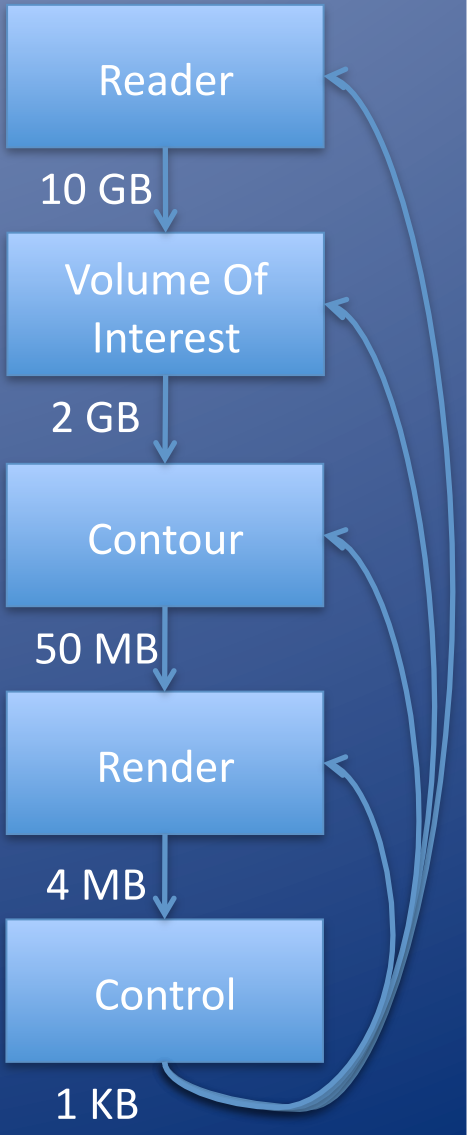
ParaView on Vis Clusters

David E DeMarle

Kitware Inc.

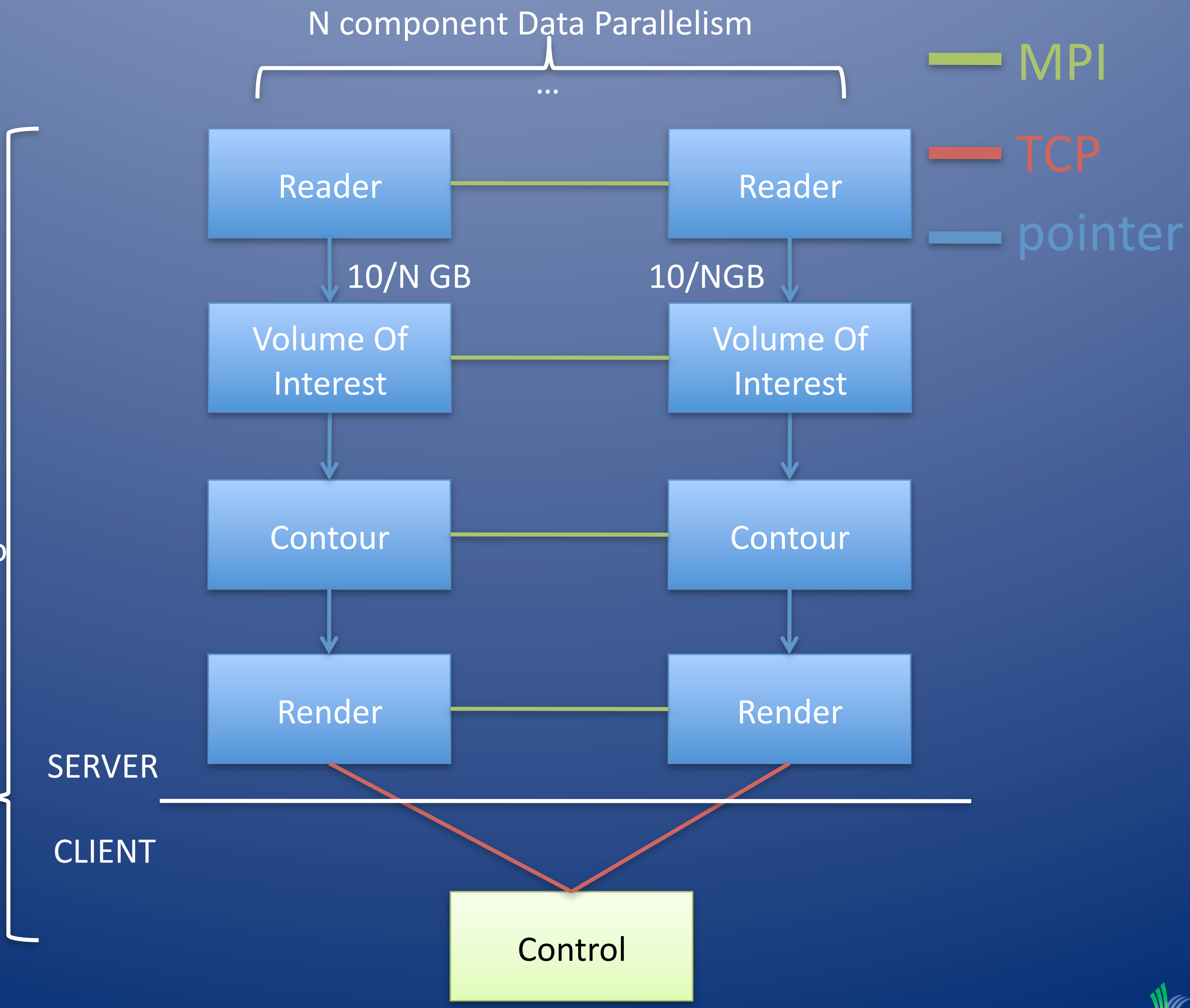
Visualization

- Most often, a process of reduction. Goal is to find the important information within the whole, or distill out characteristics of the whole
- Since data is large, ParaView uses functional AND data parallelism to scale (in terms of achievable size)



Server runs on a cluster and does the hard work. Client connects to that and makes it convenient to use.

2 component Functional Parallelism



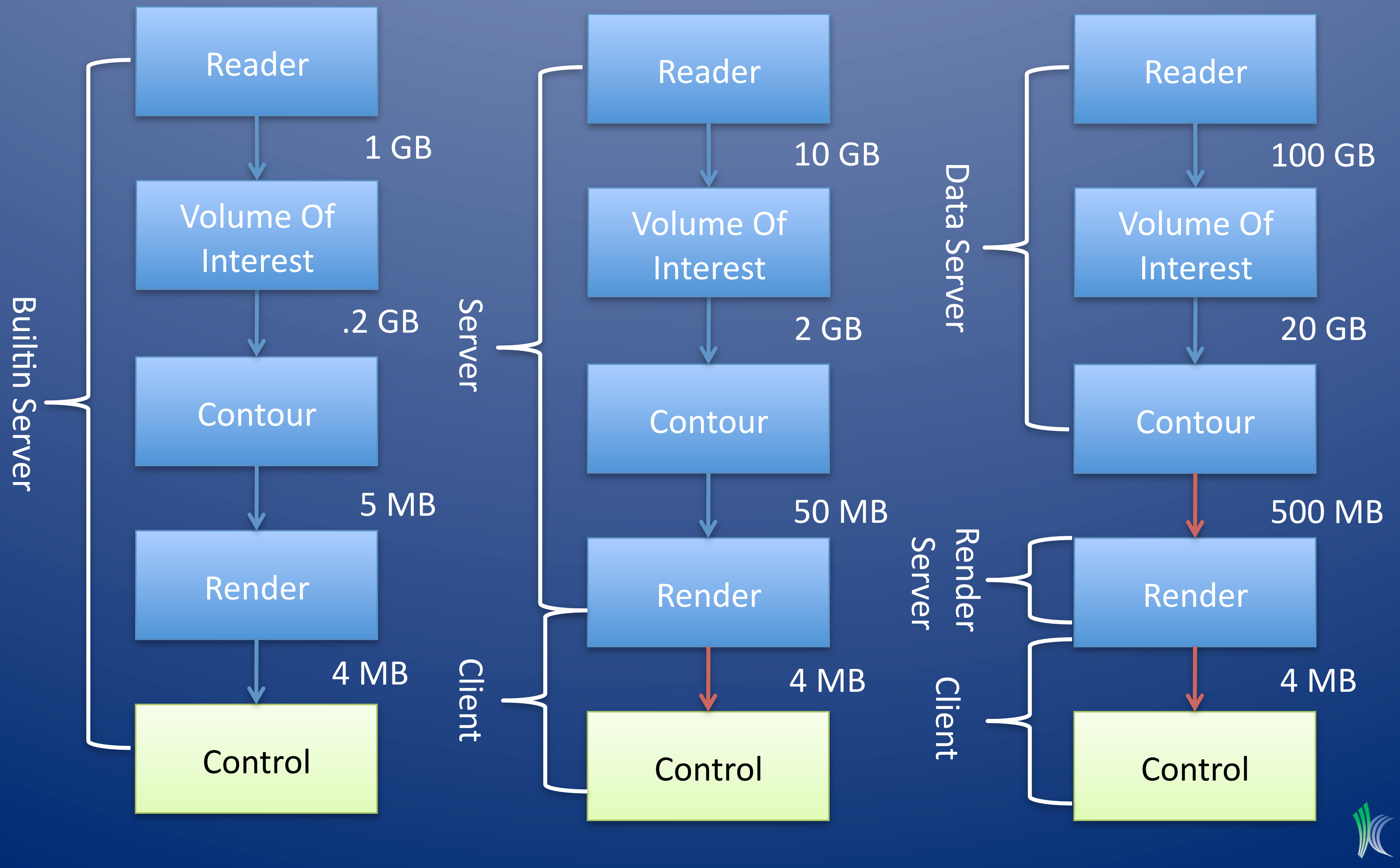
Configurations

- Limited to working with data that fits into aggregate memory*
- Functional decomposition lets you match data size to machine resources
- ParaView supports a number of configurations
- Depending on configuration, different libraries are needed, on each machine

* Streaming ParaView experimental application is a notable exception

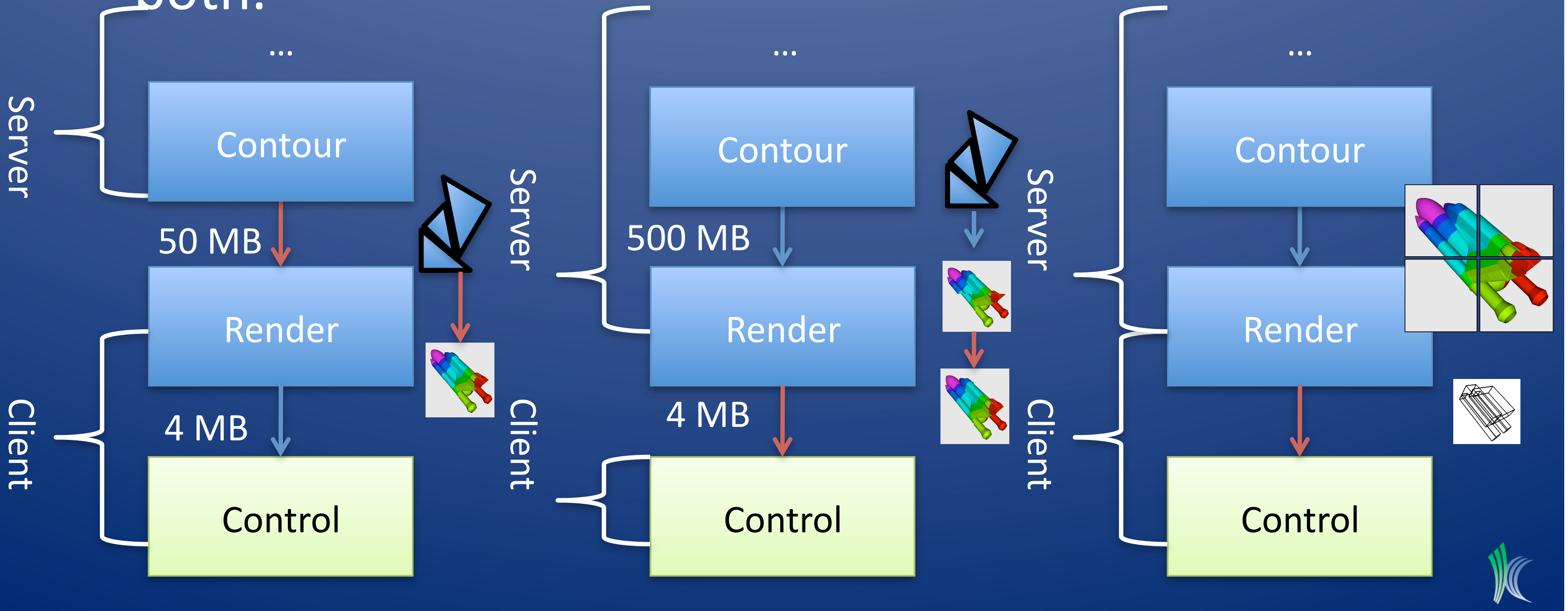
Configurations

Note: 3 component functional parallelism



Rendering

- Depending on renderable geometry size, ParaView will dynamically render locally (sending geometry) or remotely (sending images). Or, with tiled display, both.



Libraries Required

- MPI$_{\text{almost}}$ always needed on server never on Client
- TCP needed everywhere, except when batch processing on server (Cray, etc)
- Qt$_{\text{almost}}$ always needed on client, never on server
- OpenGL
  - always needed on client and renderserver, not necessary on data server
  - does not imply need graphics hardware (or even display) Mesa and OSMesa are widely used http://mesa3d.org

Machine Requirements

- Processors
  - CPUs Minimal?
    - netbook OK for client and processing of small data
  - GPUs Minimal?
    - Mesa OK, none required
    - it will take full advantage of advanced GPU if available

Machine Requirements

- Memory
  - Restricted to data that can fit in aggregate RAM
  - Data parallelism replicates pipeline N times
  - Each cluster node works on 1/Nth (+ a little)
  - Need at least as much as file size, plus enough for each filter's output
  - Information Tab shows each filter's output size, but much of each filter's output is a copied by reference of its input's, so sum is <

Machine Requirements

- DISK
  - Each reader needs to see files
  - Files shown in file browser are on server's file system
  - Well written readers (Exodus, XDMF) read only local part
  - "dumb" readers read all everywhere, then crop

  - Replication – works and minimizes contention, but a waste of disk space and prep time
  - NFS – better, but potential bottleneck when all nodes read simultaneously
  - Parallel file systems – PVFS, LUSTRE, etc – more bandwidth, better performance

Machine Requirements

- Interconnect hardware
  - Intent of data parallel architecture is to minimize inter-process communication
  - Still, the faster the better. Works well on 100MB.
- MPI : on server(s)
  - most implementations are fine
  - openmpi, mpich, or vendor supplied MPI for Myrinet, Quadrics, Infiniband, SCI, etc
- TCP : between server and client and data and render server
  - Not needed at all in clientless batch mode
  - About firewalls:
    - pvserver --reverse_connection --client-host clientIPaddr
    - pvserver --server-port #tell it what port to wait on
    - consider vpn or ssh port forwarding through firewalls

```
ssh –L lport:destIP:destPort
ssh localhost:lport \
    mpirun –np N pvserver
```

Machine Requirements

- Remote login and program execution
- Without typed password
- ssh authentication
  - users copy ssh pub key to their login on each node
  - exec ssh-agent $SHELL
  - ssh-add <type your key once locally>
  - thereafter, ssh remotemachine command, does not prompt for password
- PATH : ssh command that runs on server needs to find pvserver executable (absolute path OK)

Display

- To take advantage of GPUs, server processes need local windows to create graphics contexts
- No X Forwarding! ("ssh –X" BAD)
- Two approaches:
  - Have users run X at log in

    ```
    > srun X:0 &
    ```
  - Always run X, disable X11 security
    - Make gdm auto login a dedicated X account
    - .xsession for that account runs blank X window and disables security (xhost +) so any user can map windows
- In either case

  ```
  mpirun -np 4 /bin/env DISPLAY=localhost:0 ./pvserver
  ```

  Or specify DISPLAY mapping in machines.pvx file (PV guide page p134)

No GPU? No problem!

Without GPUs on cluster two options:

- Make server do data processing only

  ```
  > pvserver --disable-composite
  ```

  tells server to always send geometry to client for rendering
  equivalent to unchecked(=infinite) Remote Render Threshold on preference dialog

- OSMesa

  Compile ParaView to know about OSMesa GL libs and

  ```
  > pvserver --use-offscreen-rendering
  ```

Compiling ParaView

- Why?
  - Kitware's binary releases do not link to MPI
  - Server beed MPI to do data parallelism
  - for client, binary release is fine
- Requirements
  - ParaView source code : http://www.paraview.org/paraview/resources/software.html
  - ParaView Data and VTKData useful for testing
  - CMake 2.6.4+ binary http://www.cmake.org/cmake/resources/software.html
  - A compiler : visual studio express, make and g++, etc
  - About an hour : 2 core 1.8GHz Intel CPU, 2GB RAM, virgin build

Compiling

1. create a build directory and enter it
2. ccmake (or cmake-gui) path_to_source
3. populate required options, configure
4. repeat step 3 until no new dependent options
5. generate to create build environment
6. make (or in VisStudio, build solution)
7. install

   Install is optional, wait till you get it working well then install it somewhere that everyone can see

Configuration Options (Server)

- PARAVIEW_BUILD_QT_QUI=OFF
- VTK_DATA_ROOT=location of VTK regression test data
- PARAVIEW_DATA_ROOT=location of ParaView regression test data

- If server will render (and defaults chosen are not acceptable)
  - OPENGL_INCLUDE_DIR = directory where GL/GL.h resides
  - OPENGL_gl_LIBRARY = location of libGL.so ex,
  - OPENGL_glu_LIBRARY = location of libGLU.so ex,

- To use pure software rendering, with no display at all,
  - VTK_OPENGL_HAS_OSMESA = ON
  - OSMESA_LIBRARY = location of libOSMesa.so
  - VTK_USE_OFFSCREEN = ON
  - start server with --use-offscreen-rendering

Configuration Options (Server)

- PARAVIEW_USE_MPI=ON
  - MPI_INCLUDE_PATH= directory where mpi.h is
    /ThirdParty/MPIs/openmpi-1.2.6-build/include
  - MPI_LIBRARY = location of libmpi.so
    /ThirdParty/MPIs/openmpi-1.2.6-build/lib/libmpi.dylib
  - MPI_EXTRA_LIBRARY* = location of libmpi_cxx.so
    /ThirdParty/MPIs/openmpi-1.2.6-build/lib/libmpi_cxx.dylib

* “;” separators in MPI_LIBRARY and MPI_INCLUDE_PATH allow any number of additional dependencies needed for your MPI (see mpiCC –showme to find out what they might be)

Validating Setup

- How to tell if it is configured right?
  - ssh machine “uname -a”

  Shouldn’t have to log in

  - mpirun -np 2 –machinefile “machines.txt” /usr/bin/uname –a

  Same as above, and should get multiple machine names back

  - mpirun -np 2 helloworld_mpi

  Should print out rank etc

Validating Setup

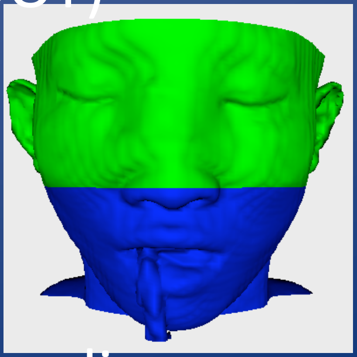
- How to tell if it is configured right?
  - mpirun -np 2 /bin/env DISPLAY=localhost:0 /usr/X11R6/bin/glxgears

  Windows should appear on remote displays, not locally and should spit out reasonable frame rates

  - VTK parallel tests (assuming VTK_DATA_ROOT)
    - ctest -R ParallelIso -V | grep command
    - `command` + -I, lets you interact

  Should get a two tone face on first node and a one tone, partial face on the second

  When you drag mouse shouldn't have bleeding lines (turn antialiasing off) or backside triangles in front (something is covering a window)



Validating Setup

- How to tell if it is configured right?
  - PV tests
    - ctest -I ,,10 run every tenth test to get sense of correctness

  Should have at least 95% success or something is drastically wrong. If using cvs head, check day’s dashboard, might be unlucky

  - When running ParaView and connected to a cluster, try “process id scalars” filter. It shows which processor generated what data.

Running

- Run server
  - mpirun -np N pvserver
  - Terminal should say "Listen on port: 11111 \n Waiting for client…"
- Run client
  - paraview
- Connect to server
  - File->Connect, add server, supply a nickname and hostname, configure, startup type to manual, save
  - Double click on nickname
  - Dialog box should say connected and disappear, pvserver terminal should say connected.
  - Pipeline browser: "cs://hostname:11111" instead of "builtin:"
- Now, optionally change to an automatic startup instead of manual
  - type in command that will log in to cluster and mpirun pvserver

Running

- Remote render threshold
- Edit->Settings->Render View->Server
  - Remote Render Threshold
    - geometry size at which PV switches from server sending geometry or images to client
    - unchecked means rendering always done on client
    - checked and set to 0 MB, then next render causes server to pop up windows (which should be on remote machine's display)
  - Subsample Rate
    - to maintain interactivity when remote rendering
    - how grossly are images down sampled,
    - only active while interacting and while server is rendering
    - drag mouse, everything pixelated
    - release mouse, returns to full resolution

Additional Resources

- **ParaView Guide chapter 13 and 14**
- **Wiki Page**

  General
  http://www.paraview.org/Wiki/ParaView
  Building
  http://www.paraview.org/Wiki/ParaView:Build_And_Install
  Cluster Setup
  http://www.paraview.org/Wiki/Setting_up_a_ParaView_Server
- **Mailing List**

  Sign up->http://public.kitware.com/mailman/listinfo/paraview
  Search ->http://markmail.org/search/?q=list:paraview
  - **Bug Tracker (**Project = ParaView3)

  http://www.paraview.org/Bug/my_view_page.php
- **Source Code Documentation**

  http://www.paraview.org/ParaQ/Doc/Nightly/html/annotated.html