parallel rendering in MPI job of combined processes. control from TCP connected client. "mpiexec -n x pvserver &; paraview"#||pvpython #+ Connect
Batch	DS RS	Server is an MPI job which directly runs a python script "mpiexec –n x pvbatch \ vis_script.py"
Split server	DS RS Client	Data processing and parallel rendering are both MPI jobs. " mpiexec –n x pvdataserver&; \ mpiexec –n y pvrenderserver &; \ paraview" #+ Connect

Is my ParaView parallel capable?

- MPI is required for large data processing
  - Kitware binaries not really suitable ("right" MPI for all installations is?)
- Enable Python is strongly recommended
  - enables scriptability = unattended use, reproducibility, simplifies parameter studies, additional features, …
- How to get access to Parallel ParaView
  - Option 1: access a system with it already installed
    - http://paraview.org/Wiki/ParaView/HPC_Installations
  - Option 2: check your distro (e.g. Fedora has paraview-mpi)
  - Option 3: build your own
    - PARAVIEW_USE_PYTHON, PARAVIEW_USE_MPI = ON, QT=OFF
    - http://paraview.org/Wiki/ParaView:Build_And_Install
    - http://paraview.org/Wiki/ParaView/Superbuild

Kitware

Python on a Cray? (are you nuts!?)

- Wait, pvbatch is only scripting interface?
- No pvbatch without python??
- No python without shared libs???
- Modern Cray OS (Cluster Compatibility Mode) has shared libs
  - although not ideal - 10k cores opening same shared library simultaneously is a bad thing
- Best to compile statically
- PV 4.1 has frozen python (NO runtime disk access at all even for simple .py modules that we use)
- ParaViewSuperBuild git://paraview.org/ParaViewSuperbuild.git does so for Cray and BlueGene

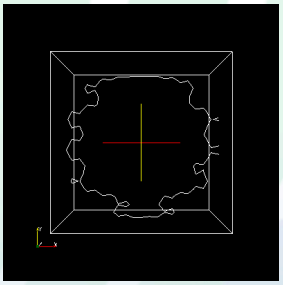
Kitware

How to Learn ParaView Scripting

1. **Trace:**
   ParaView -> Tools -> Start Trace
   Do something in GUI
   ParaView -> Tools -> Stop Trace
   File -> Save
2. Tools -> Python Shell
   For tab completion and immediate synchronization with GUI
3. http://paraview.org/Wiki/ParaView/Python_Scripting
   For the nitty gritty details

Kitware

Exercise : record/playback



1. Record
   - Start ParaView GUI
   - Tools->Start Trace
   - Sources->Wavelet (Apply)
   - Filters->Slice, ZNormal=1 (Apply)
   - Filters->Contour (Apply)
   - Tools->Stop Trace (filename exercise.py)
2. Playback
   - bin/pvbatch exercise.py
3. Tailor
   - add WriteImage("exercise.png") at end of exercise.py
   - bin/pvbatch exercise.py

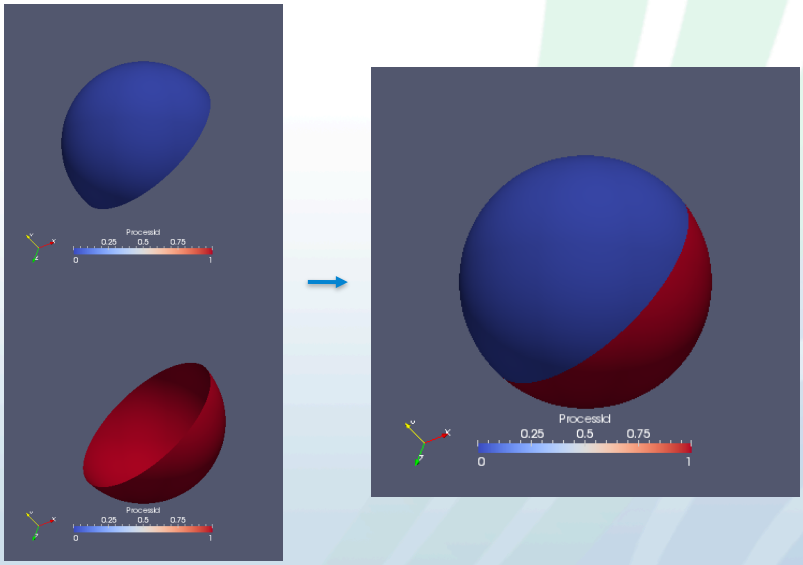
Kitware

OpenGL on a Cray? (are you nuts!?)

- CPU render cost vs ship to GPU render cost
  - Transfer time significant whenever GPU is not local and you have big data
  - Current generation of supercomputers have GPUs on each node
  - The tradeoff of when to use OSMesa may shift
- Typically use OSMesa compiled statically
  - Configure ParaView to use it as a pure offscreen rendering solution
  - for instructions and recommendations:
    See ParaView Guide Parallel Rendering chapter
    and
    http://www.paraview.org/Wiki/ParaView/ParaView_And_Mesa_3D

Kitware

Depth Compositing



Kitware

Visualization on Big Iron

- Environment search paths
  - Where ParaView binaries, libraries and python can be found

mypaths.sh

```
set dir = /lustre/widow2/scratch/demarled/pv4.1.0/GNUINSTALL
setenv PATH ${dir}:${PATH}
setenv PYTHONHOME ${dir}/pythonhome #where my python is
setenv PYTHONPATH ${dir}/pythonlib  #where paraview module is
```

  - Or more typically when sys admins installed ParaView

```
soft add +paraview-4.1.0
```

Or perhaps

```
module load paraview-4.1.0
```

Kitware

Visualization on Big Iron

Access resource through the queue - syntax varies per scheduler (MOAB, Oracle Grid Engine, Microsoft HPC server…).

- Unattended/Batch Session

```
qsub -v PYTHONHOME=${dir}/pythonhome … \
  -q batch -A yourProject \
  -l size=1600 -l walltime=03:00:00 \
  aprun –n 1600 pvbatch parallelREALLYHUGESphere.py
```

- Interactive Session

```
qsub -I \
  -q debug -A yourProject \
  -l size=16 -l walltime=00:60:00
```

Wait for session to be given to you

```
source mypaths.sh
aprun -n 16 pvbatch parallelSphere.py
```

look at result

```
aprun -n 16 pvbatch doSomethingElse.py
```

Kitware

Visualization on Big Iron

Access resource through the queue - syntax varies per scheduler (MOAB, Oracle Grid Engine, Microsoft HPC server…).

- Unattended/Batch Session

```
qsub -v PYTHONHOME=${dir}/pythonhome … \
  -q batch -A yourProject \
  -l size=1600 -l walltime=03:00:00 \
  aprun –n 1600 pvbatch parallelREALLYHUGESphere.py
```

- Interactive Session

```
qsub -I \
  -q debug -A yourProject \
  -l size=16 -l walltime=00:60:00
```

Wait for session to be given to you

```
source mypaths.sh
aprun -n 16 pvbatch parallelSphere.py
```

look at result

```
aprun -n 16 pvbatch doSomethingElse.py
```

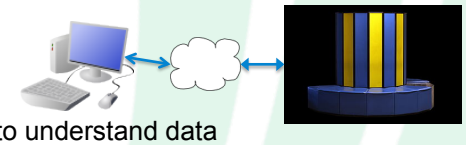
Kitware

Exercise : batch visualization

- connect to (ssh -R) tukey/mira.alcf.ornl.gov
  - ssh -X -R 99999:localhost:22 user@tukey.alcf.anl.gov
- reserve some nodes
  - qsub -I -A projname -t #minutes -n #nodes
- set paths
  - soft add +paraview-4.1.0
- runscript
  - mpiexec -n 2 -machinefile $COBALT_NODEFILE \ /soft/visualization/paraview/v4.1.0/bin/pvbatch exercise.py
- send result back and view them
  - scp -P 99999 file user@localhost:/directory

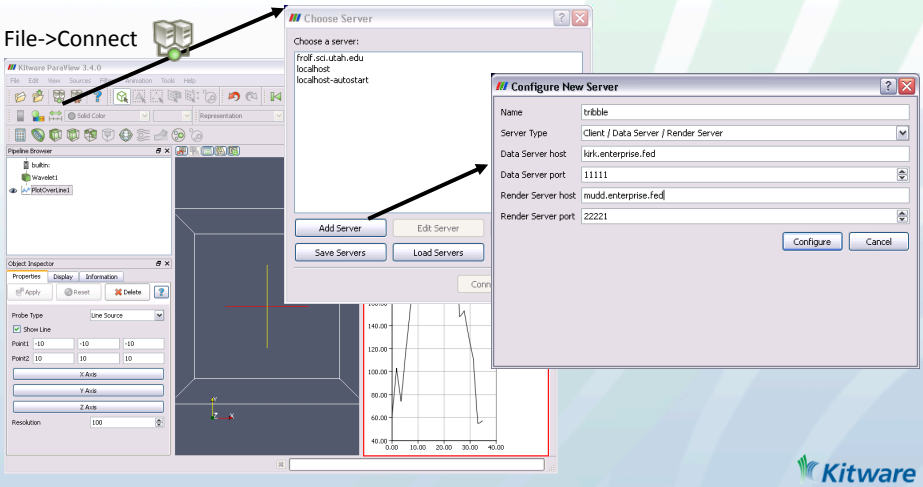
Kitware

Interacting with the data



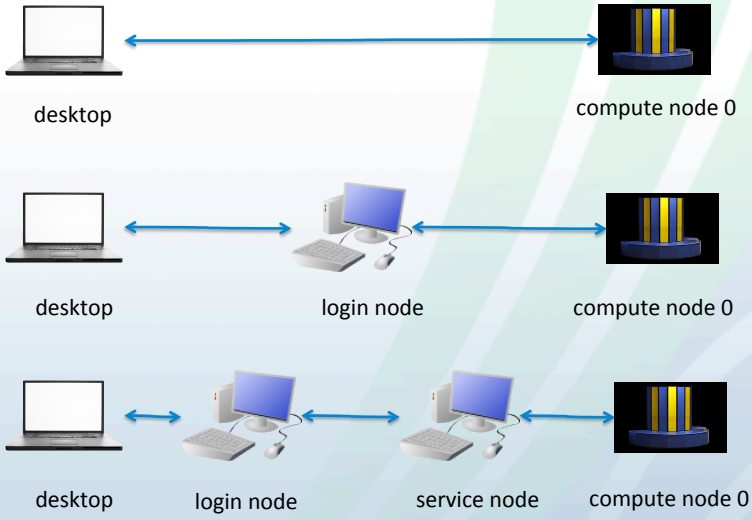
- Interaction is sometimes important to understand data
- Establish a connection to control ParaView server from your desktop

File->Connect



Kitware

Path from laptop to compute node



Kitware

Establishing the connection

File->Connect
"Add Server"

- Server Type:
  - Client/Server - data and render servers combined, and wait for client to contact it
  - … **(reverse connection)** - the client will wait for the server to contact it instead
- Startup Type:
  - Manual - start the server job yourself outside of the GUI
  - **Command** - enter commands that ParaView will run to start the job for you
- NOTE: client and server release numbers must match
  - Platform (mac/win/lin) doesn't matter, only rel number and VTK_USE_64BIT_IDS
- Examples:

```
1. mpiexec -n 8 pvserver &
2. ssh remote_machine mpiexec -n 8 pvserver &
3. xterm -e ssh remote_machine mpiexec -n 8 pvserver &
4. ssh -R 22222:local_machine:11111 remote_machine \
   mpiexec -n N pvserver \
   --reverse-connect --server-port=22222 &
5. ssh -R 22222:local_machine:11111 script.sh &
   Where script.sh contains:
   qsub reserve_then_tunnel2_backto_login_and_run_server.sh
```

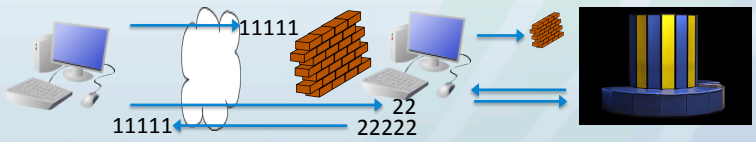
Kitware

Tunneling

- Examples:

```
4. ssh -R 22222:local_machine:11111 remote_machine \
   mpiexec -n N pvserver \
   --reverse-connect --server-port=22222 &
```

```
5. ssh -R 22222:local_machine:11111 script.sh &
   Where script.sh contains:
   qsub reserve_then_tunnel2_backto_login_and_run_server.sh
```



- http://paraview.org/Wiki/Reverse_connection_and_port_forwarding

Kitware

.pvsc file

- The client saves all connection settings in an XML text file
  - %APPDATA%\ParaView\servers.pvsc on windows
  - ~/.config/ParaView/servers.pvsc elsewhere
  - Options entries build simple GUIs

```
<Option name="PORTNUM"/> <Option name="NUMPROC"/>
```

  - Arguments entries pass user's choices into the command

```
<command exec ="xterm"> <Arguments> …
  <argument value="$PORTNUM$:localhost:22222"/>
  <argument value="qsub -l size=$NUMPROC$"/> …
</arguments>
```

- Since 3.14 you can import them within ParaView
  - **Connect->Fetch Servers**
  - Enter URL of a public web/wiki page where your sys admin has posted a .pvsc
  - ParaView will scan that and add connections it finds
- http://www.paraview.org/Wiki/ParaView/HPC_Installations

Kitware

Exercise : obtain .pvsc for Tukey

- File -> Connect
- "Fetch Servers" Button
- "Edit Sources" Button
- Add

pvsc http://www.paraview.org/Wiki/images/9/99/Argonne.xml ANL

- Select tukey.4.1.1
- "import selected" Button
- Quit # to make sure changes saved in preferences

Kitware

Exercise : interactive vis on Tukey

- File -> Connect
- Select tukey.4.1.1
- When prompted in the xterm, login with secureId key
- “s” to show queue
- When view returns
- File->Open

/projects/ATPESC2014/visualization/ \
ParaViewTutorialData/can.ex2

Kitware

Local view of remote data

- File->Open : always shows you data server’s file
- Big data stays on server
- Visualization (typically) produces small subset of full data:
  - At one moment in data time
  - External surface polygons or pixels sent to client for display
  - Rendering is easy, let client do it when possible
- Run client on desktop, not on login node
  - Login nodes generally not very powerful and shared resource among many
  - X11 forwarding is best avoided
    - Lots of unnecessary communication underneath
    - Can end up with “parallel rendering” meaning N nodes ask your client to render
  - VNC session can be OK
  - Let ParaView take care of graphics delivery

Kitware

About Remote Rendering

- Preferences/Settings->Render View->General ->
  Use Immediate Mode Rendering
  - Make sure it is checked (enabled)
  - Display lists only work well with small datasets
- Preferences/Settings -> Render View -> Server ->
  Remote Render Threshold

surface polygons < threshold

- Client side rendering
- Root node of server gathers polygons
- Sends polygons to Client
- Client renders polygons
- Communicates only when data changes

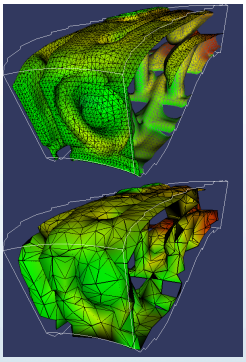
surface polygons > threshold

- Server side rendering
- N nodes render 1/n'th of data
- Server decides nearest pixels to client
- Server sends images to client
- Communicates on every camera move

- Preferences/Settings -> Render View -> Server ->
  Client/Server Parameters -> Presets

Kitware

Level of Detail – to maintain interactivity



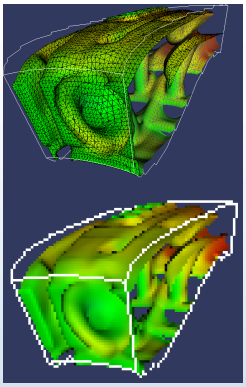
Type 1: Spatially based

- Edit->Settings->Render View->General
  LOD threshold

  Down-samples geometry while interacting

Kitware

Level of Detail – to maintain interactivity



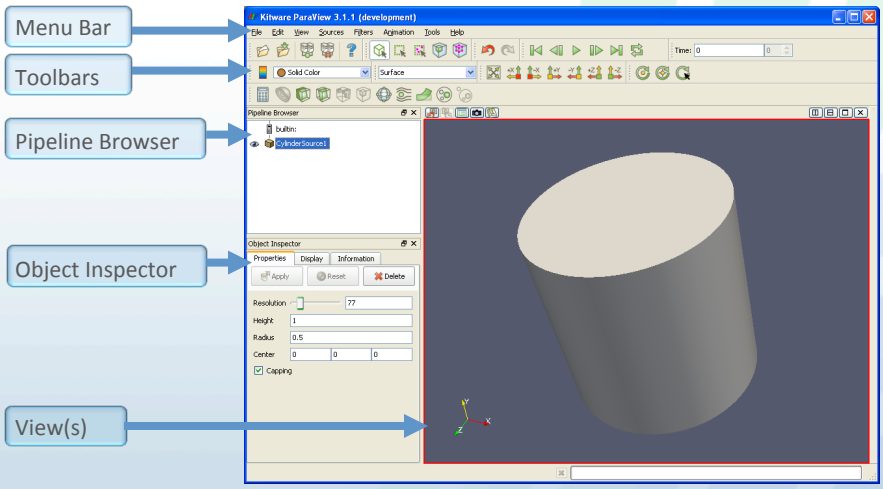
Type 2: Image Based

- Edit->Settings->Render View->Server Interactive Subsample Rate
  Down-samples pixels while interacting
- Image compression
  Compress pixel stream on the fly

Try preset for your situation first

Kitware

User interface

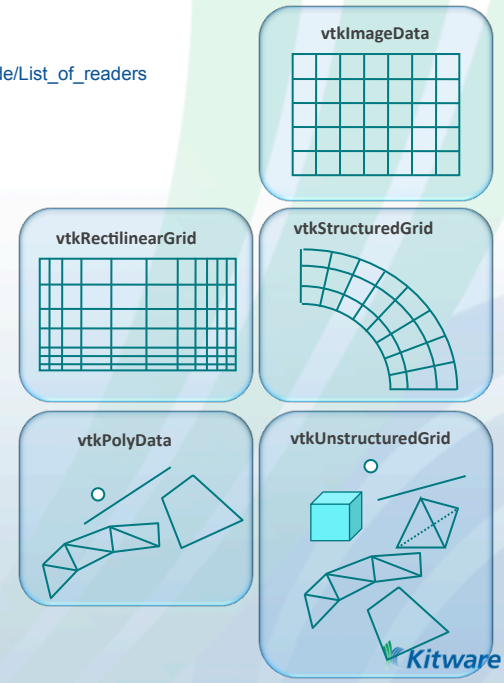


Kitware

File->Open

http://paraview.org/Wiki/ParaView/Users_Guide/List_of_readers

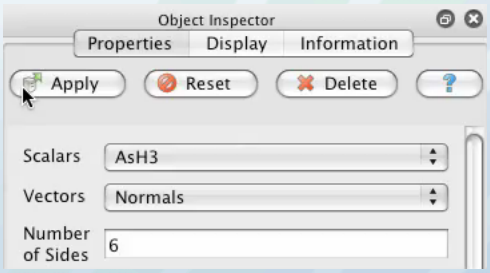
- Kitware formats: (.vtk, .pvd, .vt?, pvt*)
- Exodus
- Xdmf annotated hdf5 (.xmf, .xdmf)
- netCDF CF (.ncdf, .nc)
- SpyPlot CTH
- EnSight (.case, .sos)
- Protein Data Bank (.pdb)
- Digital Elevation Map (.dem)
- Tecplot ASCII (.tec, .tp)
- Fluent Case Files (.cas)
- OpenFOAM Files (.foam
- LANL VPIC (.vpc)
- SLAC netCDF mesh, mode and particle data
- Stanford Polygonal (.ply)
- PNG Image Files
- NRRD image files (.nrrd)
- Comma Separated Values (.csv)
- All visit formats: enzo, miranda, pixie, samrai, silo, +visit extensions
- NCAR vapor data format
- And many more



Kitware

Apply

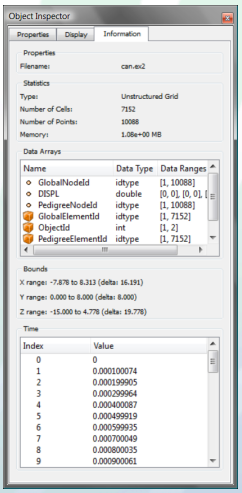
- ParaView is meant to process large data
- It waits for you to commit to any action that might take a long time on a really large data set
- Net result is you won't see any data change until you hit the glowing Apply button on the Properties tab of the Object inspector



Kitware

Inspect the data

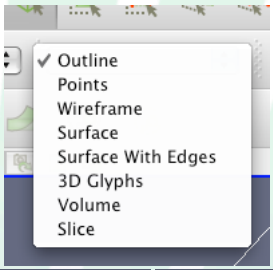
- Information about the Active Filter's output
- DataObject structure
- Size (Bytes, #points, #cells)
- Geometric bounds
- Structured bounds
- Arrays:
  - Name
  - Association =point, =cell)
  - Data Type
  - Data Ranges (and scalar/ vector)
- Temporal Domain

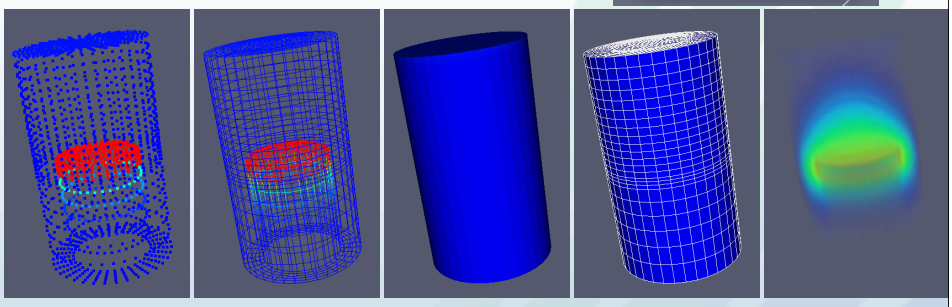


Kitware

Display the data

- Representations (aka Displays): visual characteristics of one particular data set in one particular view



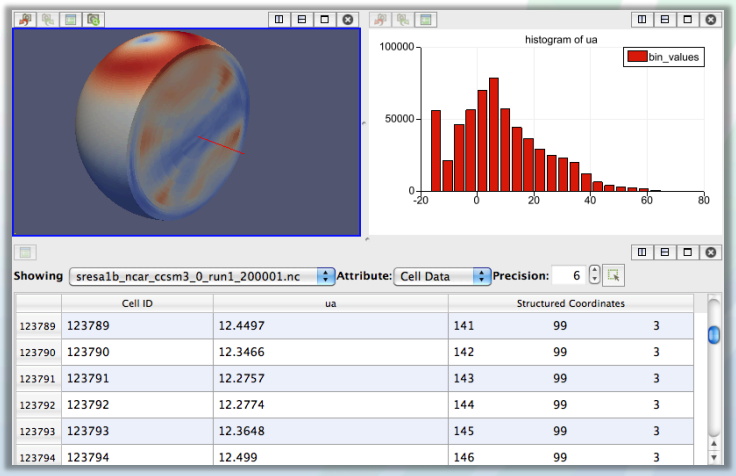


Points | Wireframe | Surface | Surface with Edges | Volume

Kitware

Display the data

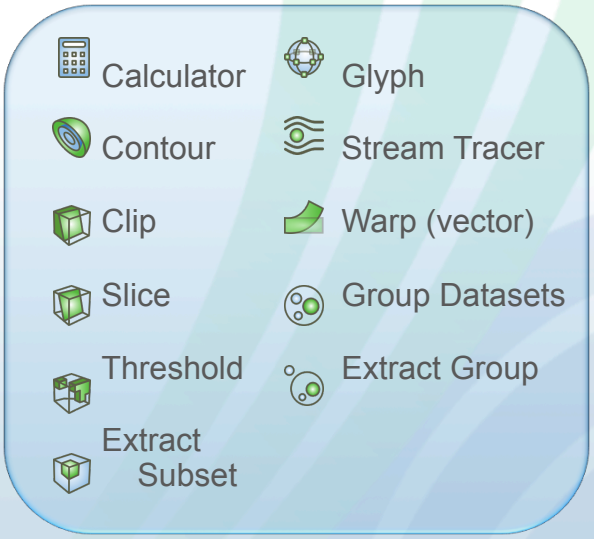
- Views – Windows onto one or more data sets



Kitware

Manipulate the data

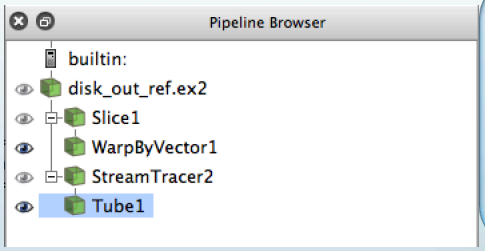
- Filters Menu
  - Recent
  - Common
  - Data Analysis
  - Statistical
  - Temporal
  - Alphabetical
- Quick Launch
  - PC/Linux *CTRL-Space*
  - Mac *ALT-Space*
- Apply Undo/Redo

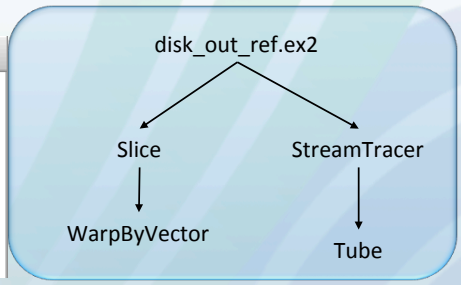


Kitware

Pipeline Browser : condensed pipeline graph

- Use pipeline browser to navigate the graph
- Select a reader/filter to make it active, then object inspector, information tab and display tab pertain to it

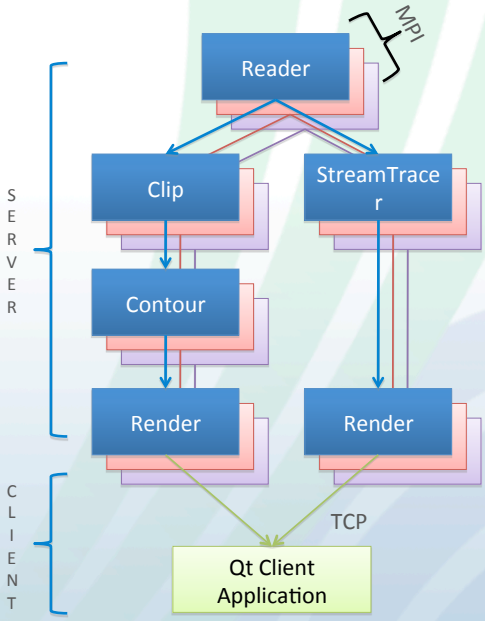




Kitware

ParaView Architecture

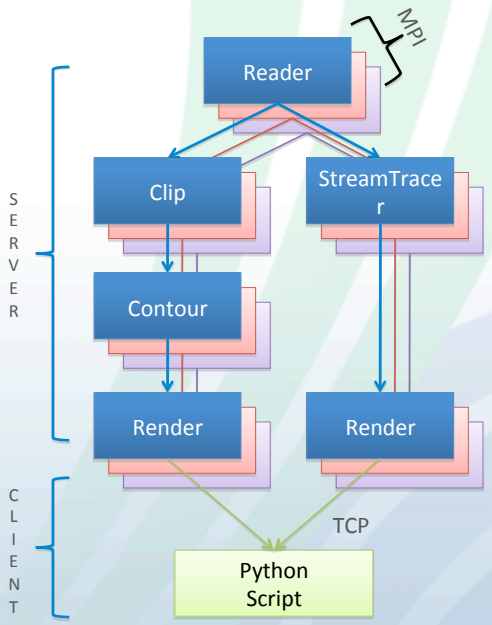
- VTK Pipeline,in parallel,on remote server(s),
- controlled by and feeds into client application.



Kitware

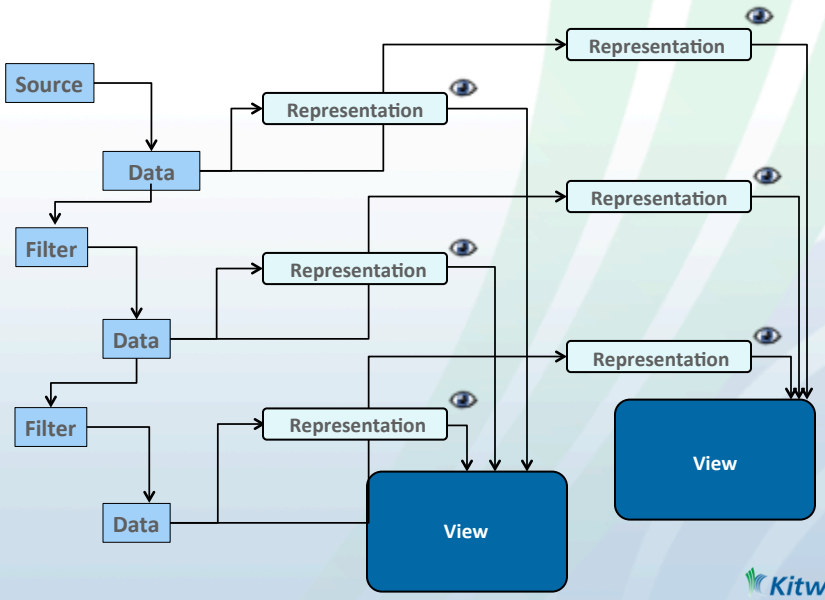
ParaView Scripting

- VTK Pipeline,in parallel,on remote server(s),
- controlled by and feeds into client application.



Kitware

Multi-View visualization pipeline



Kitware

Current Directions

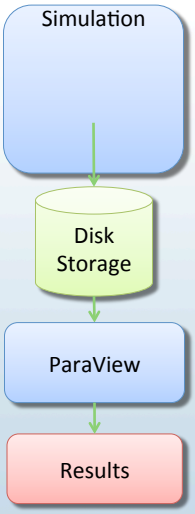
- Catalyst
  - In situ ParaView http://catalyst.paraview.org
- Web and Mobile
  - ParaViewWeb front end http://paraviewweb.kitware.com/PW
  - VES/KiwiViewer http://www.kiwiviewer.org
- OpenGL rendering overhaul
  - http://www.kitware.com/news/home/browse/459
- SMP and GPGPU acceleration
  - http://viz.lanl.gov/projects/PISTON.html
  - http://www.daxtoolkit.org/index.php/Main_Page
  - Inria&EDF vtkSMP EGPGV 2013

Kitware

CATALYST scalable in-situ analysis

From Tera- to Exa-scale
http://catalyst.paraview.org

Traditional Vis



Kitware

Reduced File IO Costs

Time of Processing	Type of File	Size per File	Size per 1000 time steps	Time per File to Write at Simulation
Post	Restart	1,300 MB	1,300,000 MB	1-20 seconds
Post	Ensight Dump	200 MB	200,000 MB	> 10 seconds
In Situ	PNG	.25 MB	250 MB	< 1 second

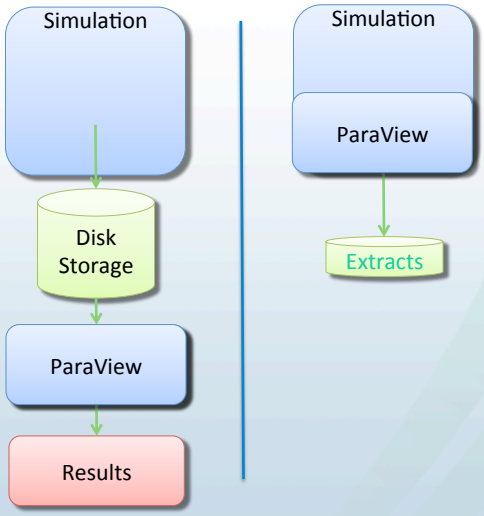
Ex. Helicopter simulation output size for a single time step
- Full data set – 448 MB
- Surface of blades – 2.8 MB
- Image – 71 KB

Kitware

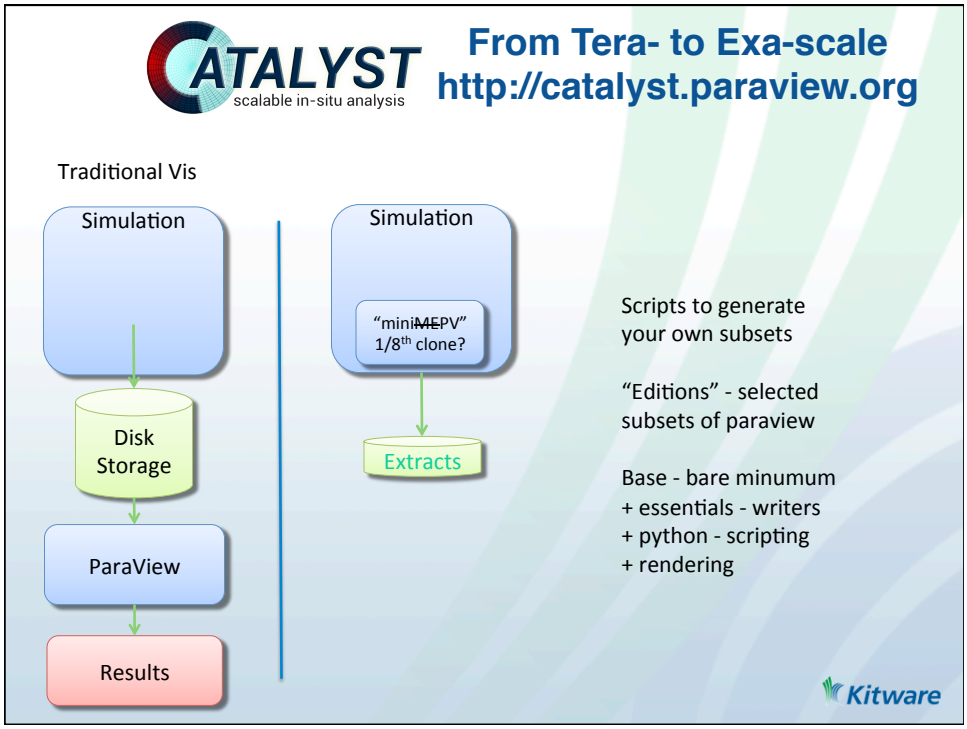
CATALYST scalable in-situ analysis

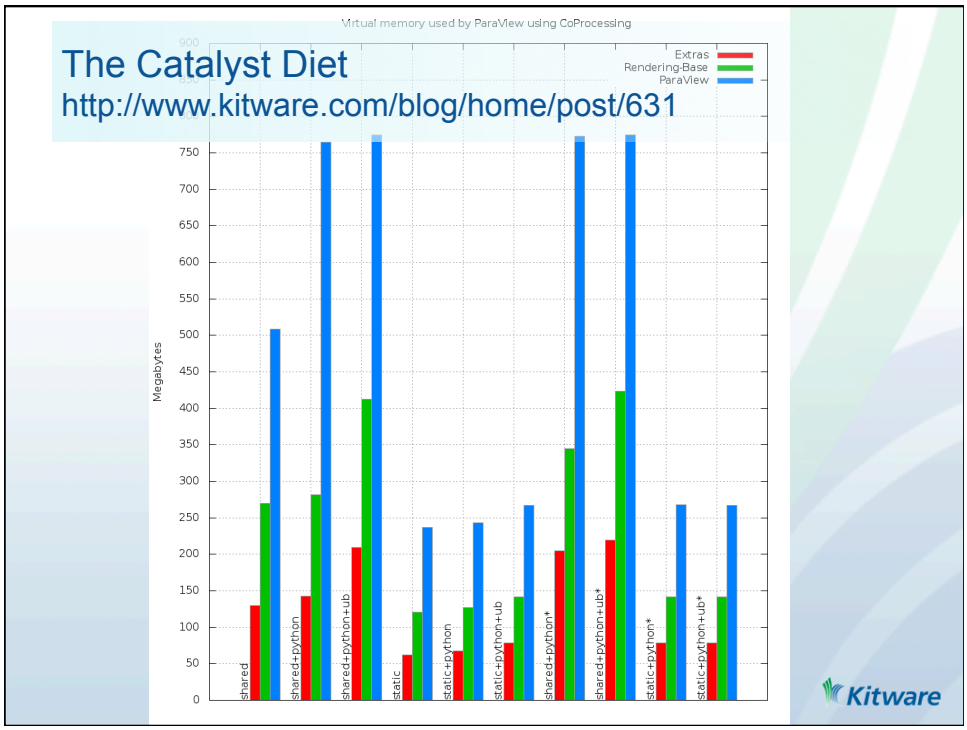
From Tera- to Exa-scale
http://catalyst.paraview.org

Traditional Vis

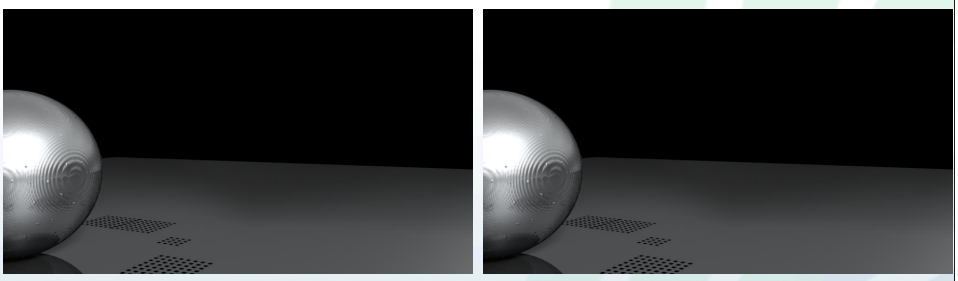


Kitware


CATALYST
scalable in-situ analysis
From Tera- to Exa-scale
http://catalyst.paraview.org
Traditional Vis
Simulation
Disk Storage
ParaView
Results
Simulation
"mini~~ME~~PV"
1/8th clone?
Extracts
Scripts to generate your own subsets
"Editions" - selected subsets of paraview
Base - bare minumum
+ essentials - writers
+ python - scripting
+ rendering
Kitware


The Catalyst Diet
http://www.kitware.com/blog/home/post/631
Virtual memory used by ParaView using CoProcessing
Extras
Rendering-Base
ParaView
Megabytes
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
shared
shared+python
shared+python+ub
static
static+python
static+python+ub
shared+python*
shared+python+ub*
static+python*
static+python+ub*
Kitware

Catalyst: Access More Data



Roughly equal data stored at simulation time

Reflections and Shadows added in post-processing

Kitware

mira@anl - build against it

/soft/visualization/paraview/v4.1.0

…/catalyst/<edition>

Where <edition> is:

base[+essentials][+extras][+python]

In each:

- */source
- */install
- */build_host - host side compilation tools
- */build_cross - compute side library to link to

Kitware

Thank you!

- www.kitware.com
  - portal to all communities kitware referees

  (p.s. we are always hiring)
  http://jobs.kitware.com/opportunities.html

- www.paraview.org
  - portal to all things ParaView
  - Especially: **User Guide, Tutorial, Webinars, Mailing lists, Wiki**

- Please contact us, we are here to help.

Kitware