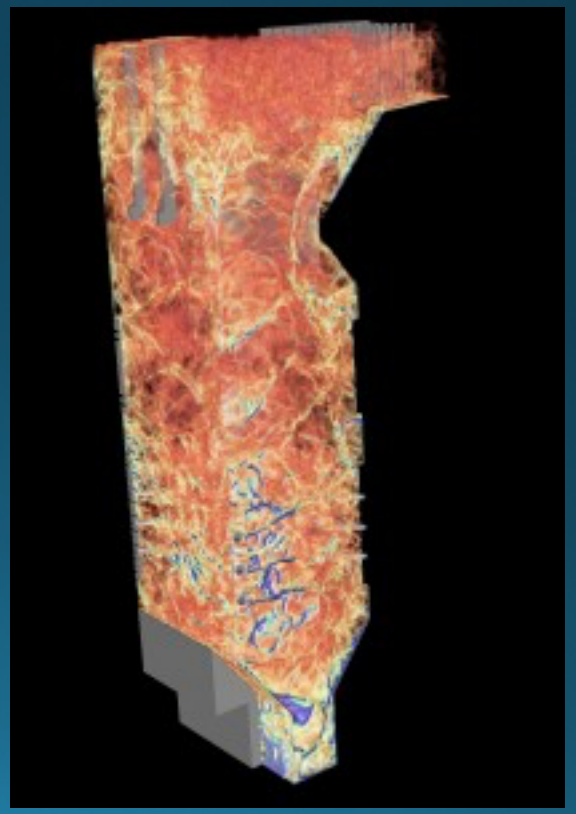
The Uintah – VisIt coupled workflow

Allen R. Sanderson, Alan Humphrey,
John Schmidt, Chuck Hansen, Martin Berzins
Scientific Computing and Imaging Institute,
The University of Utah, Salt Lake City, USA

SIAM CSE 2017

Uintah

- The Uintah software suite is a set of libraries and applications for simulating and analyzing complex chemical and physical reactions.
- http://www.uintah.utah.edu/projects.html

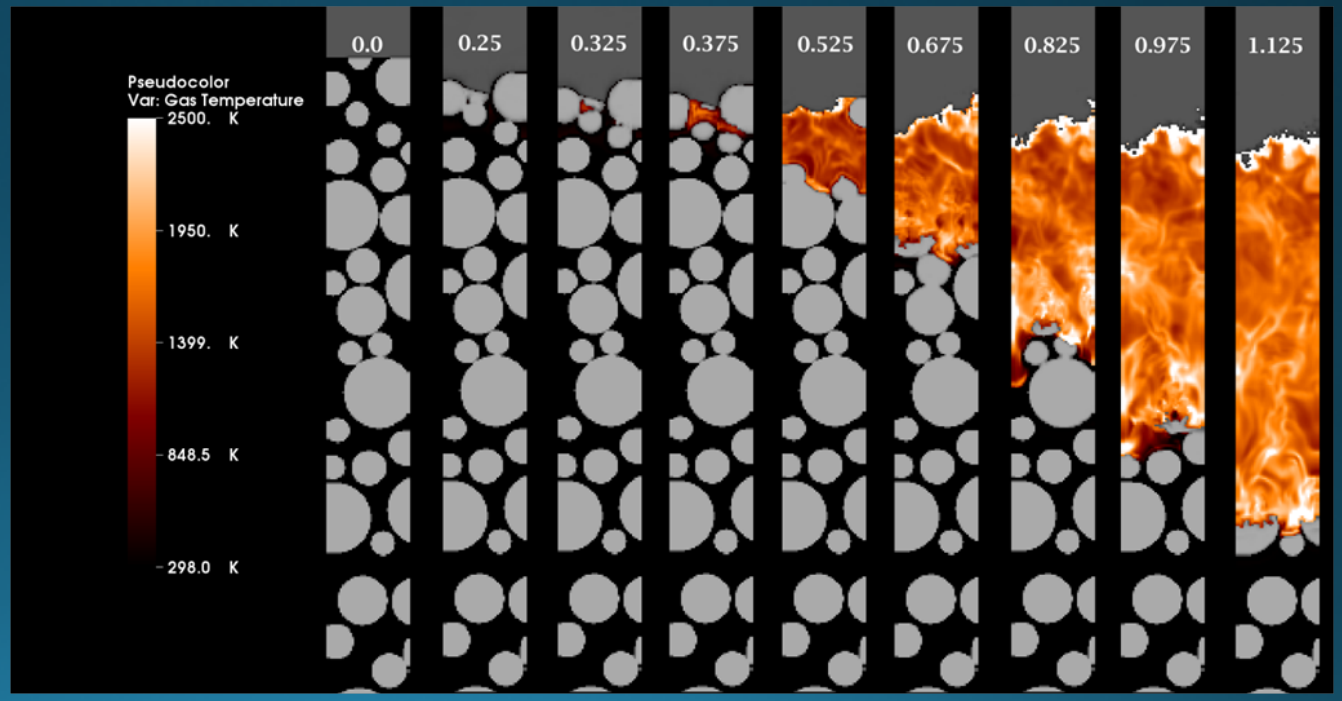


- **The Carbon-Capture Multidisciplinary Simulation Center –** exa-scale computing with V&V/UQ to more rapidly deploy a new technology for providing low cost, low emission electric power generation.

Uintah

- The Uintah software suite is a set of libraries and applications for simulating and analyzing complex chemical and physical reactions.
- http://www.uintah.utah.edu/projects.html

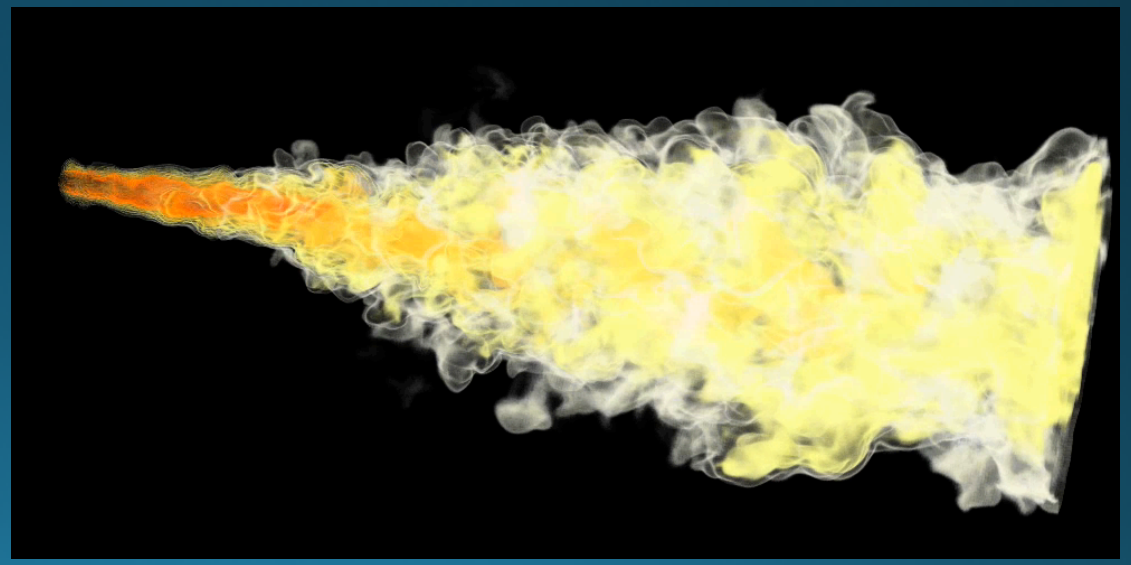
- **Multiscale Multidisciplinary Modeling of Electronic Materials Collaborative Research Alliance**



Uintah

- The Uintah software suite is a set of libraries and applications for simulating and analyzing complex chemical and physical reactions.
- http://www.uintah.utah.edu/projects.html

- **Clean Energy from Fossil Fuels –** model various energy technologies from traditional air-fired coal, oxy-fired coal/natural gas, fluidized bed coal combustion and coal gasification to more exotic coal technologies such as chemical looping and under ground thermal treatment.



VisIt – libsim

- Allows a connection between the application and VisIt via sockets.
  - Dynamically loads VisIt's runtime libraries that allows the simulation to act as VisIt's compute engine.
  - Uses a middle layer to move application data to VisIt.

- Using VisIt's libsim as part of the work flow for the:
  - **Runtime layer** where computer scientists develop and maintain infrastructure – i.e. memory, mpi scheduling.
  - **Application layer** where domain scientists develop and maintain the meshing and physics.
  - **User layer** where scientists visualization and analyze the simulation results.

VisIt – libsim

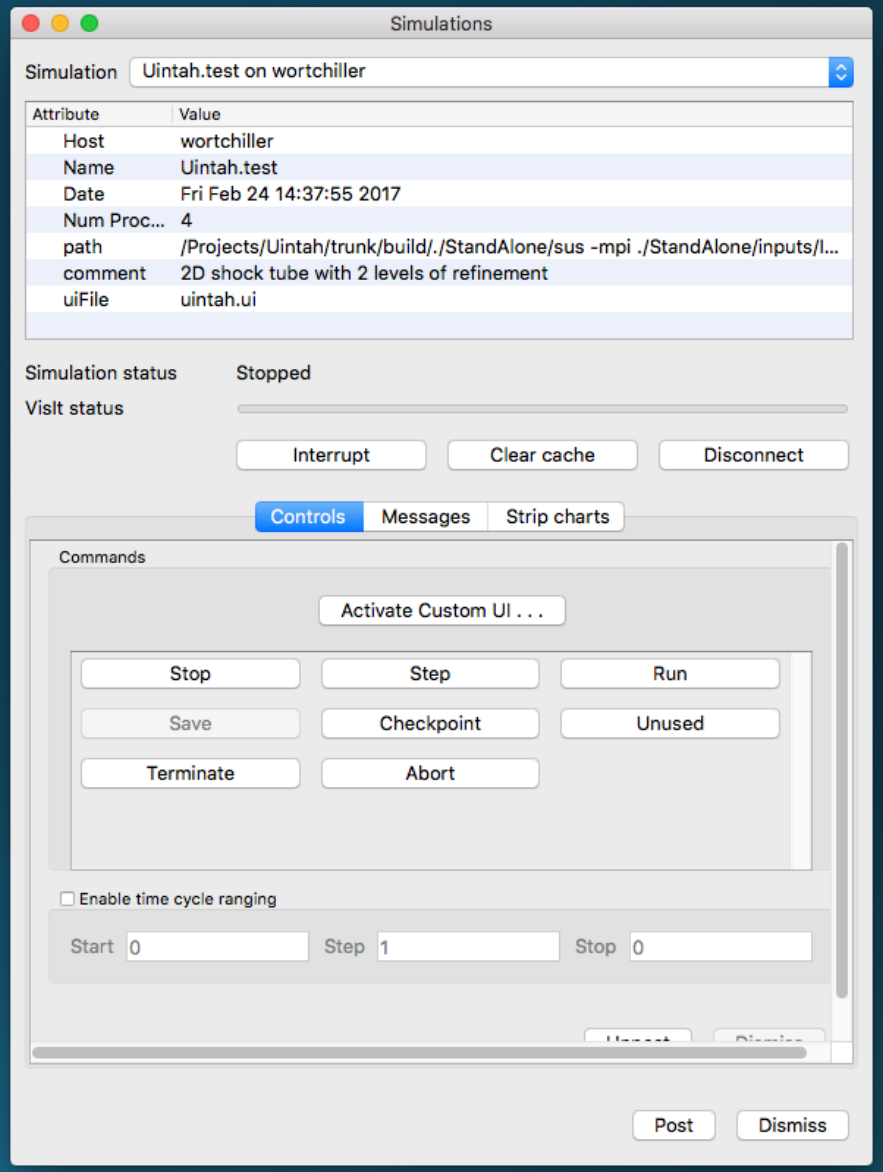
- VisIt's libsim provides for a custom UI
  - Qt based UI
    - Any Qt widgets may be used but VisIt must be able to handle the Signals and Slots.
      - Line Edits, Checkbox, Combobox, etc.
  - Custom UI becomes an application specific dashboard.
  - Setup by the application at runtime – can have different dashboards for different simulation types (though the application must support each dashboard).

VisIt – libsim

- Missing bits
  - Better connection to the simulation variables and the runtime performance values.
  - Simulation variables are global to the simulation and can be of the following:
    - Time based (time step)
    - Mesh based (AMR)
    - Problem spec (initial values, solution order)
    - Operational (output and dump file frequency)
    - Runtime analysis variables
  - Runtime performance values (memory, I/O, task time, mpi timimg)

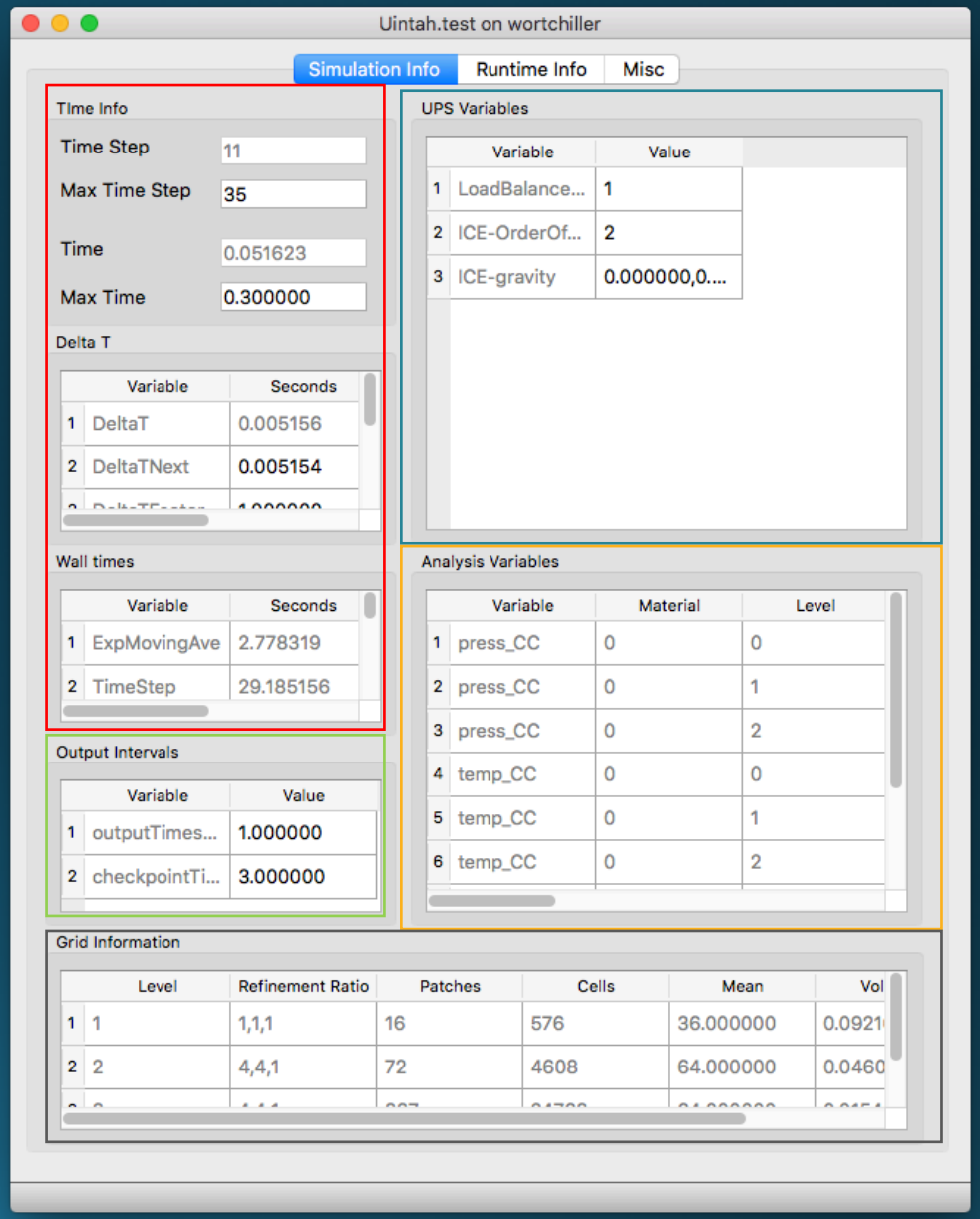
VisIt – libsim

- Simulation Window
  - Tabs are common to all simulations
    - Controls Tab
      - Commands are a unique to the application
      - Time cycle ranging now enabled.
    - Message Tab –
      - Reporting messages that might otherwise go application or error log files.
    - Strip Charts –
      - Global values over time.



Vislt – Custom UI

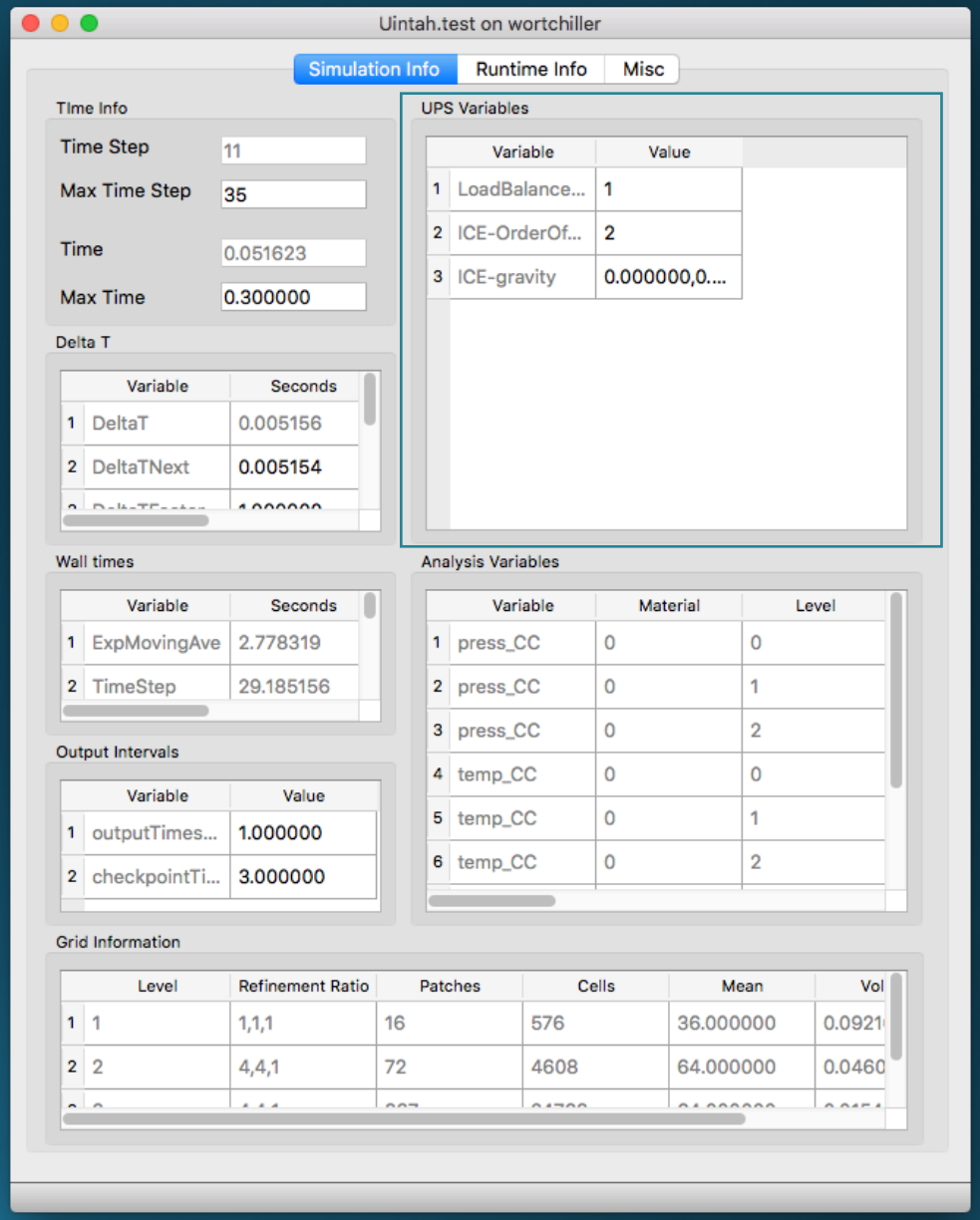
- Simulation dashboard
  - Time info
  - Output parameters
  - Grid information
  - UPS Variables
    - (Uintah Problem Specification)
  - User defined analysis



VisIt – Custom UI

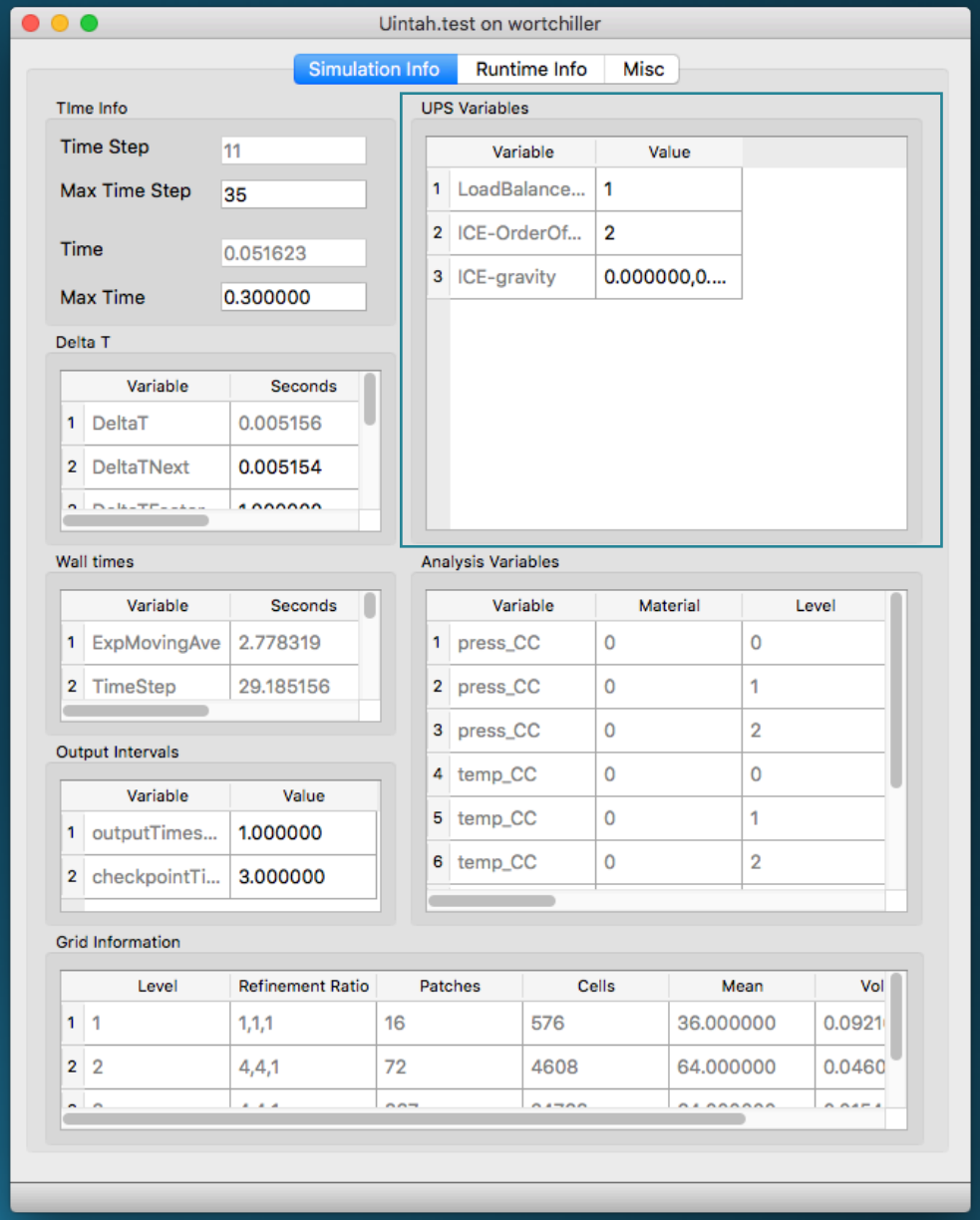
- Tables with generic variables uses a very lightweight structure to communicate between the application and libsim:

```
struct interactiveVar {
        std::string name;
        TypeDescription::Type type;
        int*    Ivalue;
        double* Dvalue;
        Vector* Vvalue;
        bool    modifiable; // If true the user may modify the value,
                            // otherwise it is read-only.
        bool    modified;   // If true the variable was modified by the
                            // user.
        bool    recompile;  // If true and the variable was modified force
                            // the task graph to be recompiled.
};
```



Vislt – Custom UI

- Application developer setups the structure at startup and inspects the structure before each time step execution.
  - Variable may require the task graph to be recompiled.
- Libsim callback is responsible updating pointer and modified flag.
- UPS Variables are generic across the application components.

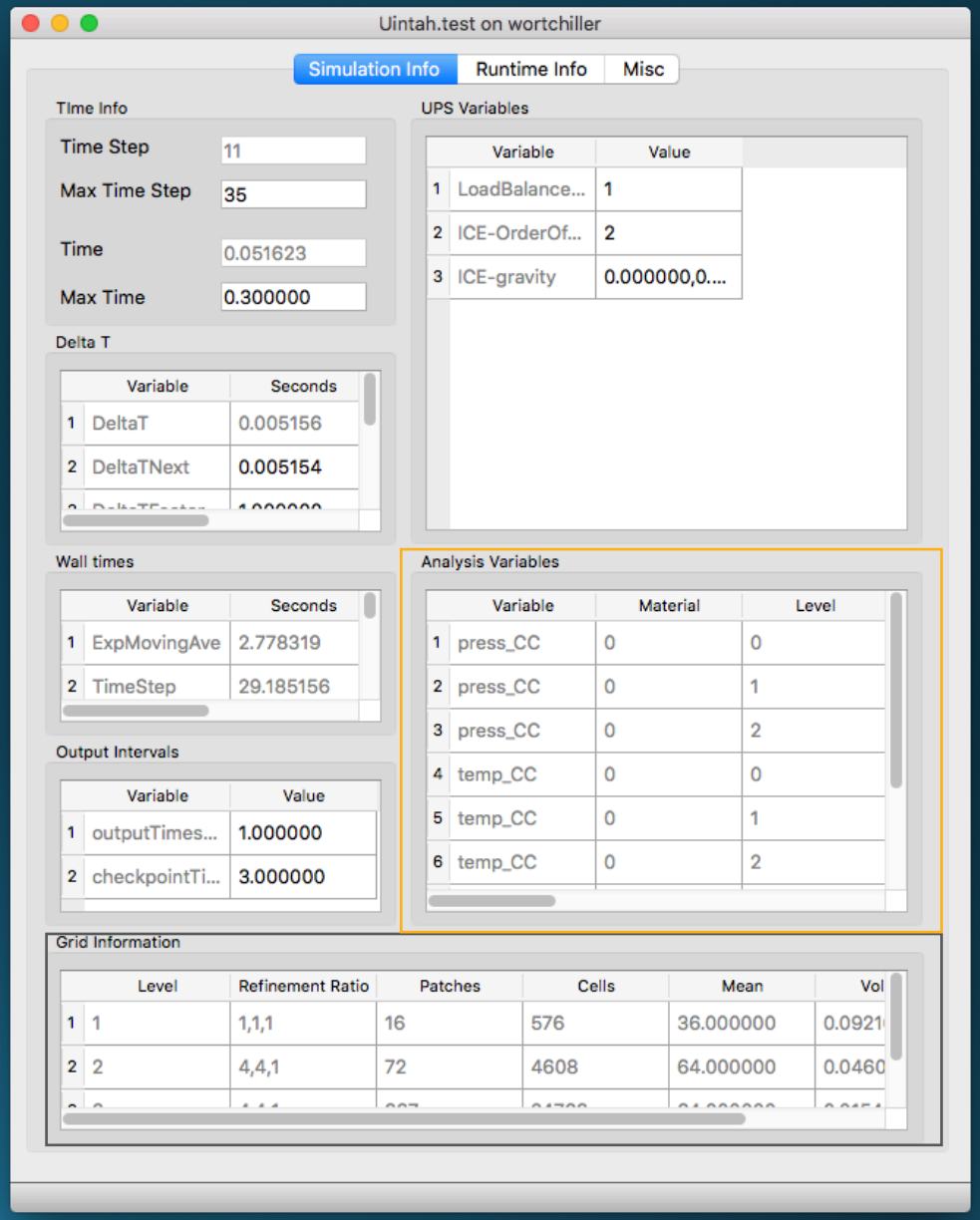


VisIt – Custom UI

- Tables with analysis variables uses a very lightweight structure to communicate between the application data archive and libsim:

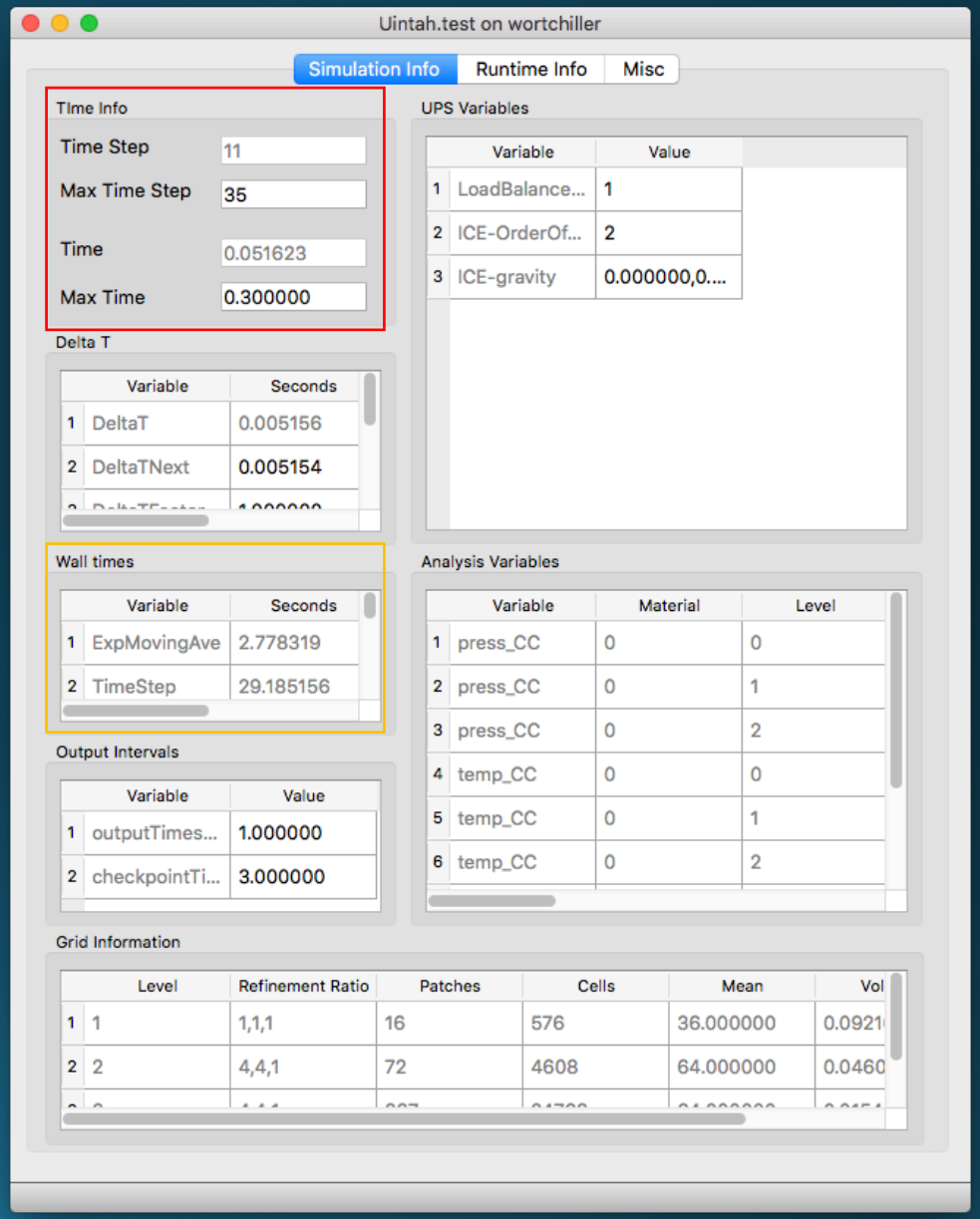
```
struct analysisVar {
        AnalysisType analysisType;
        VarLabel* label;
        VarLabel * reductionMinLabel;
        VarLabel * reductionMaxLabel;
        };
```

- Not yet generic to all analysis tools.
- Grid information – uses application grid information directly. AMR data is shared.



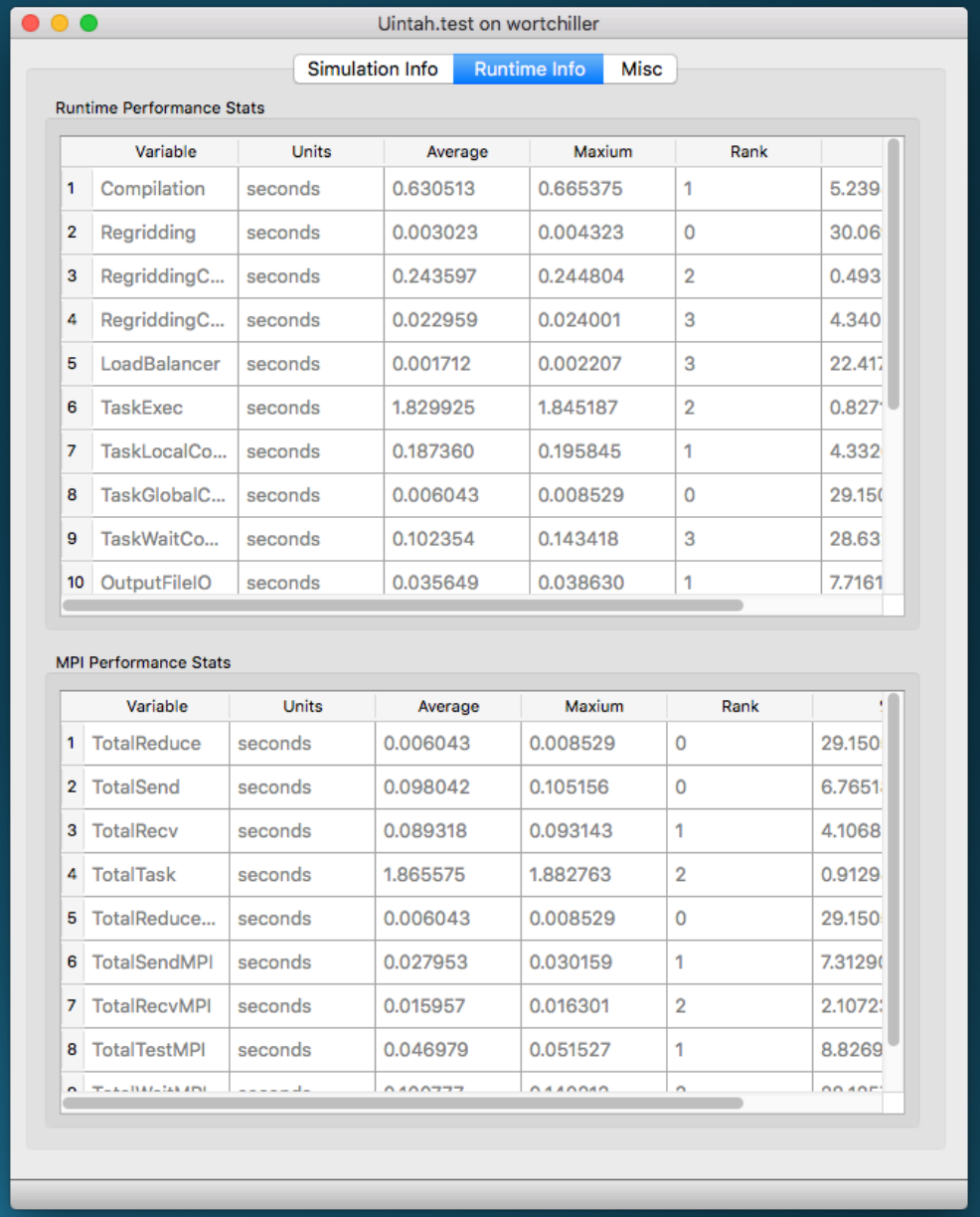
Vislt – Custom UI

- Simulation dashboard
  - Individual vs table variables.
    - Individual variables require their own callback methods.
    - Tables can be modified on the fly (i.e. as the simulation runs variables can be added or removed)
  - Variables are directly linked to the application or via access methods.
    - Can do computational steering



VisIt – Custom UI

- Runtime performance variables.
  - Uses a lightweight templated mapper class that holds:
    - The variable (a enumerated key value).
    - The value.
    - The variable name.
    - The variable units.
  - Application developer sees a std::map
    - Insert method is over ridden.
  - Derived type allows reductions for average value and maximal value with rank.
  - Mapper is shared with libsim.

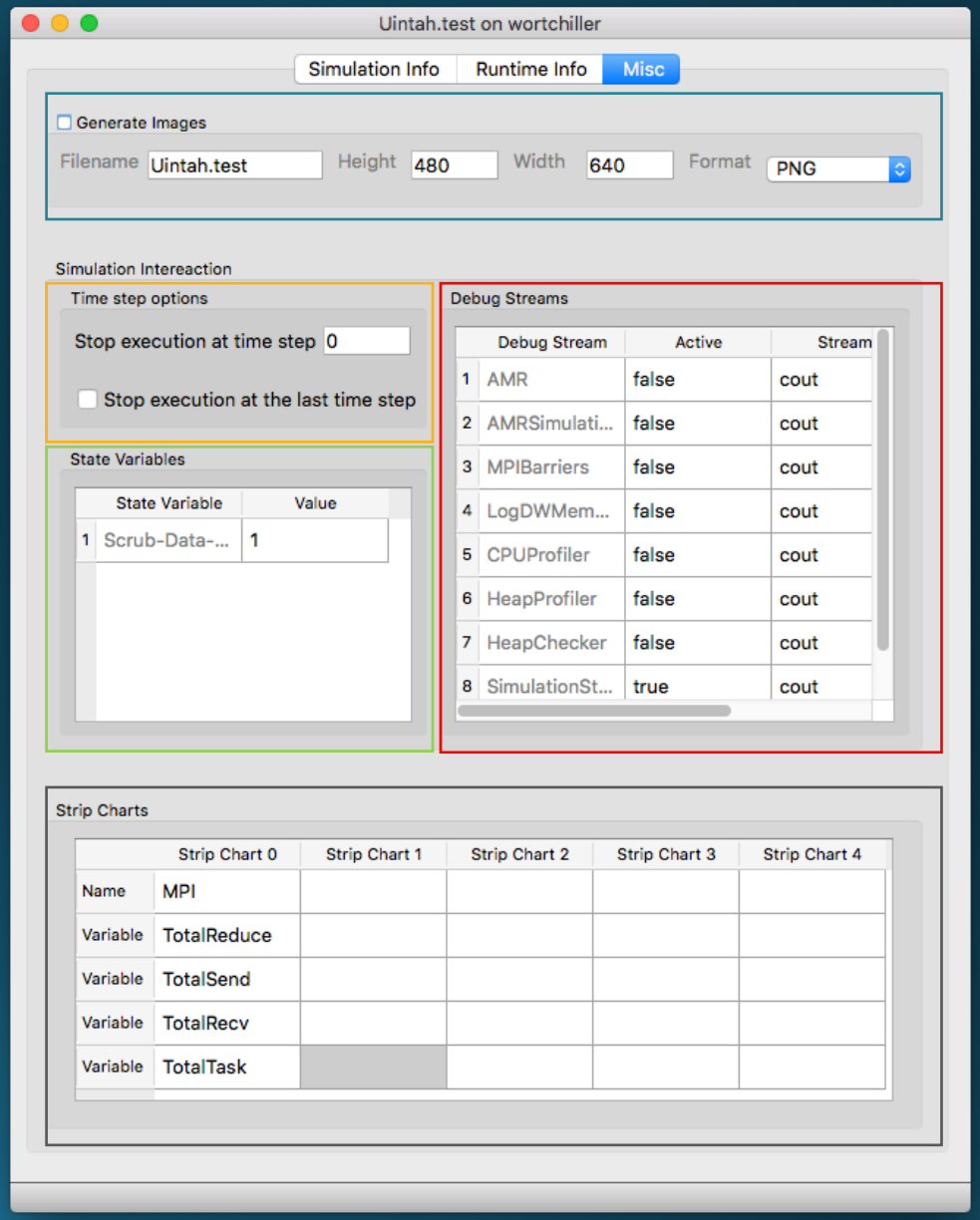


	Variable	Units	Average	Maxium	Rank	
1	Compilation	seconds	0.630513	0.665375	1	5.239
2	Regridding	seconds	0.003023	0.004323	0	30.06
3	RegriddingC...	seconds	0.243597	0.244804	2	0.493
4	RegriddingC...	seconds	0.022959	0.024001	3	4.340
5	LoadBalancer	seconds	0.001712	0.002207	3	22.41
6	TaskExec	seconds	1.829925	1.845187	2	0.827
7	TaskLocalCo...	seconds	0.187360	0.195845	1	4.332
8	TaskGlobalC...	seconds	0.006043	0.008529	0	29.15
9	TaskWaitCo...	seconds	0.102354	0.143418	3	28.63
10	OutputFileIO	seconds	0.035649	0.038630	1	7.7161

	Variable	Units	Average	Maxium	Rank	
1	TotalReduce	seconds	0.006043	0.008529	0	29.150
2	TotalSend	seconds	0.098042	0.105156	0	6.7651
3	TotalRecv	seconds	0.089318	0.093143	1	4.1068
4	TotalTask	seconds	1.865575	1.882763	2	0.9129
5	TotalReduce...	seconds	0.006043	0.008529	0	29.150
6	TotalSendMPI	seconds	0.027953	0.030159	1	7.3129
7	TotalRecvMPI	seconds	0.015957	0.016301	2	2.1072
8	TotalTestMPI	seconds	0.046979	0.051527	1	8.8269

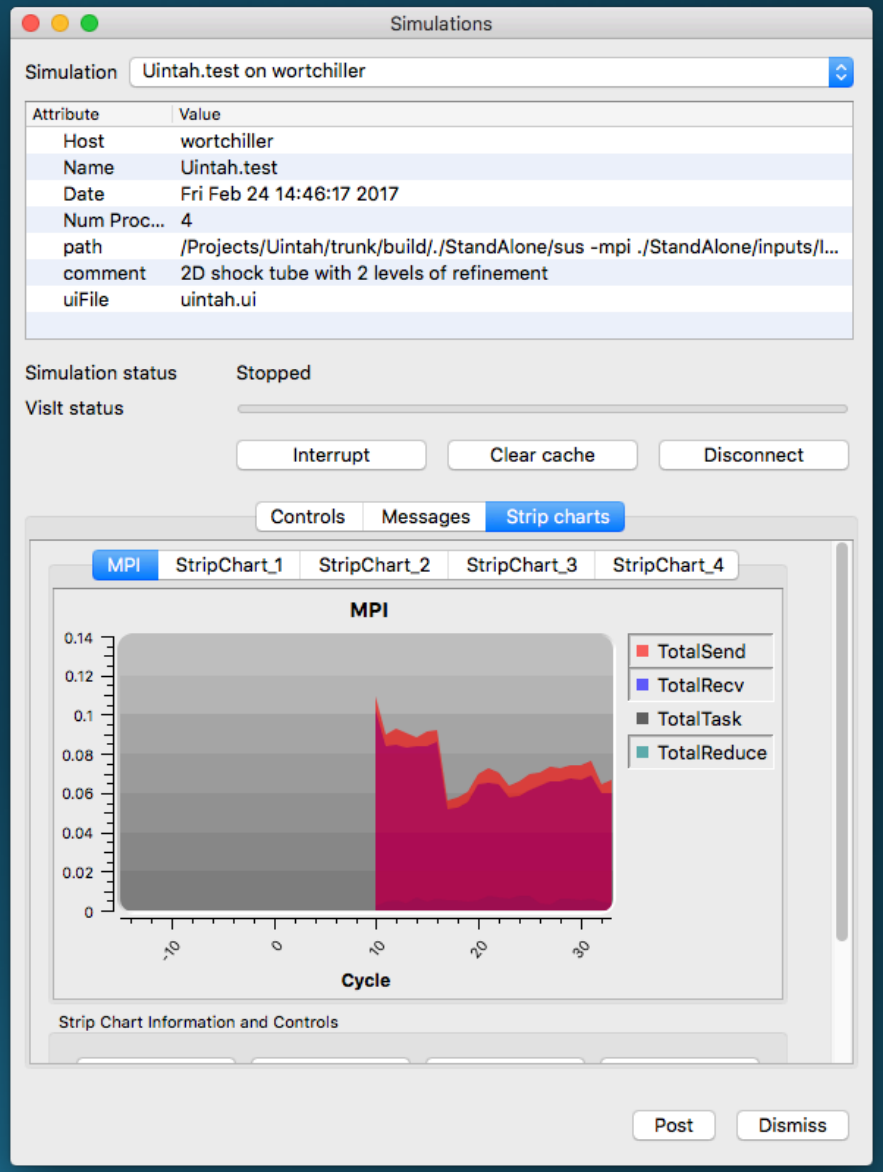
VisIt – Custom UI

- Nice to have features.
  - Generate image frames automatically
    - Used when data is too large to dump to disk.
  - Stopping the execution at specific time steps.
    - Useful for debugging
  - Global state variables
    - Controlling Data Warehouse intermediate variables – used for visual debugging.
  - Controlling debug streams.
    - Direct pointer link is shared with libsim.
  - Strips Charts
    - Allows for the monitoring of global values over the life time of the simulation.



VisIt – Libsim UI

- Strip Charts
  - Allows for the monitoring of global values over the life time of the simulation.

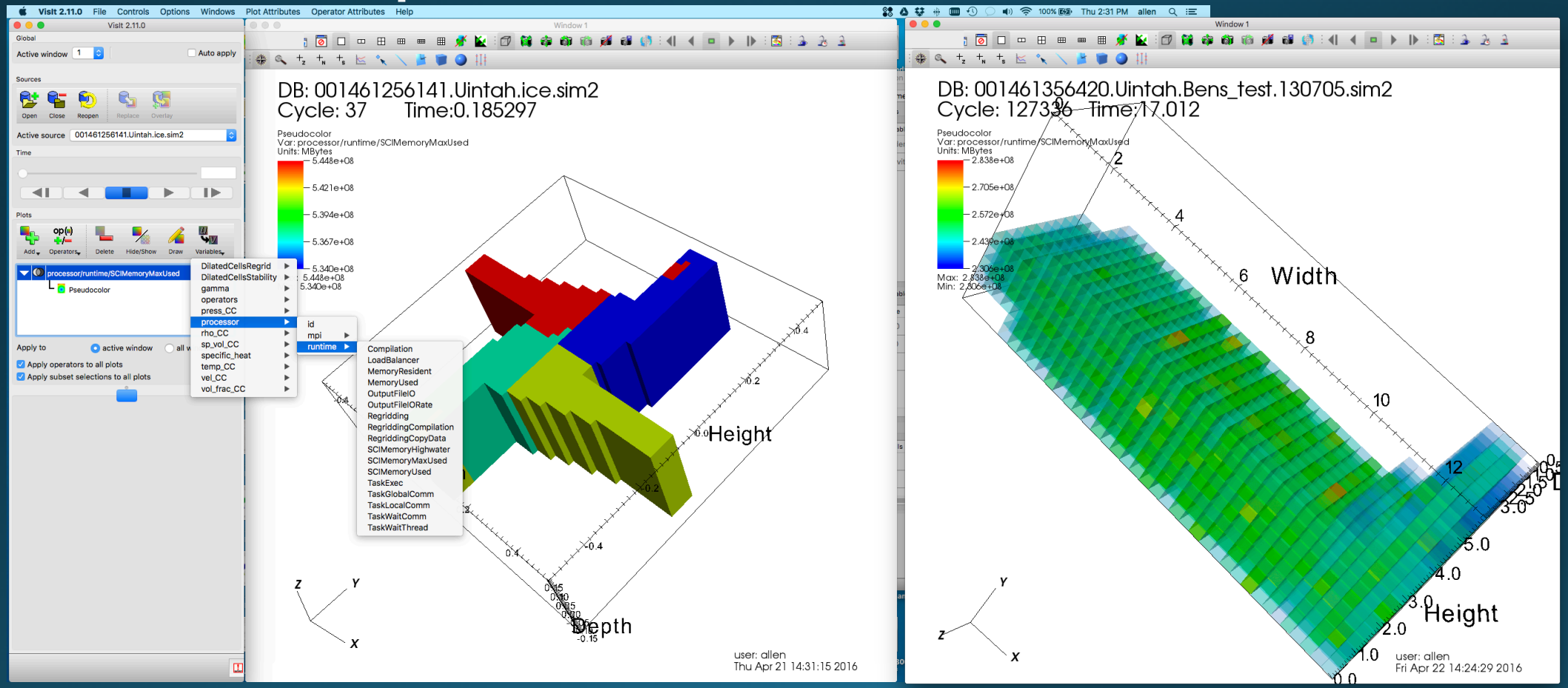


Uintah – VisIt coupled workflow

- Lightweight wrappers are key
  - Structures and methods that are on the application side and used by the application are preferred.
  - Qt Table very generic but not exactly the nicest UI.

- Wrappers are used by both the infrastructure and application scientists as part of development workflow.

Visualization of runtime performance values – processor based.



The Uintah – VisIt coupled workflow

- Documentation for developers and users:
  - http://uintah-build.sci.utah.edu/trac/wiki/VisitUintahInSitu

- Acknowledgements
  - Our many colleagues on the VisIt and Uintah teams.
  - This material is based upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number(s) DE-NA0002375.