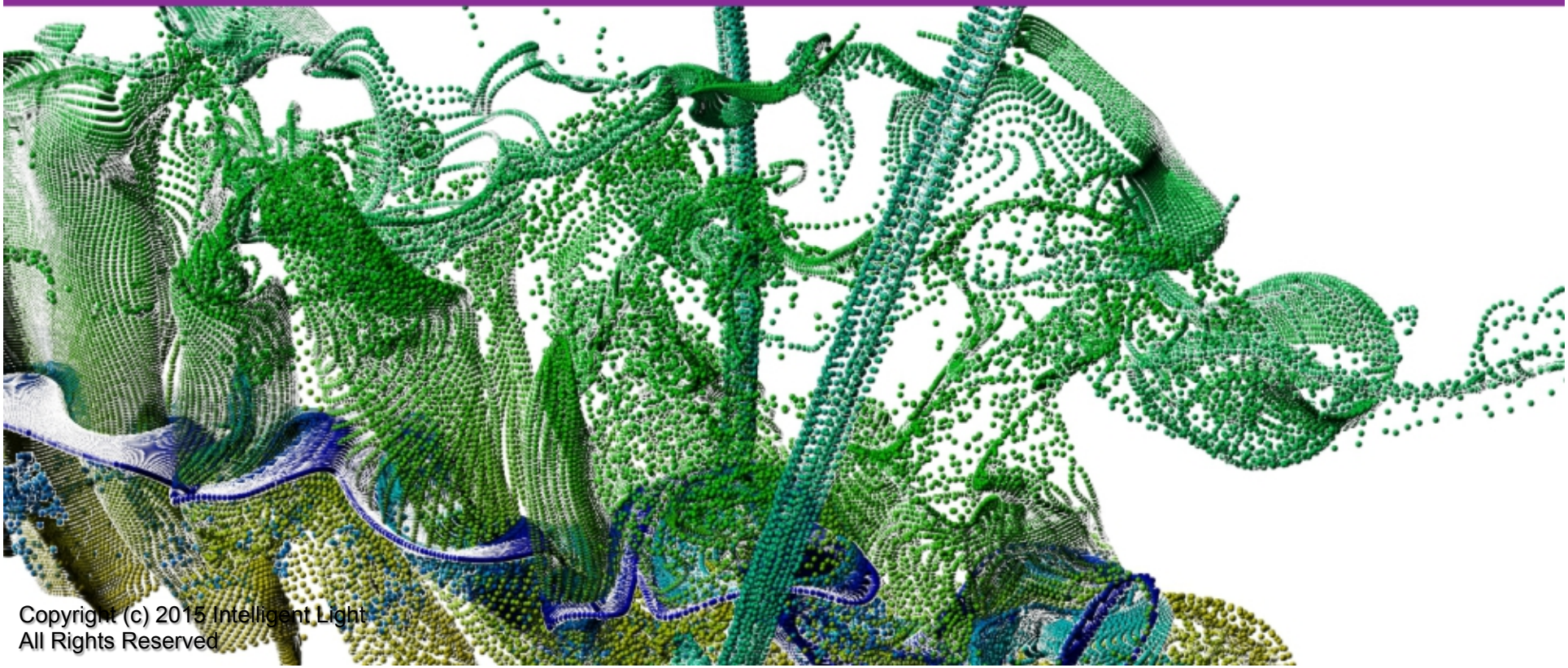
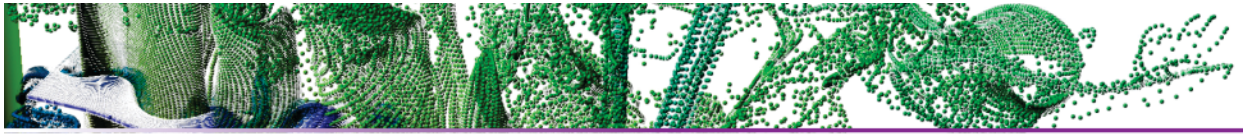
Applied Research Group

Seeking Answers, Deploying Solutions

Intelligent Light



Copyright (c) 2015 Intelligent Light
All Rights Reserved



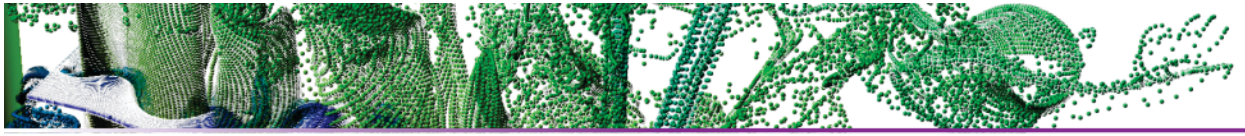
Brad Whitlock

Intelligent light

IN SITU POST-PROCESSING WITH VISIT, LIBSIM AND FIELDVIEW

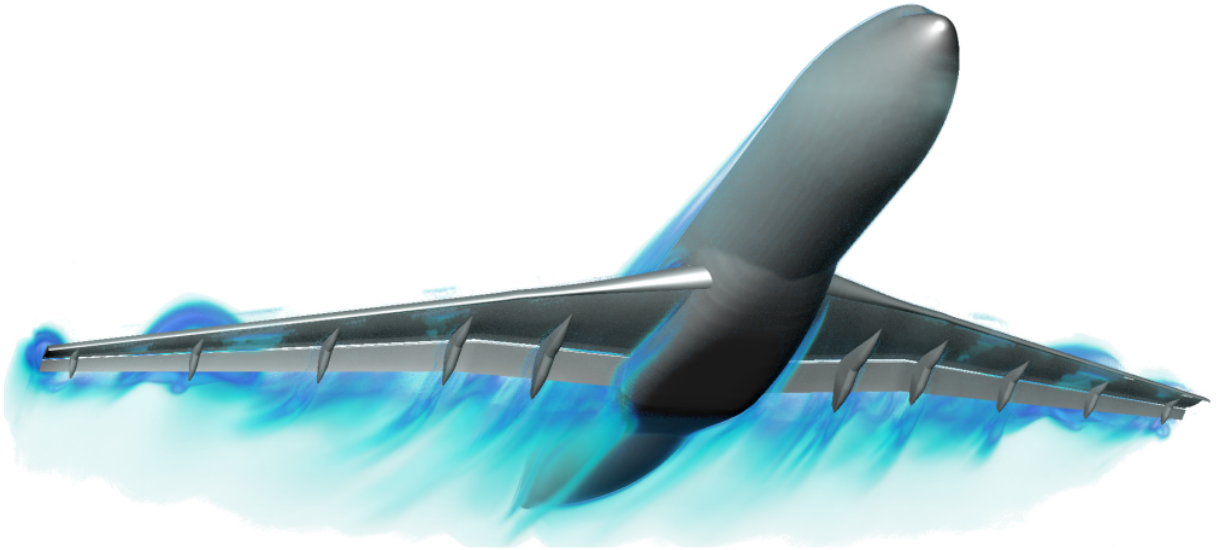
Applied Modeling & Simulation (AMS) Seminar Series
NASA Ames Research Center, January 16, 2015

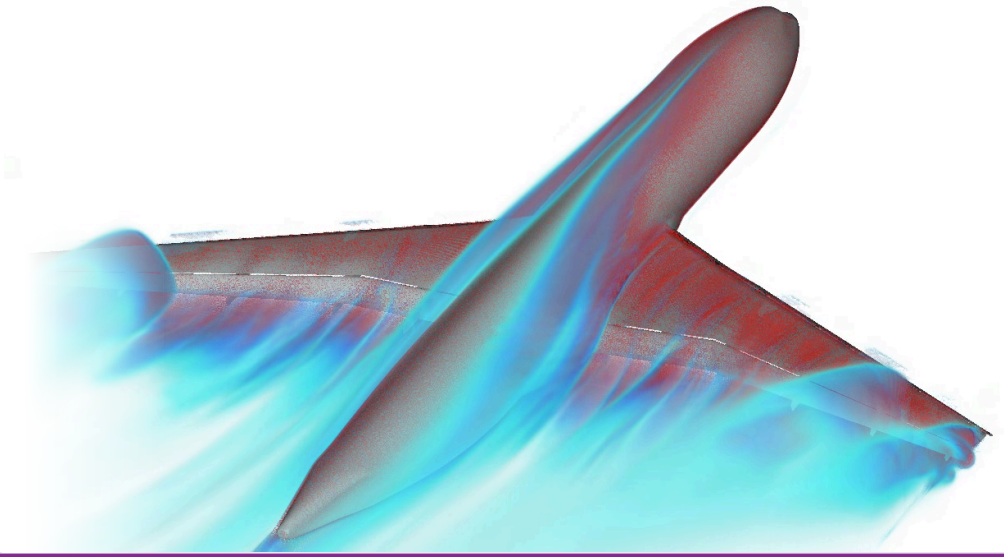
Copyright (c) 2015 Intelligent Light
All Rights Reserved



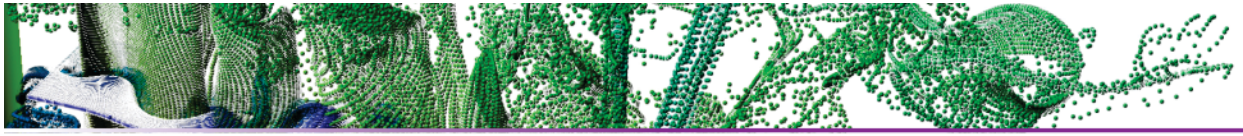
Overview

- Intelligent Light Overview
- In situ
- VisIt/Libsim and FieldView
- Workflow success story
- Instrumenting a simulation using Libsim





Copyright (c) 2015 Intelligent Light
All Rights Reserved



Intelligent Light

- Established in 1984
  - Nearly three decades in the software & services business
  - **FieldView** launched in 1990
  - Global Customer Base
  - Truly Solver Independent
  - Multiple CFD practitioners on staff
- We bridge CFD & IT for customers

IT

Microsoft, hp invent, AMD, NVIDIA, CRAY THE SUPERCOMPUTER COMPANY, sgi, DELL, intel

CFD Solvers

ANSYS, CD-adapco, CORVID TECHNOLOGIES, newmerical TECHNOLOGIES INT., CRADLE Software Cradle Co., Ltd., Metacomp Technologies, Altair, POINTWISE®

Aerospace & Defense

LOCKHEED MARTIN, BOEING, GE Aviation, NASA, ONERA, JAXA, DoD HPC, BETTIS, DLR, Sikorsky A United Technologies Company, Rolls-Royce, eurocopter an EADS Company, BAE SYSTEMS, Bell Helicopter, EADS, AIRBUS

Automotive & Transport

TOYOTA, HONDA, VOLVO, MITSUBISHI MOTORS, Visteon, JOE GIBBS RACING, LOLA, Red Bull racing, NISSAN, JR-EAST, BEHR, DENSO, Cummins, KORAIL KOREA RAIL

Consumer & Heavy Industry

Kawasaki, MITSUBISHI HEAVY INDUSTRIES LTD., P&G, HITACHI, Whirlpool, HYUNDAI, Canon, ALCOA, LEXMARK

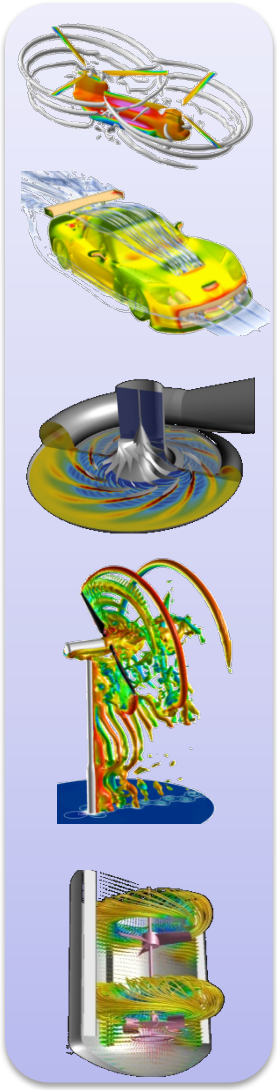
Energy & Environmental

Vestas, LM LM Glasfiber, SIEMENS, ALSTOM, NALCO Mobotec, Chevron

Copyright (c) 2015 Intelligent Light
All Rights Reserved

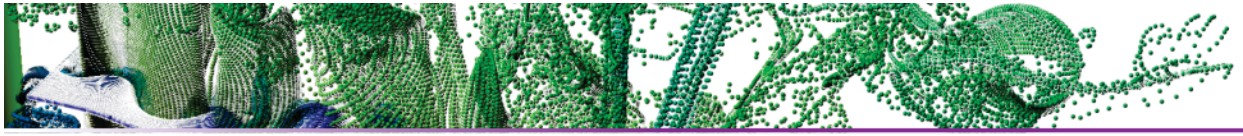
Serving the CFD Community

- FieldView Software Products
  - Comprehensive scope, from laptops to HPC
    - Visualization, numerical analysis
    - Data management & Automation
- CFD Consulting Services Team
  - Training (on-site or at Intelligent Light)
  - Script development, FieldView customization
  - Workflow Automation & Optimization
- Applied Research Group (ARG)
  - R&D in advanced post-processing & CFD methods
  - Feeds technology into FieldView



Our Mission: ***To help our customers using CFD to do more with less and make better decisions***

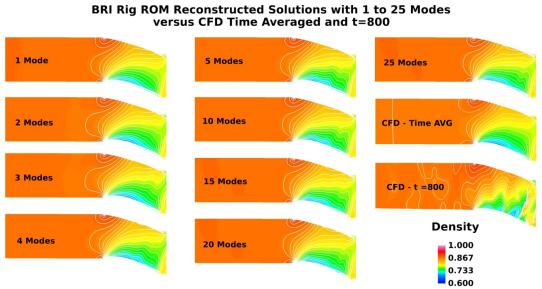
Copyright (c) 2015 Intelligent Light
All Rights Reserved



Applied Research Group

Strategic R&D for CFD, post-processing & visualization

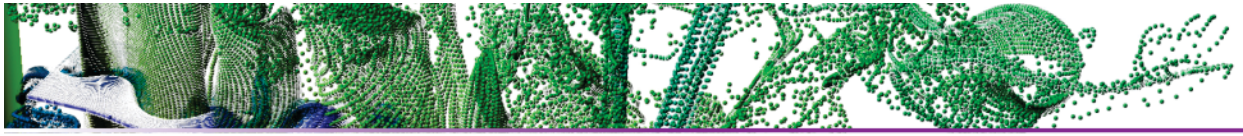
- Led by Dr. Earl P.N. Duque
- Air Force Research Lab – **EPISODE**
  - Large Scale Extracts
  - POD
  - Reduced Order Models
- Department of Energy – **FieldView-VisIt**
  - Open Source Post-Processing Software
  - Commercialized for High End Users





3D Fractal Isosurface from VisIt on BlueGene/Q

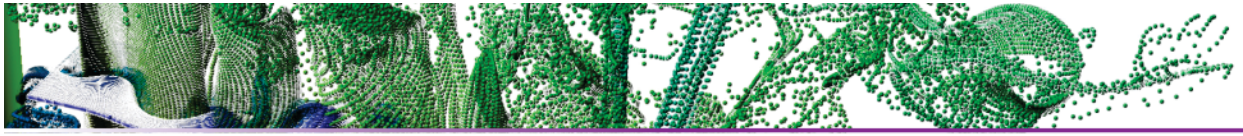
Copyright (c) 2015 Intelligent Light
All Rights Reserved



Active Deployment / Continuous Development

- DOE has chosen Intelligent Light to commercialize VisIt for engineering use – SBIR Phase 2
  - Binary FieldView XDB libs with *VisIt* Open source
  - Open-source *VisIt* engine, with FieldView GUI in client
  - Supported Software
- Libsim already coupled to
  - CREATE/AV – Kestrel
  - FUN3D
- Other significant efforts
  - DOE-OASCR: Scalable Analysis Methods and In Situ Infrastructure for Extreme Scale Knowledge Discovery
    - Infrastructure for ExaScale
    - Team with Lawrence Berkeley National Lab (Lead), Argonne, Georgia Tech & Kitware
  - Lawrence Livermore National Laboratory: Blue Gene Q Port

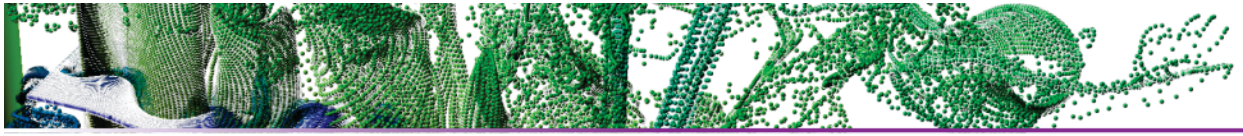
Copyright (c) 2015 Intelligent Light
All Rights Reserved



In Situ

- In situ processing couples data analysis and visualization with the simulation's execution so both are done in tandem
- There are different forms:
  - In transit – operates on the data when it is staged to another compute resource
  - Tightly coupled – operates on the data in the same address space as the simulation

Copyright (c) 2015 Intelligent Light
All Rights Reserved



Case For Using In Situ

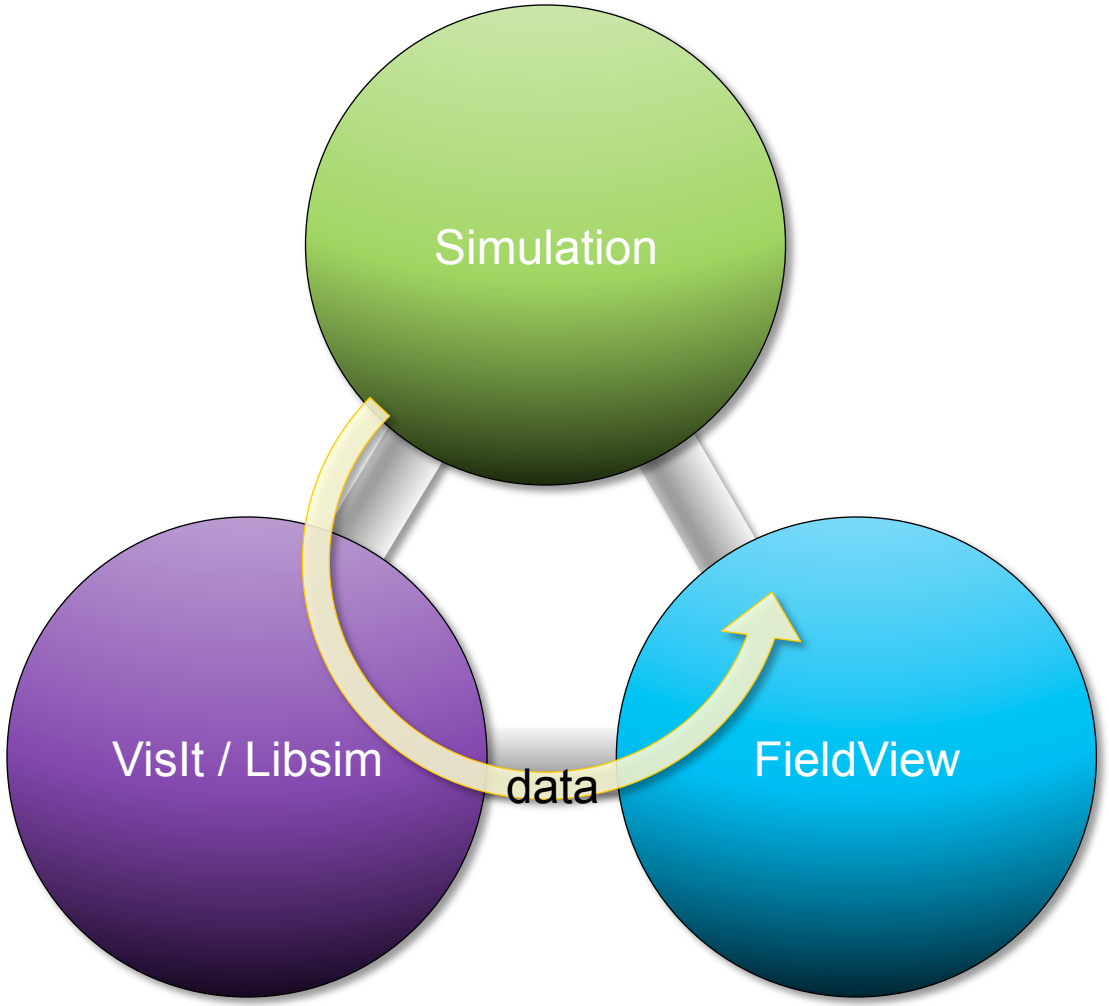
- I/O in supercomputers has not kept pace with compute power
  - Some applications report 90% of time spent in I/O *[Peterka et al.]*
  - Post processing simulation files requires write then read, paying for I/O twice in different application
- In situ reduces I/O costs and makes it feasible to save analyzed results (much smaller) at higher temporal frequency
- In situ provides an opportunity to analyze more of the data rather than just the portion that is typically saved

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	0.001%	3200 sec

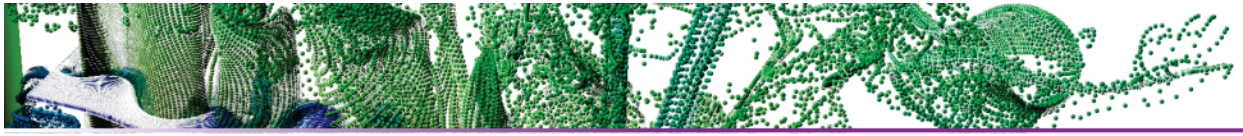
Copyright (c) 2015 Intelligent Light
All Rights Reserved

Software to Enable In Situ Workflow

- The simulation is the host program that controls execution and generates data
- VisIt/Libsim forms the in situ infrastructure and its supporting runtime library which creates data products such as XDB extracts for post hoc analysis
- FieldView is the postprocessor and ultimate consumer for data products created in situ



Copyright (c) 2015 Intelligent Light
All Rights Reserved



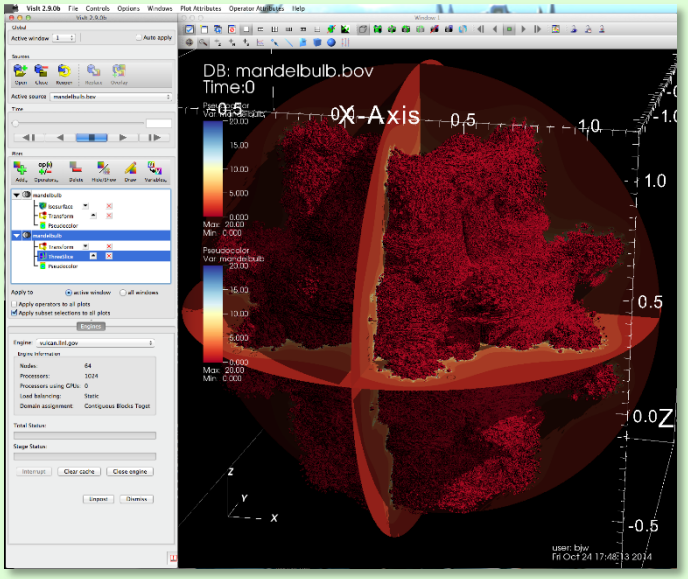
VisIt

- VisIt is open source software for visualizing and analyzing petascale simulation datasets

- Started Summer 2000
- R&D 100 award winner 2005
- Used worldwide
- Target use cases:
  - Quantitative Analysis
  - Comparative Analysis
  - Data Exploration
  - Visual Debugging
  - Presentation Graphics
- Intelligent Light's VisIt work partially supported by DOE Grant SC0007548.

VisIt is made for large problems

8 trillion cells | **98K cores**

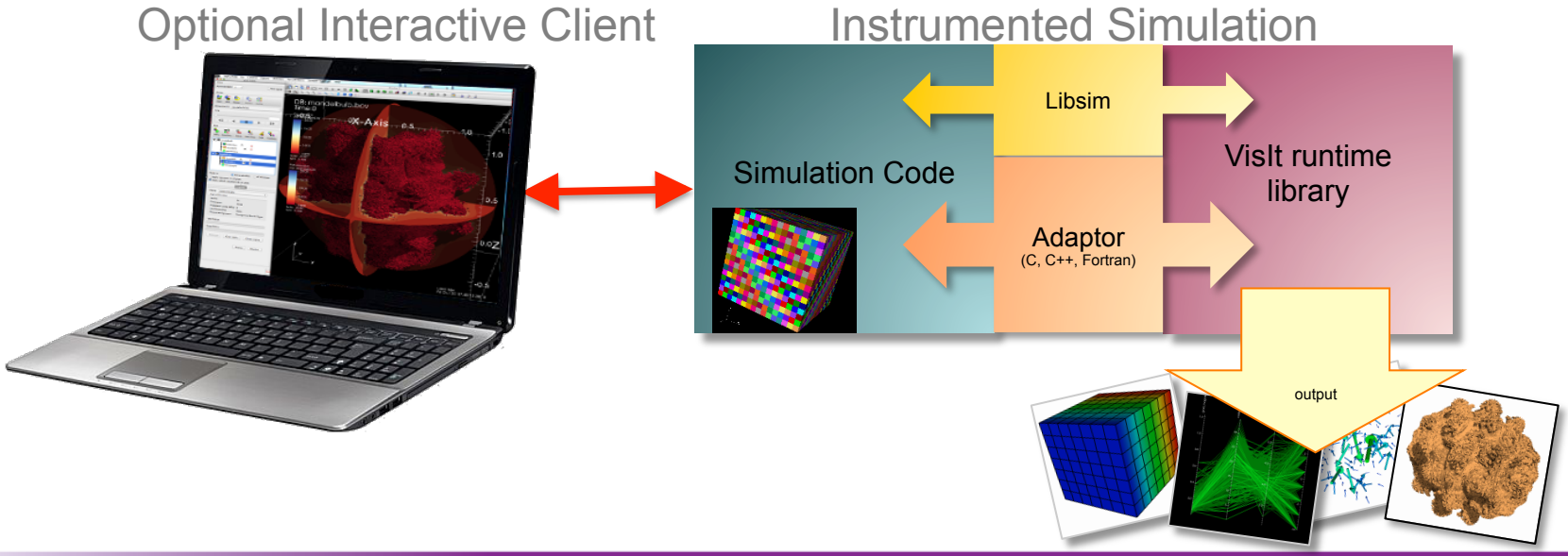


3D Fractal Dataset on LLNL Vulcan BlueGene/Q (98 billion cells)

Copyright (c) 2015 Intelligent Light
All Rights Reserved

Libsim

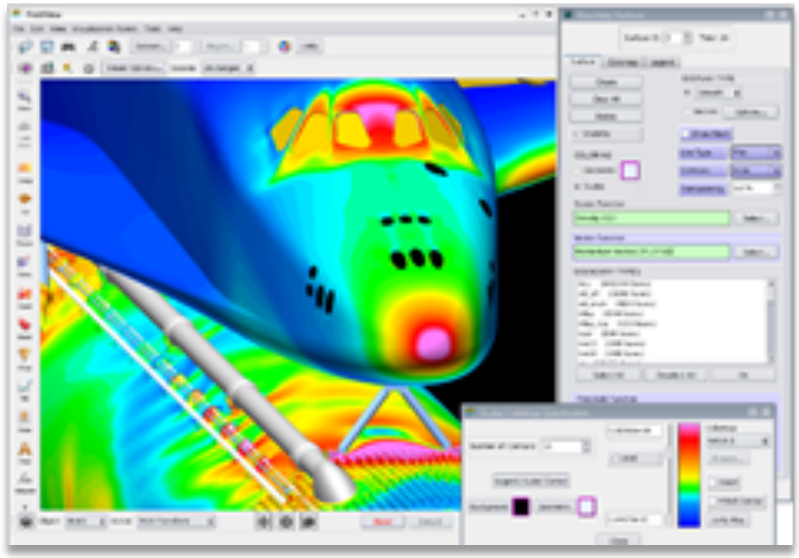
- Libsim enables simulations perform data analysis and visualization in situ by applying VisIt algorithms to data from the simulation.
- Libsim supports both interactive connections to running simulations as well as batch-only modes of operation.
- Codes: Ale3D, Mercury, Kull, Nek5000, Kestrel, FUN3D, …

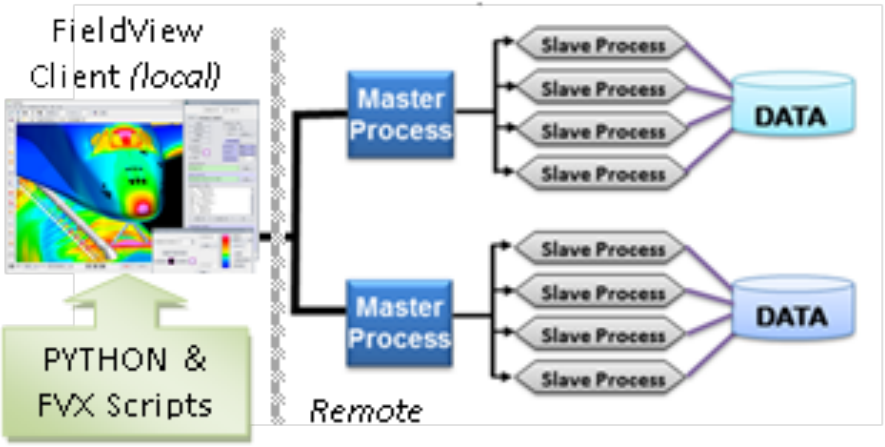


Copyright (c) 2015 Intelligent Light
All Rights Reserved

FieldView

- FieldView been on the market for CFD post-processing since 1991
- Graphic User Interface design created by close collaboration with Industry engineers and analysts
- Over 3000 licenses of FieldView in use today throughout the world
- Industries ranging from aerospace and automotive to nuclear engineering, turbomachinery, wind energy and food processing.
- Scales well to 256 nodes per dataset, 30 Bn cell unsteady as high water mark

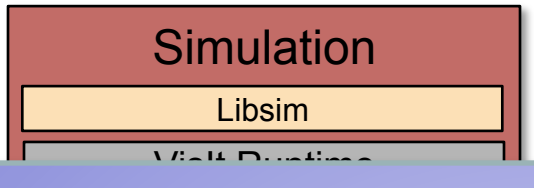




Copyright (c) 2015 Intelligent Light
All Rights Reserved

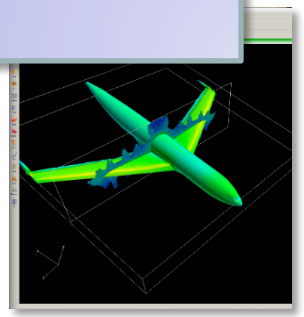
Put it all together: In Situ Extracts

- VisIt's export mechanism lets it



Extracts overcome in situ's greatest weakness – that you need to have some idea of what you want to see in the end. Extracts are small enough to save frequently and permit interactive exploration using traditional post-processing methods.

enabling frequent in situ extract dumps



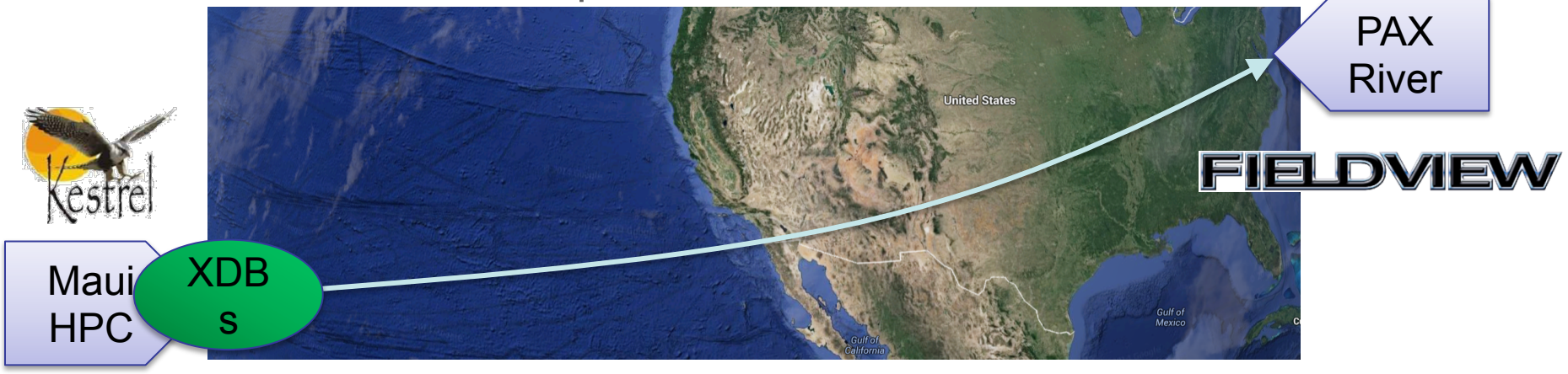
FieldView

Copyright (c) 2015 Intelligent Light
All Rights Reserved

A successful result

- Many timesteps of unsteady data are produced at Maui, Hawaii HPC center
- Post-processing via FieldView (interactive and movie making) done at Pax River, Maryland
- *"The processing time of the XDB extracts are a very small fraction… I can run through 100 frames of data in minutes using the extracts compared to hours using the full solution."*
- About 15% of runtime spent on in situ

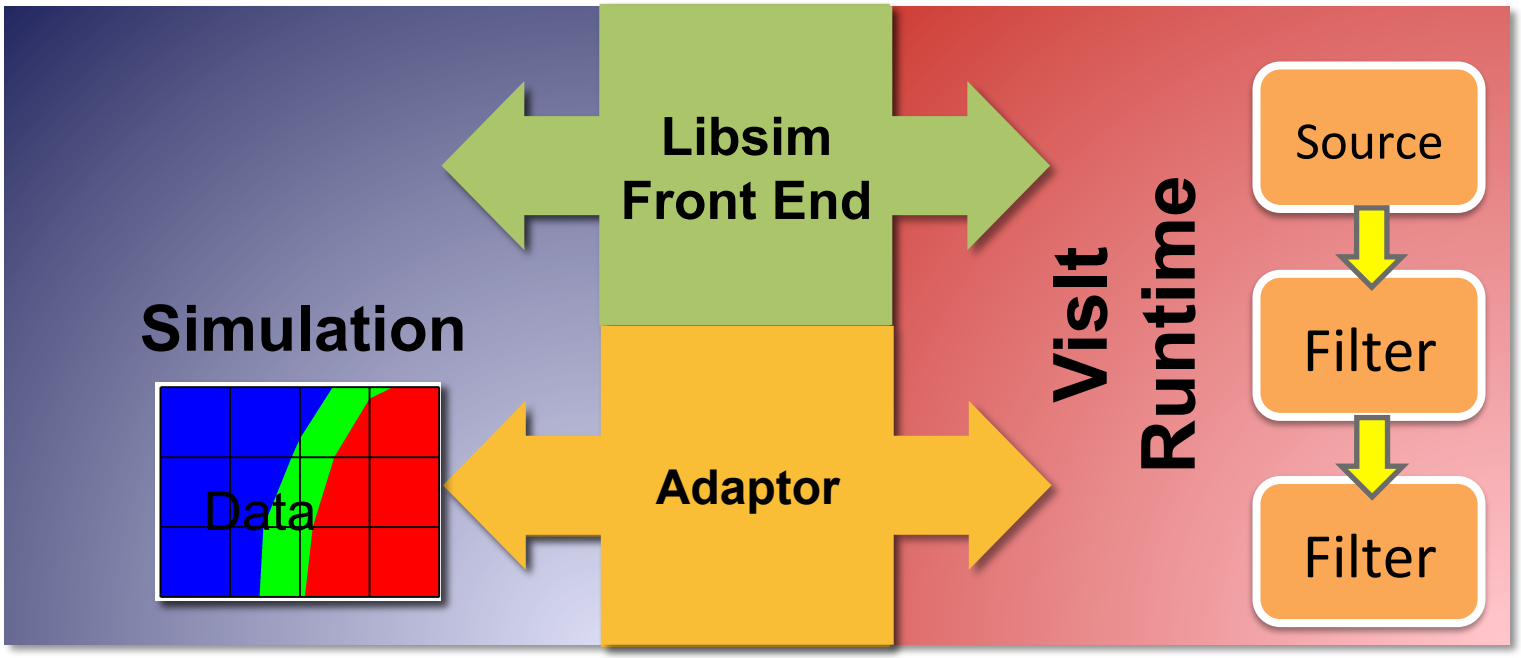




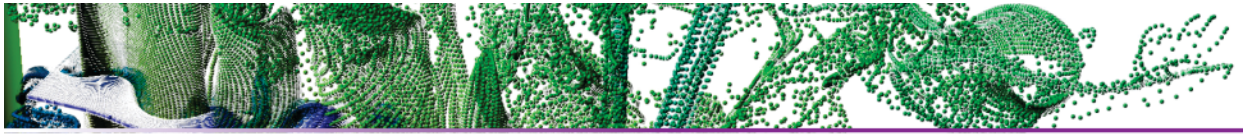
Copyright (c) 2015 Intelligent Light
All Rights Reserved

Libsim puts VisIt in situ

- VisIt provides Libsim, a library that simulations may use to let VisIt connect and access their data
- Share simulation's arrays to avoid I/O and data movement



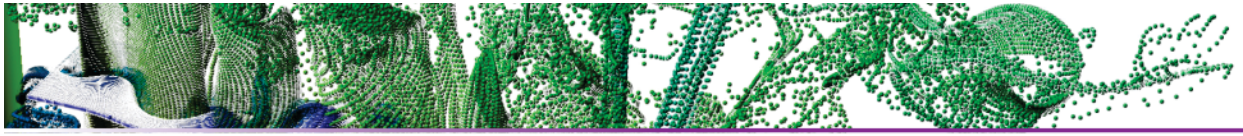
Copyright (c) 2015 Intelligent Light
All Rights Reserved



Libsim Programming Interface

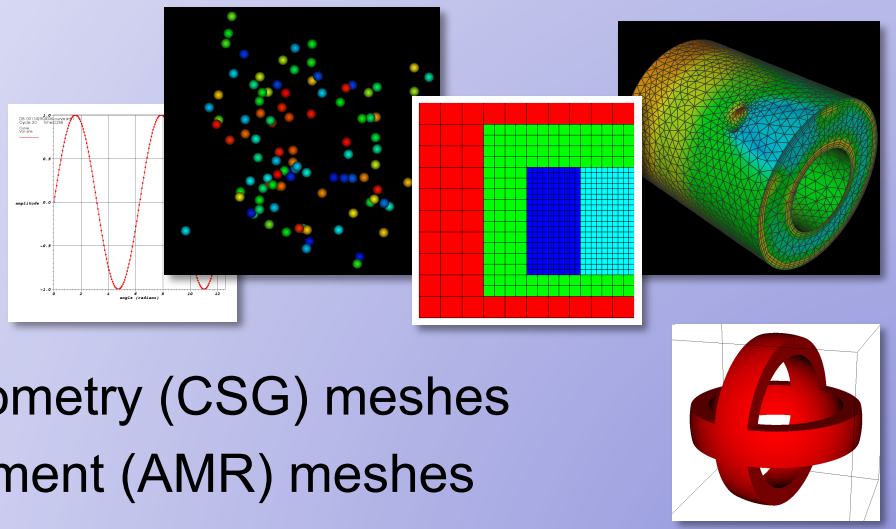
- Control Interface handles connections and processing commands
- Data Interface handles passing data back to Libsim
- Libsim bindings exist for C, C++, Fortran, Python
  - Fortran functions may have abbreviated names, all lower case
  - Fortran functions take a length parameter after each string parameter
- Libsim allows for a lot of flexibility
  - Interactive vs Batch (or support both)
  - Blocking vs Polling
  - A lot of common patterns can be copied from examples with little modification

Copyright (c) 2015 Intelligent Light
All Rights Reserved

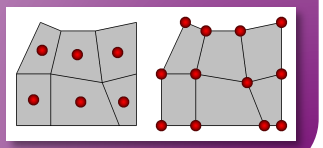


VisIt/Libsim Data Model

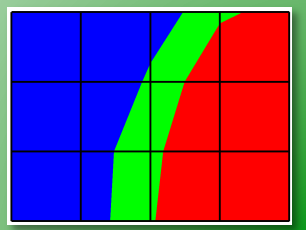
- Mesh Types
  - Structured meshes
    - Rectilinear/Curvilinear
    - I-Blanking
  - Particle meshes
  - Constructive Solid Geometry (CSG) meshes
  - Adaptive Mesh Refinement (AMR) meshes
  - Unstructured & Polyhedral meshes



- Variables
  - 1 to N components
  - Zonal and Nodal
  - Enumerated type



- Materials
- Species



Copyright (c) 2015 Intelligent Light
All Rights Reserved

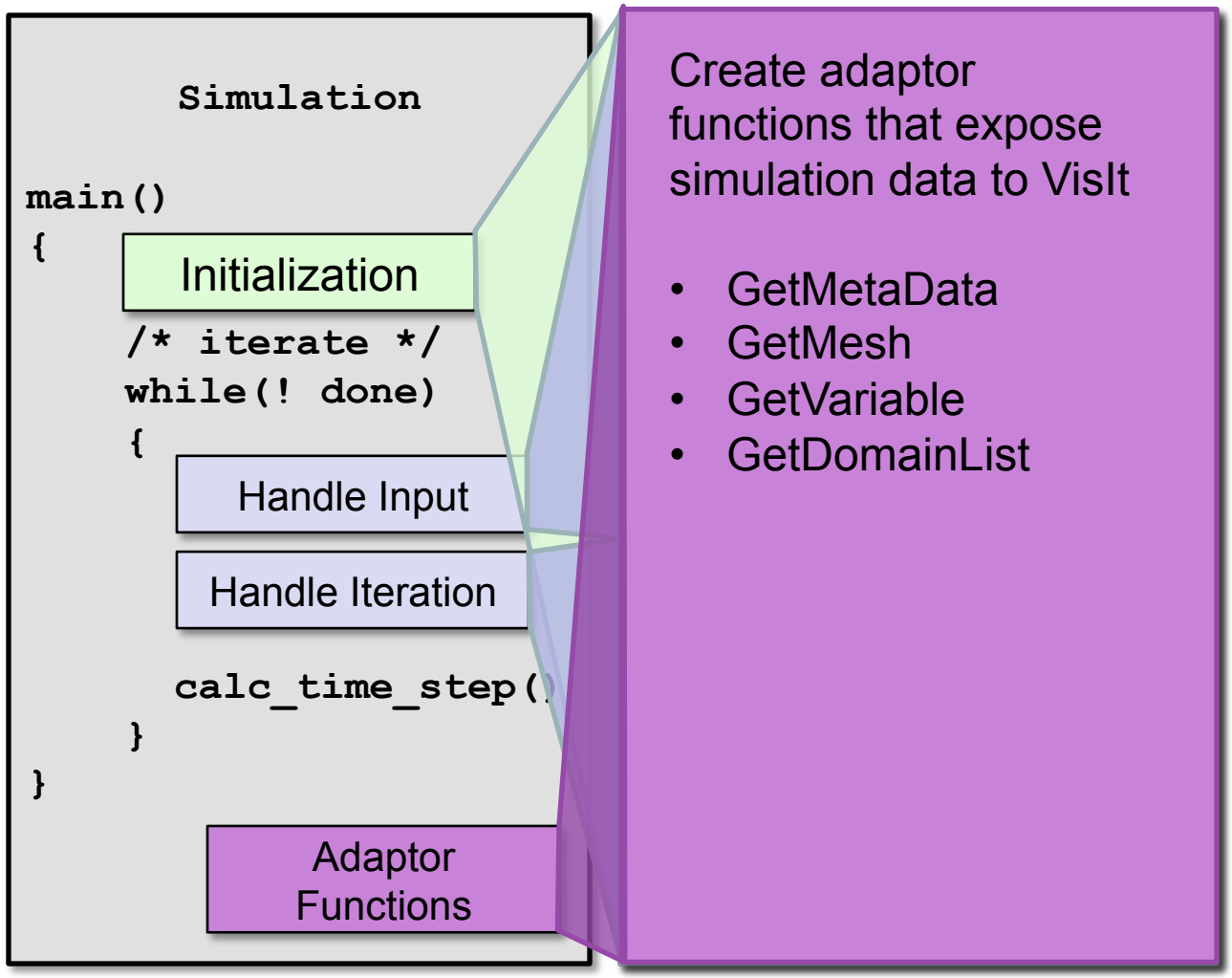
Instrumenting a Simulation

Instrumentation can be performed incrementally in steps

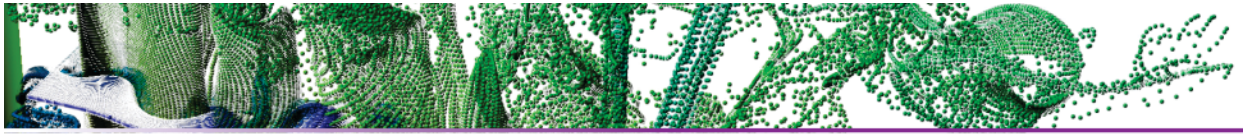
Step 1: Initialization

Step 2: Iteration

Step 3: Adaptor



Copyright (c) 2015 Intelligent Light
All Rights Reserved



Environment / Setup

Step 1

- Pass options to Libsim, such as path to VisIt
- Libsim needs to know about the environment to load the VisIt runtime library

```
VisItSetDirectory(
    "/usr/local/visitdir");
VisItSetOption("-debug 5");
```

```
/*Read environment on rank 0*/
char *env = NULL;
if(par_rank == 0)
  env = VisItGetEnvironment();
/* Pass the environment to all
other processors collectively.
*/
VisItSetupEnvironment2(env);
if(env != NULL) free(env);
```

Copyright (c) 2015 Intelligent Light
All Rights Reserved

Broadcast Functions (Parallel)

Step 1

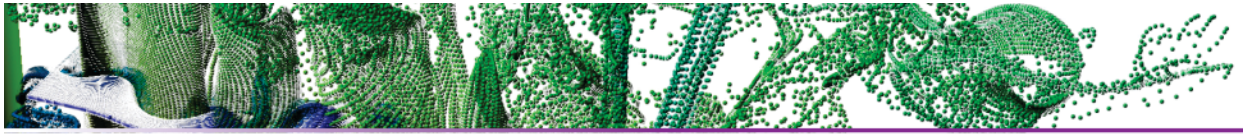
- Libsim will on occasion need to communicate among parallel ranks
- The application must register functions to be used for communication *(keeps MPI dependency out of Libsim)*

Fortran Versions

```
c-----------------------------------------------------------------------
c visitbroadcastintfunction
c-----------------------------------------------------------------------
      integer function visitbroadcastintfunction(value, sender)
      implicit none
      include "mpif.h"
      integer value, sender, IERR
      call MPI_BCAST(value,1,MPI_INTEGER,sender,MPI_COMM_WORLD,ierr)
      visitbroadcastintfunction = 0
      end

c-----------------------------------------------------------------------
c visitbroadcaststringfunction
c-----------------------------------------------------------------------
      integer function visitbroadcaststringfunction(str, lstr, sender)
      implicit none
      include "mpif.h"
      character*8 str
      integer     lstr, sender, IERR
      call MPI_BCAST(str,lstr,MPI_CHARACTER,sender,MPI_COMM_WORLD,ierr)
      visitbroadcaststringfunction = 0
      end
```

Copyright (c) 2015 Intelligent Light
All Rights Reserved



Set Rank, Parallel Flag, and Communicator (Parallel)

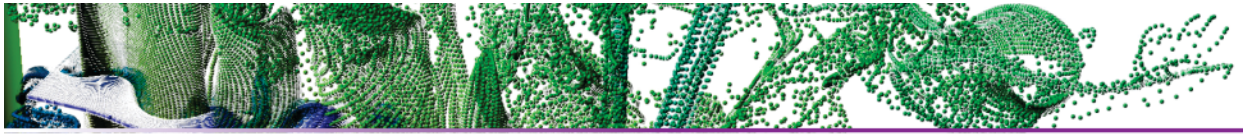
Step 1

- Libsim needs to know the rank and size of the process group
- An MPI communicator can be installed for Libsim that can be used to restrict operations to a subset of processors

```
/* Set parallel flag and rank*/
int par_rank = 0, par_size  = 1;
MPI_Comm_rank (MPI_COMM_WORLD,
               &par_rank);
MPI_Comm_size (MPI_COMM_WORLD,
               &par_size);
VisItSetParallel(par_size > 1);
VisItSetParallelRank(par_rank);

/* Tell Libsim which MPI
communicator to use. */
MPI_Comm comm;
MPI_Comm_dup(MPI_COMM_WORLD,
             &comm);
VisItSetMPICommunicator(
    (void *)&comm);
```

Copyright (c) 2015 Intelligent Light
All Rights Reserved



Batch vs Interactive

Step 1

Libsim permits multiple ways of instrumenting the main loop

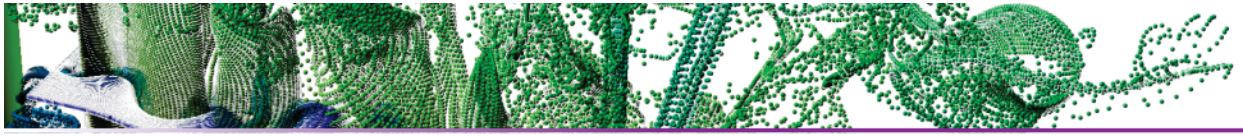
Batch

- VisIt 2.9.0 extends Libsim with a batch-only support
  - Forces load of VisIt runtime library
  - Does need to listen for interactive connections (simpler to implement)
  - Does not need VisIt clients to set up plots for in situ

Interactive

- The simulation must call Libsim periodically to respond to VisIt connection requests or commands
  - Opens a listen socket for inbound connections
  - Writes a sim2 file that VisIt can use to initiate a connection
  - A successful connection causes the VisIt runtime library to be loaded

Copyright (c) 2015 Intelligent Light
All Rights Reserved



Batch Initialization

Step 1

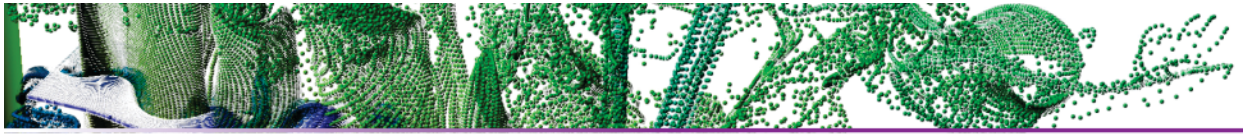
- Batch Initialization requires the VisIt runtime library to be loaded explicitly
- Once the runtime is loaded, register data adaptor functions
- Call functions to set up visualization

```
VisItInitializeRuntime();

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

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

Copyright (c) 2015 Intelligent Light
All Rights Reserved



Interactive Initialization

Step 1

- Interactive initialization assumes that code for input processing will be added to the main loop
- VisIt connections are initiated by reading a *".sim2"* file created by the simulation on rank 0

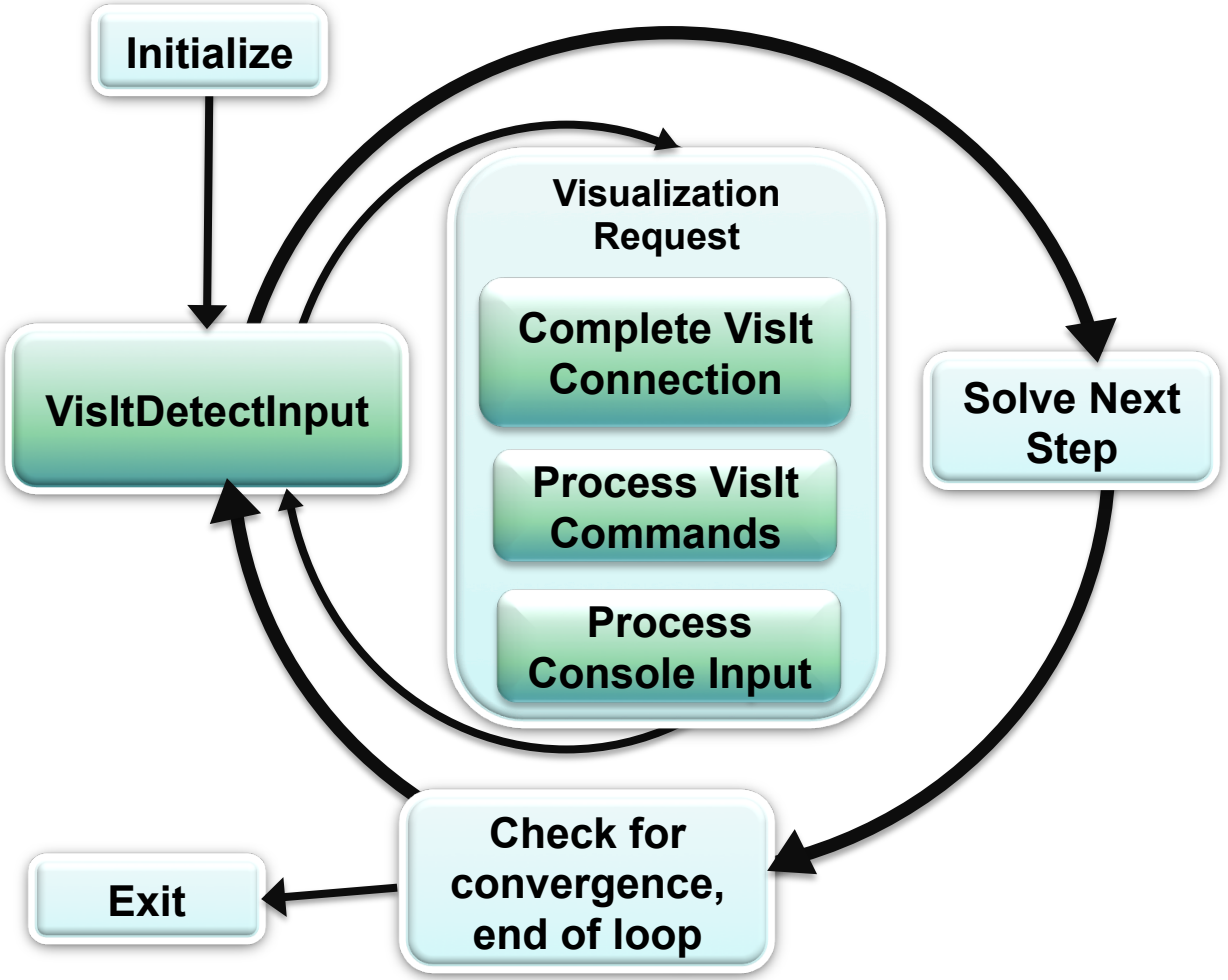
```
if(par_rank == 0)
{
    /* Write out .sim2 file that VisIt uses to connect. */
    VisItInitializeSocketAndDumpSimFile("sim_name",
        "A useful description of the simulation",
        "/path/to/where/sim/was/started",
        NULL, /* reserved */
        NULL, /* reserved */
        NULL  /* optional: pass filename for sim2 file */
    );
}
```

Copyright (c) 2015 Intelligent Light
All Rights Reserved

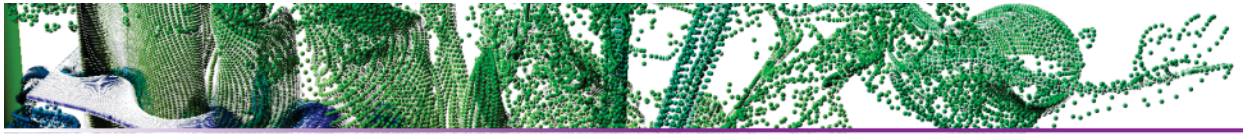
Interactive Main Loop

Step 2

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



Copyright (c) 2015 Intelligent Light
All Rights Reserved

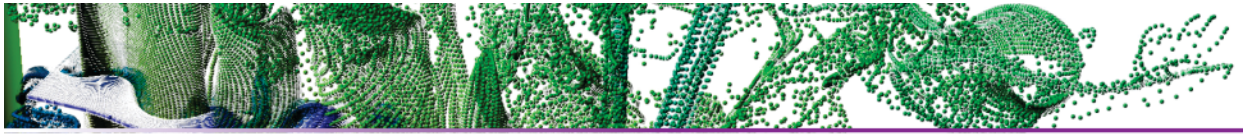


VisItDetectInput

Step 2

```
int mainloop_interactive(simulation_data *sim)
{
    int visitstate, blocking = 0;
    while(1) {
        if(sim->par_rank == 0) {
            visitstate = VisItDetectInput(blocking, -1);
        }
        switch(visitstate) {
        case 0: /* 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)
                SetupCallbacks(sim);
            break;
        case 2: /* VisIt wants to tell the engine something. */
            if(!ProcessVisItCommand(sim)) {
                /* Disconnect on an error or closed connection. */
                VisItDisconnect();
            }
            break;
        case 3: /* VisItDetectInput detected console input */
            break;
        default: /* Error */
            return visitstate;
        }
    }
    return 0;
}
```

Copyright (c) 2015 Intelligent Light
All Rights Reserved



Operations During an Iteration

Step 2

- The simulation can tell VisIt that the time step changed so new metadata will be pushed to VisIt
- The simulation can request that all plots be updated with new simulation data
- The simulation can save the plots to an image or it can export them

```
VisItTimeStepChanged();
VisItUpdatePlots();
```

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

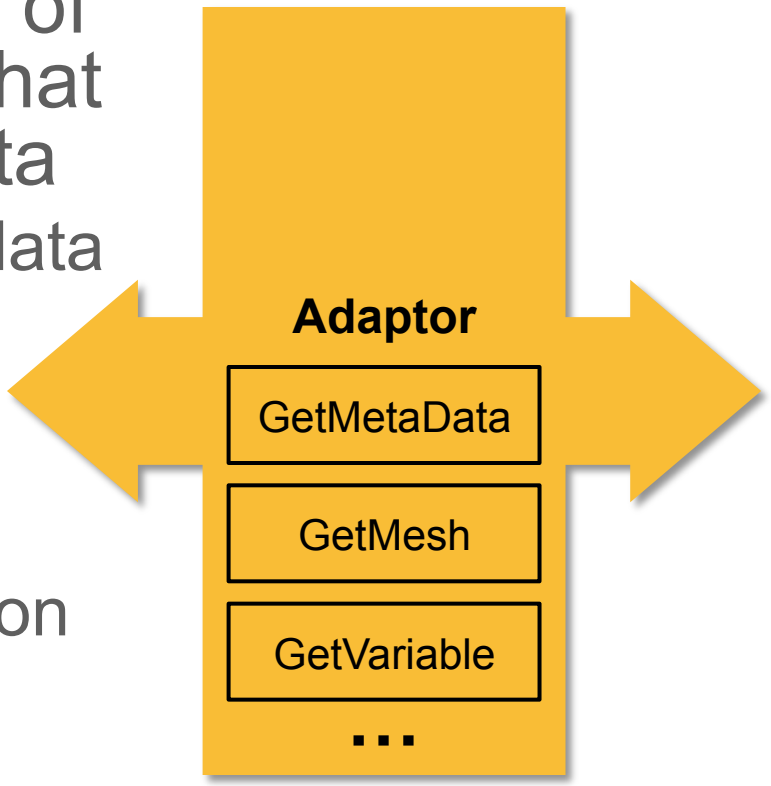
```
visit_handle vars = VISIT_INVALID_HANDLE;
VisIt_NameList_alloc(&vars);
VisIt_NameList_addName(vars, "default");
sprintf(fn, "export%04d", count++);
VisItExportDatabase(fn,"FieldViewXDB_1.0",
                    vars);
VisIt_NameList_free(vars);
```

Copyright (c) 2015 Intelligent Light
All Rights Reserved

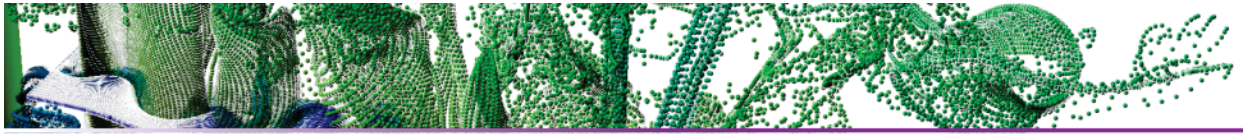
Adaptor

Step 3

- An adaptor comprises a set of functions in the simulation that VisIt calls when it needs data
  - Packages your simulation's data in terms that VisIt can understand
  - Return actual pointers to simulation data (*zero copy*)
  - Return alternate representation that VisIt can free
  - Written in C, C++, Fortran, Python



Copyright (c) 2015 Intelligent Light
All Rights Reserved



Adaptor Functions

Function	Description
GetMetaData	Creates a metadata object that tells VisIt the entities advertised from the simulation
GetMesh	Returns a mesh object that contains the simulation's mesh coordinates and connectivity
GetVariable	Returns a data array object containing a simulation field
GetMaterial	Return a material object describing how the mesh can be decomposed into various materials
GetSpecies	Return a species object indicating how the mesh's materials are decomposed into various material species
GetDomainList	Return a list of domains owned by the current MPI rank

Additional adaptor functions return data for advanced features

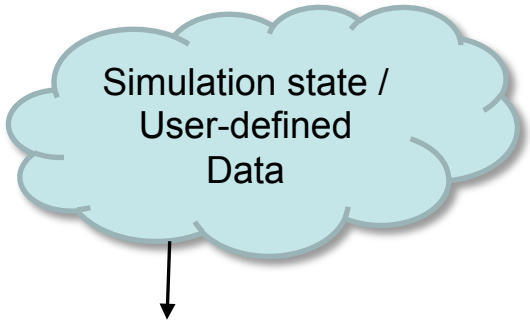
Copyright (c) 2015 Intelligent Light
All Rights Reserved

Registering Adaptor Functions

Step 3

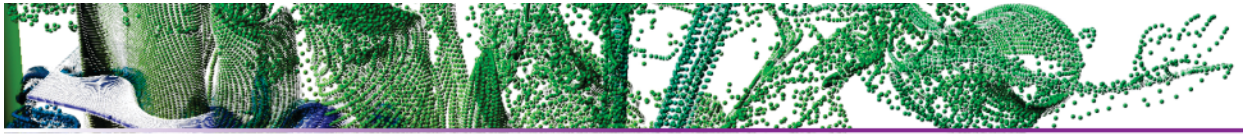
- Adaptor functions need to be registered with Libsim at runtime, once the VisIt runtime library has been loaded
  - Fortran adaptors rely on functions with specific names

```
visit_handle SimGetMetaData(void *cbdata)
{
    visit_handle md = VISIT_INVALID_HANDLE;
    if(VisIt_SimulationMetaData_alloc(&md)
       == VISIT_OKAY)
    {
        /* Add items here */
    }
    return md;
}
```




```
VisItSetGetMetaData(SimGetMetaData, (void*)sim);
VisItSetGetMesh(SimGetMesh, (void*)sim);
VisItSetGetCurve(SimGetVariable, (void*)sim);
```

Copyright (c) 2015 Intelligent Light
All Rights Reserved



Fortran Adaptors

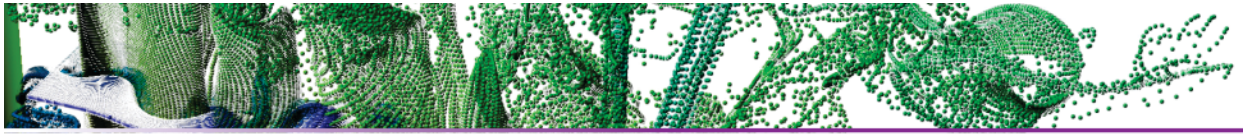
Step 3

- Fortran simulations rely on specific function names being linked into the simulation to satisfy the Libsim bindings
- All adaptor functions must be provided for a successful link (functions can be minimal)

Fortran Adaptor Functions
visitgetmetadata
visitgetmesh
visitgetvariable
visitgetmaterial
visitgetcurve
visitgetdomainlist
visitgetdomainbounds
visitgetdomainnesting

```
integer function visitgetmesh(domain,name,lname)
implicit none
character*8 name
integer     domain, lname
include "visitfortransimV2interface.inc"
visitgetmesh = VISIT_INVALID_HANDLE
end
```

Copyright (c) 2015 Intelligent Light
All Rights Reserved



Example GetMetaData Function

Step 3

- GetMetaData returns the inventory of data that will be exposed to VisIt (meshes, scalars, etc)
- Used to populate menus, etc

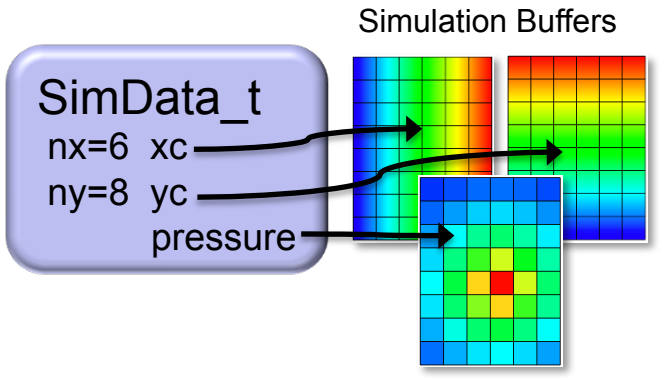
```
visit_handle SimGetMetaData(void *cbdata)
{
    visit_handle md = VISIT_INVALID_HANDLE;
    if(VisIt_SimulationMetaData_alloc(&md) == VISIT_OKAY)
    {
        visit_handle mmd = VISIT_INVALID_HANDLE;
        VisIt_SimulationMetaData_setMode(md,
            VISIT_SIMMODE_RUNNING);
        VisIt_SimulationMetaData_setCycleTime(md, 0, 0.);

        if(VisIt_MeshMetaData_alloc(&mmd) == VISIT_OKAY)
        {
            VisIt_MeshMetaData_setName(mmd, "mesh2d");
            VisIt_MeshMetaData_setMeshType(mmd,
                VISIT_MESHTYPE_RECTILINEAR);
            VisIt_MeshMetaData_setTopologicalDimension(mmd, 2);
            VisIt_MeshMetaData_setSpatialDimension(mmd, 2);
            VisIt_MeshMetaData_setNumDomains(mmd, 1);
        }
        VisIt_SimulationMetaData_addMesh(md, mmd);
    }
    return md;
}
```

Copyright (c) 2015 Intelligent Light
All Rights Reserved

Example GetMesh Function

Step 3




```
visit_handle
SimGetMesh(int domain, const char *name,
    void *cbdata)
{   SimData_t *sim = (SimData_t*)cbdata;
    visit_handle h = VISIT_INVALID_HANDLE;
    if(strcmp(name, "mesh") == 0) {
        if(VisIt_CurvilinearMesh_alloc(&h) != VISIT_ERROR) {
            visit_handle hxc, hyc;
            VisIt_VariableData_alloc(&hxc);
            VisIt_VariableData_alloc(&hyc);
            VisIt_VariableData_setDataF(hxc, VISIT_OWNER_SIM, 1,
                sim->nx, sim->xc);
            VisIt_VariableData_setDataF(hyc, VISIT_OWNER_SIM, 1,
                sim->ny, sim->yc);
            VisIt_CurvilinearMesh_setCoordsXY(h, hxc, hyc);
        }
    }
    return h;
}
```

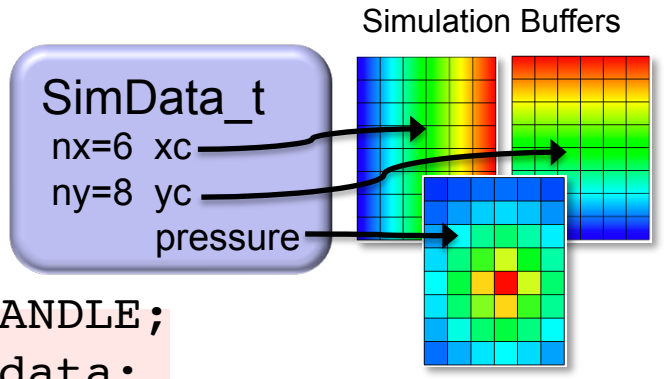
Copyright (c) 2015 Intelligent Light
All Rights Reserved

Example GetVariable Function

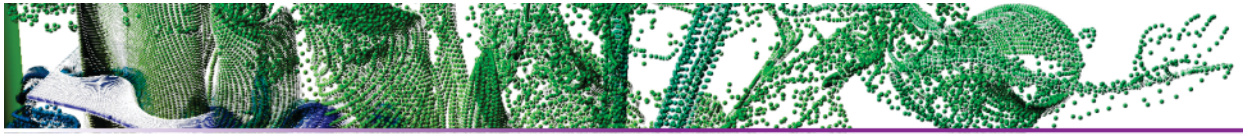
Step 3

```
// Example
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



Copyright (c) 2015 Intelligent Light
All Rights Reserved



Example GetDomainList Function

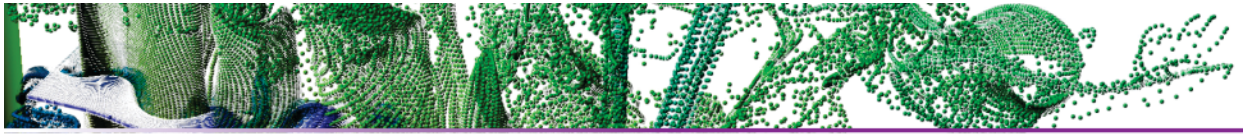
Step 3

- The GetDomainList function returns the indices of the grids that exist on the current MPI rank
- Used in load balancer to constrain work assignments
- Possible to return decompositions for multiple meshes
- Mesh decomposition free to change over time (*as in AMR*)

```
C-----------------------------------------------------------------------
c visitgetdomainlist
C-----------------------------------------------------------------------
      integer function visitgetdomainlist()
      implicit none
      include "visitfortransimV2interface.inc"
ccc   PARALLEL state common block
      integer par_rank, par_size
      common /PARALLEL/ par_rank, par_size
ccc   local vars
      integer h, dl, err

c     Tell VisIt that there are as many domains as processors and this
c     processor just has one of them.
      h = VISIT_INVALID_HANDLE
      if(visitdomainlistalloc(h).eq.VISIT_OKAY) then
          if(visitvardataalloc(dl).eq.VISIT_OKAY) then
              err = visitvardataseti(dl, VISIT_OWNER_SIM, 1, 1,
     .                               par_rank)
              err = visitdomainlistsetdomains(h, par_size, dl)
          endif
      endif
      visitgetdomainlist = h
      end
```

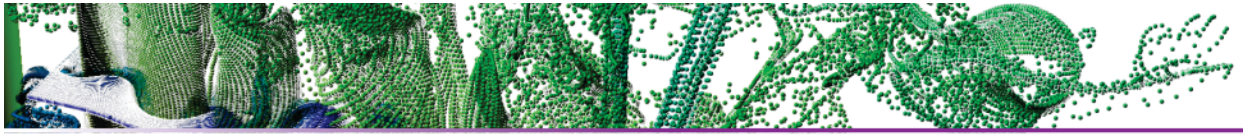
Copyright (c) 2015 Intelligent Light
All Rights Reserved



Conclusions

- We can leverage VisIt's data extraction capabilities at scale via Libsim to write XDB data for FieldView from running simulations
- Extracted files are far smaller than volume data and enable more productive post processing on smaller resources
- This should be of interest to people who develop or support solver codes, as it helps people to use their codes

Copyright (c) 2015 Intelligent Light
All Rights Reserved



Resources

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

Copyright (c) 2015 Intelligent Light
All Rights Reserved