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
• 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
Features come from:
• open source external developers and/or
• Kitware consulting development
  - Why go through Kitware?
    Most familiar with code, save development time.

The development frontier Kitware’s business model
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
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

What to expect from parallel ParaView?

- **Amdahl’s Law**
  \[ Speedup(CPUs) = \frac{1}{\text{Serial} + \frac{\text{Parallel}}{\text{CPUs}}} \]
  aka Strong scaling
  If data size is fixed, don’t expect great scalability.
  More processors != faster

- **Gustafson’s Law**
  \[ Speedup(Machines) = \frac{\text{Machines} - \text{Serial} \times (\text{Machines} - 1)}{\text{Machines}} \]
  aka Weak scaling
  As data size grows, you must have more resources.
  More disk and memory = higher resolution possible
ParaView is for Extremely Large Data

1 billion cell asteroid detonation simulation

½ billion cell weather simulation

source: Sandia National Lab

Data Parallel Pipeline

Data
Server

Read
Isosurface
Read
Isosurface
Read
Isosurface
Read
Isosurface

Render
Sort-last
Composite
Render
Sort-last
Composite
Render
Sort-last
Composite
Render
Sort-last
Composite

Client

Display
ParaView’s running modes

<table>
<thead>
<tr>
<th>Running Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Built-in aka Standalone aka Serial</strong></td>
<td>all components within one process (client may be GUI or pvpython) “paraview”</td>
</tr>
<tr>
<td><strong>Combined Server</strong></td>
<td>Data processing and parallel rendering in MPI job of combined processes. control from TCP connected client. “mpiexec -n x pvserver &amp;; paraview”</td>
</tr>
<tr>
<td><strong>Batch</strong></td>
<td>Server is an MPI job which directly runs a python script “mpiexec -n x pvbatch \ vis_script.py”</td>
</tr>
<tr>
<td><strong>Split Server</strong></td>
<td>Data processing and parallel rendering are both MPI jobs. “mpiexec -n x pvdataserver &amp;; \ mpiexec -n y pvrenderserver &amp;; \ paraview” #+ Connect</td>
</tr>
</tbody>
</table>
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](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:Build_And_Install)
    • [http://paraview.org/Wiki/ParaView/Superbuild](http://paraview.org/Wiki/ParaView/Superbuild)

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
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

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
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
    - [http://www.paraview.org/Wiki/ParaView/ParaView_And_Mesa_3D](http://www.paraview.org/Wiki/ParaView/ParaView_And_Mesa_3D)

Depth Compositing
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

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
Visualization on Big Iron

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

- **Unattended/Batch Session**
  
  ```bash
  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**
  
  ```bash
  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
  ```

---

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
Interacting with the data

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

Path from laptop to compute node
Establishing the connection

- **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:
  ```bash
  qsub reserve_then_tunnel2_backto_login_and_run_server.sh
  ```

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:
  ```bash
  qsub reserve_then_tunnel2_backto_login_and_run_server.sh
  ```

- [http://paraview.org/Wiki/Reverse_connection_and_port_forwarding](http://paraview.org/Wiki/Reverse_connection_and_port_forwarding)
.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

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
Exercise : interactive vis on Tukey

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

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

Local view of remote data

- File->Open : always shows you data server’s file system
- 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
### 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
  - surface polygons > threshold
- Client side rendering
- Root node of server gathers polygons
- Sends polygons to Client
- Client renders polygons
- Communicates only when data changes
- 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**

### Level of Detail – to maintain interactivity

**Type 1: Spatially based**

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

  Down-samples geometry while interacting
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

User interface

Menu Bar
Toolbars
Pipeline Browser
Object Inspector
View(s)
File->Open

- Kitware formats: (.vtk, .pvd, .vti, .pvt)
- Exodus
- Xdmf annotated hdf5 (.xml, .xdmf)
- netCDF CF (.ncdf, .nc)
- SpyPlot CTH
- EnSight (.case, .sof)
- 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

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
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

Display the data

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

- Views – Windows onto one or more data sets

Manipulate the data

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

- Calculator
- Glyph
- Contour
- Stream Tracer
- Clip
- Warp (vector)
- Slice
- Group Datasets
- Threshold
- Extract Group
- Extract Subset
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

ParaView Architecture

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

- VTK Pipeline, in parallel, on remote server(s),

- controlled by and feeds into client application.

Multi-View visualization pipeline
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

• SMP and GPGPU acceleration
  – http://www.daxtoolkit.org/index.php/Main_Page
  – Inria&EDF vtkSMP EGPGV 2013

From Tera- to Exa-scale
http://catalyst.paraview.org
Reduced File IO Costs

<table>
<thead>
<tr>
<th>Time of Processing</th>
<th>Type of File</th>
<th>Size per File</th>
<th>Size per 1000 time steps</th>
<th>Time per File to Write at Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post</td>
<td>Restart</td>
<td>1,300 MB</td>
<td>1,300,000 MB</td>
<td>1-20 seconds</td>
</tr>
<tr>
<td>Post</td>
<td>Ensight Dump</td>
<td>200 MB</td>
<td>200,000 MB</td>
<td>&gt; 10 seconds</td>
</tr>
<tr>
<td>In Situ</td>
<td>PNG</td>
<td>.25 MB</td>
<td>250 MB</td>
<td>&lt; 1 second</td>
</tr>
</tbody>
</table>

Ex. Helicopter simulation output size for a single time step
- Full data set – 448 MB
- Surface of blades – 2.8 MB
- Image – 71 KB
From Tera- to Exa-scale
http://catalyst.paraview.org

Traditional Vis

Simulation

Disk Storage

ParaView

Results

Simulation

"miniDEPV" 1/8^n clone?

Extracts

Scripts to generate your own subsets

“Editions” - selected subsets of paraview

Base - bare minumum
+ essentials - writers
+ python - scripting
+ rendering

The Catalyst Diet
http://www.kitware.com/blog/home/post/631
Catalyst: Access More Data

Post Processing

In situ Processing

Dump Times

Roughly equal data stored at simulation time
Reflections and Shadows added in post-processing

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
Thank you!

• **www.kitware.com**
  – portal to all communities kitware referees
  (p.s. we are always hiring)
  [http://jobs.kitware.com/opportunities.html](http://jobs.kitware.com/opportunities.html)

• **www.paraview.org**
  – portal to all things ParaView

• Please contact us, we are here to help.