UT-BATTELLE

A *Very Brief* Introduction to the Common Component Architecture

David E. Bernholdt

Computer Science and Mathematics Division, Oak Ridge National Laboratory

and

Center for Component Technology for Terascale Simulation Software

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY

Background & Motivation

- Commodity component models have limitations for HPC use
  - CORBA, COM/DCOM, Enterprise JavaBeans
  - Human timescales, no parallelism, language limitations, larger burden on legacy code
- Visualization tools
  - AVS, OpenDX, VTK, etc.
  - Data-flow based
- Domain-specific component environments
  - Overture, HDDA/DAGH, POOMA, Sierra, Hypre, SAMR
  - Hard to get interoperability & reuse on large scale (esp. cross-cutting components)

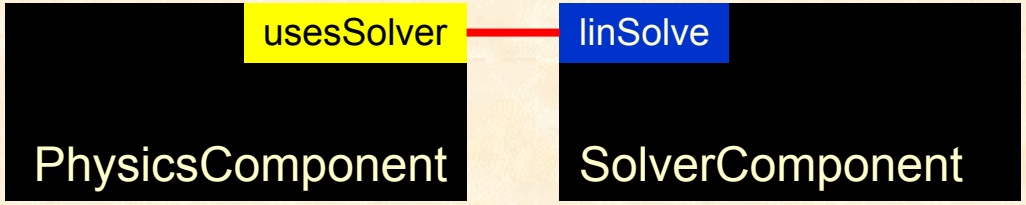
The Common Component Architecture

- A component model specifically designed for high-performance computing
- Supports both parallel and distributed applications
- Designed to be implementable without sacrificing performance
- Minimalist approach makes it easier to *componentize* existing software

CCA Concepts: Components

- A component encapsulates a useful chunk of functionality
  - Presents a well-defined interface to the outside world
  - Outside world knows nothing of internal implementation
  - “Size” of component up to architect/developer
- Based on OO concepts
- Conceptually similar to a library, but *not* the same

CCA Concepts: Ports



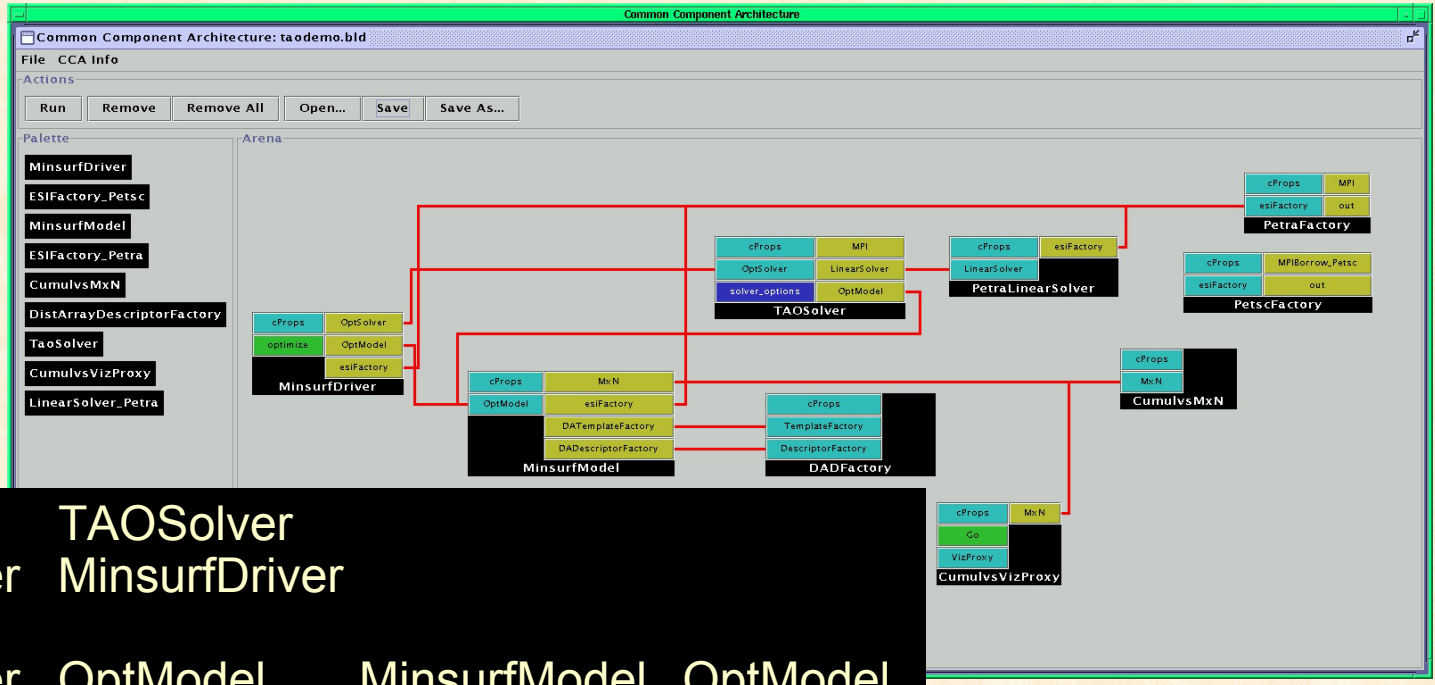
- Components interact through well-defined interfaces, or *ports*
- Ports follow a uses/provides pattern
  - A component may *use* a port (interface) *provided* by another
  - Components can *provide* ports by implementing the interface
- Components may use and provide any number of ports
- Note: Links denote a caller/callee relationship, ***not dataflow!***
  - e.g., linSolve port might contain: *solve(in A, out x, in b)*

CCA Concepts: Frameworks

- The framework provides the means to “hold” components and compose them into applications
- The framework is the application’s “main” or “program”
- Frameworks allow exchange of ports among components without exposing implementation details
- Frameworks may support sequential, distributed, or parallel execution models, or any combination they choose
- Frameworks provide a small set of standard services to components
  - BuilderServices allow programs to compose CCA apps
- Frameworks may make themselves appear as components in order to connect to components in other frameworks

What Does This Look Like?

- Launch framework (w/ or w/o GUI)
- Instantiate components required for app.
- Connect appropriate provided and used ports
- Start application (i.e. click Go port)



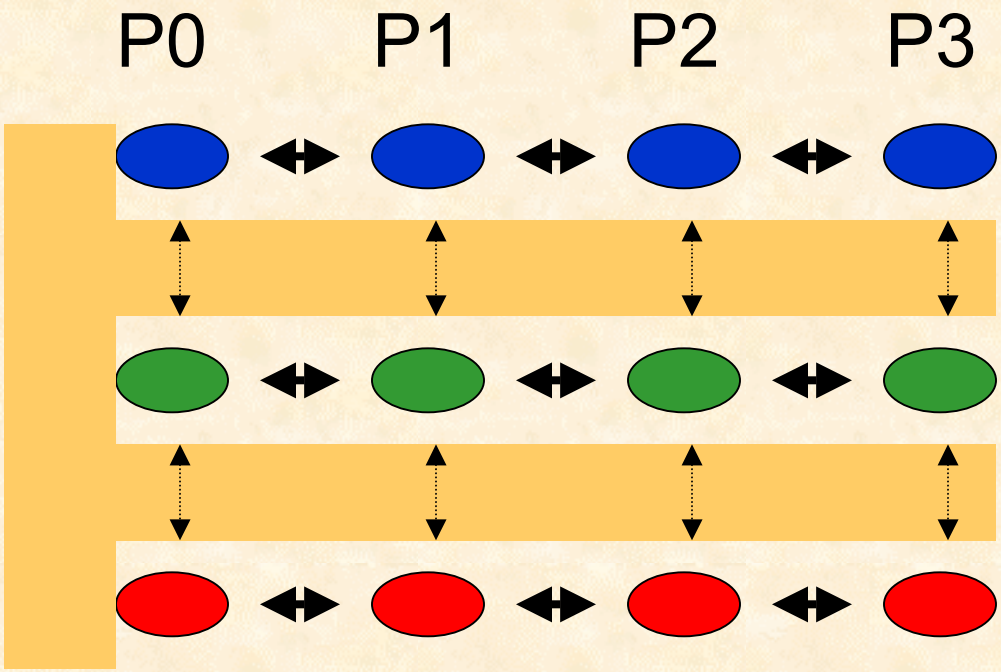

```
create   TaoSolver       TAOSolver
create   MinsurfDriver   MinsurfDriver
…
connect  MinsurfDriver   OptModel     MinsurfModel   OptModel
connect  MinsurfDriver   OptSolver    TAOSolver      OptSolver
…
```

CCA Concepts: Direct Connection

- Components loaded into separate *namespaces* in same *address space* (process) from shared libraries
- getPort call returns a pointer to the port's function table
- Invoking a method on a port is equivalent to a C++ virtual function call: lookup function, invoke
- Maintains performance (lookup can be cached)

CCA Concepts: 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)
- Also supports MPMD/MCMD



Components: Blue, Green, Red

Framework: Beige

Incorporation of Existing Code

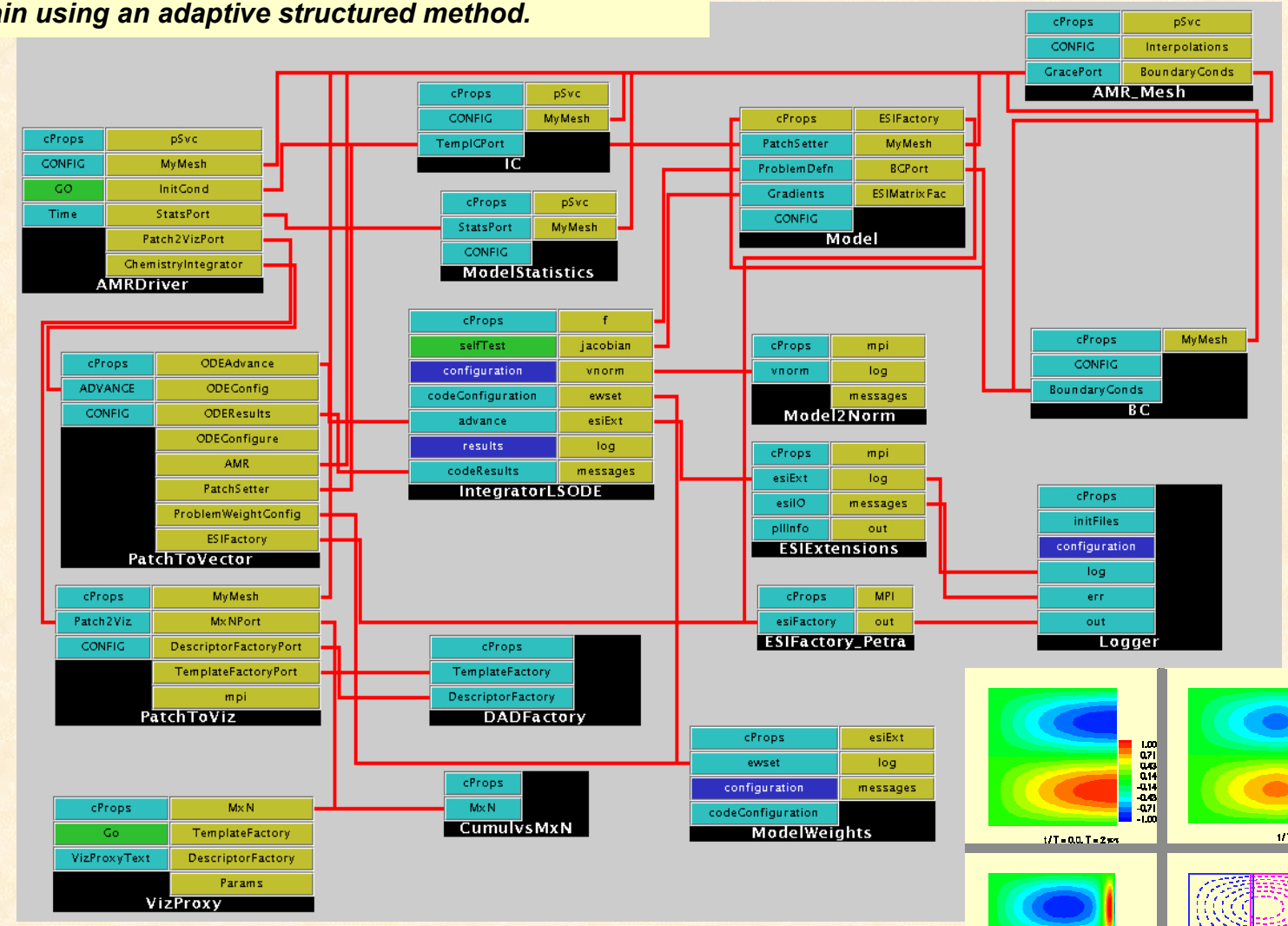
- Well-structured code needs *one* additional method to be a CCA component
  - setServices is where component declares ports provided and used
  - Called by framework when component instantiated
- Component requires null constructor

- To use a port, must getPort and when done releasePort
- Calls to methods on ports look like (in C++)
  port->method()

Current Status of CCA

- Specification version 0.5
- Working prototype frameworks
- Working multi-component parallel and distributed demonstration applications
- Draft specifications for
  - Basic scientific data objects
  - MxN parallel data redistribution
- SC01 demonstrations
  - four different “direct connect” applications, add’l distributed
  - DC demos: 31 distinct components, up to 17 in any single application, 6 used in more than one application
  - Components leverage and extend parallel software tools including CUMULVS, GrACE, LSODE, MPICH, PAWS, PETSc, PVM, SUMAA3d, TAO, and Trilinos.
- More than 15 projects adopting CCA
- CCA already used for “serious” applications

Solution of a two-dimensional heat equation on a square domain using an adaptive structured method.



IntegratorLSODE provides a second-order implicit time integrator, and ***Model*** provides a discretization. The remaining components are essentially utilities that construct the global ODE system or adaptors that convert the patch-based data structures of the mesh to the globally distributed array structure used for runtime visualization.

The CCA Forum

- The Common Component Architecture is merely a specification for what is required to be a CCA component and a CCA framework
- Specification determined by the CCA Forum
- Forum also encourages development of community/domain “standard” interfaces, components, etc.
- CCA Forum has been meeting regularly since January 1998
  - Face-to-face meetings quarterly
  - Next meeting: 24-25 June @ ANL
- Open membership
- Voting privileges based on attendance at recent meetings

The Center for Component Technology for Terascale Simulation Software (CCTTSS)

- A SciDAC Integrated Software Infrastructure Center (ISIC)
- *Mission:* Advance research in high-performance component technology and bring CCA from a conceptual prototype to a full-fledged production-quality environment
- *Participants:*
  - Argonne, Livermore, Los Alamos, Oak Ridge, Pacific Northwest, and Sandia National Laboratories;
  - Indiana University, and University of Utah

  (subset of CCA Forum)

CCTTSS's R&D Agenda

- Frameworks
  - Integration of prototype frameworks
  - Language interoperability tools (Babel/SIDL)
  - Component deployment
- Scientific Components
  - Abstract interfaces and component implementations
  - Scientific data; Linear, nonlinear, and optimization solvers; Steering and visualization; Multi-threading and load redistribution; Fault tolerance
  - Quality of service research
- MxN Parallel Data Redistribution
- Applications Integration
  - Chemistry and Climate work within CCTTSS
  - Close collaboration with other SciDAC infrastructure projects (especially TOPS, TSTT)
  - Strong liaison with adopting groups

CCA Resources

- Tutorials
  - In conjunction with quarterly CCA Forum meetings
    - Next: 24 June at ANL (Forum meeting 24-25 June)
  - Proposal submitted to SC02
  - On request or as opportunities arise
  - Latest materials posted on cca-forum.org
  - SC01 software distribution on cca-forum.org
- Collaborative environment (cca-forum.org)
  - Web server
  - CVS repository (spec, framework, components)
  - Accounts available on request
- People
  - Diverse backgrounds and expertise
  - Geographically distributed
  - All CCTTSS sites have Access Grid, email, phones, etc.

Information Pointers

- http://www.cca-forum.org
- http://www.cca-forum.org/ccttss/
- Mailing list: cca-forum@cca-forum.org (sign up at http://www.cca-forum.org/mailman/listinfo/cca-forum/)
- http://www.cca-forum.org/tutorials/
- CCTTSS contacts:

Lead PI	**Rob Armstrong**	**SNL**	**rob@sandia.gov**
Frameworks	Scott Kohn	LLNL	skohn@llnl.gov
Scientific Data Components	Lois McInnes	ANL	mcinnes@anl.gov
MxN Data Redistribution	Jim Kohl	ORNL	kohlja@ornl.gov
Applications Integration	**David Bernholdt**	**ORNL**	**bernholdtde@ornl.gov**