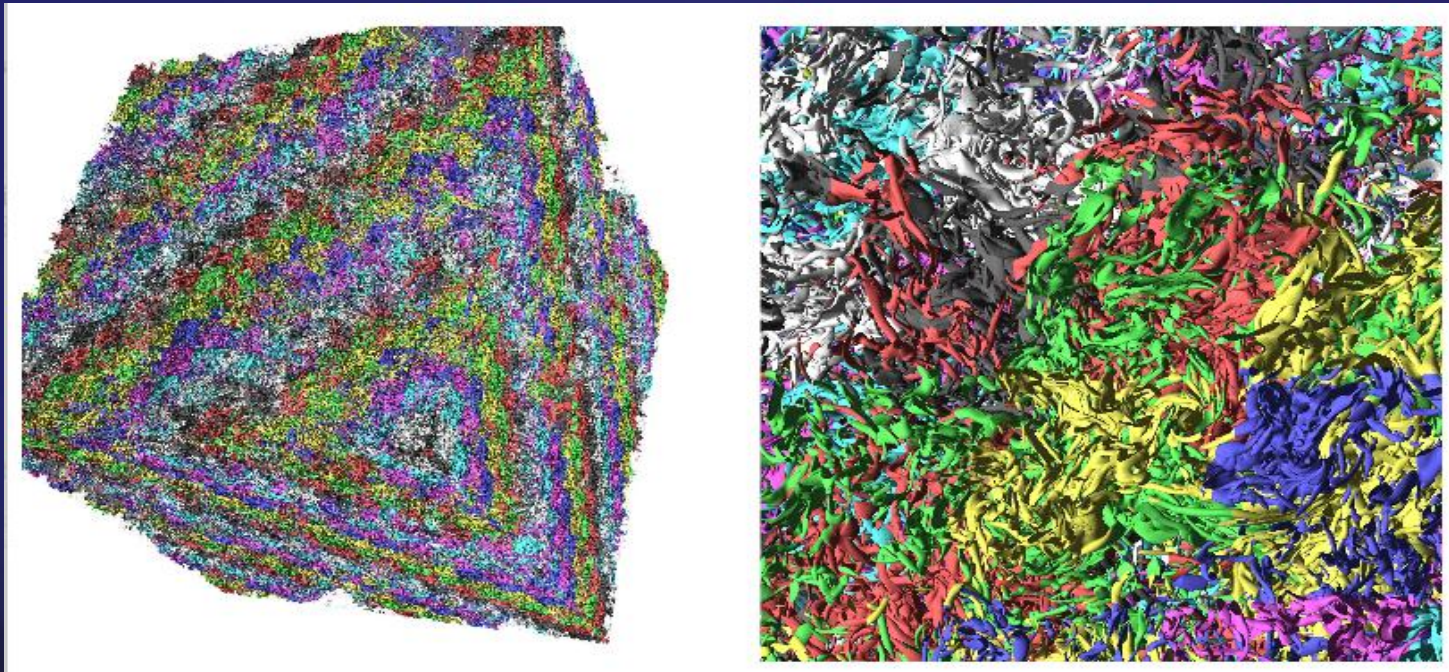
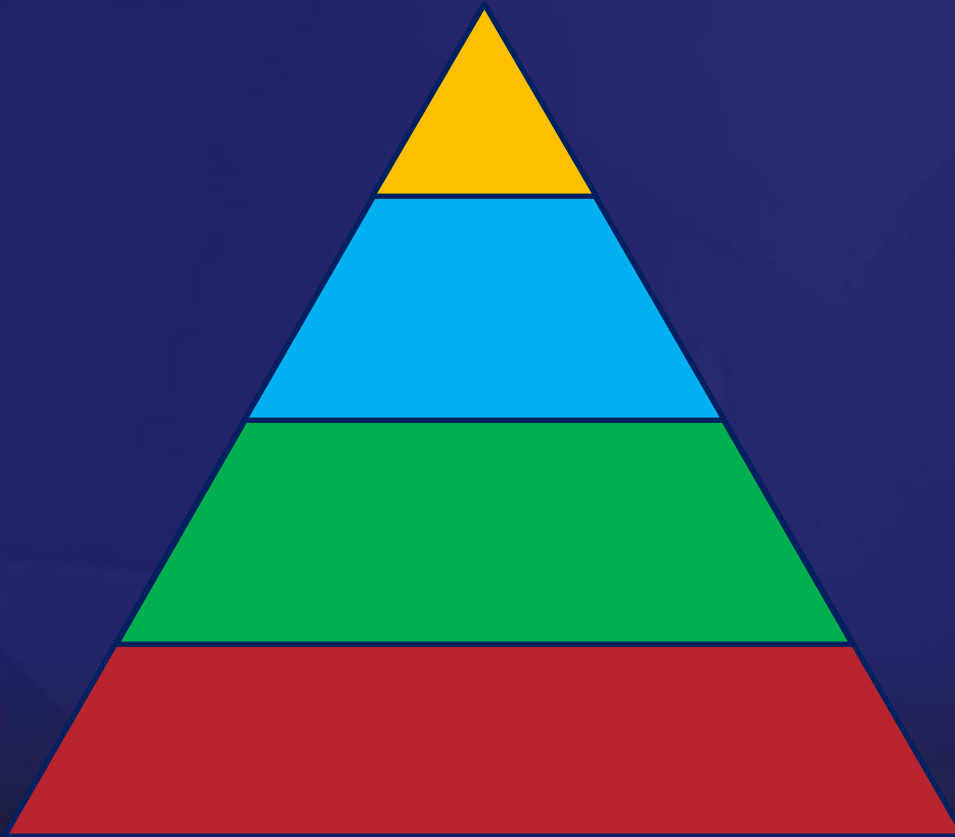


# Parallel Visualization At TACC

Greg Abram



# Visualization Problems



## Huge problems:

- Data cannot be moved off system where it is computed
- Visualization requires equivalent resources as source HPC system

## Large problems:

- Data are impractical to move over WAN, costly to move over LAN
- Visualization requires parallel systems for even more memory, lots of memory, fast CPU and GPU

## Medium problems:

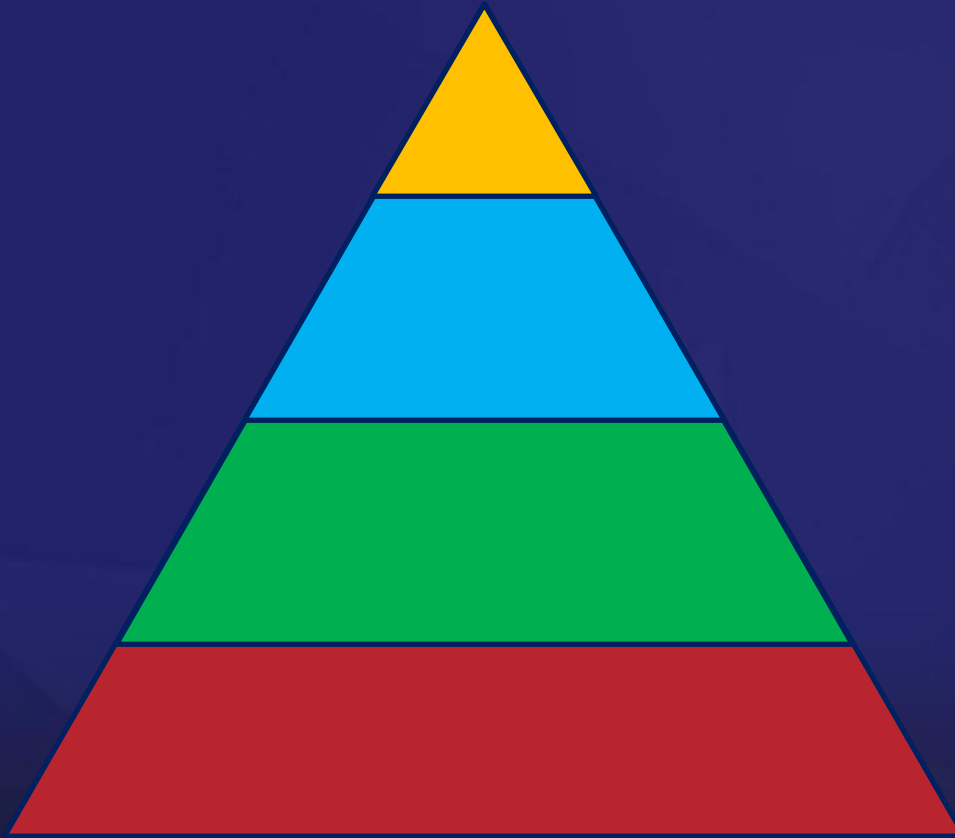
- Data are costly to move over WAN
- Visualization requires lots of memory, fast CPU and GPU

## Small problems:

- Data are small and easily moved
- Office machines and laptops are adequate for visualization

\* With thanks to Sean Ahern for the metaphor

# Visualization Problems



## Huge problems

Don't move the data; in-situ and co-processing visualization minimizes or eliminates data I/O

## Large problems

Move your data to visualization server and visualize using *parallel* high-performance systems and software at TACC

## Medium and small problems

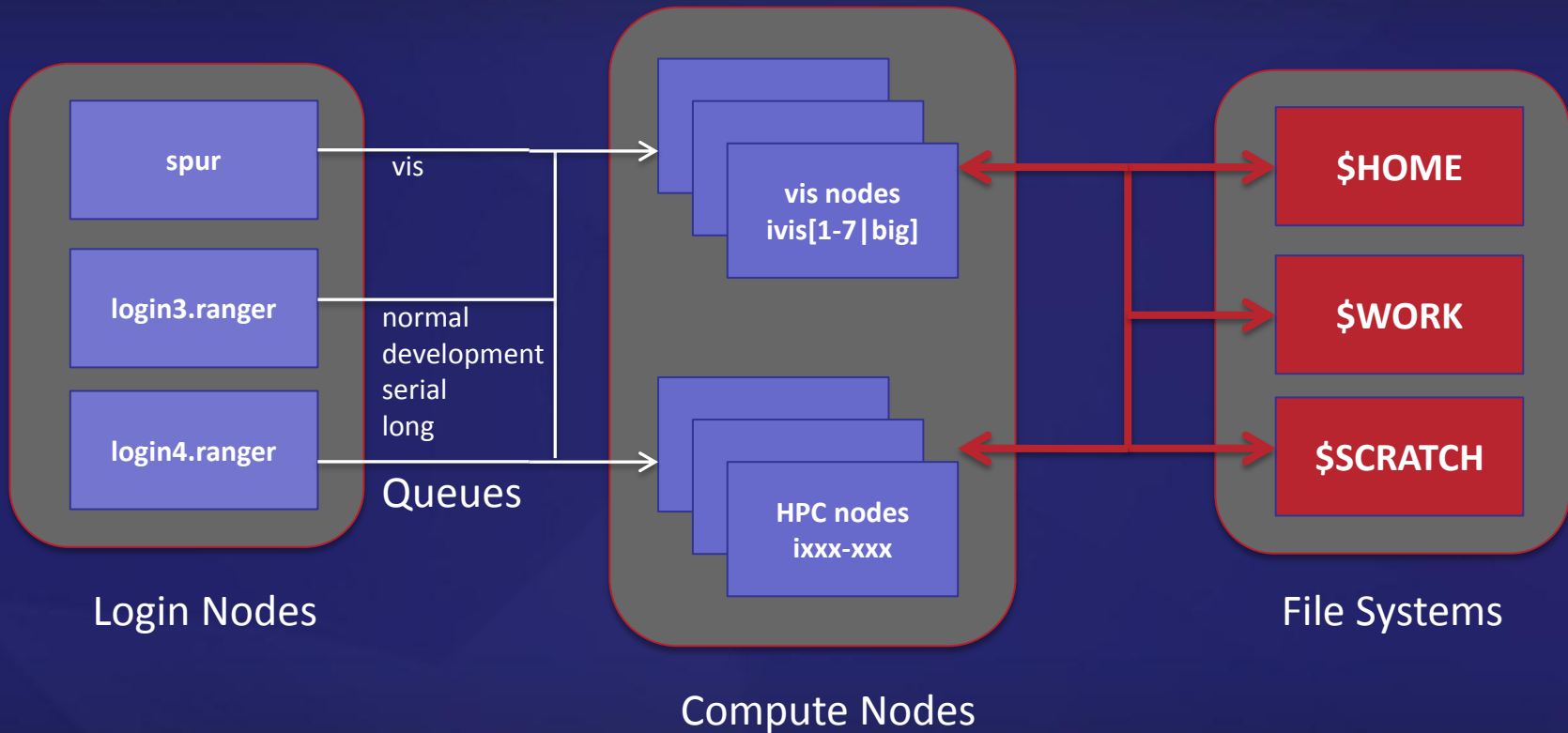
Move your data to visualization server and visualize using high-performance systems at TACC

\* With thanks to Sean Ahern for the metaphor

# Visualization Servers: Spur

- Really part of Ranger
  - Shares interconnect
  - Shares file systems
- 8 compute nodes:
  - visbig
    - 8 dual-core AMD Opteron
    - 256 GB
    - 4 NVIDIA FX 5600 GPUs
  - vis1-6
    - 4 dual-core AMD Opteron
    - 128 GB
    - 4 NVIDIA FX 5600 GPUs
- Access (see Spur User Guide):
  - Run `qsub job.vnc` on Spur
  - Use `vis` queue to allocate 1 or 2 nodes

# Spur Architecture



↔ Read/Write File System Access

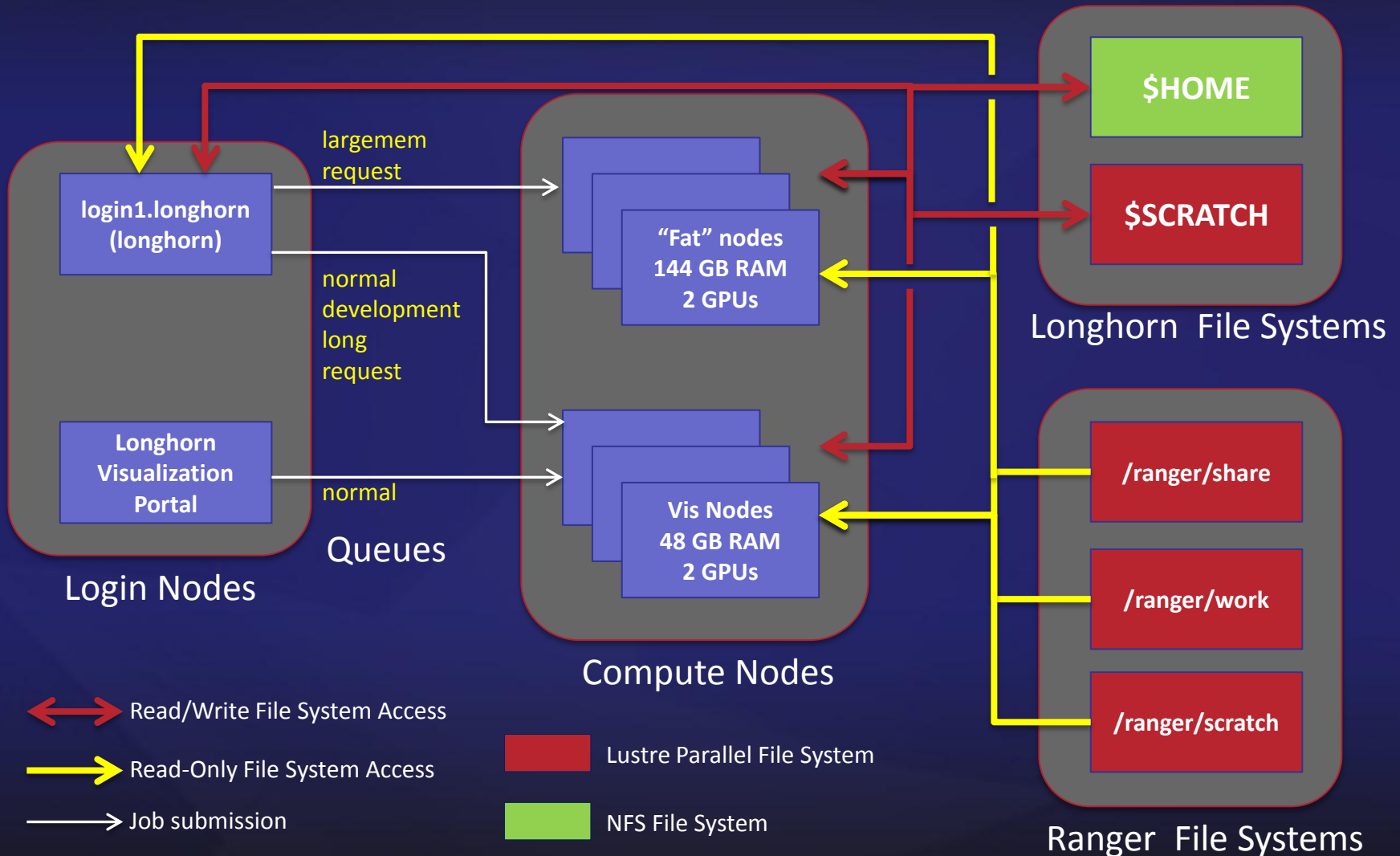
→ Job submission

■ Lustre Parallel File System

# Visualization Servers: Longhorn

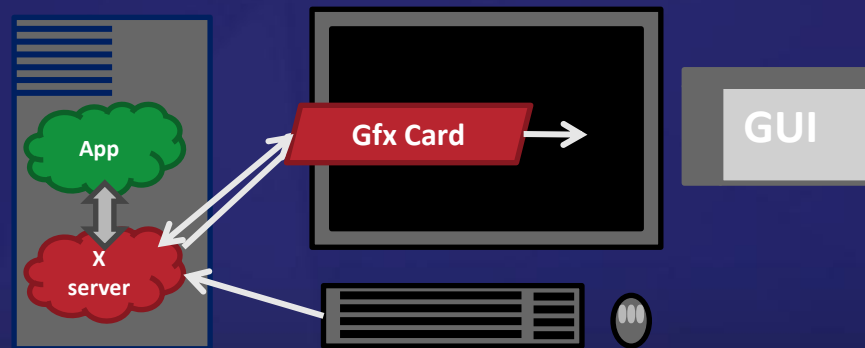
- Access to Ranger file systems (see Ranger User Guide)
- 256 compute nodes:
  - 16 “Fat” nodes -
    - 8 Intel Nehalem cores
    - 144 GB
    - 2 NVIDIA Quadro FX5800 GPUs
  - 240 standard nodes
    - 8 Intel Nehalem cores
    - 48 GB
    - 2 NVIDIA Quadro FX5800 GPUs
- Access (see Longhorn User Guide):
  - Longhorn Portal, or
  - Run `qsub job.vnc` on Longhorn using `normal`, `development`, `long`, `largemem`, `request` queues

# Longhorn Architecture



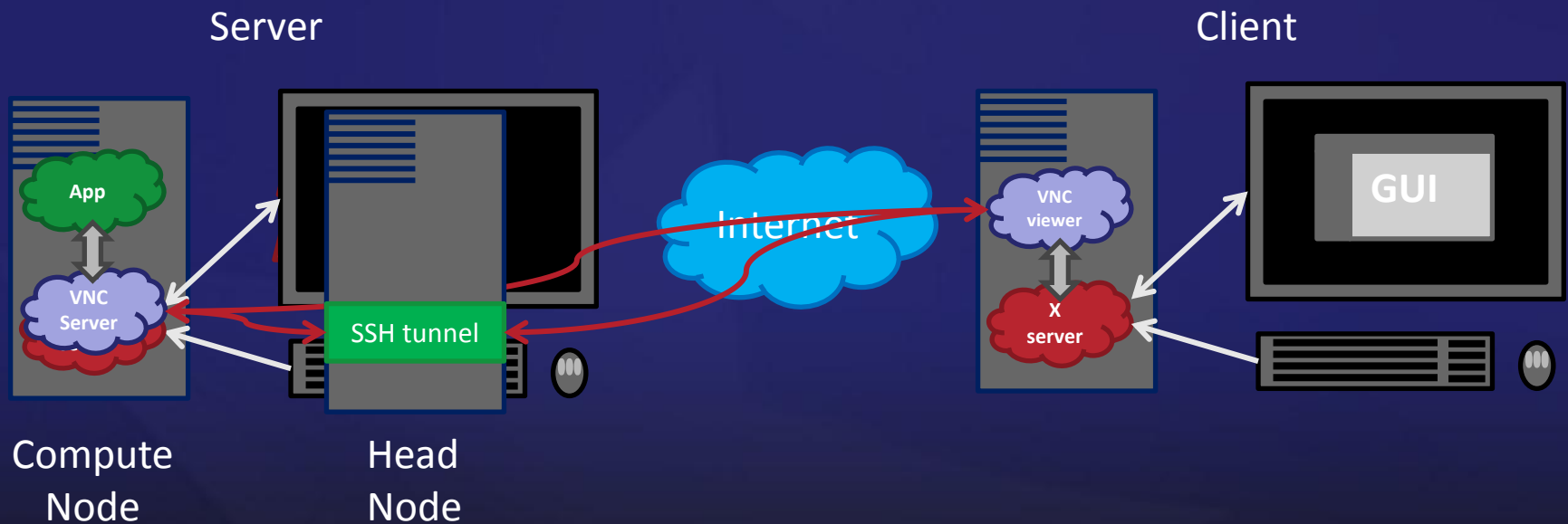
# Intro to Remote Viz

- (Full details tomorrow)
- Linux/X11 Applications



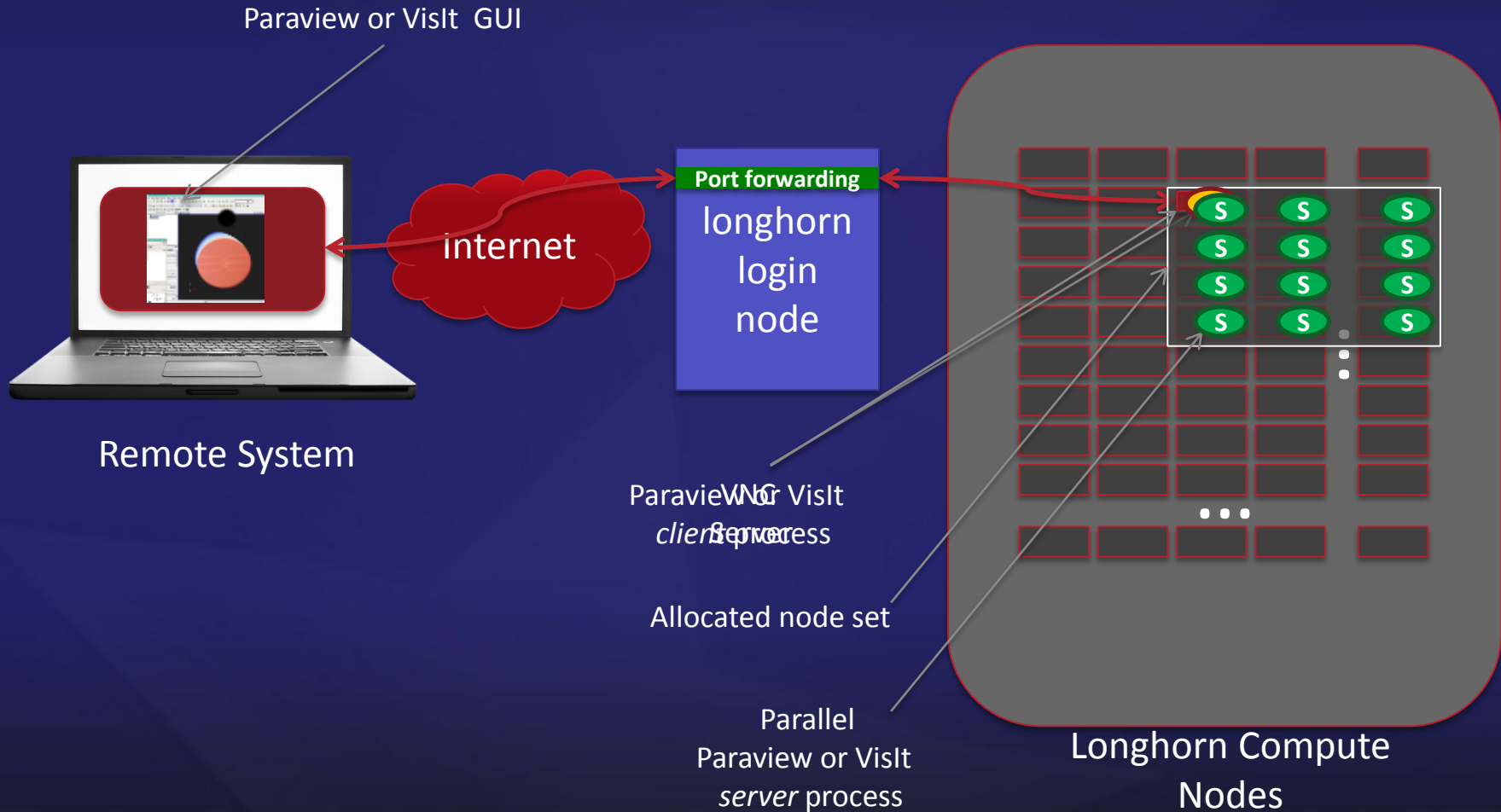
# Intro to Remote Viz

- (Full details tomorrow)
- VNC: Virtual Network Computing



# Demo: The Longhorn Portal

# Remote *Parallel* Visualization



# Parallel Visualization Software

- Good news! Paraview and Visit both run in parallel on and look *just the same!*
- Client/Server Architecture
  - Allocate multiple nodes for vncserver job
  - Run *client* process serially on root node
  - Run *server* processes on all nodes under MPI

# Parallel Visualization Software

- **Bad news! *They aren't multi-threaded!***
  - Even on a single multi-core node, you need to run using separate server processes to use all the cores.
  - Available memory is allocated evenly to all parallel processes
  - *if you are using  $k$  processes on a node, the max memory available to any one process will be  $1/k$  of the memory available on the node*

# Parallel Session Settings

- Number of nodes  $M$ 
  - more nodes gives you:
    - More total memory
    - More I/O bandwidth (up to a limit determined by file system and other system load)
    - More CPU cores, GPUs (though also affected by wayness)
- Wayness  $N$ 
  - processes to run on each node
  - $N < k$  gives each process more memory, uses fewer CPU cores for  $k$  = number of cores per node
  - Need  $N > 1$  (Longhorn; 3 on Spur) to utilize all GPUs
- Longhorn portal:
  - Number of Nodes and Wayness pulldowns
  - Limited maximum number of nodes
- `qsub -pe  $N$ way  $M$`  argument
  - $M$  specifies number of nodes (actually, specify  $k * M$ - see user guide for full documentation)
  - $N$  specifies wayness

# Running Paraview In Parallel

1. Set the environment variable **NO\_HOSTSORT** to **1**
  - (csh) `setenv NO_HOSTSORT 1`
  - (bash) `export NO_HOSTSORT=1`
2. Run Paraview as before (with any desired client-side options)
3. In Paraview GUI:
  - **File->Connect** to bring up the **Choose Server** dialog
  - If this is the first time you've used PV in parallel (or failed to save your connection configuration from your prior run):
    - Click **Add Server**
    - Set the server configuration name to **ibrun**
    - Click **Configure**, and set **Startup Type** to **Command** and enter the command:  
`ibrun tacc_xrun pvserver [server options]`
    - Click **Save**
  - In **Choose Server** dialog, select **ibrun** and click **Connect**

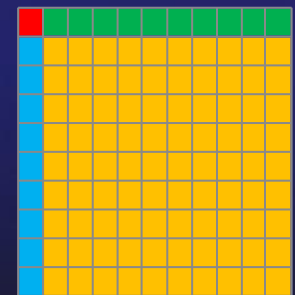
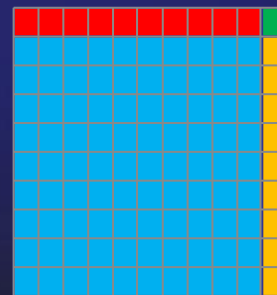
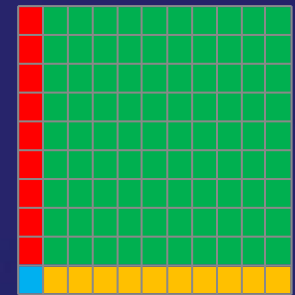
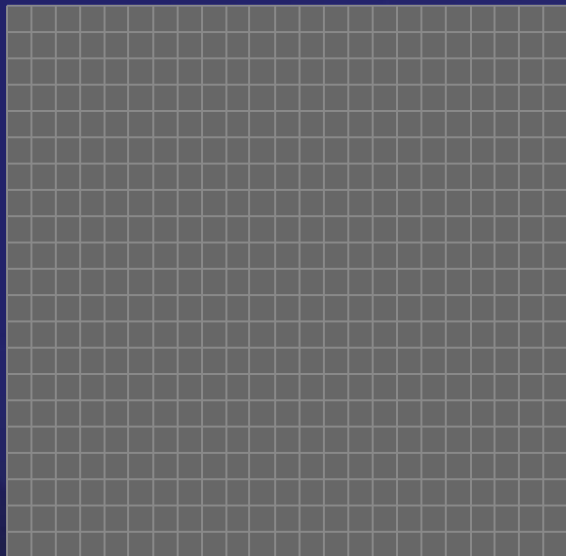
The Output Messages window will show the server job being run and the connection being made.

# Running Visit In Parallel

- Run Visit as before; it'll do the right thing

# Data-Parallel Visualization Algorithms

- Spatially partitioned data are distributed to participating processes...

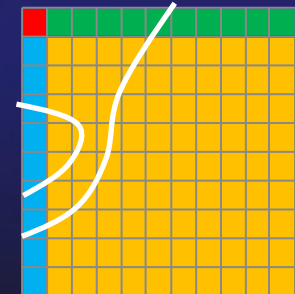
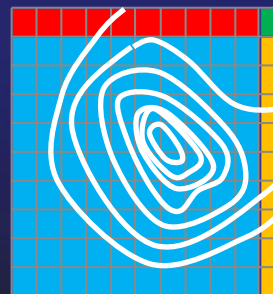
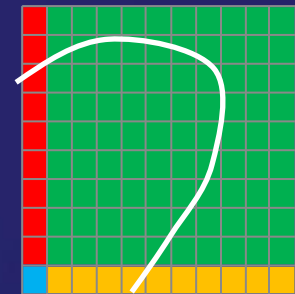
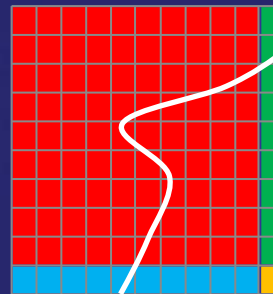


# Data-Parallel Algorithms

- Sometimes work well...
- Iso-contours

- Surfaces can be computed in each partition concurrently and (with ghost zones) independently

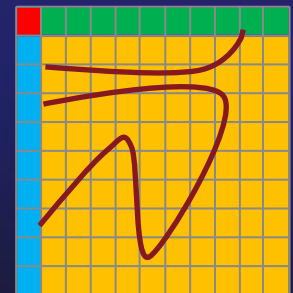
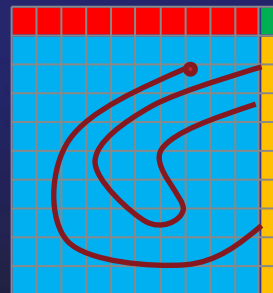
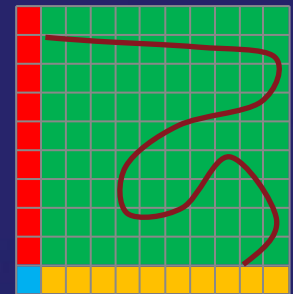
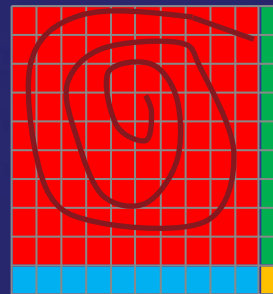
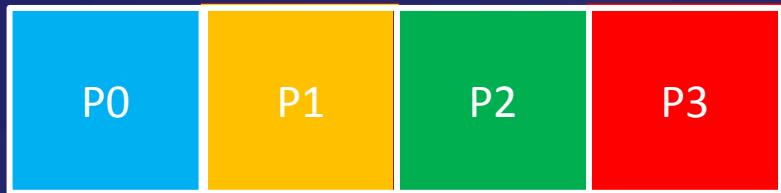
- However, since surfaces may not be evenly distributed across partitions, may lead to poor load balancing



# Data-Parallel Algorithms

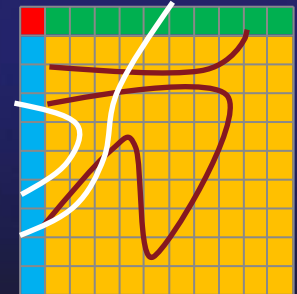
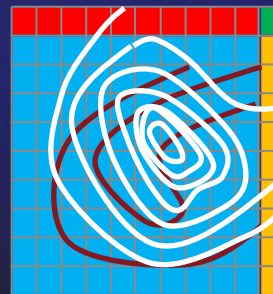
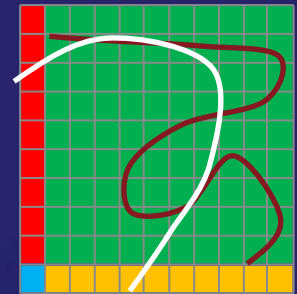
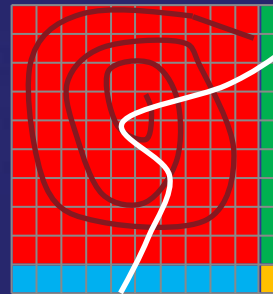
- Sometimes not so much...

- Streamlines are computed incrementally



# Parallel Rendering

- Collect-and-render
  - Gather all geometry on 1 node and render
- Render-and-composite
  - Render locally, do depth-aware composite
- Both PV and Visit have both, offer user control of which to use

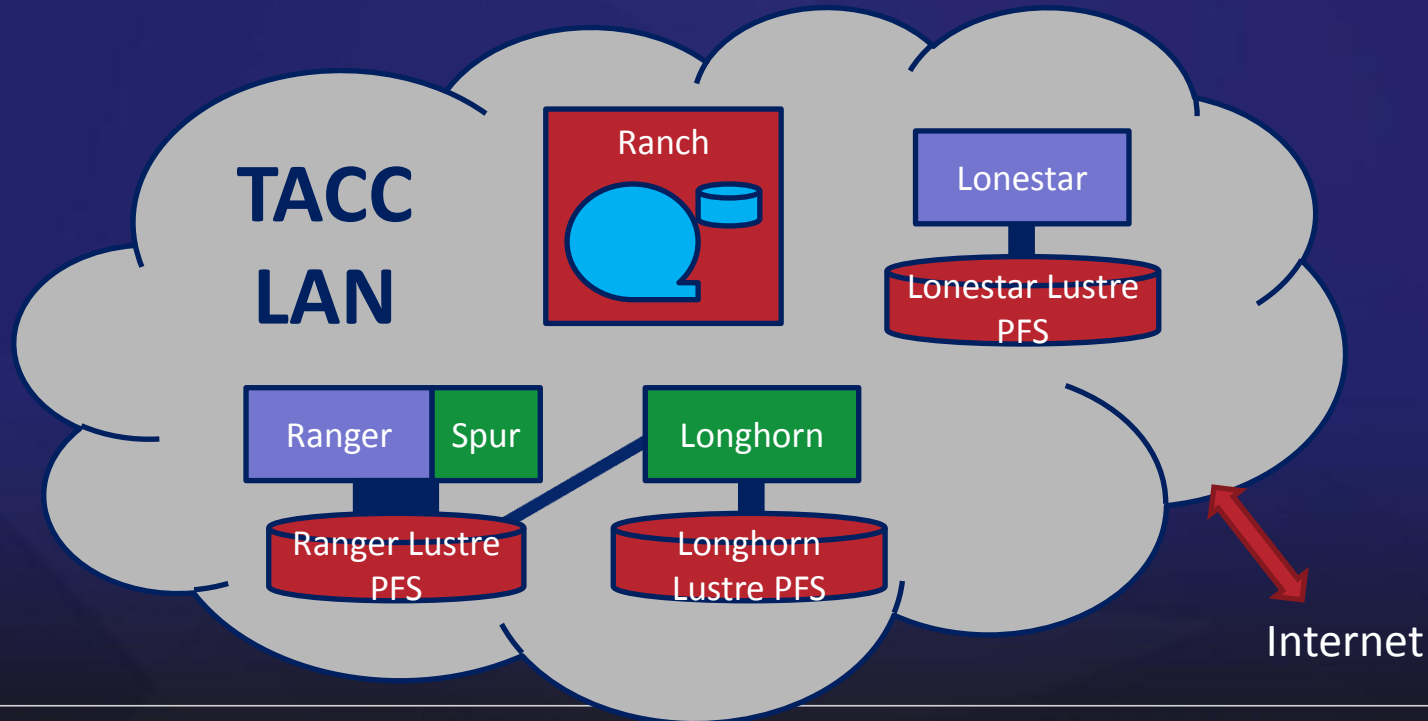


# Parallel Data Formats

- To run in parallel, data must be distributed
- Parallel formats enable compute nodes to import partition data in parallel
  - SILO format (Paraview, VisIt)
  - Paraview parallel formats (Paraview)
  - Brick-of-floats formats (Paraview, VisIt)
- Paraview will import non-partitioned data, but:
  - It imports it into one node
    - memory footprint
    - serial I/O
  - Has to partition it and distribute it
  - Its s l o w ...

# Data Location

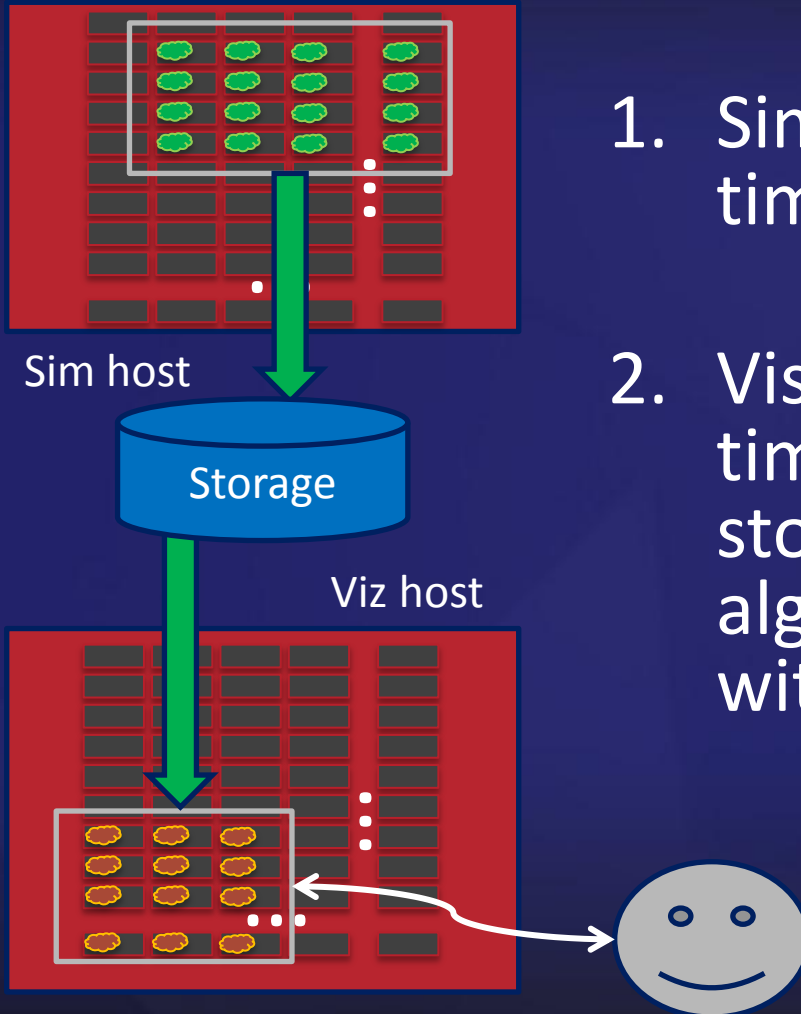
- Data must reside on accessible file system
- Movement *within* TACC faster than across Internet, but can still take a long time to transfer between systems



# Huge Data: *Co- and In-Situ* Processing

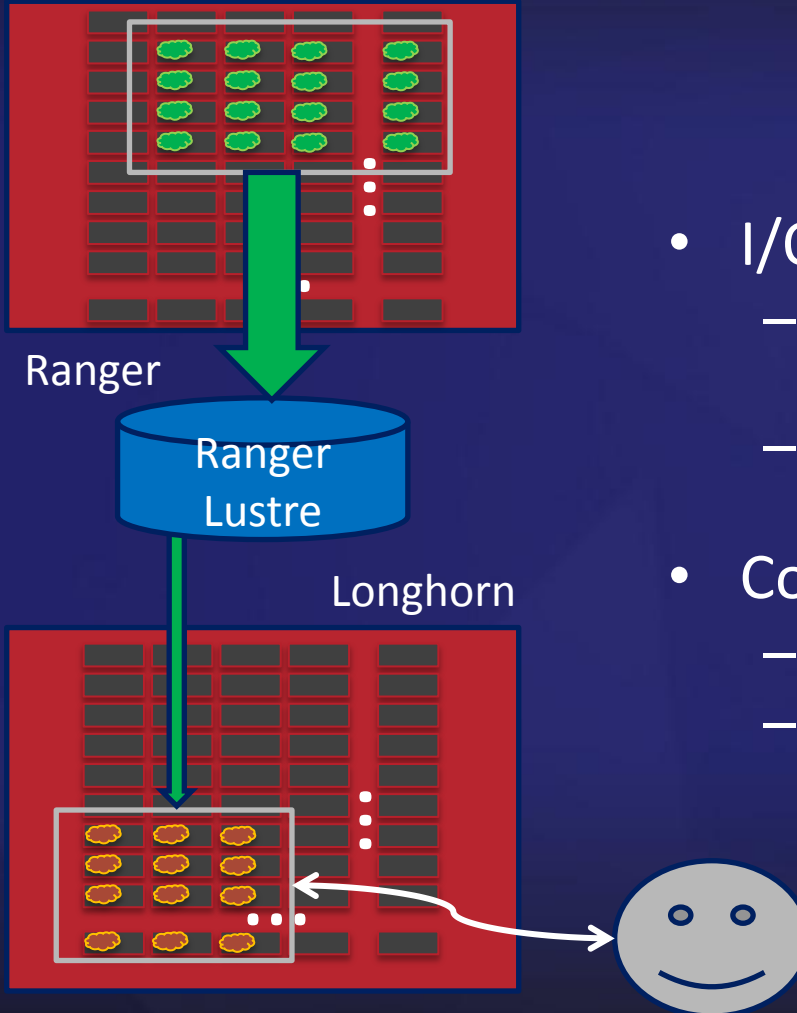
- Visualization requires equivalent horsepower
  - Not all visualization computation is accelerated
  - Increasingly, HPC platforms include acceleration
- I/O is *expensive*: simulation to disk, disk to disk, disk to visualization
  - I/O is not scaling with compute
  - Data is not always written at full spatial, temporal resolution

# Post-Processing



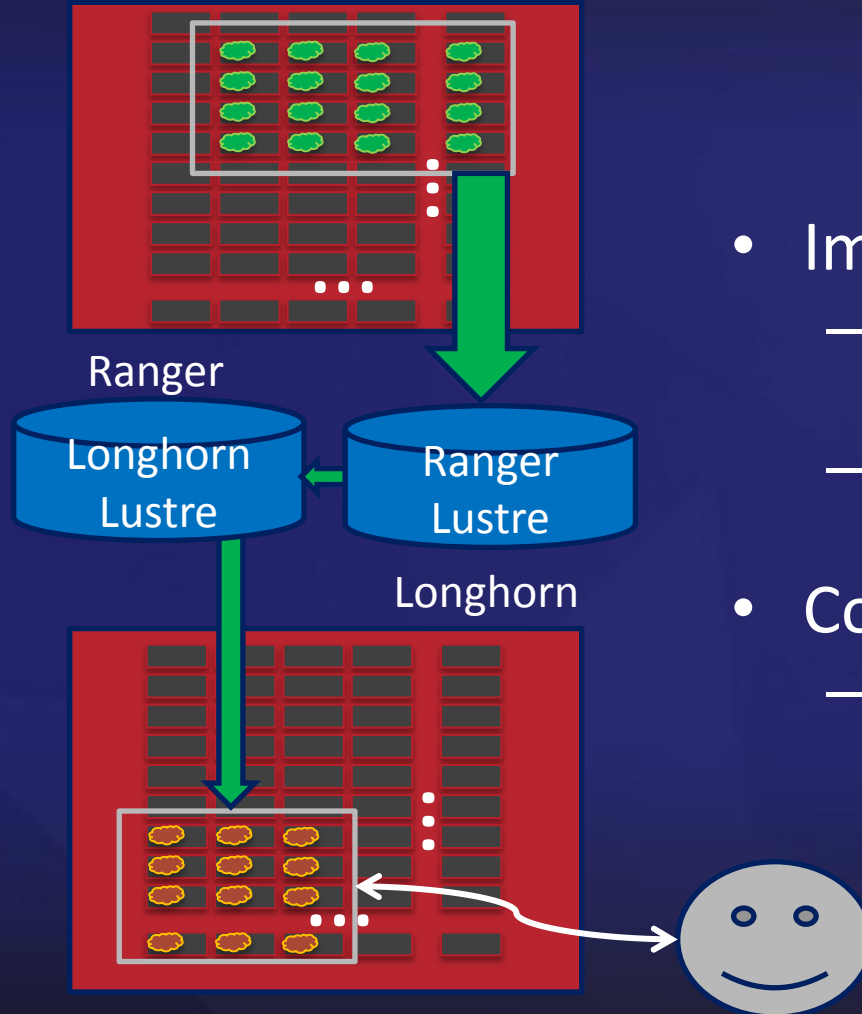
1. Simulation writes periodic timesteps to storage
2. Visualization loads timestep data from storage, runs visualization algorithms and interacts with user

# ... On Ranger and Longhorn



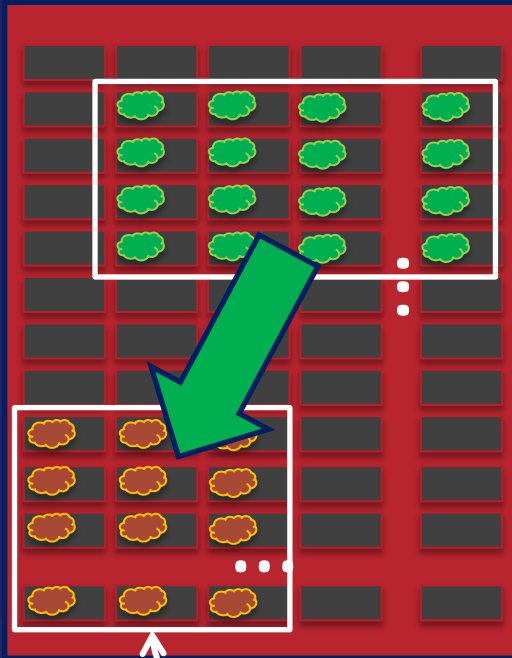
- I/O Imbalance
  - Ranger: max 35 GB/sec to Ranger Luster
  - Longhorn: 0.5 GB/sec from Ranger Luster
- Compute Imbalance
  - Ranger: >62K cores
  - Longhorn: 2K cores

# ... On Ranger and Longhorn



- Improve I/O Imbalance
  - Move dataset to Longhorn *offline*
  - Get ~5 GB/sec from Longhorn Lustre
- Compute Imbalance
  - ... well, Longhorn has a lot of *GPU* cores

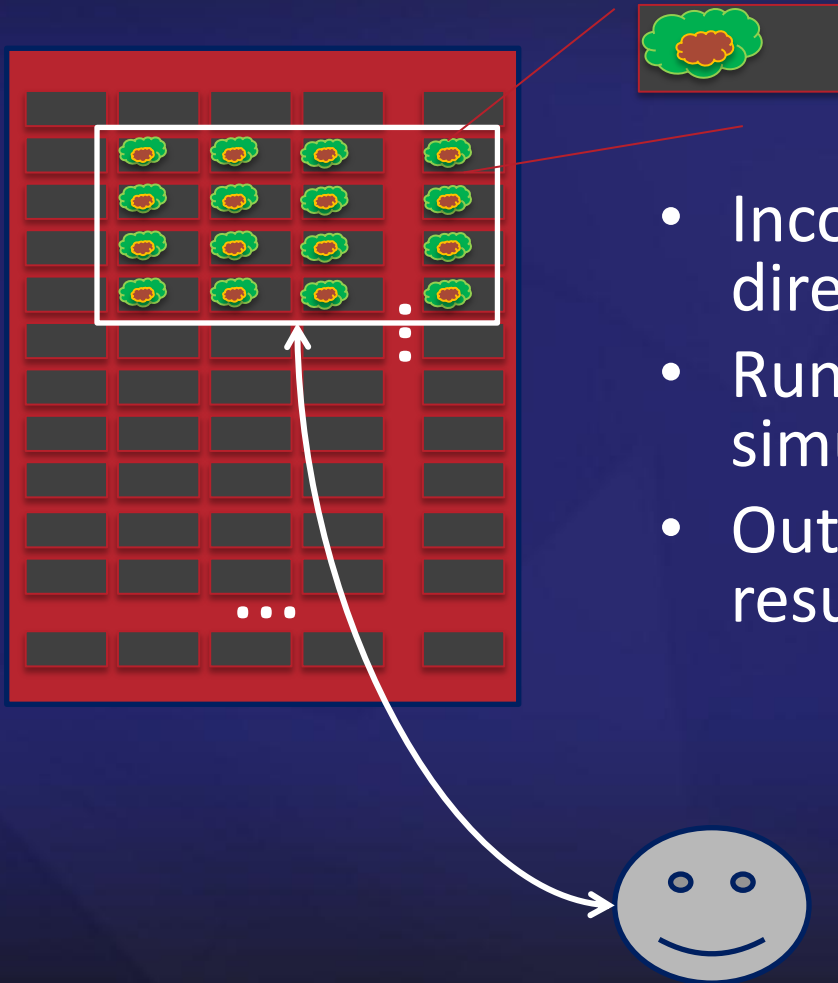
# Co-Processing



- Perform simulation and visualization on same host
  - Concurrently
  - Using high-performance interconnect to communicate



# In-Situ Processing



- Incorporate visualization directly *into* simulation
- Run visualization algorithms on simulation's data
- Output only visualization results

# *Co- and In-Situ Processing*

- *Not a panacea*
  - Limits scientist's exploration of the data
    - Can't go back in time
    - May pay off to *re-run* the simulation
  - Impacts simulation
    - May require source-code modification of simulation
    - May increase simulation node's footprint
    - May affect simulation's stability
  - Simulation host may not have graphics accelerators
    - ... but visualizations are often not rendering-limited
    - ... and more and more HPC hosts are *including* accelerators

# *Co- and In-Situ Status*

- *Bleeding* edge
- Coprocessing capabilities in Paraview, VisIt
  - Did I say bleeding edge?
- In-Situ visualization is not simple
- We can help

# Summary

- Parallel visualization is only *partly* about upping the compute power available, its also about getting sufficient memory and I/O bandwidth.
- I/O is a *really big* issue. Planning how to write your data for parallel access, and placing it where it can be accessed quickly, is critical.
- The TACC visualization groups are here to help you!