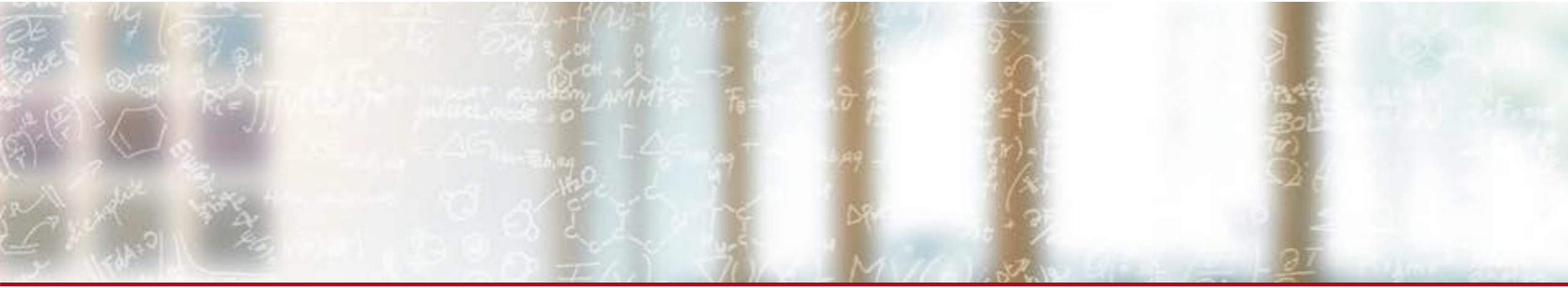
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



VisIt Libsim. An *in-situ* visualisation library

December 2017

Jean M. Favre, CSCS

Outline

- Motivations
- *In-situ* visualization
  - In-situ processing strategies
  - VisIt's libsim library
  - Enable visualization in a running simulation
  - Source code instrumentation

Facts

- Parallel simulations are now ubiquitous
- The mesh size and number of timesteps are of unprecedented size
- The traditional ***post***-processing model "*compute-store-analyze*" does not scale

Consequences:

- **Datasets are often under-sampled**
- **Many time steps are never archived**
- **It takes a supercomputer to re-load and visualize supercomputer data**

Motivations Statements

Having a real-time monitoring capability on all supercomputing resources is essential **to avoid wasting valuable time on computational resources**…

Scientists have needs for both run-time monitoring and for **coupling of those codes...with other codes**

There are great opportunities to do better science (analysis) when access to the full spatio-temporal resolution data is possible.

History has shown how compute and I/O capacities are unbalanced

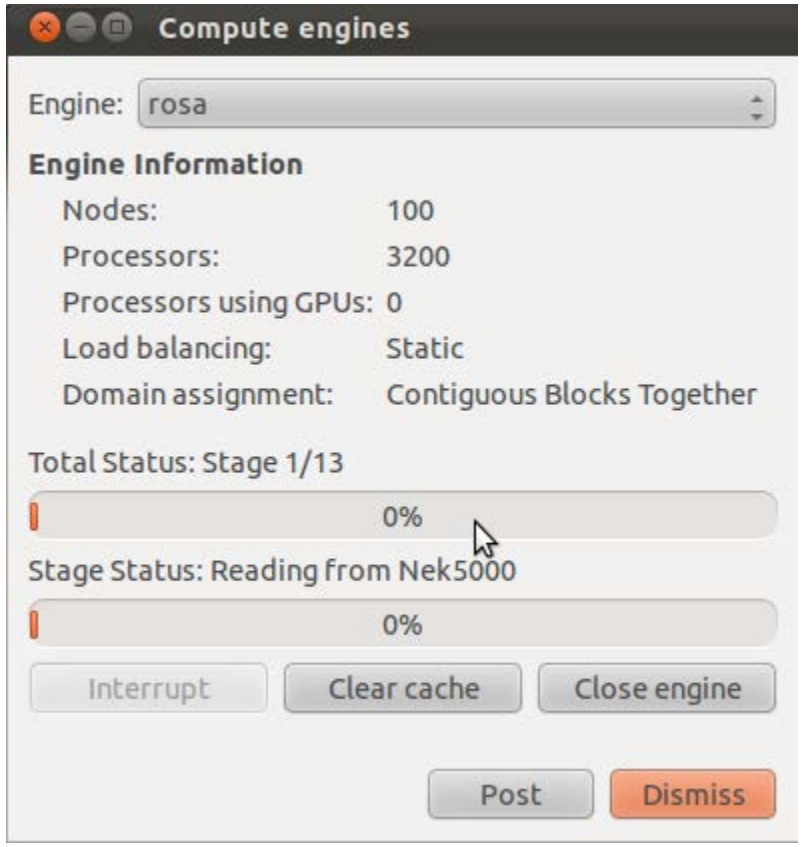
And the future is no different !!!

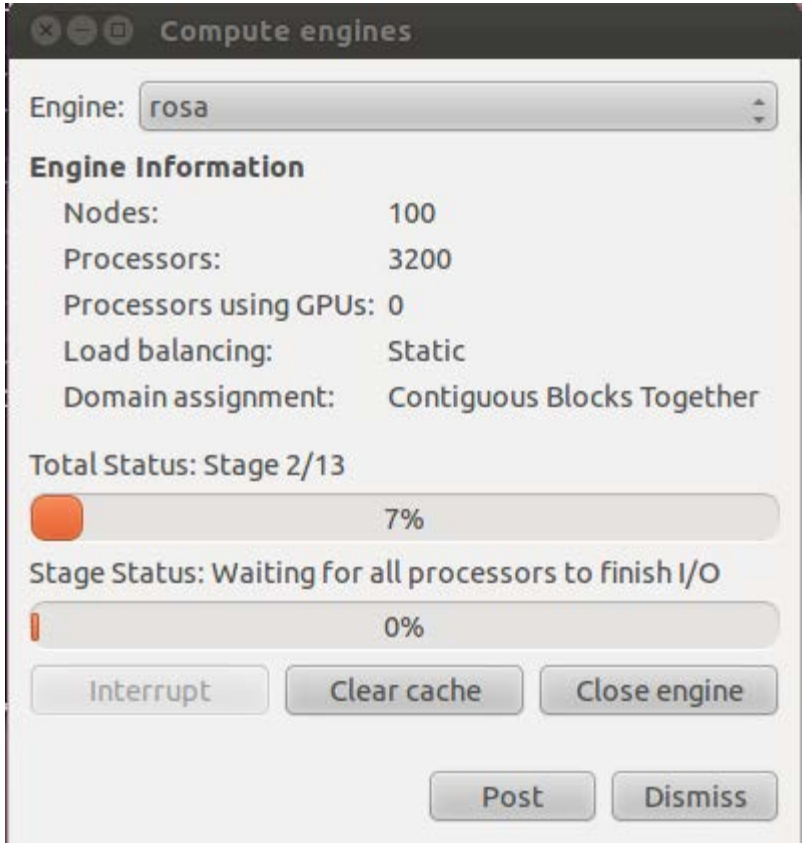
How does Summit compare to Titan

Feature	Summit	Titan
Application Performance	5-10x Titan	Baseline
Number of Nodes	~3,400	18,688
Node performance	> 40 TF	1.4 TF
Memory per Node	>512 GB (HBM + DDR4)	38GB (GDDR5+DDR3)
NVRAM per Node	800 GB	0
Node Interconnect	NVLink (5-12x PCIe 3)	PCIe 2
System Interconnect (node injection bandwidth)	Dual Rail EDR-IB (23 GB/s)	Gemini (6.4 GB/s)
Interconnect Topology	Non-blocking Fat Tree	3D Torus
Processors	IBM POWER9™ NVIDIA Volta™	AMD Opteron™ NVIDIA Kepler™
File System	120 PB, 1 TB/s, GPFS™	32 PB, 1 TB/s, Lustre®
Peak power consumption	10 MW	9 MW

Data courtesy A. Geist (ORNL)

Typical situation…





When there is too much data…

Several strategies are available to mitigate the data problem:

- read less data:
  - multi-resolution,
  - on-demand streaming,
- out-of-core, etc...

- Do no read data from disk but from memory:

in-situ **visualization**

in-situ (parallel) visualization

Instrument parallel simulations to:

- Eliminate I/O to and from disks
- Use all grid data with or without ghost-cells
- Have access to all time steps, all variables
- Use the available parallel compute nodes
- Maximize features and capabilities
- Minimize code modifications to simulations
- Minimize impact to simulation codes
- Allow users to start an in-situ session *on demand* instead of deciding before running a simulation
  - Debugging
  - Computational steering

- scalable vis infrastructure accessible in situ
  - VisIt/Libsim
  - Paraview/Catalyst
- ADIOS: I/O library approach
- SENSEI: generic in situ interface

Existing in-situ approaches

ADIOS and GLEAN both provide tools for in situ I/O and some analysis

- They allow simulations to adopt in situ techniques by leveraging their advanced I/O infrastructures that enable co-analysis pipelines rather than changing the simulator.
- The non-intrusive integration provides resilience to third party library bugs and possible jitter in the simulation.

ParaView and VisIt both provide tools for in situ analysis and visualization

- Catalyst can be tightly or loosely linked to a simulation, allowing the simulation to share data with Catalyst for analysis and visualization.
- Similar capabilities are available within VisIt with the Libsim library.
- Catalyst-Live, Libsim, and ADIOS enable the opposite flow of information, sending data from the client to the simulation, enabling the possibility of in situ and/or monitoring/simulation steering.

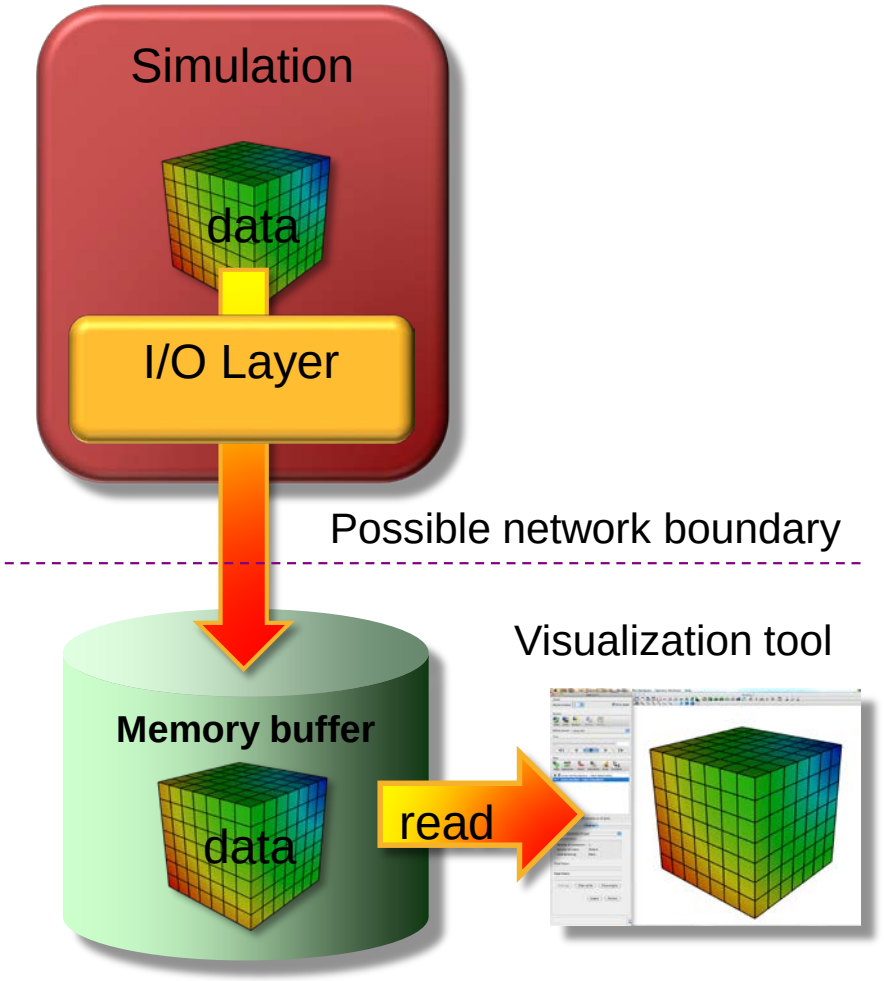
(text source SENSEI SC17 tutorial)

in-situ Processing Strategies

In Situ Strategy	Description	Negative Aspects
Loosely coupled a.k.a. "Concurrent processing"	Visualization and analysis run on concurrent resources and access data over network	1) Data movement costs 2) Requires separate resources
Tightly coupled a.k.a. "Co-processing"	Visualization and analysis have direct access to memory of simulation code	1) Very memory constrained 2) Large potential impact (performance, crashes)
Hybrid	Data is reduced in a tightly coupled setting and sent to a concurrent resource	1) Complex 2) Shares negative aspects (to a lesser extent) of others

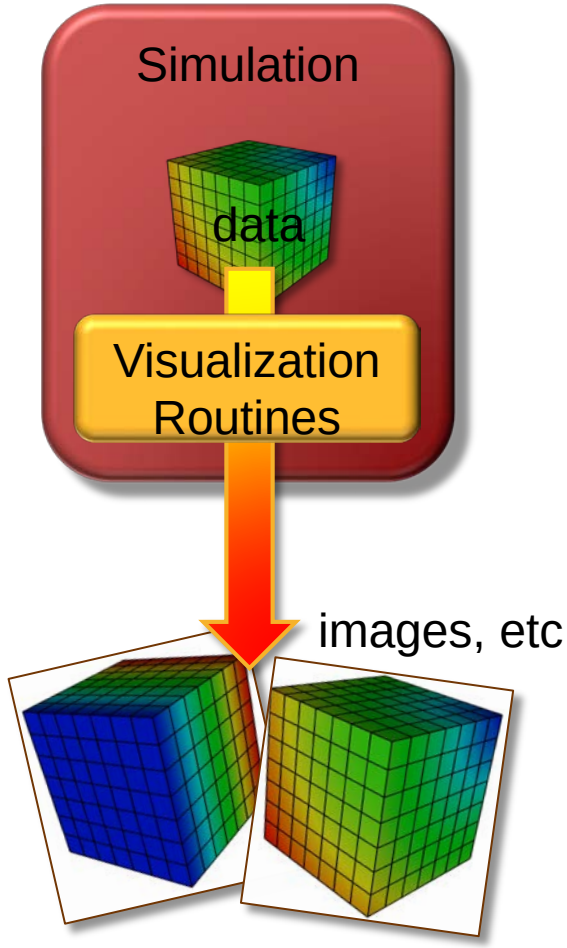
Loosely Coupled in-situ Processing

- I/O layer stages data into secondary memory buffers, possibly on other compute nodes
- Visualization applications access the buffers and obtain data
- Separates visualization processing from simulation processing
- Copies and moves data



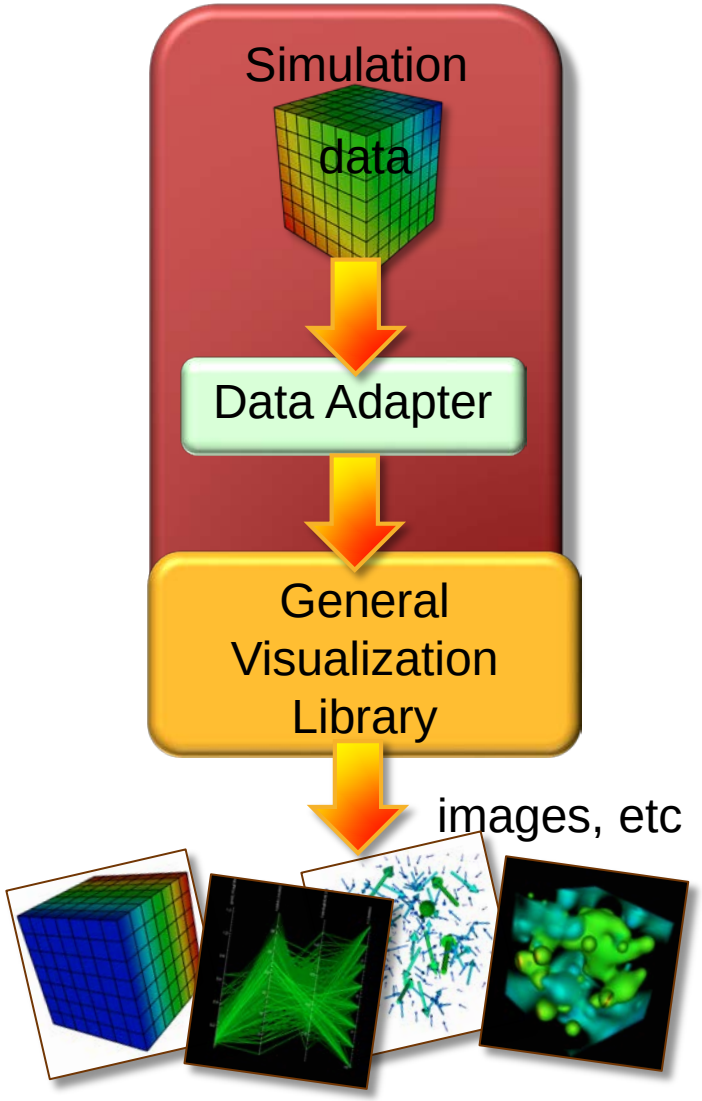
Tightly Coupled *Custom* in-situ Processing

- Custom visualization routines are developed specifically for the simulation and are called as subroutines
  - Create best visual representation
  - Optimized for data layout
- Tendency to concentrate on very specific visualization scenarios
- *Write once, use once*

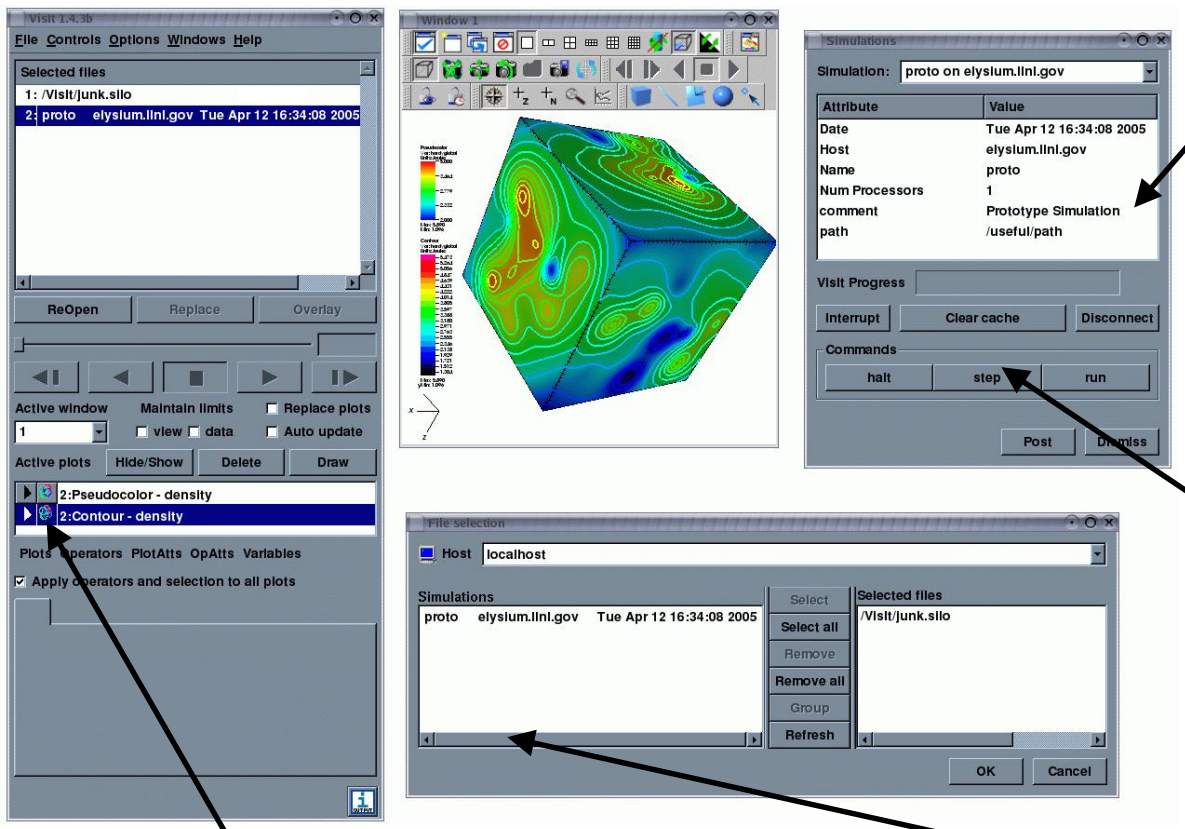


Tightly Coupled *General* in-situ Processing

- Simulation uses data adapter layer to make data suitable for general purpose visualization library
- Rich feature set can be called by the simulation
- Operate directly on the simulation's data arrays when possible
- *Write once, use many times*



Libsim in VisIt



The Simulation's window shows meta-data about the running code

Control commands exposed by the code are available here

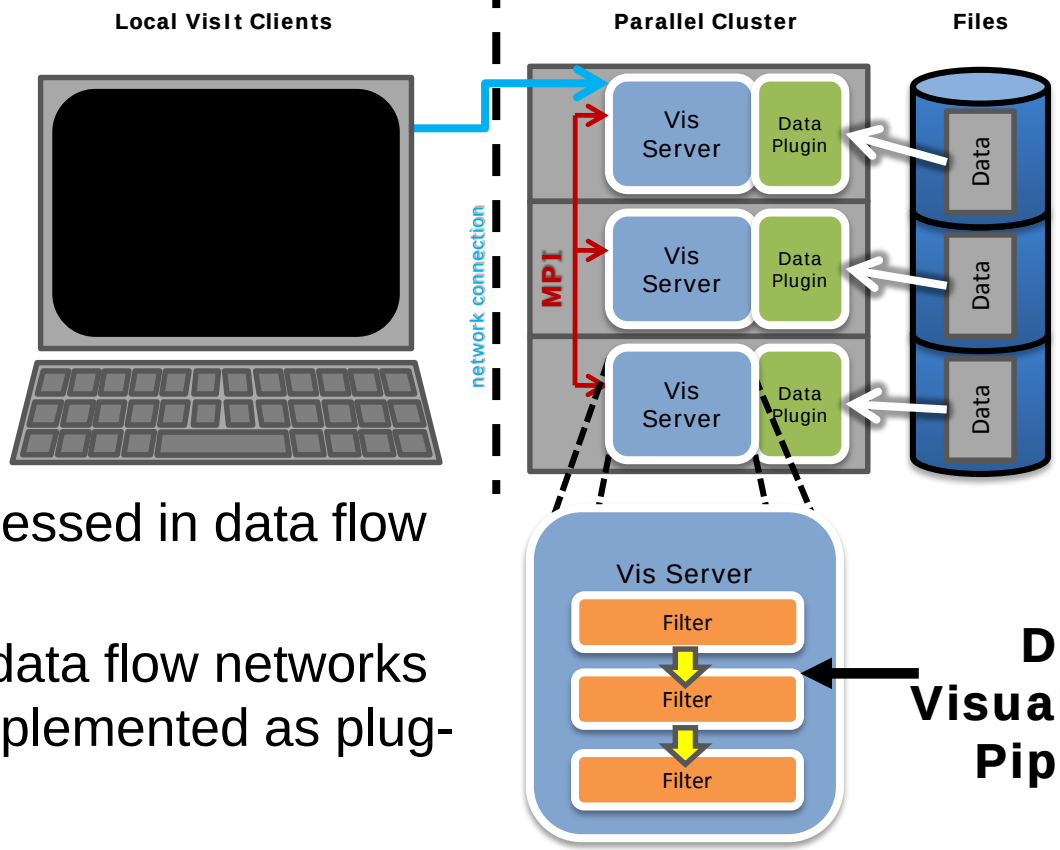
Users select simulations to open as if they were files

All of VisIt's existing functionality is accessible

Visualization Tool Architecture

- Clients runs locally and display results computed on the server

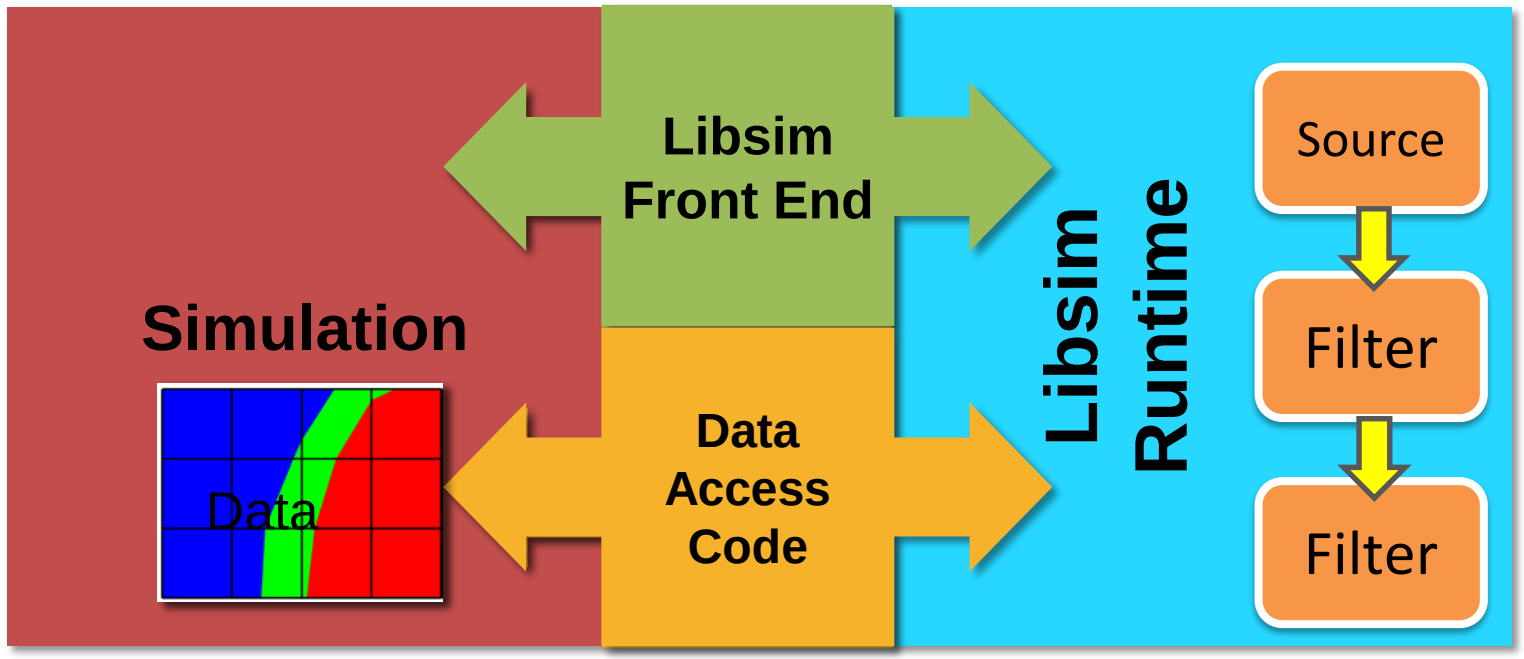
Server runs remotely in parallel, handling data processing for client



- Data processed in data flow networks
- Filters in data flow networks can be implemented as plug-ins

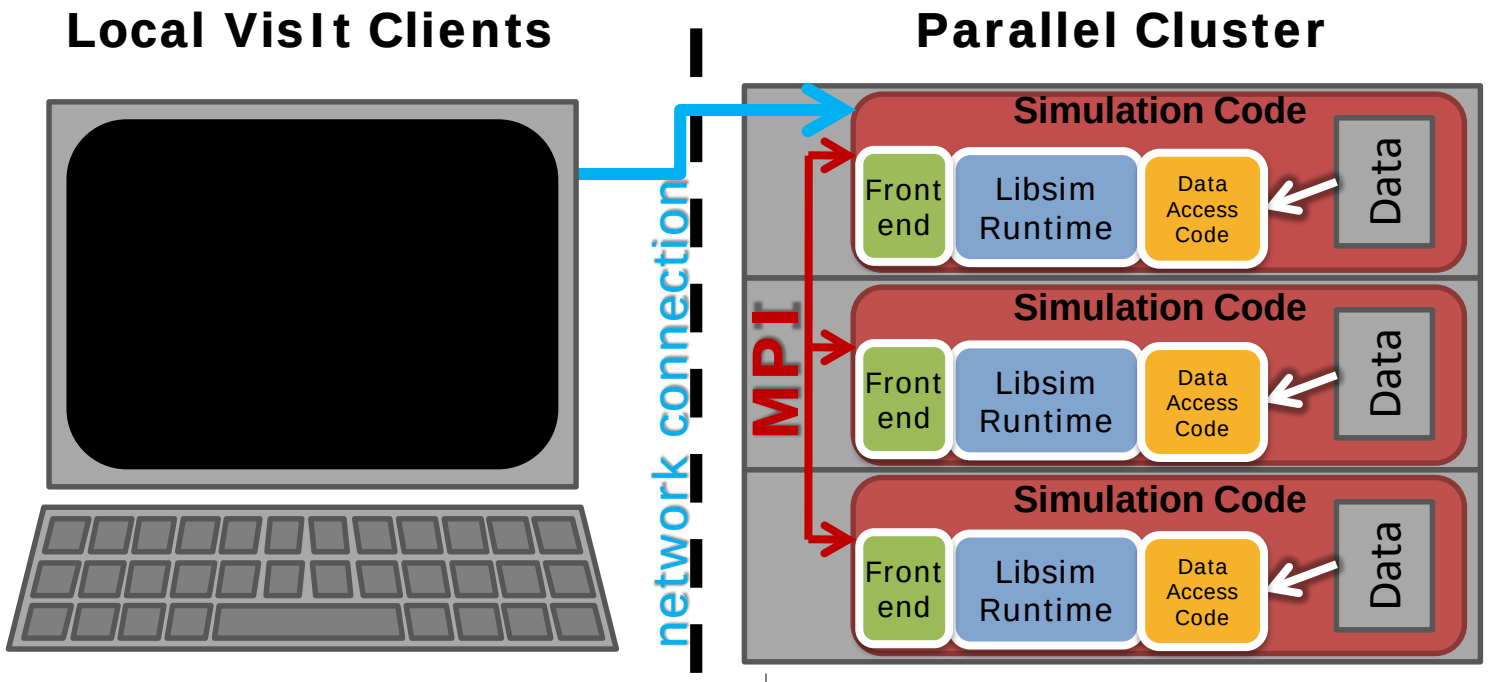
Coupling of Simulations and VisIt

Libsim is a VisIt library that simulations use to enable couplings between simulations and VisIt. Not a special package. It is an integral part of VisIt.



A Simulation using Libsim

- Front-end library lets VisIt connect
- Runtime library processes the simulation's data
- Runtime library obtains data on demand through user-supplied *Data Access Code* callback functions



In Situ Processing Workflow

1. The simulation code launches and starts execution
2. The simulation regularly checks for connection attempts from visualization tool
3. The visualization tool connects to the visualization
4. The simulation provides a description of its meshes and data types
5. Visualization operations are handled via Libsim and result in data requests to the simulation

Instrumenting a Simulation

Additions to the source code are usually minimal, and follow three incremental steps:

Initialize Libsim and alter the simulation's main iterative loop to listen for connections from VisIt.

Create *data access callback* functions so simulation can share data with Libsim.

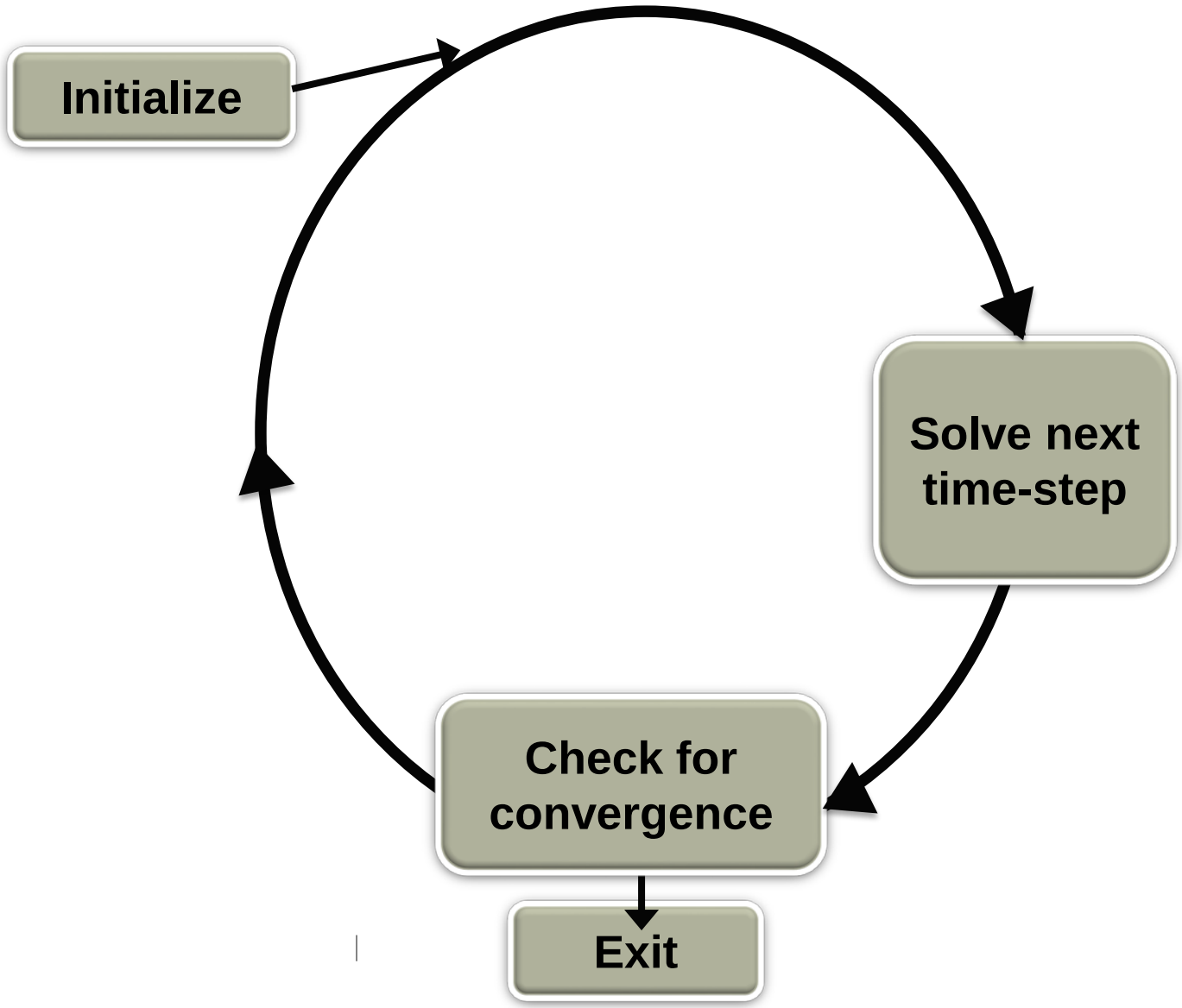
Add control functions that let VisIt steer the simulation.

Instrumenting Application's flow diagram (before and after)

Connection to the visualization library is optional

Execution is *step-by-step* or in *continuous* mode

Live connection can be closed and re-opened at later time

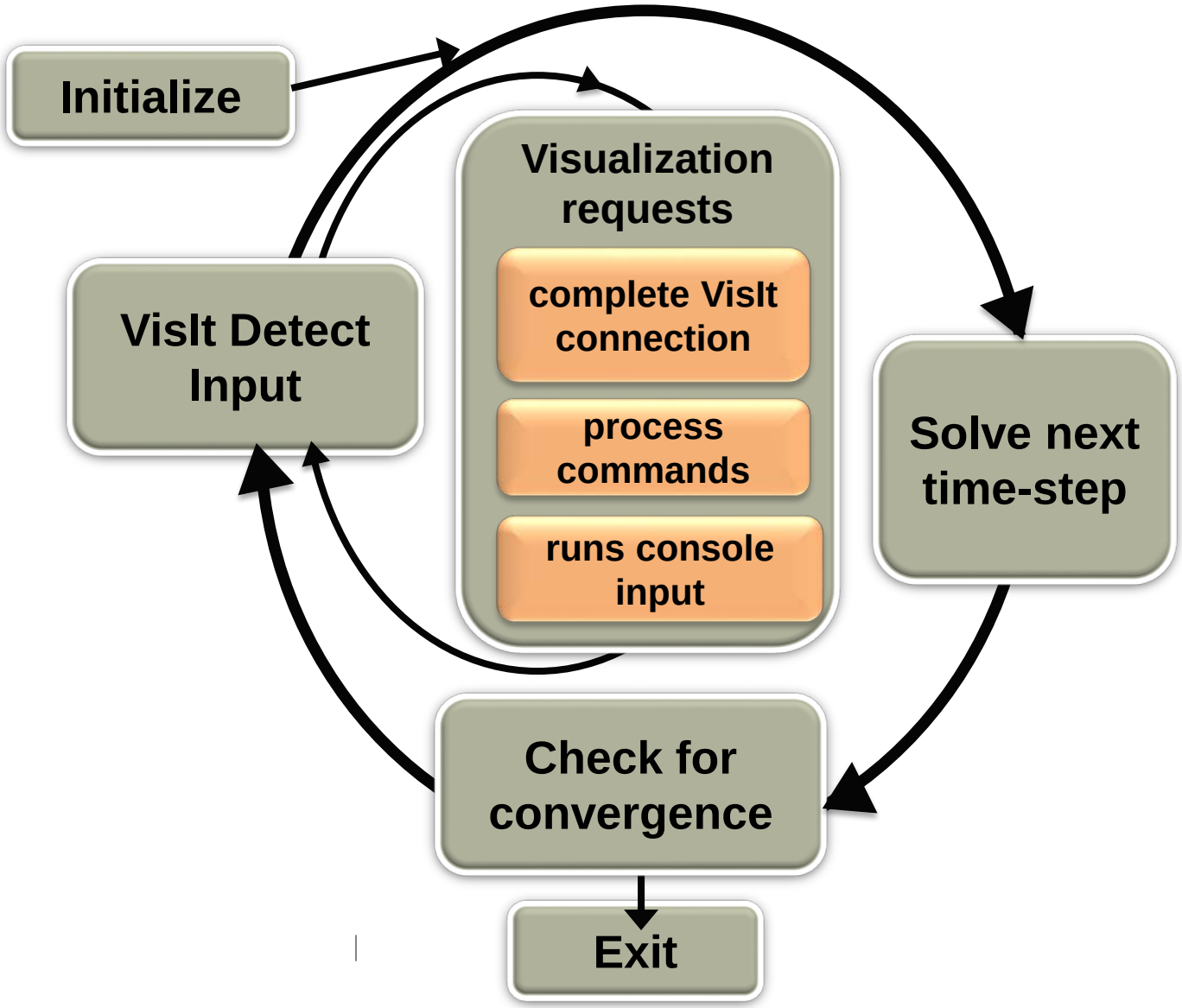


VisIt in-the-loop

Libsim opens a socket and writes out connection parameters

VisItDetectInput checks for:

- Connection request
- VisIt commands
- Console input



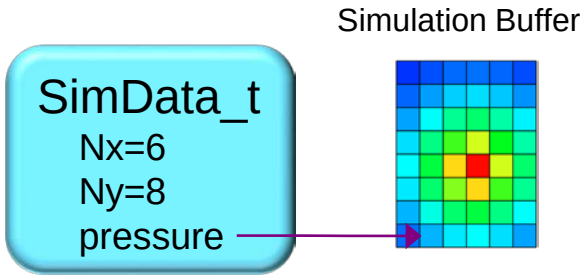
Data-access callbacks

VisIt requests data *on demand* through data access callback functions

- Return actual pointers to your simulation's data (nearly zero-copy)
- Return alternate representation that Libsim can free
- Written in C, C++, Fortran, Python

Sharing Data Example

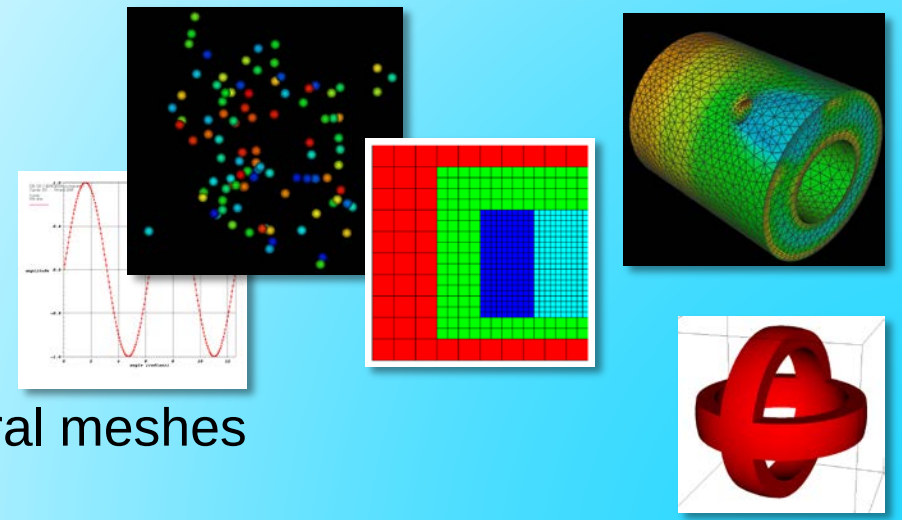
```
// Example Data Access Callback
visit_handle
GetVariable(int domain, char *name,
void *cbdata)
{
    visit_handle h = VISIT_INVALID_HANDLE;
    SimData_t *sim = (SimData_t *)cbdata;
    if(strcmp(name, "pressure") == 0)
    {
        VisIt_VariableData_alloc(&h);
        VisIt_VariableData_setDataD(h,
            VISIT_OWNER_SIM,
            1, sim->nx*sim->ny,
            sim->pressure);
    }
    return h;
}
```



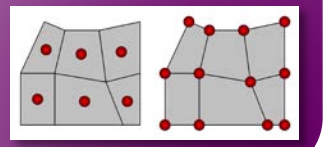
Pass simulation buffer to Libsim

Supported Data Model

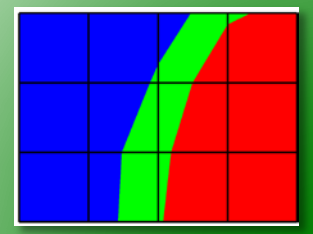
- Mesh Types
  - Structured meshes
  - Point meshes
  - CSG meshes
  - AMR meshes
  - Unstructured & Polyhedral meshes



- Variables
  - 1 to N components
  - Zonal and Nodal

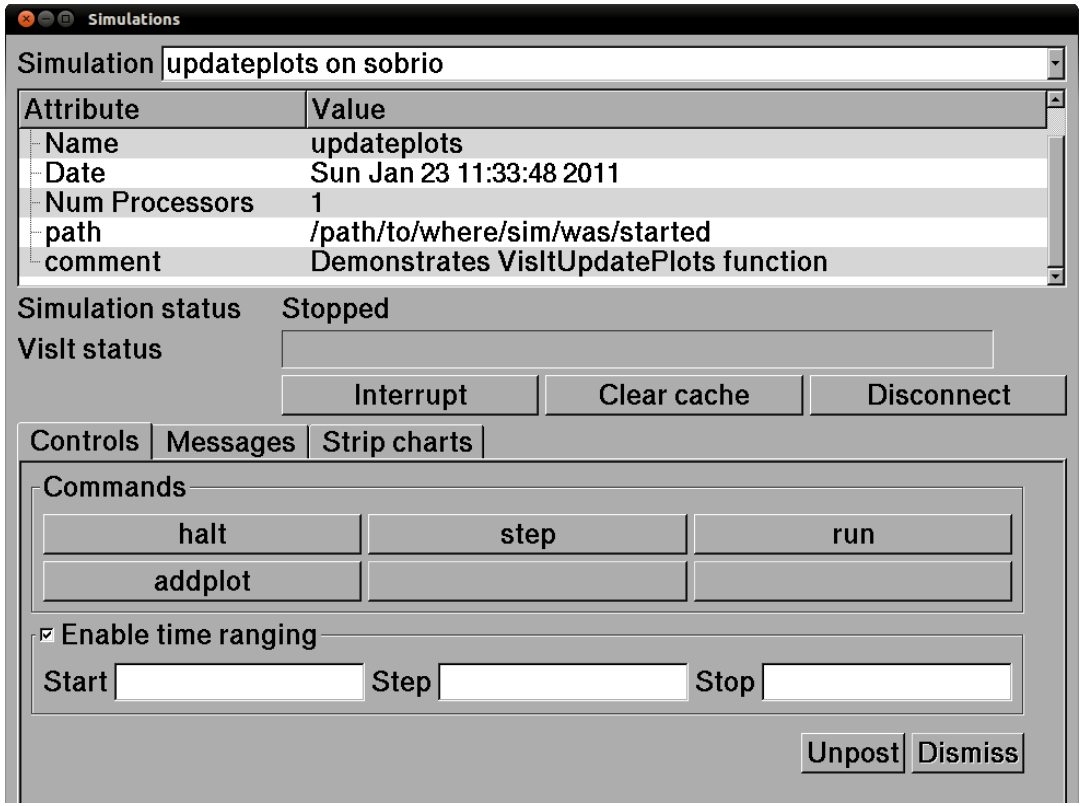


- Materials
- Species



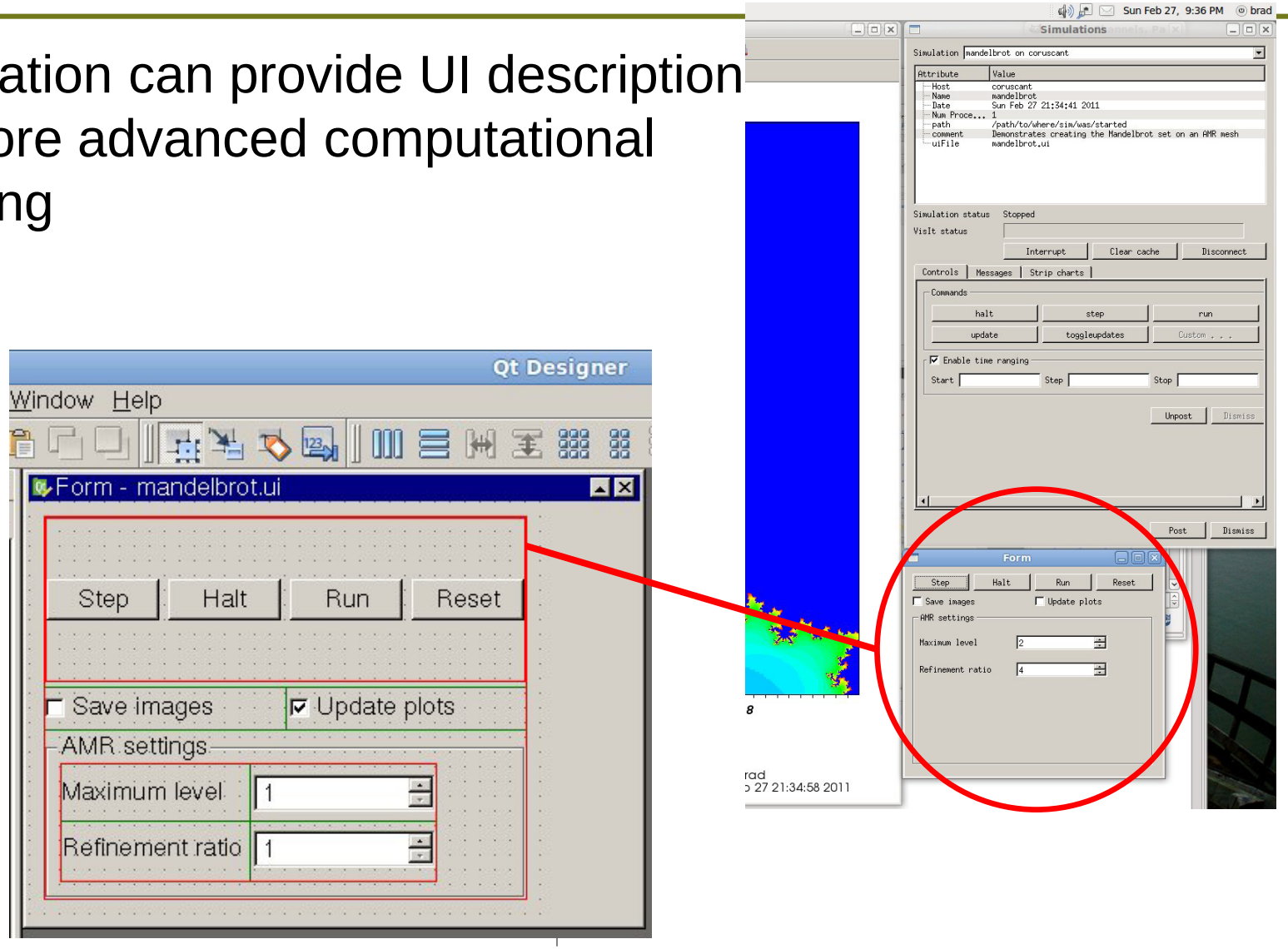
Adding Control Functions

- The simulation provides commands to which it will respond
- Commands generate user interface controls in Simulations Window



Custom User Interfaces

Simulation can provide UI description for more advanced computational steering



Advantages compared to saving files

- The greatest bottleneck (disk I/O) is eliminated
- Not restricted by limitations of any file format
- No need to reconstruct ghost-cells from archived data
- All time steps are potentially accessible
- All problem variables can be visualized
- Internal data arrays can be exposed or used
- Parallel compute nodes are already allocated

- The simulation can watch for a particular event and trigger the update of the VisIt plots

Libsim enables flexible workflows

Interactive exploration:

- Use the VisIt client to connect to your simulation and explore
- Simulations are like any other data source

Batch mode data extracts:

- Create automated routines to generate data in batch
  - Program directly using Libsim
  - Use VisIt session files

Libsim resources

Information about instrumenting a simulation can be found here:

- Getting Data Into VisIt
- (https://wci.llnl.gov/codes/visit/2.0.0/GettingDataIntoVisIt2.0.0.pdf)
- VisIt Example Simulations
- (http://visit.ilight.com/trunk/src/tools/DataManualExamples/Simulations)
- VisIt Wiki (http://www.visitusers.org)
- VisIt Email List (visit-users@email.ornl.gov)