An Overview of the Common Component Architecture (CCA)

Presented by

The CCA Forum
and the
Center for Technology for Advanced Scientific Component Software (TASCS)

See companion presentation:
How the Common Component Architecture Advances Computational Science

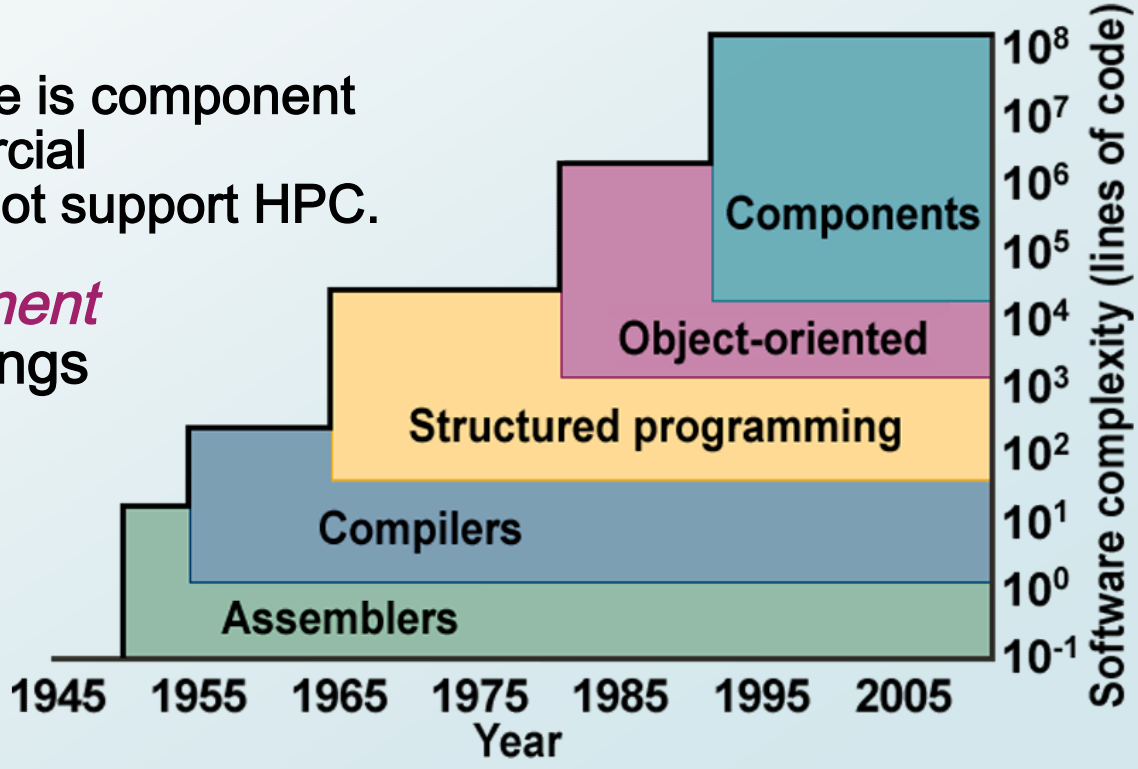
SC07

Office of Science
U.S. DEPARTMENT OF ENERGY

SciDAC
Scientific Discovery through Advanced Computing

Motivation

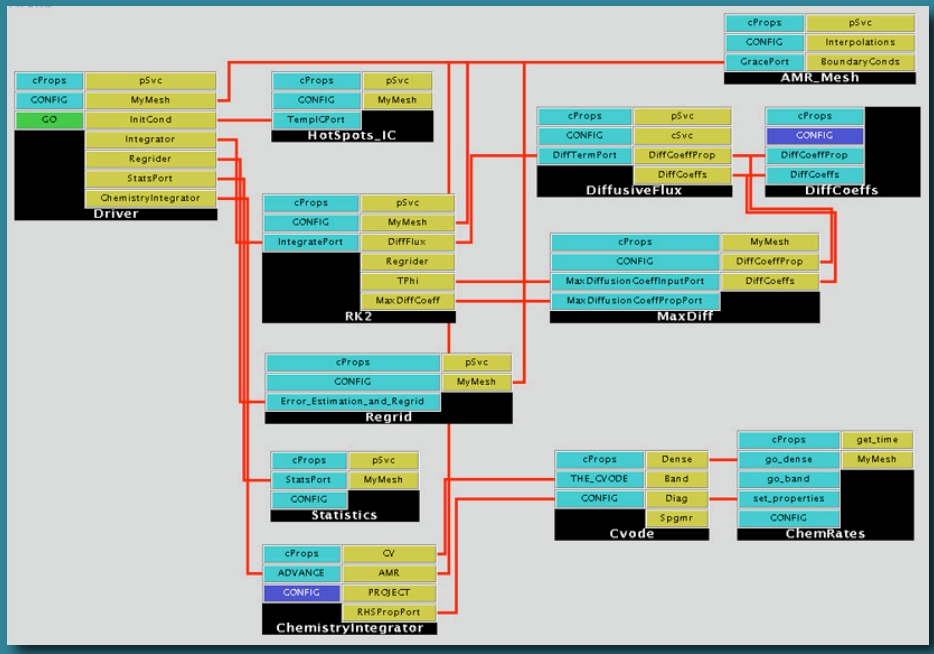
- Complexity of scientific software increases with simulation fidelity, multi-physics coupling, computer power ➡ *software crisis.*
- Component technology is well established outside of high-performance computing (HPC) as a way to manage software complexity.
  - All enterprise software is component software, but commercial implementations do not support HPC.
- *The Common Component Architecture* (CCA) brings component software approach to scientific HPC.
  - Grass-roots effort launched in 1998.



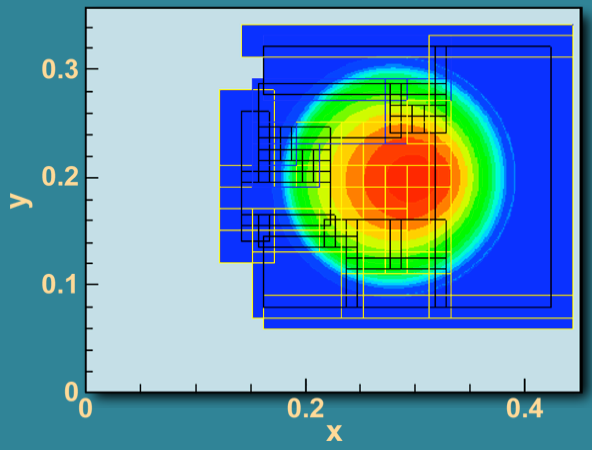
CCA Common Component Architecture

Benefits to software developers

- Components are natural units of decomposition and interaction for both software and developers:
  - Manage software complexity.
- They enable scientists to work together as a cohesive scientific enterprise, across disciplines, geographical boundaries, and technical preferences by facilitating…
  - Collaboration around software development,
  - Interoperability and reuse of software tools,
  - Community standards for scientific software,
  - Coupling of disparate codes.



CCA-based simulation of OH concentration in advective-diffusive-reactive simulation using 4th order Runge-Kutta-Chebyshev integrator on 4 levels of adaptively refined mesh

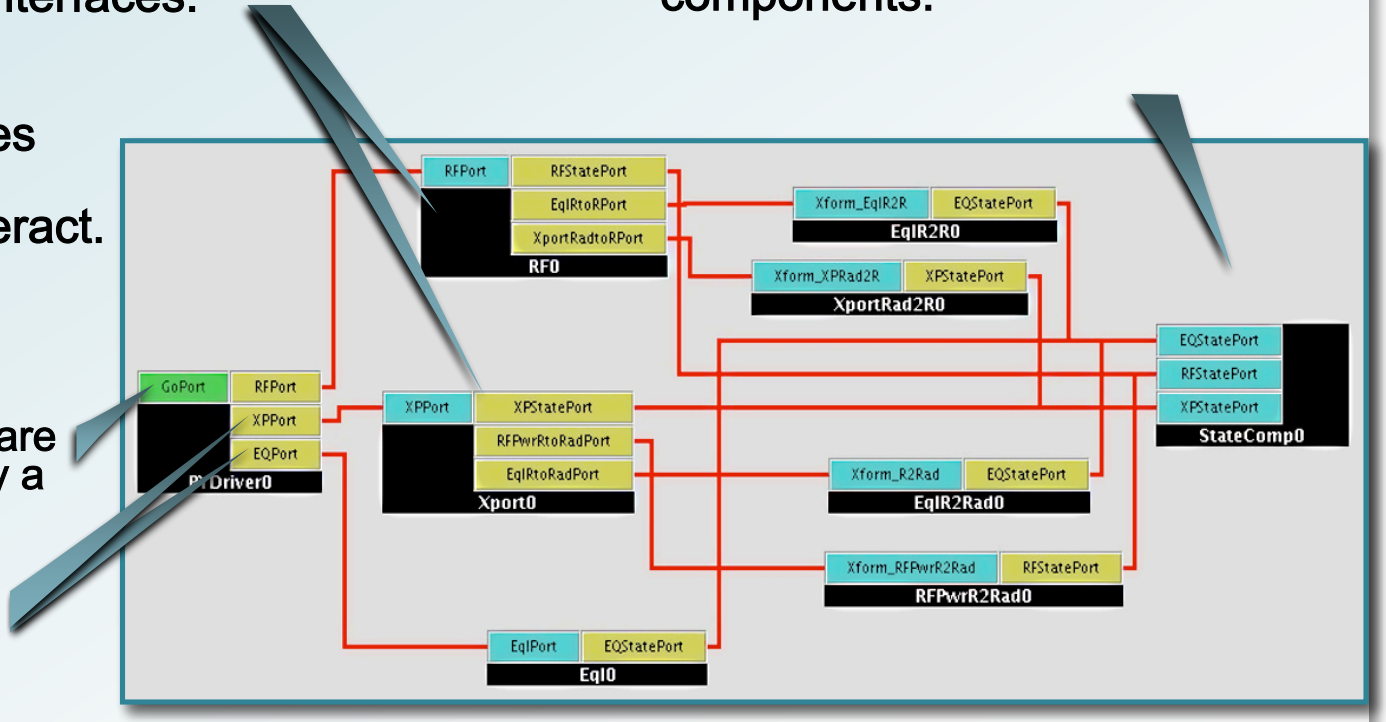


Courtesy of J. Ray, Sandia National Laboratories

CCA Common Component Architecture

Basic CCA concepts

- ***Components***
  - Are units of software development/ functionality.
  - Interact only through well-defined interfaces.
  - Can be composed into applications based on their interfaces.
- ***Ports***
  - Are the interfaces through which components interact.
  - Follow a provides/uses pattern:
    - Provided ports are implemented by a component.
    - Used ports are functionality a component needs to call.
- ***Frameworks***
  - Hold components while applications are assembled and executed.
  - Control the connections of ports.
  - Provide standard services to components.

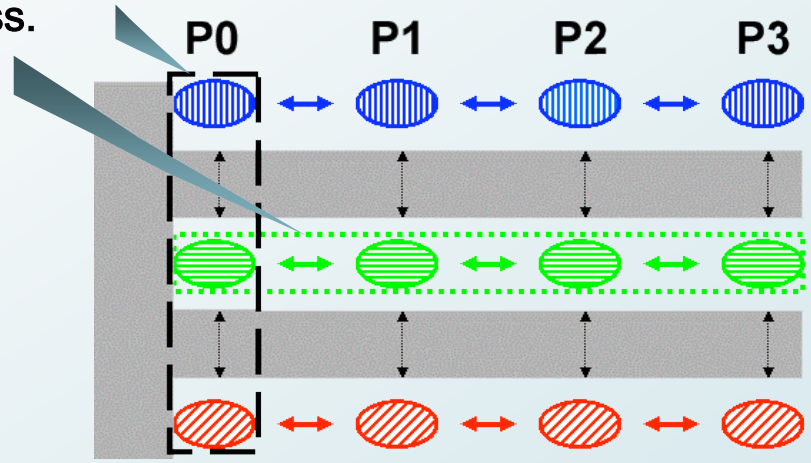


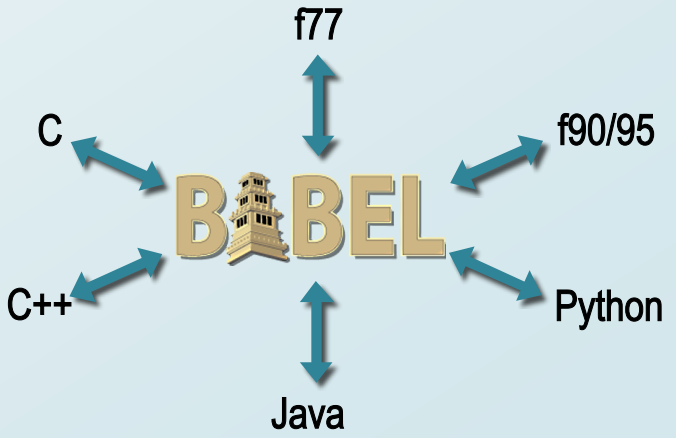
Screenshot of application in the Ccaffeine framework's GUI

CCA Common Component Architecture

CCA features for scientific HPC

- Parallel computing
  - Component mechanisms apply within a process.
  - Parallelism across processes is up to each component–
    - Usual tools: MPI, Global Arrays, PVM…
  - Both SPMD and MPMD supported.
- Distributed computing
  - Supported transparently to components.
- Performance
  - Components in same process share memory:
    - Small overhead on inter-component calls.
  - No overhead on parallel communication.
  - Minimal language interoperability overhead.
- Language interoperability
  - Implementation language of component shouldn't matter to others.
  - Babel treats all supported languages as peers.
  - SIDL allows language-neutral specification of interfaces.





CCA Common Component Architecture

Current status of the CCA

- CCA specification well established and stable.
  - Approaching “1.0” completeness.
- Suite of tools implement the CCA environment.
  - Babel, Chasm (language interop), Ccaffeine (framework).
  - Other frameworks also available.
- CCA tools and concepts are used by more than 25 different application groups in diverse fields.
  - CCA provides a common infrastructure for developing simulation toolkits and frameworks, coupling disparate codes, and many other types of applications.
  - CCA benefits users in many different ways.
  - See companion presentation *How the Common Component Architecture Advances Computational Science.*

CCA research and development plans

- Leverage the component environment to provide important new capabilities to software developers
  - Adapt running applications for performance, accuracy, faults, and other criteria
  - Improve software quality via software contracts, testing, and verification
  - Use high-end hardware with massive parallelism, heterogeneous processors
- Mature the CCA environment and tools to production quality
- Grow a “component ecosystem”
  - Enable plug-and-play application development using off-the-shelf scientific components
- Help computational scientists effectively use component technology

The CCA community

- The CCA Forum is the standards body and user group.
  - Quarterly face-to-face meetings, mailing lists, collaboration resources
- DOE SciDAC-funded Center for Technology for Advanced Scientific Component (TASCS) Software core CCA development team.
- Many other projects and sponsors contribute to development and use of CCA.

CCA Common Component Architecture

For more information

- See companion presentation:
  How the Common Component Architecture Advances Computational Science
- ORNL booth at SC2007
  - David E. Bernholdt, Wael R. Elwasif, James A. Kohl (ORNL)
  - Tom Epperly, Gary Kumfert (LLNL)
  - Ben Allan, Rob Armstrong, Jaideep Ray (SNL)
- Other booths at SC2007
  - Ames Laboratory (Booth 181)
  - Argonne National Laboratory (Booth 551)
  - Indiana University (402)
  - NNSA/ASC (1617)
  - Pacific Northwest National Laboratory (581)
  - Tech-X Corporation (190)
  - University of Utah (287)
- On the internet
  - http://www.cca-forum.org
  - cca-forum@cca-forum.org