

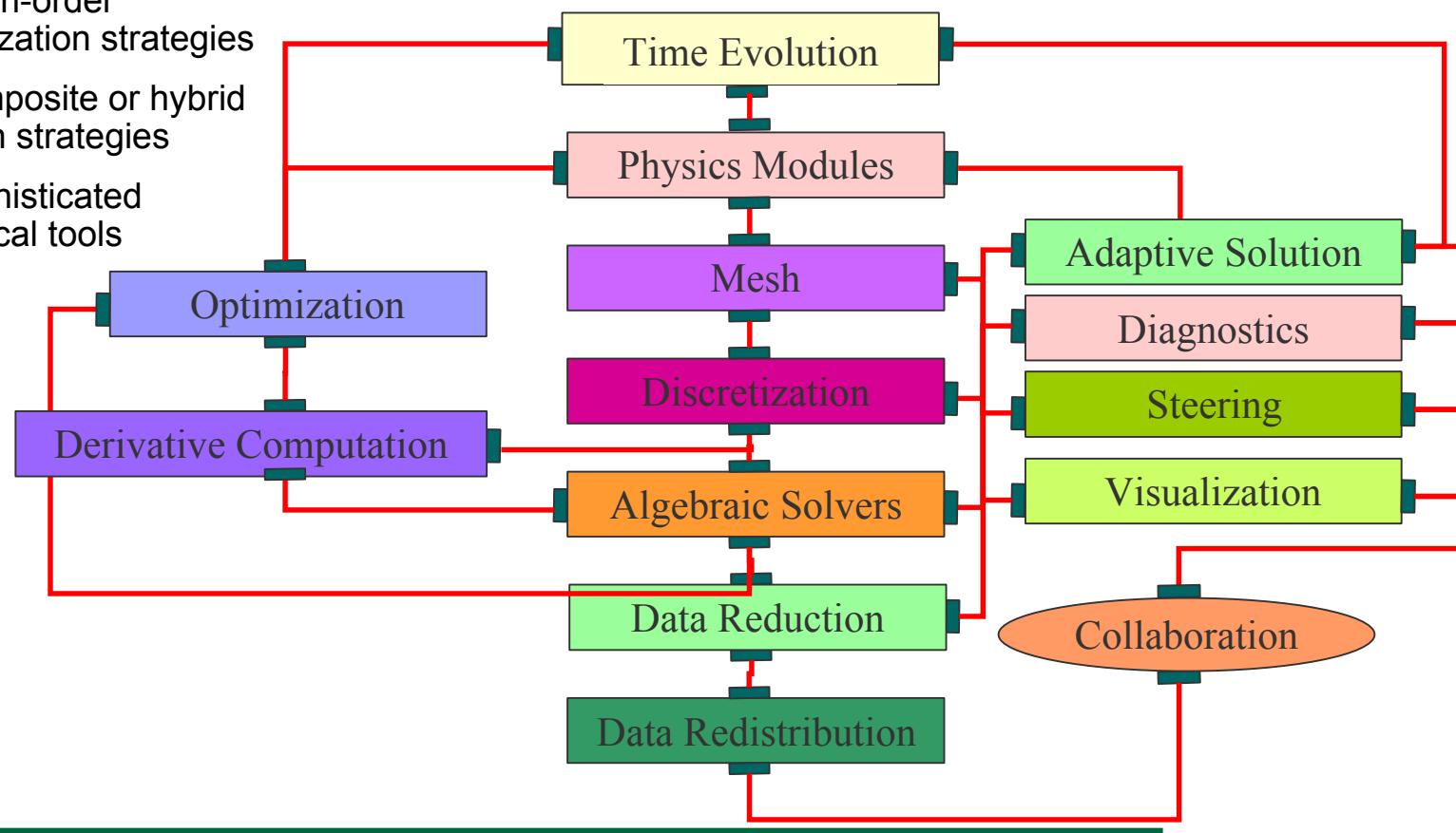
# **Enabling Scientific Applications with the Common Component Architecture**

**David E. Bernholdt**  
Oak Ridge National Laboratory

*for the CCA Forum and the CCTTSS*  
***<http://www.cca-forum.org>***

# Modern Scientific Software Development

- Terascale computing will enable high-fidelity calculations based on multiple coupled physical processes and multiple physical scales
  - Adaptive algorithms and high-order discretization strategies
  - Composite or hybrid solution strategies
  - Sophisticated numerical tools



# Component-Based Software Engineering

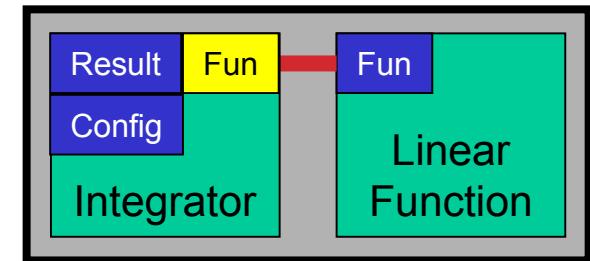
- CBSE methodology is emerging, especially popular in business and internet areas
- **Software productivity**
  - Provides a “plug and play” application development environment
  - Many components available “off the shelf”
  - Abstract interfaces facilitate reuse and interoperability of software
- *“The best software is code you don’t have to write” [Jobs]*
- **Software complexity**
  - Components encapsulate much complexity into “black boxes”
  - Plug and play approach simplifies applications & adaptation
  - Model coupling is natural in component-based approach
- **Software performance** (indirect)
  - Plug and play approach and rich “off the shelf” component library simplify changes to accommodate different platforms

# What is the CCA? (User View)

- A component model specifically designed for **high-performance scientific computing**
- Supports both **parallel and distributed** applications
- Designed to be implementable without sacrificing **performance**
- **Minimalist approach** makes it easier to componentize existing software
- A **tool** to enhance the productivity of scientific programmers
  - Make the hard things easier, make some intractable things tractable
  - Support & promote reuse & interoperability
  - **Not a magic bullet**

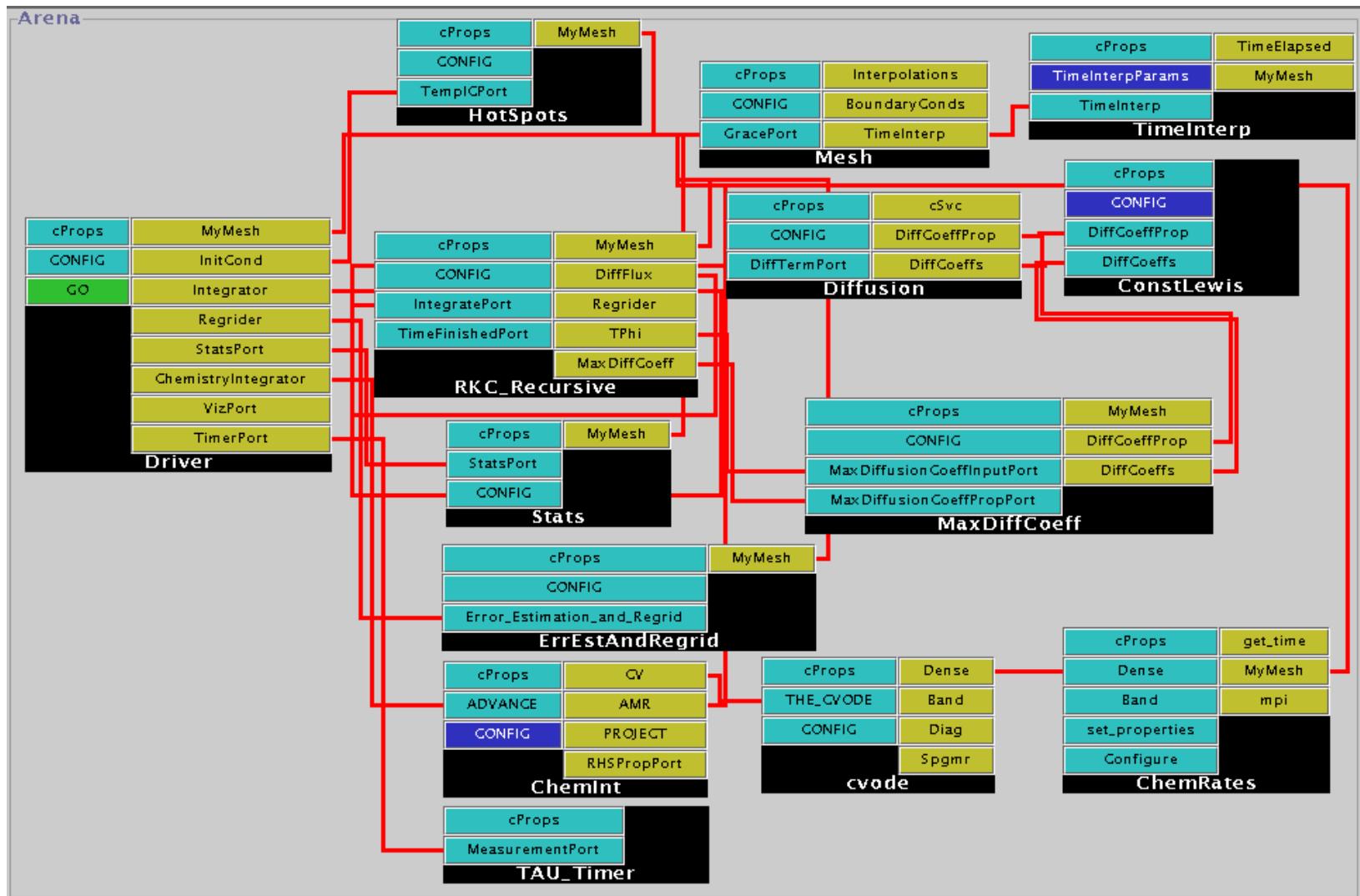
# Basic CCA Terminology

- **Port (aka *interface*)**
  - Procedural interface (not just dataflow!)
  - Like C++ abst. virtual class, Java interface
  - Provides/uses design pattern
- **Component**
  - A unit of software deployment/reuse (i.e. has interesting functionality)
  - Interacts with the outside world only through well-defined **interfaces**
  - Implementation is opaque to the outside world
  - Components are **peers**
- **Framework**
  - Holds **components** during application composition and execution
  - Controls the “**exchange**” of **interfaces** between components (while ensuring implementations remain hidden)
  - Provides a small set of standard, ubiquitous services to components
    - CCA spec doesn’t specify a *framework per se*, so components can be constructed to provide framework-like services





# “Wiring Diagram” for CFRFS App.



# CCA Supports Both Parallel and Distributed Computing

- Tightly-coupled parallel computing approach allows components to use whatever parallel programming model they prefer
  - CCA does not impose a specific approach
  - No CCA-related overhead (no framework involvement)
  - Components using different models can be mixed in a single application
- Distributed computing approach: framework handles remote method invocation

# CCA Maintains High Performance

- No CCA overhead on **calls within components**
- CCA-related overheads on **calls to other ports**
  - Invocation cost (small for direct connection)
  - Language interoperability costs (“translate” some data types)
  - **Design application architecture to minimize impact**
    - Methods in ports should do enough work to amortize overheads
    - Language costs can be minimized for most scientific computing
- No CCA overhead on **parallel interactions**
- Costs for **distributed computing** depend on network protocols, etc.

# Adapting Existing Code Into Components

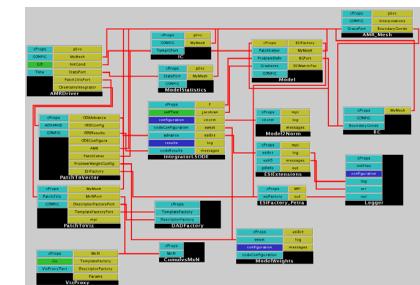
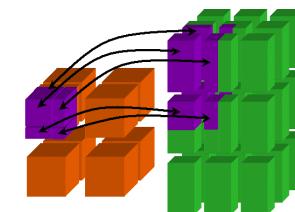
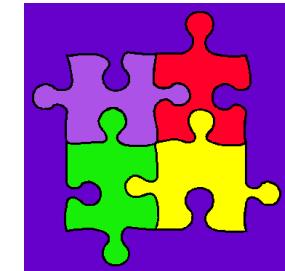
Suitably structured code (programs, libraries) should be relatively easy to adapt to the CCA. Here's how:

1. Decide **level of componentization**
  - Can evolve with time (start with coarse components, later refine into smaller ones)
2. Define **interfaces** and write wrappers between them and existing code
3. Add **framework interaction code** for each component
  - `setServices`
4. Modify component internals to use other **components** as appropriate
  - `getPort`, `releasePort` and method invocations



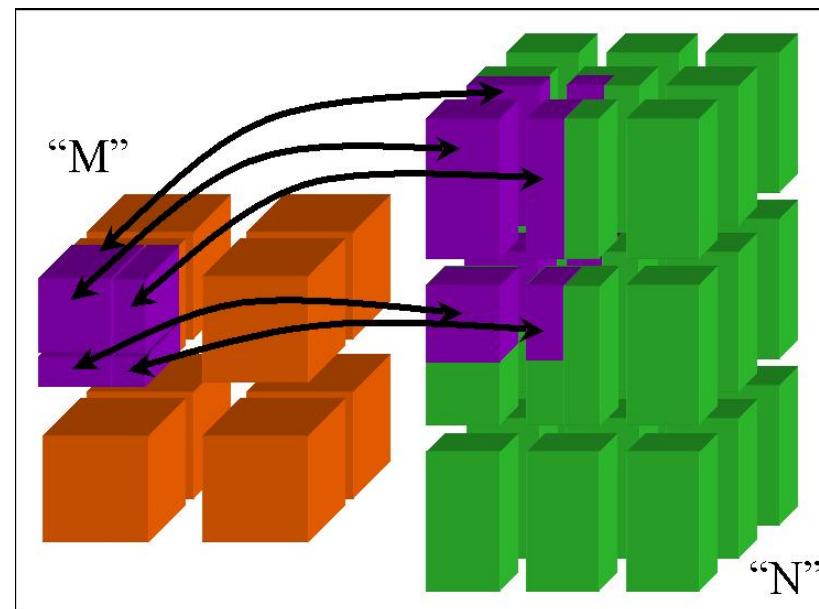
# CCA Research Thrusts

- Scientific Components
  - Scientific Data Objects  
Lois Curfman McInnes, ANL ([curfman@mcs.anl.gov](mailto:curfman@mcs.anl.gov))
- “MxN” Parallel Data Redistribution
  - Jim Kohl, ORNL ([kohlja@ornl.gov](mailto:kohlja@ornl.gov))
- Frameworks
  - Language Interoperability / Babel / SIDL
  - Component Deployment / Repository  
Gary Kumfert, LLNL ([kumfert@llnl.gov](mailto:kumfert@llnl.gov))
- User Outreach
  - David Bernholdt, ORNL ([bernholdtde@ornl.gov](mailto:bernholdtde@ornl.gov))



# MxN Parallel Data Redistribution

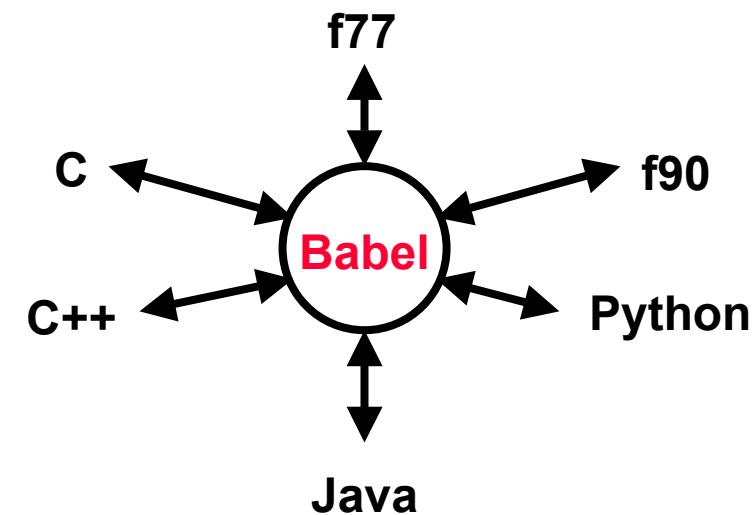
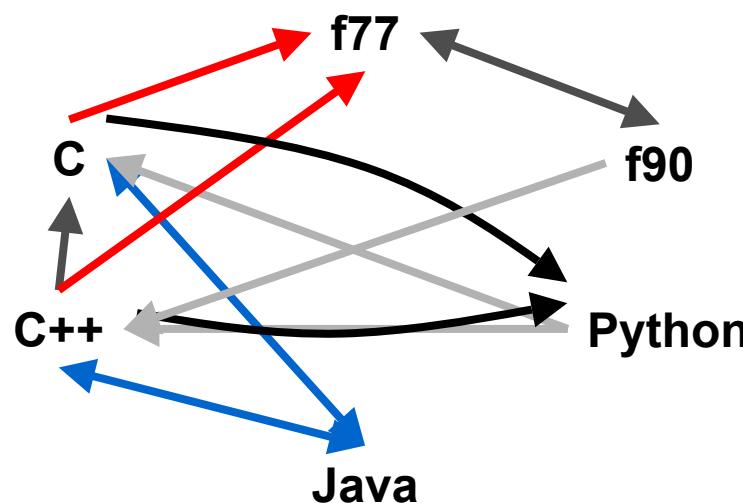
- Share Data Among Coupled Parallel Models
  - Disparate Parallel Topologies ( $M$  processes vs.  $N$ )
  - e.g. Ocean & Atmosphere, Solver & Optimizer...
  - e.g. Visualization ( $M \times 1$ , increasingly,  $M \times N$ )



*Research area -- tools under development*

# Language Interoperability

- Existing language interoperability approaches are “point-to-point” solutions
- Babel provides a unified approach in which all languages are considered peers
- Babel used primarily at interfaces

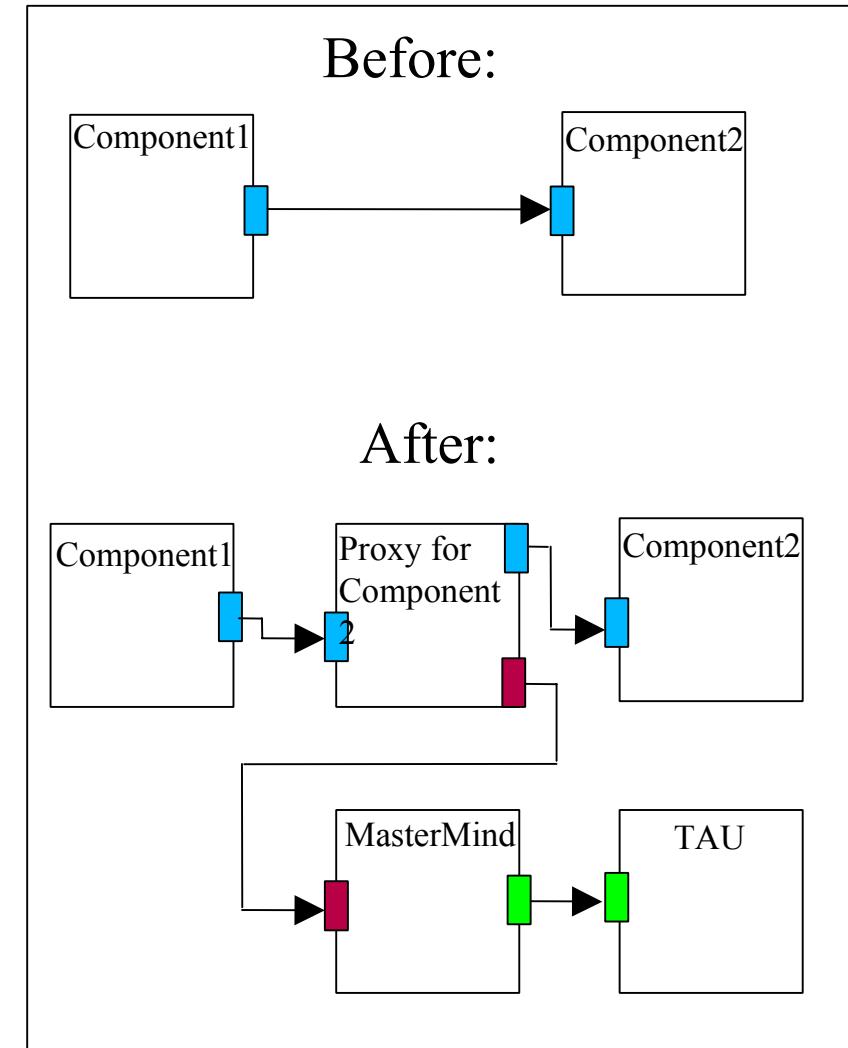




# “Integrated” Performance Measurement Capability

Measurement infrastructure:

- Proxy
  - Notifies MasterMind of all method invocations of a given component, along with performance dependent inputs
- MasterMind
  - Compiles and stores all measurement data
- TAU
  - Makes all performance measurements



# Component Inventory:

## Data Management, Meshing and Discretization

- ***Global Array Component*** – M. Krishnan and J. Nieplocha (PNNL) – classic and SIDL – Provides capabilities for manipulating multidimensional dense distributed arrays; supports DADF common interface.
- ***TSTTMesh*** – L.F. Diachin (SNL, formerly ANL) – classic – Provides prototype capabilities for querying unstructured meshes based on interfaces being designed within the TSTT SciDAC Center.
- ***FEMDiscretization*** – L.F. Diachin (SNL, formerly ANL) – classic – Provides finite element discretization of diffusion and advection PDE operators and linear system assembly capabilities.
- ***GrACEComponent*** – J. Ray (SNL) – classic – Provides parallel AMR infrastructure, which follows a hierarchy-of-patches methodology for meshing and includes load-balancing; based on GrACE (Rutgers); being used in combustion applications within the SciDAC CFRFS project.

# Component Inventory:

## Integration, Optimization, and Linear Algebra

- **CvodesComponent** – Radu Serban (LLNL, TOPS collaborator) – classic – Provides a generic implicit ODE integrator and an implicit ODE integrator with sensitivity capabilities; based on CVODES; used in combustion applications within the CFRFS.
- **TaoSolver** – S. Benson, L.C. McInnes, B. Norris, and J. Sarich (ANL) – SIDL – Provides solvers for unconstrained and bound constrained optimization problems, which build on infrastructure within TAO (ANL); uses external linear algebra capabilities.
- **LinearSolver** – B. Norris (ANL) – classic – Provides a prototype linear solver port; in the process of evolving to support common interfaces for linear algebra that are under development within the TOPS SciDAC center.

# Component Inventory:

## Parallel Data Description, Redistribution, and Visualization

- ***DistArrayDescriptorFactory*** – D. Bernholdt and W. Elwasif (ORNL) – classic – Provides a uniform means for applications to describe dense multi-dimensional arrays; based upon emerging interfaces from the CCA Scientific Data Components Working Group.
- ***CumulvsMxN*** – J. Kohl, D. Bernholdt, and T. Wilde (ORNL) – classic – Builds on CUMULVS (ORNL) technology to provide an initial implementation of parallel data redistribution interfaces that are under development by the CCA “MxN” Working Group.
- ***VizProxy*** – J. Kohl and T. Wilde (ORNL) – classic – Provides a companion “MxN” endpoint for extracting parallel data from component-based applications and then passing this data to a separate front-end viewer for graphical rendering and presentation. Variants exist for structured data and unstructured triangular mesh data as well as text-based output.

# Component Inventory:

## Services, Graphical Builders, and Performance

- **Ccaffeine Services** – B. Allan, R. Armstrong, M. Govindaraju, S. Lefantzi, and E. Walsh (SNL) – classic and SIDL – Services for parameter ports, connections between SIDL and classic ports, MPI access, connection events, etc.
- **Graphical Builders** – B. Norris (ANL) and S. Parker (Univ. of Utah) – Prototype graphical builders that can be used to assemble components, set parameters, execute, and monitor component-based simulations.
- **Performance Observation** – S. Shende and A. Malony (Univ. of Oregon), C. Rasmussen and M. Sotille (LANL), and J. Ray (SNL) – classic and SIDL – Provides measurement capabilities to components, thereby aiding in the selection of components and helping to create performance aware intelligent components; based on TAU (Oregon).

# Current CCA Application Areas

## SciDAC:

- Combustion (CFRFS)
- Climate Modeling (CCSM)
- Meshing Tools (TSTT)
- (PDE) Solvers (TOPS)
- IO, Poisson Solvers (APDEC)
- Fusion (CMRS)
- Supernova simulation (TSI)
- Accelerator simulation (ACCAST)

## DOE Outside of SciDAC:

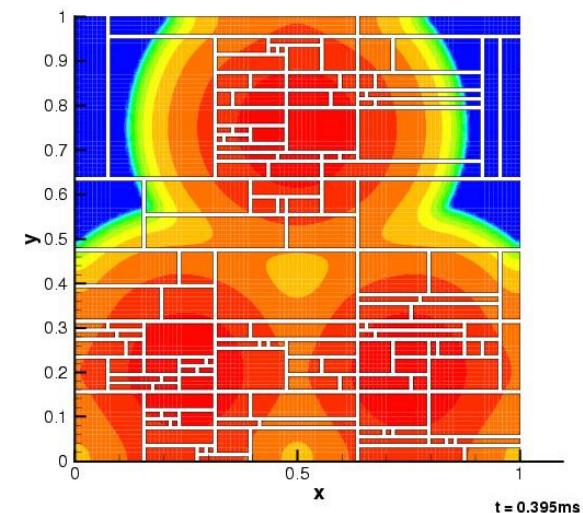
- ASCI: C-SAFE, Views, Data Svc's
- Quantum Chemistry
- Materials Science (ORNL LDRD)
- Fusion (ORNL LDRD)

## Outside of DOE:

- NASA: ESMF, SWMF
- Etc....

# Computational Facility for Reacting Flow Science (CFRFS)

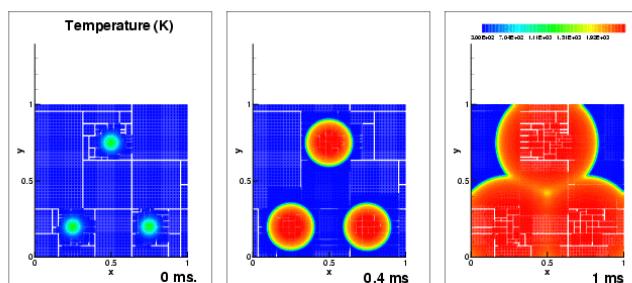
- SciDAC BES project, H. Najm PI
- **Investigators:** Sofia Lefantzi (SNL), Jaideep Ray (SNL), Sameer Shende (Oregon)
- **Goal:** A “plug-and-play” toolkit environment for flame simulations
- **Problem Domain:** Structured adaptive mesh refinement solutions to reaction-diffusion problems





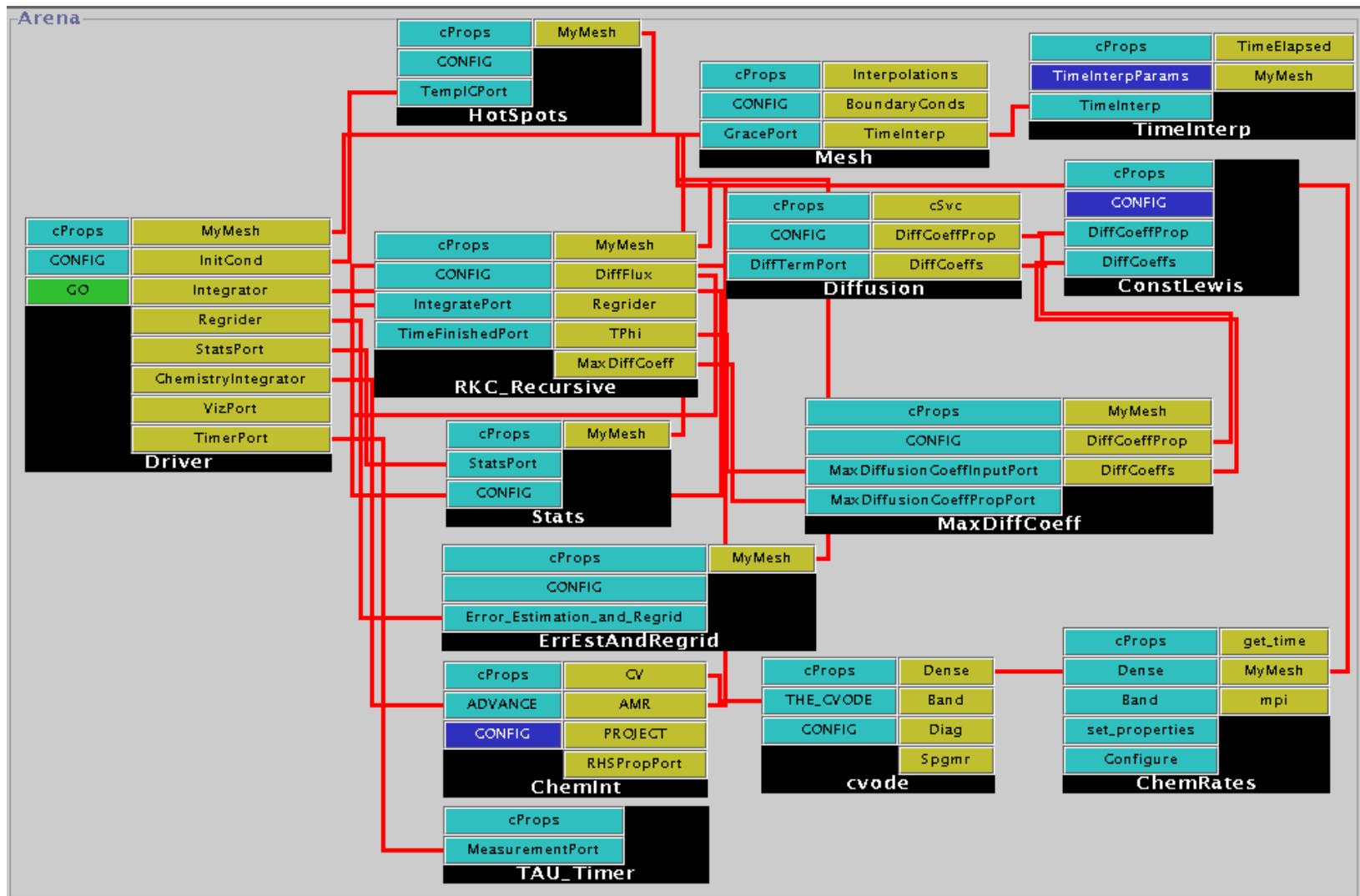
# Scientific and Technical Summary

- H<sub>2</sub>-Air ignition on a structured adaptive mesh, with an operator-split formulation
- RKC for non-stiff terms, BDF for stiff
- 9-species, 19-reactions, stiff mechanism
- 1cm x 1cm domain; max resolution = 12.5 microns
- Kernel for a 3D, adaptive mesh low Mach number flame simulation capability in SNL, Livermore
- Components are usually in C++ or wrappers around old F77 code
- Developed numerous components
  - Integrator, spatial discretizations, chemical rates evalutator, transport property models, timers etc.
  - Structured adaptive mesh, load-balancers, error-estimators (for refining/coarsening)
  - In-core, off-machine, data transfers for post-processing
- TAU for timing (Oregon, PERC)
- CVODES integrator (LLNL, TOPS)



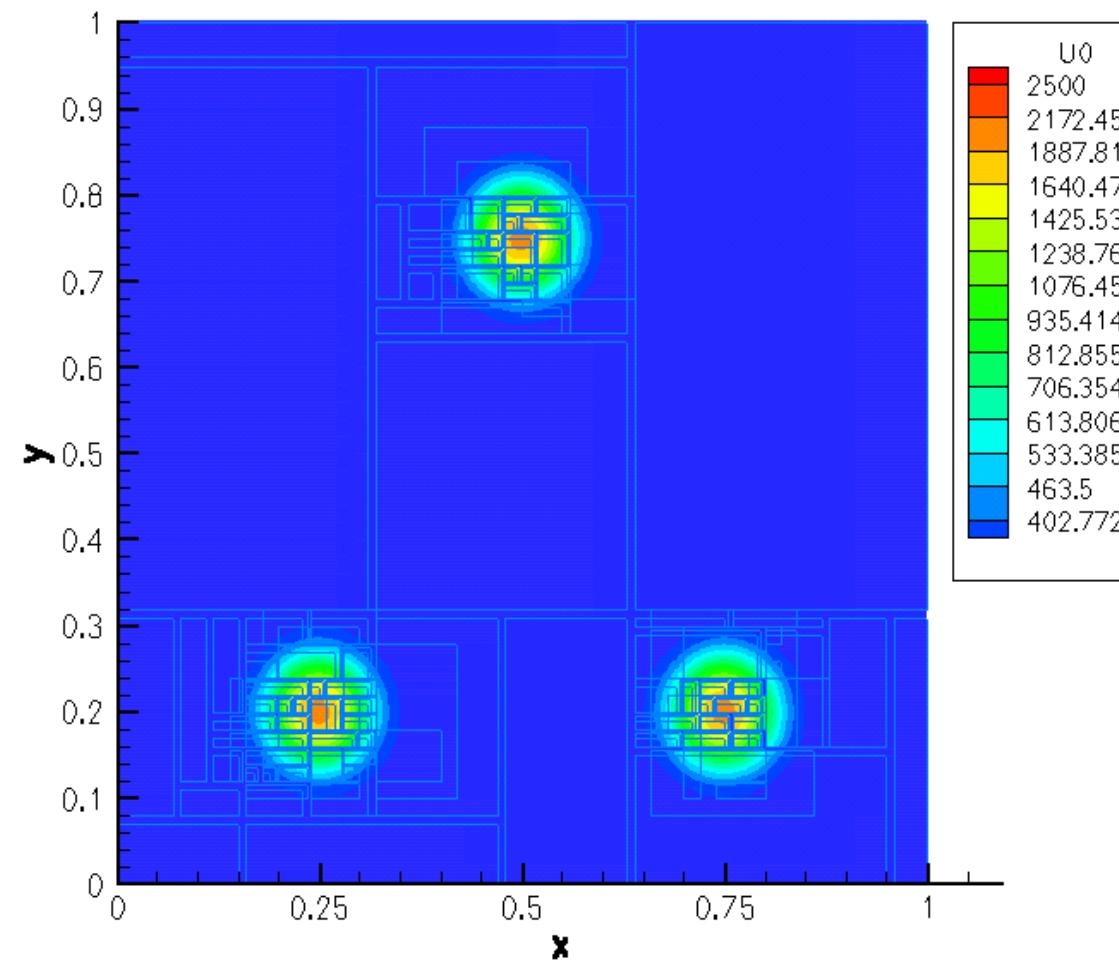


# “Wiring Diagram” for CFRFS App.





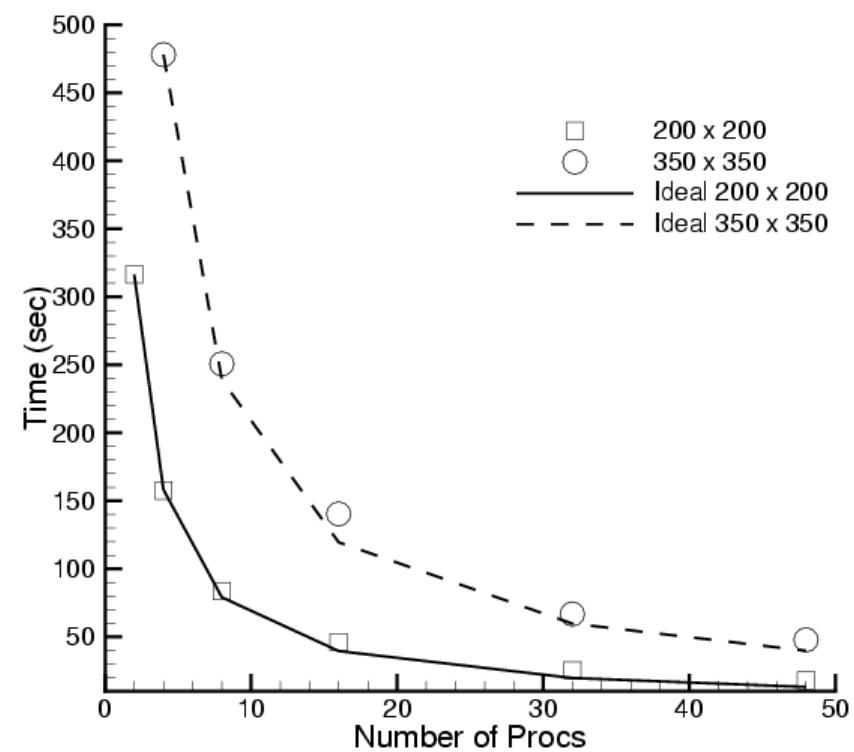
# Animation of the Temperature Field





# Scalability of Scientific Data Components in CFRFS Combustion Applications

- Investigators: S. Lefantzi, J. Ray, and H. Najm (SNL)
- Uses GRACEComponent, CvodesComponent, etc.
- Shock-hydro code with no refinement
- 200 x 200 & 350 x 350 meshes
- Cplant cluster
  - 400 MHz EV5 Alphas
  - 1 Gb/s Myrinet
- Negligible component overhead
- Worst perf : 73% scaling efficiency for 200x200 mesh on 48 procs



Reference: S. Lefantzi, J. Ray, and H. Najm, Using the Common Component Architecture to Design High Performance Scientific Simulation Codes, *Proc of Int. Parallel and Distributed Processing Symposium*, Nice, France, 2003, accepted.

# Interoperability of the CFRFS and APDEC Adaptive Mesh Refinement Toolkits

**Investigators:** Jaideep Ray (SNL), Brian van Straalen (LBL), and Phil Colella (LBL)

## Goals

- Enable the AMR toolkits from CFRFS and APDEC to interoperate (exchange / reuse modules)
  - Exchange solvers
  - Reuse parallel data I/O tools developed by APDEC



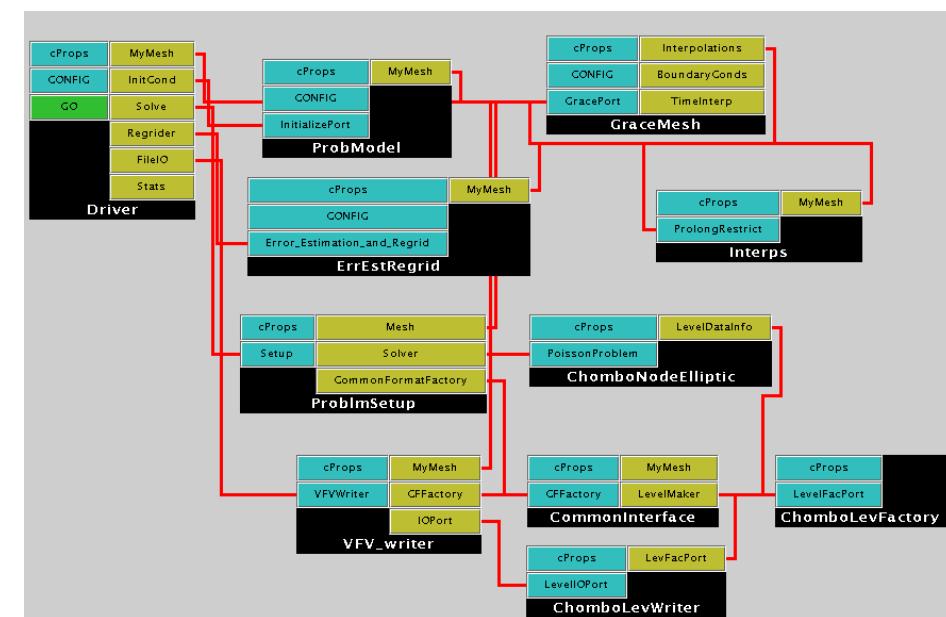
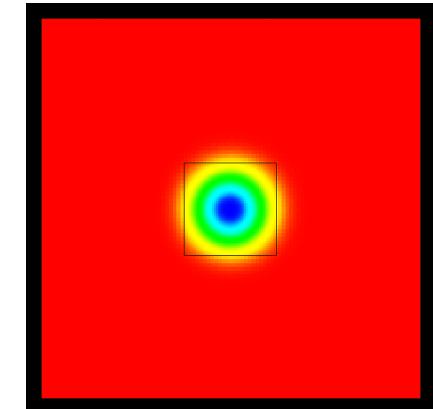
# Approach

## Simple Problem: Solve a Poisson problem and visualize it

- Initialize data and the discretized domain using CFRFS components
- Make both accessible/mutable from the common interface
- Pose the problem to an APDEC solver (available in the Chombo package)
  - The APDEC solver creates its own data structures using the agreed-to interface
  - Solves the problem
  - Writes the solution to source via the agreed-to interface
- Use APDEC's HDF5 writer component to save the files
- Use ChomboVis to visualize

$$\nabla^2 \Phi = R$$

Solution



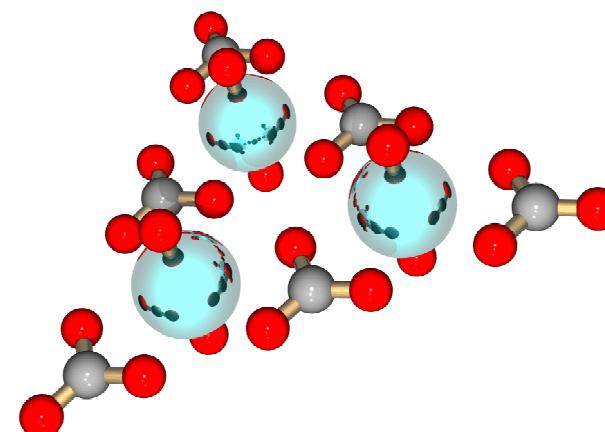
# Component-Based Integration of Chemistry and Optimization Packages: **Molecular Geometry Optimization**

Yuri Alexeev, Manoj Krishnan, Jarek Nieplocha, Theresa Windus (PNNL)

Curtis Janssen, Joseph Kenny (SNL)

Steve Benson, Lois McInnes, Jason Sarich (ANL)

David Bernholdt (ORNL)

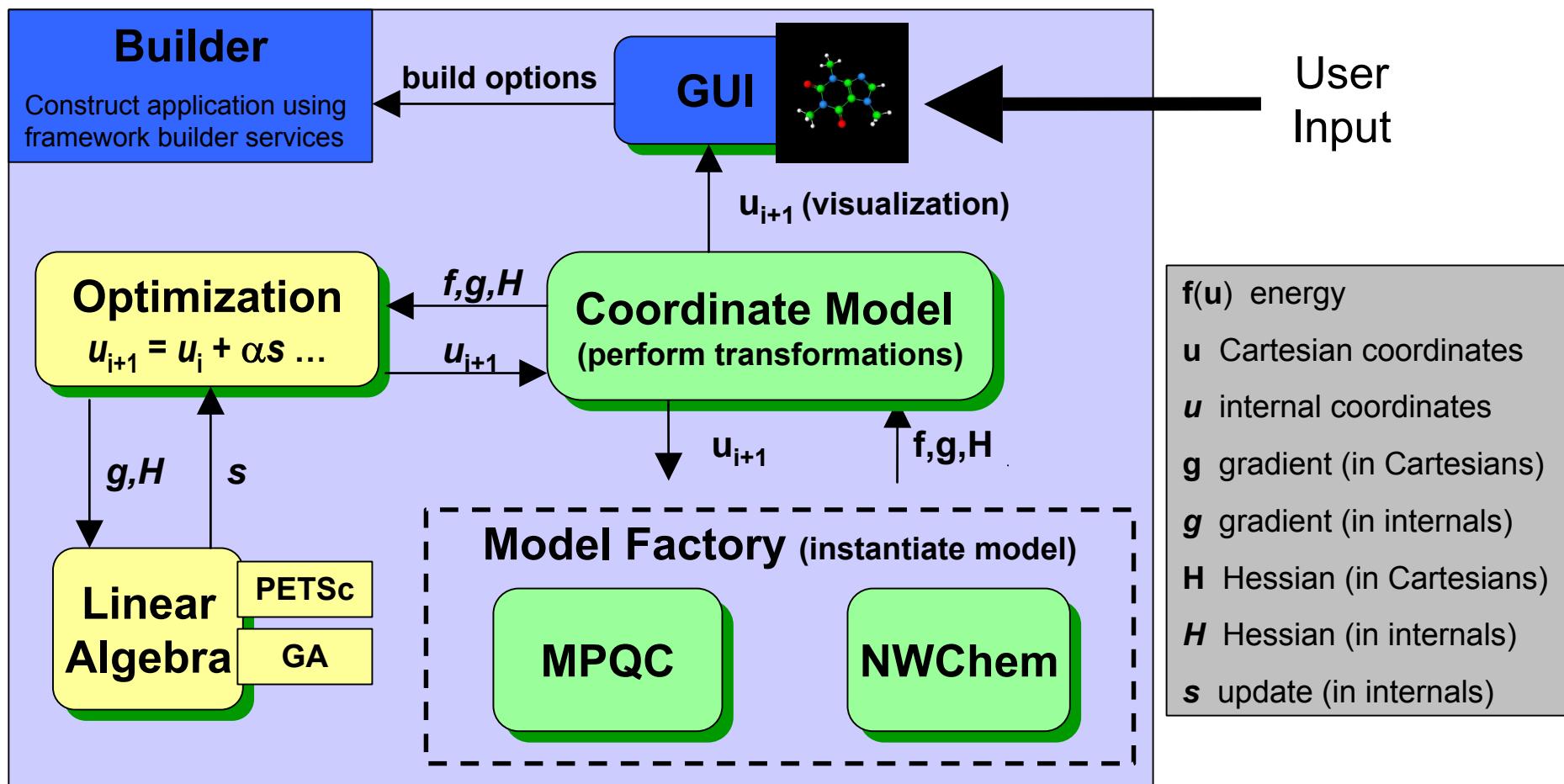


# CCA Quantum Chemistry Project Overview

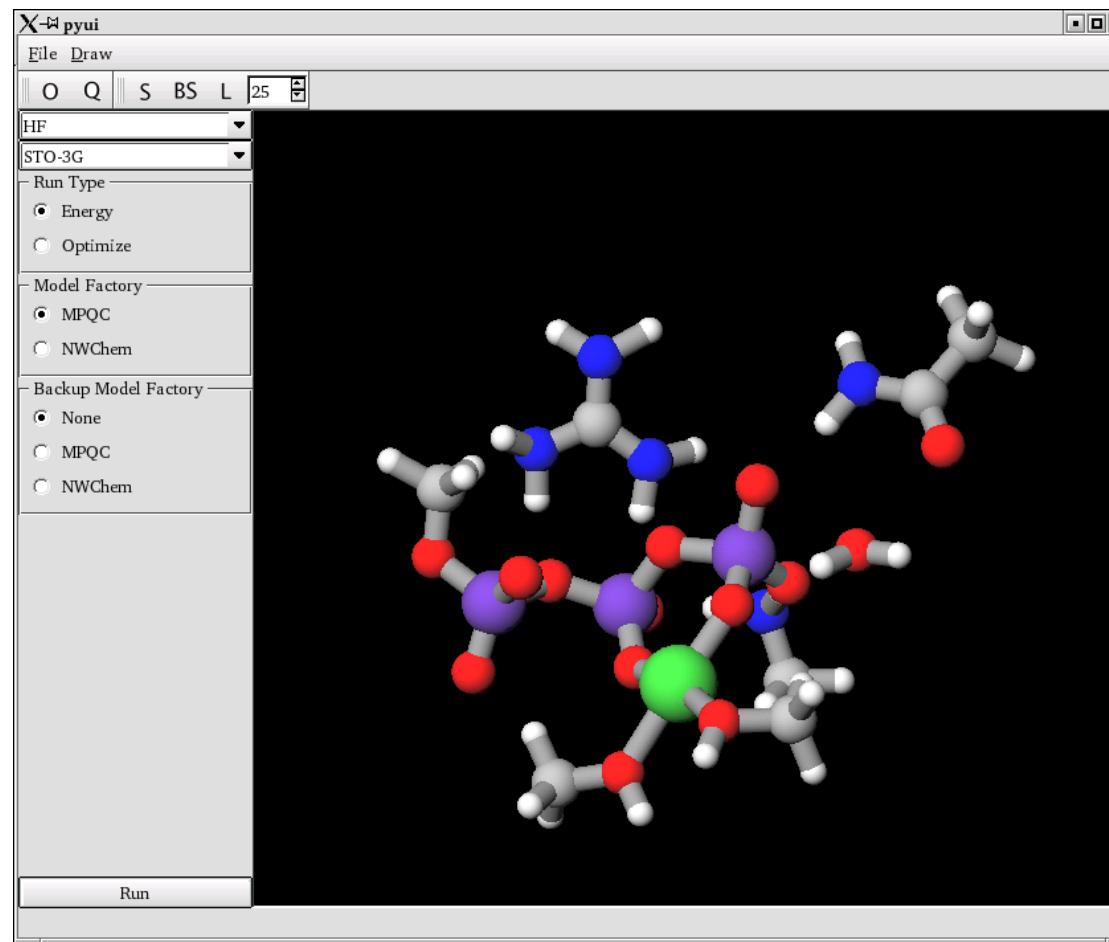
- **Previous Work:** Demonstrated interoperability of NWChem and MPQC chemistry packages with Toolkit for Advanced Optimization (TAO)
- **Current Focus:** Moving from “proof of concept” stage toward real end-user applications
  - **Performance evaluation** of optimization components
    - Examine efficiency of algorithms provided by TAO (ANL) for optimization of quantum chemistry models based on NWChem (PNNL) and MPQC (SNL) with core linear algebra support from Global Arrays (PNNL) and PETSc (ANL)
  - **Further development** of optimization capabilities
    - Develop interfaces and implementations providing internal coordinate generation, constrained optimization, and configurable convergence checking
  - **Graphical User Interface** to assemble and run applications
    - Provide user-friendly front-end and visualization
    - Provide feedback for framework builder services development
- **Future plans:** Exploring chemistry package integration through hybrid calculation schemes and sharing of lower-level intermediates such as integrals and wavefunctions

# Molecular Geometry Optimization

Compute the molecular geometry with minimum energy, i.e. solve  $\min f(u)$ , where  $f: R_n \Rightarrow R$ .



# PyView: Graphical User Interface



- A CCA component
- Driver for chemistry application builder

# Preliminary Performance Evaluation

Chemical System	Statistic	NWChem	NWChem/TAO
<b>4-waters</b> (52 basis functions)	Iterations Energy Eval Gradient Eval <b>Time (sec)</b>	105 202 105 <b>915</b>	132 139 139 <b>881</b>
<b>h3po4-water</b> (68 basis functions)	Iterations Energy Eval Gradient Eval <b>Time (sec)</b>	130 246 130 <b>4283</b>	147 160 160 <b>3831</b>
<b>isoprene</b> (65 basis functions)	Iterations Energy Eval Gradient Eval <b>Time (sec)</b>	63 124 63 <b>2007</b>	65 68 68 <b>1560</b>
<b>glycine</b> (55 basis functions)	Iterations Energy Eval Gradient Eval <b>Time (sec)</b>	51 95 51 <b>1109</b>	92 98 98 <b>1625</b>

# Applications: Climate Modeling

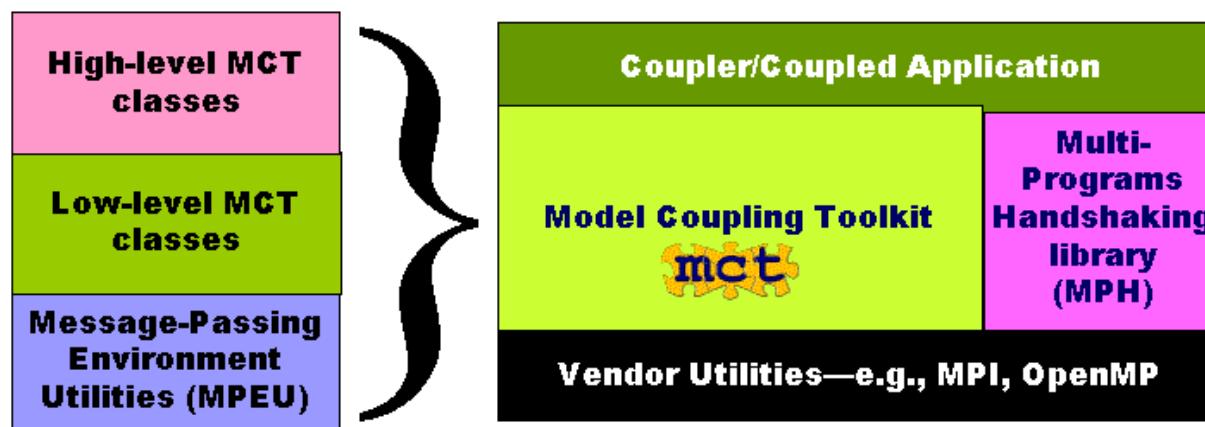
## Community Climate System Model

- SciDAC BER project, John Drake and Robert Malone PIs
- **Goals:** Investigate model coupling and parameterization-level componentization within models
- **Investigators:** John Drake (ORNL), Wael Elwasif (ORNL), Michael Ham (ORNL), Jay Larson (ANL), Everest Ong (ANL), Nancy Collins (NCAR), Craig Rasmussen (LANL)

## Earth System Modeling Framework

- NASA project, Tim Killeen, John Marshall, and Arlindo da Silva PIs
- **Goal:** Build domain-specific framework for the development of climate models

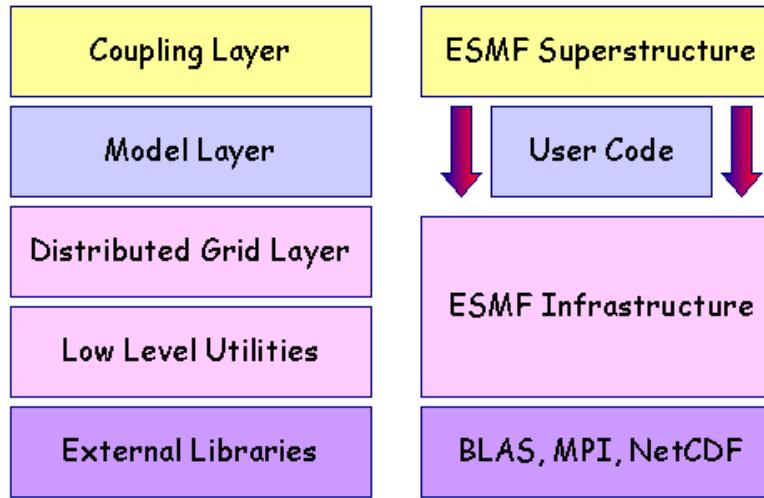
# Community Climate System Model



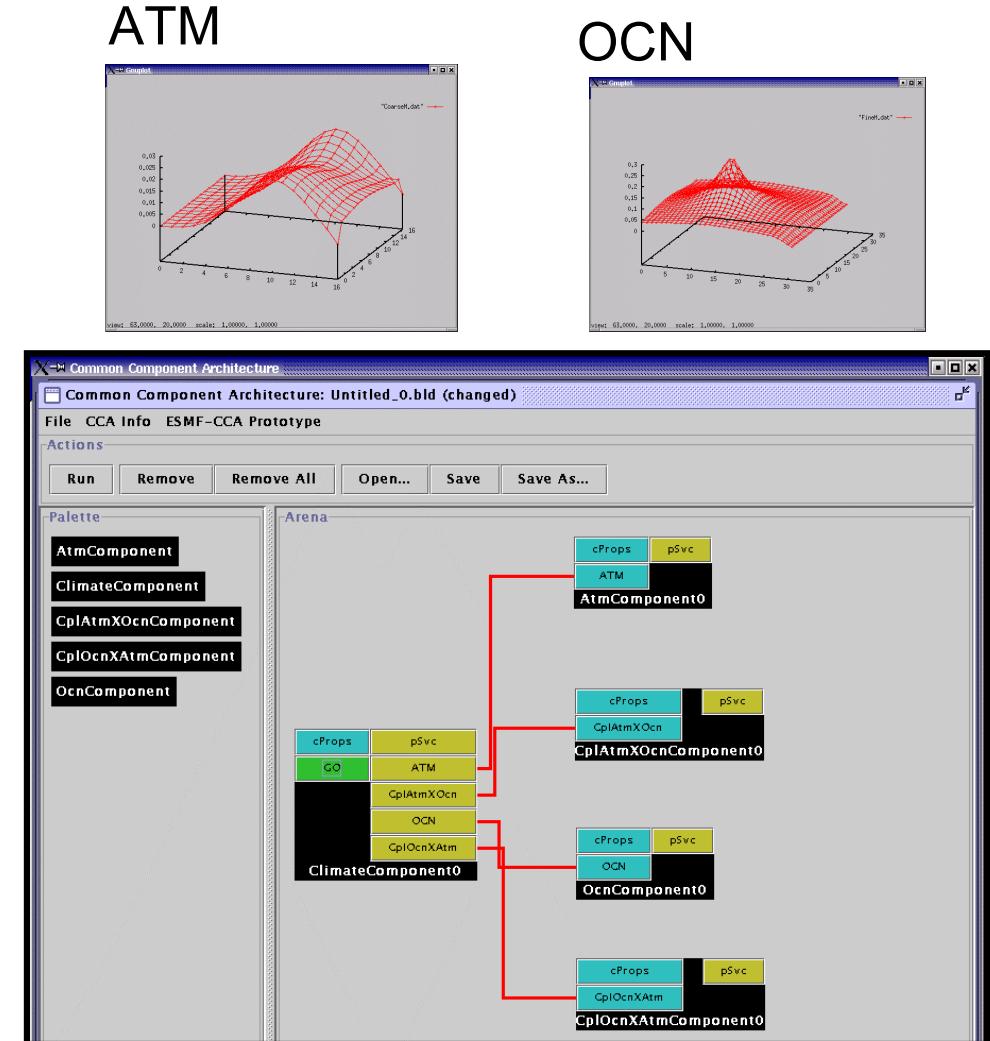
- Model Coupling Toolkit (MCT)
  - Coupler for CCSM
  - Basis for ESMF coupler
  - Contributions to MxN
  - River runoff model
- Community Atmosphere Model (CAM)
  - Componentization at physics/dynamics interface



# Earth System Modeling Framework



- Prototype superstructure
- Investigating grid layer interfaces



Courtesy Shujia Zhou, NASA Goddard

# Information Pointers & Acknowledgements

- <http://www.cca-forum.org>
  - Tutorial presentations, software, etc. available
- SC2003 Exhibit Hall
  - SciDAC    NASA                      PNNL
  - ORNL       NCAR                      Research in Indiana
  - ANL          NCSA/Alliance           University of Utah
  - LANL        NNCSA/ASCI
- Thanks to the *many* people who have contributed to the development and use of the CCA, who's work this talk represents!

Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, U.S. Dept. of Energy. Oak Ridge National Laboratory is managed by UT-Battelle, LLC for the US Dept. of Energy under contract DE-AC-05-00OR22725