SC11 Seattle, WA Connecting communities through HPC

LLNL-PRES-477655

Lawrence Livermore National Laboratory

In Situ Visualization using VisIt

Brad Whitlock
Lawrence Livermore National Laboratory

Jean M. Favre
Swiss National Supercomputing Centre

Jeremy S. Meredith
Oak Ridge National Laboratory

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

What We're Doing

- We have created a library that lets simulations interface to a fully featured parallel visualization system
  - One goal is to avoid the high costs of I/O associated with writing and then reading the data
  - Another goal is to interactively explore data
- We have used our library to instrument a parallel cosmology code to investigate some of its performance aspects compared to doing I/O

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

SC11 Seattle, WA Connecting communities through HPC

Case For Using In Situ

- I/O in supercomputers has not kept pace with compute power
- Some applications report 90% of time spent in I/O *[Peterka et al.]*
- Post processing simulation files requires write then read, paying for I/O twice in different application
- In Situ may let us avoid some I/O

Machine	Year	Writable FLOPS	Whole-System Checkpoint
ASCI Red	1997	0.075%	300 sec
ASCI Blue Pacific	1998	0.041%	400 sec
ASCI White	2001	0.026%	480 sec
ASCI Red Storm	2005	0.035%	660 sec
ASCI Purple	2005	0.025%	500 sec
NCCS XT4	2007	0.004%	1400 sec
Roadrunner	2008	0.005%	480 sec
NCCS XT5	2008	0.005%	1250 sec
ASC Sequoia	2012 (planned)	0.001%	3200 sec

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

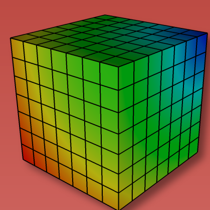
A Marriage Between Two Fairly Inflexible Partners…

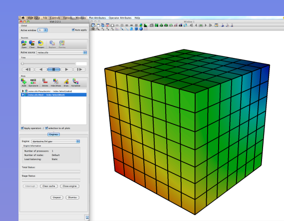
Our library enables a general purpose visualization tool to be flexibly coupled with a simulation.

Simulation

Between

Analysis Application





This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

SC11 Seattle, WA Connecting communities through HPC

In Situ Processing Strategies

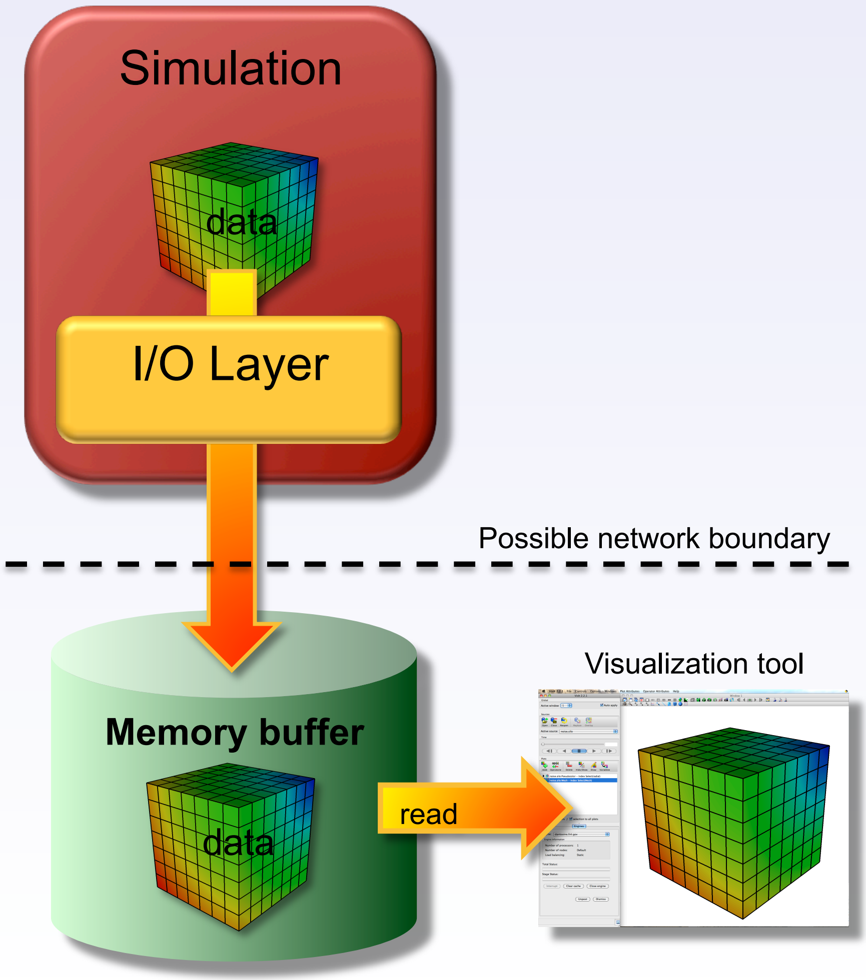
We find 3 main strategies for in situ processing:

In Situ Strategy	Description	Negative Aspects
Loosely coupled	Visualization and analysis run on concurrent resources and access data over network	1) Data movement costs 2) Requires separate resources
Tightly coupled	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

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Loosely Coupled In Situ Processing

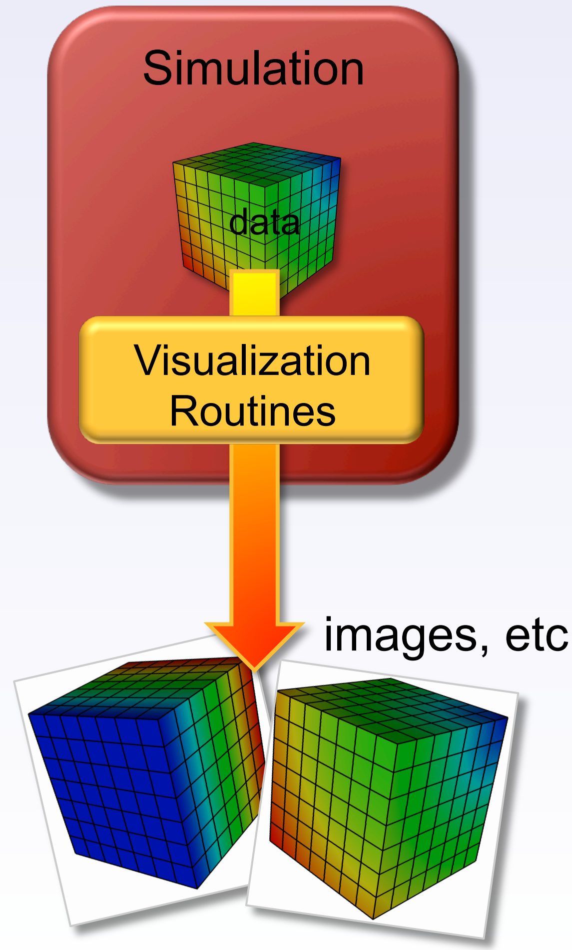
- I/O layer stages data into secondary memory buffers, possibly on other compute nodes
- Visualization applications access the buffers and obtain data
- Separates visualization processing from simulation processing
- Copies and moves data



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Tightly Coupled Custom In Situ Processing

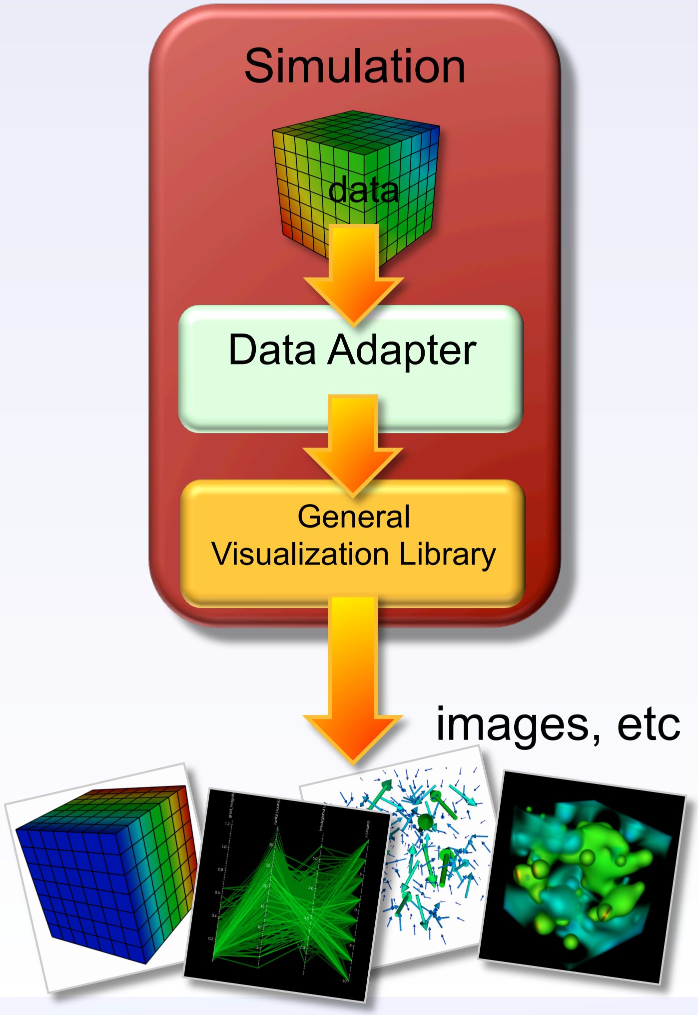
- Custom visualization routines are developed specifically for the simulation and are called as subroutines
  - Create best visual representation
  - Optimized for data layout
- Tendency to concentrate on very specific visualization scenarios
- *Write once, use once*



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

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*



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Which Strategy is Appropriate?

There have been many excellent solutions in this space. Different circumstances often merit different solutions.

	Tightly Coupled	Loosely Coupled	Hybrid
Custom	✖		
General	✖	✖	

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Design Philosophy

- Visualization and analysis will be done in the same memory space as the simulation on native data to avoid duplication
- 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
  - Emphasis on interactive exploration
  - Debugging
  - Computational steering

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

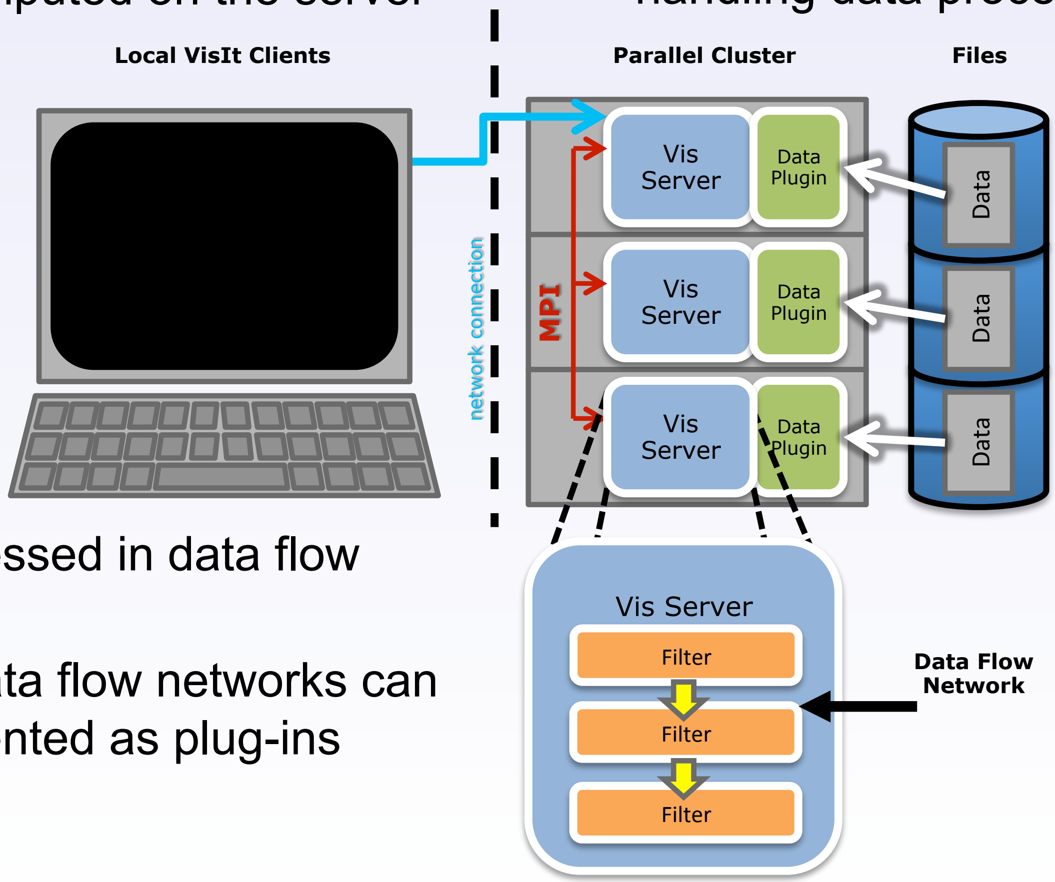
Selecting an In Situ Strategy

- Our strategy is tightly coupled, yet general
- Fully featured visualization code connects interactively to running simulation
  - Allows live exploration of data for when we don't know visualization setup a priori
  - Opportunities for steering
- We chose VisIt as the visualization code
  - VisIt runs on several HPC platforms
  - VisIt has been used at many levels of concurrency
  - We know how to develop for VisIt

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Visualization Tool Architecture

- Clients runs locally and display results computed on the server
- Server runs remotely in parallel, handling data processing for client

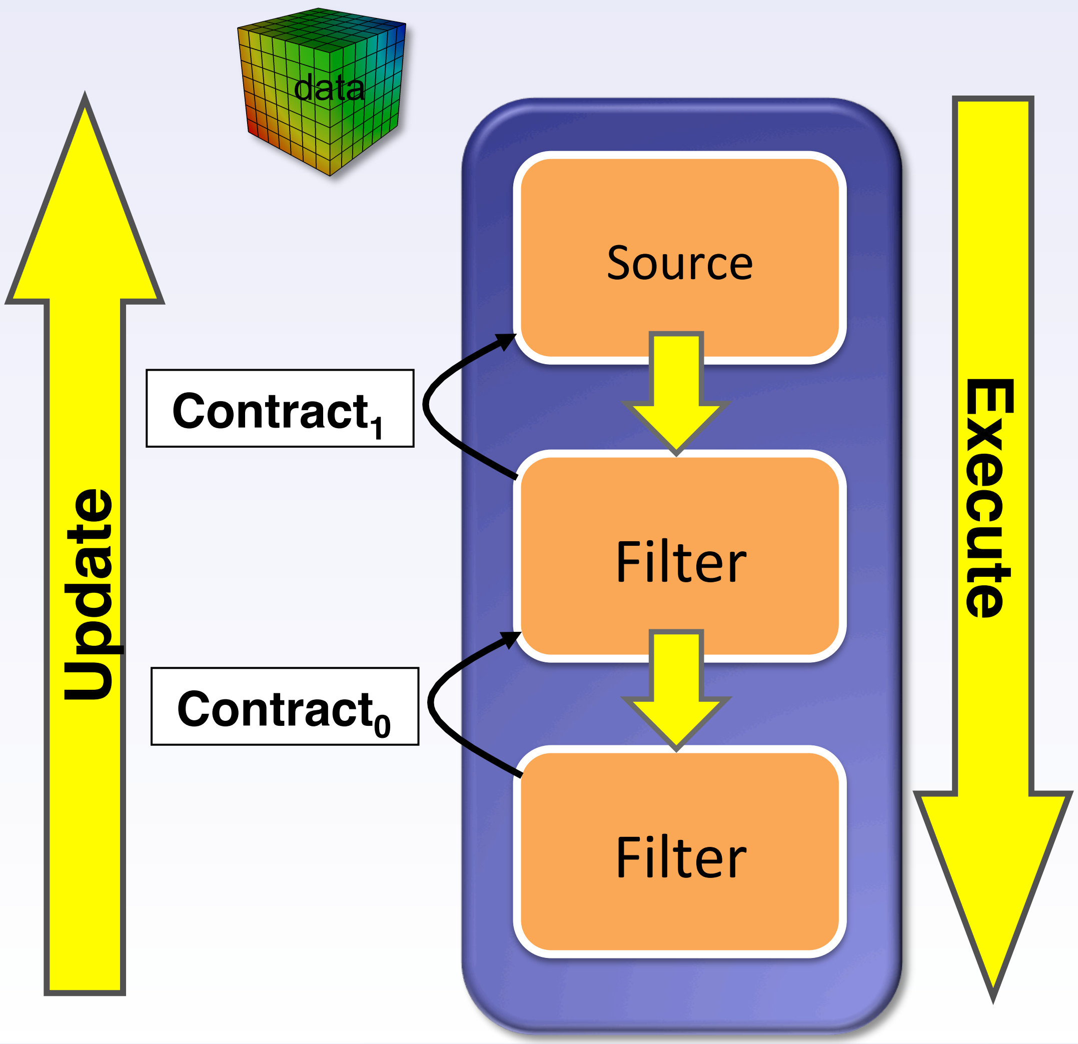


- Data processed in data flow networks
- Filters in data flow networks can be implemented as plug-ins

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL-PRES-477655

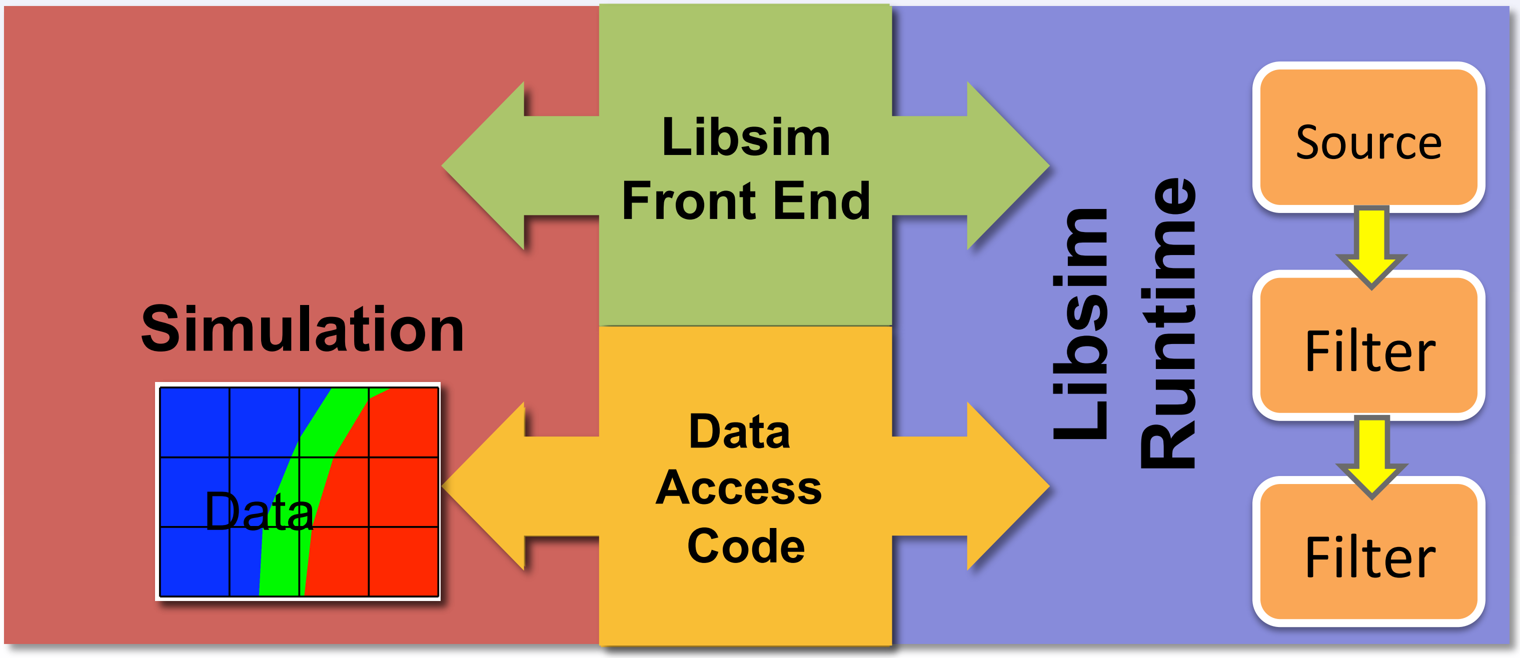
Coordination Among Filters Using Contracts



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Coupling of Simulations and VisIt

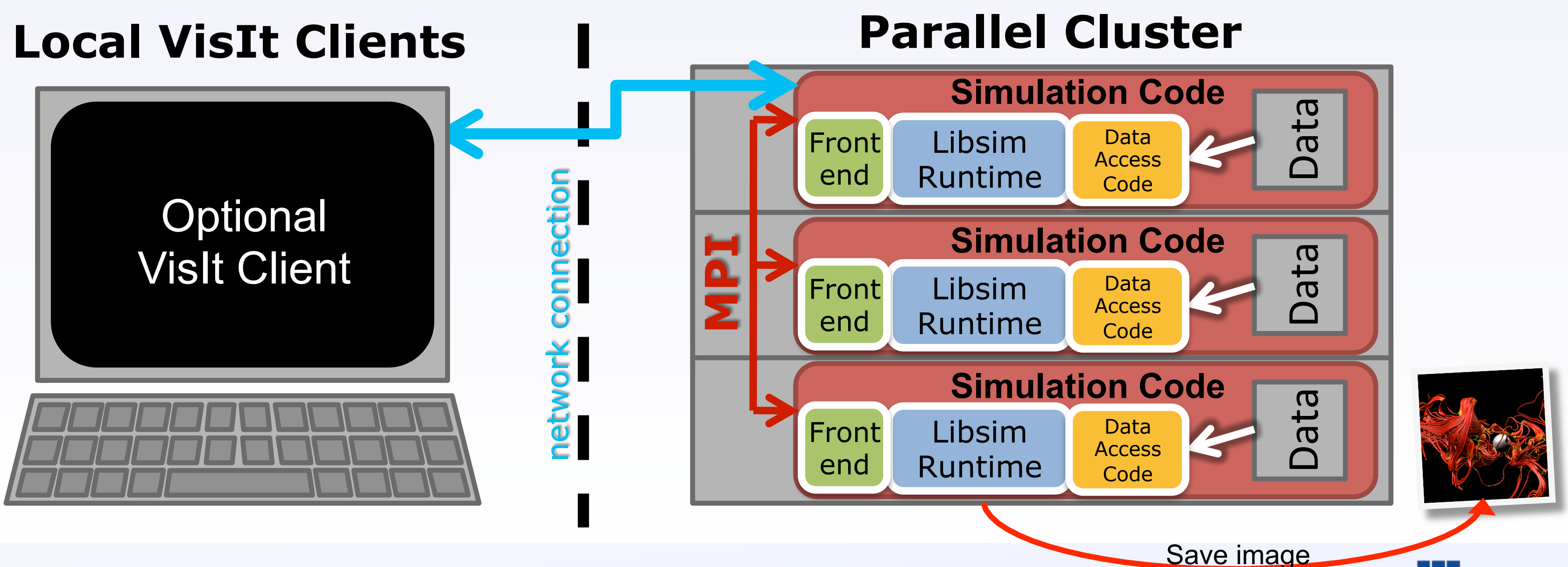
- We created Libsim, a library that simulations use to let VisIt connect and access their data



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Libsim Implements Tight Coupling

- Front end library controls access and plotting
- Data requested on demand through user-supplied *Data Access Code* callback functions
- Data shared via pointers
- Ported to Linux, Windows, and MacOS X
- Distributed in every version of VisIt



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

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 may connect to the simulation or the simulation may save images by itself
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

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

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.

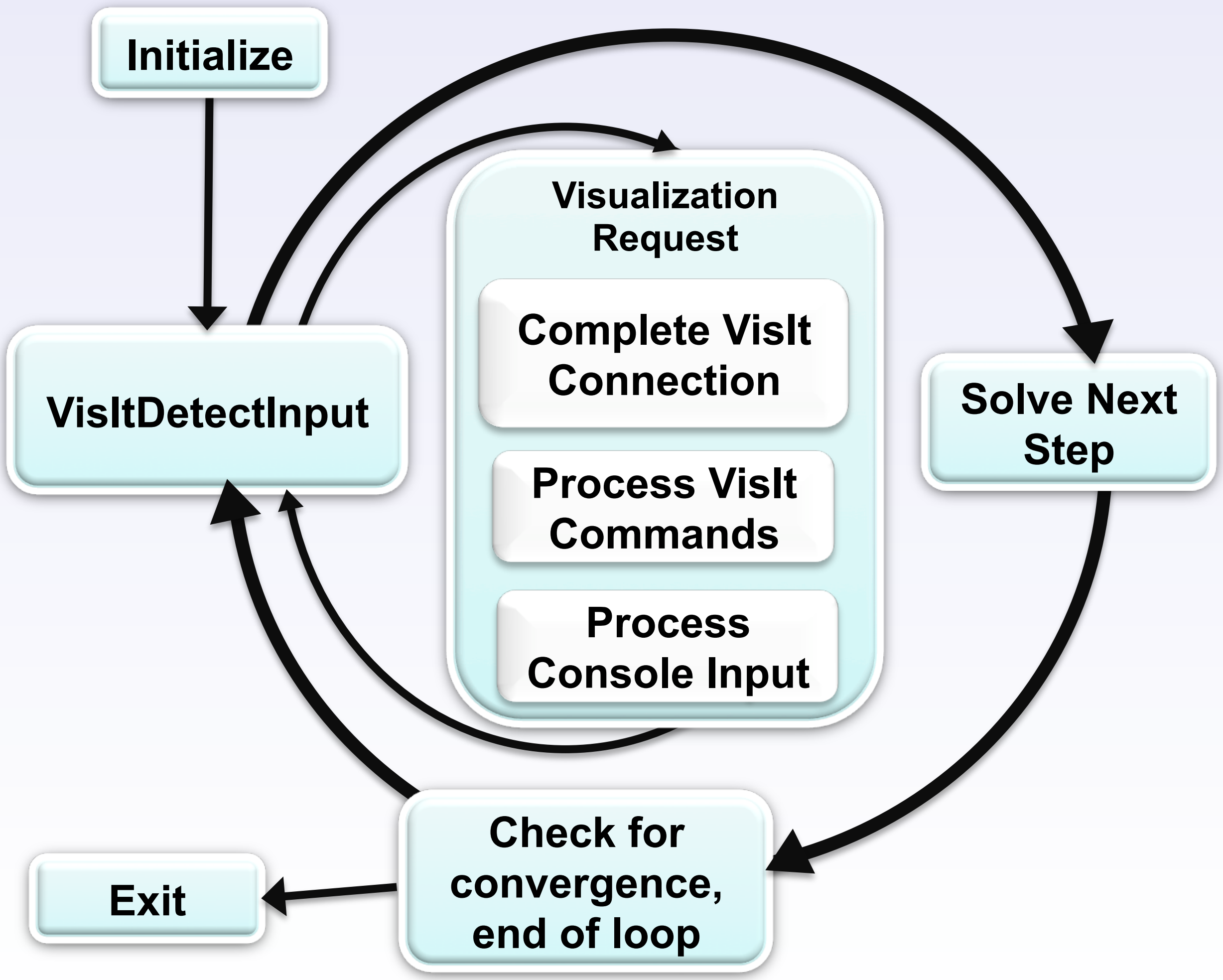
Create *data access callback* functions so simulation can share data with Libsim.

Add control functions that let VisIt steer the simulation.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Adapting the Main Loop

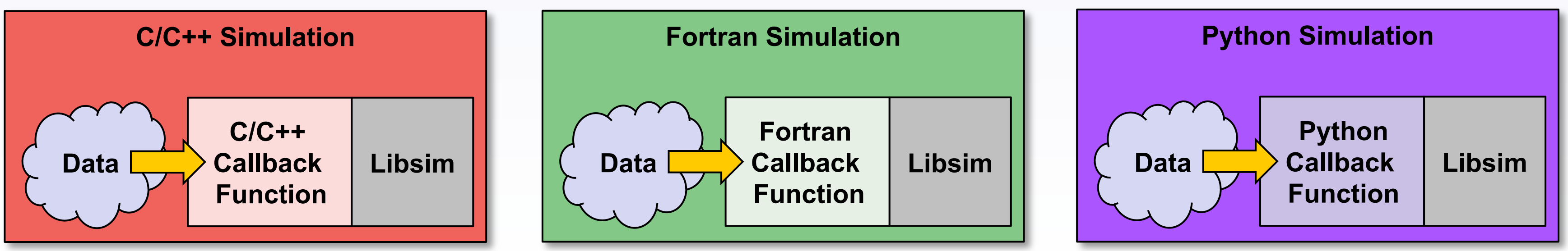
- Libsim opens a socket and writes out connection parameters
- VisItDetectInput checks for:
  - Connection request
  - VisIt commands
  - Console input



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Sharing Data

- VisIt requests data on demand through *Data Access Callback Functions*
  - The best-suited simulations allocate large contiguous memory arrays
  - Return actual pointers to your simulation's data
  - Return alternate representation that Libsim can free
  - Written in C, C++, Fortran, Python

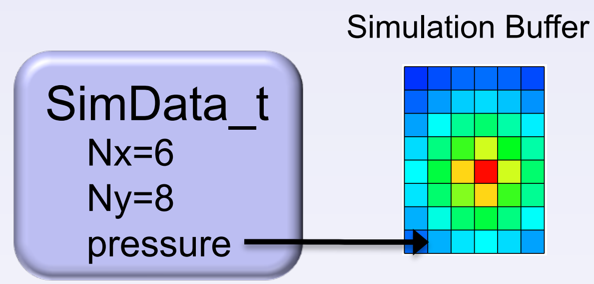


This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

SC11 Seattle, WA Connecting communities through HPC

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

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Data Access Callbacks

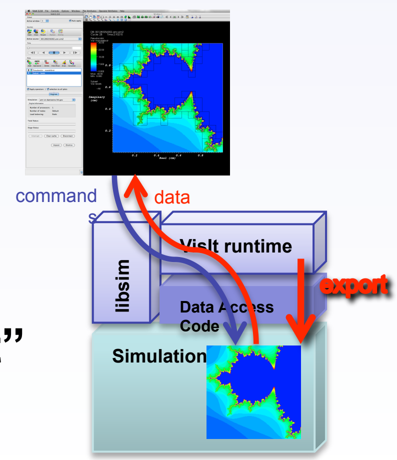
- GetMetaData
- GetMesh
- GetVariable
- GetMaterial
- GetSpecies
- GetDomainList
- GetDomainBoundaries
- GetDomainNesting

GetMetaData, GetMesh, GetVariable: Most simulations implement these

GetMaterial, GetSpecies: Simulations that use materials

GetDomainList, GetDomainBoundaries: Parallel simulations

GetDomainNesting: AMR simulations

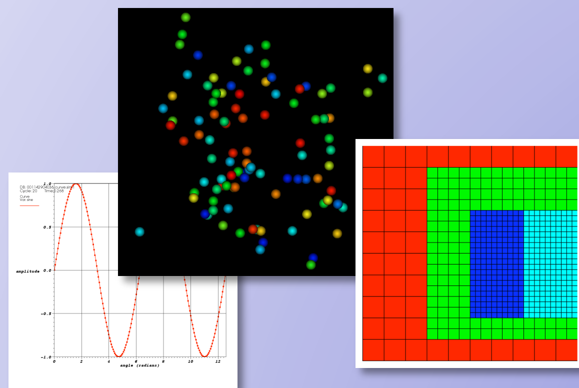


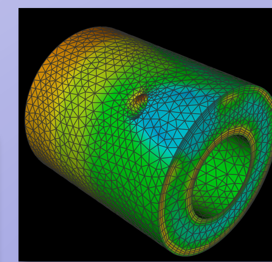
There are also write callbacks that can be used to “export” the processed data back to the simulation

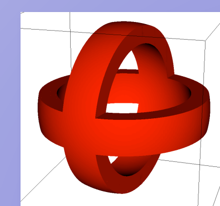
This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Supported Data Model

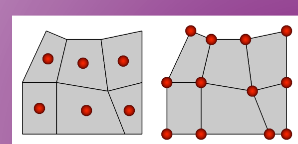
- Mesh Types
  - Structured meshes
  - Point meshes
  - CSG meshes
  - AMR meshes
  - Unstructured & Polyhedral meshes



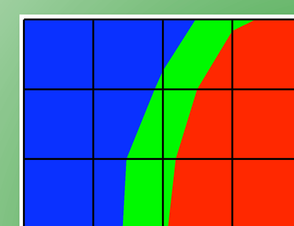




- Variables
  - 1 to N components
  - Zonal and Nodal



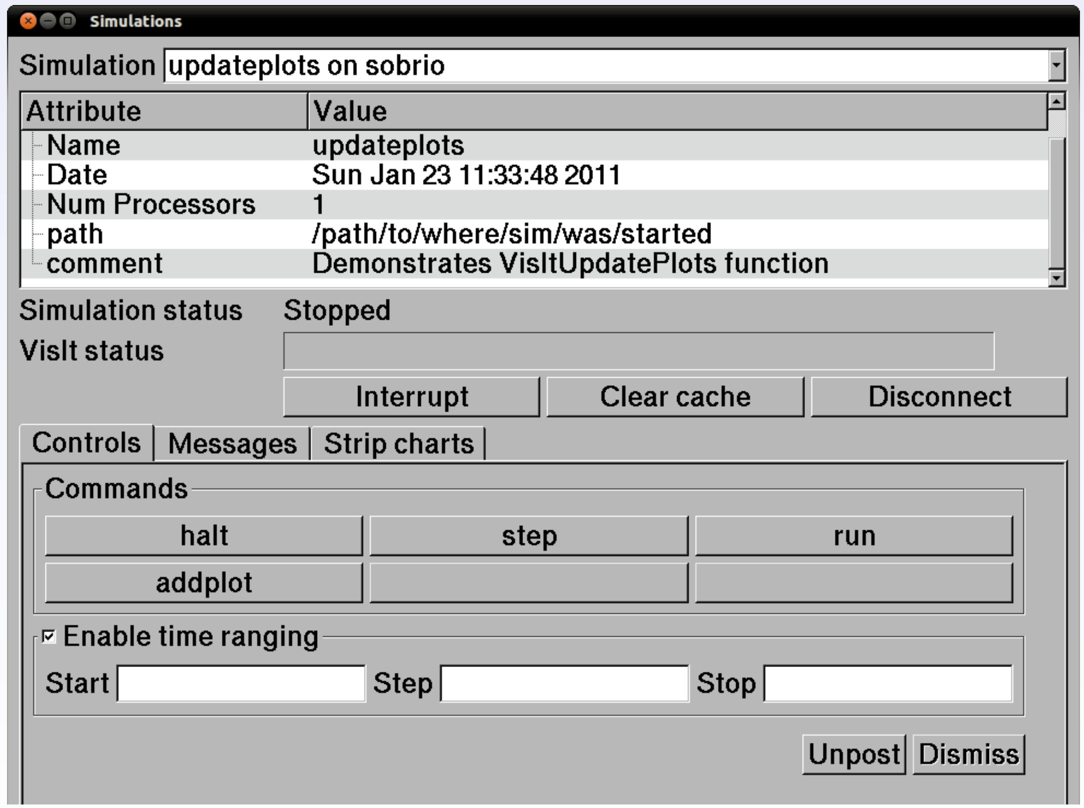
- Materials
- Species



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Adding Control Functions

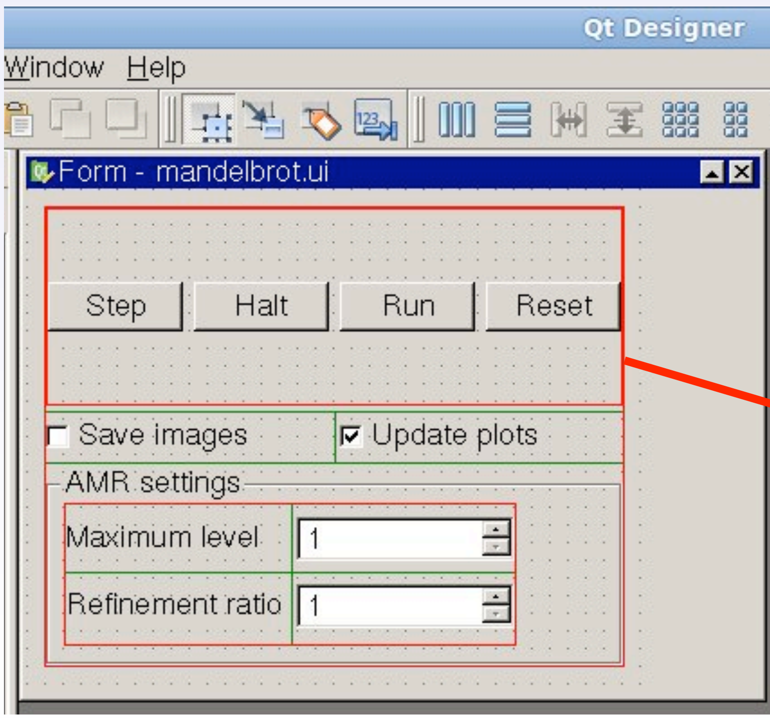
- The simulation provides commands to which it will respond
- Commands generate user interface controls in Simulations Window

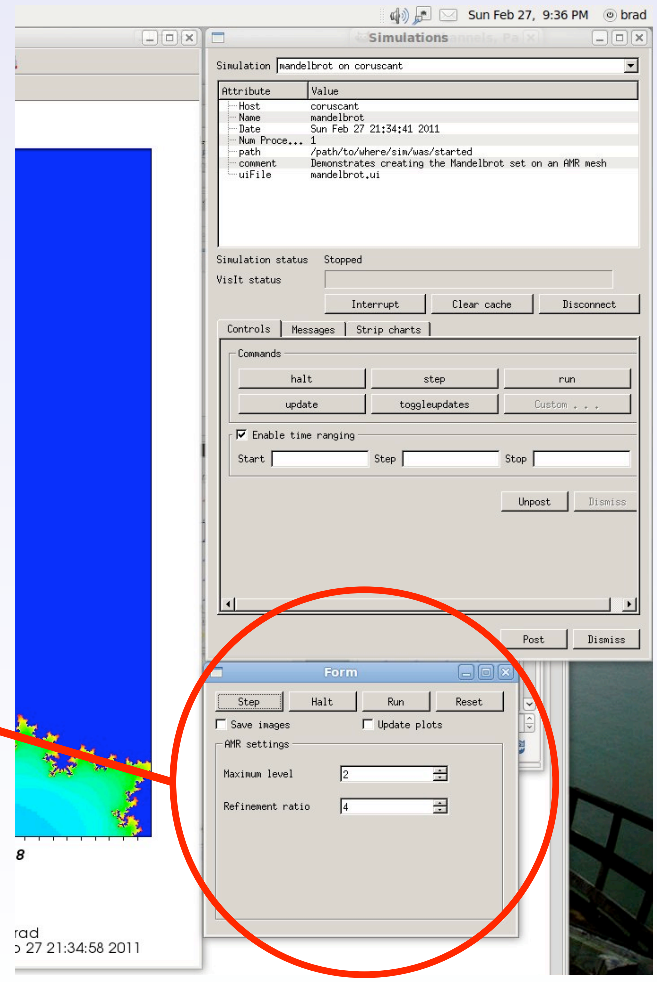


This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Custom User Interfaces

- Simulation can provide UI description to generate custom simulation window in VisIt





Libsim in Practice

- We instrumented GADGET-2, a popular cosmology code, with Libsim and measured performance
- We conducted our experiments on a 216 node visualization cluster
  - Two, 6 core 2.8GHz Intel Xeon 5660 processors
  - 96Gb of memory per node
  - InfiniBandQDR high-speed interconnect
  - Lustre parallel file system
- We measured the impact of Libsim on a simulation’s main loop, without connecting to VisIt
- We measured memory usage after loading VisIt

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Impact on the Main Loop

- Measure cost of calling Libsim in the main loop
- Instrumenting the main loop for a parallel simulation requires calling *VisItDetectInput* and *MPI_Bcast*
  - We timed how long it took to call both using 512 cores
  - 10K main loop iterations

Can now avoid penalty by static linking

Cores	VisItDetectInput overhead	MPI_Bcast overhead	Overhead loading VisIt runtime libraries
512	2µs	8µs	1s (1 time cost)

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Impact on Memory Usage

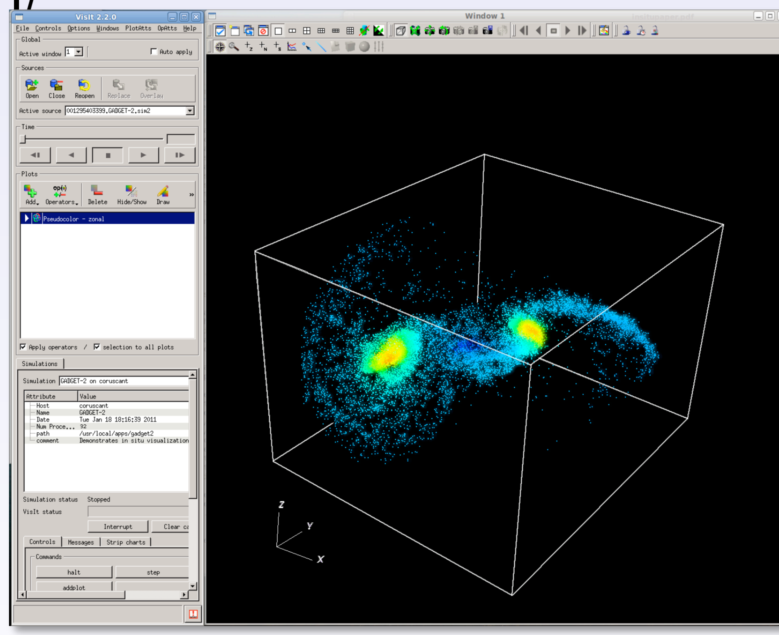
- Measure memory used before and after VisIt is connected
- Measured our updateplots example program
- Read values from /proc/<pid>/smaps

Event	Size	Resident Set Size
Simulation startup	8.75 Mb	512 Kb
After Libsim Initialization	8.75 Mb	614 Kb
After Loading VisIt	222 Mb	43.5 Mb

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Libsim/GADGET-2 Timing Results

- In situ competitive or faster than single file I/O with increasing cores
- It should be possible to do several in situ operations in the time needed for I/O
- Time savings compared to simulation followed by post processing



16M particles	32 cores	256 cores
I/O 1 file	2.76s	4.72s
I/O N files	0.74s	0.31s
In situ	0.77s	0.34s

100M particles	32 cores	256 cores	512 cores
I/O 1 file	24.45s	26.7s	25.27s
I/O N files	0.69s	1.43s	2.29s
In situ	1.70s	0.46s	0.64s

- I/O results are the average of 5 runs per test case
- In Situ results are averaged from timing logs for multiple cores

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

VisIt runs well at massive scale on diverse architectures

- 8K-32K cores
- Weak scaling study ~62.5M cells/core

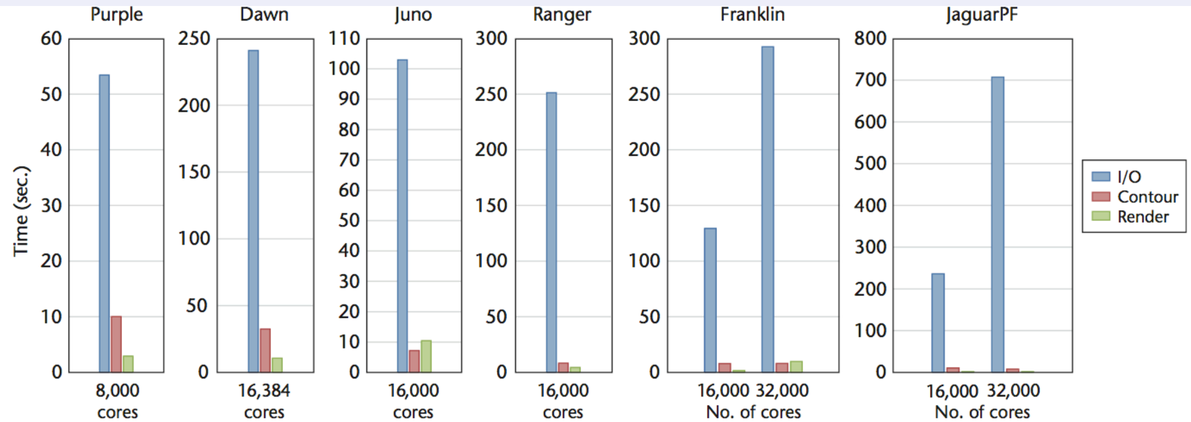
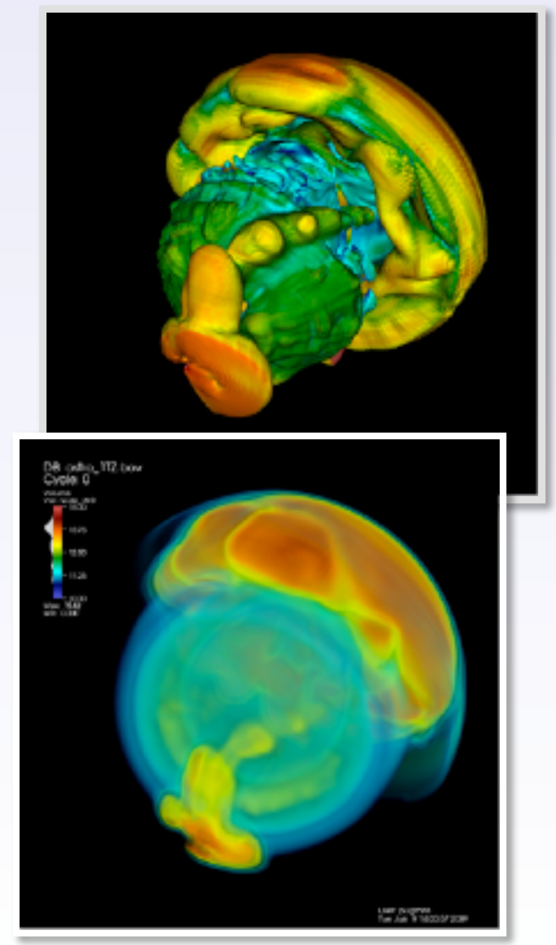


Figure 2. Runtimes for I/O, contouring, and rendering. These results show that, although there is variation across the supercomputers, I/O is the slowest phase.

Table 2. Performance across diverse architectures.

Machine	No. of cores	Data set size (TCells)	Total I/O time (sec.)	Contour time (sec.)	Total pipeline execution time (sec.)†	Rendering time (sec.)
Purple	8,000	0.5	53.4	10.0	63.7	2.9
Dawn	16,384*	1.0	240.9	32.4	277.6	10.6
Juno	16,000	1.0	102.9	7.2	110.4	10.4
Ranger	16,000	1.0	251.2	8.3	259.7	4.4
Franklin	16,000	1.0	129.3	7.9	137.3	1.6
JaguarPF	16,000	1.0	236.1	10.4	246.7	1.5
Franklin	32,000	2.0	292.4	8.0	300.6	9.7
JaguarPF	32,000	2.0	707.2	7.7	715.2	1.5

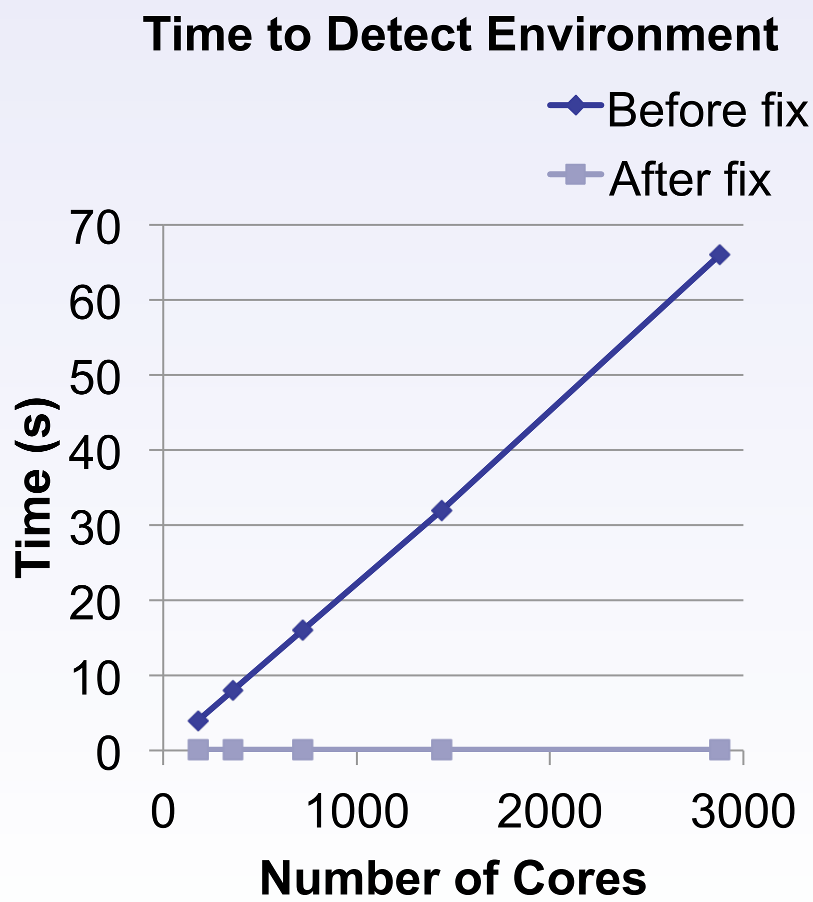
** Dawn requires that the number of cores be a power of two.*
† This measure indicates the time to produce the surface.



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Additional Results

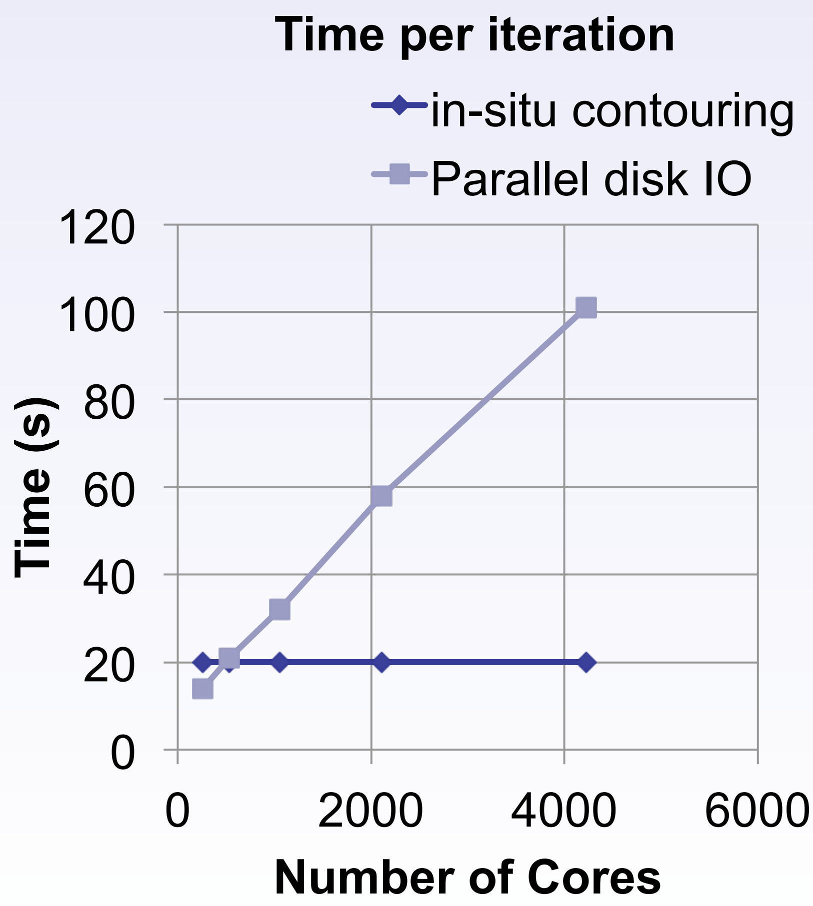
- We have recently instrumented additional simulations to investigate Libsim's scaling properties on a Cray XE6 using up to 4224 cores
- We identified and corrected a performance bottleneck in Libsim's environment detection functions



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Additional Results

- Simulation was run on 11, 22, 44, 88 and 176 nodes (24 cores/node)
- Each MPI task had a 512x512x100 block of data to isocontour at 10 different thresholds
- Parallel I/O to disk was done with netCDF-4, in files of size 27, 55, 110, 221, and 442 Gb per iteration



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

SC11 Seattle, WA Connecting communities through HPC

Work in progress

- Adding functions to create plots and operators so simulation can set up visualization without user intervention
- Libsim runtime loaded automatically; no VisIt session needed

```
int pcId = -1, sliceId = -1;
VisItAddPlot("Pseudocolor", "pressure", &pcId);
VisItAddOperator(pcId, "Slice");
VisItSetPlotOptionsS(pcId, "colorTable", "rainbow");
VisItDrawPlots(pcId);
VisItSaveWindow("image0000.png", 800, 800);
```


This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Limitations of Implementation

- Memory intensive
  - Runtime library cost is larger than with static-linking since we use the whole feature set
  - Filters may use intermediate memory
  - Zero-copy is not fully implemented
- Works best with an interactive session

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Future Work

- Continue adding functions for setting up visualization so in situ processing can be less user-driven
- Further limit resources consumed by the VisIt runtime libraries in order to lessen the impact that in situ analysis has on the simulation
- Characterize performance costs of using shared libraries on larger scale runs
- Simplify static linking

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Conclusion

- We have implemented Libsim, an easy to use library that enables in situ computations
  - Provides access to a fully featured, parallel visualization and analysis tool that excels at scale
  - Minimizes impact to simulation performance
  - Minimizes the amount of new code that must be written
  - Fully integrated with the open-source distribution of VisIt

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.