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

• Session 2:
  – Hands on practice using ParaView at ANL
• 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
Software Process

- Kitware’s software R&D foundation
- Experience managing small projects through large open-source collaborations
- CMake toolset: configure, build, test, report on, and package projects on multiple platforms
  - Used by thousands of software teams; over 3,000 downloads per day
Computer Vision

- Detection & Tracking
- Recognition by Function
- Content-based Retrieval
- Images, Video, Point Clouds
- Event & Activity Recognition
- Anomaly Detection
- 3D Extraction and Compression
- Normalcy Modeling and Anomaly Detection (DARPA PANDA and PerSEAS)
- Football Play Recognition (DARPA CARVE)
- Wide-area Monitoring Imagery Threat Detection and Nodal Analysis (DARPA PerSEAS)
- Complex Event Recognition in Internet Videos (IARPA Aladdin)
- Content-based Video Retrieval by AcMons (DARPA VIRAT)
- 3D model-based video compression (DARPA GRID) and super-resolved 3D reconstruction (DARPA Super 3D)
Medical Computing

- Surgical guidance and simulation
- Interactive medical applications and visualizations
- Vascular analysis
- Digital pathology
- Orthopedic analysis
- Quantitative imaging
- Longitudinal and population shape analysis
- Electronic health records
HPC & Visualization

Massive data visualization

1 billion cell asteroid simulation

½ billion cell weather simulation

Mobile visualization

Large displays and virtual reality

Computational Chemistry

Simulation

Web visualization

Introducing KiwiViewer
Explore geometric datasets on your multi-touch mobile devices

Co-processing

Setup co-processing
Run simulation with co-processing
Visualize extracts

Visualize at run-time
Reconfigure co-processing
What is ParaView?

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

• 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 ~3 months
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
What to expect from parallel ParaView?

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

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

1 billion cell asteroid detonation simulation

½ billion cell weather simulation

source: Sandia National Lab
N component Data Parallelism for X GByte

Reader
- White Box
- Contour

Reader
- White Box
- Contour

Render Server
- Depth Composite

Client
- Control, Display and Rendering of Small Data

http://www.paraview.org/Wiki/ParaView/
ParaView_Readers_and_Parallel_Data_Distribution
## ParaView’s running modes

<table>
<thead>
<tr>
<th>Built-in aka Standalone aka Serial</th>
<th>DS</th>
<th>RS</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Combined Server</strong></td>
<td>DS</td>
<td>RS</td>
<td>Client</td>
</tr>
<tr>
<td><strong>Batch</strong></td>
<td>DS</td>
<td>RS</td>
<td>Client</td>
</tr>
<tr>
<td><strong>Split server</strong></td>
<td>DS</td>
<td>RS</td>
<td>Client</td>
</tr>
</tbody>
</table>

### Built-in aka Standalone aka Serial
- All components within one process (client may be GUI or pvpython)
  - “paraview” || “pvpython”

### Combined Server
- Data processing and parallel rendering in MPI job of combined processes. Control from TCP connected client.
  - “mpiexec –n x pvserver &; paraview” || “pvpython” #+ Connect

### Batch
- Server is an MPI job which directly runs a python script
  - “mpiexec –n x pvbatch \ vis_script.py”

### Split server
- 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
• 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 (Fedora example 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
- ParaViewSuperBuild git://paraview.org/ParaViewSuperbuild.git does so for Cray and BlueGene
- PV 4.1 will have frozen python (NO disk access at all)
Static python linking

- A simple python module example

```python
import mymodule
mymodule.say_hello()
```

- Pure python module: mymodule.py
- Extension module: mymodule.so
  - mymodule.so is dynamically loaded
  - contains C function: void initmymodule()
static python linking

#include <Python.h>

int main() {
    Py_Initialize();
    const char* code = "import mymodule
                        mymodule.say_hello()
";
    PyRun_SimpleString((char*)code);
    Py_Finalize();
}

Not linked to mymodule.so, will use dlopen at runtime to load mymodule.so
Static python linking

```c
#include <Python.h>
extern void initmymodule();

int main() {
    PyImport_AppendInittab("mymodule ", initmymodule);
    Py_Initialize();
    const char* code = "import mymodule \n" "mymodule.say_hello() \n";
    PyRun_SimpleString((char*)code);
    Py_Finalize();
}
```

When compiled statically uses symbols that it found in mymodule.a.
Static python linking

- Use static linking to avoid shared library dependencies

- Must compile python statically
  - configure –disable-shared
  - libpython2.6.a instead of libpython2.6.so

- Python standard library includes many extension modules
  - want to compile math.c into libpython2.6.a, instead of producing math.so
  - So, after configuring python, edit file Modules/Setup to indicate which modules should be included as part of libpython2.6.

- Or, use CMake! CMake handles static extension modules and cross compiling
  - Python build in ParaViewSuperbuild patches to cmakeify then doe this
pvbatch example

parallelSphere.py

from paraview.simple import *
numProcs = paraview.servermanager.
    ActiveConnection.
        GetNumberOfDataPartitions()
sphere = Sphere()
sphere.ThetaResolution=512+numProcs
pids = ProcessIdScalars()
rep = Show()
lut = GetLookupTableForArray(
    "ProcessId",1,
    RGBPoints=[0.0, 0.23, 0.299, 0.754, 0.706, 0.016, 0.15],
    ColorSpace='Diverging')
rep.LookupTable = lut
rep.ColorArrayName = 'ProcessId'
Render()
WriteImage("parallelSphere.png")

How to Learn ParaView Scripting

1. http://paraview.org/Wiki/ParaView/Python_Scripting

2. Tools -> Python Shell

3. Trace:
   ParaView -> Tools -> Start Trace
   Do something in GUI
   ParaView -> Tools -> Stop Trace
   File -> Save
Visualization on Big Iron

• Offscreen rendering useful/essential
  OSMesa easy option: see pure offscreen rendering section of http://paraview.org/Wiki/ParaView/Users_Guide/Parallel_Rendering

• Environment search paths
  – Where ParaView binaries, libraries and python can be found
    mypaths.sh
    set dir = /lustre/widow2/scratch/demarled/pv3.14.0/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-3.98.1
    Or perhaps
    module load paraview-3.98.1
Visualization on Big Iron

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

• Interactive Session
  
  qsub -I \n  -q debug -A yourProject \n  -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

• Unattended Session
  
  qsub -v PYTHONHOME=${dir}/pythonhome ... \n  -q batch -A yourProject \n  -l size=1600 -l walltime=03:00:00 \n  aprun -n 1600 pvbatch parallelREALLYHUGESphere.py
Demonstration of batch visualization

- connect to (ssh -R) titan.alcf.ornl.gov
- reserve some nodes
- set paths
- mpiexec pvbatch
- send result back and view them
From Tera- to Exa-scale
http://catalyst.paraview.org

Traditional Vis

Simulation

Disk Storage

ParaView

Results

Simulation

Catalyst

Disk Storage

Results

Co-Processing

Simulation

Catalyst

Disk Storage

Results

Separate MPI

Catalyst
Catalyst: Access More Data

Post Processing

In situ Processing

Roughly equal data stored at simulation time

*Reflections and Shadows added in post-processing*
Interacting with the data

- Interaction is sometimes important to understand data
- Establish a connection to control ParaView server from your desktop
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 - you start the server job however you want
  - 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
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](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>

• File can be moved, edited, emailed, restored manually
• 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://paraview.org/Wiki/ParaView:Server_Configuration
Demonstration of live connection

- Use port forward and sub script to connect to titan
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

- Preferences/Settings -> Render View -> Server -> Client/Server Parameters -> Presets
OpenGL on a Cray? (are you nuts!)

- Typically use OSMesa compiled statically
  - Configure ParaView to use it as a pure offscreen rendering solution

- 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

- See ParaView Guide Parallel Rendering chapter
  - for instructions and recommendations on either case
Depth Compositing
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
Where to go to for more information

www.paraview.org portal to all things ParaView

- Wiki Page
  - http://www.paraview.org/Wiki/ParaView

- Mailing List
  - Sign up->http://public.kitware.com/mailman/listinfo/paraview
  - Search ->http://markmail.org/search/?q=list:paraview

- User Voice - feature request voting
  - “Tell us what you think” link
  - http://paraview.uservoice.com/forums/11350-general

- Source Code Documentation

- Support
Big Data Analysis with ParaView

Going beyond the built in server
Session 2: Hands on Practice
Hands On Practice

Get comfortable using ParaView at ORNL/ANL Using both modes batch and interactive

• General Use
  1. Create viz pipeline with GUI on builtin server

• Batch
  1. Recreate viz pipeline in a python script
  2. Run as batch job on HPC server

• Interactive
  1. Connect GUI to local server
  2. Connect to HPC server interactively
Exercise: General Use #1
Create viz pipeline with GUI on builtin server

You will learn how to:
- Start ParaView
- Open disk_out_ref exodusII file
- What have we got?
- Slice along Z and warp slices to show vector field
- Insert Streamlines to show vector field in another way
- Save screen shot

See also: http://vimeo.com/41009606
User Interface

- Menu Bar
- Toolbars
- Pipeline Browser
- Object Inspector
- View(s)
Multi-View visualization pipeline
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 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

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

Points  Wireframe  Surface  Surface with Edges  Volume
Display the data

- Views – Windows onto one or more data sets
Now go ahead and do it

1. Start ParaView
2. File -> Open disk_out_ref.ex2
3. Information Tab : What have we got?
4. Slice it along Z
5. Warp the slices
6. Insert Streamlines
7. File -> Save Screenshot
Exercise: Batch #1
Recreate viz pipeline as python script

You will learn how to:

- Create a python script that sets up a viz pipeline
- Play it back on local machine

See also: http://paraview.org/Wiki/ParaView/Python_Scripting
Two Options: State or Trace

**State File**
- Record of current pipeline
- File -> Save State -> filename.pvsm
- Load within python script

```python
myfile.py:
servermanager.LoadState(filename.pvsm)
SetActiveView(GetRenderView())
Render()
```
- `pvbatch myfile.py`

**Trace File**
- GUI actions translated to python
- Tools -> Start Trace
- Do something
- Tools -> Stop Trace
- File -> Save myTrace.py
- `pvbatch recordedTrace.py`
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.
Building a Pipeline

• Branch by changing the active source, like choosing in GUI’s PipelineBrowser

```python
>>> SetActiveSource(aReader)

>>> aST = StreamTracer()
>>> aST.Vectors = "Momentum"
>>> aST.SeedType.Center = [3,2,28]
>>> aST.SeedType.Radius = 2
>>> aST.SeedType.NumberOfPoints = 100

>>> Show(aContour)
>>> Show(aST)
>>> Render()
```
Navigating the Pipeline

- Don’t have to use active source to branch, can assign at creation
- Can change after the fact
- Can inspect ActiveSource
- Can get a hold of all or any particular SourceProxy

```python
>>> aST = StreamTracer(Input=aReader)

>>> aST = StreamTracer()
>>> aST.Input = aReader
>>> aST.Input = aClip

>>> aSource = GetActiveSource()

>>> GetSources()
>>> someSource = FindSource("Contour1")
```
Multi-View visualization pipeline

- Can show output of any SourceProxy

- Parallel Flexible Display Pipeline complexity encapsulated by “Representations” in “Views”

- **Representation** – visual qualities of an output ≈ Mapper + Actor + parallel transport. Show() returns a Representation

- **View** - Visual qualities of a window ≈ Renderer + Camera + Lights + RenderWindow. Render() returns a View

- To make it easier to build, commands default to working with the Active Representation and Active View
Controlling Display

- Many methods take ActiveRepresentation and ActiveView as default arguments
- But can get hold of and then control any particular View and Representation
- Screen Shots

```python
>>> activeSourcesRep = GetDisplayProperties()
>>> clipFiltersRep = GetDisplayProperties(aClip)
>>> clipFiltersRepInMyView = GetDisplayProperties(aClip, myView)

>>> GetRenderViews()
>>> view0 = GetRenderViews()[0]
>>> allReps = view0.Representations()
>>> rep0_0 = allReps[0]

>>> WriteImage(filename, view=ActiveView, Magnification=0.0)
```
pvbatch

- Run a recorded paraview script under pvbatch
- Use mpi to spawn it as a parallel job

```
[mpiexec -N <numprocessors> [other-args-for-mpi]] \ pvbatch [args-for-pvbatch] \ script-filename [args-for-script]
```

mpiexec to run a parallel program
pvbatch to run offline paraview script processor
script-filename.py – your paraview python script
Now go ahead and do it

1. Restart ParaView
2. Tools -> Start Trace
3. Recreate Vis pipeline
4. Tools -> Stop Trace
5. Save -> mytrace.py
6. Edit mytrace.py and add Save Image Command at End
   \[
   \text{WriteImage(filename, view=ActiveView, Magnification=0.0)}
   \]
7. Play it back under pvbatch
   \[
   \text{pvbatch mytrace.py} \]

Exercise: Batch #2
Run batch job on big iron

You will learn how to:
– Run paraview on HPC server
Visualization on Big Iron

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

- **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
  ```

- **Unattended Session**
  ```
  qsub -v PYTHONHOME=${dir}/pythonhome ... \\
  -q batch -A yourProject \\
  -l size=1600 -l walltime=03:00:00 \\
  aprun -n 1600 pvbatch parallelREALLYHUGESphere.py
  ```
Integration with system scheduler

Encapsulate command to run pvbatch in pbs submission:

```
qsub –N jobname –l environment_spec –j eo –e logfilename.txt mpiexec ...
```

Or

```
qsub shellscript.sh
```

where shellscript.sh =

```
#PBS –N jobname -l environment_spec ...
module load paraview
mpiexec –N .... pvbatch ... script-filename ...
```
Now go ahead and do it

1. scp data file and script to server
   ssh -R 99999 localhost:22 ruser@hostname
   cd /somewhereonlustre
   scp -P 99999 -r luuser@localhost:~/mywork

2. Edit demoscript.pbs, make it run mytrace.py
3. Edit mytrace.py, tell it to load disk_out_ref.ex2 from correct directory
4. qsub demoscript.pbs
5. scp -P 99999 result.png ~/mywork
Exercise: Interactive #1
Connect GUI to local server and use it

You will learn how to:
- Start up paraview interactive server
- Connect paraview GUI client to it
- Control remote processing parameters
Connecting to a Server

- builtin – the server within the client process

- see: http://www.paraview.org/Wiki/Setting_up_a_ParaView_Server
- For information on configuring a server cluster
Connecting to a Server

- File->Connect
- First time, Add Server
  - Name this connection to reuse it later
  - Client/Server most common
  - Host, Port = IP address of a machine to run pvserver on
  - Manual:
    - if it is already running or you prefer to start it by hand
- or
  - Command:
    - a shell command to start pvserver on that machine
      
      ```
      ssh -R 22222:local_machine:11111 xterm -e script.sh &
      Where script.sh contains:
      qsub reserve_then_tunnel2_backto_login_and_run_server.sh
      ```

Thereafter just choose the connection you set up above
Now go ahead and do it

• Connect GUI to local server and use it
  1. Start pvserver
     pvserver & (or mpirun pvserver & if you’ve got it locally)
  2. Start paraview GUI
     paraview
  3. Connect “localhost”
     File->Connect
     Add Server
     Name = “localhost”, Configure
     Type = Manual, Save
     Select “localhost”, Connect
  4. Create visualization
Now go ahead and do it part2

- Connect GUI to local server and use it
  1. Start pvserver
  2. Start paraview GUI
  3. Connect “localhostAUTO”
     File->Connect
     Add Server
     Name = “localhostAUTO”, Configure
     Type = Command, Save
     Command = /path/to/pvserver/pvserver &
     Select “localhost”, Connect
  4. Create visualization
Exercise: Interactive #2
Connect GUI to remote server and use it

You will learn how to:
– Spawn a remote pvserver session
– Connect the local GUI to it through reverse connection tunnels
Path from laptop to compute node

desktop

login node

service node

compute node 0

desktop

login node

compute node 0

desktop

login node

service node

compute node 0
PVSC

- Your connections settings are stored in a servers.pvsc config file
- a text file: edit it, save it as a backup, replace it
- OPTION entries (username for example) changeable via GUI

- Since 3.14 you can import servers entries from the GUI
  - 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

- Only sysadmins have to go to trouble of establishing the path

- Trying to post connections to all public HPC centers we know
Now go ahead and do it

- Connect GUI to remote server and use it
  1. Start ParaView GUI
  2. File->Connect, Fetch Servers
  4. Select “eureka”
  5. Fill in your username and project
  6. Save
  7. Connect
  8. Create visualization
Thank you!

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

- [www.paraview.org](http://www.paraview.org)  
  - portal to all things ParaView

- Please contact us, we are here to help.