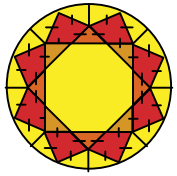


CCA
Common Component Architecture

The State of the CCA ISIC (aka CCTTSS)

PI: Rob Armstrong
rob@sandia.gov

Co-Investigators:

David Bernholdt (ORNL), Lori Freitag Diachin (SNL), Dennis Gannon (Indiana Univ.), James Kohl (ORNL), Gary Kumfert (LLNL), Lois Curfman McInnes (ANL), Jarek Nieplocha (PNNL), Steven Parker (Univ. of Utah), Craig Rasmussen (LANL)

http://www.cca-forum.org/ccttss

March 2003

Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, U. S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

Sandia National Laboratories
Lawrence Livermore National Laboratory
ORNL Oak Ridge National Laboratory
The University of Utah
Los Alamos National Laboratory
Pacific Northwest National Laboratory
Argonne National Laboratory, University of Chicago
Office of Science
U.S. Department of Energy

Outline

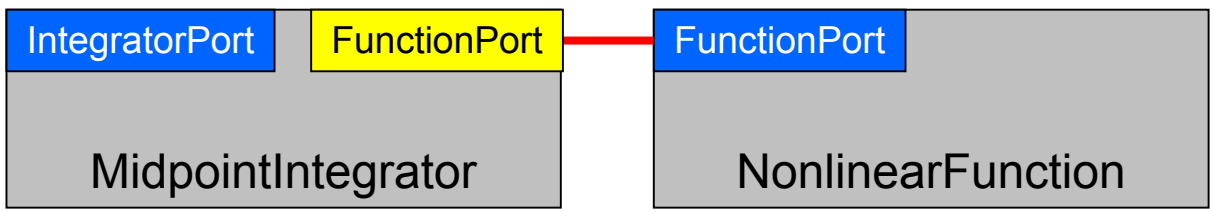
- Introduction
- Frameworks and Infrastructure
- Scientific Components
- “MxN” Parallel Data Redistribution
- User Outreach and Applications Integration
- Conclusion

Introduction

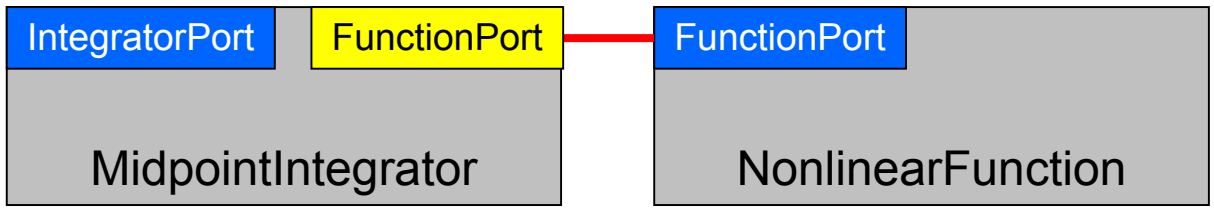
- Who are we?
  - Center for Component Technology for Terascale Simulation Software (CCTTSS)
  - Subset of members of the Common Component Architecture (CCA) Forum, an open grass-roots effort that started in 1998
- What are HPC components?
  - Generic vocabulary
  - CCA's idea of what they are
  - Parallel component vocabulary
- State of the CCA
  - Applications involvement, present and future
  - CCA responds to developers
  - Collaborations external to SciDAC

What *are* Components, Frameworks?

- **No universally accepted definition…yet**
  - Working definition: software that is composable
  - Framework is everything that is not a component
- **Interacts with the outside world *only* through well-defined interfaces**
  - Otherwise implementation is opaque to the outside world
- **Can be composed with other components**
  - "Plug-and-play" model to build applications
  - Composition based on these interfaces



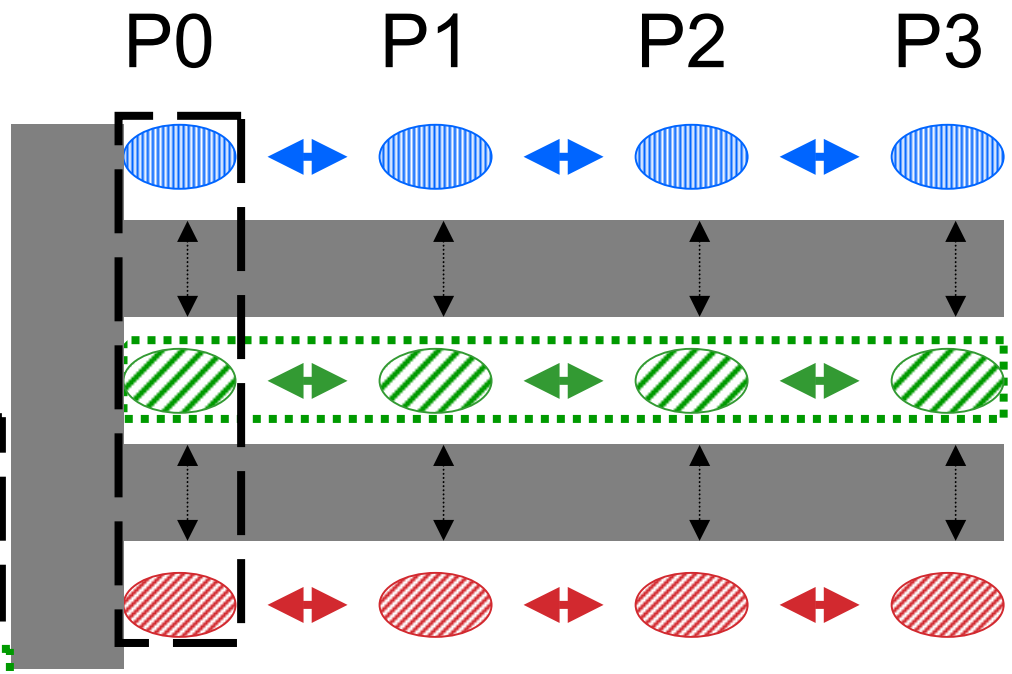
Ports: Connections Interface Exchange



- Components interact through well-defined interfaces, or *ports*
  - In OO languages, a port is a virtual class or interface
  - In Fortran, a port is a bunch of subroutines or a module
- Components may *provide* ports – implement the class or subroutines of the port
- Components may *use* ports – call methods or subroutines in the port
- Links denote a caller/callee relationship, ***not dataflow!***
  - e.g., FunctionPort could contain: *evaluate(in Arg, out Result)*

Framework Stays "Out of the Way" of Component 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

CCA Research Thrusts and Application Domains

- Frameworks
  - Framework interoperability
  - Language interoperability
  - Deployment
- Scientific Components
  - Domain-specific interfaces
  - Component implementations
  - Etc.…
- MxN Parallel Data Redistribution
  - Component-based
  - Framework-based
- Applications Outreach
  - Education
  - Best practices for use
  - Chemistry, climate

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:

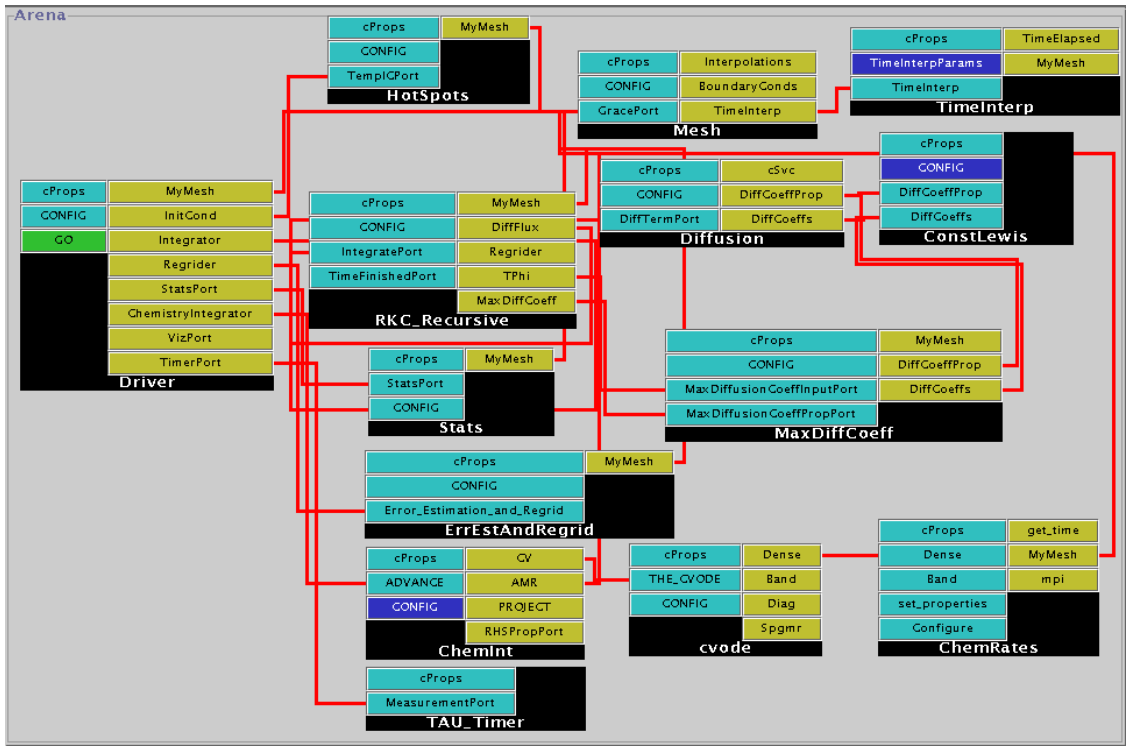
- ASCI: C-SAFE, Views, Data Svc's
- *Quantum Chemistry*

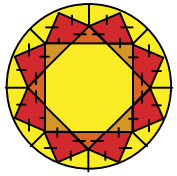
Outside of DOE:

- *NASA: ESMF, SWMF*
- Etc.…

Reacting Flow Software "Facility"

- *A Computational Facility for Reacting Flow Science* (CFRFS), SciDAC application center, PI: H. Najm
- Epitome of componentized applications
  - There is no one application, but a component "facility"





CCA
Common Component Architecture

CCA Frameworks and Infrastructure

Coordinator:
Gary Kumfert (LLNL)

Sandia National Laboratories
Lawrence Livermore National Laboratory
ORNL Oak Ridge National Laboratory
The University of Utah
Los Alamos National Laboratory
Pacific Northwest National Laboratory
Operated by Battelle for the U.S. Department of Energy
Office of Science
U.S. Department of Energy

Original Frameworks Deliverables

’01	’02	’03	’04	’05
SCMD build & implementation (SNL)	Generalize SCMD for HPC (SNL)	Improve UI of SCMD (SNL)	Iterate	Iterate
Finish initial distributed framework (UI)	Integrate MS .NET. Initial Grid support (IU)	Beta version of framework (IU,SNL)		Final version (IU,SNL)
XML to access Alexandria	Remote access by frameworks (LLNL)	Deploy Alexandria (LLNL)		
Add Python and Java to Babel (LLNL)	Add Fortran 90 to Babel (LLNL)	Add Matlab to Babel (LLNL)	Add Perl or other to Babel (LLNL)	Add COM and .NET to Babel (LLNL)
Complete concurrency standard (all)	Evolve concurrency (all)			
Develop distributed multiplexer and XML prototype (IU,SNL)	Iterate on SCMD/Grid multiplexer (IU,SNL)			

July 2002	July 2003	July 2004	July 2005	July 2006
SCMD build & implementation (SNL)	Generalize SCMD for HPC (SNL)	Improve UI of SCMD (SNL)	Iterate	Iterate
Finish initial distributed framework (UI)	Integrate MS .NET. Initial Grid support (IU)	Beta version of framework (IU,SNL)		Final version (IU,SNL)
XML to access Alexandria	Remote access by frameworks (LLNL)	Deploy Alexandria (LLNL)		
Add Python and Java to Babel (LLNL)	Add Fortran 90 to Babel (LLNL)	Add Matlab to Babel (LLNL)	Add Perl or other to Babel (LLNL)	Add COM and .NET to Babel (LLNL)
Complete concurrency standard (all)	Evolve concurrency (all)			
Develop distributed multiplexer and XML prototype (IU,SNL)	Iterate on SCMD/Grid multiplexer (IU,SNL)			

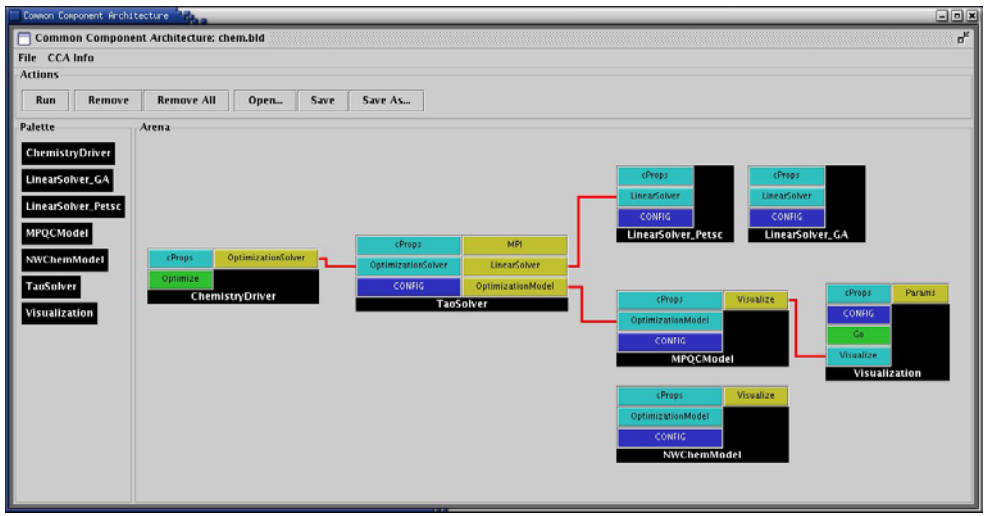
July 2002	July 2003	July 2004	July 2005	July 2006
SCMD ~~build &~~ implementation (SNL)	Generalize SCMD for HPC (SNL)	Improve UI of SCMD (SNL)	Iterate	Iterate
Finish initial distributed framework (UI)	Integrate MS .NET. Initial Grid support (IU)	Beta version of framework (IU,SNL)		Final version (IU,SNL)
XML to access Alexandria	Remote access by framework (LLNL)			
Add Python and Java to Babel (LLNL)	Add Fortran 90 to Babel (LLNL)	Add Matlab to Babel (LLNL)	Add Perl or other to Babel (LLNL)	Add COM and .NET to Babel (LLNL)
Complete concurrency standard (all)	Evolve concurrency (all)			
Develop distributed multiplexer and XML prototype (IU,SNL)	Iterate on SCMD/Grid multiplexer (IU,SNL)			

Ccaffeine

Ccaffeine

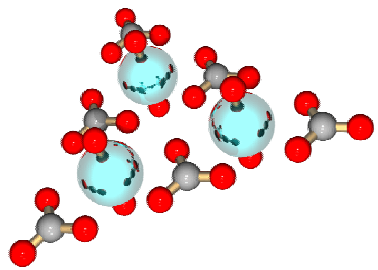
Characteristics

- SPMD
- GUI and scripted interfaces
- Interactive or batch
- Serial or parallel



Achievements

- Separates CCA pattern from implementation
  - Three bindings
    - Classic components
    - SIDL components
    - Chasm components
- Demonstrated at SC2001 & SC2002
- Used in CCA tutorials



July 2002	July 2003	July 2004	July 2005	July 2006
SCMD build & implementation (SNL)	Generalize SCMD for HPC (SNL)	Improve UI of SCMD (SNL)	Iterate	Iterate
Finish initial distributed framework (UI)	Integrate MS .NET. Initial Grid support (IU)	Beta version of framework (IU,SNL)		Final version (IU,SNL)
XML to access Alexandria	Remote access by frameworks (LLNL)	Depl… Alexa… (LLNL		
Add Python and Java to Babel (LLNL)	Add Fortran to Babel (LL…		…Perl or …r to Babel (LLNL)	Add COM and .NET to Babel (LLNL)
Complete concurrency standard (all)	Evolve concurrency (all)			
Develop distributed multiplexer and XML prototype (IU,SNL)	Iterate on SCMD/Grid multiplexer (IU,SNL)			

XCAT

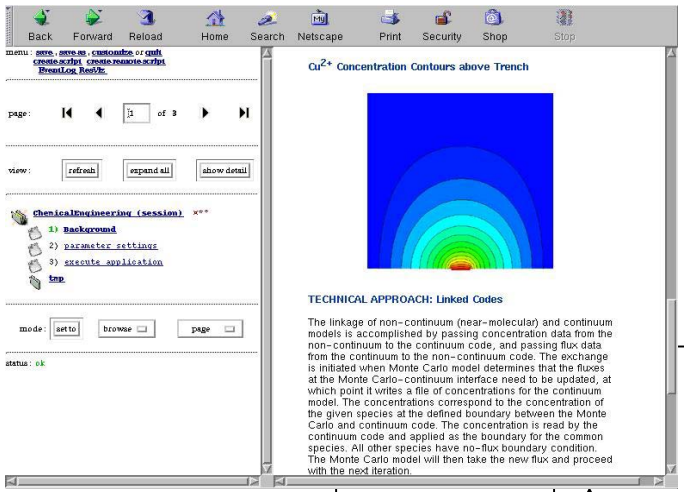
XCAT

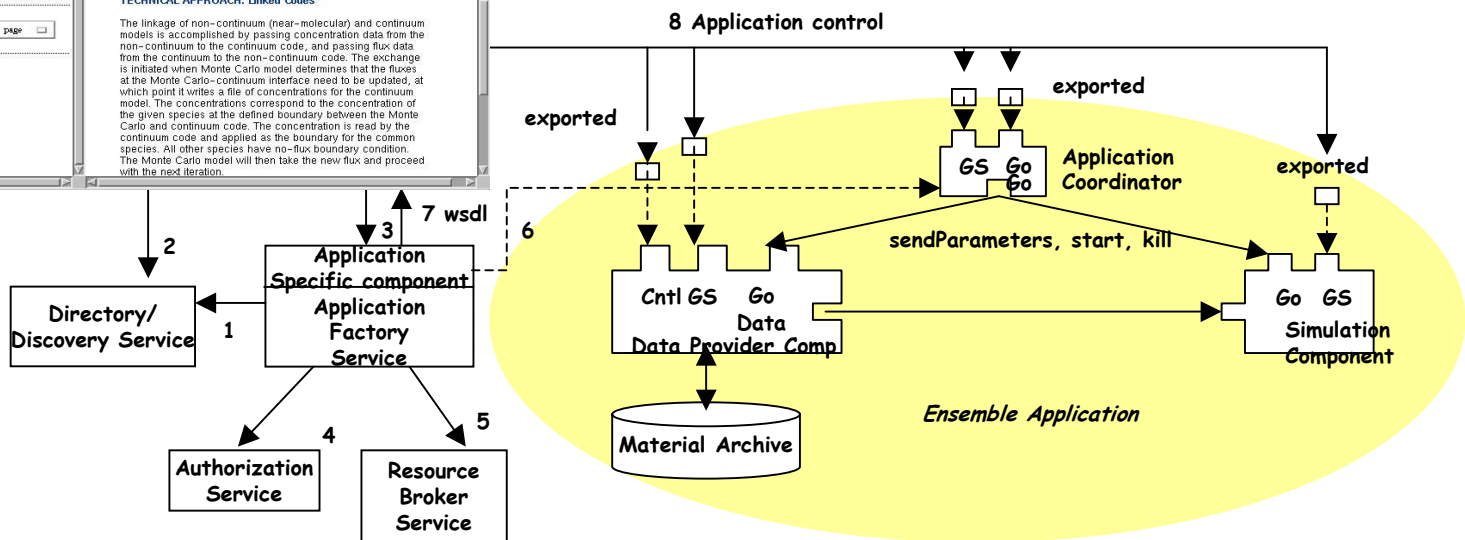
Characteristics

- Distributed/Grid model
- Web services

Achievements

- Developed Proteus multi-protocol communication package
- Novel MxN work at MPI-I/O level





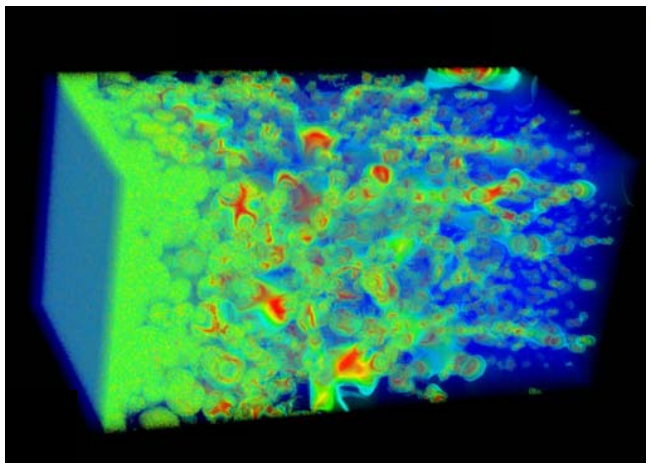
July 2002	July 2003	July 2004	July 2005	July 2006
SCMD build & implementation (SNL)	Generalize SCMD for HPC (SNL)	Improve UI of SCMD (SNL)	Iterate	Iterate
Finish initial distributed framework (UI)	Integrate MS .NET. Initial Grid support (IU)	Beta version of framework (IU,SNL)		Final version (IU,SNL)
XML to access Alexandria	Remote by fram (LLNL)	(LLNL)		
Add Python and Java to Babel (LLNL)	Add Fortran 90 to Babel (LLNL)	Add M to Babel	Add Perl or other to Babel (LLNL)	Add COM and .NET to Babel (LLNL)
Complete concurrency standard (all)	Evolve concurrency (all)			
Develop distributed multiplexer and XML prototype (IU,SNL)	Iterate on SCMD/Grid multiplexer (IU,SNL)			

Uintah

SCIRun/BioPSE/Uintah

Characteristics

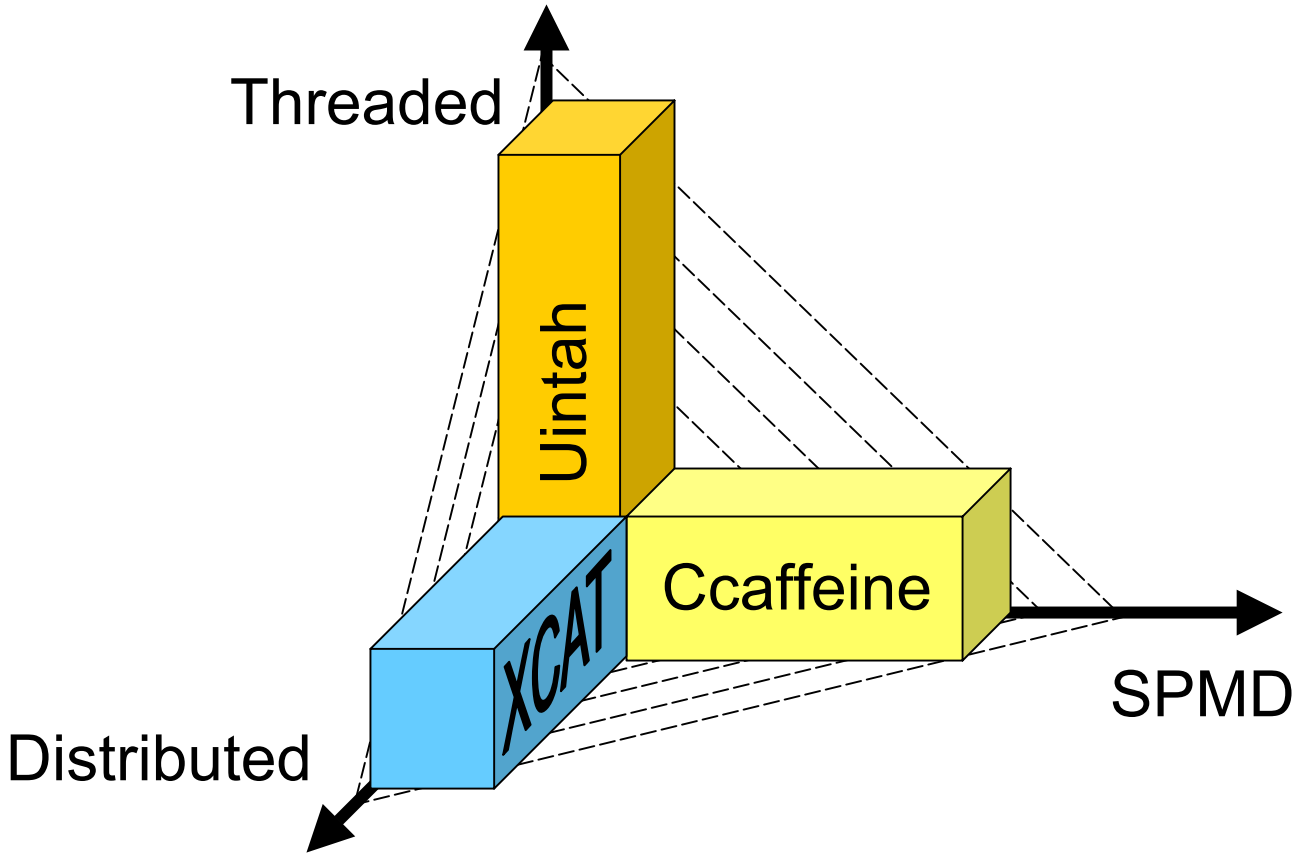
- Multithreaded & distributed
- C++ only
- Used in “real science” (gov’t, academic, commercial) 2K procs



Achievements

- Testbed for CCA Concepts
- Novel work in IDL-based MxN
- Parallelism and concurrency research for CCA done on Uintah
- SCIRun2 will integrate SCIRun, BioPSE and Uintah

Why three CCA Frameworks?



- We address three different types of high-performance scientific computing.

July 2002	July 2003	July 2004	July 2005	July 2006
SCMD build & implementation (SNL)	Generalize SCMD for HPC (SNL)	Improve UI of SCMD (SNL)	Iterate	Iterate
Finish initial distributed framework (I	Integr	Beta version of framework (IU,SNL)		Final version (IU,SNL)
XM	access by frameworks (LLNL)	Deplo Alexan (LLNL)		
Python and Java to Babel (LLNL)	Ad to		Add Perl or other to Babel (LLNL)	Add COM .NET
Complete concurrency standard (all)	Evolve concur			
Develop distributed multiplexer and XML prototype (IU,SNL)	Iterate on SCMD/Grid multiplexer (IU,SNL)			

Ccaffeine-XCAT coupling demonstrated at SC2002

Framework Integration

March 4: SCIRun 2 successfully loaded a SIDL Ccaffeine component without modification

Course Correction: Framework Interoperability, not Integration

- Before
  - We planned a single CCA “reference framework”
- Now
  - We better understand the goals/constraints of different teams
    - SCIRun has commercial clients
    - XCAT needs to experiment and generate papers
    - Ccaffeine supports the base of the CCA community
  - We realize that interoperability is sufficient
    - Frameworks can load same SIDL components
    - Frameworks expose themselves as components to foreign frameworks

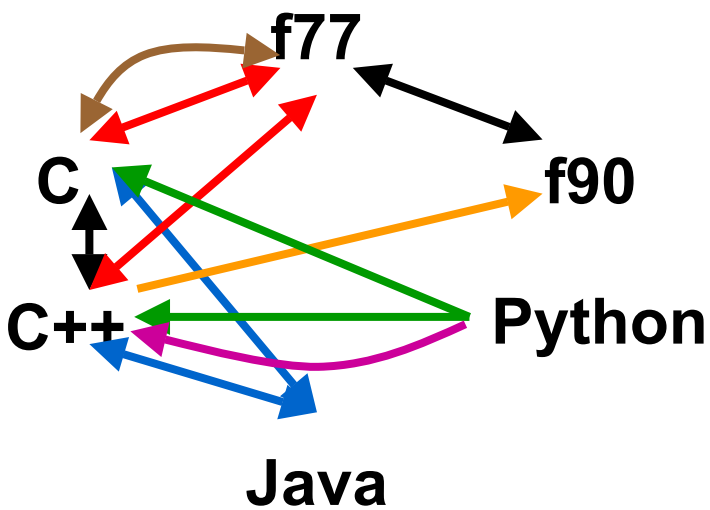
July 2002	July 2003	July 2004	July 2005	July 2006
SCMD build & implementation (SNL)	Generalize SCMD for HPC (SNL)	Improve UI of SCMD (SNL)	Iterate	Iterate
Finish initial distributed framework (UI)	Integrate MS .NET. Initial Grid support (IU)	(IU,SN		Final version (IU,SNL)
XML to access Alexandria	Remote access by frameworks (LLNL)	Deploy Alexa (LLNL)		
Add Python and Java to Babel (LLNL)	Add Fortran 90 to Babel (LLNL)	Add Matlab to Babel (LLNL)	Add Perl or other to Babel (LLNL)	Add COM and .NET to Babel (LLNL)
Complete concurrency standard (all)	Evolve concurrency (all)	Finish server-side Java		
Develop distributed multiplexer and XML prototype (IU,SNL)	Iterate on SCMD/Grid multiplexer (IU,SNL)			

Babel

SIDL/Babel

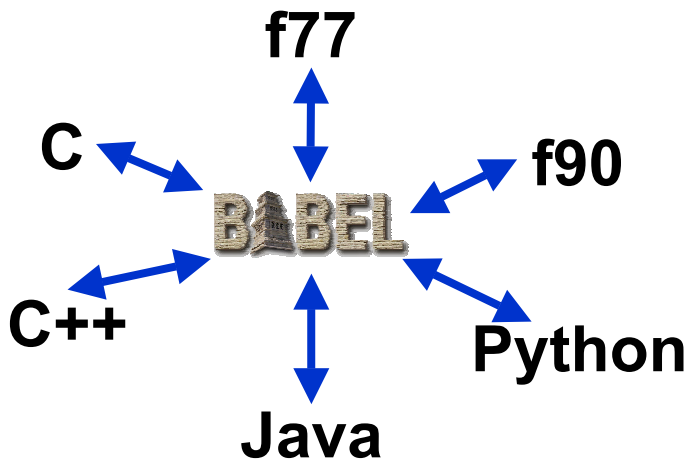
Characteristics

- SIDL: Scientific Interface Definition Language
- Babel: uses SIDL to generate wrappers



Achievements

- CCA specification is in SIDL
- Decaf:
  - 1st Babelized CCA framework (experimental)
  - Code reused in Ccaffeine's SIDL components
- SIDL used in TOPS & TSTT



Course Correction: Open Babel

- Before
  - No CVS access, quarterly releases
  - All development at LLNL
- Now
  - On CCA's critical path
  - CVS access on cca-forum.org
  - babel-dev@llnl.gov includes LLNL team + SNL + Utah + IU + ANL
  - Developer meetings (2x / year)
  - Babel extensions
  - Babel published by Debian & has official Debian maintainer

Course Correction: Fortran 90

- Before
  - Demand for Fortran 90 much higher than anticipated
  - Fortran 77-style binding was inadequate for users
- Now
  - Fortran 90 is Babel's top priority
    - completed original vision 6 months early
    - continuous refining based on community feedback
  - Engaging Fortran 90 developers via cca-fortran@cca-forum.org
  - Leveraging Chasm's technology for Fortran 90 Arrays

July 2002	July 2003	July 2004	July 2005	July 2006
SCMD build & implementation (SNL)	Generalize SCMD for HPC (SNL)	Improve UI of SCMD (SNL)	Iterate	Iterate
Finish initial distributed framework (UI)	Integrate MS .NET. Initial Grid support (IU)	Beta version of framework (IU,SNL)		Final version (IU,SNL)
XML to access Alexandria	Remote access by frameworks (LLNL)	Deploy Alexandria		
Add Python and Java to Babel (LLNL)	Add Fortran to Babel (LLNL)	Babel (LLNL)	rl or other to Babel (LLNL)	Add COM and .NET to Babel (LLNL)
Complete concurrency standard (all)	Evolve concurrency (all)			
Develop distributed multiplexer and XML prototype (IU,SNL)	Iterate on SCMD/Grid multiplexer (IU,SNL)			

Chasm

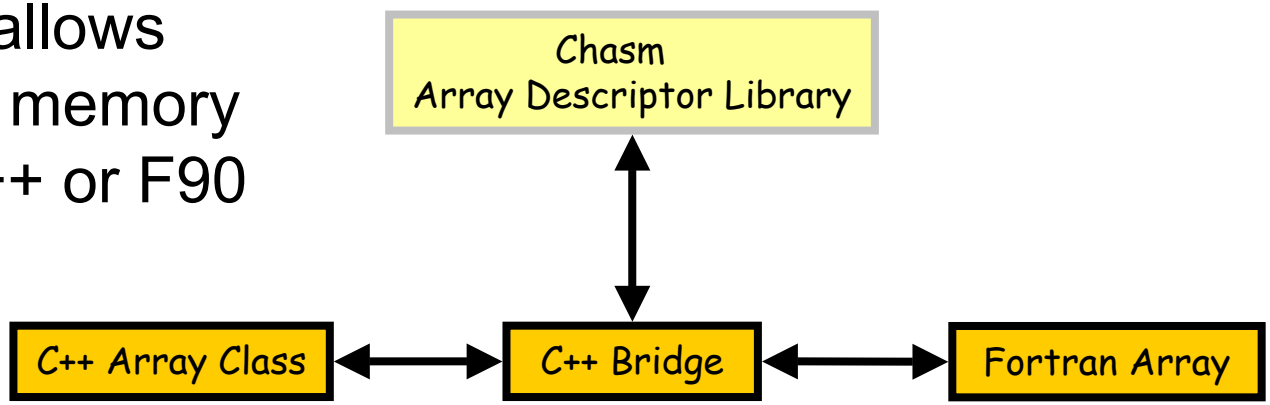
Chasm

Characteristics

- Integral player in CCA language interoperability
- Parses C++/F90 source code to generate wrappers
- Array library allows exchange of memory created in C++ or F90

Contribution/Impact

- Babel will use Chasm's Array Cracking library
- Chasm will generate SIDL



July 2002	July 2003	July 2004	July 2005	July 2006
SCMD build & implementation (SNL)	Generalize SCMD for HPC (SNL)	Improve UI of SCMD (SNL)	Iterate	Iterate
Finish initial distributed framework (LLNL)	Integrate MS .NET. Initial Grid support (IU)	Beta version of framework (LLNL,SNL)		Final version (IU,SNL)
XML to access Alexandria	Remote access by frameworks (LLNL)	Deploy Alexandria (LLNL)		
Add Python and Java to Babel (LLNL)	Add Fortran 90 to Babel (LLNL)	Add [illegible] to Babel [illegible]L)	Add Perl or other to Babel (LLNL)	Add COM and .NET to Babel (LLNL)
Complete concurrency standard (all)	Evolve concurrency (all)			
Develop distributed multiplexer and XML prototype (IU,SNL)	Iterate on SCMD/Grid multiplexer (IU,SNL)			

Alexandria

Component Packaging and Deployment

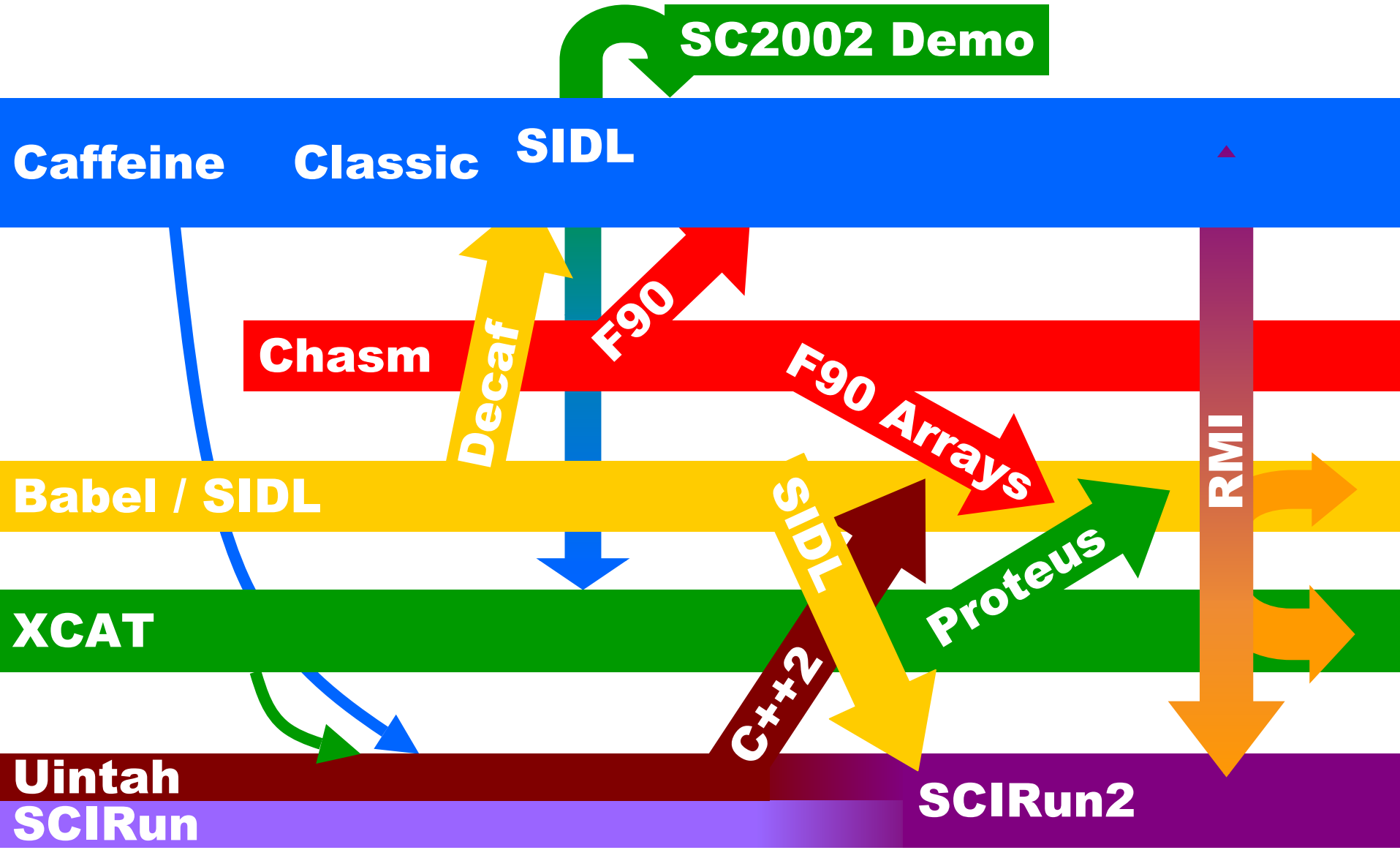
Characteristics

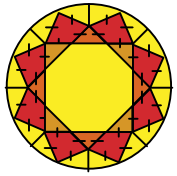
- Source code distribution of components
  - Hard problem!
  - Not done in commercial world
- Long term:
  - XML-based specification of components, build info, meta-data
  - Alexandria repository to handle human and automated requests for components

Achievements

- Short term:
  - RPMs of Babel, Ccaffeine, and various components (more info in 'scientific components' section)

Cross Fertilization





CCA
Common Component Architecture

Scientific Components

Coordinator:
Lois Curfman McInnes (ANL)

Sandia National Laboratories
Lawrence Livermore National Laboratory
ornl Oak Ridge National Laboratory
The University of Utah
Los Alamos National Laboratory
Pacific Northwest National Laboratory
Operated by Battelle for the U.S. Department of Energy
Argonne National Laboratory, University of Chicago
Office of Science
U.S. Department of Energy

Approach

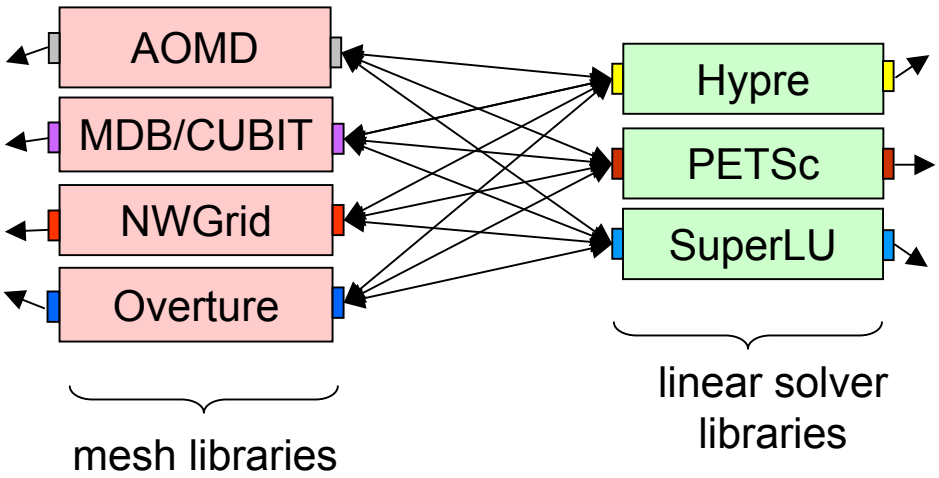
- Define domain-specific common interface specifications
- Develop high-performance scientific components
- Explore related research issues of importance in the DOE computational science environment

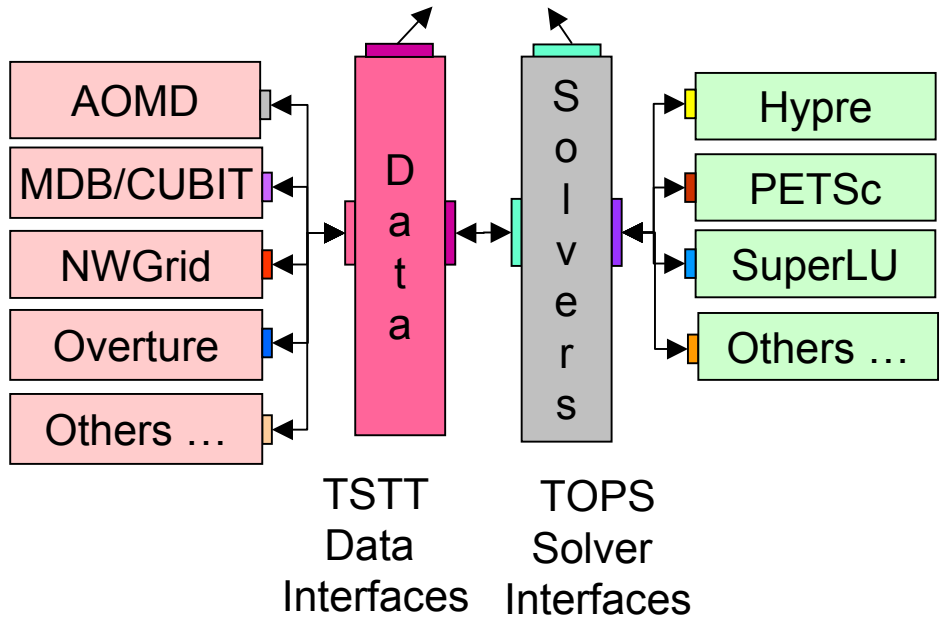
This is an iterative process, in collaboration with scientists outside of the CCTTSS, including

- SciDAC applications groups
- Other SciDAC Integrated Software Infrastructure Centers (ISICs)
- The high-performance scientific computing community in general

Motivation for Common Interfaces

- ***Many-to-Many* couplings require *Many* ^{2} interfaces**
  - Often a heroic effort to understand details of both codes
  - Not a scalable solution
- **Common Interfaces: Reduce the *Many-to-Many* problem to a *Many-to-One* problem**
  - Allow plug-and-play interchangeability & interoperability
  - Require domain specific experts
  - Typically difficult & time-consuming
  - A success story: MPI
  - Challenges
    - Interface agreement
    - Functionality limitations
    - Maintaining performance





Current Interface Development Activities

CCA Scientific Data Components Working Group

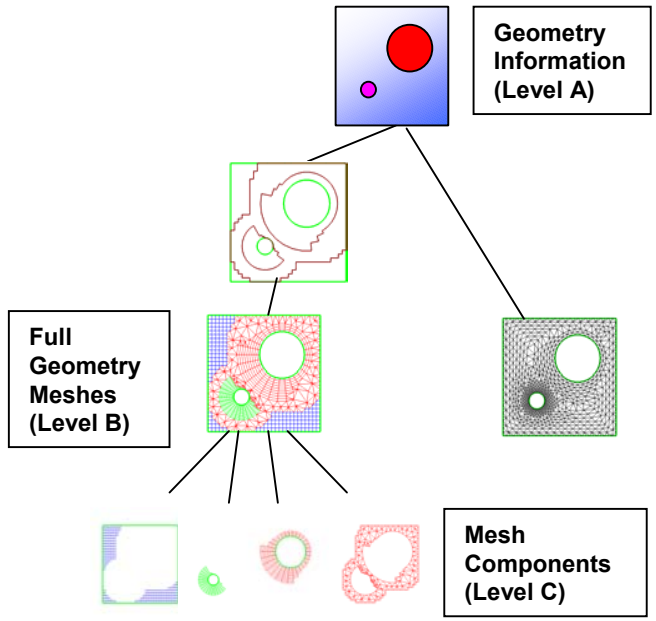
- **Basic Scientific Data Objects**
  - Lead: D. Bernholdt (ORNL)
- **Unstructured Meshes**
  - Lead: L.F. Diachin (SNL, formerly ANL)
  - Collaboration with TSTT ISIC
  - TSTT = Terascale Simulation Tools and Technologies, PIs: J. Glimm, D. Brown, L.F. Diachin, http://www.tstt-scidac.org
- **Structured Adaptive Mesh Refinement**
  - Lead: P. Colella (LBNL)
  - Collaboration with APDEC ISIC
  - APDEC = Algorithmic and Software Framework for Applied PDEs, PI: P. Colella, http://davis.lbl.gov/APDEC

Other Groups

- **Linear and nonlinear solvers, eigensolvers, and optimizers**
  - Coordinator: L.C. McInnes (ANL)
  - Collaboration with TOPS ISIC
  - TOPS = Terascale Optimal PDE Simulations, PI: D. Keyes, http://tops-scidac.org
- **MxN Parallel Data Redistribution**
  - Lead: J. Kohl (ORNL)
  - Part of CCTTSS MxN Thrust
- **Quantum Chemistry**
  - Leads: C. Janssen (SNL) and T. Windus (PNNL)
  - Part of CCTTSS Applications Integration Thrust

A Case Study: The TSTT/CCA Mesh Interface

- Goal is to enable
  - Hybrid solution strategies
  - High-order discretization
  - Adaptive techniques
- Approach
  - Create small set of interfaces that existing packages can support
  - Balance performance and flexibility
  - Work with tool developers and application community to ensure applicability

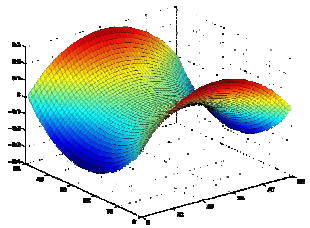


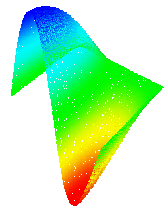
- Required interfaces
  - Entity queries (geometry, adjacencies), entity iterators, array-based query, workset iterators, mesh/entity tags, mesh services

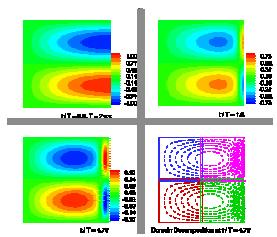
TSTT: Issues that have arisen …

- Nomenclature is harder than we first thought
- Cannot achieve the 100 percent solution, so...
  - What level of functionality should be supported?
    - Minimal interfaces only?
    - Interfaces for convenience and performance?
  - What about support of existing packages?
    - Are there atomic operations that all support?
    - What additional functionalities from existing packages should be required?
  - What about additional functionalities such as locking?
- Language interoperability is a problem
  - Most TSTT tools are in C++, while most target applications are in Fortran
  - How can we avoid the “least common denominator” solution?
  - Exploring SIDL/Babel for language interoperability

Scientific Components and Applications







- Recognized as one of the "Top Ten Science Achievements in 2002" by the DOE Office of Science (see http://www.sc.doe.gov/sub/accomplishments/top_10.htm)
- Many demonstrated at SC2001 and SC2002
- Combine application-specific components with more general-purpose components that can be reused across a range of applications
  - More than 40 components, many reused in apps such as
    - PDEs on unstructured and adaptive structured meshes
    - Unconstrained minimization problems
- Leverage and extend parallel software developed at different institutions
  - Including CUMULVS, CVODES, Global Arrays, GrACE, MPICH, PETSc, PVM, and TAO
- Download source code & documentation: http://www.cca-forum.org/software.html
- Component Inventory: See Appendix A of progress report
  - Current snapshot; all components are evolving

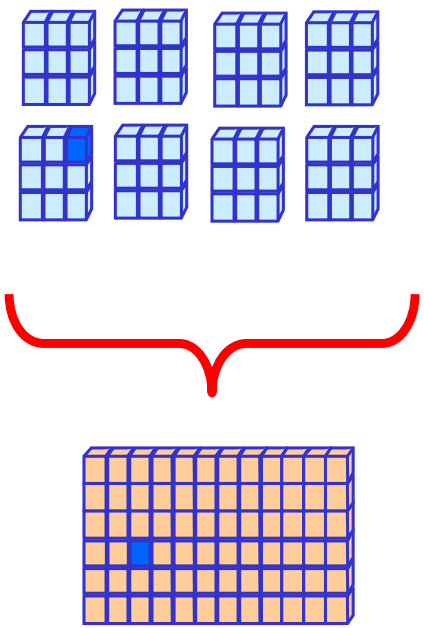
Component Inventory:

Data Management, Meshing and Discretization

- ***Global Array Component*** – M. Krishnan and J. Nieplocha (PNNL) – classic and SIDL – Provides capabilities for manipulating multidimensonal 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.

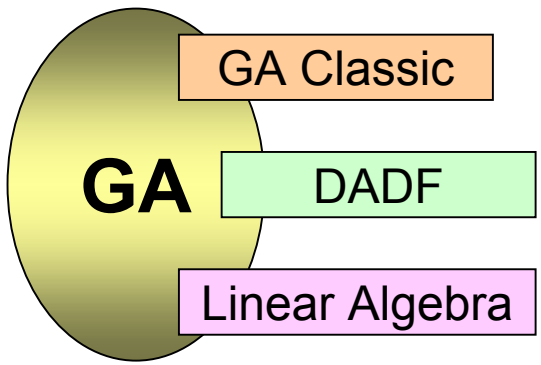
A Component Example Close-up: GAComponent

physically distributed dense array



single, shared data structure
global indexing

- **Global Arrays**: Efficient and portable shared-memory programming environment for distributed-memory computers, supporting dense distributed arrays.
- **GAComponent**: Classic and SIDL Interfaces
- 36+98 (direct+indirect) global arrays classic methods are available through ***GAClassicPort***
- ***GADADFPort*** provides methods, proposed by the CCA Scientific Data Working Group for creating array descriptors and templates



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

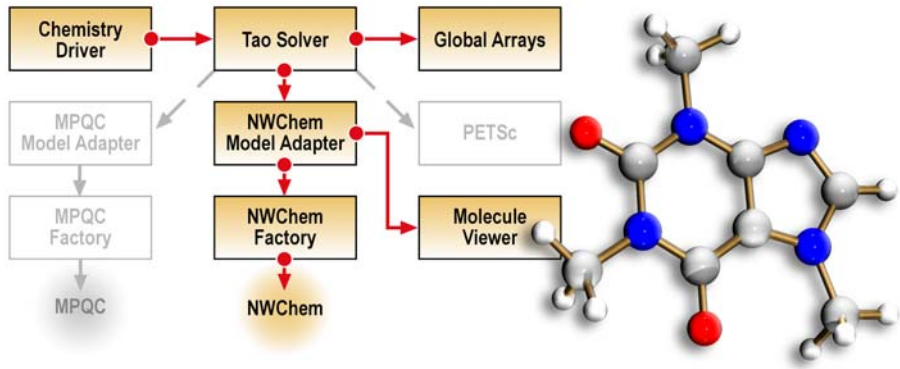
- ***DistArrayDescriptorFactory*** – D. Bernholdt and W. Elwasif (ORNL) – classic – Provides a uniform means for applications to describe dense multi-dimensional arrays; based upon emerging interfaces from the CCA Scientific Data Components Working Group.
- ***CumulvsMxN*** – J. Kohl, D. Bernholdt, and 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

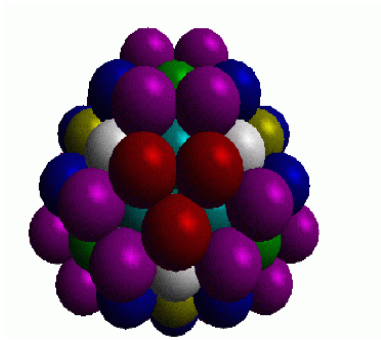
- ***Ccaffeine Services*** – B. Allan, R. Armstrong. M. Govindaraju, S. Lefantzi, and E. Walsh (SNL) – classic and SIDL – Services for parameter ports, connections between SIDL and classic ports, MPI access, connection events, etc.
- ***Graphical Builders*** – B. Norris (ANL) and S. Parker (Univ. of 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).

GA and TaoSolver Components:
Interoperability, Performance, and Applications

- **GA/TaoSolver Interoperability**
  - GA provides core linear algebra for manipulating vectors, matrices, and linear solvers through a ***LinearAlgebraPort***; these can be used by TaoSolver

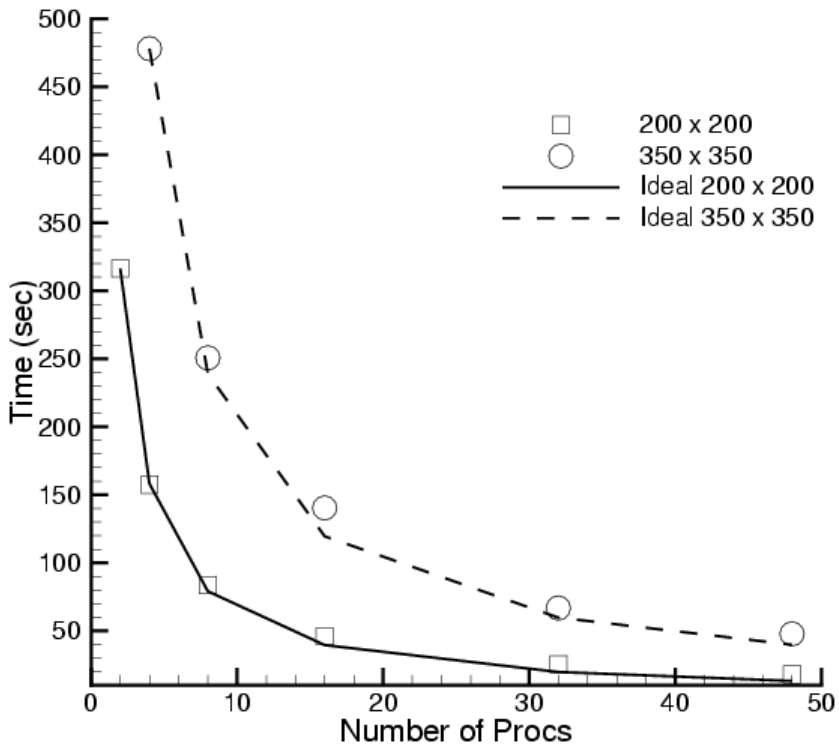


- **Applications Usage**
  - Lennard-Jones molecular dynamics
    - Uses force decomposition method and dynamic load balancing
    - Component overhead is negligible (<1%)
    - Good scaling (simulation of 12,000 atoms yields a speedup of 7.86 on 8 processors of PNNL Linux cluster)
  - Molecular geometry optimization
    - Ongoing collaboration, including SC2002 demo
    - Electronic structure components based on NWChem (PNNL) and MPQC (SNL) for energy, gradient, and Hessian computations
    - Optimization components based on TAO (ANL)
    - Linear algebra components based on GA (PNNL) and PETSc (ANL)
    - Future work: Refine interfaces, examine performance issues, explore more complex problems



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



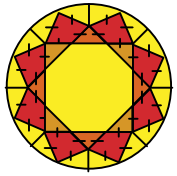
Reference: S. Lefantzi, J. Ray, and H. Najm, Using the Common Component Architecture to Design High Performance Scientific Simulation Codes, *Proc of Int. Parallel and Distributed Processing Symposium*, Nice, France, 2003, accepted.

Collaborations with the TSTT, APDEC, and TOPS SciDAC Centers

- **Approach:** Define common interfaces and component implementations
  - **Goal:** These can eventually be employed by scientists who collaborate directly with these mathematics ISICs
- **TSTT and APDEC Centers**
  - **Focus:** Collaborating with the CCA Scientific Data Working Group to define interfaces for accessing mesh geometry and topology information
  - **Current status:** Prototype TSTT component implementations are used in simple PDE-based examples and CCA tutorials
  - **Future work:** Ongoing further development and testing with various apps
- **TOPS Center**
  - **Focus:** Collaborating to define interfaces for linear and nonlinear solvers, eigensolvers, and unconstrained and constrained optimizers
  - **Current status:** Have explored early prototype interfaces that use SIDL as the interface definition language
  - **Future work:** Next will experiment with these interfaces in various apps, including a SciDAC fusion simulation under development by the Center for Magnetic Reconnnection Studies (CMRS, PI: A. Bhattacharjee)

Future Work Summary

- Cross-cutting integration among various SciDAC centers
  - Collaborations with TSTT, APDEC, and TOPS ISICs
  - Collaborations with SciDAC applications in chemistry, climate, fusion, etc.
- Evaluate, extend, and refine first-generation components
  - Complete development of abstract interfaces and component prototypes
  - Develop new component functionality
    - multi-threading, load redistribution, computational steering, fault tolerance, etc.
  - Develop advanced components, including
    - multi-level nonlinear solvers
    - hybrid mesh management schemes
    - application-specific components for chemistry, combustion, and climate
- Investigate Quality-of-Service (QoS) issues for numerical components
  - In collaboration with the PERC ISIC
  - Recently started preliminary work to identify motivating scenarios, analyze requirements, develop a catalog of QoS metrics for numerical components, and design a QoS architecture



CCA
Common Component Architecture

"MxN" Parallel Data Redistribution

Coordinator: James Kohl (ORNL)

David Bernholdt (ORNL), Randy Bramley (Indiana),

Pat Fasel (LANL), Kate Keahey (ANL, formerly LANL),

Jay Larson (ANL), Sue Mniszewski (LANL),

Steven Parker (Utah), Reid Rivenburgh (LANL)

Sandia National Laboratories

Lawrence Livermore National Laboratory

ORNL Oak Ridge National Laboratory

The University of Utah

Los Alamos National Laboratory

Pacific Northwest National Laboratory
Operated by Battelle for the U.S. Department of Energy

Argonne National Laboratory, University of Chicago

Office of Science

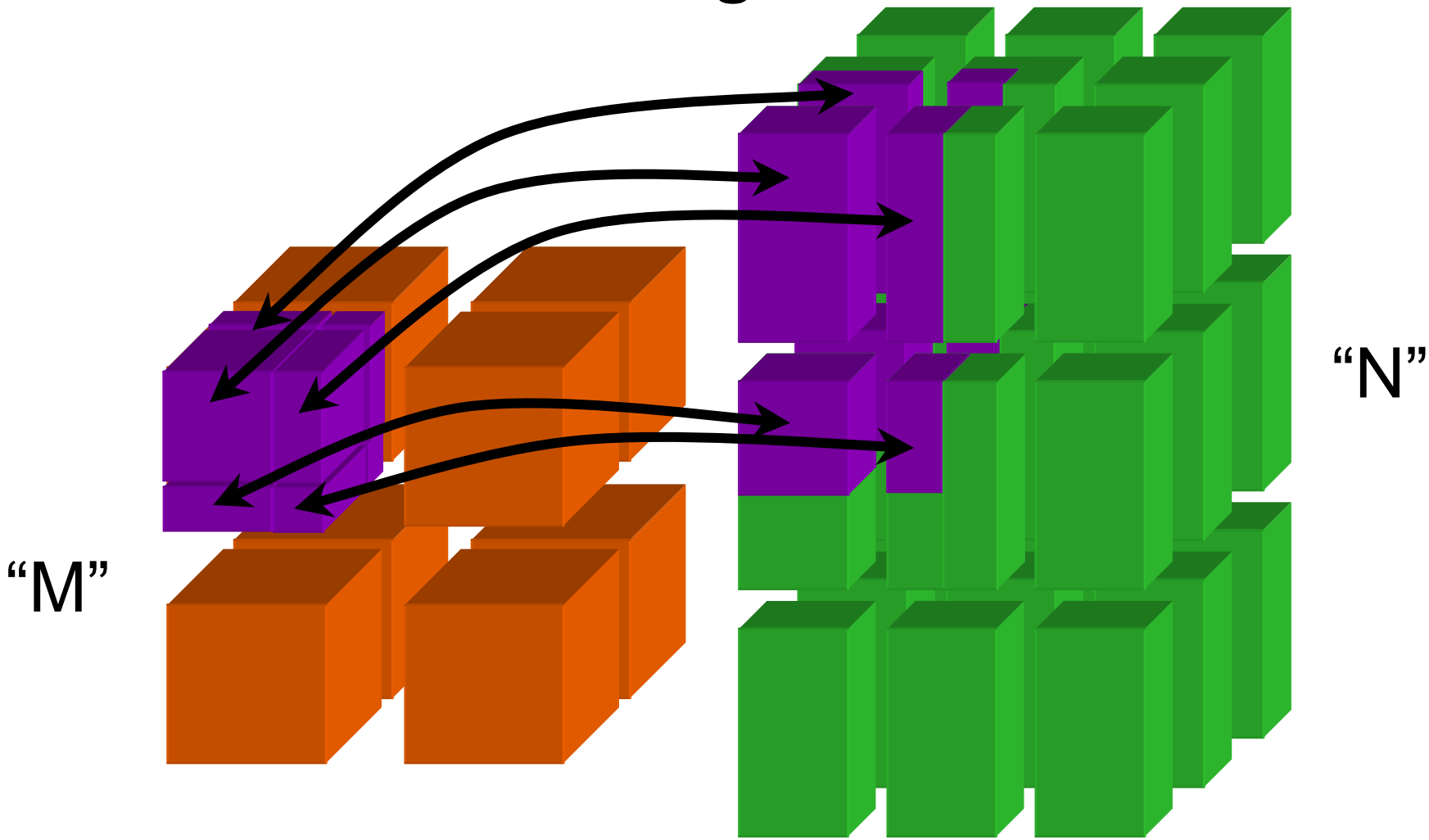
U.S. Department of Energy

Cooperating Parallel Components…?!

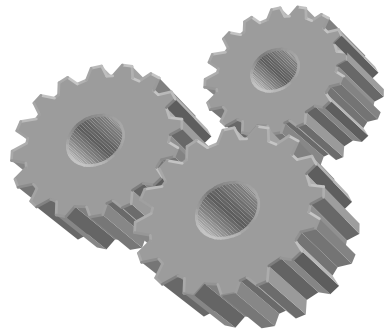
- CCA-specific capability
- Requirement of emerging scientific apps
  - Generalized model coupling
  - Efficient daisy-chaining of numerical libraries
  - Visualization (parallel rendering / serial gathering)
- Fundamental issue ~ data decompositions
  - Different distribution for each parallel component
  - Need to reconcile disparate data organizations
- And, more than just parallel data …

  … Parallel method invocations, too!

The "Basic" Problem: Parallel Data Exchange



Current MxN Capabilities



- Parallel data exchange operations
  - Describe and register data / decompositions
  - Map communication schedules
  - Build MxN connections
    - Various synchronization options
  - Initiate data transfers, per parallel instance
    - Local dataReady() method
- Foundation for generalized model coupling
  - Spatial and temporal interpolation
  - Units conversion …

MxN Progress to Date…

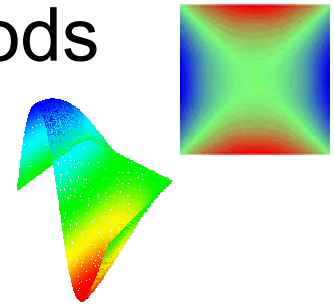
- Generalized MxN specification
  - Built on CUMULVS (ORNL) and PAWS (LANL)
- SCMD peer MxN component solution
  - Visualization, combustion and climate apps …
- Parallel Remote Method Invocation (PRMI)
  - Preliminary semantics and specifications
- Distributed MxN solution
  - MPI I/O approach

MxN Interface Specification

- Generalizes many existing tools
  - Primarily CUMULVS (ORNL) and PAWS (LANL)
    - Also “DataCutter” at U. Maryland and others
  - Several synchronization models subsumed
    - Point-to-point send & receive (PAWS)
    - Persistent channel w/periodic transfers (CUMULVS)
  - Dense, rectangular data distributions supported
    - A la HPF ~ e.g. block, cyclic, collapse
    - Some unstructured mesh experiments (incl. triangular)
- Several interface evolutions (ongoing)
  - Reconciling the appropriate level of detail & flexibility
  - End result surpasses original tools

MxN "Explicit" Component Solution

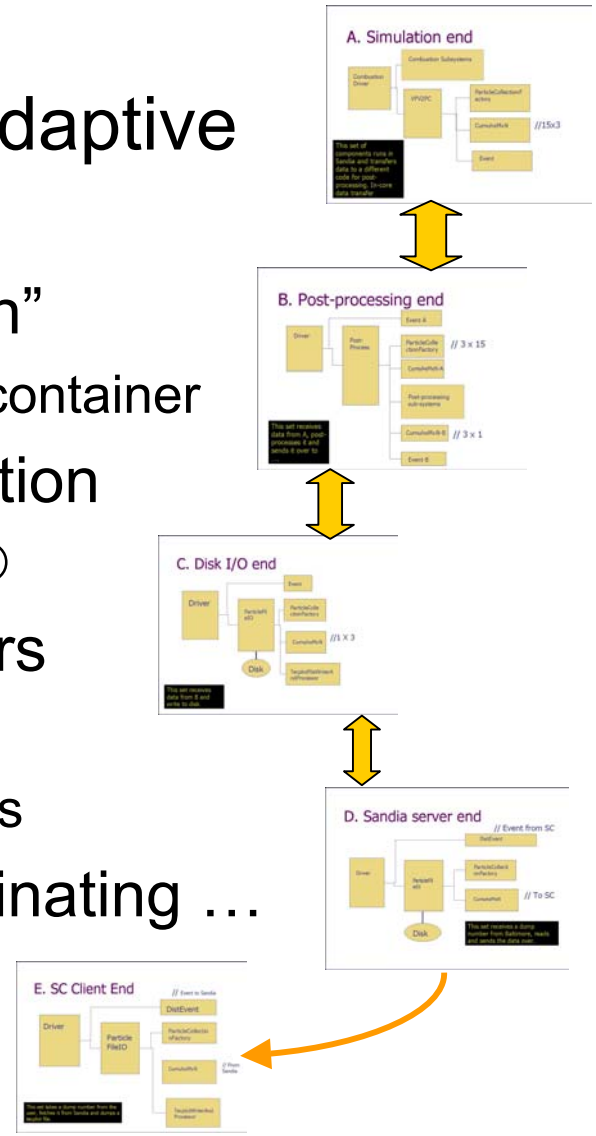
- Port-based direct invocation of MxN methods
  - Most general solution, but …
  - Most challenging to the end-user scientist
    - "Assembly language" level interface …
    - Preliminary platform for experimentation, higher-level functs



- Several demonstrations (SC2001, SC2002)
  - Increasingly elaborate visualization solutions
    - Incorporated generalized data description since SC2001 (DADF)
  - Two main prototypes (reused in several apps)
    - CumulvsMxN ~ first inter-framework component capabilities!
    - PawsMxN ~ ping-pong coupling experiments

MxN and Combustion (CFRFS)

- No coupling needs, yet useful for adaptive chemistry and post-processing…
  - Replace DADF with “ParticleCollection”
    - Map adaptive mesh patches to “Particle” container
  - Interactively offload data for simplification
    - “Real” parallel-to-parallel MxN usage… ☺
  - Connect several computational clusters
    - Plus front-end visualization node(s)
    - Glues together multiple SCMD frameworks
  - Initial “CumulvsMxNp” prototype culminating …



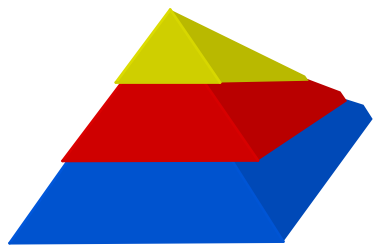
MxN and Climate (MCT / CCSM)

- Model Coupling Toolkit (MCT - ANL)
  - Climate-specific coupling models and technology
  - Amenable to generalized MxN specification
  - Two-way integration underway
    - Re-package MCT as components
    - Build MxN component on top of MCT

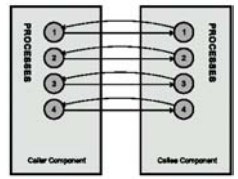
- Ongoing reconciliation of terms and concepts
  - Between MxN and MCT, and CCSM and ESMF…
- Componentization of CCSM models
  - Atmosphere, ocean, sea-ice, land-surface, river-runoff, plus flux couplers …

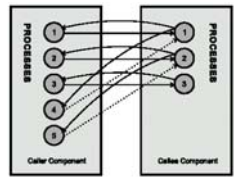
MxN Future Work ~ Implicit Solutions

- Need simpler, high-level interfaces
  - For the non-expert … even automated handling
- Target built-in framework services
  - Capture method invocations via port indirection
  - Implicitly apply MxN functions to reconcile parallel data in method arguments & results
- Increases framework complexity
  - Use pluggable service registration!
  - Requires additional method specifications …

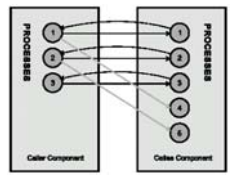


Parallel Remote Method Invocation (PRMI)

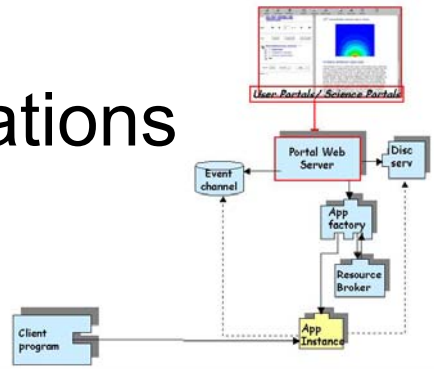




- Next step beyond “simple” data exchange
  - Method itself has parallel context
  - Specification of semantics and policies is key!
- Preliminary PRMI progress:
  - PAWS prototype and early policy identification
    - Invocation scheduling, marshalling arguments & results
  - 2nd SCIRun prototype explores method specification
    - SIDL extensions for “independent” and “collective” methods
    - With sub-grouping, generalizes PRMI invocation semantics
- Still much research ahead …
  - Transport mechanisms (SOAP, etc.)
  - Parallel data argument and results specifications

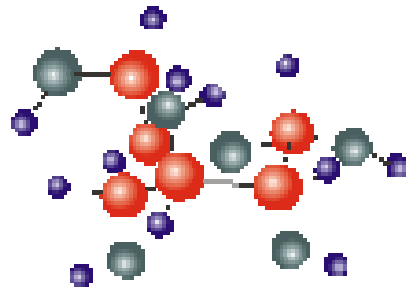


Distributed MxN Scenarios



- Large body of distributed HPC applications
  - E.g., sensor networks…
- SCMD MxN solutions incompatible…
  - No co-location of components in distributed world
  - Fundamental differences in connection semantics
- Early exploration (Indiana) ~ MxN via MPI-I/O
  - Stream-based communication paradigm
    - Analog to file-based exchanges in serial codes
    - Parallelization of data transfers hidden internally
  - Eases integration of existing applications
    - Also useful for unit testing of application components

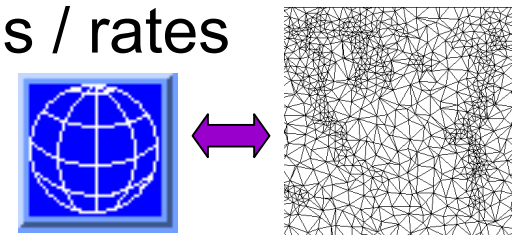
Distributed MxN Future Work



- Construct full prototype
  - Using ROMIO MPI-I/O (e.g., MPICH and LAM)
  - Enables support for derived data types
- Assemble full MxN component solution
  - Mediates data transfers among MPI programs
- Scalability tests
  - Between 2 large clusters (200 CPUs each)

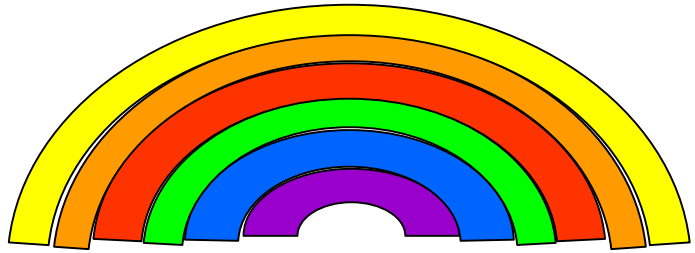
Model Coupling Future Work

- MxN parallel data redistribution is just the beginning of real model coupling …
  - Fundamental data exchange technology
- Need interpolation and data conversion
  - Spatial ~ different meshes & coordinate spaces
  - Temporal ~ different time frames / rates
  - Flux Conservation
  - Units conversion

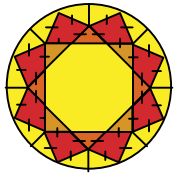


- Must explore composing “filters” with MxN
  - Pipeline efficiency and “super-components”

MxN Summary



- Stable specification and component solutions
  - For visualization, emerging combustion and climate
  - Next step ~ implicit framework services
- Parallel Remote Method Invocation (PRMI)
  - Initial semantics being defined, much to do
  - High-level handling of MxN transfers
- Distributed MxN experiments
  - Prototypes using MPI-I/O
- Tip of the iceberg for production model coupling
  - Need development of suite of interpolation “filters”



CCA
Common Component Architecture

User Outreach and Applications Integration

Coordinator:
David E. Bernholdt (ORNL)

Sandia National Laboratories
Lawrence Livermore National Laboratory
ornl Oak Ridge National Laboratory
The University of Utah
Los Alamos National Laboratory
Pacific Northwest National Laboratory
Office of Science

Approach

- Education and General Outreach
  - Tutorials
  - Talks
  - Papers
  - Other outreach activities
- Applications
  - “Internal” application focus efforts
    - Quantum Chemistry
    - Climate Modeling
  - Other applications

The CCA Tutorial

- CCA Forum Tutorial Working Group
  - Rob Armstrong (SNL), David Bernholdt (ORNL, chair), Lori Freitag Diachin (SNL), Wael Elwasif (ORNL), Dan Katz (JPL), Jim Kohl (ORNL), Gary Kumfert (LLNL), Lois Curfman McInnes (ANL), Boyana Norris (ANL), Craig Rasmussen (LANL), Jaideep Ray (SNL), Torsten Wilde (ORNL)
- Eight modules:
  - Introduction to Components
  - CCA Concepts
  - A Simple Component Example
  - Language Interoperability using Babel
  - Writing CCA Components
  - Introduction to the Ccaffeine Framework
  - More Complex Component-Based Applications
  - CCA Status and Plans

- Simple numerical integration example (software)

Tutorial Presentations

Jan 2002	CCA Forum Mtg	Santa Fe, NM
Apr 2002	CCA Forum Mtg	Townsend, TN
Jun 2002	CCA Forum Mtg	ANL
Sep 2002	ACTS Collection Workshop	LBL
Oct 2002	CCA Forum Mtg	Half Moon Bay, CA
Oct 2002	LACSI Symposium	Santa Fe, NM
Nov 2002	SC2002	Baltimore, MD
Jan 2003	CCA Forum Mtg	Pasadena, CA

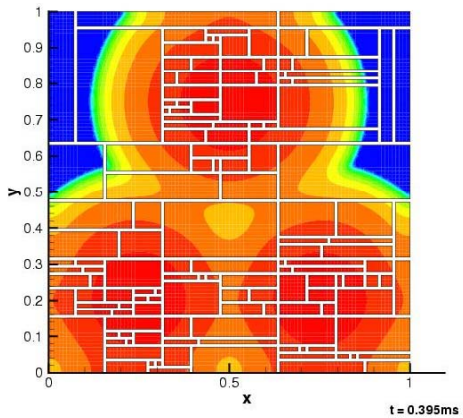
Publications, Presentations, and Other Activities

- More than 80 presentations and papers
  - Includes numerous presentations at organized meetings/workshop of prospective users
  - National and international conferences in computer science and other domains
  - Journal publications
- Other outreach activities
  - Special meetings with (prospective) users
  - Organized symposia at major meetings
  - SC2001, SC2002

Outreach – Future Plans

- Continue tutorials based on demand
- Consider offering tutorials over Access Grid
- Emphasize publications and written documentation
- Develop “best practices” guidelines

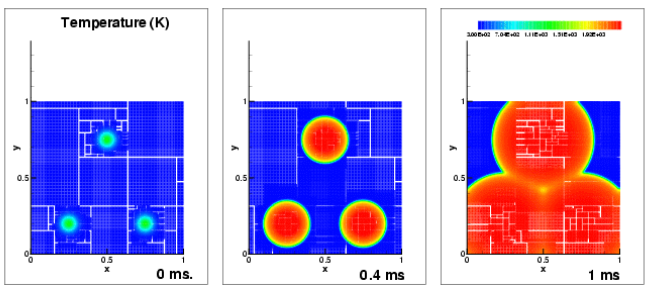
Computational Facility for Reacting Flow Science (CFRFS)



- 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
- **Interaction Model:** Key investigator (Ray) with joint CCA/CFRFS affiliation

Scientific and Technical Summary

- H_2-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)

Impact of CCA

- Basis of “plug-and-play” toolkit environment
  - Would otherwise have been a complex software problem
- Rapid, focused development
  - Second-generation components for higher accuracy and stabilized numerics developed in 4 months
  - Swapped in without effecting remainder of application
- Reuse of external software: TAU, CVODES

Future Plans

- New convective physics capabilities
- New adaptive mesh numerical schemes

Computational Chemistry

- **Molecular Optimization**
- **Investigators:** Steve Benson (ANL) Curtis Janssen (SNL), Liz Jurris (PNNL), Manoj Krishnan (PNNL), Lois McInnes (ANL), Jarek Nieplocha (PNNL), Boyana Norris (ANL), Jason Sarich (ANL), Theresa Windus (PNNL)
- **Goal:** Demonstrate interoperability between packages, develop experience with large existing code bases, seed interest in Chemistry domain
- **Problem Domain:** Optimization of molecular structures using quantum chemical methods
- **Interaction Model:** "Internal" project of CCA and TOPS researchers

- **Advanced Software for the Calculation of Thermochemistry, Kinetics, and Dynamics**
- SciDAC BES project, Al Wagner PI
- **Investigators:** Ronald Duchovic (Indiana-Purdue Fort Wayne), Wael Elwasif (ORNL), Lois McInnes (ANL), Craig Rasmussen (LANL)
- **Goal:** Develop a standard interface to provide numerical potential energy surface information to reaction dynamics codes
- **Problem Domain:** Reaction dynamics
- **Interaction Model:** Consultation

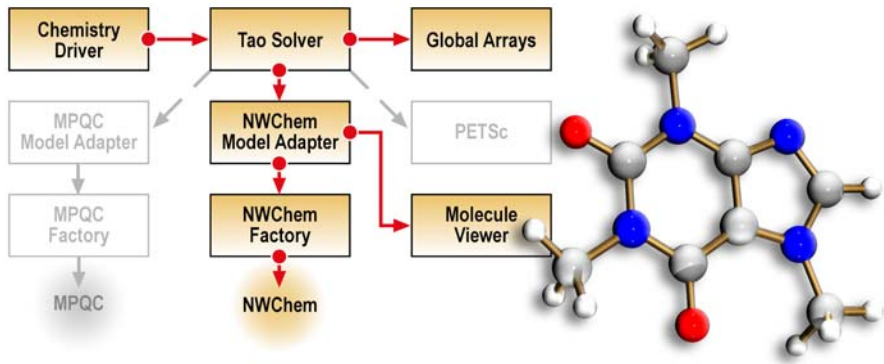
Scientific and Technical Summary

Molecular Optimization

- Decouple geometry optimization from electronic structure
- Demonstrate interoperability of electronic structure components
- Build towards more challenging optimization problems, e.g., protein/ligand binding studies
- Software:
  - Electronic structure: MPQC, NWChem,
  - Optimization:TAO
  - Linear algebra: Global Arrays, PETSc

Reaction Dynamics

- Software: POTLIB
- Developing a component interface for POTLIB
  - Original interface makes extensive use of Fortran common blocks



CCA Impact

- Demonstrated unprecedented interoperability in a domain not known for it
- Demonstrated value of collaboration through components
- Gained experience with several very different styles of “legacy” code

Future Plans

- Extend to more complex optimization problems
- Extend to deeper levels of interoperability
- Work towards merging interfaces

Climate Modeling

CCSM

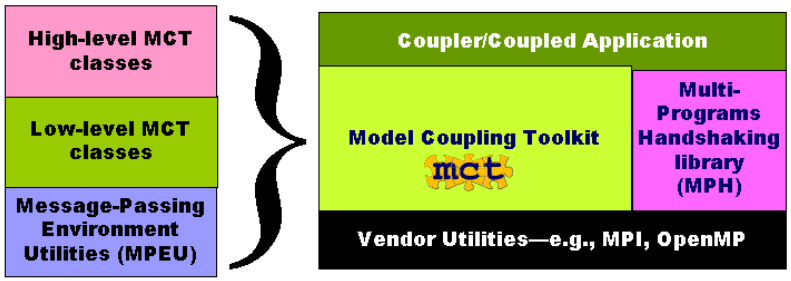
- SciDAC BER project, John Drake and Robert Malone PIs
- **Goals:** Investigate model coupling and parameterization-level componentization within models

ESMF

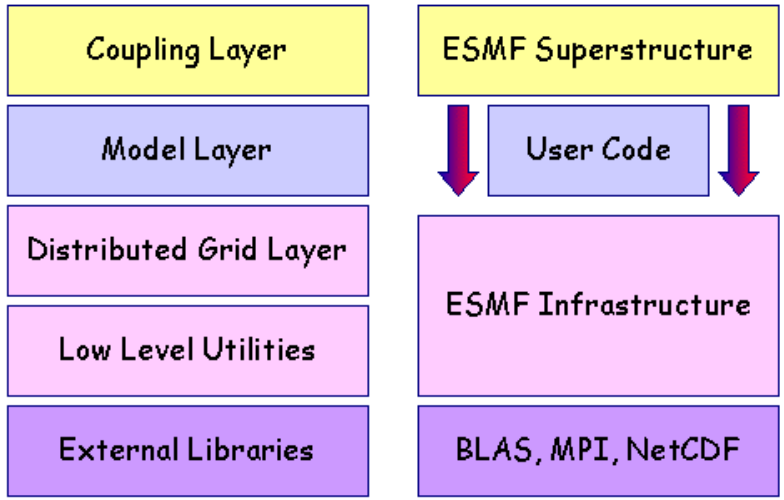
- NASA project, Tim Killeen, John Marshall, and Arlindo da Silva PIs
- **Goal:** Build domain-specific framework for the development of climate models

- **Investigators:** John Drake (ORNL), Wael Elwasif (ORNL), Michael Ham (ORNL), Jay Larson (ANL), Everest Ong (ANL)
- **Interaction Model:** Key researchers with joint affiliation: Larson (CCA, CCSM, ESMF), Ham (CCA, CCSM)

Scientific and Technical Summary



- 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



- ESMF
  - Prototype superstructure
  - Investigating grid layer interfaces
- Throughout: F90 critical
  - Babel, Chasm

CCA Impact

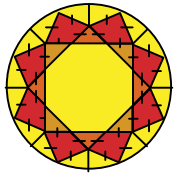
- Time-critical opportunity to influence construction of ESMF
- Two-way exchange on model coupling
- Introducing components in conjunction with CAM refactorization

Future Plans

- Continue componentization of MCT
- Continue componentization of CAM
- Continue interactions with ESMF
- Small adjustments to deliverables

Other Applications

- CCA is gaining awareness and visibility
- We've interacted with many groups; adoption is up to them
- We expect that many will migrate to CCA to use forthcoming tools from Math ISICs
- "Serious" applications available now and in near future will help convince others that CCA is usable
- Interactions expected to shift toward increasing consultation



CCA
Common Component Architecture

Conclusion

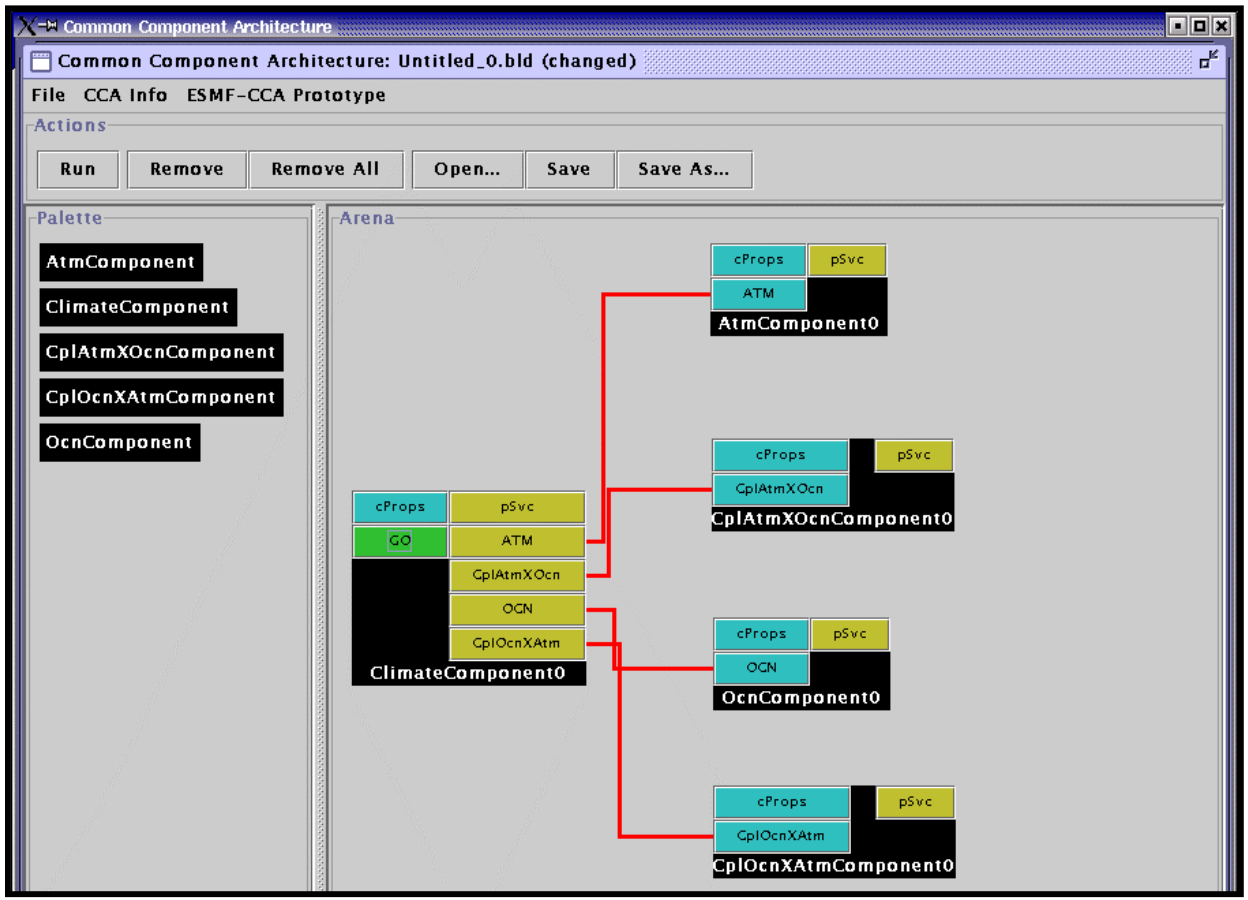
Sandia National Laboratories
Lawrence Livermore National Laboratory
ornl Oak Ridge National Laboratory
The University of Utah
Los Alamos National Laboratory
Pacific Northwest National Laboratory Operated by Battelle for the U.S. Department of Energy
Argonne National Laboratory University of Chicago
Office of Science
U.S. Department of Energy United States of America

Current Impact of CCA in SciDAC

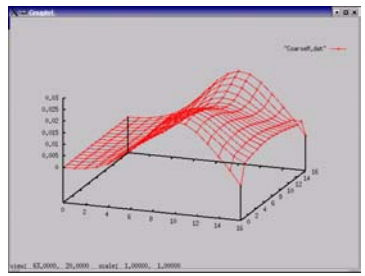
- Chemistry and combustion applications
  - In active use by the investigators for their research
- Catalyst for interoperability
  - Interfaces from APDEC, TOPS, TSTT imported to applications (Combustion and other apps)
  - Climate: MCT interfaces CCA'ified into CCSM
  - Open Babel, Chasm, etc.: language interoperability for developers and users
- Successful education and outreach
  - Tutorials and manuals

CCA Impact Outside of SciDAC

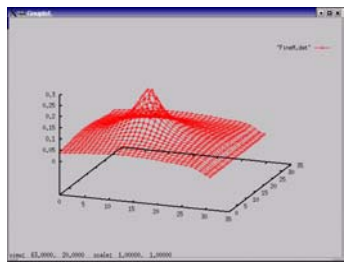
- Climate prototype (Courtesy Shujia Zhou, NASA Goddard)



ATM



OCN



CCA in the SciDAC Future: Research in Service of the HPC Community

- The Climate Year
  - Emphasize climate applications
  - Community Climate Simulation Model (CCSM) and the greater climate community
- The Fortran Year
  - Many very cool ideas were developed last year
  - And are now being applied to climate and other applications
- Research in
  - MxN: cooperating but separate parallel decompositions
  - QoS: balancing resources for numerical components
  - Conceptual design patterns: abstracted interoperabilty
- More and different outreach
  - Greater emphasis on papers and meetings
  - Ongoing tutorials