

Accelerator Simulation using CCA Components (CCA/Synergia2)

SAP Participants: Douglas Dechow, Tech-X Corporation
Lois Curfman McInnes, Boyana Norris, ANL

Physics Collaborators: James Amundson, Panagiotis
Spentzouris, FNAL

SciDAC Math/CS Collaborators: TASCs, PERI, TOPS

Recent Progress: December 2008



Motivation

- **Complexity of scientific software increases with simulation fidelity, multi-physics coupling, and computer power**
 - ➔ **software crisis**
- **Tools**
 - **Computational accelerator community:** Tremendous investment in software applications, codes written in a variety of languages, targeting a range of computational platforms
 - **Common Component Architecture (CCA) component vision:** Enable the HPC community to leverage existing applications, creating modular, reusable software components that facilitate the combined use of historically independent codes to add new capabilities (see www.cca-forum.org)
- **Long-term Goal:** Foster a component community in computational accelerator physics, with emphasis on easily incorporating new algorithms and performance enhancements

Approach

- Using CCA tools and specifications, prototype an accelerator simulation from existing codes that were not originally designed to work together
- Develop components based on codes for beam dynamics; also incorporate external numerical libraries
 - Synergia2 beam physics framework (FNAL)
 - MaryLie/Impact beam physics application (LBNL/U. Maryland)
 - High-performance numerical tools via TOPS (including PETSc (ANL), LBNL numerics)

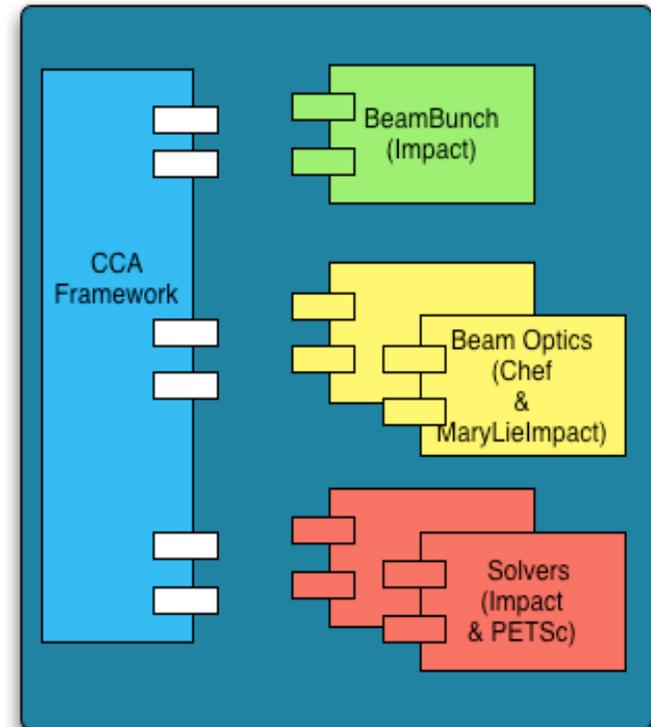


Benefits

- Key features of CCA components
 - Programming language interoperability via SIDL
 - Dynamic composability
 - Encouragement of common interfaces
- Benefits to COMPASS
 - Improved Productivity
 - Reuse physics components across COMPASS project and the accelerator community
 - Leverage tools and libraries developed by experts in other specialties (math/cs)
 - Enhanced Performance
 - Exploit highly optimized numerical software
 - Enable adaptive method configuration and selection to better match dynamically changing computational requirements

Initial Work: Beam Dynamics Components

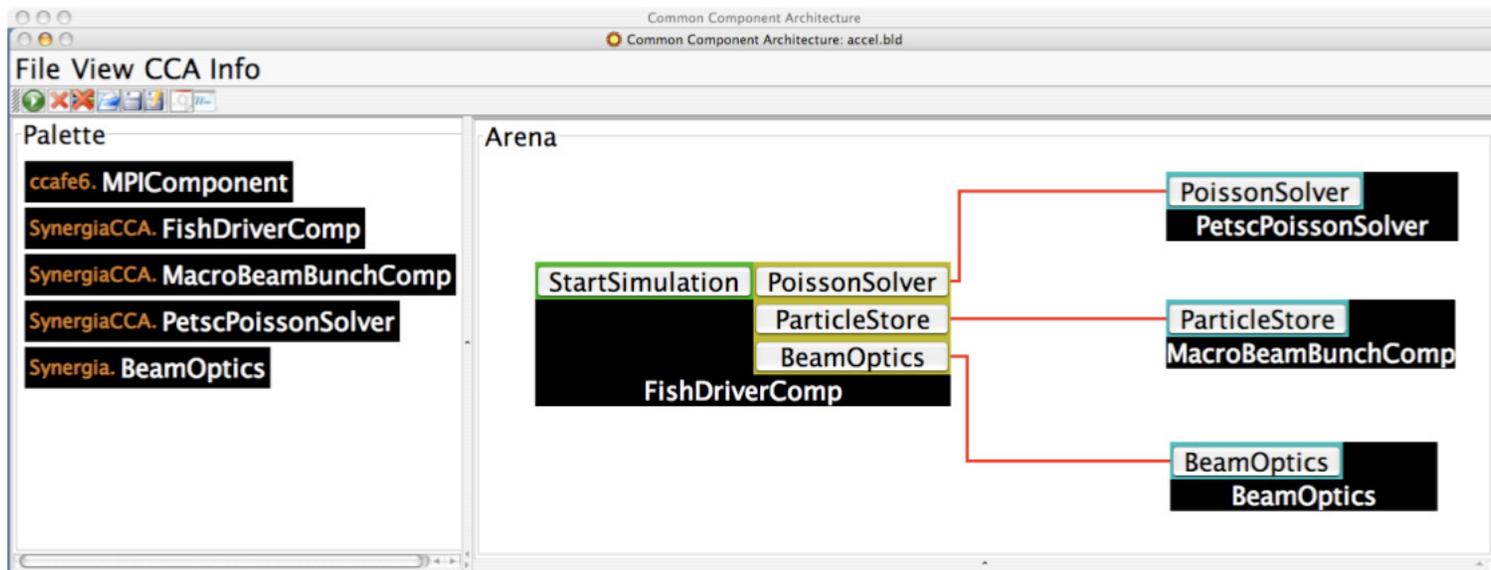
- Accelerator modelling components (collaboration with TASCs):
 - F90-based beam optics components (quadrupoles and drifts) from the MaryLie/Impact application (LBNL)
 - C++ and F90 particle store components from the Synergia2 framework (FNAL)
 - A newly implemented C++-based space charge solver, Sphyaena, which makes use of Synergia2, PETSc (ANL), and FFTW.
- Component interfaces allowed us to capture and make available only the functionality that was desired from the existing codes.
- Reference: D. Dechow, B. Norris, and J. Amundson, *The Common Component Architecture for Particle Accelerator Simulations*, Proceedings of HPC-GECO/CompFrame'07, October 21-22, 2007, Montreal, Quebec, Canada, ACM, 2007.



Prototype CCA Beam Dynamics Toolkit

Developed prototype Contractor-enabled CCA beam dynamics toolkit

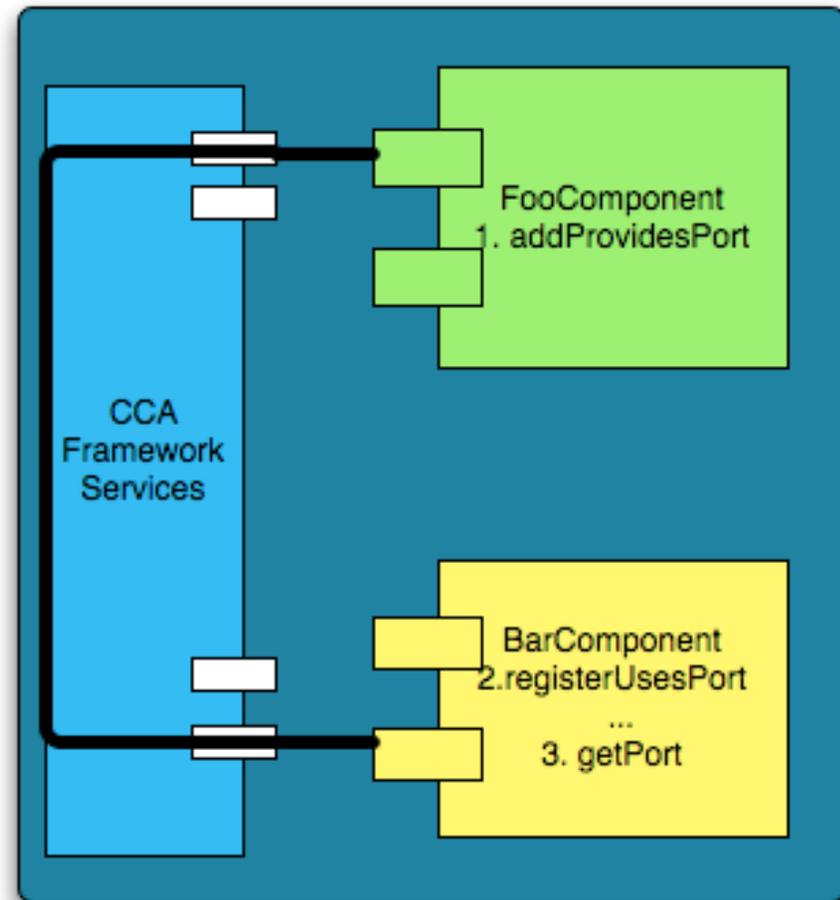
- **Components:** interact only through well-defined interfaces
- **Ports:** well-defined interfaces through which components interact
 - Follow a provides/uses pattern
- **Frameworks:** hold components while applications are assembled and executed, control the connection of ports, provide services to components



Screenshot of Ccaffeine framework's GUI

Provides-Uses Pattern

- FooComponent:
 - Declares that it will support a particular port by calling `addProvidesPort()`
- BarComponent:
 - Declares that it requires a set of services by calling `registerUsesPort()`
 - Retrieves reference to port by calling `getPort()`



Implemented Components

Synergia.BeamOptics: an ML/I component for obtaining transfer map

testsp.Fquad3: a driver component for testing the Synergia.BeamOptics; provides a *go* port and uses a Synergia.BeamOpticsPort port

Synergia.BeamBunch: a Synergia2 component for managing the particles of a beam bunch in a 6-D representation

testsp.BeamDriver: a driver component for testing Synergia.BeamBunch; the component provides a *go* port and uses a Synergia.ParticleStore port

Synergia.BeamSolver: a Synergia2 component that uses a recently developed Synergia2 solver, Sphyraena, for solving Poisson's equation; this component provides a Synergia.PoissonSolver port and uses a Synergia.ParticleStore port

Implemented Components (cont.)

testsp.SolverDriver: a driver component for testing the Synergia.BeamSolver and Synergia.BeamBunch components; the component provides a *go* port and uses a Synergia.PoissonSolver port

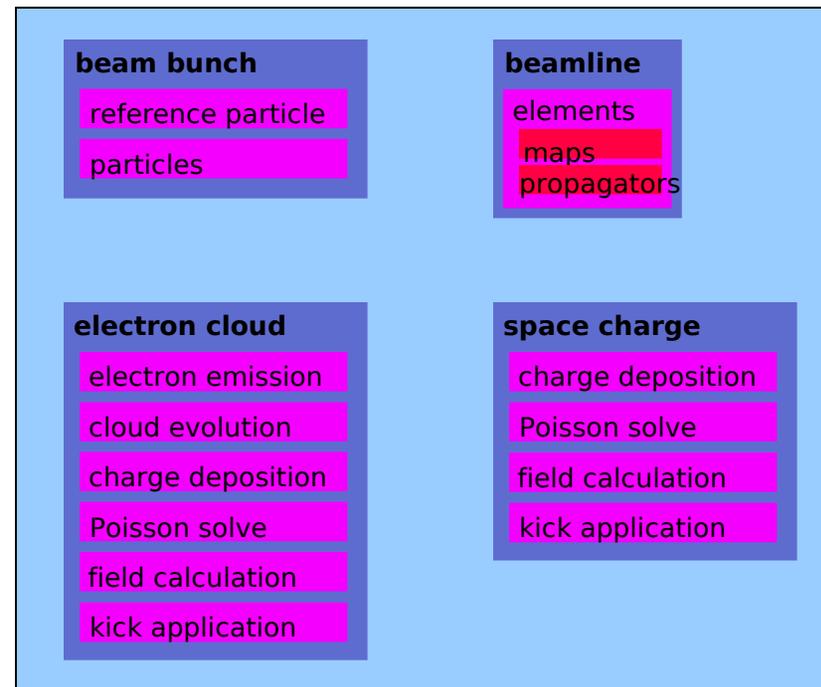
Synergia.ChannelSimSolver: a Synergia2 component also using Sphyaena; this component provides a Synergia.PoissonSolver port and uses both a Synergia.ParticleStore port and a Synergia.BeamOpticsPort port

testsp.ChannelSimDriver: a driver component for testing the Synergia.BeamSolver, Synergia.BeamBunch, and Synergia.BeamOptics components; the component provides a *go* port and uses a Synergia.PoissonSolver port

testsp.ChannelDriver: a standalone driver component used for prototyping and testing the pieces of the Synergia2 framework that can be imported into a CCA-based framework; the component provides a *go* port

Beam Dynamics: Defining Interfaces

- Refactoring Synergia2 and exploring interface issues for common functionalities
 - Beam bunch
 - Demonstrated interchanging CHEF and MaryLie beamline components at the map level, even though beamline models themselves are very different
 - Beamline
 - Synergia2 can use space charge modules from either IMPACT or Sphraena
 - Space charge
 - Electron cloud
- Challenges
 - **Granularity:** Overheads that apply *per particle* get an extra factor of $\sim 10^7$
 - unacceptable ... use aggregation
 - **Parallel decomposition** of fields, etc., must be compatible: may force coarser granularity



Reference: *Multiscale, Multiphysics Beam Dynamics Framework Design and Applications*, J. Amundson, D. Dechow, L. McInnes, B. Norris, P. Spentzouris and P. Stoltz, J. Phys.: Conf. Ser. 125 (2008) 012001, available via:

<http://www.iop.org/EJ/abstract/1742-6596/125/1/012001>

Initial Simulations Using Synergia2 Components

- See code at pcac.fnal.gov
 - FODO cell demo
 - Apply space charge kick
- Exploring performance, including scalable Poisson solvers, which are essential for beam dynamics simulations
 - FFT via IMPACT
 - Sphyaena-based implementations
 - Possibly others developed by ComPASS participants

Ongoing and Future Work

- **Immediate priorities:** Critical for COMPASS component integration
 - Collaborate with TASCs to address
 - Babel/SIDL interlanguage capabilities with struct support, broad support of Fortran compilers
 - Ability to run on leadership class facilities (including Cray XT4, BG/P)
 - Complete initial componentization of Synergia2
 - Evaluate performance of original Synergia application and component variant on Project-X and LARP simulations, with emphasis on beam dynamics applications for space charge and wakefields
- **Longer-term challenges:** Collaborate with TASCs, PERI, and TOPS to address issues in Computational Quality of Service (CQoS) for accelerator simulations,
 - How, during runtime, can we make sound choices for reliability, accuracy, and performance, taking into account the problem instance and computational environment?
 - **Composition:** select initial component implementations and configuration parameters
 - **Reconfiguration:** change parameters
 - **Substitution:** change implementations

Synergia2 & CQoS

