

Increasing the Productivity of Computational Scientists

David E. Bernholdt

Computer Science and Mathematics
Oak Ridge National Laboratory

bernholdtde@ornl.gov

Research supported by the Mathematics, Information and Computational Sciences
Office, Office of Science, U.S. Dept. of Energy, and the Laboratory Directed
Research and Development Program of Oak Ridge National Laboratory

The Vicious Cycle of Computational Science

- As computers grow more powerful, computational simulations provide **better results**
 - More accurate results for more realistic models
- Better results **increase the demand** for computational simulation
 - Must model larger systems with increased fidelity
- More demanding and larger simulations require **more powerful computers**
 - Iterate from 1945 – 2004 and beyond
- **Mostly a *good* thing. Except...**

Complexity: The Price of Success?

- As models increase fidelity, software implementing them gets **more complex**
- As problems get larger, **more complex** software is required to implement them efficiently
- As computers increase in capability, they become **more complex**
 - Hardware complexity is exposed to the programmer
 - Must be managed by the software to obtain the best performance
- Larger, more complex simulations produce larger and **more complex** results, which must be analyzed and understood to advance science

Complexity Reduces Productivity

- Complex software takes longer to develop
- Complex software takes longer to optimize
- Larger and more complex results take more time to manage and analyze
- ... so computational scientists spend less time “doing science”

Three Approaches to Raising Productivity

- **The Common Component Architecture (CCA)**
 - Improving the scientific software development process through the use of component-based software engineering
- **The Tensor Contraction Engine (TCE)**
 - A high-level language and optimizing compiler specialized to the needs of a class of applications in quantum chemistry and physics
- **The Earth System Grid (ESG)**
 - Providing efficient management and community access to terabyte-scale climate simulation data

The Common Component Architecture

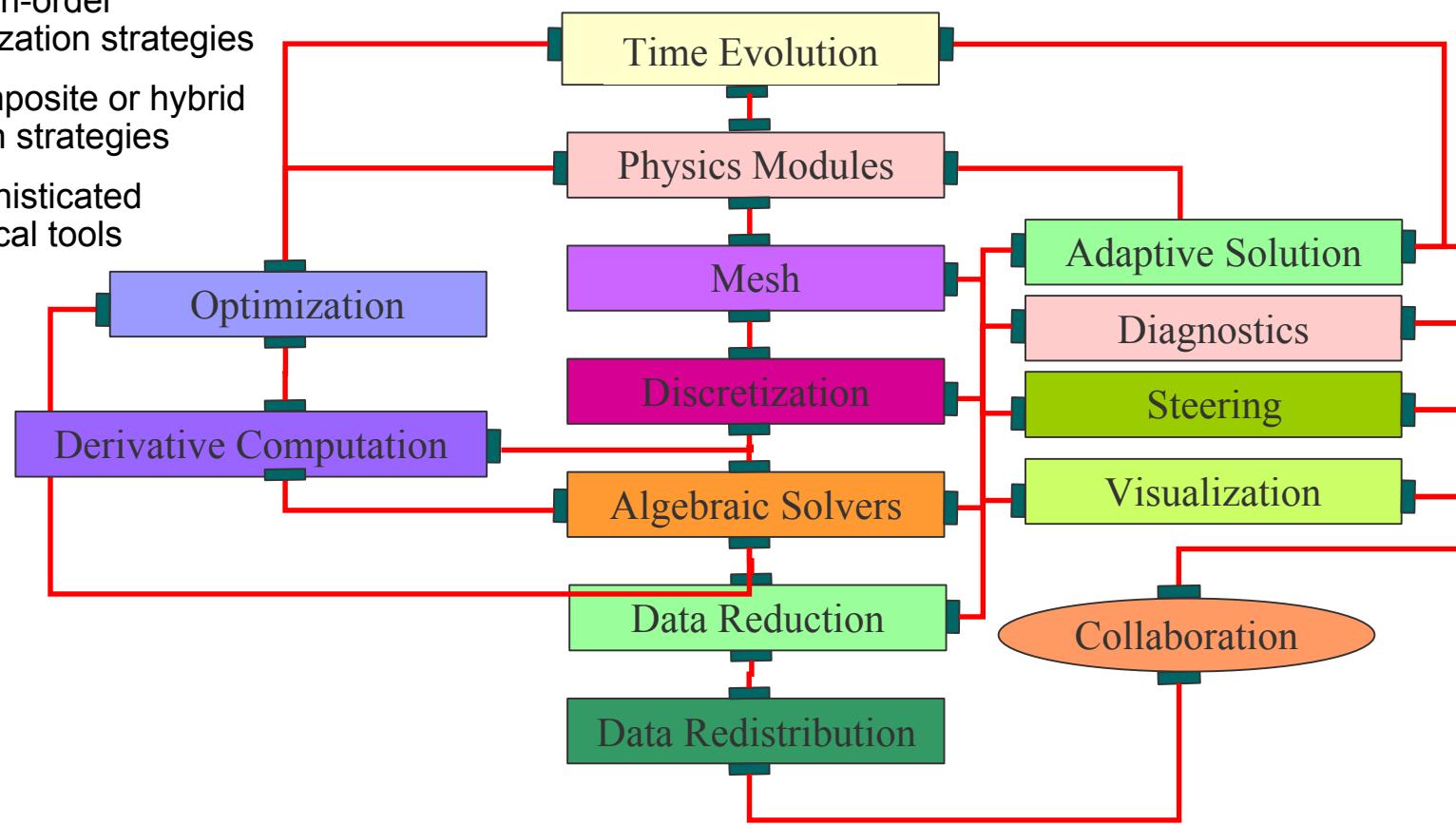


work by members of the
Center for Component Technology for
Terascale Simulation Software
(ANL, Indiana, LANL, LLNL, ORNL,
PNNL, SNL, Utah)
and the
CCA Forum

<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 (and successful) 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

CCA Compared to Other Component Models

- A component model specifically designed for **high-performance scientific computing**
 - “Commodity” component models (i.e. CORBA, COM, EJB) aren’t adequate
- Supports both **parallel and distributed** applications
 - Commodity models focus on distributed only
- Designed to be implementable without sacrificing **performance**
 - Distributed timescales $O(ms\text{-}s)$; parallel timescale $O(ns\text{-}\mu s)$
- **Minimalist approach** makes it easier to componentize existing software
 - CCA requires just one new method, minimal modifications elsewhere to be a useful component

Components Compared to Traditional Scientific Programming

- Components are typically discussed as objects or collections of objects
 - Interfaces generally designed in OO terms, but...
 - Component internals need not be OO
 - OO languages are not required
- Component environments can enforce the use of published interfaces (prevent access to internals)
 - Libraries can not
- Libraries are often hard to compose together
- Components *must* include some code to interface with the framework/component environment
 - Libraries and objects do not
- Components only exist within a component environment (framework)

Domain-Specific Frameworks vs Generic Component Architectures

Domain-Specific

- Often known as “frameworks”
- Provide a significant software infrastructure to support applications in a **given domain**
 - Often attempts to generalize an existing large application
- Often hard to adapt to use outside the original domain
 - Tend to assume a **particular structure/workflow** for application
- Relatively **common**

Generic

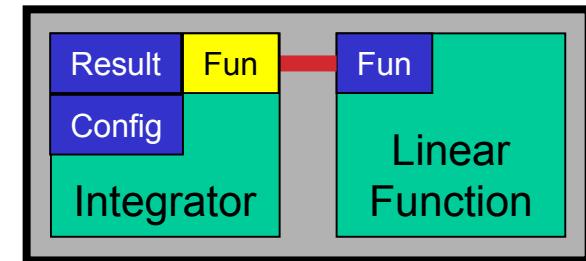
- Provide the infrastructure to **hook components** together
 - Domain-specific infrastructure can be built as components
- Usable in **many domains**
 - Few assumptions about application
 - **More opportunities for reuse**
- Better supports **model coupling** across traditional domain boundaries
- Relatively **rare** at present
 - Commodity component models often not so useful in HPC scientific context

Key Features of the CCA

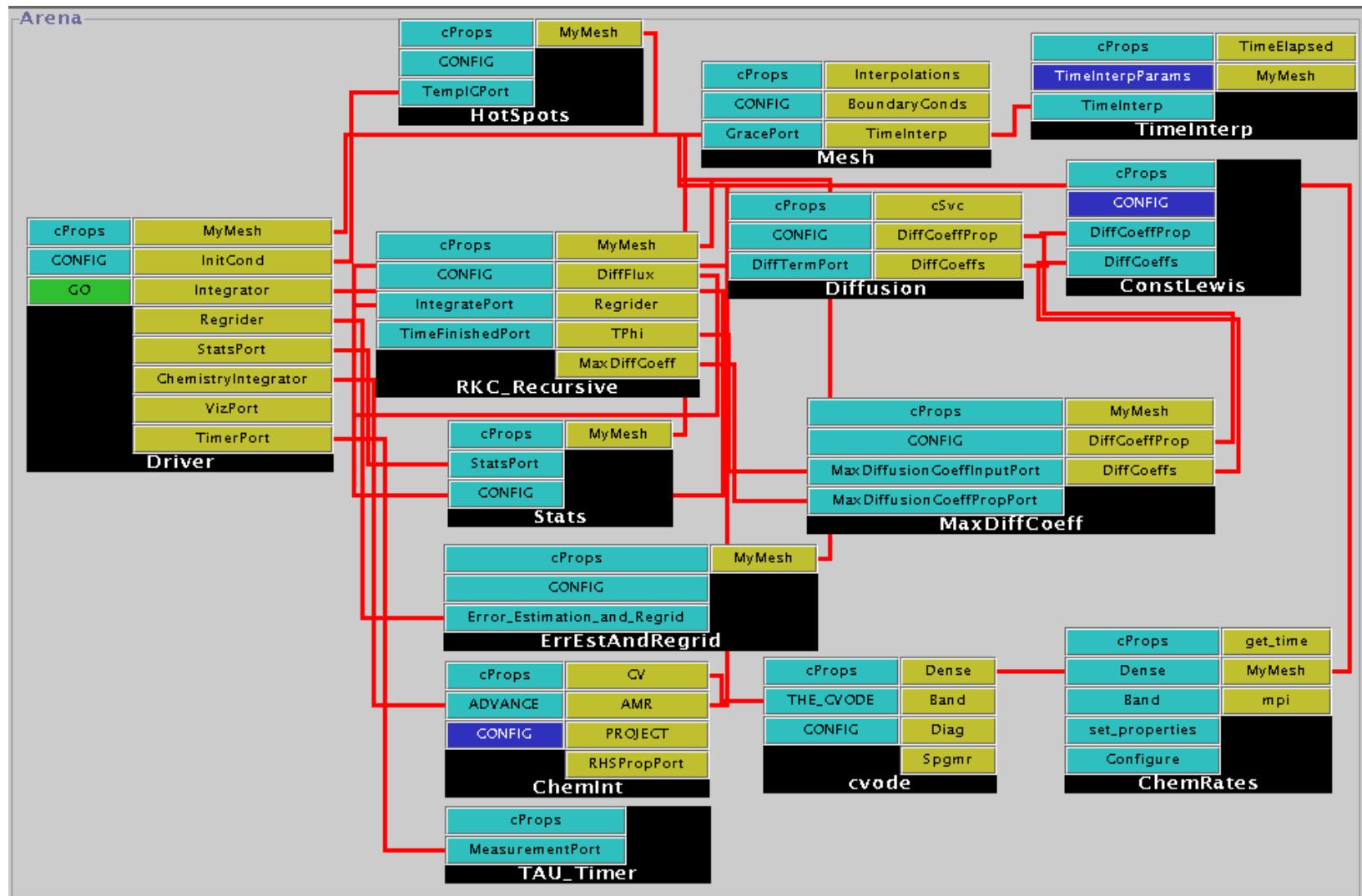
- Visual “programming” paradigm for application assembly
- Frameworks maintain performance of local and parallel components
- Language independence for components
- Express interfaces independent of implementation
- A *tool* to enhance the productivity of scientific programmers
 - Make the hard things 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.

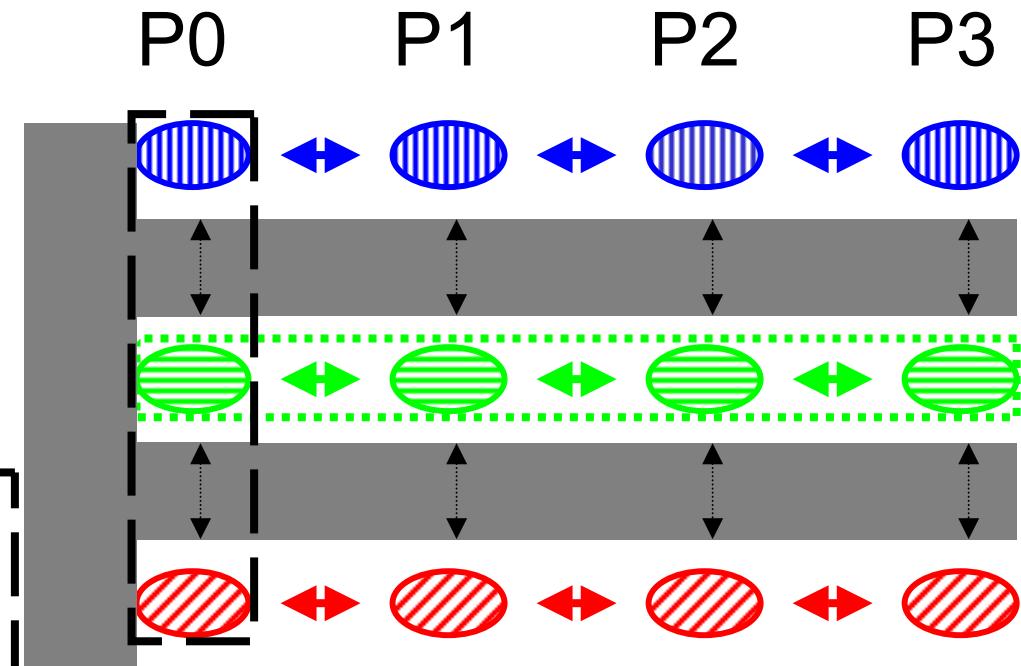


High Performance and Parallelism

- Single component multiple data (SCMD) model is component analog of widely used SPMD model
- Each process loaded with the same set of components wired the same way

- Different components in same process “talk to each” other via ports and the framework

Same component in different processes talk to each other through their favorite communications layer (i.e. MPI, PVM, GA)



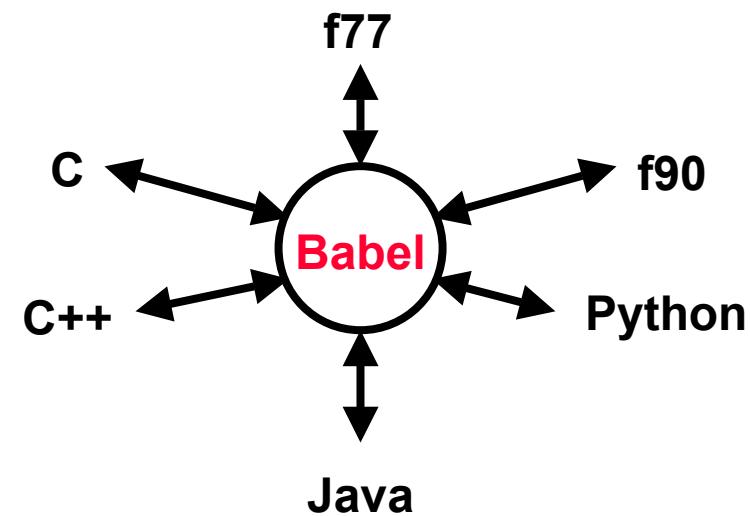
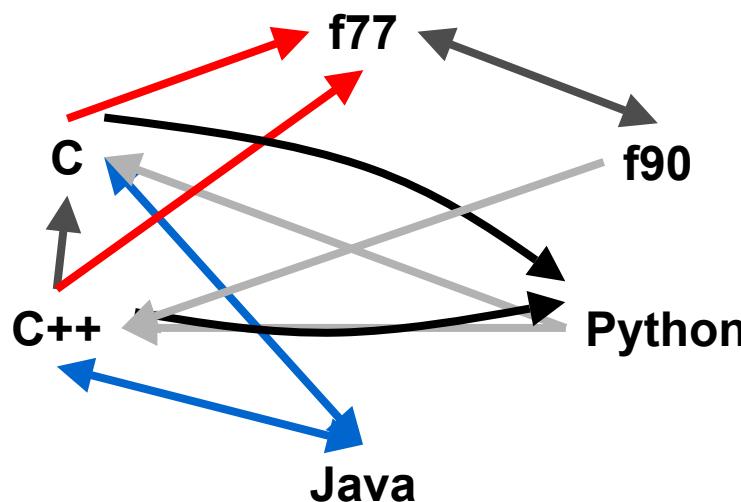
Components: Blue, Green, Red

Framework: Gray

MCMD/MPMD also supported
Other component models
ignore parallelism entirely

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

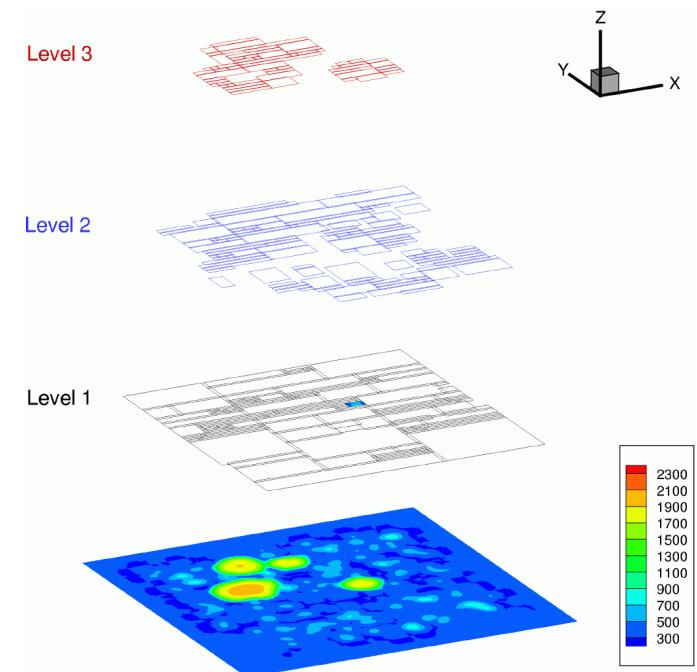


CCA Application Areas & Component Infrastructure

- Combustion
- Global Climate Modeling
- Quantum Chemistry
- Fusion
- Materials Science & Nanoscience
- Underground radionuclide transport
- Scientific Data Management
- Large-Scale Visualization
- Biomedical Engineering
- Distributed Arrays and basic parallel linear algebra
- Parallel Data Redistribution
- Meshing and discretization
- PDE Solvers
- ODE Integrators
- Optimization
- Structured Adaptive Mesh Refinement
- Performance Observation

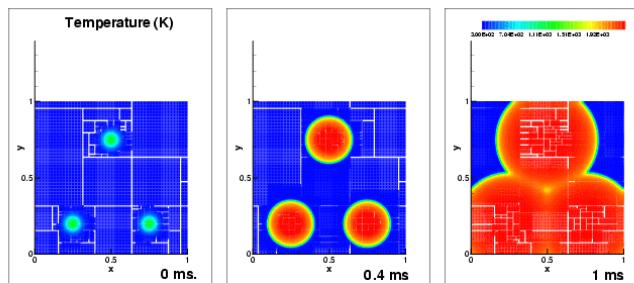
Computational Facility for Reacting Flow Science (CFRFS)

- 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

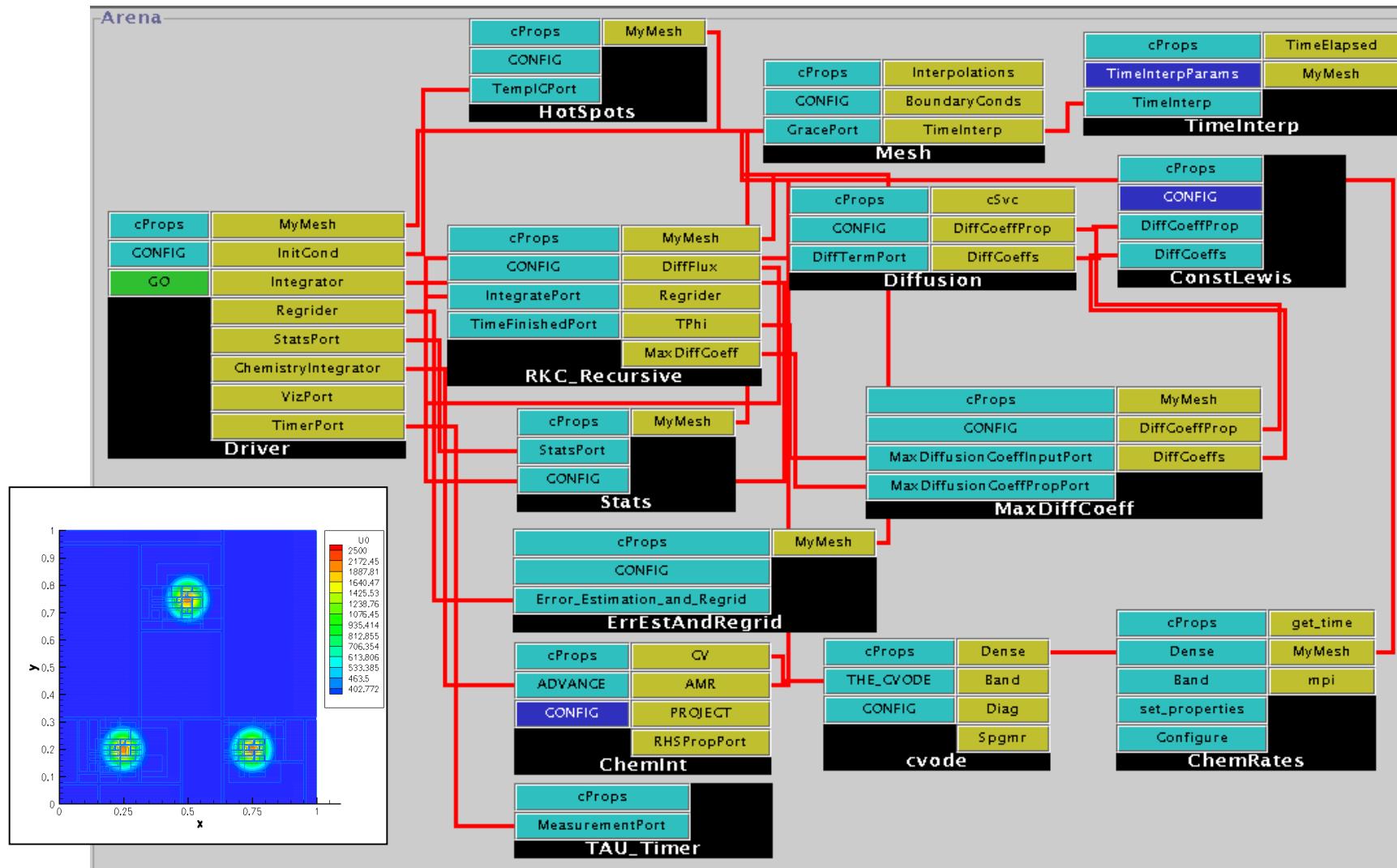


Scientific and Technical Summary

- H₂-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
- 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)
- CVODES integrator (LLNL)



“Wiring Diagram” for CFRFS App.



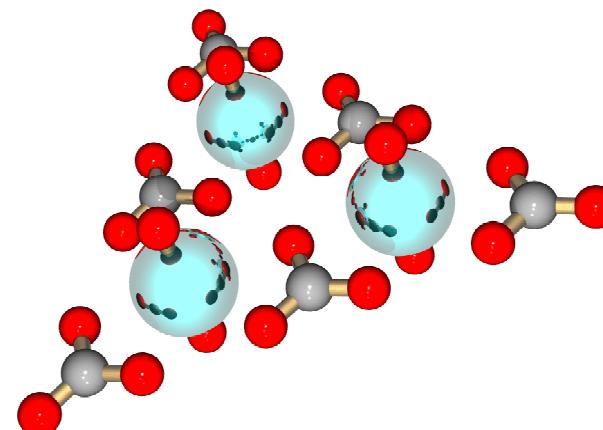
Component-Based Integration of Chemistry and Optimization Packages: Molecular Geometry Optimization

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

Curtis Janssen, Joseph Kenny (SNL)

Steve Benson, Lois McInnes, Jason Sarich (ANL)

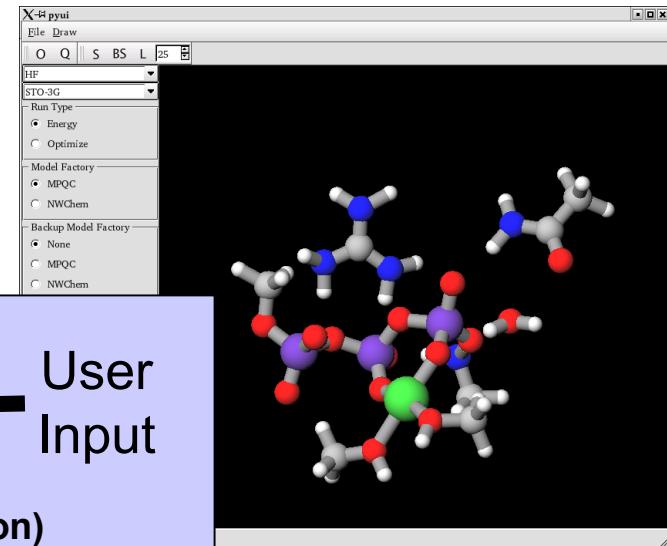
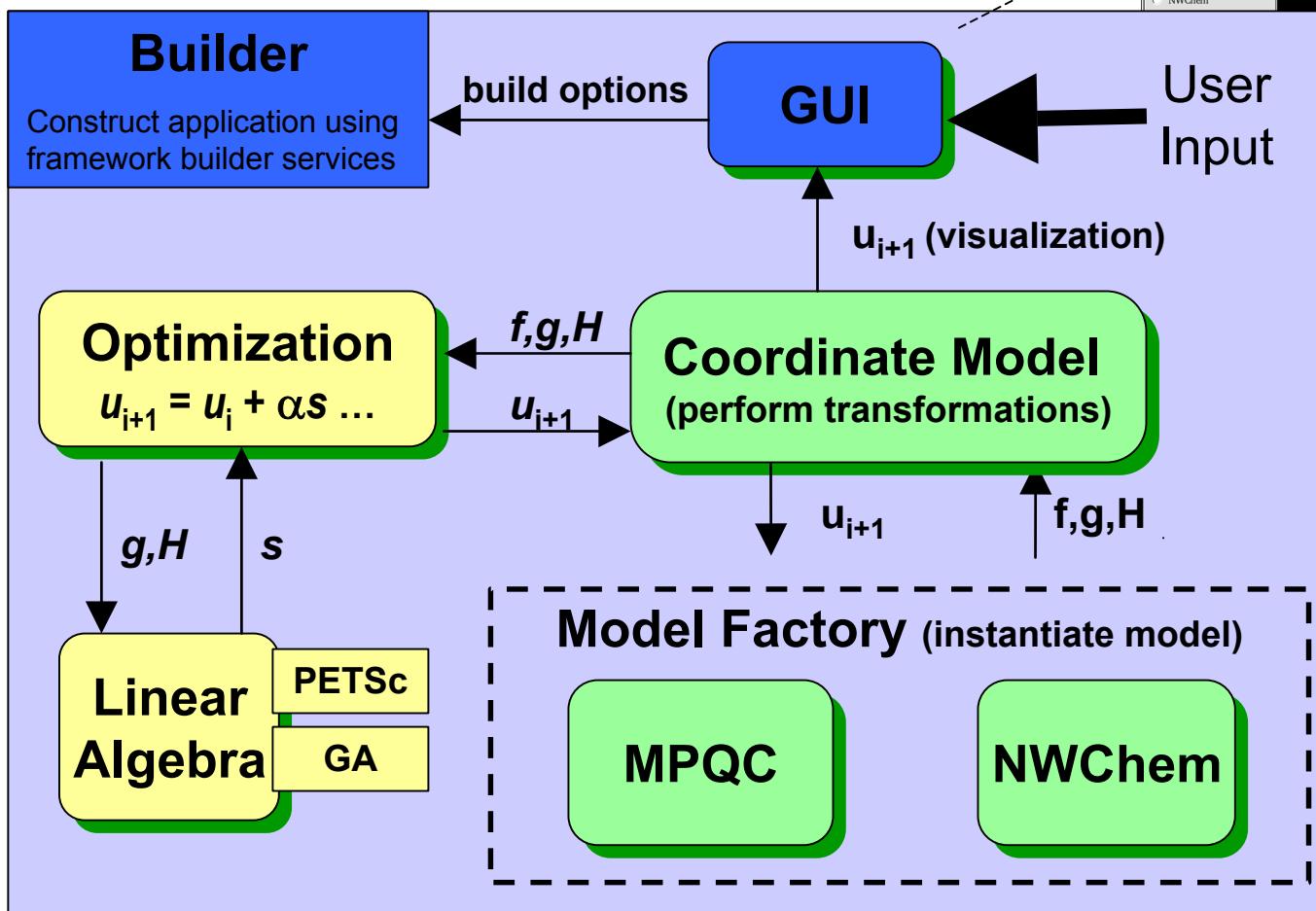
David Bernholdt (ORNL)



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

Software Architecture



$f(u)$ energy
 u Cartesian coordinates
 u internal coordinates
 g gradient (in Cartesians)
 g gradient (in internals)
 H Hessian (in Cartesians)
 H Hessian (in internals)
 s update (in internals)

Preliminary Performance Evaluation

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

CCA Summary

- Component-based software engineering is a tool to...
 - Simplify construction of complex software systems
 - Facilitate interoperability and reuse of software
- The CCA is specially designed for high-performance scientific computing
 - Close interaction between domain scientists and computer scientists
- Theoretical benefits of CBSE are being borne out by initial CCA applications
 - Combustion, Quantum Chemistry, Climate Modeling, Fusion...

The Tensor Contraction Engine: A Tool for Quantum Chemistry

Oak Ridge National
Laboratory

*David E. Bernholdt,
Venkatesh Choppella, Robert
Harrison*

Pacific Northwest National
Laboratory

So Hirata

Louisiana State University
J Ramanujam

Ohio State University

*Gerald Baumgartner, Alina
Bibireata, Daniel Cociorva,
Xiaoyang Gao, Sriram
Krishnamoorthy, Sandhya
Krishnan, Chi-Chung Lam,
Quingda Lu, Russell M.
Pitzer, P Sadayappan,
Alexander Sibiryakov*

University of Waterloo
*Marcel Nooijen, Alexander
Auer*

Research at ORNL supported by the Laboratory Directed Research and Development Program
Oak Ridge National Laboratory is managed by UT-Battelle, LLC for the US Dept. of Energy under contract DE-AC-05-00OR22725.

Research at PNNL supported by the Office of Basic Energy Sciences, U. S. Dept. of Energy
Research at OSU, Waterloo, and LSU supported by the National Science Foundation Information Technology Research Program

Problem Domain: High-Accuracy Quantum Chemical Methods

- Coupled cluster methods are widely used for very high quality electronic structure calcs.
- Typical Laplace factorized CCSD(T) term:

$$A3A = \frac{1}{2} (X_{ce,af} Y_{ae,cf} + X_{\bar{ce},\bar{af}} \bar{Y}_{\bar{ae},\bar{cf}} + X_{\bar{ce},\bar{af}} Y_{\bar{ae},cf} \\ + X_{\bar{ce},\bar{af}} \bar{Y}_{\bar{ae},\bar{cf}} + X_{\bar{ce},\bar{af}} \bar{Y}_{\bar{ae},\bar{cf}} + X_{\bar{ce},\bar{af}} \bar{Y}_{\bar{ae},\bar{cf}})$$

$$X_{ce,af} = t_{ij}^{ce} t_{ij}^{af}$$

$$Y_{ae,cf} = \langle ab | ek \rangle \langle cb | fk \rangle$$

Typical methods will have tens to hundreds of such terms

- Indices i, j, k $O(O=100)$ values, a, b, c, e, f $O(V=3000)$
- Term costs $O(OV^5) \approx 10^{19}$ FLOPs; Integrals ~ 1000 FLOPs each
- $O(V^4)$ terms ~ 500 TB memory each

CCSD Doubles Equation

```

hbar[a,b,i,j] == sum[f[b,c]*t[i,j,a,c],{c}] -sum[f[k,c]*t[k,b]*t[i,j,a,c],{k,c}] +sum[f[a,c]*t[i,j,c,b],{c}] -sum[f[k,c]*t[k,a]*t[i,j,c,b],{k,c}] -
sum[f[k,j]*t[i,k,a,b],{k}] -sum[f[k,c]*t[j,c]*t[i,k,a,b],{k,c}] -sum[f[k,i]*t[j,k,b,a],{k}] -sum[f[k,c]*t[i,c]*t[j,k,b,a],{k,c}] -
+sum[t[i,c]*t[j,d]*v[a,b,c,d],{c,d}] +sum[t[i,j,c,d]*v[a,b,c,d],{c,d}] +sum[t[j,c]*v[a,b,i,c],{c}] -sum[t[k,b]*v[a,k,i,j],{k}] -
+sum[t[i,c]*v[b,a,j,c],{c}] -sum[[k,a]*v[b,k,j,i],{k}] -sum[t[k,d]*t[i,j,c,b]*v[k,a,c,d],{k,c,d}] -sum[t[i,c]*t[j,k,b,d]*v[k,a,c,d],{k,c,d}] -
sum[t[j,c]*t[k,b]*v[k,a,c,i],{k,c}] +2*sum[t[j,k,b,c]*v[k,a,c,i],{k,c}] -sum[t[j,k,c,b]*v[k,a,c,i],{k,c}] -sum[t[i,c]*t[j,d]*v[k,a,d,c],{k,c,d}] -
+2*sum[t[k,d]*t[i,j,c,b]*v[k,a,d,c],{k,c,d}] -sum[t[k,b]*t[i,j,c,d]*v[k,a,d,c],{k,c,d}] -sum[t[j,d]*t[i,k,c,b]*v[k,a,d,c],{k,c,d}] -
+2*sum[t[i,c]*t[j,k,b,d]*v[k,a,d,c],{k,c,d}] -sum[t[i,c]*t[j,k,d,b]*v[k,a,d,c],{k,c,d}] -sum[t[j,k,b,c]*v[k,a,i,c],{k,c}] -
sum[t[i,c]*t[k,b]*v[k,a,j,c],{k,c}] -sum[t[i,k,c,b]*v[k,a,j,c],{k,c}] -sum[t[i,c]*t[j,d]*t[k,a]*v[k,b,c,d],{k,c,d}] -
sum[t[k,d]*t[i,j,a,c]*v[k,b,c,d],{k,c,d}] -sum[t[k,a]*t[i,j,c,d]*v[k,b,c,d],{k,c,d}] +2*sum[t[j,d]*t[i,k,a,c]*v[k,b,c,d],{k,c,d}] -
sum[t[j,d]*t[i,k,c,a]*v[k,b,c,d],{k,c,d}] -sum[t[i,c]*t[j,k,d,a]*v[k,b,c,d],{k,c,d}] -sum[t[i,c]*t[k,a]*v[k,b,c,j],{k,c}] -
+2*sum[t[i,k,a,c]*v[k,b,c,j],{k,c}] -sum[t[i,k,c,a]*v[k,b,c,j],{k,c}] +2*sum[t[k,d]*t[i,j,a,c]*v[k,b,d,c],{k,c,d}] -
sum[t[j,d]*t[i,k,a,c]*v[k,b,d,c],{k,c,d}] -sum[t[j,c]*t[k,a]*v[k,b,i,c],{k,c}] -sum[t[j,k,c,a]*v[k,b,i,c],{k,c}] -sum[t[i,k,a,c]*v[k,b,j,c],{k,c}] -
+sum[t[i,c]*t[j,d]*t[k,a]*t[l,b]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[k,b]*t[i,d]*t[i,j,a,c]*v[k,l,c,d],{k,l,c,d}] -
2*sum[t[k,a]*t[l,d]*t[i,j,c,b]*v[k,l,c,d],{k,l,c,d}] +sum[t[k,a]*t[l,b]*t[i,j,c,d]*v[k,l,c,d],{k,l,c,d}] -
2*sum[t[j,c]*t[i,d]*t[i,k,a,b]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[j,d]*t[i,k,a,b]*v[k,l,c,d],{k,l,c,d}] -
+sum[t[j,d]*t[l,b]*t[i,k,c,a]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,c]*t[l,d]*t[j,k,b,a]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,c]*t[l,a]*t[j,k,b,d]*v[k,l,c,d],{k,l,c,d}] -
+sum[t[i,c]*t[l,b]*t[j,k,d,a]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,k,c,d]*t[j,l,b,a]*v[k,l,c,d],{k,l,c,d}] +4*sum[t[i,k,a,c]*t[j,l,b,d]*v[k,l,c,d],{k,l,c,d}] -
2*sum[t[i,k,c,a]*t[j,l,b,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,k,a,b]*t[j,l,c,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,k,a,c]*t[j,l,d,b]*v[k,l,c,d],{k,l,c,d}] -
+sum[t[i,k,c,a]*t[j,l,d,b]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,c]*t[j,d]*t[k,l,a,b]*v[k,l,c,d],{k,l,c,d}] +sum[t[i,j,c]*t[k,l,a,b]*v[k,l,c,d],{k,l,c,d}] -
2*sum[t[i,j,c,b]*t[k,l,a,d]*v[k,l,c,d],{k,l,c,d}] -2*sum[t[i,j,a,c]*t[k,l,b,d]*v[k,l,c,d],{k,l,c,d}] +sum[t[j,c]*t[k,b]*t[l,a]*v[k,l,c,i],{k,l,c}] -
+sum[t[i,c]*t[j,k,b,a]*v[k,l,c,i],{k,l,c}] -2*sum[t[i,a]*t[j,k,b,c]*v[k,l,c,i],{k,l,c}] +sum[t[i,a]*t[j,k,c,b]*v[k,l,c,i],{k,l,c}] -
2*sum[t[k,c]*t[j,l,b,a]*v[k,l,c,i],{k,l,c}] +sum[t[k,a]*t[j,l,b,c]*v[k,l,c,i],{k,l,c}] +sum[t[k,b]*t[j,l,c,a]*v[k,l,c,i],{k,l,c}] -
+sum[t[j,c]*t[l,k,a,b]*v[k,l,c,i],{k,l,c}] +sum[t[i,c]*t[j,l,b]*v[k,l,c,i],{k,l,c}] +sum[t[i,c]*t[j,k,a,b]*v[k,l,c,i],{k,l,c}] -
2*sum[t[i,c]*t[l,k,a,c]*v[k,l,c,j],{k,l,c}] +sum[t[i,b]*t[i,k,c,a]*v[k,l,c,j],{k,l,c}] +sum[t[i,c]*t[k,l,a,b]*v[k,l,c,j],{k,l,c}] -
+sum[t[j,c]*t[l,d]*t[i,k,a,b]*v[k,l,d,c],{k,l,c,d}] +sum[t[j,d]*t[i,k,a,c]*v[k,l,d,c],{k,l,c,d}] +sum[t[j,d]*t[i,k,c,b]*v[k,l,d,c],{k,l,c,d}] -
2*sum[t[i,k,c,d]*t[j,l,b,a]*v[k,l,d,c],{k,l,c,d}] -2*sum[t[i,k,a,c]*t[j,l,b,d]*v[k,l,d,c],{k,l,c,d}] +sum[t[i,k,c,a]*t[j,l,b,d]*v[k,l,d,c],{k,l,c,d}] -
+sum[t[i,k,a,b]*t[j,l,c,d]*v[k,l,d,c],{k,l,c,d}] +sum[t[i,k,c,b]*t[j,l,d,a]*v[k,l,d,c],{k,l,c,d}] +sum[t[i,k,a,c]*t[j,l,d,b]*v[k,l,d,c],{k,l,c,d}] -
+sum[t[k,a]*t[l,b]*v[k,l,i,j],{k,l}] +sum[t[k,l,a,b]*v[k,l,i,j],{k,l}] +sum[t[k,b]*t[l,d]*t[i,j,a,c]*v[l,k,c,d],{k,l,c,d}] -
+sum[t[k,a]*t[l,d]*t[i,j,c,b]*v[l,k,c,d],{k,l,c,d}] +sum[t[i,c]*t[l,d]*t[j,k,b,a]*v[l,k,c,d],{k,l,c,d}] -2*sum[t[i,c]*t[l,a]*t[j,k,b,d]*v[l,k,c,d],{k,l,c,d}] -
+sum[t[i,c]*t[l,a]*t[j,k,d,b]*v[l,k,c,d],{k,l,c,d}] +sum[t[i,j,c,b]*t[k,l,a,d]*v[l,k,c,d],{k,l,c,d}] +sum[t[i,j,a,c]*t[k,l,b,d]*v[l,k,c,d],{k,l,c,d}] -
2*sum[t[i,c]*t[i,k,a,b]*v[l,k,c,j],{k,l,c}] +sum[t[i,b]*t[i,k,a,c]*v[l,k,c,j],{k,l,c}] +sum[t[i,l,a]*t[i,k,c,b]*v[l,k,c,j],{k,l,c}] +v[a,b,i,j]

```

In the coupled cluster method with single and double excitations (CCSD) the “singles” and “doubles” equations are iterated until convergence and that solution is used to evaluate the molecular energy

The “Tensor Contraction Engine” Addresses Programming Challenges

- User describes computational problem (tensor contractions, a la many-body methods) in a simple, high-level language
 - Similar to what might be written in papers
- Compiler-like tools translate high-level language into traditional Fortran (or C, or...) code
- Generated code is compiled and linked to libraries providing computational infrastructure
- **Productivity**
 - User writes simple, high-level code
 - Code generation tools do the tedious work
- **Complexity**
 - Significantly reduces complexity visible to programmer
- **Performance**
 - Perform optimizations prior to code generation
 - Automate many decisions humans make empirically
 - Tailor generated code to target computer
 - Tailor generated code to specific problem

So What's New About This Project?

- The creation of “little languages” and code generation tools has a long history in chemistry and other domains
- **Usually viewed only as productivity tools**
 - Imitate what researcher would do – but quicker
- **We treat it as a computer science problem**
 - Similar to (not identical to) an optimizing compiler
 - Algorithmic choices are explored and evaluated rigorously and (in most cases) exhaustively
 - Make use of machine architecture & performance models to specialize generated code to target system
- **Target applications**
 - Rapid experimentation with new many-body methods
 - Implementation of high-complexity methods
 - Improving computational efficiency on parallel machines
 - Also for nuclear physics...

A High-Level Language for Tensor Contraction Expressions

```
range V = 3000;  
range O = 100;  
  
index a,b,c,d,e,f : V;  
index i,j,k : O;  
  
mlimit = 1000000000000;
```

$$A3A = \frac{1}{2} (X_{ce,af} Y_{ae,cf} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} \\ + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}}) \\ X_{ce,af} = t_{ij}^{ce} t_j^{af} \quad Y_{ae,cf} = \langle ab \parallel ek \rangle \langle cb \parallel fk \rangle$$

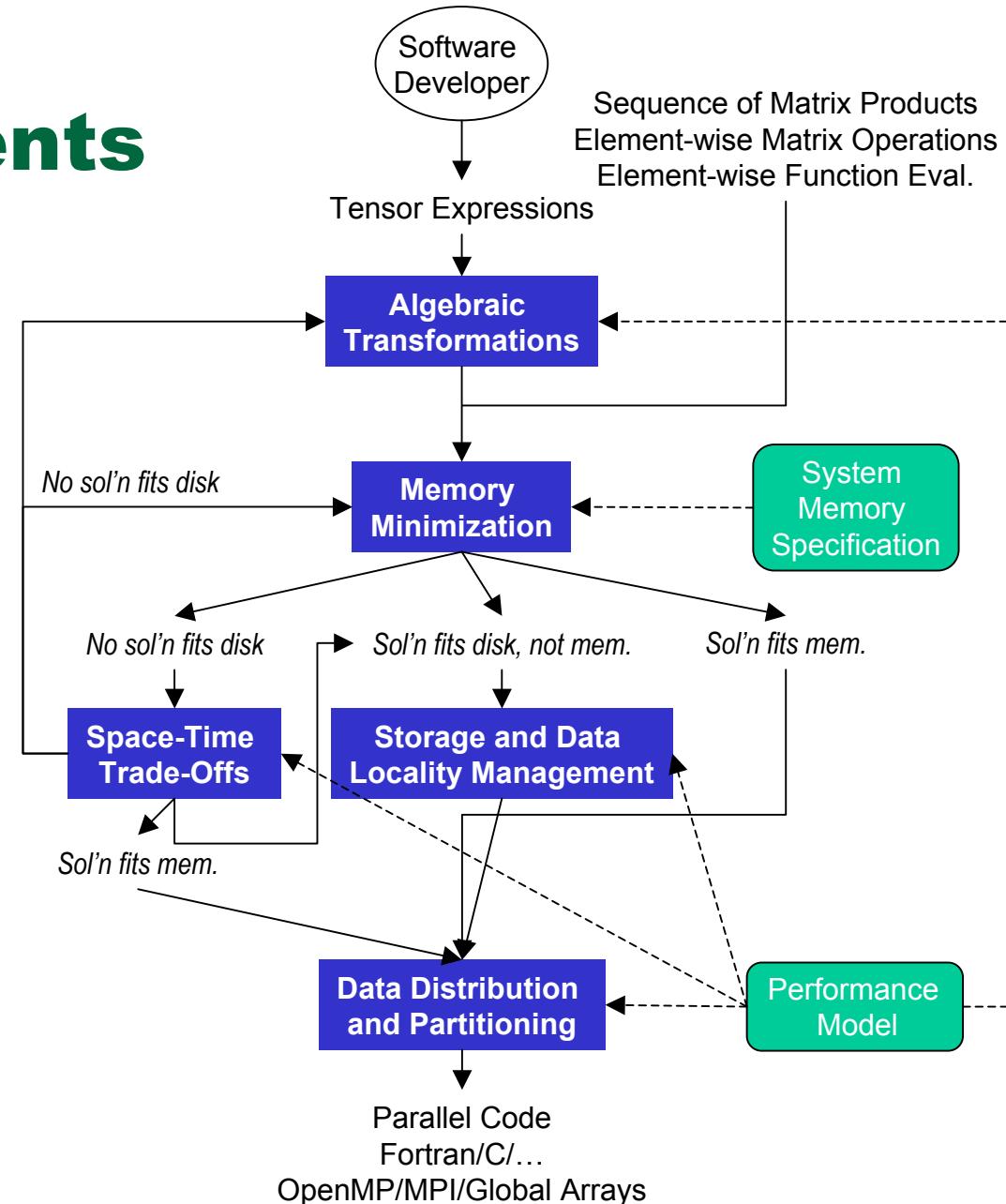
```
function F1(V,V,V,O);
function F2(V,V,V,O);

procedure P(in T1[O,O,V,V], in T2[O,O,V,V], out X)=

begin
    X == sum[ sum[F1(a,b,f,k) * F2(c,e,b,k), {b,k}]
              * sum[T1[i,j,a,e] * T2[i,j,c,f], {i,j}],
              {a,e,c,f}];
end
```

TCE Components

- Algebraic Transformations
 - Minimize operation count
- Memory Minimization
 - Reduce intermediate storage
- Space-Time Transformation
 - Trade-offs btw storage and recomputation
- Storage Management and Data Locality Optimization
 - Optimize use of storage hierarchy
- Data Distribution and Partitioning
 - Optimize parallel layout



Operation-Minimal and Memory-Minimal Forms

for a, e, c, f

$$\begin{bmatrix} \text{for } i, j \\ \quad \boxed{X_{aecf}} += T_{ijae} T_{ijcf} \end{bmatrix}$$

for c, e, b, k

$$\begin{bmatrix} \quad \boxed{T1_{cebk}} = f1(c, e, b, k) \end{bmatrix}$$

for a, f, b, k

$$\begin{bmatrix} \quad T2_{afbk} = f2(a, f, b, k) \end{bmatrix}$$

for c, e, a, f

$$\begin{bmatrix} \quad \text{for } b, k \\ \quad \boxed{Y_{ceaf}} += T1_{cebk} T2_{afbk} \end{bmatrix}$$

for c, e, a, f

$$\begin{bmatrix} \quad E += X_{aecf} Y_{ceaf} \end{bmatrix}$$

array	space	time
X	$V^4 \rightarrow 1$	$V^4 O^2$
T1	$V^3 O \rightarrow VO$	$C_{f1} V^3 O$
T2	$V^3 O$	$C_{f2} V^3 O$
Y	$V^4 \rightarrow 1$	$V^5 O$
E	1	V^4

$$\begin{aligned} A3A &= \frac{1}{2} \left(\boxed{X_{ce,af} Y_{ae,cf}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} \right. \\ &\quad \left. + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} \right) \\ X_{ce,af} &= t_{ij}^{ce} t_{ij}^{af} \quad Y_{ae,cf} = \langle ab \| ek \rangle \langle cb \| fk \rangle \end{aligned}$$

a .. f: range V = 1000 .. 3000
 i .. k: range O = 30 .. 100

for a, f, b, k

$$\begin{bmatrix} \quad T2_{afbk} = f2(a, f, b, k) \end{bmatrix}$$

for c, e

$$\begin{bmatrix} \quad \text{for } b, k \\ \quad \boxed{T1_{bk}} = f1(c, e, b, k) \end{bmatrix}$$

for a, f

$$\begin{bmatrix} \quad \text{for } i, j \\ \quad \boxed{X} += T_{ijae} T_{ijcf} \end{bmatrix}$$

for b, k

$$\begin{bmatrix} \quad \boxed{Y} += T1_{bk} T2_{afbk} \\ \quad E += X Y \end{bmatrix}$$

Tiling to Reduce Recomputation

for a^t, e^t, c^t, f^t

for a, e, c, f

for i, j

$$X_{aecf} += T_{ijae} T_{ijcf}$$

for b, k

for c, e

$$T1_{ce} = f1(c, e, b, k)$$

for a, f

$$T2_{af} = f2(a, f, b, k)$$

for c, e, a, f

$$Y_{ceaf} += T1_{ce} T2_{af}$$

for c, e, a, f

$$E += X_{aecf} Y_{ceaf}$$

array	space	time
X	B^4	$V^4 O^2$
T1	B^2	$C_{f1}(V/B)^2 V^3 O$
T2	B^2	$C_{f2}(V/B)^2 V^3 O$
Y	B^4	$V^5 O$
E	1	V^4

Tiling provides a **controlled** compromise between minimal operations and minimal memory (full fusion)

Current TCE Capabilities

Capability	Prototype TCE	Optimizing TCE
Basic sequential code generation for CC-based methods	Yes	Yes
QC Packages Interfaced: <ul style="list-style-type: none">• File based• General (file, memory, direct)	NWChem, UTChem	NWChem Under development
Symmetry Support: <ul style="list-style-type: none">• Spin• Spatial• Permutational	Spin orbitals Abelian Fermions	General, in progress Abelian, in progress General, in progress
Optimizations: <ul style="list-style-type: none">• Operation Minimization• Memory Minimization• Space-Time Transformation• Data Locality	Partial Partial No Partial	Yes Yes Yes Yes
Parallel code generation	Limited general	General, in progress

Methods Implemented to Date using TCE

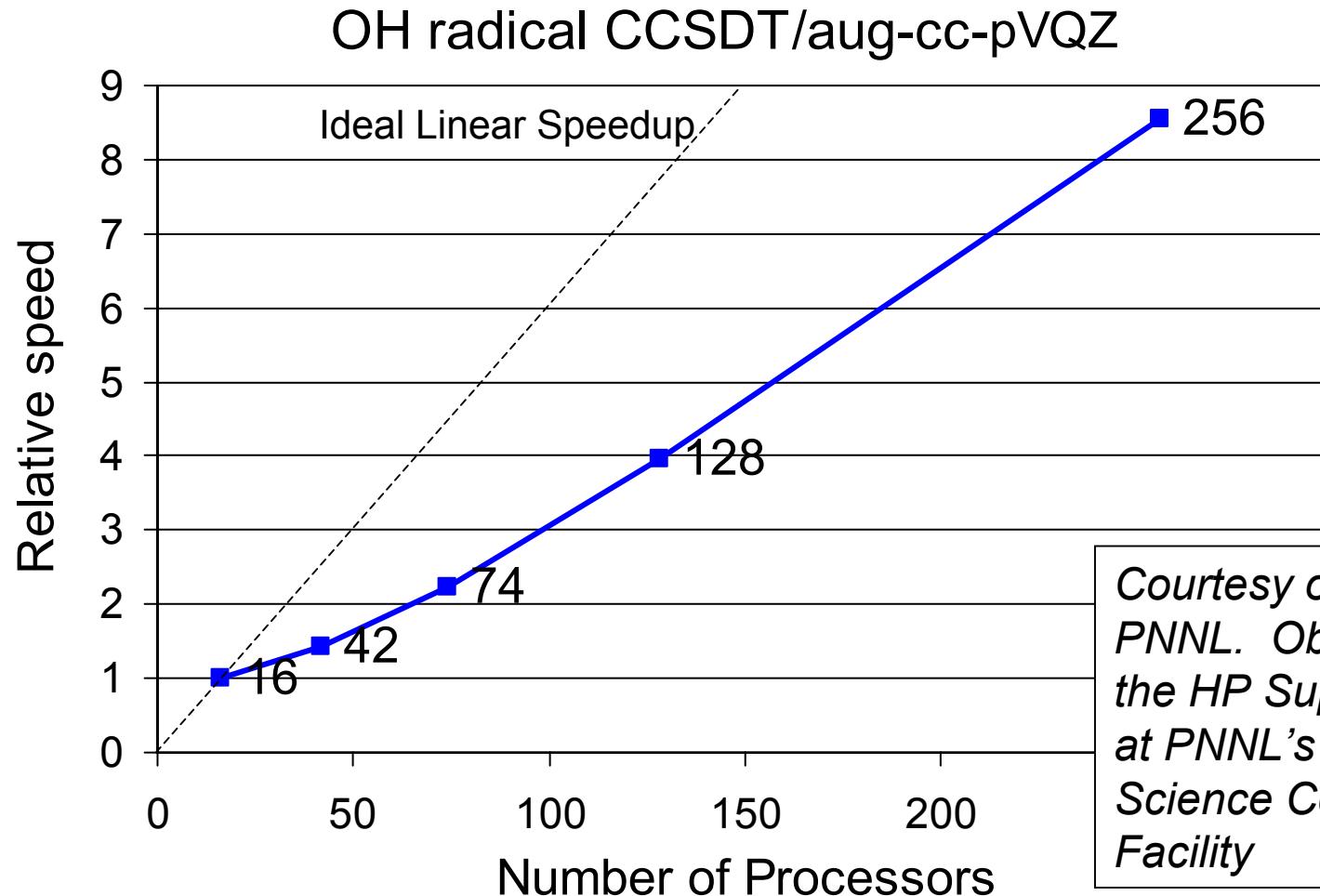
Prototype TCE

- Parallel
 - MBPT(2), MBPT(3), MBPT(4)
 - CCD, CCSD, CCSD(T), QCISD, CCSQT, CCSDTQ
 - CISD, CISDT, CISDTQ
 - CCSD, CCSQT, CCSDTQ lambda equations
 - CCSD, CCSQT, CCSDTQ dipole moments
 - Parallel EOM-CCSD
 - Local/AO-based CCSD
- Sequential
 - Relativistic 2/4-component CI, CC, & MBPT

Optimizing TCE

- *Parallel (limited)*
 - CCD, CCSD
- Code availability
 - Prototype TCE to be released with NWChem 4.5
 - Optimizing TCE still under intense development
 - Some prototype-generated implementations to appear in ...
 - NWChem 4.5
 - UTChem 2003

Parallel Scalability of Prototype TCE-Generated Code



The Value of Optimization (Space-Time Trade-Offs)

```

range V = 3000;
range O = 100;

index a,b,c,d,e,f : V;
index i,j,k : O;

mlimit = 1000000000000;

function F1(V,V,V,O);
function F2(V,V,V,O);

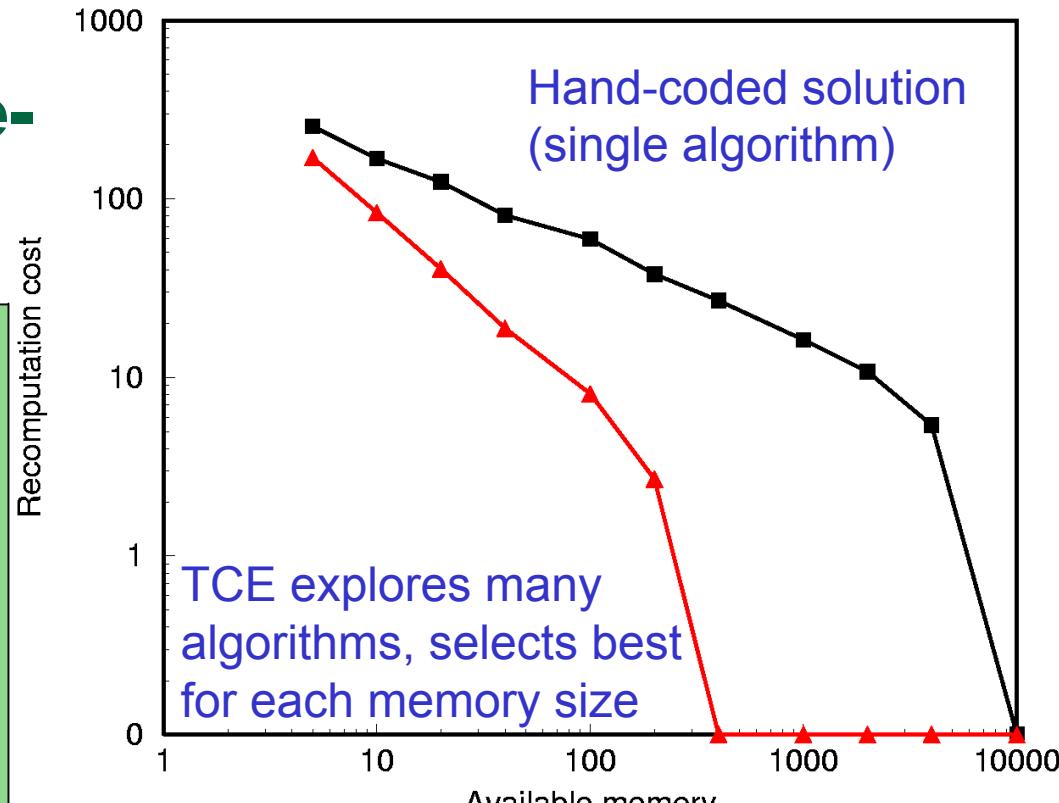
```

```
procedure P(in T1[O,O,V,V], in T2[O,O,V,V], out X)=
```

```

begin
  X == sum[ sum[ F1(a,b,f,k) * F2(c,e,b,k), {b,k}]
            * sum[ T1[i,j,a,e] * T2[i,j,c,f], {i,j}],
            {a,e,c,f}];
end

```



$$\begin{aligned}
 A3A &= \frac{1}{2} (X_{ce,af} Y_{ae,cf} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}\bar{f}} + X_{\bar{c}\bar{e},\bar{a}\bar{f}} Y_{\bar{a}\bar{e},\bar{c}f} \\
 &\quad + X_{\bar{c}\bar{e},\bar{a}f} Y_{\bar{a}e,\bar{c}f} + X_{\bar{c}e,\bar{a}f} Y_{\bar{a}e,\bar{c}f} + X_{\bar{c}e,\bar{a}f} Y_{\bar{a}e,\bar{c}f}) \\
 X_{ce,af} &= t_{ij}^{ce} t_{ij}^{af} \quad Y_{ae,cf} = \langle ab \| ek \rangle \langle cb \| fk \rangle
 \end{aligned}$$

On the Drawing Board...

- More flexibility in sequencing and controlling optimizations
- Common sub-expression elimination
- Global factorization (across equations)
 - Complex problem
- Improving parallel code generation
 - Multi-level parallelism
 - Threads
 - Multiple loosely coupled tasks
- More sophisticated performance models
- Develop approximate algorithms for opt.
 - Address situations where exhaustive search too expensive
 - i.e. Deliver best result spending at most 3 min on code gen.
 - ... or 60 min ... or 3 days ...
- Generalizations beyond electronic structure

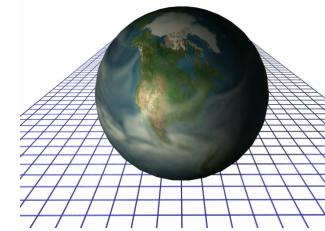
Ways to Use the TCE

Mode	Prototype TCE	Optimizing TCE
Explore/develop new methods rapidly	Now	Soon
Develop parallel implementations	Now	Soon
Develop large-scale high-performance parallel implementations	Under consideration	Under development

TCE Summary

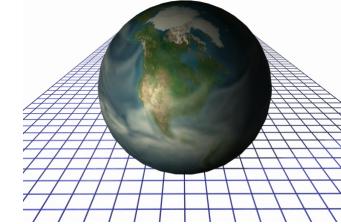
- Automatic generation of code from high-level algebraic expressions
 - Approach problem like a compiler
 - Use of “high-level language” allows automation of design decisions usually made by human software developer
 - Produce robust, reliable code
- Addresses productivity, complexity, and performance
 - Compiler-like optimizations key to full utility of code generation approaches
- Strong interdisciplinary collaboration between chemists and computer scientists
 - Problem from chemists, solutions from computer scientists (w/ significant help from chemists)

The Earth System Grid: Turning Climate Simulation Data into Community Resources

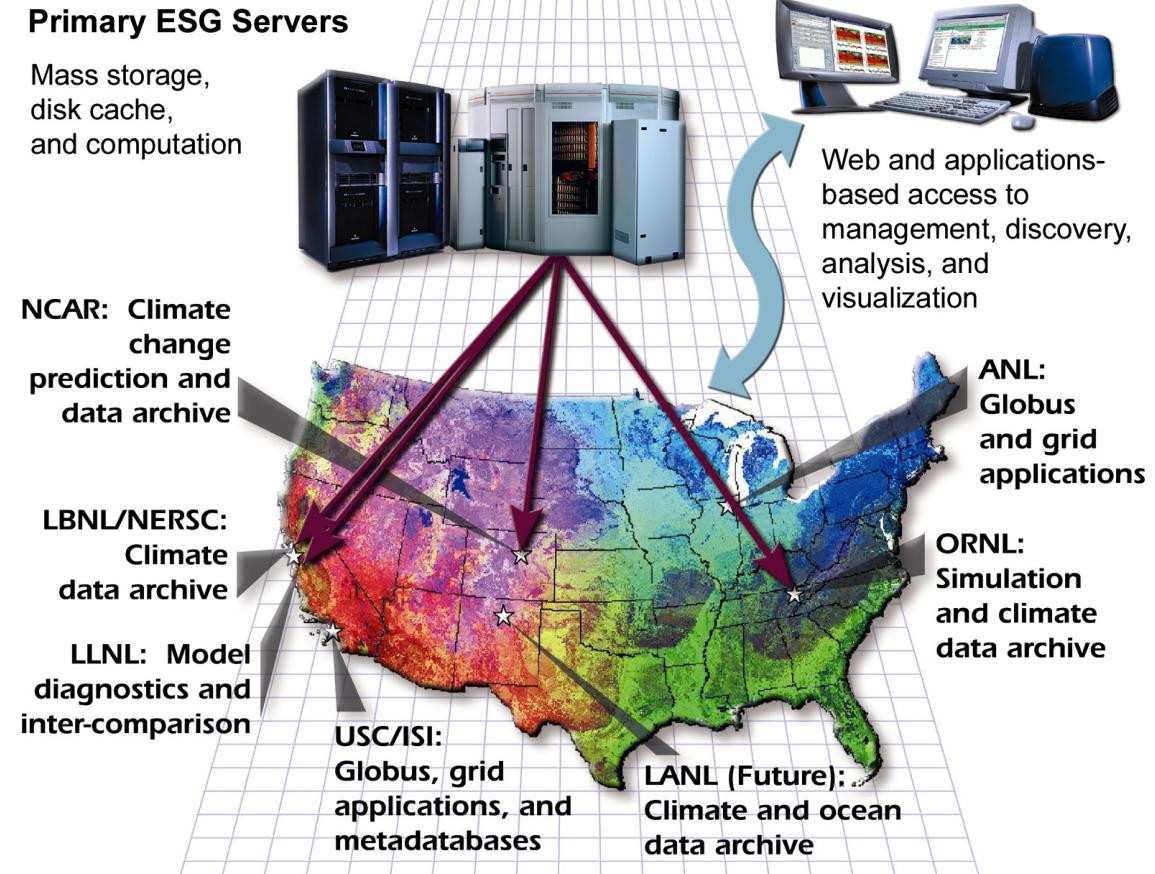


Argonne, Lawrence Berkley, Lawrence
Livermore, and Oak Ridge National
Laboratories,
National Center for Atmospheric Research,
and
Information Sciences Institute

The Earth System Grid



- A “data grid” project to facilitate widespread access to climate (simulation) data
- A few sites in the US generate climate data
- Dozens to hundreds use it at various levels
- Datasets are TB-scale, in 10,000+ individual files
- Users typically need only a fraction of the data
- Current process very human-intensive

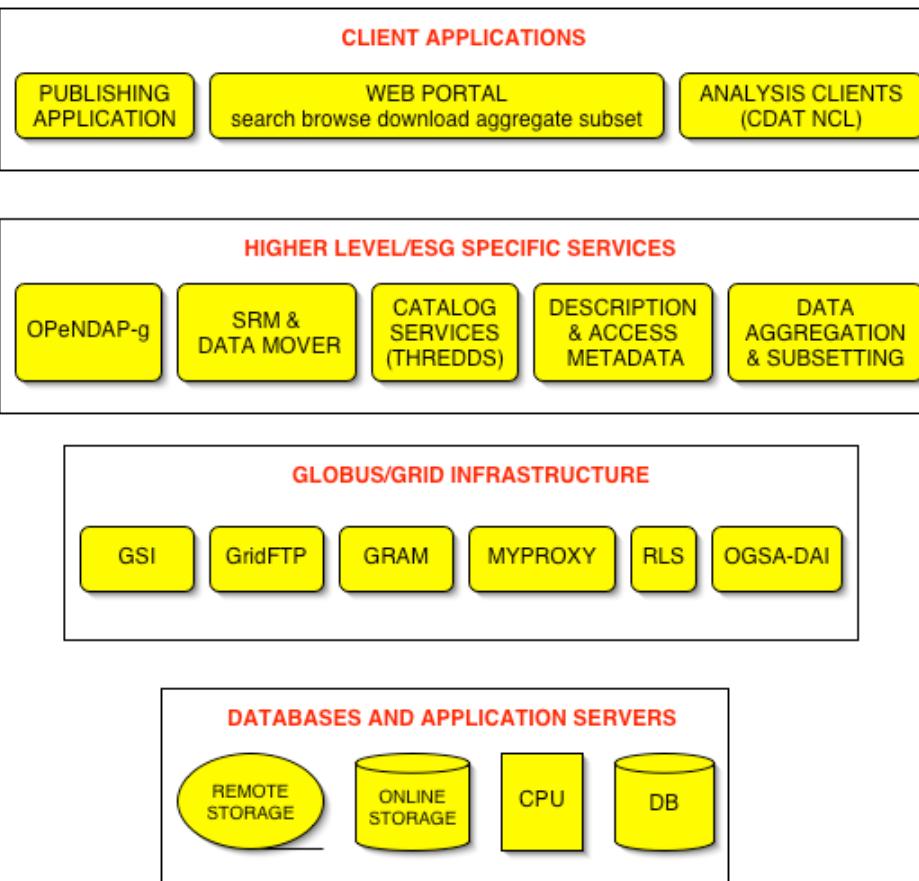


www.earthsystemgrid.org

ESG Strategies and Focus Areas

- Move data a minimal amount, keep it close to computational point of origin when possible
 - Data access protocols, distributed analysis
- When we must move data, do it fast and with a minimum amount of human intervention
 - Storage Resource Management, fast networks
- Keep track of what we have, particularly what's on deep storage
 - Metadata and Replica Catalogs
- Harness a federation of sites
 - Globus Toolkit -> The Earth System Grid -> The UltraDataGrid

ESG Architecture



- Web portal at NCAR
- Analysis clients at NCAR, LLNL, ORNL
- ESG-specific services at all sites
- Globus/Grid infrastructure at all sites
- Mass storage systems at LBNL, NCAR, ORNL

ESG Web Portal

Earth System Grid - Netscape

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop https://dataportal.ucar.edu:8443/esg/index.jsp Search Print N

Mail AIM Home Radio Netscape Search Shop Bookmarks ACD CNN ESG GAZZETTA GLOBUS NETS SCD VETS UCAR

Earth System Grid

ESG @ WORK APPLICATIONS INDEX

Data Search and Discovery

- ESG Data Discovery**
This integrated application allows identification of data (logical files and replica locations) based on a query to the ESG Metadata Catalog and Replica Catalog, display of logical files and logical collections metadata, and transfer of files.
- Browse ESG Data Catalogs**
Hierarchical browsing of ESG Data Catalogs in THREDDS format, and display of associated metadata.
- Query ESG Metadata Catalog**
Query of static and user defined attributes associated with logical collections or logical files in the ESG Metadata Catalog.

Data Management

- Data Transfer**
Web interface to HRM (Hierarchical Resource Manager). HRM allows high speed, reliable, parallel streams, multiple files transfer between permanent storage systems and ESG nodes.
- Metadata Extraction**
Web Service for the automatic extraction of metadata in NcML format from any network retrievable netcdf file.

Data Analysis and Visualization

- DODS**
Access to the ESG DODS servers.
- LAS**
Access to the ESG LAS servers.
- CDAT**
Access to the ESG CDAT servers.

ESG System

- Monitor Status**
Monitor the status and availability of the resources comprising the ESG system.
- Query Logs**
Interactive query of log files recording ESG access and usage.

ACKNOWLEDGMENTS: ESG applications are based on a variety of computing and information technologies. In particular, we wish to acknowledge the use of Globus technology and related COG (Commodity Grid Kit) toolkits.

Document: Done (3.195 secs)

start VETS_REPORTS CDP Earth System Grid - N... Microsoft PowerPoint 3:22 PM

Earth System Grid - Netscape

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop https://dataportal.ucar.edu:8443/esg/dataDiscovery.jsp?collectionName=PCM/B06.20/atm Search Print N

Mail AIM Home Radio Netscape Search Shop Bookmarks ACD CNN ESG GAZZETTA GLOBUS NETS SCD VETS UCAR

Earth System Grid

ESG through Advanced Computing

ESG Home ESG Login CAS proxy ESG Logout ESG @ Work: |

ESG DATA DISCOVERY

INSTRUCTIONS: This interface allows search and discovery of data through association performed to the **MCS (Metadata Cataloguing Services)** to retrieve all matching logical collections. The user can then query the **Replica Location Services** to find all registered replica locations, sizes, and access methods for each logical collection. The data can be transferred from disk or remote storage (**HPSS, MSS**) and transfer them with the **SRM (Storage Resource Manager)** to another location. The data can be transferred via **gridFTP** or **openDAP** servers. Optionally, the user may also request a logical collection or logical file. Please note that currently the query supports only a logical collection or logical file.

QUERY FORM

LOGICAL COLLECTION NAME: **PCM/B06.20/atm** (required)

VARIABLE NAME: (optional)

START YEAR: **1980** (required)

STOP YEAR: **1985** (required)

SUBMIT QUERY

SESSION ID: 00AFCA2870A97FD8E572993D94F168BB

DN: CN=proxy,CN=proxy,CN=proxy,CN=Globus User,OU=Network Engineering and Test Center for Atmospheric Research,O=NCAR

Document: Done (0.271 secs)

start VETS_REPORTS CDP Earth System Grid - N...

Earth System Grid - Netscape

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop https://dataportal.ucar.edu:8443/esg/HRM/hrm.jsp Search Print N

Mail AIM Home Radio Netscape Search Shop Bookmarks ACD CNN ESG GAZZETTA GLOBUS NETS SCD VETS UCAR

Earth System Grid

HRM WEB INTERFACE

INSTRUCTIONS: Please check all physical files that you wish to transfer. The HRM (Hierarchical Resource Manager) is a product of the Scientific Data Management Research Group at the Lawrence Berkely National Laboratory. HRM allows fast, reliable, tunable transfer of files to/from disk and remote storage (HPSS, MSS) through the gridFTP protocol and implementation. This prototype web interface is being developed by the Earth System Grid collaboration.

DATA TRANSFER FORM

	PHYSICAL FILE	SIZE	SERVICE
<input checked="" type="checkbox"/>	/raid/f1/ESG_SC2002/PCM/B06.20/atm/B06.20.atm.1980.nc	268722384	DISK@NCAR
<input checked="" type="checkbox"/>	/dataportal.ucar.edu/PCM1/pcm/B06.20/atm/B06.20.atm.1981.nc	322461980	MSS@NCAR
<input checked="" type="checkbox"/>	/hpss.ccs.ornl.gov/home/asim/esg/sc2002/pcm/b06.20/atm/B06.20.atm.1982.nc	322461980	HPSS@ORNLL
<input checked="" type="checkbox"/>	/archive.nersc.gov/nersc/gc5/asim/esg/sc2002/pcm/b06.20/atm/B06.20.atm.1983.nc	322461980	HPSS@LBNL
<input checked="" type="checkbox"/>	/data/esg/sc2002/PCM/B06.20/atm/B06.20.atm.1984.nc	322461980	DISK@LBNL

INPUT FILES:

OUTPUT DIRECTORY: /scratch/sc2002/tmp/

USER ID: esguser (user@host)

gridFTP PARAMETERS:

Block size:	1000000
TCP buffer size:	1000000
Number of streams:	4

SUBMIT REQUEST **RESET FORM**

start VETS_REPORTS CDP Earth System Grid ... Earth System Grid ... Microsoft PowerPoint 3:28 PM

ESG Web Portal

Data Discovery and Movement

Live Access to Climate Data - Netscape

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop http://dataportal.ucar.edu/esg-las/main.pl? Search Print N

Mail AIM Home Radio Netscape Search Shop Bookmarks ACD CNN ESG GAZZETTA GLOBUS NETS SCD VETS UCAR

Live Access to Climate Data

Home Help Options Data Sets

B06.20.atm.1980.nc

Average of Cloud when omega is < 0 (up)
Average of Omega when omega is < 0 (up)
BEVAP
BTRAN
CH4VMR
Clearsky net longwave flux at surface
Clearsky net longwave flux at top
Clearsky net solar flux at surface
Clearsky net solar flux at top
Cloud fraction
CO2VMR
convective adjustment tendency of water vapor
Convective precipitation rate
Convective snow rate (water equivalent)
Counter-gradient coefficient on surface
kinematic fluxes
DMI
East-west gravity wave drag surface stress
Effective cloud fraction
F11VMR
F12VMR

Document: Done (0.15 secs)

start VETS_REPORTS CDP Earth System Grid ... Live Access to Cli... Microsoft PowerPoint 3:34 PM

THE EARTH SYSTEM GRID ESG Scientific Discovery through Advanced Computing

B06.20.atm.1980.nc

Average of Cloud when omega is < 0 (up)

Select view xy (lat/lon) slice Get Data

Select single variable comparison

Go Full Region

87.8638000
180.0 W 180.0 E
87.8638000

Zoom In Zoom Out

Select depth 4.8092999458313 4.8092999458313

Select time 05-Mar-1980 05-Mar-1980

Select product Shaded plot (GIF) in 800x600 window

DATA SET: B06.20.atm.1980.nc

Sea level pressure (Pa)

80°S 180°W 160°W 140°E 120°E 100°E 80°E 60°E 40°E 20°E 0° 20°W 40°W 60°W 80°W 100°W 120°W 140°W 160°W 180°W

104400 104000 103600 103200 102800 102400 102000 101600 101200 100800 100400 100000 99600 99200 98800 98400 98000 97600 97200 96800

QBOT QDRAI QINFL QOVER Random overk amount Reference hei Relative humic RSW Sea level pres Solar heating Solar insolatio specific humic

Modify plot

start VETS_REPORTS CDP Earth System... Live Access t... Microsoft Po... 3:35 PM

ESG Web Portal

Visualization

ESG Conclusions

- Already > 100 TB data cataloged in ESG
- Preparing to support US runs for Intergovernmental Panel on Climate Change
 - 100 TB new data by mid-August (119,200 node-hours)
- Many TB previously generated data to be cataloged
- Future: federation with other climate data grid efforts

Summary

- **The Common Component Architecture (CCA)**
 - Improving the scientific software development process through the use of component-based software engineering
- **The Tensor Contraction Engine (TCE)**
 - A high-level language and optimizing compiler specialized to the needs of a class of applications in quantum chemistry and physics
- **The Earth System Grid (ESG)**
 - Providing efficient management and community access to terabyte-scale climate simulation data
- Collaborations between domain experts and computer scientists