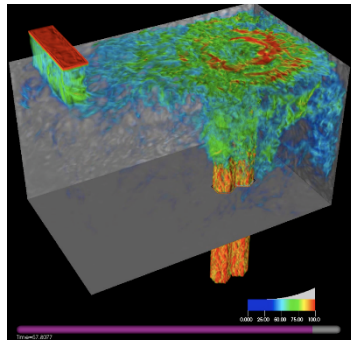
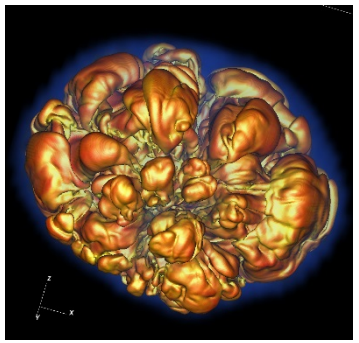
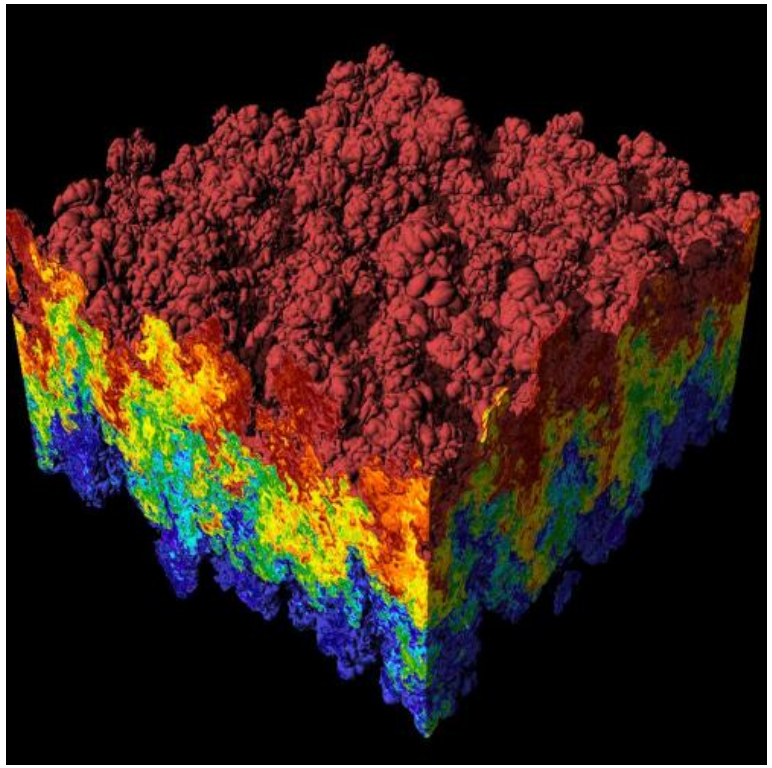
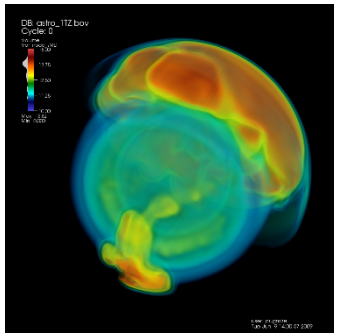
VisIt LibSim Tutorial

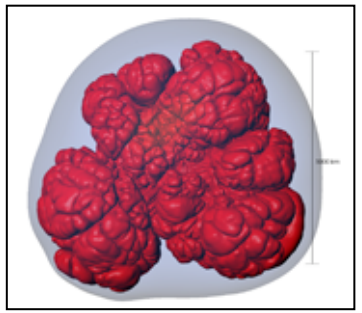
2nd Annual ECP Annual Meeting, February 2018

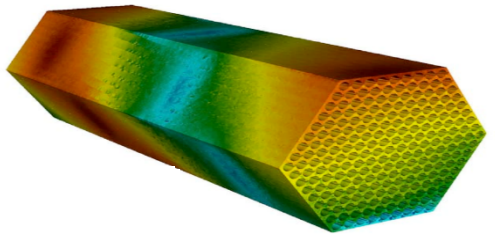


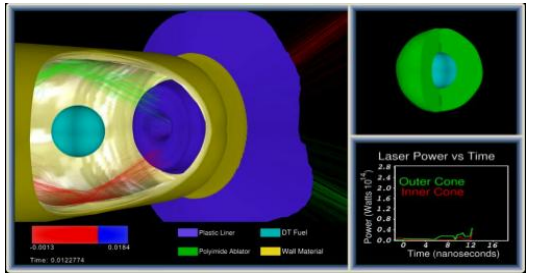












EXASCALE COMPUTING PROJECT

National Nuclear Security Administration

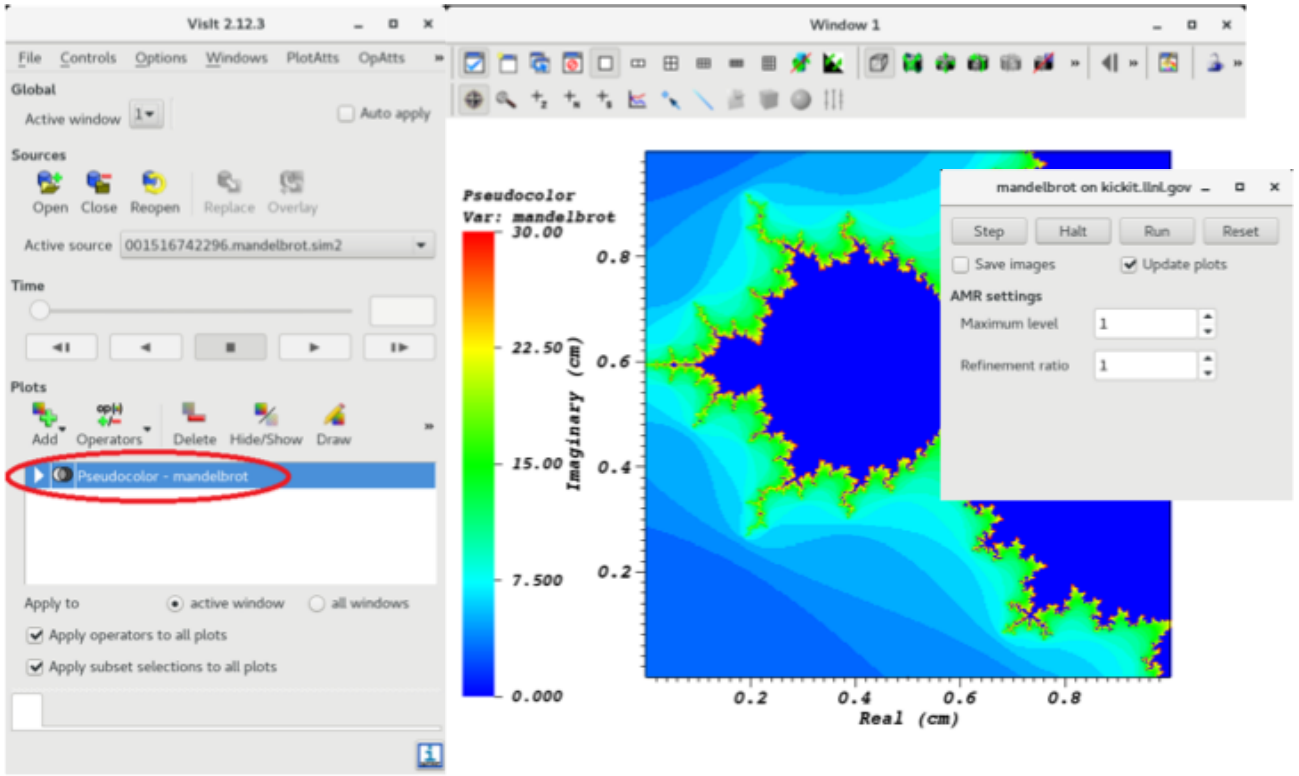
U.S. DEPARTMENT OF ENERGY | Office of Science

LLNL-PRES-745518

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

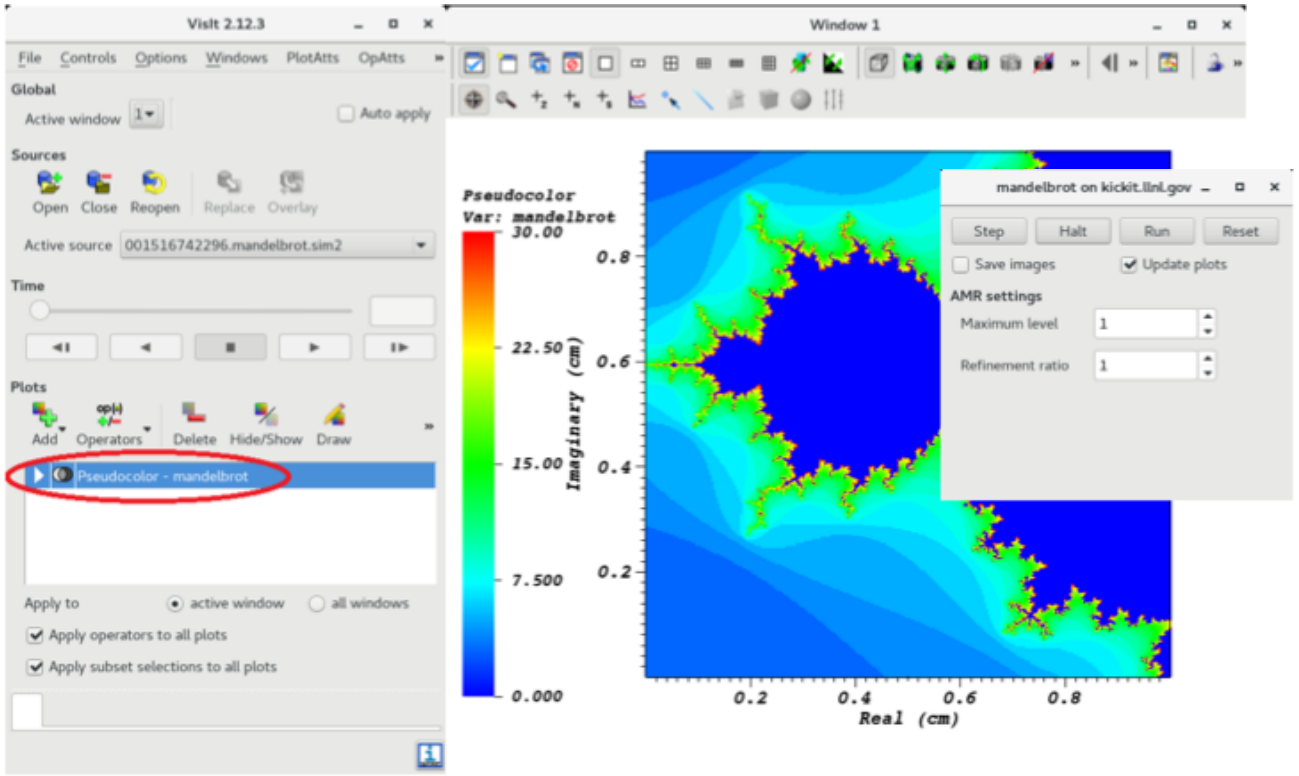
Agenda

- LibSim overview
- LibSim in action
- VisIt overview
- Integration details
- Sample integration with an AMR code
  - https://www.visitusers.org/index.php?title=VisIt-tutorial-in-situ
  - Go to visitusers.org and search on “in situ tutorial”



Agenda

- LibSim overview
- LibSim in action
- VisIt overview
- Integration details
- Sample integration with an AMR code



LibSim overview

Early History:

History of in-situ support in VisIt

- 2004 -- In-situ support was added by Jeremy Meredith
  - Structured meshes
  - Unstructured meshes
  - Scalar data
  - Materials
- 2005 – Whitlock added Fortran bindings
- 2009 – Whitlock added support for new data types
  - Species
  - Vector,Tensor data
  - AMR meshes
  - CSG meshes
- 2010 – Whitlock changed to function-based API
  - Does not rely on certain symbols being exposed
  - Additional error checking
  - Users don't allocate memory
  - Single style for C and Fortran

(Snapshot From Brad Whitlock's SC10 Tutorial Slides)

LibSim overview

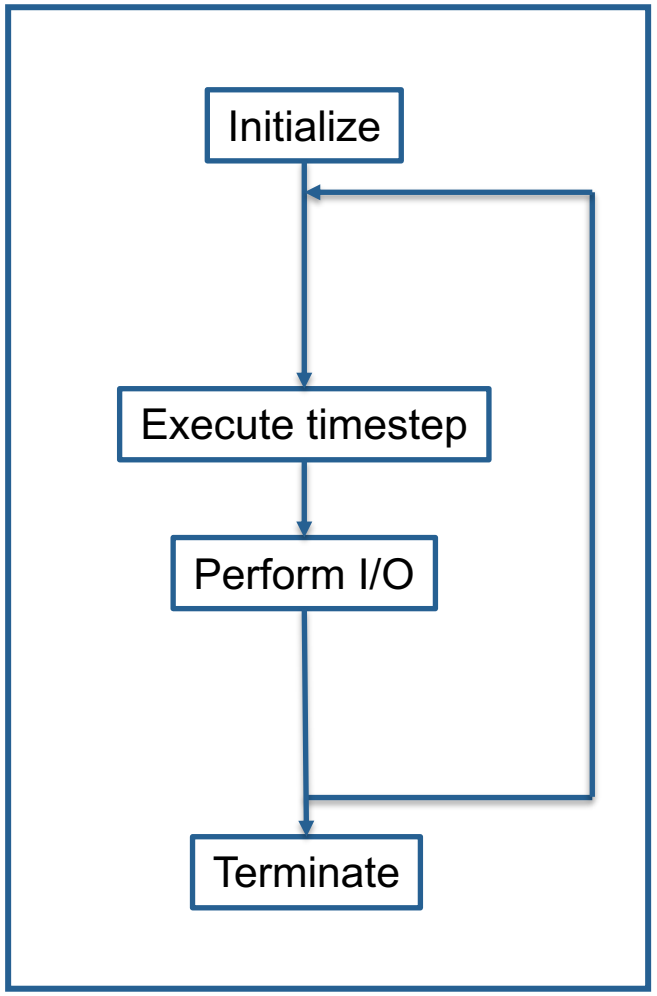
- LibSim was originally developed to allow VisIt to directly access a running simulations data
  - You interact with the simulation through the VisIt GUI
  - Instead of opening a file you open a simulation
  - You create plots and interact with the simulation as you would a file
  - All the functionality of VisIt is be available
- Later we added the ability to use LibSim in a batch mode to allow predefined visualizations to be executed in situ

LibSim overview

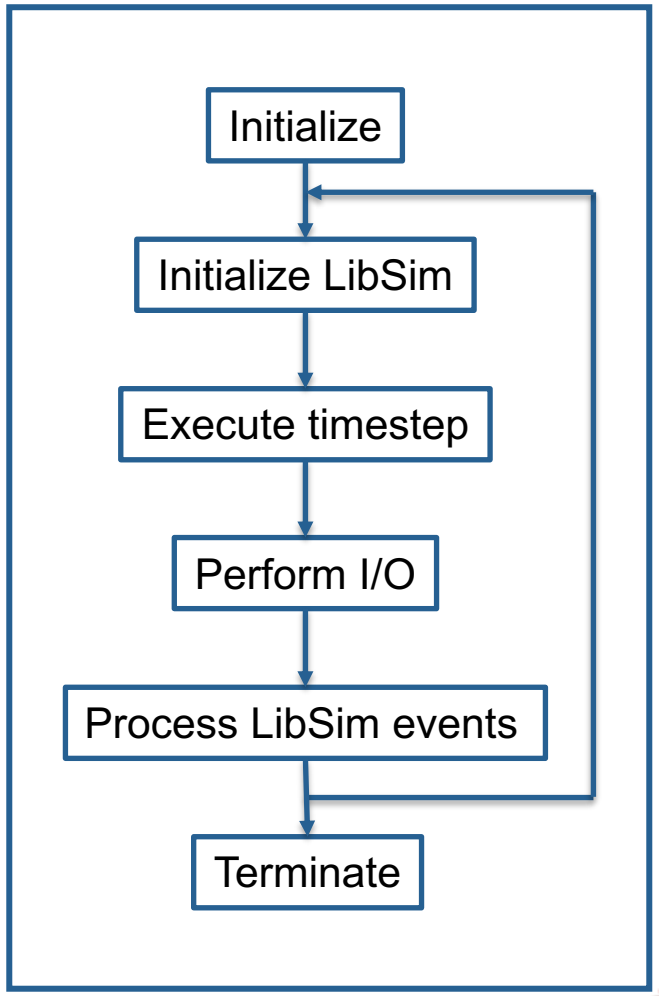
- LibSim has two modes of interaction
  - Interactive through the VisIt GUI
  - Batch mode
- LibSim has a dedicated API
  - Initialize the library
  - Return data to VisIt, typically zero-copy
  - Specify operations to perform
- LibSim shares resources with the simulation
  - Shares memory space
  - Time slices execution with the simulation

Typical integration

Typical simulation

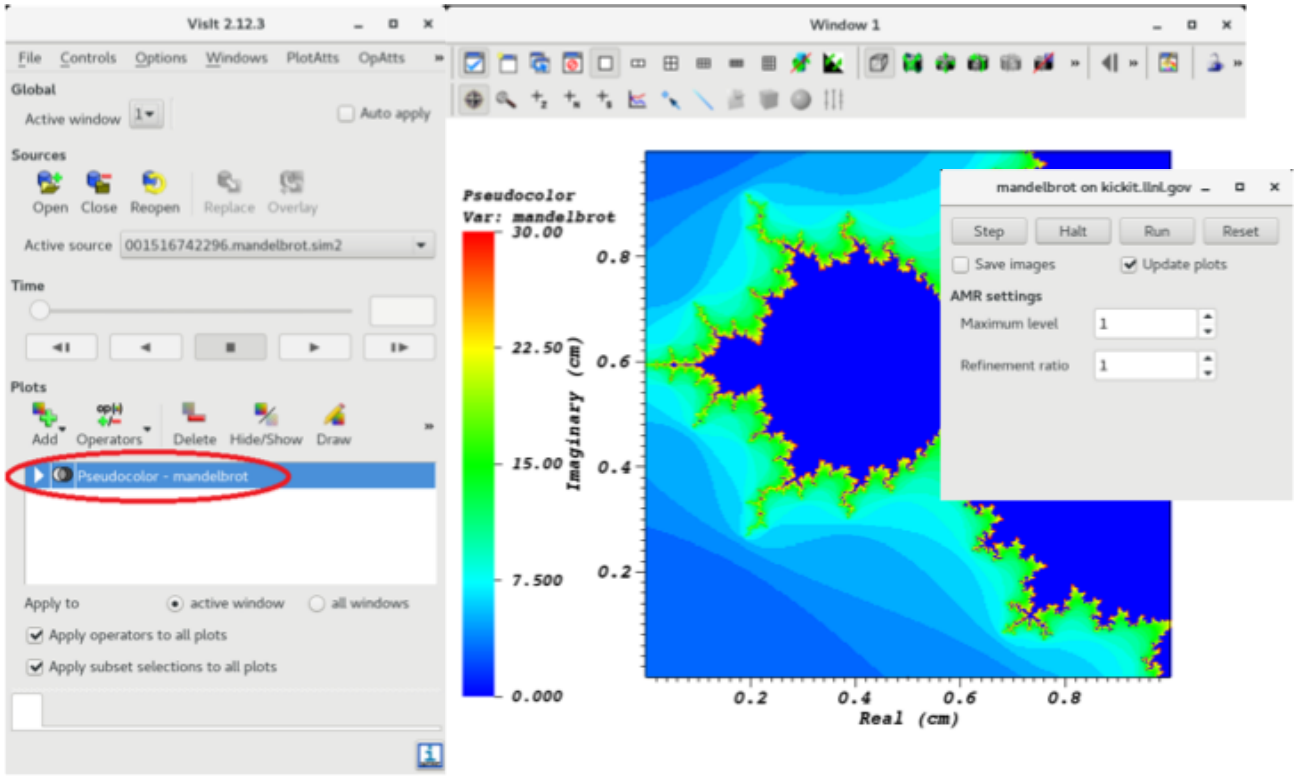


Simulation w/LibSim



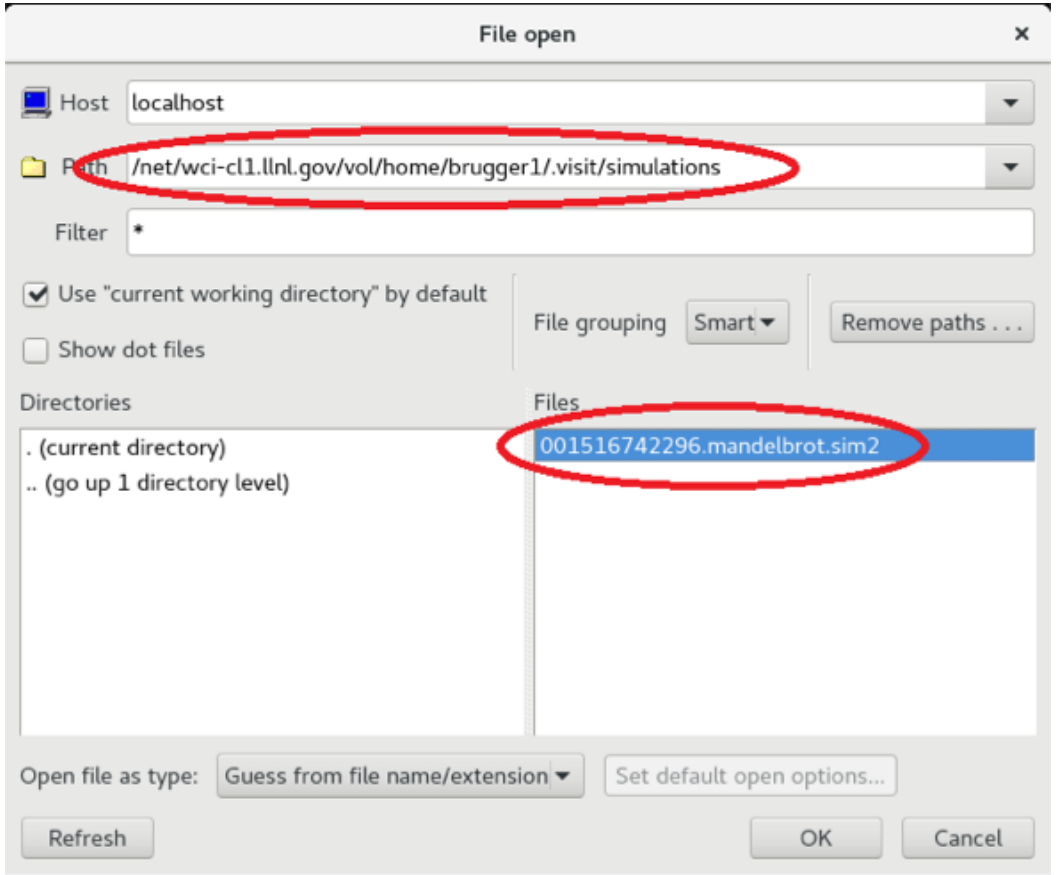
Agenda

- LibSim overview
- LibSim in action
- VisIt overview
- Integration details
- Sample integration with an AMR code

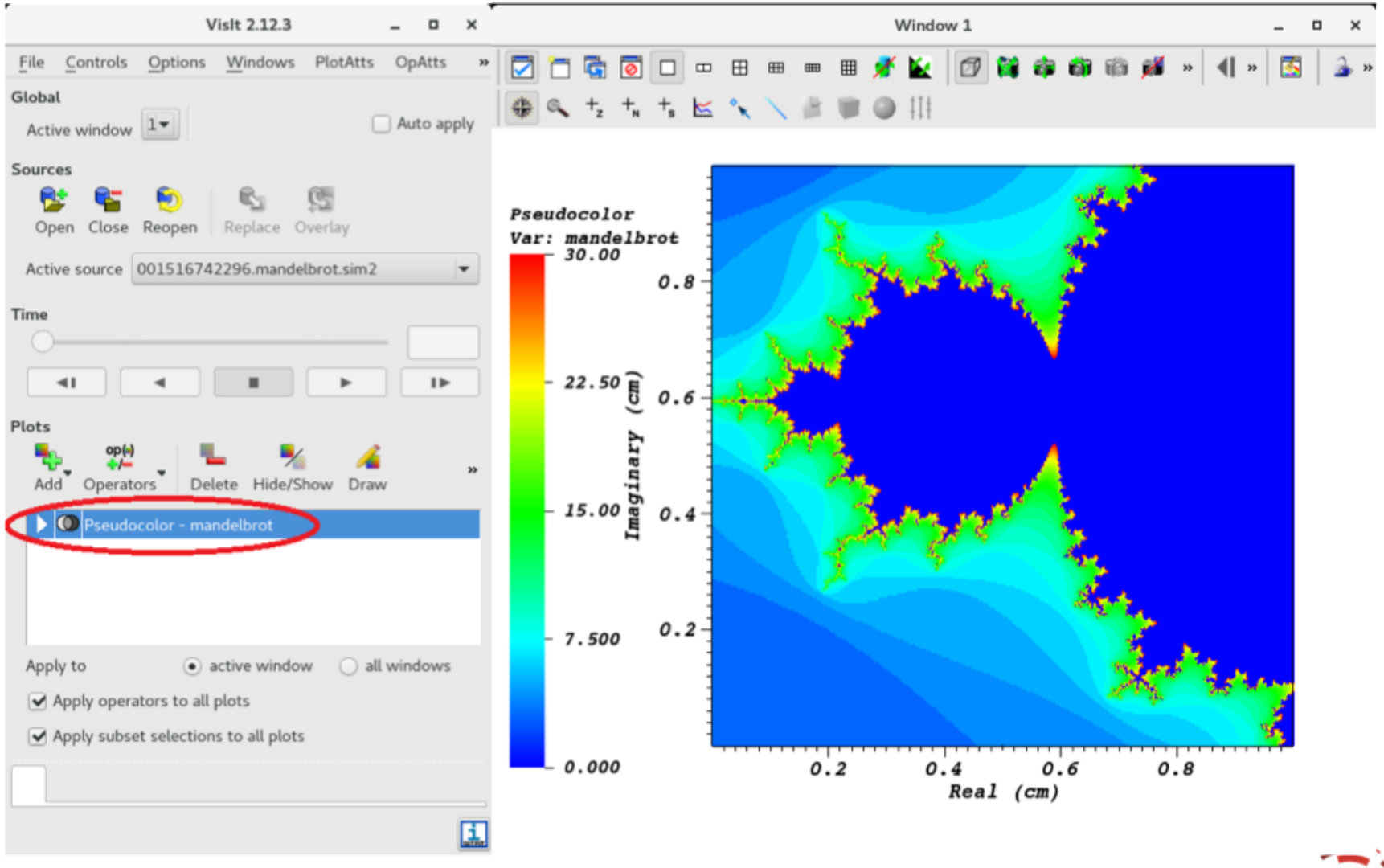


LibSim in action: Open the simulation

- When a simulation runs it places a file in the users home directory under .visit/simulations
- You open the file to connect to the simulation

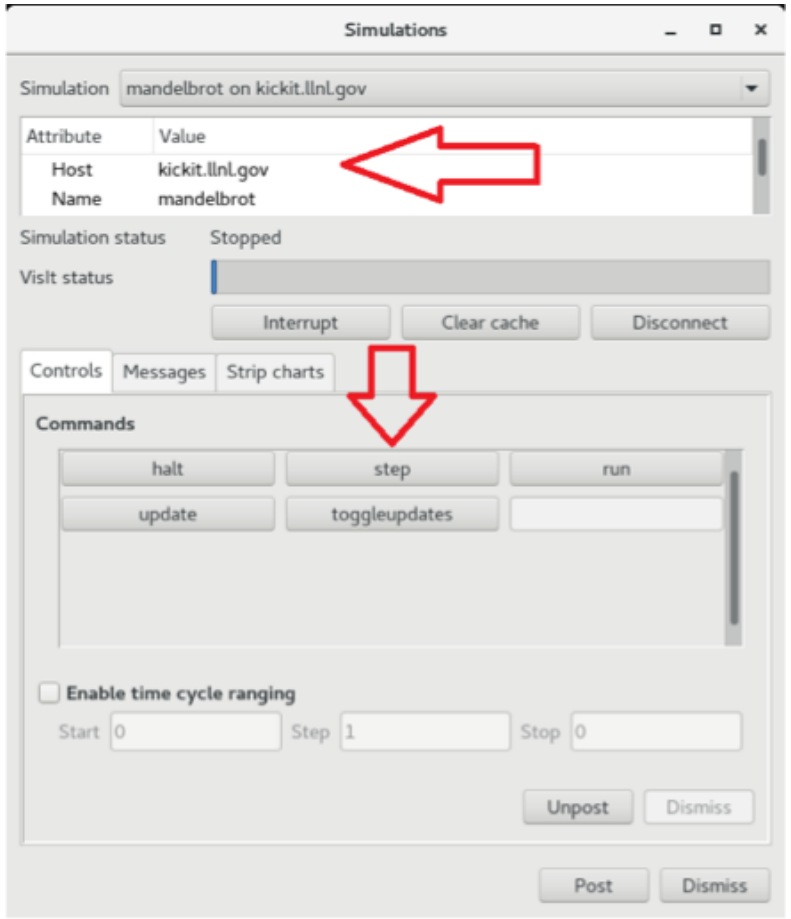


LibSim in action: Create plots



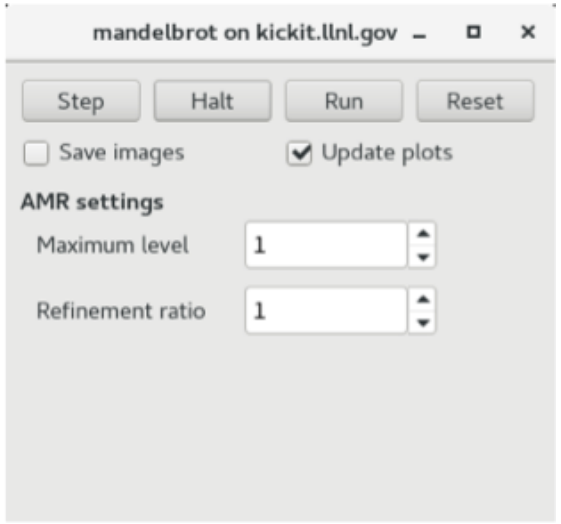
LibSim in action: Interact with the simulation

- Interact using VisIt's Simulations Window
  - *File Menu-> Simulations*
- The top portion displays information about the simulation
- Interacting with the simulation
  - Halt – stop the sim
  - Step – advance to next time step
  - Run – run the sim
  - Update – update the plots
  - Toggleupdates – turn on auto update of plots



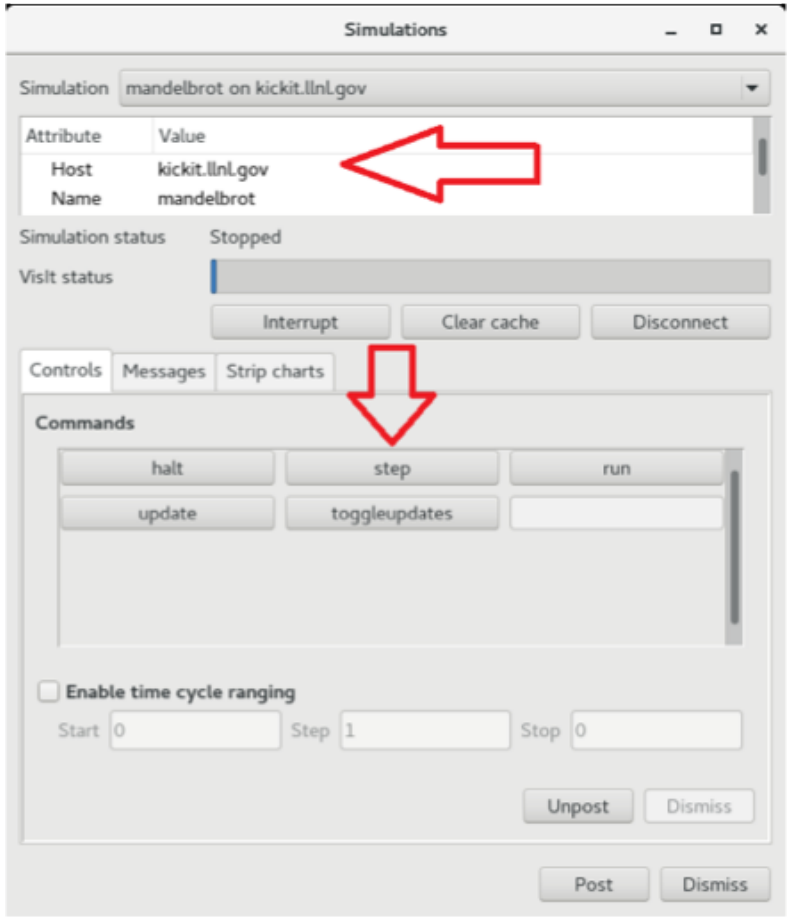
LibSim in action: Custom user interface

- The simulation can define a custom user interface
- Defined in a Qt ui file
  - Created with Qt Designer, a graphical GUI design tool
- Register additional callbacks in the simulation to handle



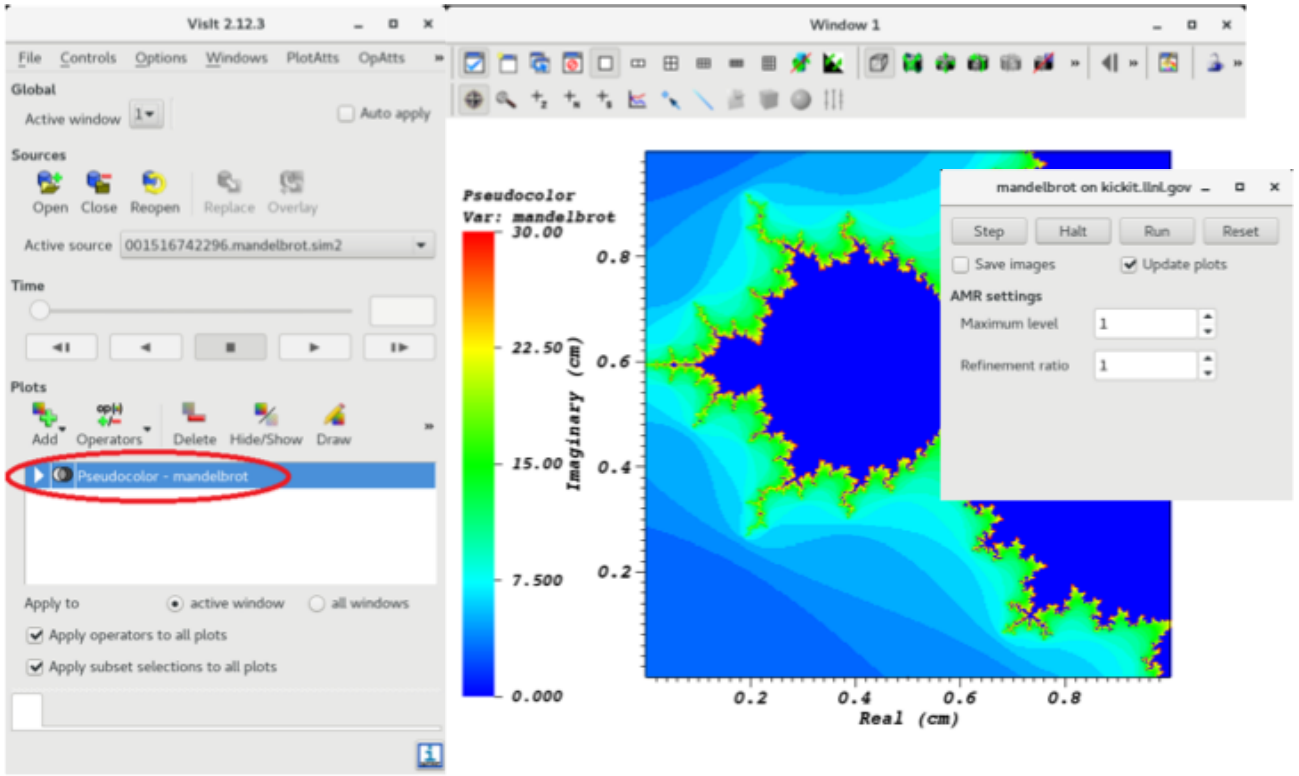
LibSim in action: Other capabilities

- Can add additional command buttons
- Can display messages from the simulation
- Can display strip charts of scalar values



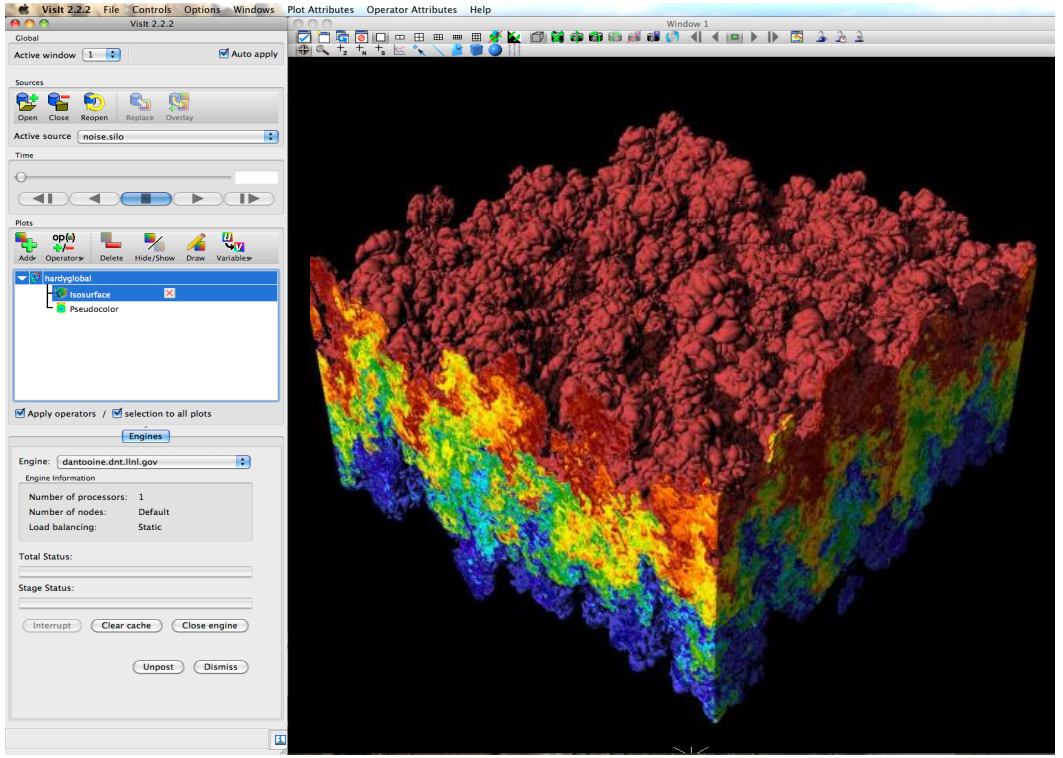
Agenda

- LibSim overview
- LibSim in action
- VisIt overview
- Integration details
- Sample integration with an AMR code



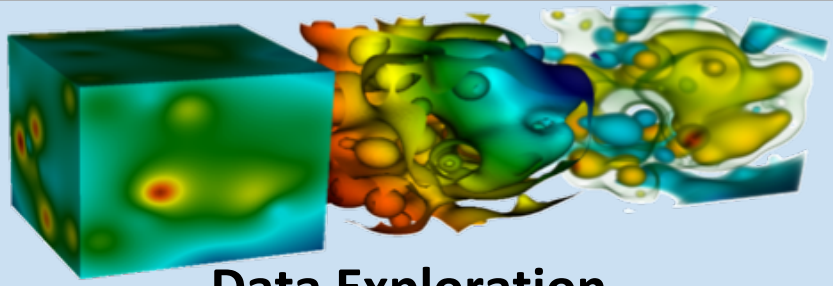
VisIt is an open source, turnkey application for data analysis and visualization of mesh-based data

- Production end-user tool supporting scientific and engineering applications.
- Provides an infrastructure for parallel post-processing that scales from desktops to massive HPC clusters.
- Source released under a BSD style license.

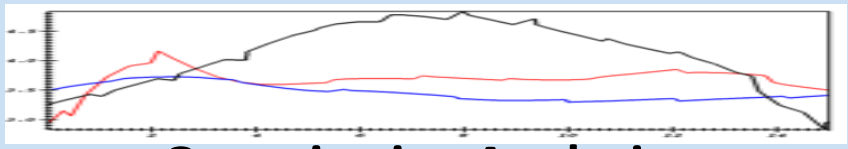


Pseudocolor plot of Density
(27 billion element dataset)

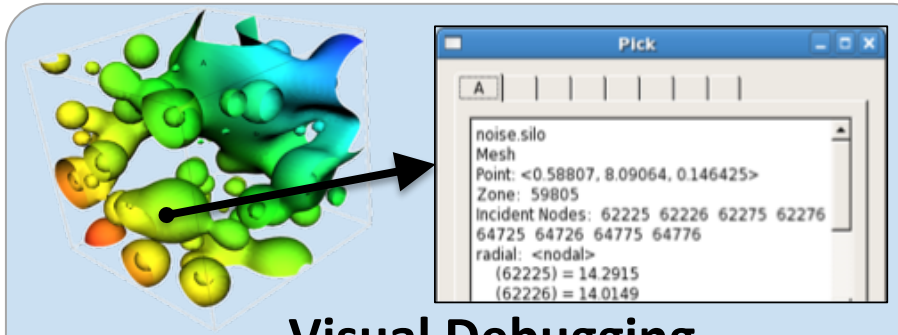
VisIt supports a wide range of use cases



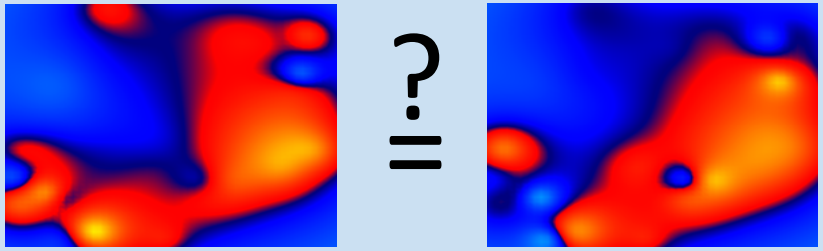
Data Exploration



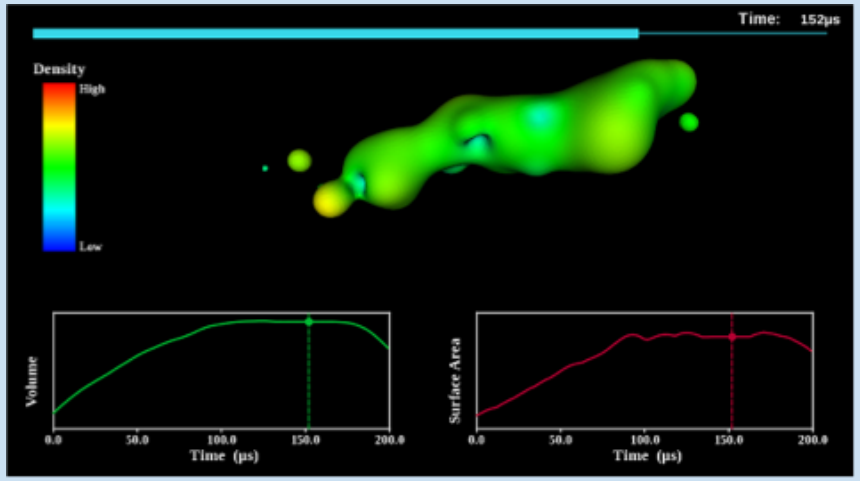
Quantitative Analysis



Visual Debugging

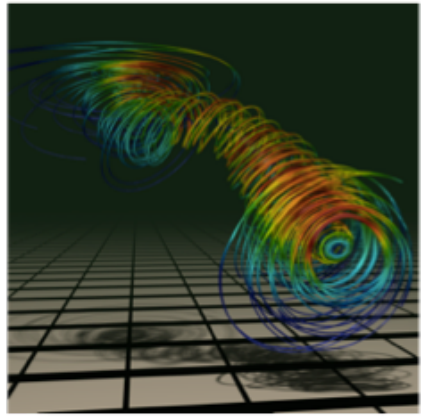


Comparative Analysis

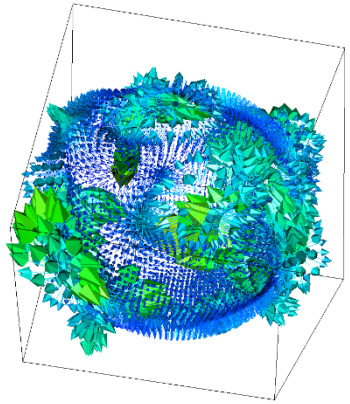


Presentation Graphics

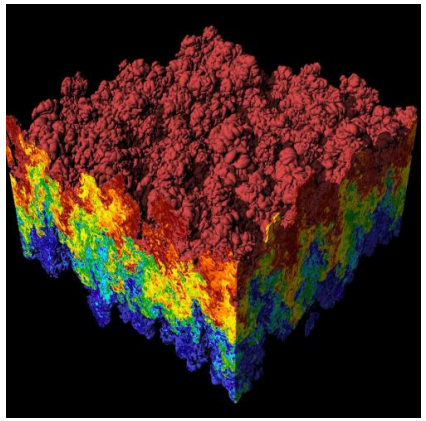
VisIt provides a wide range of plotting features for simulation data across many scientific domains



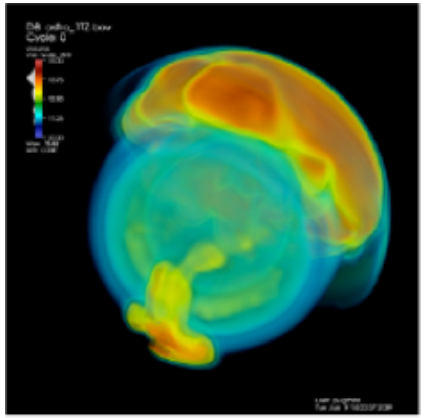
Streamlines / Pathlines



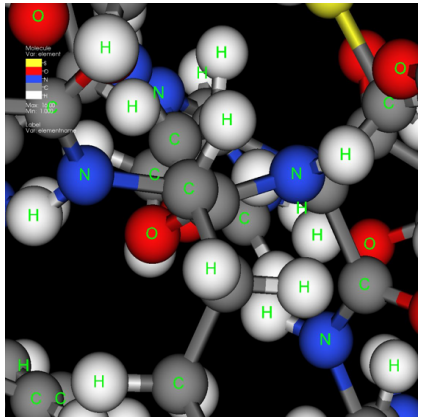
Vector / Tensor Glyphs



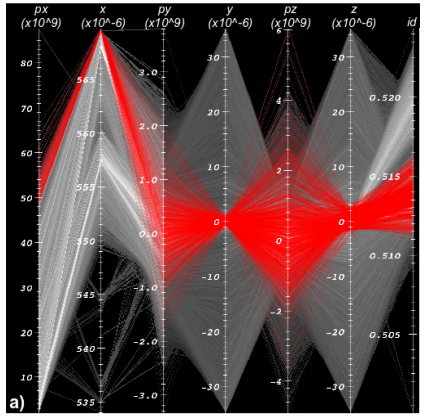
Pseudocolor Rendering



Volume Rendering



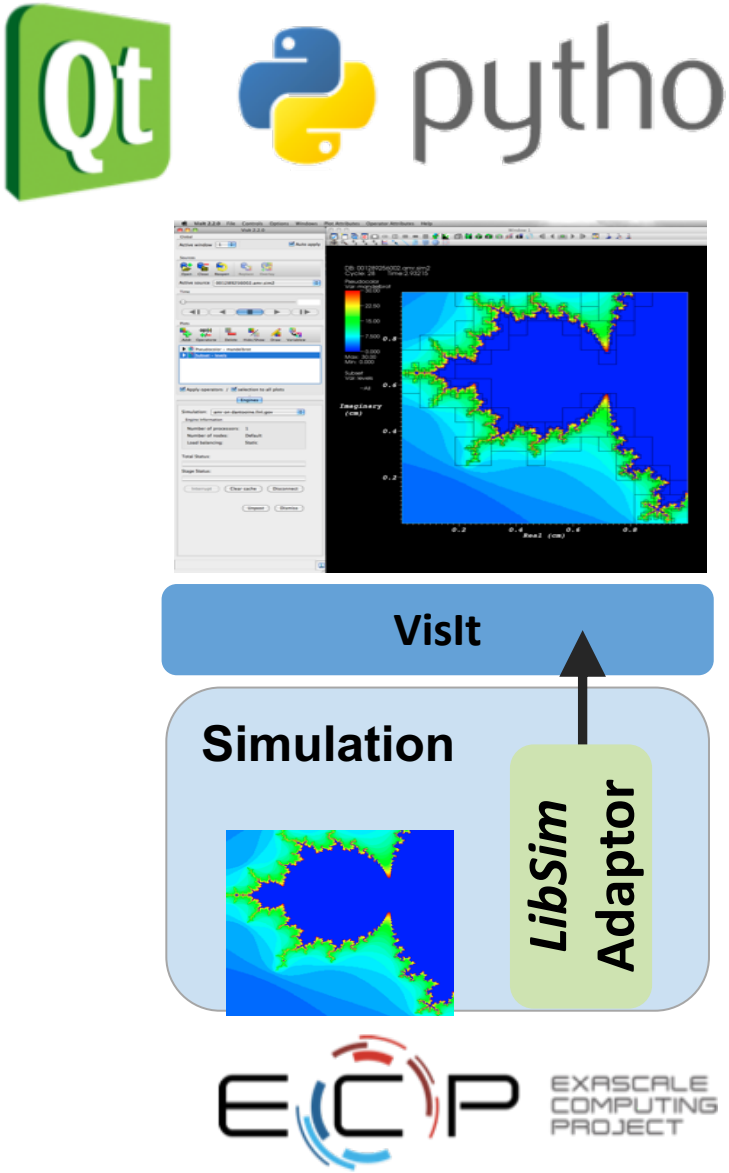
Molecular Visualization



Parallel Coordinates

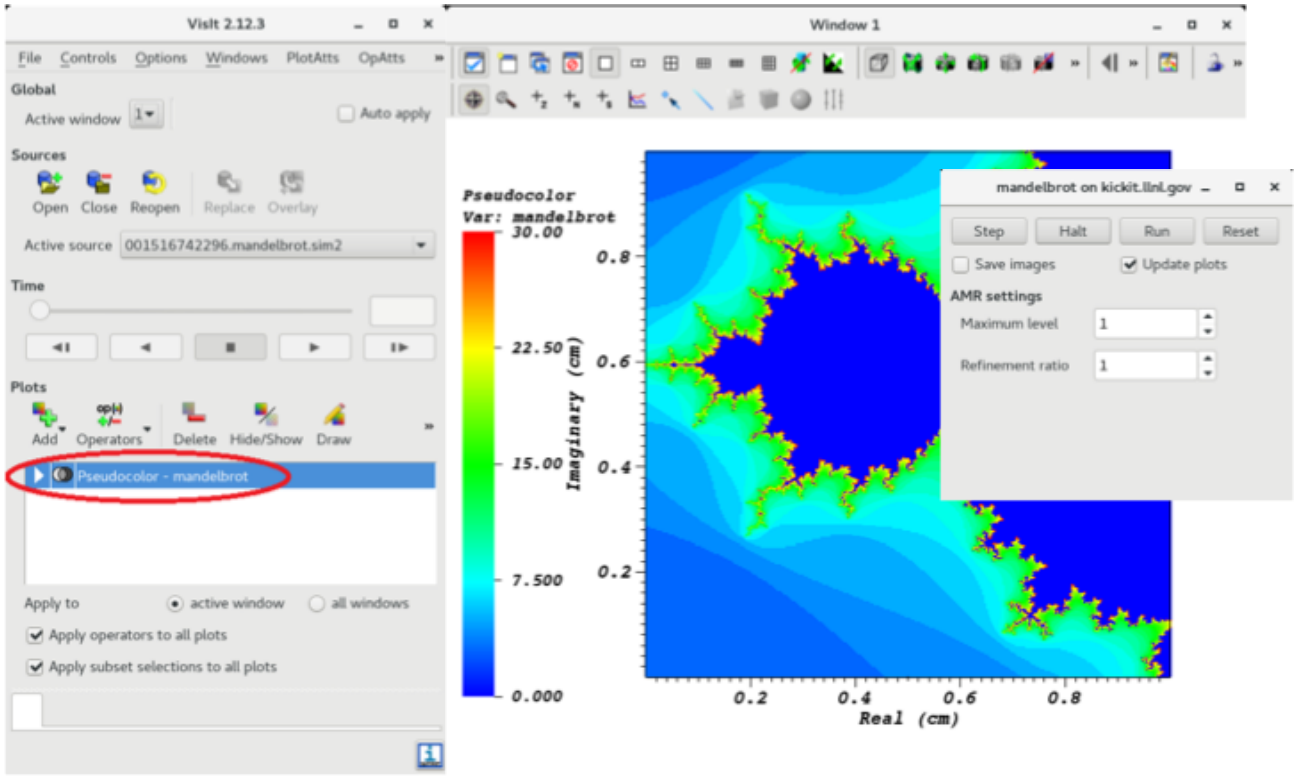
VisIt's infrastructure provides a flexible platform for custom workflows

- **C++ Plugin Architecture**
  - Custom File formats, Plots, Operators
  - Interface for custom GUIs in Python, C++ and Java
- **Python Interfaces**
  - Python scripting and batch processing
  - Data analysis via Python Expressions and Queries
- **In-Situ Coupling**
  - VisIt's *LibSim* library allows simulation codes to link in VisIt's engine for in situ visualization

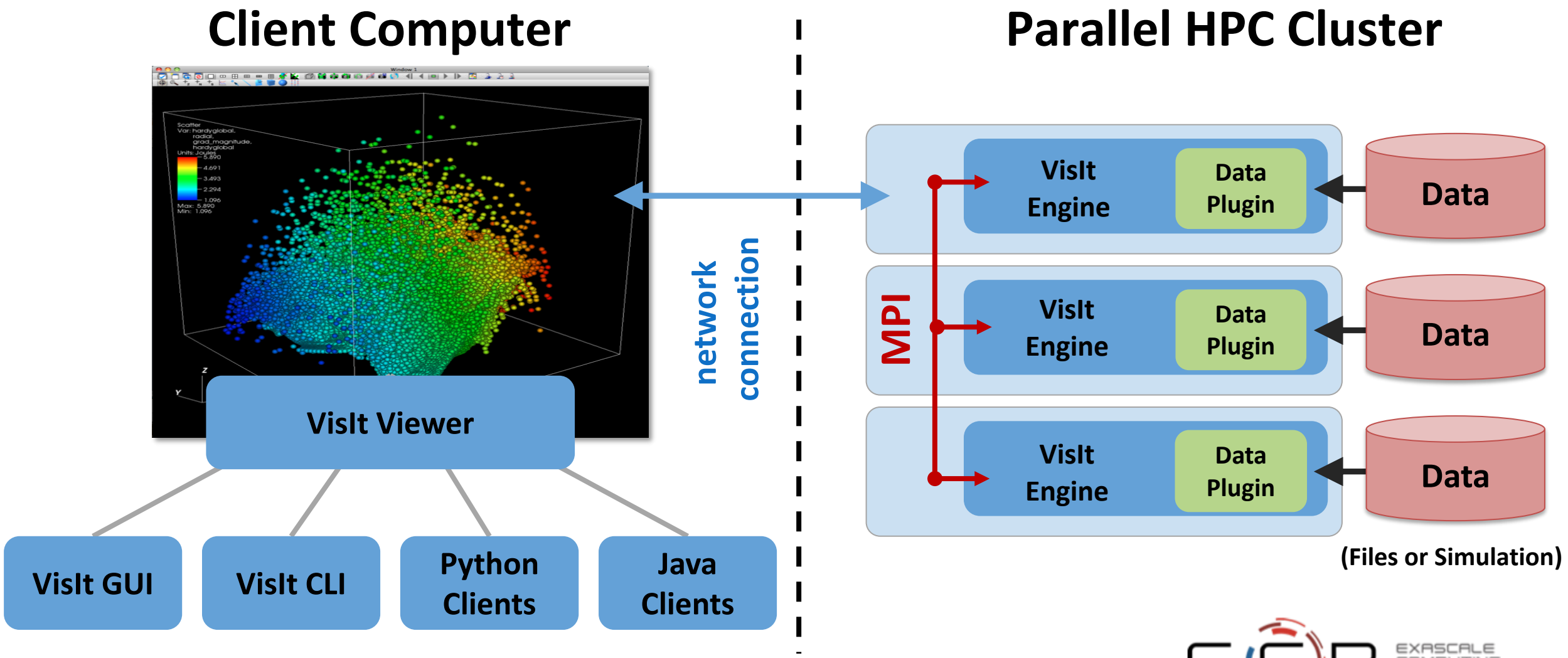


Agenda

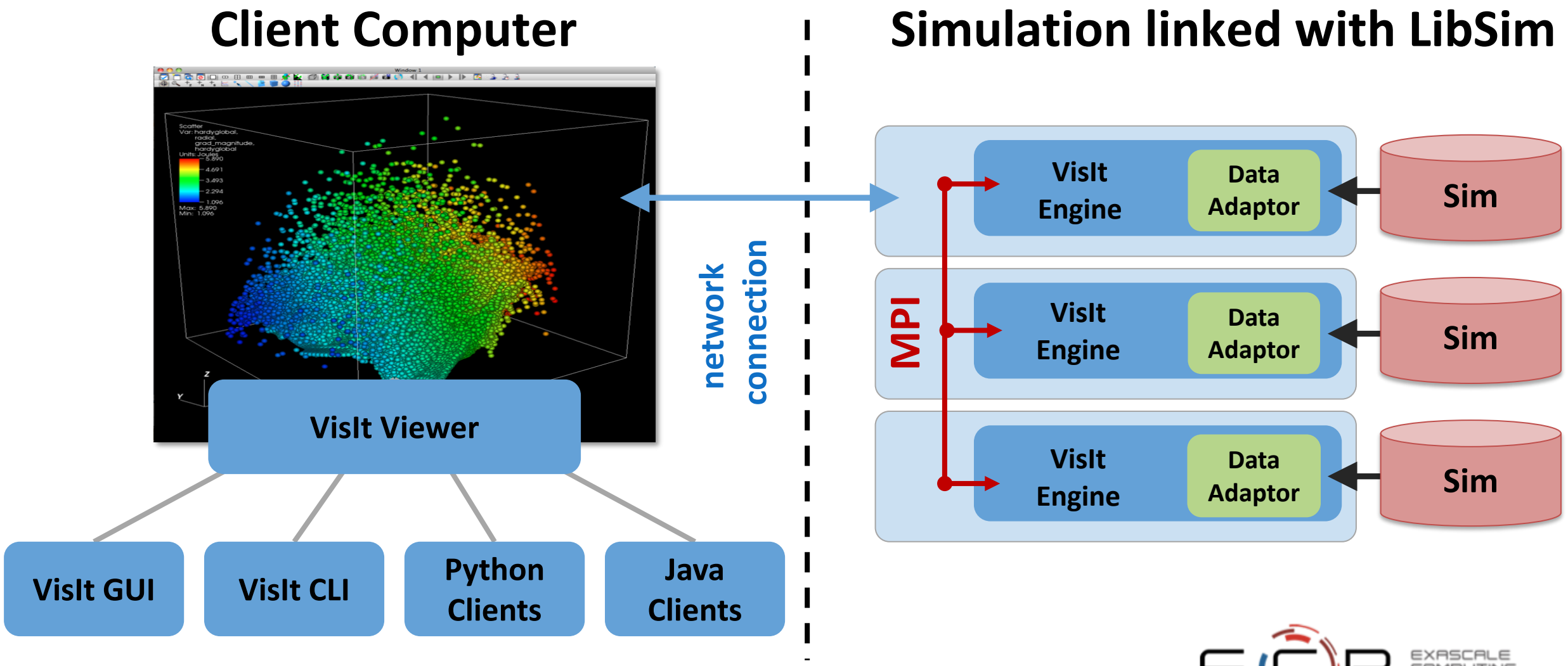
- LibSim overview
- LibSim in action
- VisIt overview
- Integration details
- Sample integration with an AMR code



VisIt employs a parallelized client-server architecture



LibSim leverages VisIt's client-server architecture

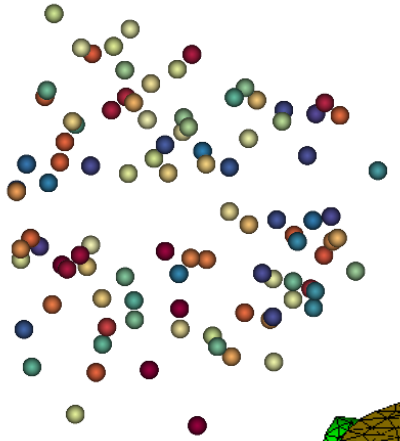


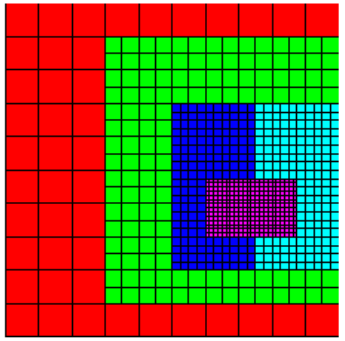
Implementation details

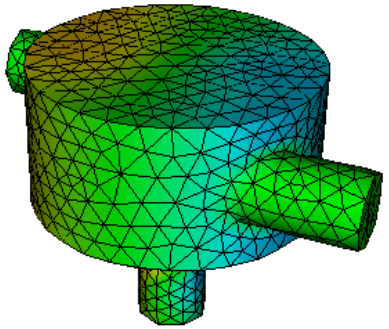
- LibSim is a small library that dynamically links the VisIt server code into your simulation when initialized
  - The overhead of linking in LibSim is minimal when not being used
- The VisIt server dynamically loads plot, operator and database plugins as needed
  - Reduces memory overhead to only need functionality
  - Support a statically linked server on systems where it is needed
- Languages supported:
  - C/C++
  - Fortran

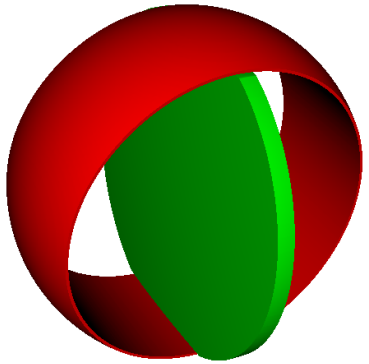
LibSim supports the full VisIt data model

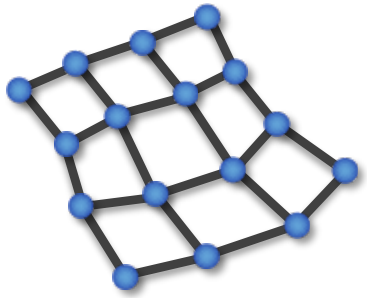
- Mesh types
  - Structured meshes
    - Rectilinear/Curvilinear
    - I-Blanking
  - Particle meshes
  - Constructive Solid Geometry (CSG) meshes
  - Adaptive Mesh Refinement (AMR) meshes
  - Unstructured & Polyhedral meshes
- Variables
  - 1 to N components
  - Zonal and Nodal
  - Enumerated type
- Materials & Species



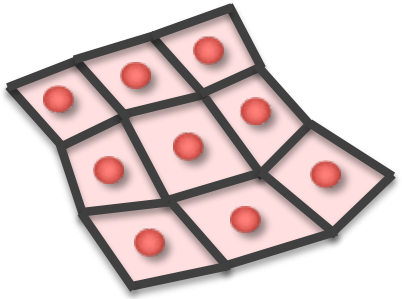








Nodal



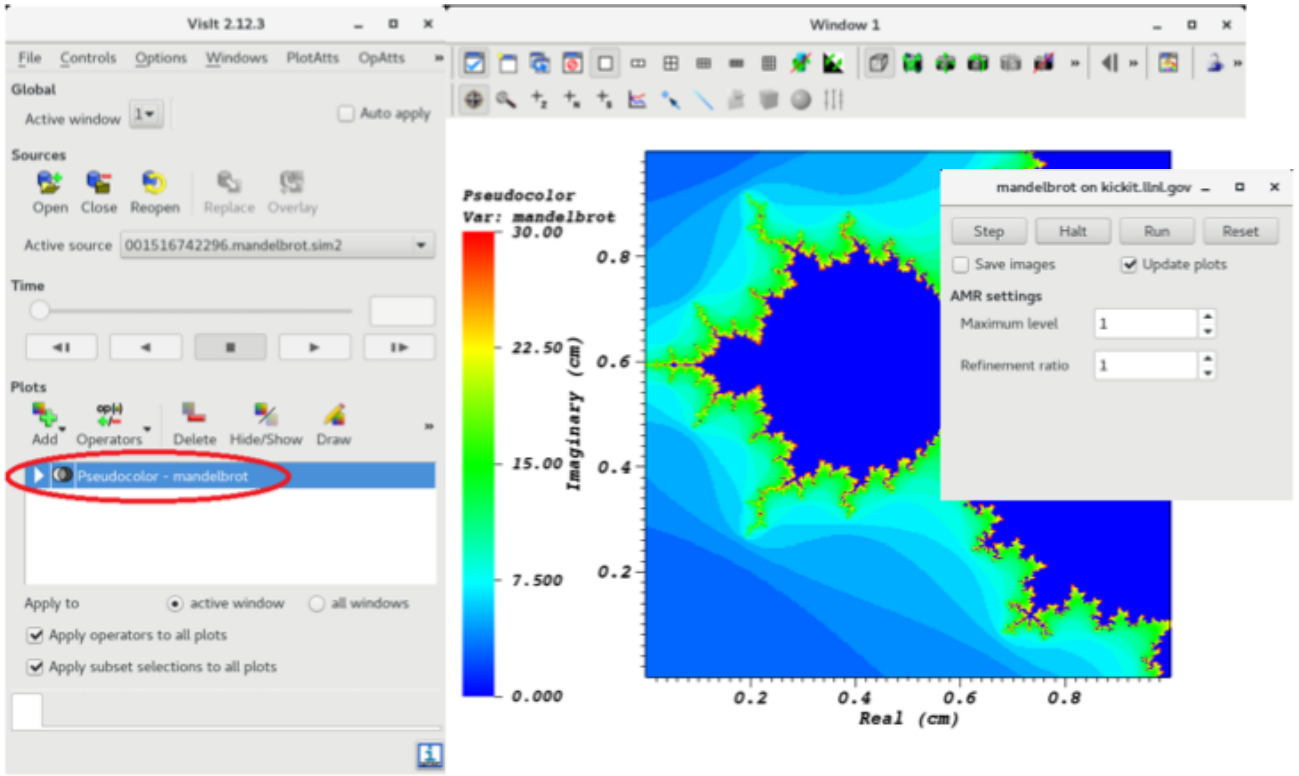
Zonal

Main Points of LibSim Integration

- Initialize LibSim
- Instrument the main physics loop
- Provide metadata and register callbacks
- Implement data callbacks

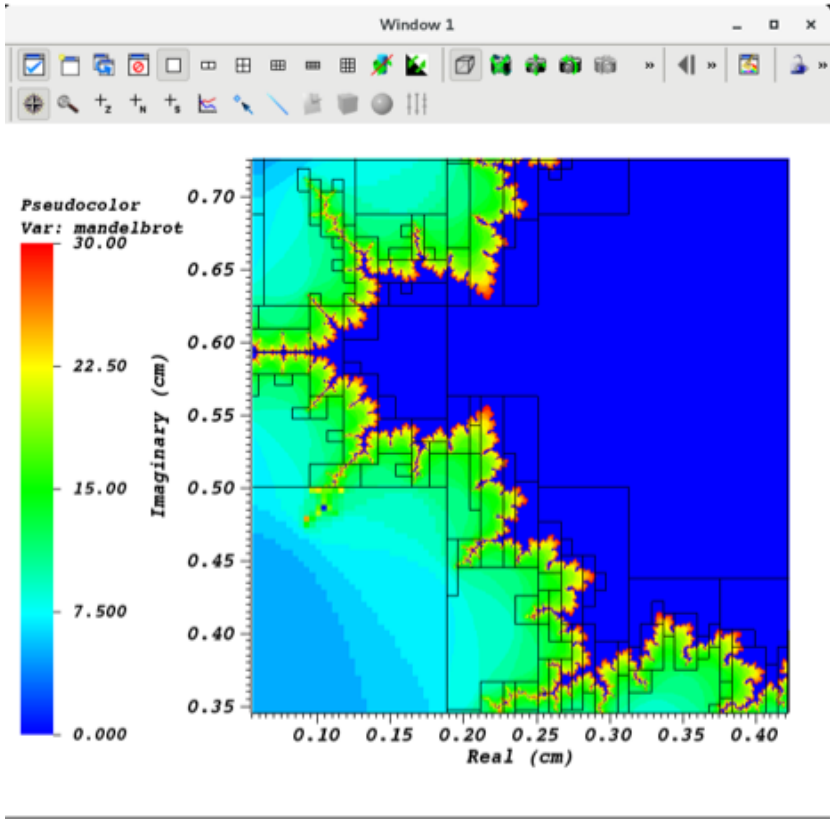
Agenda

- LibSim overview
- LibSim in action
- VisIt overview
- Integration details
- Sample integration with an AMR code



Code from an example LibSim integration

- Implements the Mandelbrot example shown earlier
  - Implemented on an AMR mesh
- Wiki page with more detailed instruction and links to source files
  - https://www.visitusers.org/index.php?title=VisIt-tutorial-in-situ



Initialization part 1

File Edit View Search Terminal Help

```
int main(int argc, char **argv)
{
    char *env = NULL;
    simulation_data sim;
    simulation_data_ctor(&sim);

#ifdef PARALLEL
    /* Initialize MPI */
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &sim.par_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &sim.par_size);
#else
    sim.par_rank = 0;
    sim.par_size = 1;
#endif

    /* Initialize environment variables. */
    SimulationArguments(argc, argv);
    VisItSetupEnvironment();

#ifdef PARALLEL
    /* Install callback functions for global communication. */
    VisItSetBroadcastIntFunction2(visit_broadcast_int_callback, (void*)&sim);
    VisItSetBroadcastStringFunction2(visit_broadcast_string_callback, (void*)&si
m);

    /* Tell libsim whether the simulation is parallel. */
    VisItSetParallel(sim.par_size > 1);
    VisItSetParallelRank(sim.par_rank);
#endif
                                                          922,1         71%
```

Initialize the LibSim environment

Parallel initialization

Initialization part 2

File Edit View Search Terminal Help

```
    /* Only read the environment on rank 0. This could happen before MPI_Init if
     * we are using an MPI that does not like to let us spawn processes but we
     * would not know our processor rank.
     */
    if(sim.par_rank == 0)
        env = VisItGetEnvironment();

    /* Pass the environment to all other processors collectively. */
    VisItSetupEnvironment2(env);
    if(env != NULL)
        free(env);

    /* Write out .sim2 file that VisIt uses to connect. */
    if(sim.par_rank == 0)
    {
        VisItInitializeSocketAndDumpSimFile(
#ifdef PARALLEL
            "mandelbrot_par",
#else
            "mandelbrot",
#endif
            "Demonstrates creating the Mandelbrot set on an AMR mesh",
            "/home/brugger1/simulation",
            NULL, "mandelbrot.ui", NULL);
    }

    /* Read input problem setup, geometry, data. */
    read_input_deck();

    /* Call the main loop. */
    mainloop(&sim);

    simulation_data_dtor(&sim);
#ifdef PARALLEL
    MPI_Finalize();
#endif

    return 0;
}
                                                    961,1         74%
```

Create the ".sim2" file which VisIt opens.

Main loop part 1

File Edit View Search Terminal Help

```
void mainloop(simulation_data *sim)
{
    int blocking, visitstate, err = 0;

    /* Register some ui actions */
    VisItUI_clicked("STEP", ui_step_clicked, &sim);
    VisItUI_clicked("HALT", ui_halt_clicked, &sim);
    VisItUI_clicked("RUN", ui_run_clicked, &sim);
    VisItUI_clicked("RESET", ui_reset_clicked, &sim);
    VisItUI_valueChanged("LEVELS", ui_levels_changed, &sim);
    VisItUI_valueChanged("REFINEMENTRATIO", ui_ratio_changed, &sim);
    VisItUI_stateChanged("SAVEIMAGES", ui_saveimages_changed, &sim);
    VisItUI_stateChanged("UPDATEPLOTS", ui_updateplots_changed, &sim);

    /* If we're not running by default then simulate once there's something
     * once VisIt connects.
     */
    if(sim->runMode == SIM_STOPPED)
        simulate_one_timestep(sim);
```

799,0-1 61%

Initialize the callbacks for the custom GUI elements. This could have been done earlier with the rest of the initialization.

Main loop part 2

File Edit View Search Terminal Help

```
    /* main loop */
    if(sim->par_rank == 0)
    {
        fprintf(stderr, "command> ");
        fflush(stderr);
    }
    do
    {
        blocking = (sim->runMode == SIM_STOPPED) ? 1 : 0;
        /* Get input from VisIt or timeout so the simulation can run. */
        if(sim->par_rank == 0)
        {
            visitstate = VisItDetectInput(blocking, fileno(stdin));
        }
#ifdef PARALLEL
        /* Broadcast the return value of VisItDetectInput to all procs. */
        MPI_Bcast(&visitstate, 1, MPI_INT, 0, MPI_COMM_WORLD);
#endif
        /* Do different things depending on the output from VisItDetectInput. */
        switch(visitstate)
        {
        case 0:
            /* There was no input from VisIt, return control to sim. */
            simulate_one_timestep(sim);
            break;
        case 1:
            /* VisIt is trying to connect to sim. */
            if(VisItAttemptToCompleteConnection() == VISIT_OKAY)
            {
                fprintf(stderr, "VisIt connected\n");
                initialize_callbacks(sim);
            }
            else
            {
                /* Print the error message */
                char *err = VisItGetLastError();
                fprintf(stderr, "VisIt did not connect: %s\n", err);
                free(err);
            }
            break;
                                                          800,13        64%
```

Check if there are any events from LibSim to process.

Process a VisIt connect event

Main loop part 3

File Edit View Search Terminal Help

```
        case 2:
            /* VisIt wants to tell the engine something. */
            if(!ProcessVisItCommand(sim))
            {
                /* Disconnect on an error or closed connection. */
                VisItDisconnect();
                /* Start running again if VisIt closes. */
                /*sim->runMode = SIM_RUNNING;*/
            }
            break;
        case 3:
            /* VisItDetectInput detected console input - do something with it.
             * NOTE: you can't get here unless you pass a file descriptor to
             * VisItDetectInput instead of -1.
             */
            ProcessConsoleCommand(sim);
            if (sim->par_rank == 0)
            {
                fprintf(stderr, "command> ");
                fflush(stderr);
            }
            break;
        default:
            fprintf(stderr, "Can't recover from error %d!\n", visitstate);
            err = 1;
            break;
        }
    } while(!sim->done && err == 0);
}
                                                          868,1         67%
```

Process a VisIt disconnect event

Process console events

Registering callbacks

File Edit View Search Terminal Help

```
            /* VisIt is trying to connect to sim. */
            if(VisItAttemptToCompleteConnection() == VISIT_OKAY)
            {
                fprintf(stderr, "VisIt connected\n");
                VisItSetCommandCallback(ControlCommandCallback, (void*)sim);
                VisItSetSlaveProcessCallback2(SlaveProcessCallback, (void*)sim);

                VisItSetGetMetaData(SimGetMetaData, (void*)sim);
                VisItSetGetMesh(SimGetMesh, (void*)sim);
                VisItSetGetVariable(SimGetVariable, (void*)sim);
                VisItSetGetDomainNesting(SimGetDomainNesting, (void*)sim);
#ifdef PARALLEL
                VisItSetGetDomainList(SimGetDomainList, (void*)sim);
#endif
            }
            else
            {
                /* Print the error message */
                char *err = VisItGetLastError();
                fprintf(stderr, "VisIt did not connect: %s\n", err);
                free(err);
            }
                                                          847,1         65%
```

Register the callbacks

SimGetMetaData: the mesh

File Edit View Search Terminal Help

```
        /* Fill in the AMR metadata. */
        if(VisIt_MeshMetaData_alloc(&mmd) == VISIT_OKAY)
        {
            /* Set the mesh's properties.*/
            VisIt_MeshMetaData_setName(mmd, "AMR_mesh");
            VisIt_MeshMetaData_setMeshType(mmd, VISIT_MESHTYPE_AMR);
            VisIt_MeshMetaData_setTopologicalDimension(mmd, 2);
            VisIt_MeshMetaData_setSpatialDimension(mmd, 2);

            int ndoms = patch_num_patches(&sim->patch);
            VisIt_MeshMetaData_setNumDomains(mmd, ndoms);
            VisIt_MeshMetaData_setDomainTitle(mmd, "patches");
            VisIt_MeshMetaData_setDomainPieceName(mmd, "patch");

            int nlevels = patch_num_levels(&sim->patch);
            VisIt_MeshMetaData_setNumGroups(mmd, nlevels);
            VisIt_MeshMetaData_setGroupTitle(mmd, "levels");
            VisIt_MeshMetaData_setGroupPieceName(mmd, "level");
            patch_t **patches = patch_flat_array(&sim->patch);
            int *pcount = (int *)malloc(nlevels * sizeof(int));
            memset(pcount, 0, nlevels * sizeof(int));
            for(int i = 0; i < ndoms; ++i)
            {
                char tmpName[100];
                sprintf(tmpName, "level%d,patch%04d", patches[i]->level, pcount[
patches[i]->level]++);
                VisIt_MeshMetaData_addDomainName(mmd, tmpName);
                VisIt_MeshMetaData_addGroupId(mmd, patches[i]->level);
            }
            FREE(pcount);
            FREE(patches);

            VisIt_MeshMetaData_setXUnits(mmd, "cm");
            VisIt_MeshMetaData_setYUnits(mmd, "cm");
            VisIt_MeshMetaData_setXLabel(mmd, "Real");
            VisIt_MeshMetaData_setYLabel(mmd, "Imaginary");

            VisIt_SimulationMetaData_addMesh(md, mmd);
        }
                                                          1021,1        79%
```

Define the AMR specific mesh metadata

Define the generic mesh metadata

SimGetMetaData: the variables & custom commands

File Edit View Search Terminal Help

```
        /* Add a variable. */
        if(VisIt_VariableMetaData_alloc(&vmd) == VISIT_OKAY)
        {
            VisIt_VariableMetaData_setName(vmd, "mandelbrot");
            VisIt_VariableMetaData_setMeshName(vmd, "AMR_mesh");
            VisIt_VariableMetaData_setType(vmd, VISIT_VARTYPE_SCALAR);
            VisIt_VariableMetaData_setCentering(vmd, VISIT_VARCENTERING_ZONE);

            VisIt_SimulationMetaData_addVariable(md, vmd);
        }

        /* Add some custom commands. */
        for(size_t i = 0; i < sizeof(cmd_names)/sizeof(const char *); ++i)
        {
            visit_handle cmd = VISIT_INVALID_HANDLE;
            if(VisIt_CommandMetaData_alloc(&cmd) == VISIT_OKAY)
            {
                VisIt_CommandMetaData_setName(cmd, cmd_names[i]);
                VisIt_SimulationMetaData_addGenericCommand(md, cmd);
            }
        }
                                                          1043,1        81%
```

Define the variable metadata

Define the custom commands that are on the main simulation control window.

SimGetMesh

File Edit View Search Terminal Help

```
        if(VisIt_RectilinearMesh_alloc(&h) != VISIT_ERROR)
        {
            /* Initialize X coords. */
            float *coordX = (float *)malloc(sizeof(float) * (patch->nx+1));
            float width  = sim->patch.window[1] - sim->patch.window[0];
            float x0 = (patch->window[0] - sim->patch.window[0]) / width;
            float x1 = (patch->window[1] - sim->patch.window[0]) / width;
            for(i = 0; i < (patch->nx+1); ++i)
            {
                float t = float(i) / float(patch->nx);
                coordX[i] = (1.f-t)*x0 + t*x1;
            }
            /* Initialize Y coords. */
            float *coordY = (float *)malloc(sizeof(float) * (patch->ny+1));
            float height = sim->patch.window[3] - sim->patch.window[2];
            float y0 = (patch->window[2] - sim->patch.window[2]) / height;
            float y1 = (patch->window[3] - sim->patch.window[2]) / height;
            for(i = 0; i < (patch->ny+1); ++i)
            {
                float t = float(i) / float(patch->ny);
                coordY[i] = (1.f-t)*y0 + t*y1;
            }

            /* Give the mesh some coordinates it can use. */
            visit_handle xc, yc;
            VisIt_VariableData_alloc(&xc);
            VisIt_VariableData_alloc(&yc);
            if(xc != VISIT_INVALID_HANDLE && yc != VISIT_INVALID_HANDLE)
            {
                VisIt_VariableData_setDataF(xc, VISIT_OWNER_VISIT, 1, patch->nx+
1, coordX);
                VisIt_VariableData_setDataF(yc, VISIT_OWNER_VISIT, 1, patch->ny+
1, coordY);
                VisIt_RectilinearMesh_setCoordsXY(h, xc, yc);
            }
            else
            {
                free(coordX);
                free(coordY);
            }
                                                          1119,1        86%
```

Return a rectilinear mesh for the requested patch.

SimGetVariable

File Edit View Search Terminal Help

```
visit_handle
SimGetVariable(int domain, const char *name, void *cbdata)
{
    visit_handle h = VISIT_INVALID_HANDLE;

    /* Get the patch with the appropriate domain id. */
    simulation_data *sim = (simulation_data *)cbdata;
    patch_t *patch = patch_get_patch(&sim->patch, domain);

    if(strcmp(name, "mandelbrot") == 0 && patch != NULL)
    {
        VisIt_VariableData_alloc(&h);
        VisIt_VariableData_setDataC(h, VISIT_OWNER_SIM, 1,
            patch->nx * patch->ny, (char *)patch->data);
    }

    return h;
}
                                                    1232,1        97%
```

Return the variable for the specified variable and patch

Batch operation specifics

- Batch initialization
  - Batch initialization requires the VisIt runtime library to be explicitly loaded
  - Once the runtime is loaded, register data callbacks
  - Call functions to set up visualization

```
VisItInitializeRuntime();

VisItSetGetMetaData(SimGetMetaData, NULL);
VisitSetGetMesh(SimGetMesh, NULL)

VisItRestoreSessionFile("/path/to/setup.session")
```

Batch operation specifics

- Saving results
  - The simulation tells VisIt the timestep changed
  - The simulation requests that all plots be updated
  - The simulation can save images or export data

```
VisItTimeStepChanged();
VisItUpdatePlots();

char fn[100];
static int count = 0;
sprintf(fn, "image%04d.png", count);
VisItSaveWindow(fn, 800, 800,
                    VISIT_IMAGEFORMAT_JPEG);

Visit_handle vars = VISIT_INVALID_HANDLE;
VisIt_NameList_alloc(&vars);
VisIt_NameList_addName(vars, "default");
sprintf(fn, "export%04d", count++);
VisItExportDatabase(fn, "VTK_1.0", vars);
VisIt_NameList_free(vars);
```

Build Process

- Include a header:
  - C/C++: VisItControlInterface_V2.h, VisItDataInterface_V2.h
  - Fortran: visitfortransimV2interface.inc
- Add a library to simulation code link line:
  - C/C++: libsimV2.a
  - Fortran: libsimV2f.a

Makefile

```
File  Edit  View  Search  Terminal  Help
########### Edit this section ############
# VisIt information
VISITHOME=/usr/gapps/visit
VISITVERSION=2.12.3
VISITARCH=linux-x86_64

# Compiler settings
CXX=g++
CPPFLAGS=
CXXFLAGS=-O3
LDFLAGS=
LIBS=

########### This section should not be edited ###########
SIMDIR=$(VISITHOME)/$(VISITVERSION)/$(VISITARCH)/libsim/V2

SIM_CXXFLAGS=-I$(SIMDIR)/include
SIM_LDFLAGS=-L$(SIMDIR)/lib
SIM_LIBS=-lsimV2 -ldl

SRC=mandelbrot.C patch.C
OBJ=$(SRC:.C=.o)

all: mandelbrot

clean:
        rm -f mandelbrot $(OBJ)

mandelbrot: $(OBJ)
        $(CXX) -o mandelbrot $(OBJ) $(LDFLAGS) $(SIM_LDFLAGS) $(SIM_LIBS) $(LIBS
)

.C.o:
        $(CXX) $(CXXFLAGS) $(SIM_CXXFLAGS) $(CPPFLAGS) -c $<
                                                      33,1-8        All
```

VisIt specific information

Compiler specific information

Questions?

VisIt & LibSim User Resources:

- Main website: http://www.llnl.gov/visit
- Wiki: http://www.visitusers.org
- Email List: visitusers@ornl.gov