Porting of VisIt To Cray XT3

Kevin Thomas, Cray Inc.

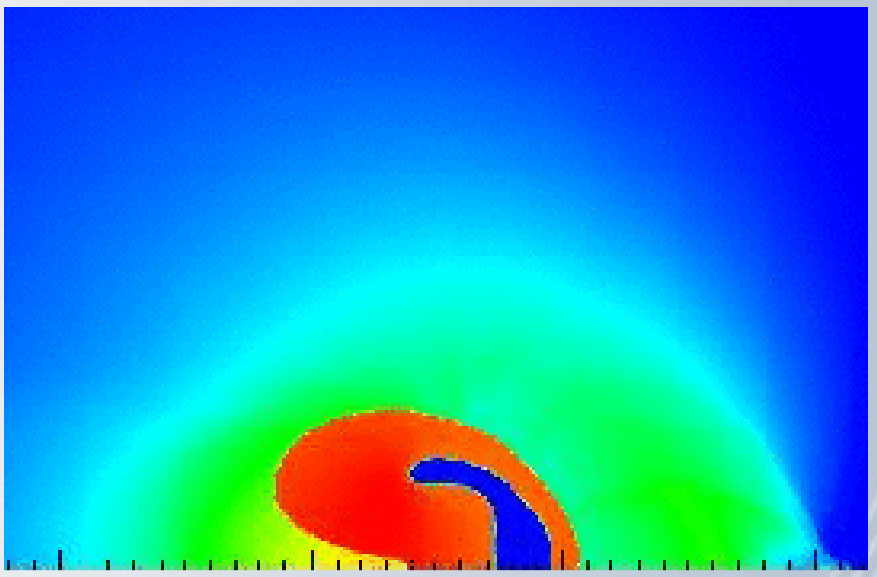
CRAY
THE SUPERCOMPUTER COMPANY

Overview

- VisIt—What is it? Why on XT3?
- VisIt Architecture
- Porting Issues
- Plugins
- Sockets
- Performance
- Conclusion

VisIt—What is it?

- Post-processor of physical simulation results



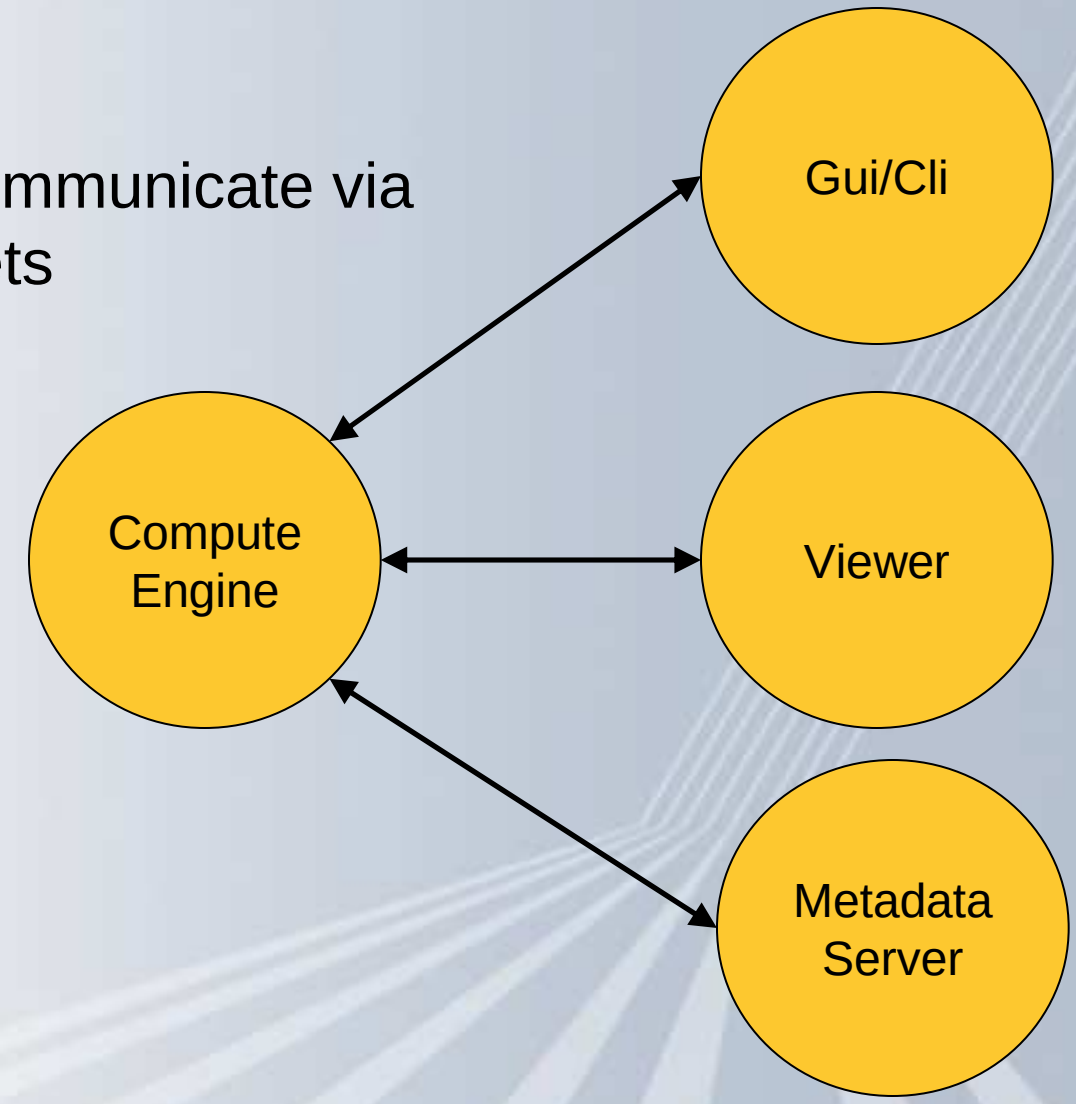
- Interactive visualization of calculated quantities
  - pressure, temperature, field intensity, etc.
- Batch processing through scripts
- Portable, but requires OS features not in Catamount

Why VisIt on XT3?

- Keeps users on one system
  - No need to purchase a dedicated visualization system
  - No need to learn how to use separate system
- Keeps data on one system
  - No need to stage data from XT3 to visualization system
  - Some datasets are too large to be moved
- Leverages XT3's strengths
  - Scalable compute
  - Scalable memory capacity
  - High speed parallel I/O

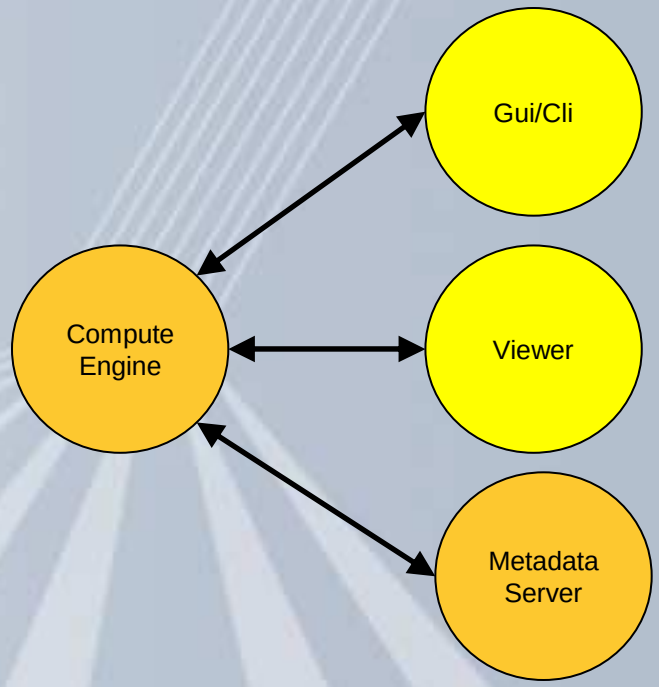
VisIt Architecture

Components communicate via TCP/IP sockets



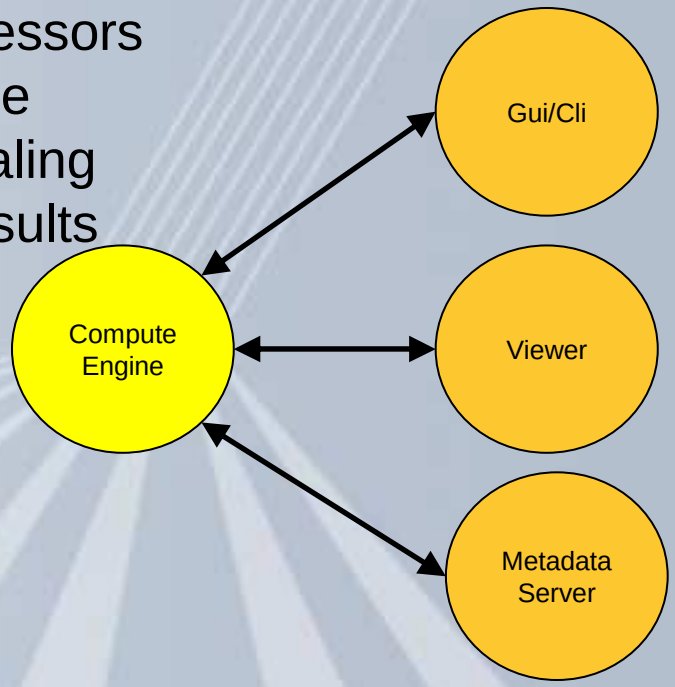
Porting Considerations

- gui, cli, and viewer are workstation-oriented components
  - Interactive
    - gui and cli are input-oriented
    - viewer is output-oriented
  - Often run on a desktop computer
  - Sometimes may need to run on XT3
  - Linux version based on X Windows
    - gui and viewer
  - Suitable for login node



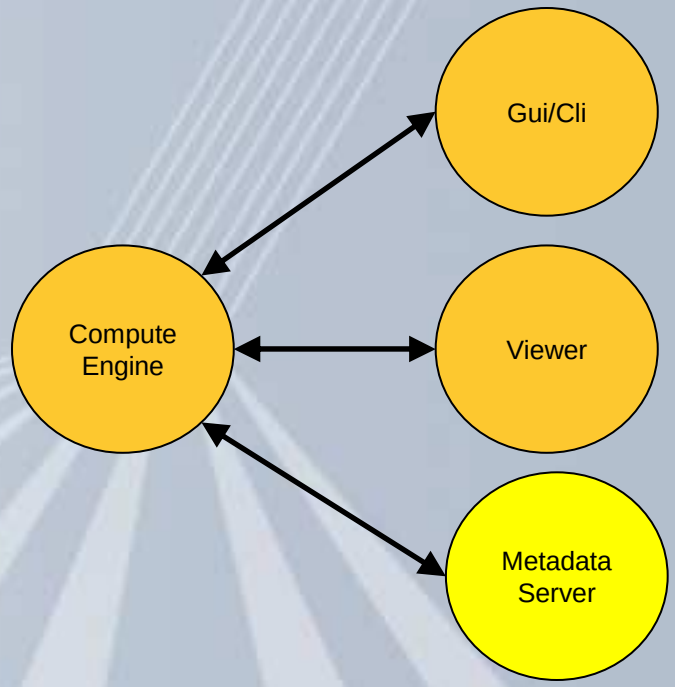
Porting Considerations

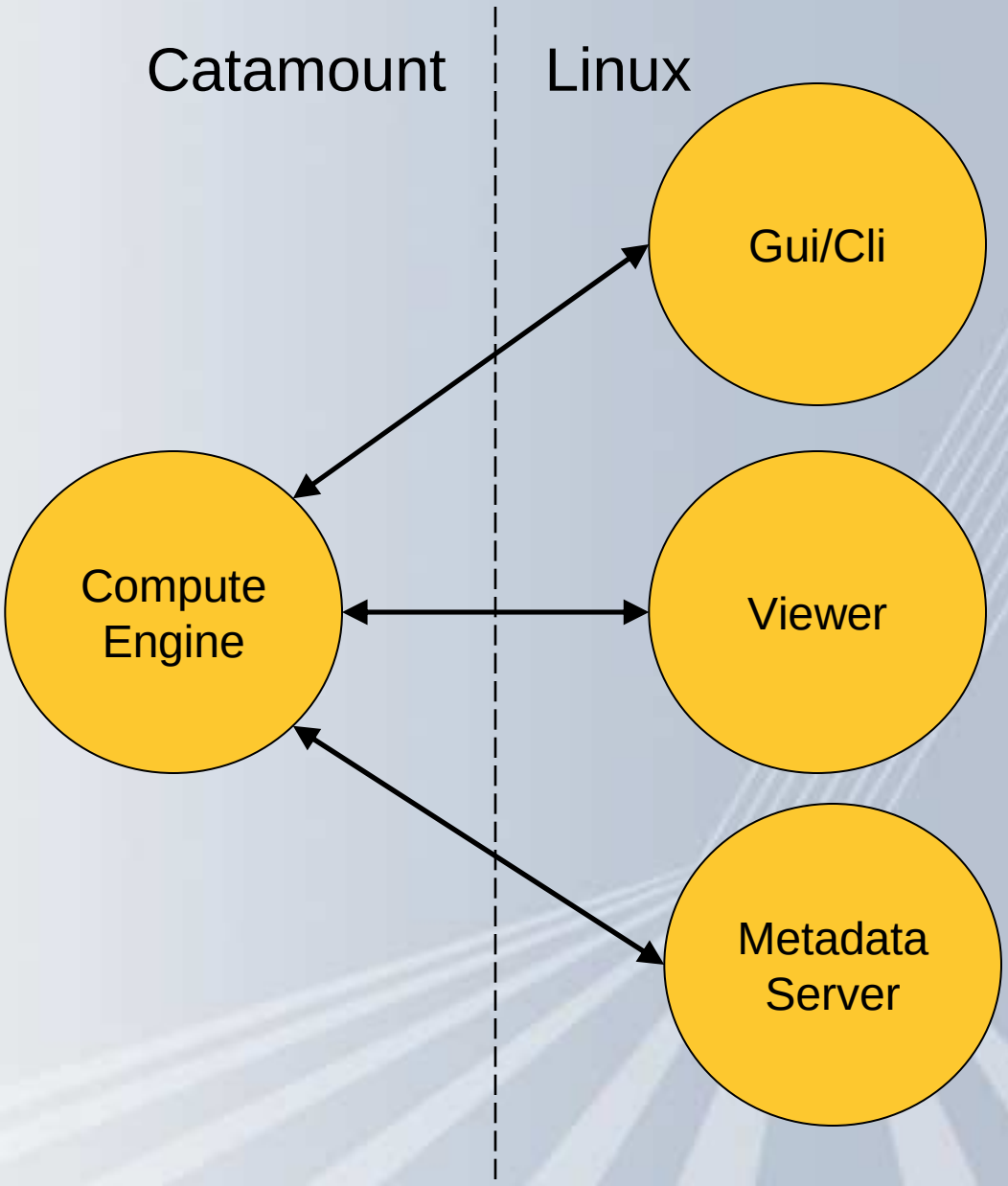
- The parallel compute engine is a good candidate for XT3
  - Uses MPI for internal communication
  - Needs access to simulation results
  - Image rendering is scalable with data set size
    - Increasing data size, i.e., *weak* scaling
    - Need more memory
    - Can take advantage of more processors
  - Image rendering is scalable in wall time
    - Constant data size, i.e., *strong* scaling
    - Use more processors for faster results
    - Maintain an interactive feel
  - Requires compute nodes
    - Catamount



Porting Considerations

- The metadata server could run anywhere
  - Not heavy compute, not parallel
  - Not interactive or graphical
  - Needs access to the simulation results
  - Usually runs where the compute engine runs
  - Fits easily as a login node process




Catamount
Linux
Gui/Cli
Compute Engine
Viewer
Metadata Server

Plugins

- Plugins are extensions to VisIt
  - Database plugins for reading various data formats
  - Operator plugins for manipulating the data
  - Plot plugins for displaying the data
- Common for users to write custom database plugins
  - AWE uses custom database plugins
- Loaded dynamically (at run time) when needed
  - A plugin is a shared library
  - dlopen, dlsym, dlclose
  - Not supported on Catamount

Supporting Plugins on Catamount

- First effort was to add full support for run time linking
  - dlopen, dlsym, dlclose
  - Easy to support leaf routines
  - Difficult to support external calls, system calls
  - All database plugins use I/O calls
- Second effort was to support the linking interface only
  - dlopen, dlsym, dlclose
  - But implemented via static linking of plugins
  - All plugins linked into the parallel compute engine
  - New plugins require relinking the compute engine

Plugin Implementation

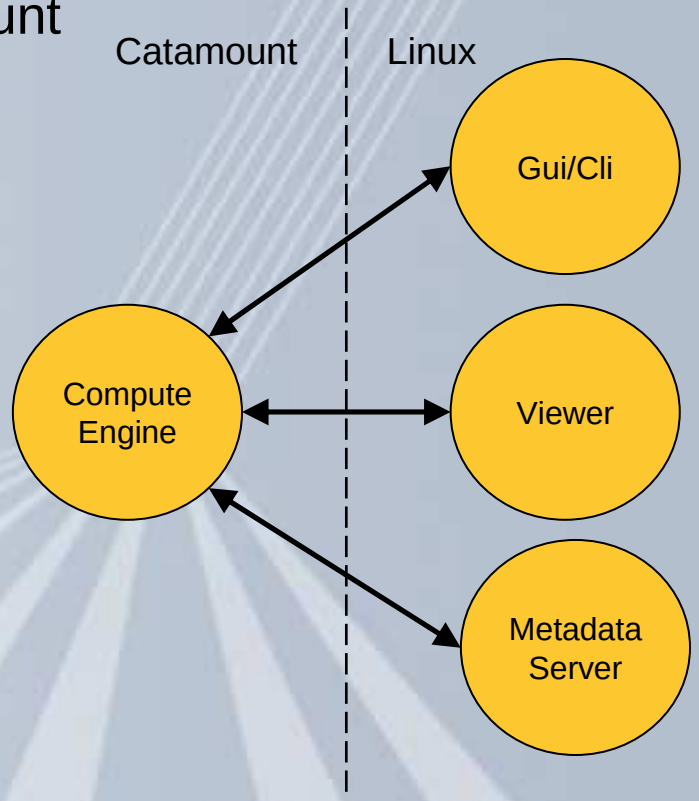
- Plugins are built as static libraries
  - Automated via configure script settings
- Plugin external symbols are renamed
  - Prefixed with the plugin name
  - GetGeneralInfo in libIPLOT3DDatabase.so
  - PLOT3DGetGeneralInfo in libIPLOT3DDatabase.a
  - Automated via a script run on plugin libraries
- Wrapper functions to provide standard interface
  - dlopen verifies the plugin is available
  - dlsym returns symbol values from a lookup table
  - dlclose frees memory allocated in dlopen
  - A script generates the table and wrappers

Other Plugin Issues

- Two plugins can contain the same routine names
  - Symbol name collision
  - Rare, partly due to coding practices
  - One occurrence in the VisIt standard plugins
- Plugin discovery in VisIt
  - Scans the plugin directories in $VISITDIR
  - Uses the Linux plugin files
  - $VISITDIR must be accessible from Catamount
- Statically linking all plugins increases code size
  - Less than 4 MB for all standard plugins

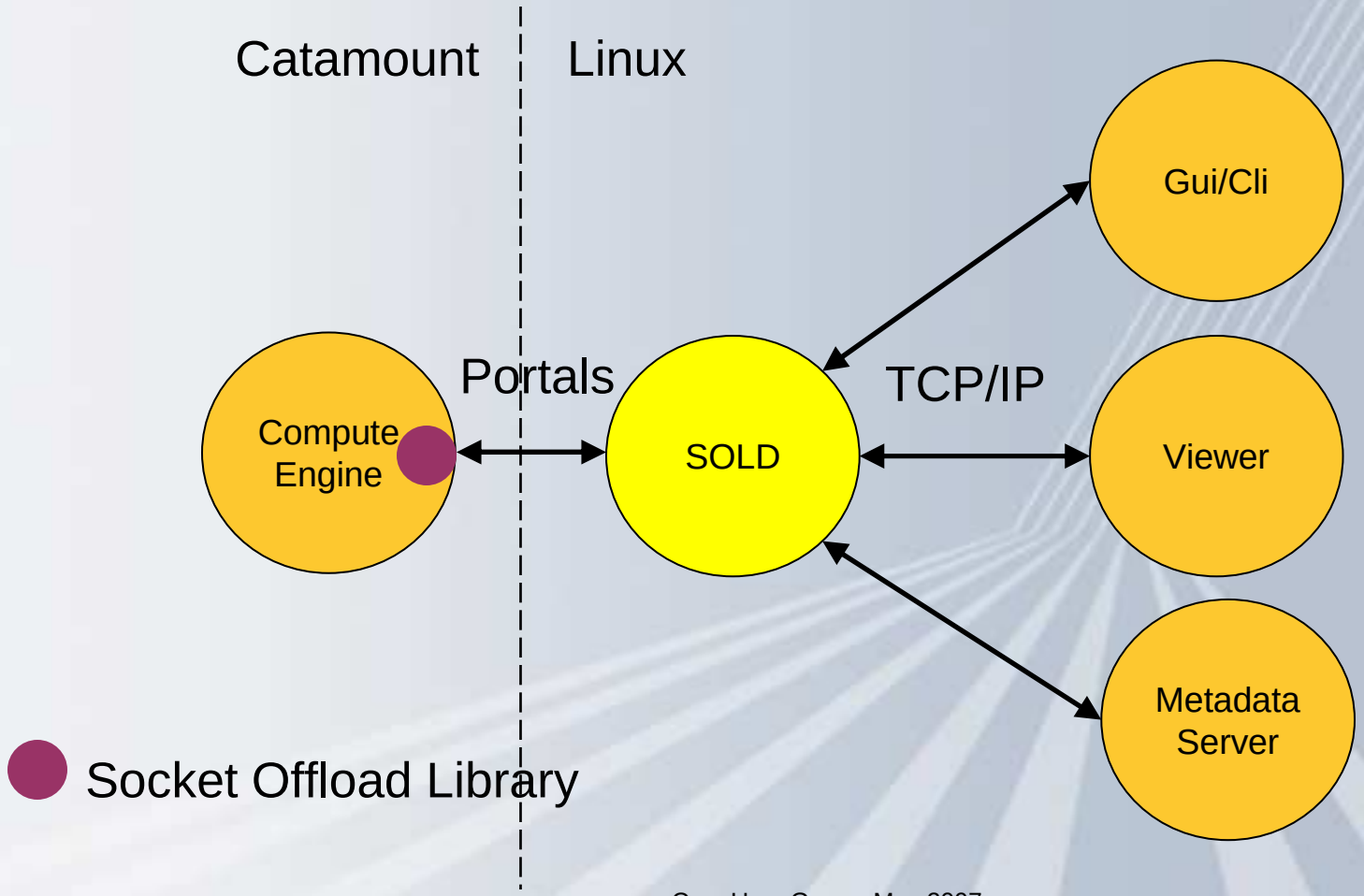
Sockets

- TCP/IP sockets are used to communicate between VisIt components
  - Allows the components to be distributed over a network
  - Sockets are fully supported on XT3 login nodes
  - Sockets are not supported on Catamount



Socket Bridge

- An intermediate process acts as a bridge
  - Compute engine is linked with socket offload library

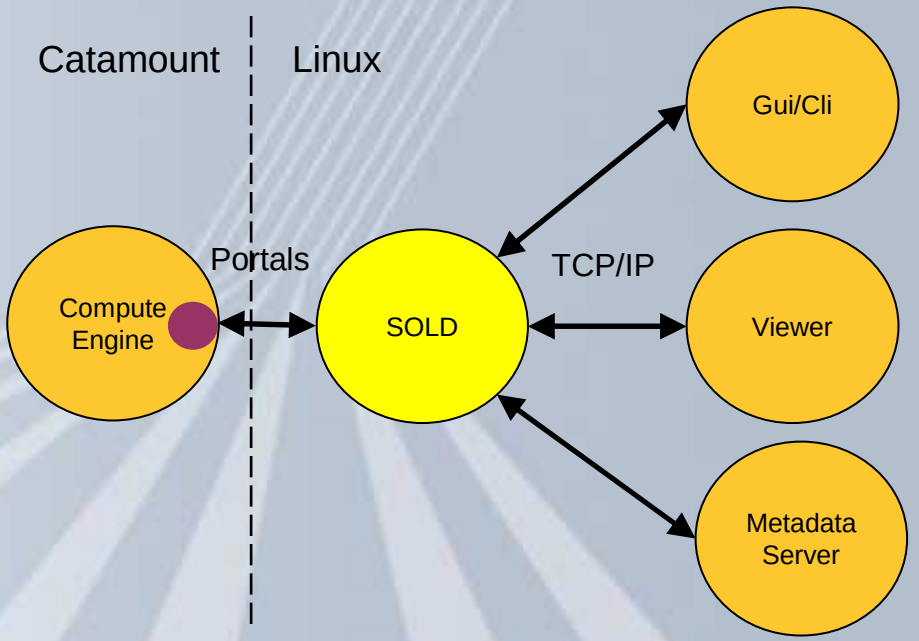


Socket Offload Library

- Provides interface for network and socket calls
  - socket, connect, sendmsg, recvmsg
  - gethostbyaddr, inet_pton
- Library communicates with SOLD process via Portals
- Functions used by VisIt are implemented
- Features used by VisIt are implemented
- Full TCP/IP socket support is not implemented
- Full networking call support is not implemented

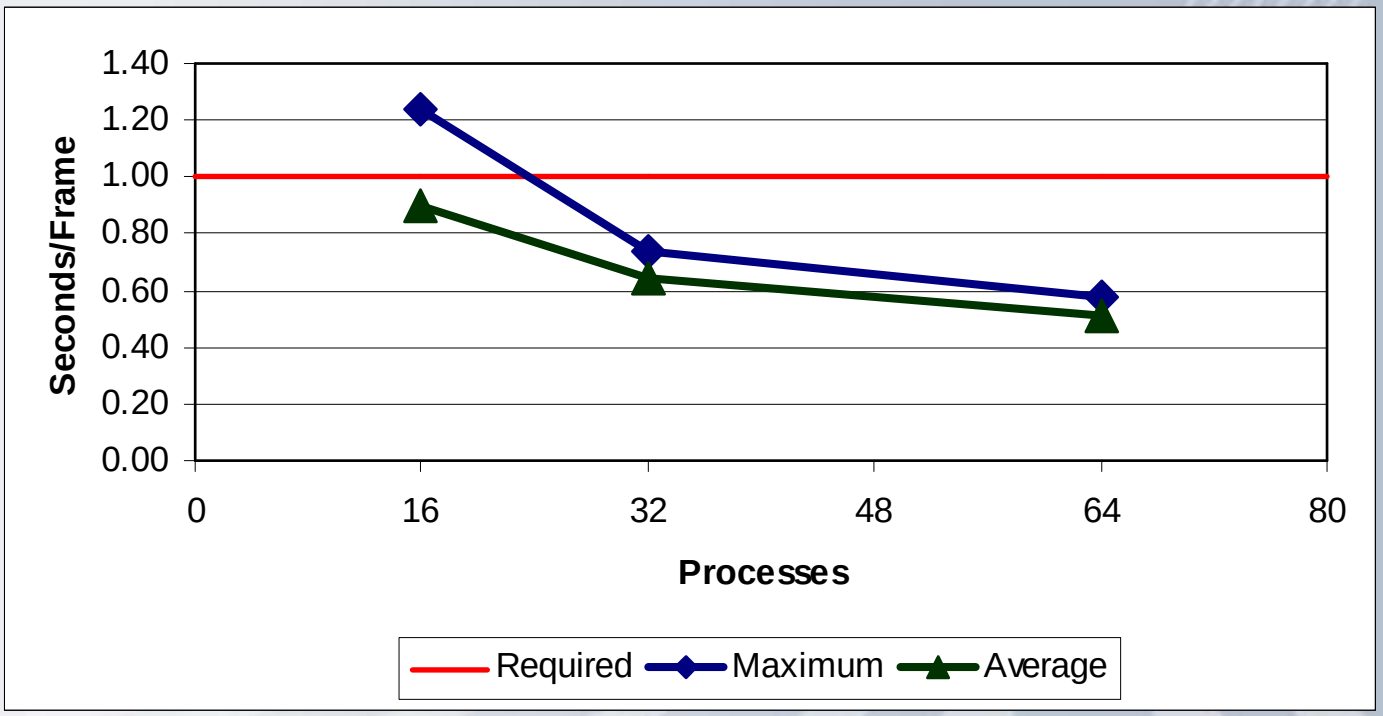
SOLD Bridge Process

- Acts as a proxy for socket and network calls made by the Catamount process
- Runs on Linux
- Accepts requests from Catamount over Portals
- Executes the request on the login node
- Returns results to Catamount



Performance

- 1 second per frame maximum required
- Average rate below 1 second/frame at 16 cores
- 32 cores required to bring down the maximum



Compute Node Linux

- CNL eliminates many of the porting difficulties
  - CNL is more compatible with operating system standards
  - TCP/IP sockets are supported
  - Some shared library support available
  - VisIt might build with a few configure settings
  - Additional work might be needed for performance tuning

Conclusions

- VisIt runs on XT3
  - Parallel execution fully supported
  - Distributed execution supported
  - Plugin support is less flexible
  - Performance goal achieved
- Significant changes in building VisIt
  - Parallel compute engine requires Catamount build
  - But for the most part automated
- No source code changes
  - SOLD for minimal support for VisIt socket calls
  - Special build procedure for plugins