

High-Performance Parallel and Distributed Computing with the Common Component Architecture

David E. Bernholdt

Computer Science and Mathematics Division

Oak Ridge National Laboratory

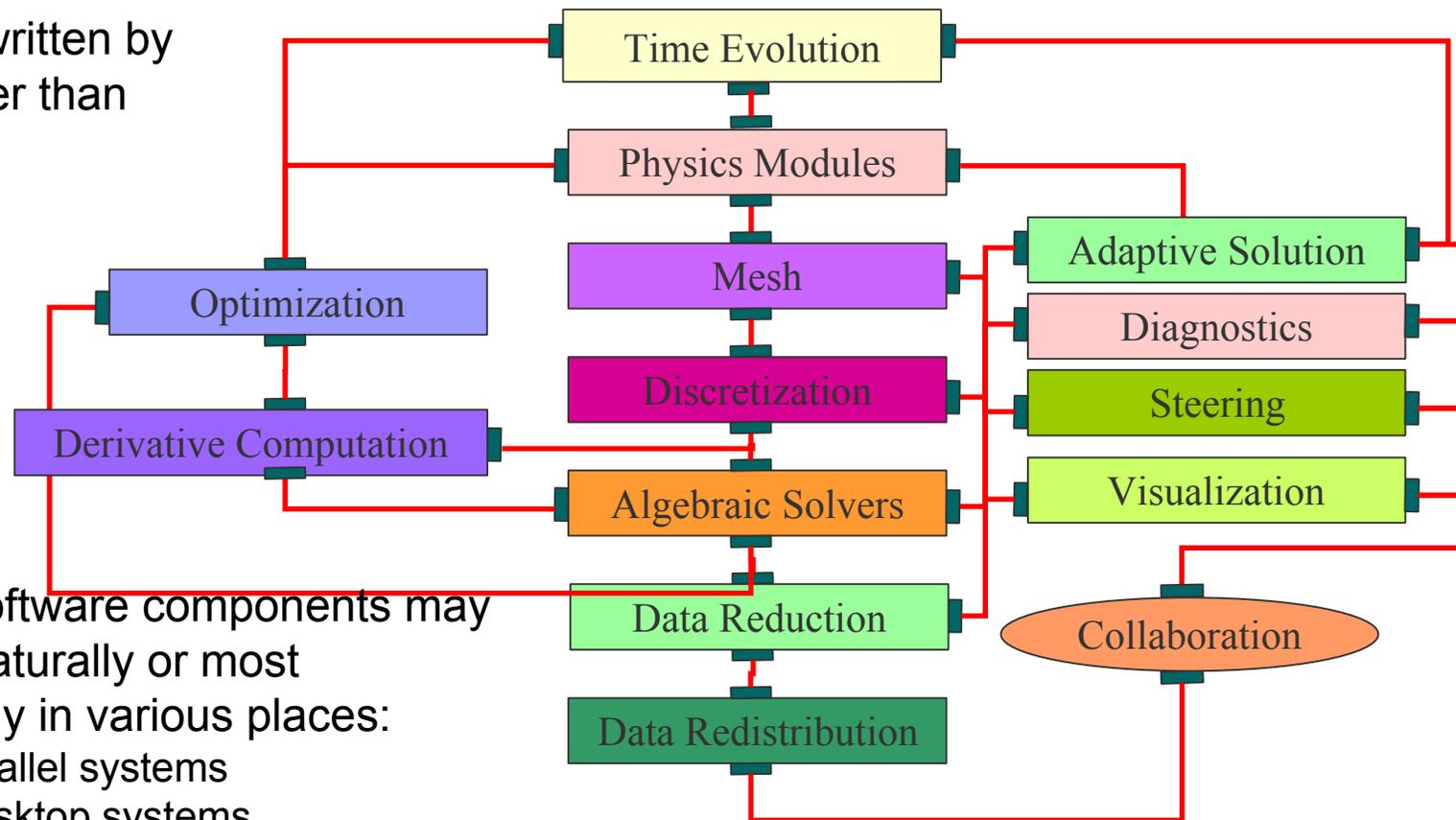
bernholdtde@ornl.gov

on behalf of the CCA Forum

<http://www.cca-forum.org>

Modern Scientific Software Development

- Complex codes, often coupling multiple types of physics, time or length scales, involving a broad range of computational and numerical techniques
- Different parts of the code require significantly different expertise to write (well)
- Generally written by teams rather than individuals



- Different software components may run most naturally or most conveniently in various places:
 - HPC parallel systems
 - Local desktop systems
 - Remote services

Component-Based Software Engineering

- **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]*
 - Facilitates collaboration around software development
- **Software complexity**
 - Components encapsulate much complexity into “black boxes”
 - Facilitates separation of concerns/interests
 - 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 is a component environment designed specifically for the needs of HPC scientific computing***

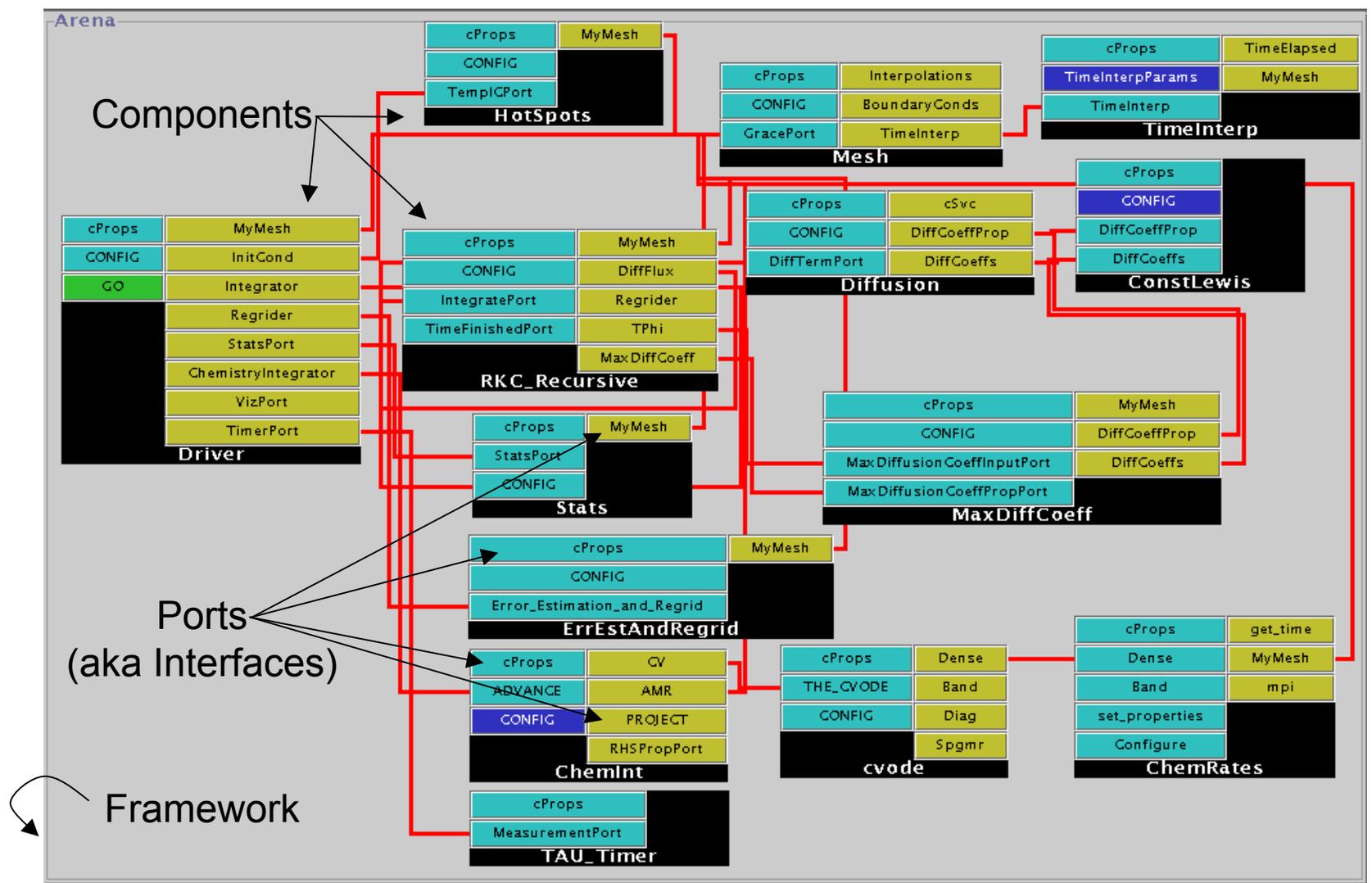
The Common Component Architecture (CCA)

- CBSE has been developed and is now widespread primarily in non-technical areas
- CBSE has not yet had much uptake in high-performance scientific computing
 - Largely due to deficiencies of “commodity” component models for HPC
- The Common Component Architecture is tailored specifically to the needs of the high-performance scientific computing community
- Supports both parallel and distributed computing
- Designed to be implementable with minimal performance impact
- Minimalist approach makes it easier to incorporate existing code into CCA
- Provides language interoperability for important languages for HPC (Fortran77/90/95, C, C++, Python, Java)

Basic CCA Concepts and Terminology

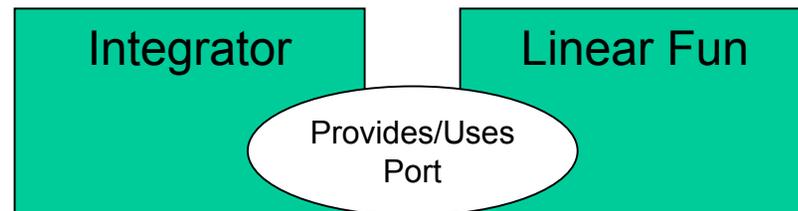
- Component
 - A unit of software deployment/reuse (i.e. has interesting functionality)
 - Interacts with the outside world only through well-defined interfaces
 - Implementation is opaque to the outside world
- Interface (a.k.a. Port in CCA)
 - Defines how components interact, distinct from implementation
 - Generally, a procedural interface
 - Some component-like environments are based strictly on data flow (e.g. AVS, Data Explorer, etc.)
 - Like C++ abstract virtual class, Java interface
- 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

“Wiring Diagram” for Typical CFRFS Combustion Application

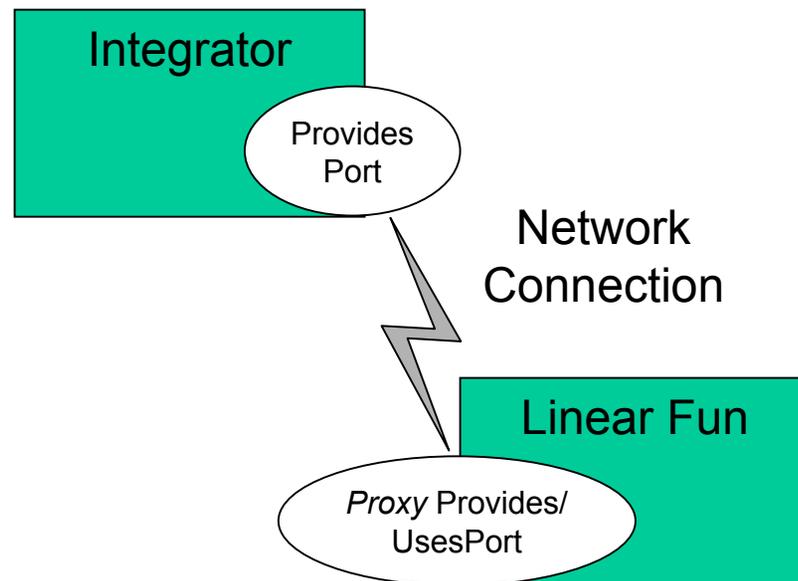


Support for Local High-Performance and Distributed Computing

- “**Direct connection**” preserves high performance of local (“in-process”) components
 - Framework makes *connection*
 - But is not involved in *invocation*
- **Distributed computing** has same uses/provides semantics, but **framework intervenes** between user and provider
 - Framework provides a *proxy* provides port local to the *uses* port
 - Framework conveys invocation from proxy to actual provides port



Direct Connection



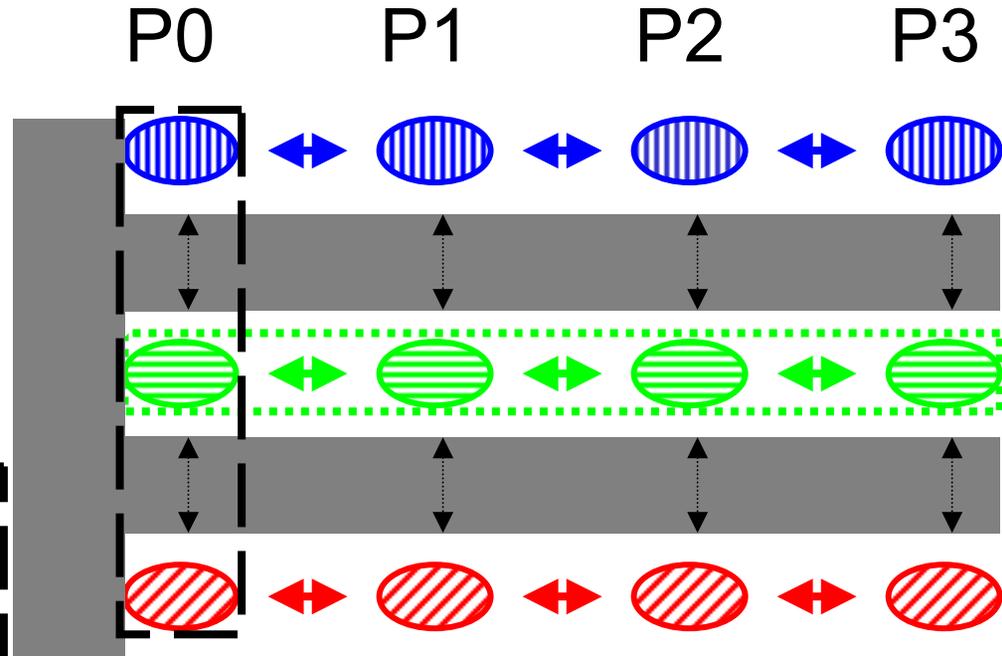
Network Connection

Support for Process-Based 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*

CCA Delivers Performance

Local

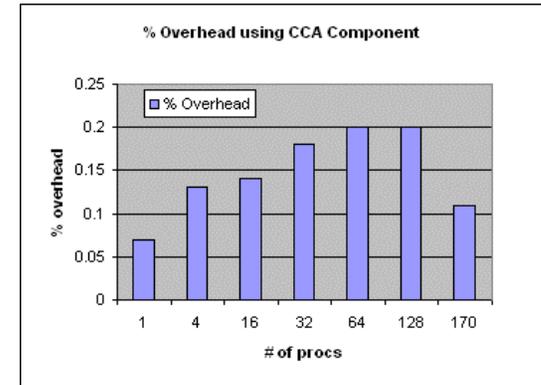
- No CCA overhead **within** components
- Small overhead **between** components
- Small overhead for **language interoperability**
- Be aware of costs & design with them in mind
 - Small costs, easily amortized

Parallel

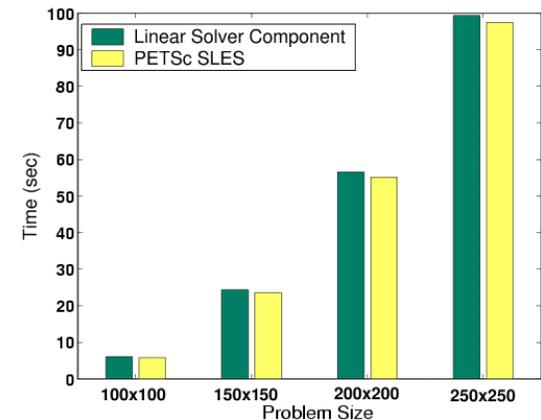
- No CCA overhead on **parallel computing**
- Use your **favorite** parallel programming model
- Supports SPMD and MPMD approaches

Distributed (remote)

- No CCA overhead – performance depends on networks, protocols
- CCA frameworks support OGSA/Grid Services/Web Services and other approaches



Maximum 0.2% overhead for CCA vs native C++ code for parallel molecular dynamics up to 170 CPUs

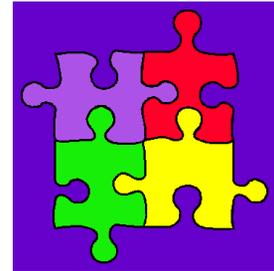


Aggregate time for linear solver component in unconstrained minimization problem w/ PETSc

CCA Research Thrusts

- Frameworks

- Frameworks (parallel, distributed)
 - Language Interoperability / Babel / SIDL
- Gary Kumfert, LLNL (kumfert@llnl.gov)

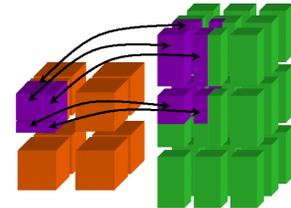


- “MxN” Parallel Data Redistribution

Jim Kohl, ORNL (kohlja@ornl.gov)

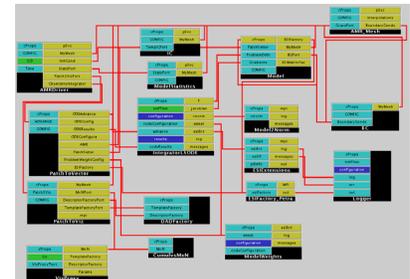
- Scientific Components

- Scientific Data Objects
- Lois Curfman McInnes, ANL (curfman@mcs.anl.gov)



- User Outreach and Applications

- Tutorials, Coding Camps
 - Interactions with users
- David Bernholdt, ORNL (bernholdtde@ornl.gov)

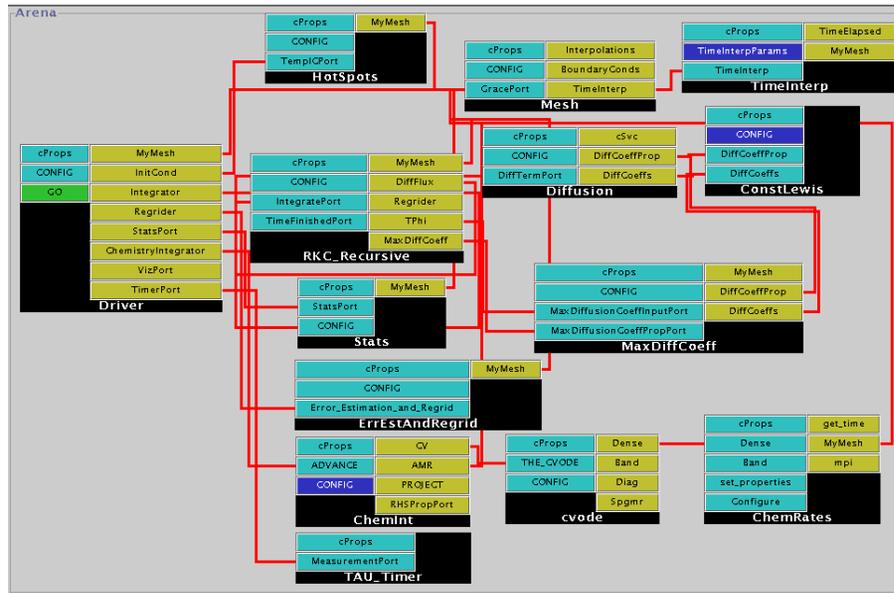


Frameworks Implement Various Programming Models & Features

Framework	Local Direct Connect	Parallelism Models			Distributed Computing Features		Meta-Component Model
		Process	Threaded	Component	Grid & Web Services	Parallel RMI	
Ccaffeine							
SCIRun2							
XCAT					OGSI, WSDL, etc.		
LegionCCA					Legion		
DCA (exptl)							

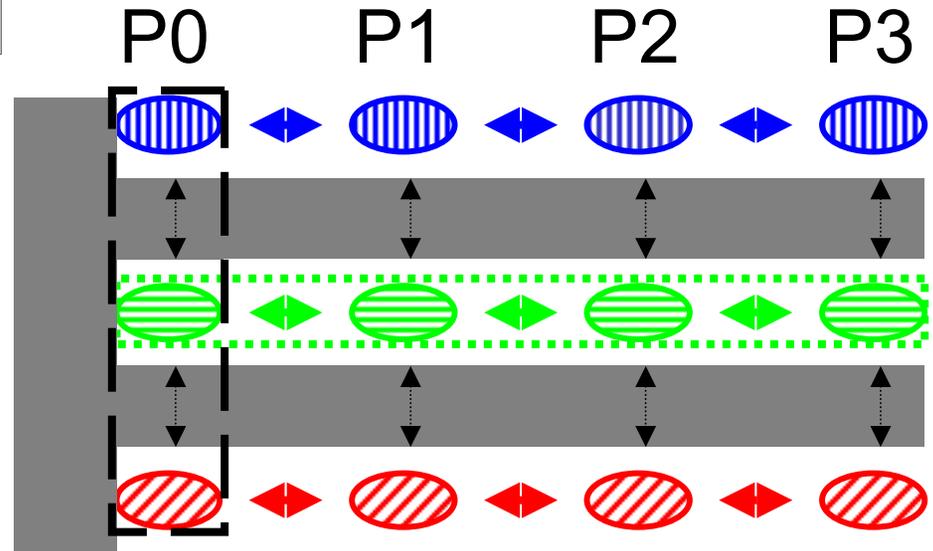
- Currently, framework interoperation primarily through BuilderService
 - Frameworks can export ports from components inside themselves
 - Then framework acts as a component in another framework
- Binary component compatibility between some frameworks

Ccaffeine



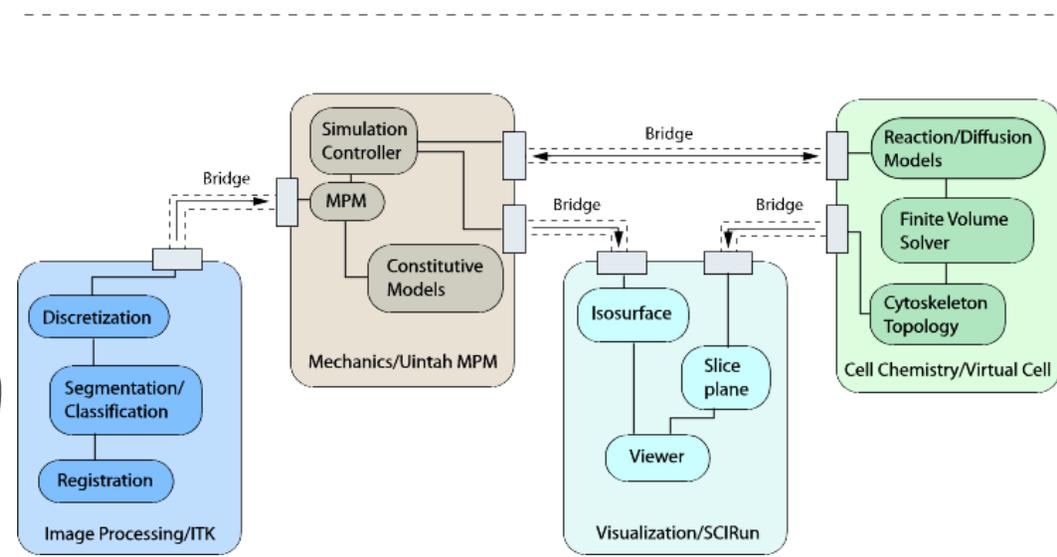
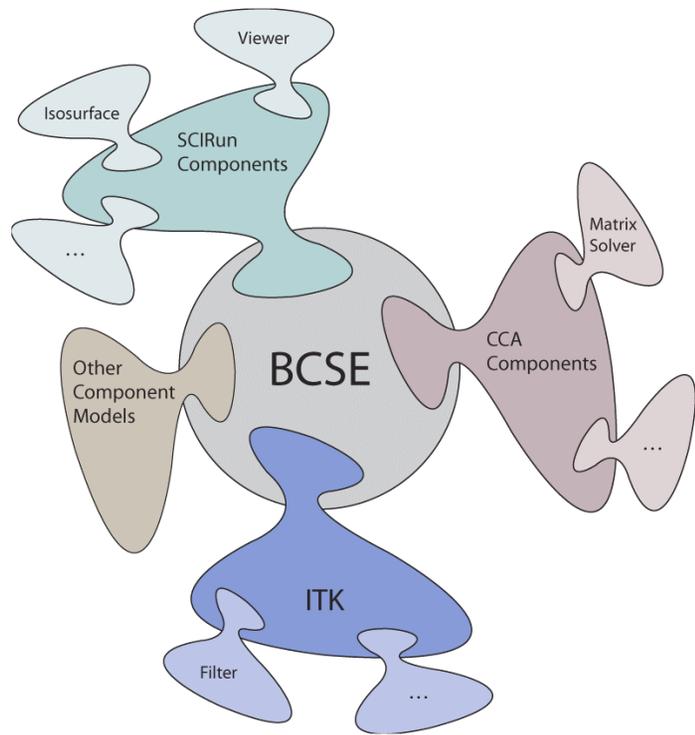
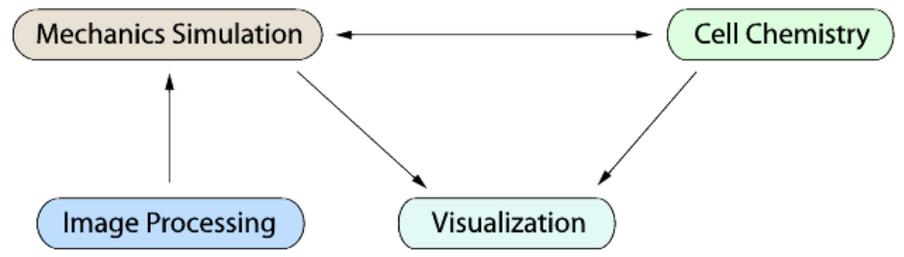
- Local direct connect
 - No framework intervention on call
 - Virtual function call overhead

- HPC process-based parallelism
 - No framework intervention or overhead



SCIRun2

- Meta components to combine:
 - CCA
 - SCIRun
 - VTK
 - CORBA
 - Other component systems



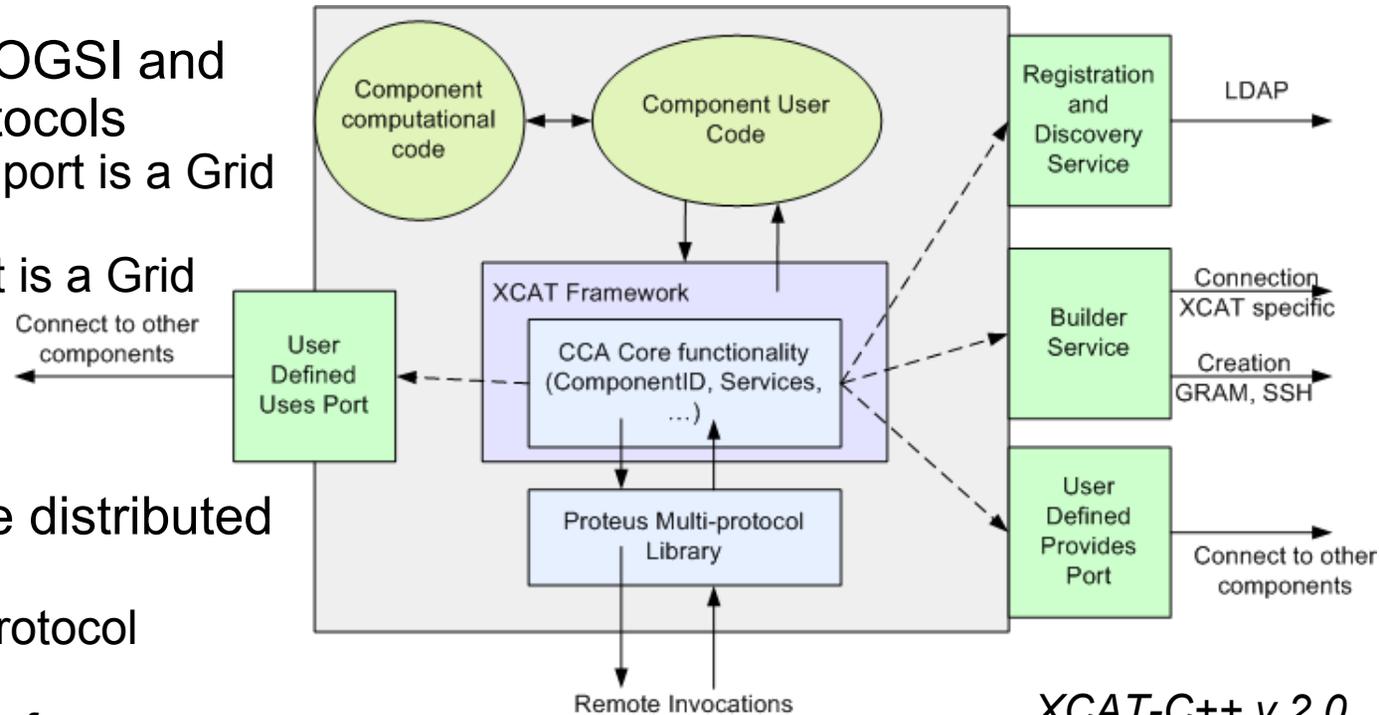
XCAT

XCAT-Java v 3.0

- Conformance w/ OGSI and web services protocols
 - Every provides port is a Grid Service
 - Every uses port is a Grid service client

XCAT-C++ v 2.0

- High-performance distributed components
 - Proteus multi-protocol architecture
 - SOAP as lingua franca
 - High-perf. Protocols used if available
- Interoperability w/ XCAT-Java v 2.0



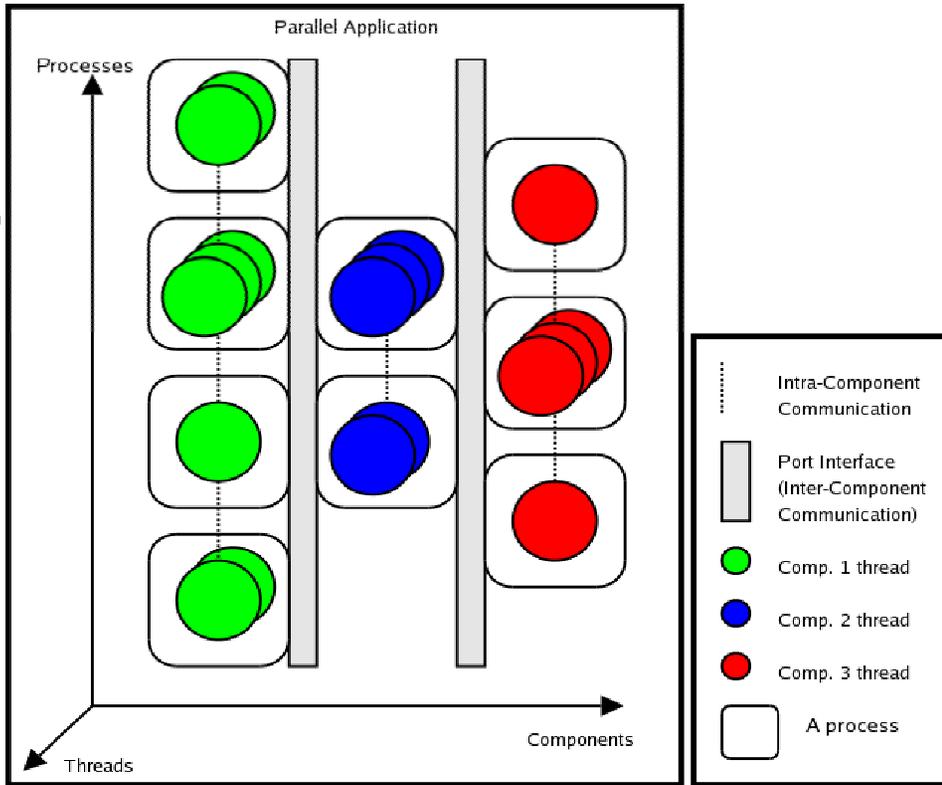
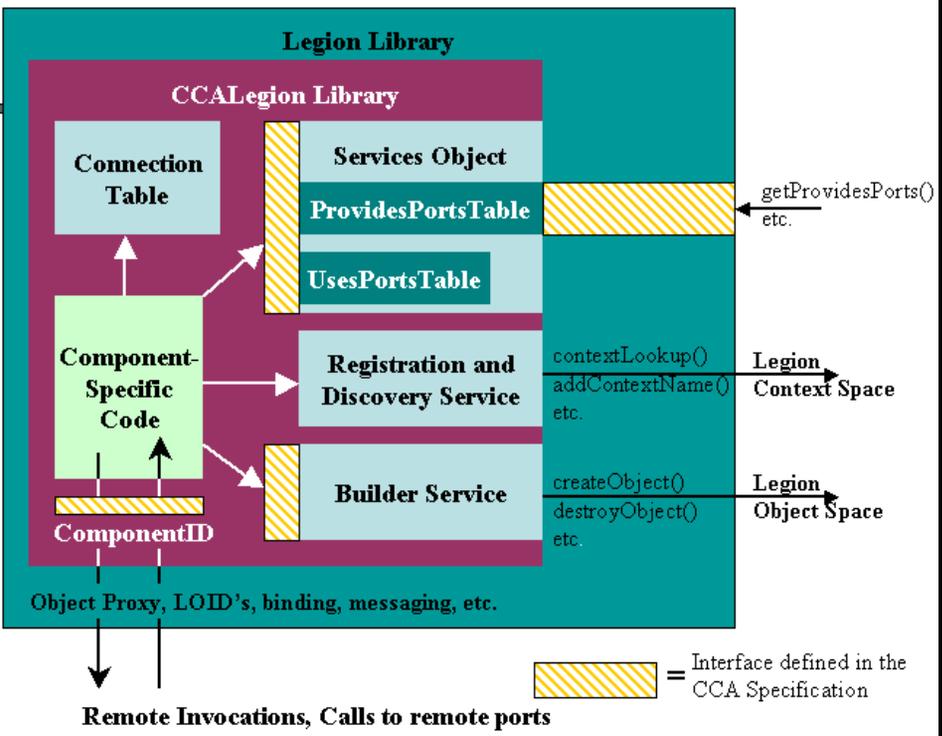
XCAT-C++ v 2.0

LegionCCA

Integrate CCA with Legion
Grid Environment

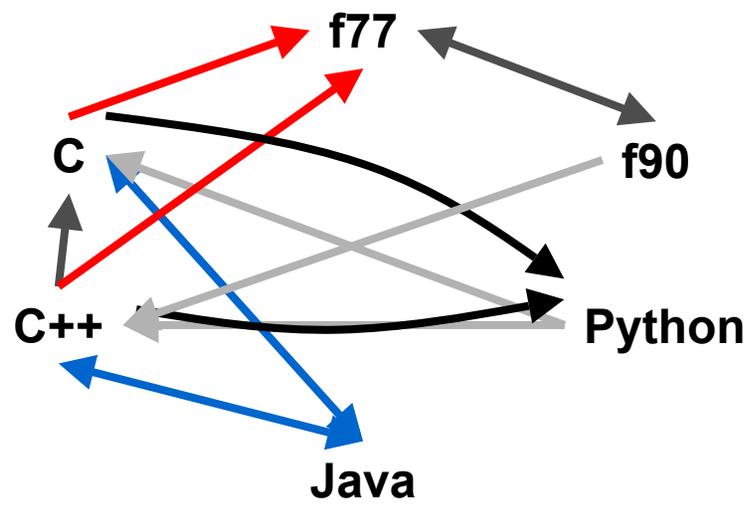
DCA

Exptl. framework for PRMI

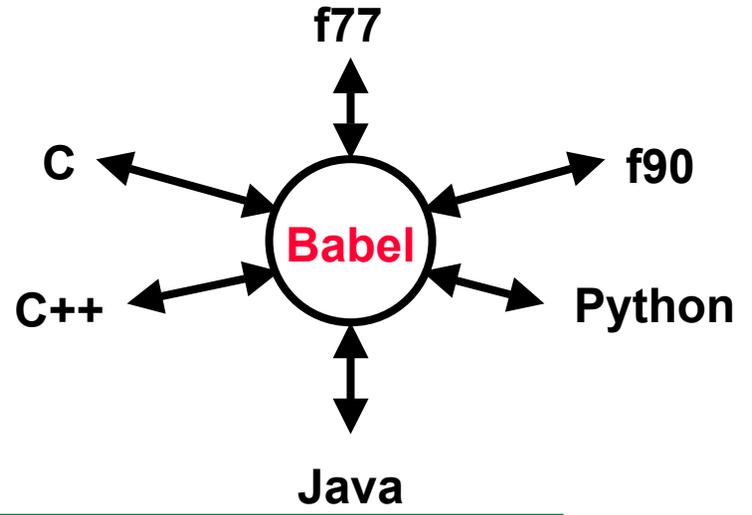


Language Interoperability

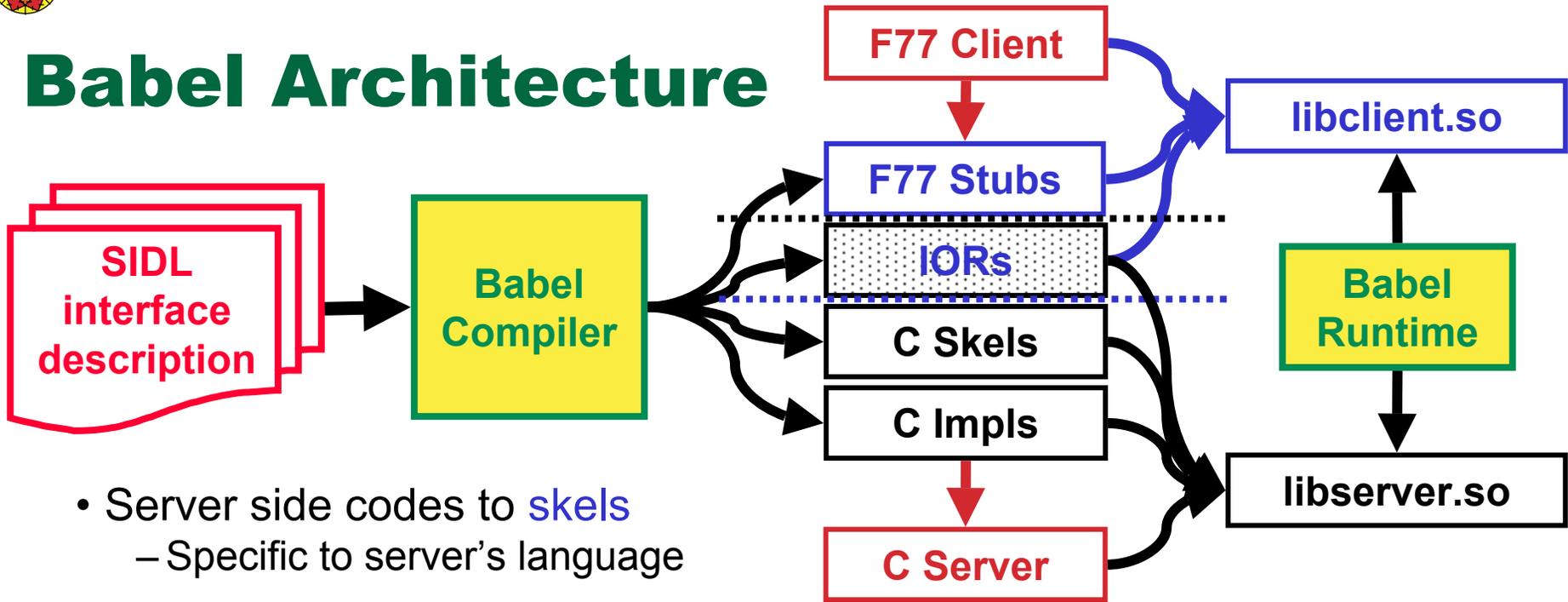
- Existing language interoperability approaches are “point-to-point” solutions



- Babel provides a unified approach in which all languages are considered peers
- Babel used primarily at interfaces

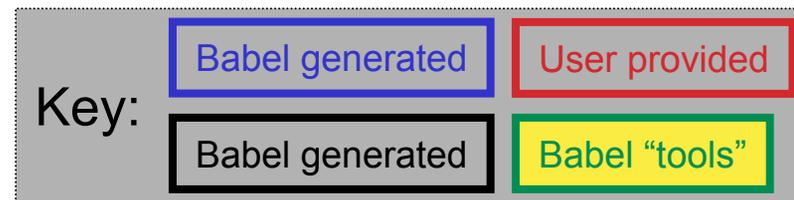


Babel Architecture



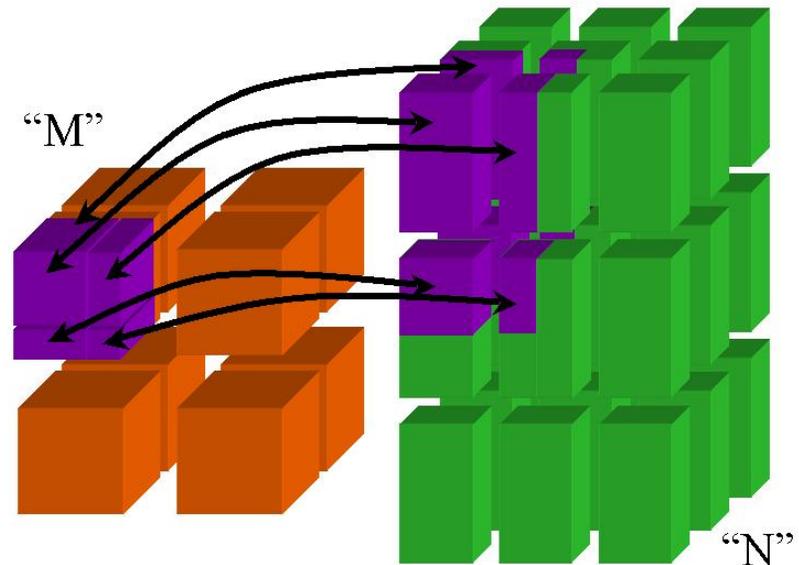
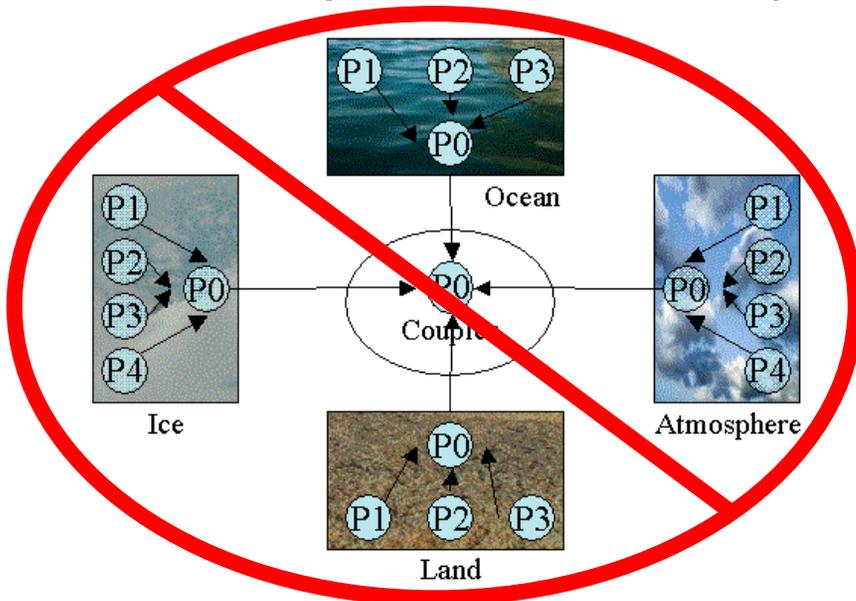
- Server side codes to **skels**
 - Specific to server’s language
- Client side codes to **stubs**
 - Specific to client’s language
- Internal Object Representation (**IOR**) bridges between
 - Implemented in C
- Babel generates glue code from **SIDL** specification of interface

- Babel includes both **code generation** and **runtime** components
- Strives to allow **natural-looking code** in each supported language



MxN Parallel Data Redistribution

- Share Data Among Coupled Parallel Models
 - Disparate Parallel Topologies (M processes vs. N)
 - e.g. Ocean & Atmosphere, Solver & Optimizer...
 - e.g. Visualization (Mx1, increasingly, MxN)



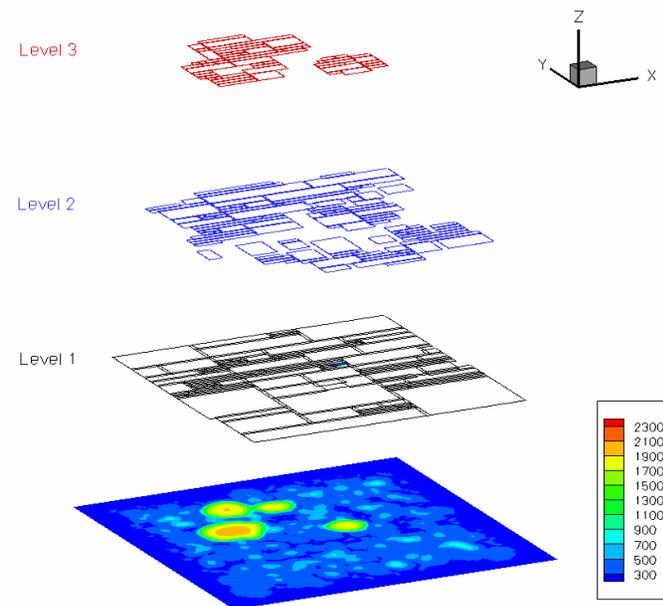
Research area -- tools under development

CCA Application Areas & Component Infrastructure

- Combustion
- Climate & Weather Modeling
- Quantum Chemistry
- Fusion
- Materials Science & Nanoscience
- Underground radionuclide transport
- Scientific Data Management
- Large-Scale Visualization
- Biomedical Engineering
- Data collection and processing (sensors)
- 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)

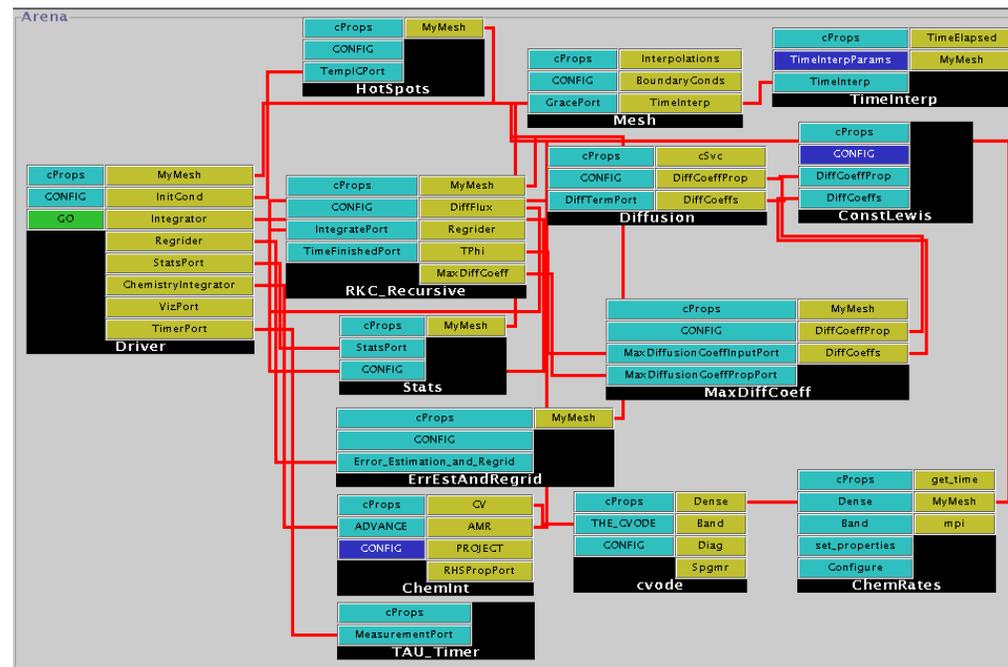
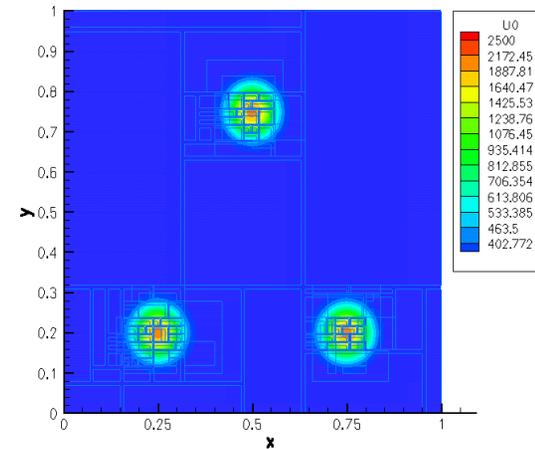
- **SciDAC BES project, H. Najm PI**
- **Investigators:** Sofia Lefantzi (SNL), Jaideep Ray (SNL), Sameer Shende (Oregon)
- **Goal:** A “plug-and-play” toolkit environment for flame simulations



- 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, Livermore

Component-Based CFRFS Applications

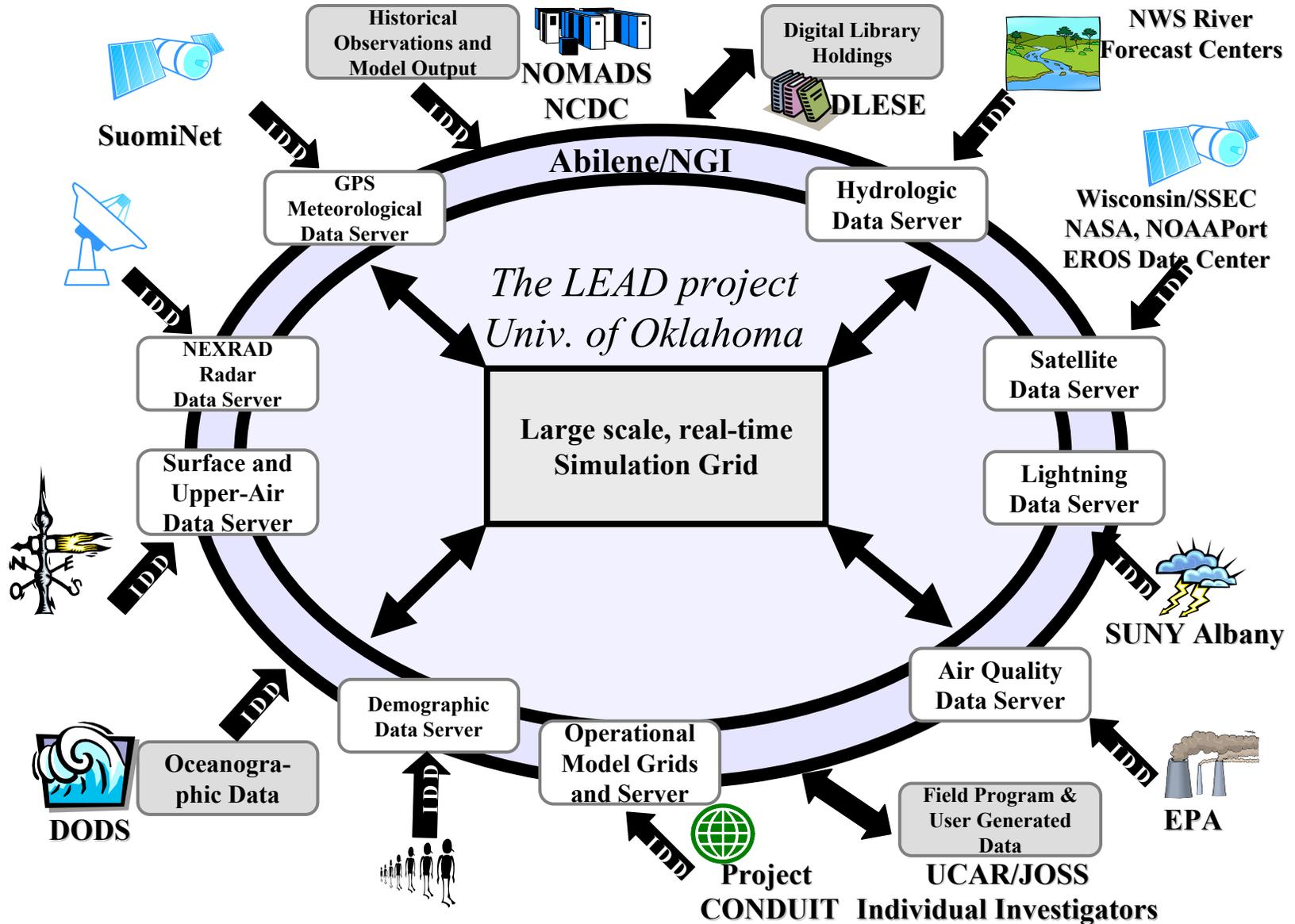
- Components mostly C++ or wrappers around old F77 code
- Developed numerous components
 - Integrator, spatial discretizations, chemical rates evaluator, etc.
 - Structured adaptive mesh, load-balancers, error-estimators (for refining/coarsening)
 - In-core, off-machine, data transfers for post-processing
- Integrating solver and viz capabilities from Chombo (LBL, APDEC)
- TAU for timing (Oregon, PERC)
- CVODES integrator (LLNL, TOPS)



Linked Environments for Atmospheric Discovery (LEAD)

- **NSF ITR Project, Kelvin Droegemeier, U Oklahoma, PI**
- **Participants:** U Oklahoma, Colorado State U, Millersville U, Indiana U, U Alabama-Huntsville, Howard U, UCAR, NCSA
- **Goals:** Better than real-time predictions of mesoscale severe storms
- Data mining of live instrument streams and historical storm metadata
- Requisition large computational resources on demand to start a large number of simulations
 - Mine simulation outputs to see which track real storm evolution
 - Refine scenarios that match incoming data
- May need to requisition bandwidth to make the needed data analysis possible
- May require real-time re-alignment of instruments

Predicting Severe Storms



Fusion: A Future Distributed+Parallel Simulation Application

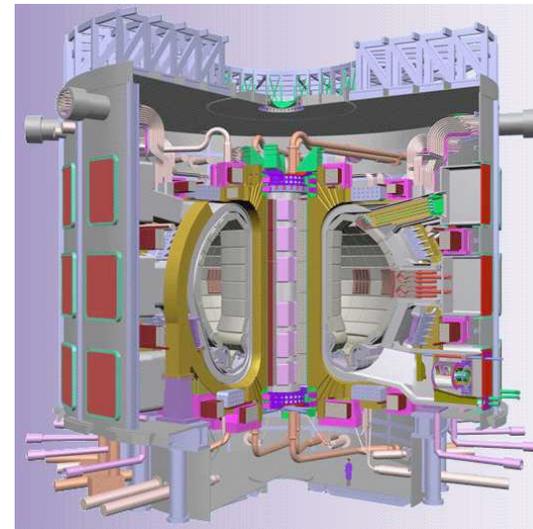
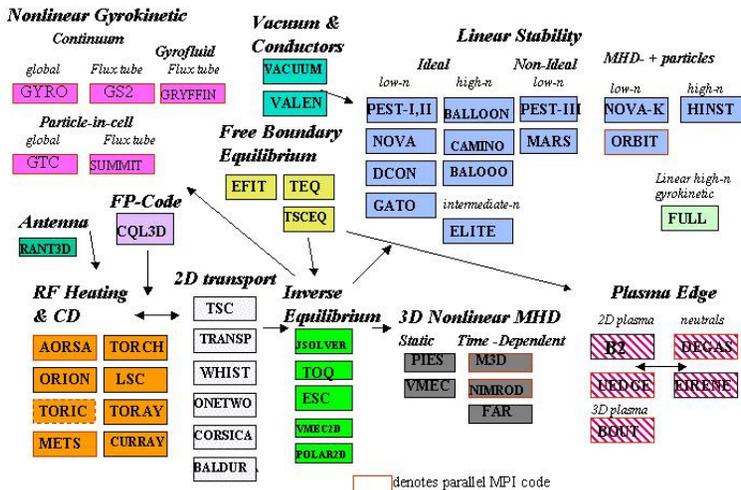
HPC Parallel

- Many physical phenomena -- many codes
- Coupled physics models under development
 - Local HPC coupling
 - Distributed loose coupling

Distributed

- Interface with exptl. Fusion devices
 - Data collection
 - Control
- Run HPC simulations between shots (10-15 min) to decide new parameters

Major U.S. Toroidal Physics Design and Analysis Codes



Summary

- CCA is a tool to help manage software **complexity**, increase **productivity**
- Designed specifically for high-performance parallel and distributed computing
 - Integrates parallel & distributed under a single model
 - Distributed CCA integrates with Grid and web technologies like OGSI, Legion, WSDL
- Under active development, with many **exciting capabilities in store**
- **Stable and robust** enough for use in applications
- CCA is allowing users to **focus on doing their science** in ways they might not have considered before
 - e.g. CFRFS toolkit/environment for combustion applications
 - e.g. LEAD mesoscale severe storm prediction

Information Pointers & Acknowledgements

- On the web: <http://www.cca-forum.org>
 - Mailing lists, meeting information, tutorial presentations, software, etc.
- CCA Forum meets quarterly
 - Combination standards body and user's group
 - Next meeting 29-30 July in Salt Lake City
- Email us: cca-pi@cca-forum.org
- Thanks to the *many* people who have contributed to the development and use of the CCA, who's work this talk represents! Especially...
 - Randy Bramley (Indiana), Felipe Bertrand (Indiana), Ken Chiu (Indiana), Dennis Gannon (Indiana), Madhu Govindaraju (SUNY Binghamton), Sriram Krishnan (Indiana), Mike Lewis (SUNY Binghamton)