Center for Center for Technology for Advanced Scientific Component Software (TASCS)

http://tascs-scidac.org

Consolidated Progress Report
July 2006–March 2009

Proposal Number:	100990
Technical Contact:	Osni A. Marques
Program Area:	SciDAC Center for Enabling Technology
Program Office:	Office of Advanced Scientific Computing Research
Program Announcements:	LAB 06-04 and DE-FG02-06ER06-04

Lead Principal Investigator:

Institution	*Principal Investigator*
Oak Ridge National Laboratory	David E. Bernholdt
PO Box 2008, MS 6016	(865) 574-3147
Oak Ridge, TN 37831–6016	bernholdtde@ornl.gov

Participating Institutions

Institution	*Institutional Lead co-PI*
Argonne National Laboratory (ANL)	Lois Curfman McInnes
Binghamton University (BU)	Madhusudhan Govindaraju
Indiana University (IU)	Randall Bramley
Lawrence Livermore National Laboratory (LLNL)	Tom Epperly
Oak Ridge National Laboratory (ORNL)	James A. Kohl
Pacific Northwest National Laboratory (PNNL)	Jarek Nieplocha
Sandia National Laboratories (SNL)	Rob Armstrong
Tech-X Corporation (Tech-X)	Svetlana Shasharina
University of Maryland (UMD)	Alan L. Sussman
University of Oregon (UO)	Matt Sottile
Virginia State University (VSU)	Kostadin Damevski

Contents

1 Introduction and Project Overview

1.1 Background and Goals

A resounding success of the Scientific Discovery through Advanced Computing (SciDAC) program is that high-performance computational science is now universally recognized as a critical aspect of scientific discovery [71], complementing both theoretical and experimental research. As scientific communities prepare to exploit unprecedented computing capabilities of emerging leadership-class machines for multi-model simulations at the extreme scale [72], it is more important than ever to address the technical and social challenges of geographically distributed teams that combine expertise in domain science, applied mathematics, and computer science to build robust and flexible codes that can incorporate changes over time. The Center for Technology for Advanced Scientific Component Software (TASCS)[1] tackles these these issues by exploiting component-based software development to facilitate collaborative high-performance scientific computing.

Why Components in Scientific Simulation? Component-based design constructs complex software by combining simpler building blocks called components [73]. This approach has been developed by the computer science community as the latest in a succession of approaches to deal with the burgeoning complexity of software systems (Fig. 1). Like the object-oriented concepts from which they evolved, components encapsulate functionality and related state, but as their name implies, components are meant to be *composed*. All components, whether intrinsically object oriented or not, must have a common means of composition, which object-oriented concepts alone do not provide.

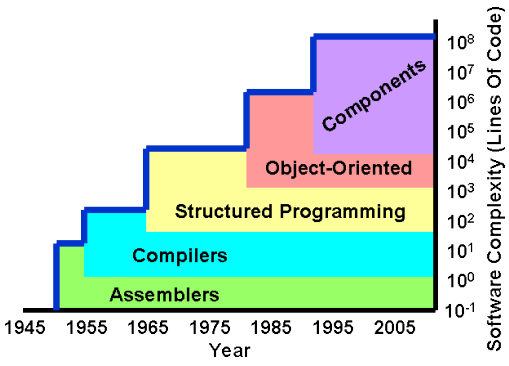

Figure 1: Computer scientists have developed a succession of techniques to address the increasing size and complexity of software systems, with component technology as the latest to achieve wide-spread acceptance. (Axes are qualitative rather than quantitative.)

Component-based applications are composed from individual components based on the interfaces they expose. Thus, components become a tool to organize large, complex software systems in terms of smaller modules of manageable scale, with components and interfaces serving as fundamental units in describing software architectures. In addition, because each component typically represents the work of an individual or a small group, component-based approaches can help address a variety of sociological and technical challenges typical in computational science teams that are increasingly multidisciplinary, geographically distributed, and dealing with multi-language programming in complex computing environments with lifetimes much shorter than the software running on them.

Besides providing an organizational basis for "software in the large," component approaches offer many additional benefits. For example, encapsulation and well-defined interfaces facilitate interoperability and reuse. While traditional software libraries also accomplish this function, components enable a key additional feature: the automation of composition under program control and rearrangement of functionality "on the fly." This automation property creates new opportunities for the real-time adaptivity of components to improve accuracy, robustness, and performance (Sec. 3.3). Similarly, the framework in which component-based applications are assembled and executed provides a platform for making services available to all participating components, such as orchestration of teams of parallel tasks (Sec. 3.1) and verification of interface contracts to improve software quality (Sec. 3.2).

[1] `http://tascs-scidac.org`

Components in SciDAC. Although component-based software engineering (CBSE) has gained wide acceptance in most other areas of computing, until recently it has made few inroads into high-performance computational science and engineering. Because scientific simulations must run in exotic high-performance environments, and there is little tolerance for inefficiencies that impede performance, computational science has special requirements that mainstream commodity component environments do not address.

In 1998, the Common Component Architecture Forum[2] was formed as a grass-roots organization of researchers at national laboratories and universities with the goal of fundamentally changing the way scientific software is developed and used by creating a component model targeted to the needs of high-performance scientific computing – the CCA. The CCA addresses these issues and provides key features of language-neutral specification of common component interfaces, interoperability for software written in programming languages important to scientific computing, and dynamic composability, all with minimal runtime overhead [74].

In 2001, a core group of CCA Forum members established the Center for Component Technology for Terascale Simulation Software (CCTTSS) under the SciDAC-1 program. The CCTTSS successfully developed the CCA to the prototype stage and demonstrated the practical and potential benefits of component-based software development (CBSD) through work with several scientific application teams. The Center also made significant efforts at community-building, outreach, and education, thereby encouraging computational scientists to make long-term investments in their software, and increasing the awareness of software engineering considerations and potential roles for components and related technologies.

The Center for Technology for Advanced Scientific Component Software began in 2006 under the SciDAC-2 program with the goal of taking the CCA and high-performance computing (HPC) component technology in general from the prototype to the production stage of maturity and quality. As discussed in Sec. 2 and Appendix A, TASCS is now enabling diverse multidisciplinary teams to investigate new scientific questions by leveraging the CCA component approach. Moreover, TASCS explores HPC component technology as a platform for providing application scientists with additional tools and productivity advantages that are not available, and in many cases would be extremely hard to deploy, in traditional HPC programming environments. As discussed in Sec. 3, these initiatives set the stage for new software capabilities that we believe are essential aspects of extreme-scale computational science collaborations on emerging leadership-class machines.

1.2 Project Organization and Research Agenda

TASCS's work has four major thrust areas, each with several activities (described below), around which the main portion of this progress report is organized (Fig. 2). The work is strongly motivated by numerous interactions with scientific applications teams and other SciDAC CETs and Institutes (Appendix A), and activities are tightly integrated, both across the four thrust areas and across the participating institutions.

- **Component Technology Initiatives** utilize and extend the component model to provide new "value added" capabilities for CCA users (Sec. 3).
  - **Emerging HPC Hardware and Software Paradigms** motivate our creation of component-based tools to help applications manage higher/hybrid levels of parallelism through a multiple-component multiple-data (MCMD) paradigm; we also are developing support for fault-tolerant components and hardware co-processors (Sec. 3.1).
  - Our work in **Software Quality and Verification** develops mechanisms to specify and verify functional software "contracts" associated with component interfaces to help both providers and users improve the quality of their codes (Sec. 3.2).

[2] `http://www.cca-forum.org`

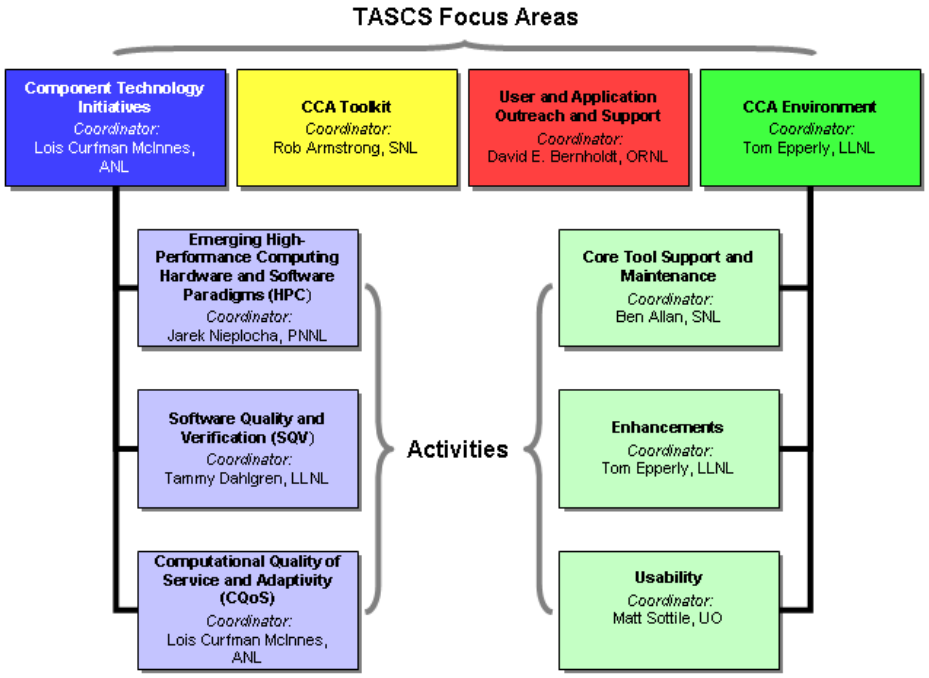

Figure 2: TASCS organizational chart.

Table 1: TASCS Research Areas and Annual Budget by Institution

Institution	Research Areas[a]	Budget ($k)
Argonne National Laboratory	*Initiatives*, *CQoS*, Core Tools, Usability, Toolkit, Outreach	505
Binghamton University	Enhancements	93
Indiana University	Toolkit, Outreach	127
Lawrence Livermore National Laboratory	*SQV*, *Environment*, Core Tools, *Enhancements*, Usability, Toolkit, Outreach	537
Oak Ridge National Laboratory	HPC, SQV, Core Tools, Enhancements, Usability, Toolkit, *Outreach*	500
Pacific Northwest National Laboratory	*HPC*, Toolkit, Outreach	329
Sandia National Laboratories	CQoS, *Core Tools*, Enhancements, Usability, *Toolkit*, Outreach	538
Tech-X Corporation	Toolkit, Outreach	87
University of Maryland	Toolkit	62
University of Oregon[b]	*Usability*	94
Virginia State University[c]	Enhancements, Toolkit	128
Total		3,000

[a] Coordinating institutions for each research area are shown in *italics*.
[b] Originally Los Alamos National Laboratory
[c] Originally University of Utah

  - Research in **Computational Quality of Service and Adaptivity** leverages CCA mechanisms to dynamically adapt long-running component-based applications in response to changing conditions (e.g., performance, accuracy, robustness) by composing, substituting, and reconfiguring components on the fly (Sec. 3.3).

- The **CCA Environment** thrust supports and improves the foundation of the CCA environment and tools for ease of use and as core technology for other initiatives (Sec. 4).
  - **Core Tool Support and Maintenance** provide essential software maintenance, porting, and support in the face of changing HPC environments (Sec. 4.1).
  - **Enhancements** extend the CCA environment with additional capabilities required by users and by other activities within TASCS (Sec. 4.2).
  - **Usability** work makes CCA technology more accessible to users through the development of tools for the (semi-)automatic wrapping of existing code into components (Sec. 4.3).

- The **CCA Toolkit** provides a diverse suite of scientific components, along with a basic software skeleton to facilitate the creation of new components (Sec. 5).

- **User/Application Outreach and Support** assist scientific teams with CCA usage through direct interactions as well as the development of documentation, tutorials, and example materials (Sec. 6).

The TASCS team currently includes researchers from five national laboratories, five universities, and one small research business (Table 1); several changes to the project team since the project's inception are described in Sec. 7. At most institutions, the funding partially supports multiple researchers, ranging from faculty and senior researchers to graduate students and undergraduate interns. Most institutions are engaged in multiple activities within the project, as noted in Table 1 and at the start of each section of this report. Note that while laboratory participants received funding beginning in July 2006, university participants did not receive their award notifications until January 2007. This fact, and funding gaps associated with institutional changes, have delayed some work as described in Sec. 4 and Sec. 5.

We also collaborate extensively with a wide variety of external projects (Appendix A). Such collaborations are an important part of TASCS. In addition to providing a testing ground for the results of our work, many of these collaborations inspire new research ideas and developments. As discussed in Sec. 3, several projects are direct research partners in our three Component Technology Initiatives.

1.3 Overview of Milestones and Deliverables

The following tables provide an overview of the milestones and deliverables of the project, taken from the Project Management Plan Version 0.91 of 4 December 2006[3], broken down by thrust area and activity. In each section, the coordinator and the participating institutions are listed.

[3] `http://tascs-scidac.org/documents/mgmt-plan.pdf`

Table 2: **Summary of Milestones for Component Technology Initiatives**
Coordinator: L.C. McInnes, ANL; *Participants:* ANL, IU, LLNL, ORNL, PNNL, SNL, UMD

Year 1	*Year 2*	*Year 3*	*Years 4–5*
Emerging HPC Paradigms	*Coordinator:* J. Nieplocha, PNNL;		*Participants:* ORNL, PNNL
• Develop multi-level parallelism model. • Define abstract model for CCA hybrid apps.	• Develop CCA model for processor groups. • Develop component interface for hybrid systems.	• Develop simple MCMD programming model. • Prototype hybrid interface for example application.	• Incorporate MCMD support for heterogeneous prog. models. • Implement hybrid & MCMD example application components.
Software Quality and Verification	*Coordinator:* T.L. Dahlgren, LLNL;		*Participants:* LLNL, ORNL
• Identify and define CQoS and domain-specific semantics; assess spec. mechanisms.	• Develop semantics prototype(s). • Design method invocation sequencing constraints enforcement.	• Introduce semantic specifications into selected Toolkit components. • Develop sequencing enforcement prototype in Babel/SIDL.	• Evaluate semantics prototype(s). • Evaluate sequencing enforcement prototype. • Revise and evaluate prototypes based on CQoS evolution.
Computational Quality of Service (CQoS)	*Coordinator:* L.C. McInnes, ANL;		*Participants:* ANL, SNL
• Collect application requirements, define metrics, perform base experiments. • Build database component.	• Populate CQoS testbed and specify initial CQoS API. • Develop initial performance models for applications. • Develop proxy port generation for CQoS usage.	• Complete design of overall CQoS strategy. • Implement application control laws. • Implement an asynchronous control infrastructure.	• Design APIs for general analysis engines. • Create a generic CQoS framework for HPC applications. • Stress test CQoS tools.

Table 3: **Summary of Milestones for CCA Environment**
Coordinator: T. Epperly, LLNL; *Participants:* ANL, BU, LLNL, ORNL, SNL, UO, VSU

Year 1	*Year 2*	*Year 3*	*Years 4–5*
Core Tool Support and Maintenance	*Coordinator:* B. Allan, SNL;		*Participants:* ANL, LLNL, ORNL, SNL
⟵ Support helpdesk and open bugtracking. ⟶			
⟵ Develop and maintain technical documentation. ⟶			
• Port CCA software stack to NLCF machines	• Complete CCA Conformance Tests	• Automated conformance testing for all CCA frameworks.	• Evaluate and port to new architectures as they emerge.

(continued)

Table 3: **Summary of Milestones for CCA Environment** *(continued)*

Year 1	*Year 2*	*Year 3*	*Years 4–5*
Enhancements	*Coordinator:* T. Epperly, LLNL;	*Participants:* BU, LLNL, ORNL, SNL, VSU	
• Adopt EventService and MPIService into standard. • Demonstrate support for BabelRMI in XCAT.	• Demonstrate CCA/Kepler interoperability. • Add structs to SIDL/Babel. • Add Fortran 2003 support to Babel. • Incorporate SOAP as module in BabelRMI, integrate with Proteus/ XCAT.	• Finalize specification for Component sub-assemblies. • Develop specification for framework interoperability. • Release full fledged version of XCAT.	• Demonstrate exchange of sub-assemblies between two CCA Frameworks. • Demonstrate framework interoperability between CCA implementations. • Extend BabelRMI communication modules for new CCA applications.
Usability	*Coordinator:* M. Sottile, UO;	*Participants:* LLNL, ORNL, SNL, UO	
• Draft CCA-Lite Spec and CCA-Lite Framework. • Document advanced component debugging techniques. • Design component test harness.	• Preliminary integration of CCA-Lite test framework with Ccaffeine framework. • Deploy component test harness.	• Demonstrate connecting CCA-Lite components to CCA components in Ccaffeine. • Integrate SIDL semantics enforcement into testing methodology. • Develop component tracing tools to facilitate debugging.	• Demonstrate source-to-source conversion of CCA-Lite component to full CCA Component. • Evaluate tradeoffs in debugging and testing CCA-Lite vs. full CCA. • Apply test harness to selected toolkit components.

Table 4: **Summary of Milestones for the CCA Toolkit**
Coordinator: R. Armstrong, SNL; *Participants:* ANL, IU, LLNL, ORNL, PNNL, SNL, Tech-X, UMD, VSU

Year 1	*Year 2*	*Year 3*	*Years 4–5*
⟵ Design, establish and, based on user feedback, iterate and improve CCA Base Installation ⟶			
• Design toolkit structure and contribute initial components to the Toolkit, and establish web distribution system	• Incorporate and promulgate Toolkit components into CCA tutorial and outreach activities, improve type and quality of the Toolkit repertoire.	• Add to Toolkit component improvements to CCA architecture since Year 1, e.g. MCMD components, templates, and CQoS plug-ins.	• Establish web-based/community process for approving/distributing component contributions from the community, as user base for Toolkit expands.

Table 5: **Summary of Milestones for Application and User Outreach and Support**
Coordinator: D.E. Bernholdt, ORNL; *Participants:* ANL, IU, LLNL, ORNL, PNNL, SNL, Tech-X

Year 1	*Year 2*	*Year 3*	*Years 4–5*
⟵ Support applications in adopting and using CCA. ⟶			
⟵ Deliver user support, incl. tutorials, coding camps, etc. ⟶			
⟵ Update tutorial and best practices documentation. ⟶			
• Revamp `cca-forum.org` web services.	• Revamp or migrate `cca-forum.org` code development services.		

2 Project Highlights

FACETS Project Uses CCA Tools for First Integrated Core-Edge Plasma Simulation. The Framework Application for Core-Edge Transport Simulations (FACETS) project (Sec. A.1.7) is developing integrated modeling capabilities for the plasma core, edge, and wall in Tokamak reactors as one of three SciDAC prototype projects for the Fusion Simulation Project. This problem involves complex physics with different dimensionalities, modeled by separate codes. The FACETS team uses the CCA's Scientific Interface Definition Language (SIDL) to express the interfaces between the components representing the core, edge, and wall physics, and the Babel language interoperability to integrate the UEDGE code into the custom FACETS framework. With this integrated application, FACETS has recently achieved the first tightly coupled code-edge fusion plasma model (Fig. 3) [75]. This result provides an important proof of concept that opens the door to further scientific discovery, parallelization research, and additional performance optimization.

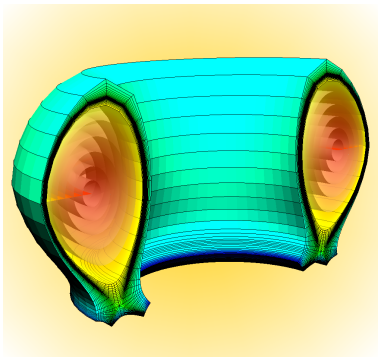

Figure 3: FACETS uses Babel to combine software components written in multiple languages to produce the first integrated core-edge simulation [75]. *Courtesy of John Cary; produced with VizSchema (Tech-X).*

CFRFS Researchers Demonstrate Their Toolkit in a Fourth-Order in Space AMR Simulation. The Computational Facility for Reacting Flow Science (CFRFS) project (Sec. A.1.6) is developing an AMR toolkit for simulating reacting flows with detailed chemistry. The CCA standard is used to incorporate contributions from combustion researchers and established software libraries (e.g., CHOMBO from the Algorithmic and Software Framework for Applied Partial Differential Equations (APDEC) (Sec. A.1.1) and hypre from the Towards Optimal Petascale Simulations (TOPS) Center (Sec. A.1.13)). Using this CCA-based toolkit, a CFRFS achievement – fourth-order spatial discretizations on a block-structured adaptive mesh – was demonstrated recently [76, 77] within the context of methane-air ignition. Fourth-order convergence was demonstrated empirically. Fig. 4 shows a snapshot from a simulation of a "bubble" of methane igniting in a pre-heated air flow in a pipe. A 3-level AMR mesh is seen to resolve the ignition spot and the edge flame that burns through a region surrounding the methane-air interface where the two mix and form a combustible mixture. CFRFS researchers are also developing a meta-partitioner for reactive flows on block-structured, adaptively refined meshes using new CCA infrastructure for computational quality of service (see Sec. 3.3).

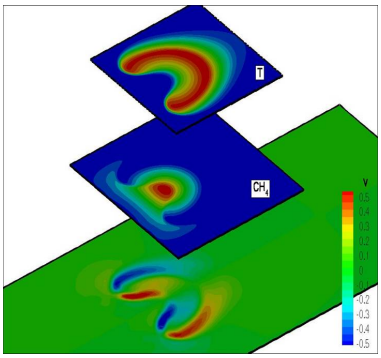

Figure 4: CFRFS researchers used a CCA-based AMR toolkit for reacting flows to simulate CH_4-Air ignition [76]. *Courtesy of Cosmin Safta (SNL).*

Components Enable Interchangeability and Interoperability in Quantum Chemistry. Researchers in the Quantum Chemistry Strategic Application Partnership (QCSAP) (Sec. A.1.10) have used the CCA to achieve interoperability among three leading high-performance chemistry applications (GAMESS, MPQC, and NWChem). SIDL/Babel has eliminated difficulties with interoperability among Fortran, C++, and Python, thereby enabling the adoption of a common high-level interface for a chemistry model component

that each chemistry package supports [78] and the use of TAO numerical optimization components from the TOPS project [81] (Sec. 5 and A.1.13). Chemists are now performing hybrid quantum/classical simulations using methods available within any of the packages [80] and are sharing low-level capabilities such as integral evaluation [79], thus enabling rapid development of novel methods. Explicitly correlated calculations including relativistic effects, which were rapidly implemented using components, were employed to revise the heats of formation of chromium compounds of interest in industrial applications by 2–3 kcal/mol [82]. Quantum chemists are also pursuing multilevel simulations using new component capabilities for emerging HPC architectures (Sec. 3.1.1) and are employing new CCA infrastructure for computational quality of service (Sec. 3.3) for dynamic and adaptive configuration [1].

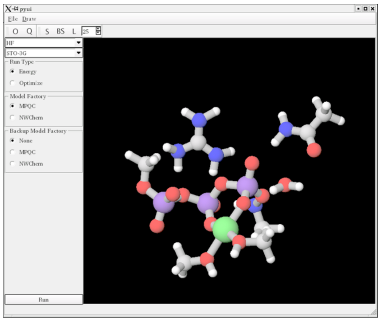

Figure 5: The CCA enables hybrid quantum/classical simulations using GAMESS, MPQC, and NWChem [79, 80]. *Courtesy of the QCSAP project.*

Bocca Revolutionizes CCA Component Development. While the CCA provides powerful ideas and tools, navigating the details of creating components in a multi-language environment can be challenging. To help lower this barrier, a small team of TASCS researchers, including Boyana Norris (ANL), Ben Allan (SNL), Wael Elwasif (ORNL) and Rob Armstrong (SNL), developed Bocca, a code-generation and project management tool for the CCA.

Given the definition of a component in terms of the ports it uses and provides, Bocca can generate a complete component in any Babel-supported language, including a build system, but *sans* the actual implementation. This capability enables a component skeleton to be built and loadable into a framework in seconds, thereby allowing computational scientists to focus on the functional parts of the code. Described in more detail in Sec 4.3, Bocca's impact has been profound, completely changing the way users approach a CCA project (see Fig. 10). After the CCA tutorial was rewritten to use it, we found that students were able to make 200% more progress through the hands-on exercises and had a better grasp of the CCA environment. Since Bocca was publicly released in April 2007, it has been adopted almost universally among CCA users.

Special Recognition for TASCS Researchers. In 2006, David Bernholdt (ORNL) was invited to teach a course as part of the CEA-EDF-INRIA Summer School on the Design of High-Performance Scientific Applications [2] in St. Rémy lés Chevreuse, France. The course, entitled *Software Architecture for High-Performance Computing: A Pragmatic Approach*, included 8.5 hours of lecture material and a 3-hour hands-on CCA tutorial.

In 2007, Ken Chiu (BU) collaborated with researchers Wei Lu, Satoshi Shriasuna, and Dennis Gannon (IU) on a paper entitled *A Hybrid Decomposition Scheme for Building Scientific Workflows* [3], which won the Best Paper Award at the High Performance Computing Symposium, in Norfolk, Virginia.

Also in 2007, David Bernholdt (ORNL) gave a keynote address entitled *The Common Component Architecture: Building Frameworks for Computational Science* [4] at the International Conference on Modeling and Simulation in the Petroleum Industry in Porto de Galinhas, Brazil.

3 Component Technology Initiatives

Coordinator: Lois Curfman McInnes, ANL

The TASCS Center is pursuing three technology initiatives based on the premise that, in addition to aiding multidisciplinary software development, the CCA component approach can facilitate the deployment of new capabilities throughout the entire lifecycle of extreme-scale scientific simulations. This work is motivated and validated through collaborations with SciDAC science teams working in combustion (Sec. A.1.6), quantum chemistry (Sec. A.1.10), fusion modeling (Sec. A.1.7, A.1.4), accelerator modeling (Sec. A.1.5), subsurface modeling (Sec. A.1.9), and proteomics (Sec. A.2.6), as well as with mathematicians in the ITAPS (Sec. A.1.2) and TOPS (Sec. A.1.13) centers and computer scientists in the Coordinated Infrastructure for Fault Tolerance in Systems (CIFTS) project (Sec. A.2.4).

These initiatives address challenges at different but related levels in the development of component-based scientific software [5]. The initiative on *Emerging HPC Hardware and Software Paradigms* (Sec. 3.1) leverages component technology for applications that target massively parallel, heterogeneous architectures. The second initiative, *Software Quality and Verification* (Sec. 3.2), investigates new approaches to the lightweight runtime enforcement of behavioral semantics that help developers to use interfaces correctly, with little impact on performance. By exploiting component automation, including capabilities for plugging and unplugging components during execution, the initiative on *Computational Quality of Service and Adaptivity* (Sec. 3.3) develops tools to help application scientists choose among alternative algorithmic implementations and parameters, thereby creating new opportunities to enhance the performance of CCA applications. Research in all three initiatives in turn motivates extensions to the CCA specification and the development of new core CCA technologies, including a new event service specification discussed in Sec. 4.2. Since the start of the TASCS project, this work has been discussed in a variety of papers, presentations, and posters [1, 5–22] as well as a Ph.D. dissertation [23].

3.1 Emerging HPC Hardware and Software Paradigms

Coordinator: Jarek Nieplocha, PNNL
Participants: ORNL, PNNL

The overall goal of this initiative is to leverage the CCA's unique capabilities for flexible high-performance software to help domain scientists fully harness the unprecedented computing power of emerging leadership-class machines. By providing an easy-to-use intermediate layer between hardware and scientific programmers, CCA components can make new advances in emerging HPC paradigms readily usable by applications teams, thereby helping domain scientists to manage the complexity of these changes over time. Three focus areas are support for multiple levels of parallelism via a Multiple-Component-Multiple-Data (MCMD) paradigm (Sec. 3.1.1), support for increasingly hybrid and heterogeneous hardware (Sec. 3.1.2), and component resilience in response to fault tolerance challenges (Sec 3.1.3).

3.1.1 Multiple-Component-Multiple-Data (MCMD)

Collaborators. TASCS participants in this initiative are Daniel Chavarria, Ian Gorton, Manojkumar Krishnan, Jarek Nieplocha (PNNL); David Bernholdt and Wael Elwasif (ORNL). To motivate and validate this work, TASCS researchers collaborate with the Groundwater Subsurface GWACCAMOLE Scientific Application Partnership (Sec. A.1.8), the NWChem team (Sec. A.2.8), the STOMP (Sec. A.1.9) project, the Center for Simulation of Wave Interactions with Magnetohydrodynamics (SWIM) (Sec. A.1.4), and the CIFTS (Sec. A.2.4) project.

Motivation. As high-end computers transition from offering thousands of processors to tens and hundreds of thousands of processors, users are increasingly challenged to find additional parallelism in their application in order to effectively utilize these emerging machines. When a single parallel activity does not scale sufficiently, it would often be desirable to launch many distinct parallel tasks, each using a subset of the

available processors. This type of programming goes by many names, including multi-level parallelism, multiple-program multiple-data (MPMD) parallelism, multi-tasking, etc. Traditional parallel programming models provide only the most basic support for this kind of programming, if they provide any support at all. The component environment, on the other hand, provides an excellent platform to make such capabilities broadly available.

Accomplishments. The dynamic nature and encapsulation of components map quite naturally to the concepts of MPMD programming. We call this approach the *Multiple Component Multiple Data* (MCMD) programming model. Our primary goal is to define, develop, and deploy high-level infrastructure for MCMD programming in a CCA environment to support applications based on multiple levels of parallelism.

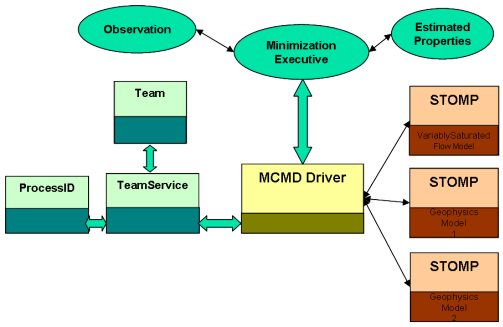

Figure 6: MCMD design of STOMP subsurface simulations for hydrologic/geophysical inverse modeling.

PNNL team members led the formation of a CCA Forum working group on MCMD programming, which held a workshop during the January 2007 meeting of the CCA Forum to explore motivating MCMD applications, including hierarchical parallelism in computational chemistry [83], co-op parallelism, ab initio nuclear structure calculations, coupled climate modeling, space weather modeling, molecular dynamics, and groundwater multiphysics simulations. Based on these use cases, the MCMD working group developed a model for MCMD programming as well as a simple yet generic processor-group model that can map to processor groups in various parallel programming models. The processor-group model supports both single-executable parallel job as well as multi-partition applications (e.g., multiple parallel jobs). Multi-partition applications can be homogeneous (e.g., all parallel jobs are MPI based) or heterogeneous (i.e., jobs may contain a combination of multiple programming models such as MPI, GA, PVM, etc). The MCMD group also developed a draft specification for a *teams* service to manage concurrent groups of processes at the component level. The concept of a CCA *team* refers to an ordered collection of processes yet is more general than an MPI group; for example, a team can contain a list of processes from multiple parallel runs.

To implement the high-level infrastructure for MCMD programming, we also developed a high-level API for process group management at the CCA level. To express and manage hierarchical parallelism though processor groups, it is essential to support processor groups at the component level. The MCMD programming model promotes parallelism at the component level, parallelism within the component, and parallelism within a subroutine. The CCA MCMD model supports various execution models (e.g., single vs. multiple mpiruns); different programming models (e.g., MPI, Global Address Space (GAS) models including Global Arrays (GA), CAF, etc.); global process and group IDs; and process and group ID translators to facilitate the translation of a CCA process group to an MPI group.

To date, we have five releases of the MCMD API specification based on feedback from several MCMD meetings and teleconferences. The MCMD specification is available for download at the MCMD working group webpage. We also developed a prototype implementation of the MCMD specification, which can be used in application evaluation.

Application Impact and Future Work. The PNNL team is working with the STOMP subsurface modeling group to add MCMD capabilities to their code. Fig. 6 shows the MCMD model for doing large-scale subsurface simulations. This work will provide a validation of the CCA *teams* model, while allowing the

STOMP group to carry out parameter studies using multiple instances of the parallel STOMP component to simulate flows through heterogeneous, porous media at the Hanford site. The MCMD implementation incorporates task parallelism between components and data parallelism within components. PNNL has also been working on an MCMD implementation to facilitate the estimation of spatially variable subsurface geologic material properties using field observations of multivariate data types and sample sizes, and to support a multimodal minimization methodology using forward models of hydrologic, geochemical, and geophysical processes to estimate the desired properties.

The ORNL team is working in collaboration with the SWIM fusion project to implement MCMD capabilities based on the *teams* approach in their component-based (but not CCA-compliant) Integrated Plasma Simulator (IPS) framework. This application is not one considered in the original design of the *teams* model, and parallel tasks are implemented at the mpirun level, thus providing broader validation of the design.

3.1.2 CCA on Heterogeneous Architectures

Collaborators. Work on heterogeneous architectures is being carried out by PNNL researchers in collaboration with Polygraph project (Sec. A.2.9) and the High Performance Mass Spectrometry group (Sec. A.2.6) at the Environmental Molecular Sciences Laboratory at PNNL.

Motivation. Hybrid high-performance computing systems, which incorporate mainstream processors as well as specialized hardware engines, are quickly becoming a reality at many supercomputer centers. Systems such as LANL's Roadrunner and the Tokyo Institute of Technology's TSUBAME cluster have been built from nodes utilizing mainstream processors (x86 CPUs) and mainstream high-performance networks (InfiniBand); these systems integrate specialized hardware accelerators on each compute node: Cell blades in the case of Roadrunner and ClearSpeed SIMD accelerators in the case of TSUBAME. Vendors such as SGI and Cray offer systems that incorporate Field Programmable Gate Array (FPGA)-based accelerators as part of mainstream HPC systems. Other projects have explored the potential of Graphics Processing Units (GPUs) for the same acceleration purpose. Hybrid approaches are of interest as alternatives to the more evolutionary path of homogeneous systems of multicore processors.

The performance advantages of specialized hardware accelerators over mainstream processors have been well studied and understood. However, the design and implementation complexity required to develop high-performance, scalable hybrid applications on such systems has not been yet addressed. In addition, most hybrid system vendors have developed proprietary interfaces to their accelerators and in some cases have managed to expose accelerated functionality through common library interfaces such as BLAS and LAPACK. However, it is clear that in most cases having accelerated libraries is not enough to satisfy the performance of applications, which involve many custom, domain-specific algorithms. In these cases, application developers need to create accelerated implementations of their custom algorithms, and they prefer to have a reasonable level of portability for their code.

Accomplishments. We have studied the use of CCA components for creating portable interfaces for custom algorithms implemented on accelerators. This work leverages the clear software engineering benefits of component design in terms of providing encapsulation, separation of concerns, and clean interfaces as well as portability and maintainability of codes. These components can cleanly encapsulate the entire low-level, platform-specific details of hardware accelerated implementations, while providing a high-level interface that is 100% compatible with components implementing software versions of the same algorithm.

To demonstrate this concept, we created a component-based version of an application in proteomics named Polygraph (Sec. A.2.9), which uses an innovative approach to extract mass spectra from experimental databases given a description of a candidate peptide. Polygraph employs a genetic algorithm approach to find the best matches for the candidate peptide sequence, which is a compute intensive process for long sequences on large databases.

We developed an FPGA-accelerated version of a key computational kernel (`fpgenerate()`) in Poly-

graph and demonstrated the ease of integrating the FPGA-accelerated kernel into the rest of the software application using CCA component technology. We also built a CCA component of the original software version of the `fpgenerate()` kernel and demonstrated seamless interoperability between the FPGA and software versions, with very low overhead compared to non-componentized, monolithic versions [7]. We used a new CCA-based high-performance event service to distribute work and coordinate execution between CPU-only nodes and nodes with attached accelerators (e.g., FPGA, GPU, etc.) on a HPC system (see Sec. 4.2). We also evaluated the use of the event service in an application for high-speed processing of data from a mass spectrometer biology application [8, 9].

Application Impact and Future Work. This preliminary work with the Polygraph team has demonstrated the effectiveness of CCA component support for heterogeneous architectures both in terms of performance and ease of use. We plan to extended functionality based on the priorities of the Polygraph project and the Environmental Molecular Sciences Laboratory at PNNL and to generalize capabilities in support of additional scientific applications.

3.1.3 Fault Tolerance

Collaborators. Current work on fault tolerance is being carried out by ORNL researchers in collaboration with the CIFTS project (Sec. A.2.4). The concepts are being prototyped in collaboration with the SWIM fusion project (Sec. A.1.4).

Motivation. The primary focus of the CIFTS project is the development of a Fault Tolerance Backplane (FTB) to disseminate fault information throughout the software stack, from hardware, device drivers, and operating systems, up to user libraries and applications, in order to facilitate coordinated responses to problems and failures. The goal of our collaboration with CIFTS is to make FTB events available to CCA applications, and to use this capability to explore fault resilience in the CCA environment.

Accomplishments. The TASCS–CIFTS–SWIM collaboration is currently working to prototype fault tolerance capabilities in the component-based (but not CCA-compliant) IPS framework for plasma simulations. An event service modeled on the proposed CCA Event Service (Sec. 4.2) has been implemented and bridged to the FTB. As mentioned above (Sec. 3.1.1), an MCMD capability is also under development to support concurrent execution of multiple parallel tasks within a coupled fusion simulation. Through this collaboration, such simulations will be able to gracefully tolerate failure and restoration of nodes, selectively restarting only failed tasks rather than the whole simulation.

Application Impact and Future Work. Once the CCA Event Service is packaged and readily available to the CCA community, we will develop the bridge between the FTB and the CCA Event Service, thus making fault-related events available to CCA applications in the same context as other types of events. With the fault-tolerant IPS, we plan to explore extensions to the CCA teams service to facilitate fault tolerance, as well as other abstractions for resilience that might usefully be encapsulated as reusable components.

3.2 Software Quality and Verification (SQV)

Coordinator: Tammy Dahlgren, LLNL
Participants: LLNL, ORNL

Collaborators. This work is performed primarily by Tammy Dahlgren (LLNL), along with Wael Elwasif and David Bernholdt (ORNL). The initiative is motivated in part by and involved collaborations with the Interoperable Technologies for Advanced Petascale Simulations (ITAPS) (Sec. A.1.2) and TOPS (Sec. A.1.13) CETs. Key collaborators in the Computational Quality of Service Initiative (see Sec. 3.3) are Li Li, Lois Curfman McInnes, and Boyana Norris (ANL); McInnes and Norris are also TOPS investigators. Lori Diachin and Kyle Chand (LLNL), as well as Carl Ollivier-Gooch (University of British Columbia), were invaluable collaborators on some of the early work with ITAPS.

Motivation. The Software Quality and Verification (SQV) Initiative focuses on improving the quality of component-based scientific software through verifiable or enforceable semantic annotations added to interface specifications [5, 6].

Much of the scientific computing community relies on documentation to describe the behavior and use of basic method signatures making up the application programming interface. Documentation is often incomplete or quickly becomes out-of-date during active software development. In the case of multiple implementations conforming to a common specification, such as the ITAPS unstructured mesh and the TOPS (non)linear solver interfaces, each implementation may be best suited to a different range of applications. These inconsistencies and limitations can lead to confusion and frustration for developers trying to use the software.

Enforceable interface contracts provide a uniform mechanism for documenting behaviors and constraints associated with components, automatically verifying component implementation conformance to the documented specification, and checking components are used by applications in a manner consistent with their designs (as defined in the specification). No routinely used HPC programming language provides such capabilities. While subroutines in traditional code often do some level of validation of inbound arguments, it is much rarer to find separate validation of outbound arguments. Moreover, such checks are not visible to the user of the routine, and are not easily controlled (turned off or on at runtime, e.g., for performance reasons). The component environment, on the other hand, provides a convenient venue for expressing interface contracts as well as for interposing enforcement and control between caller and callee without modifying component internals.

Accomplishments. Support for semantic annotations was integrated into SIDL, thereby extending basic method signatures for the expression of interface contracts independent of the implementation languages of the associated components. Currently, the specification of pre- and post-conditions and invariants are supported, though the latter is as yet untested. Contracts are translated into enforcement routines in the Babel middleware. The Babel runtime system provides control over enforcement decisions and related options. The October 2008 release of Babel 1.4.0 supports contract enforcement for C and C++ clients invoking server-side implementations in any of the Babel-supported languages (i.e., C, C++, Fortran 77 and 90, Java, and Python). Support for Python clients was recently added, and Java and Fortran 77 clients are being debugged.

```
bool entitysetGetNextWorkset (
  inout opaque workset_iterator,
  inout array<opaque>
              entity_handles)
  throws Error;
require
   workset_iterator != null;
ensure
   workset_iterator != null;
   result implies
       (entity_handles != null);
   (entity_handles != null)
       implies
       (dimen(entity_handles) == 1);
```

Figure 7: Example interface contract specification for a single method, from an early version of the ITAPS mesh interface specification.

An additional area of research is performance-sensitive contract enforcement as an alternative to simply enabling or disabling runtime enforcement [10, 12, 23]. Results to date suggest that adaptive sampling techniques can allow a much higher percentage of detected contract violations while limiting the overall performance impact of contract enforcement.

Application Impact and Future Work. A recent focus of work by our ANL collaborators involves incorporating selective, performance-driven interface contract enforcement capabilities in Babel [11] into CQoS infrastructure. They are also extending TOPS components for (non)linear solvers with assertions to improve performance and robustness. Babel's contract capabilities are being used to implement adaptive linear solver components for a parallel, nonlinear partial differential equation (PDE) application in the CQoS testbed.

Near-term plans include completing Babel support for interface contracts in all languages, extending the regression test suite and documentation, and adding an exercise on contracts to the CCA Tutorial. Longer-range work includes expanding the expressiveness of interface contracts with more built-in features, such as operations on two- to seven-dimensional SIDL arrays, basic method order sequencing constraints, and additional annotations identified through collaborations. The ORNL team recently obtained access to the

neutronics code deNovo, one of the very few scientific applications already making use of contracts, with the intent of comparing deNovo's contract capabilities to those provided in Babel. In addition to supporting the automatic enforcement of behavioral constraints, some semantic annotations could aid the identification and reconfiguration of suitable components as part of CQoS infrastructure.

LLNL recently initiated the hiring process for a student intern during summer 2009 to assist in extending the test suite, and ORNL is seeking a post-graduate researcher to expand its work in this initiative.

3.3 Computational Quality of Service (CQoS) and Adaptivity

Coordinator: Lois Curfman McInnes, ANL
Participants: ANL, SNL

Collaborators. TASCS participants in this initiative are Van Bui, Li Li, Lois Curfman McInnes, Boyana Norris (ANL); Rob Armstrong, Joseph Kenny, Nicole Lemaster, and Jaideep Ray (SNL). The design and development of CQoS infrastructure is a collaboration with the Software Quality and Verification (SQV) initiative (Sec 3.2), the TAU group (Sec. A.2.12) at the University of Oregon, and the Performance Engineering Research Institute (PERI, Sec. A.1.11). To motivate and validate this work, we interact closely with chemists in the QCSAP (Sec. A.1.10), the Computational Facility for Reacting Flow Science (CFRFS, Sec. A.1.6), the TOPS CET (Sec. A.1.13), and the FACETS (Sec. A.1.7) fusion and ComPASS (Sec. A.1.5) accelerator SciDAC projects. CQoS collaborators discussed the priorities of applications teams and strategy for design during all-hands sessions that preceded quarterly CCA Forum meetings in January 2007 and July 2007; team members communicate regularly via a project wiki, monthly telecons, and additional subproject sessions.

Motivation. As computational science progresses toward ever more realistic multiphysics applications, no single research group can effectively select or tune all components of a given application, and no solution strategy can seamlessly span the entire spectrum efficiently. Common component interfaces, along with programming language interoperability and dynamic composability, are key features of component technology that enable easy access to suites of independently developed algorithms and implementations. The challenge then becomes how to make sound choices when dynamically choosing among the available implementations and parameters, suitably compromising among accuracy, performance, and algorithmic robustness. Motivated by the needs of computational teams in quantum chemistry, combustion, fusion, and accelerator modeling [84], we are developing generic support for *computational quality of service* (CQoS) [85], or the automatic composition, substitution, and dynamic reconfiguration of components to suit a particular computational purpose and environment. CQoS embodies the familiar concept of quality of service in networking as well as the ability to specify and manage characteristics of the application in a way that adapts to the changing (computational) environment.

CQoS Testbed. An important initial step in this work has been the development of a CQoS testbed, which contains component codes that are representative of the issues faced by the fusion, accelerator, combustion, and quantum chemistry applications that motivate this work. One such example is a parallel nonlinear PDE modeling flow in a driven cavity; the primary CQoS issue for this problem is selecting and parameterizing preconditioned Newton-Krylov algorithms for TOPS solver components. As discussed in Sec. 3.2, we are collaborating with T. Dahlgren (LLNL) to incorporate performance-driven interface contracts, as a precursor to later introducing such capabilities for configuring nonlinear solvers for plasma edge simulations in the FACETS fusion SciDAC project. Initial testbed work was done by summer intern A. Berger at ANL (Sec. 6.4).

CQoS Infrastructure. Our approach to CQoS design includes capabilities for measurement and analysis infrastructure as well as control infrastructure for dynamic component replacement and domain-specific decision making [19, 21]. We designed and specified the initial CQoS API, and in August 2008 we made an

initial release of prototype computational quality of service (CQoS) database components[4], which support the management and query of historical performance data and application metadata for high-performance component applications. The database component interface design supports the management and analysis of performance and application metadata, so that the mapping of a problem to a solution that can potentially yield the best performance can be accomplished statically or at runtime. The UML diagram in Fig. 8 shows the main interfaces and some of their methods. We introduce two types of components for storing and querying CQoS performance data and metadata. The database component provides general-purpose interfaces for storing and accessing data in a physical database. The comparator interfaces compare and/or match properties of two problems under user-specified conditions. Summer intern B. Xie at ANL (Sec. 6.4) contributed to CQoS infrastructure development.

We teamed with the TAU group to extend the PerfExplorer performance analysis tool and to integrate PerfExplorer into general CQoS infrastructure to classify performance and meta-information and then suggest appropriate configurations for new problem instances. PerfExplorer, a framework for parallel performance data mining and knowledge discovery in the TAU performance system, was developed to facilitate analysis on large collections of experimental performance profiles [22]. Within the context of the CQoS initiative, PerfExplorer has been employing the classification capabilities in the Weka data-mining package to construct a runtime parameter recommendation system. After the performance data for a set of training runs has been stored in a performance database, PerfExplorer loads the data and build a classifier from it. In production application runs, the classifier is loaded into a CCA component. It is then used to obtain the best parameter settings by querying with the current values of the application-specific metadata that were used in the classification. We are testing these capabilities with CQoS testbed codes and applications in quantum chemistry and combustion.

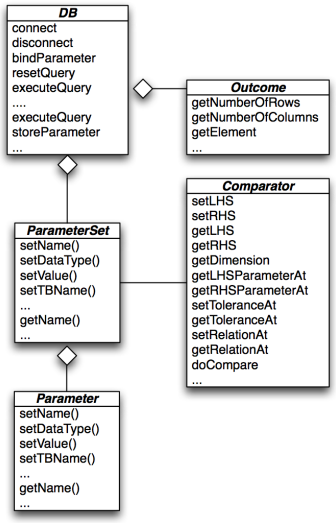

Figure 8: UML diagram of CQoS database component interfaces and methods.

CQoS in Quantum Chemistry. The QCSAP project (Sec. A.1.10) has adopted CCA components to promote interoperability among the GAMESS, MPQC, and NWChem quantum chemistry packages. Running these computations on diverse computing platforms requires specification of many options that range from the basic selection of methods, expansion basis, and convergence criteria to hardware configuration and low-level algorithmic parameters; typical educated guesses or trial and error can result in unexpectedly low performance. Motivated by the need for faster runtimes and increased productivity for chemists, we are teaming with QCSAP researchers to develop a flexible CQoS approach for quantum chemistry that uses a generic CQoS database component to create a training database with timing results and metadata for a range of calculations. The database then interacts with a chemistry CQoS component and other infrastructure to facilitate adaptive application composition for new calculations. Fig. 9 illustrates the interactions of the chemistry CQoS component with database, comparator, and analysis CQoS components [1, 18].

Our initial work involves three phases: collection of performance data in a training database, performance analysis, and adaptive application composition based on this information. The training database contains timing results for calculations spanning a range of molecular characteristics and hardware configurations. Once this database is populated on a target machine, users can request an appropriate configuration for a new molecule and calculation type from the chemistry CQoS component. The chemistry CQoS component uses the general CQoS database component interfaces to store and query performance and associated metadata, which in this case consists of the application's parallel configuration parameters. Furthermore, we defined comparator components that serve as filters when searching for appropriate parameter settings in

[4]See `http://wiki.mcs.anl.gov/cqos`.

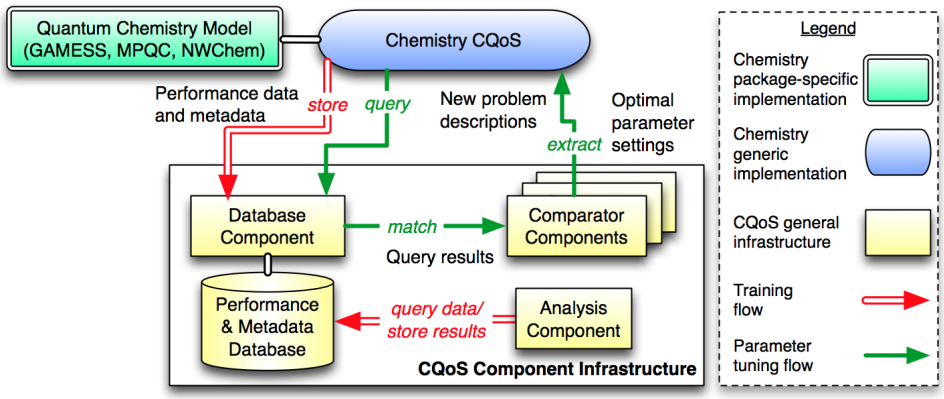

Figure 9: Components in a CQoS-enabled quantum chemistry application [1].

the database. An analysis component based on PerfExplorer provides offline performance analysis to identify sources of parallel inefficiency and determine appropriate parameters for a given configuration. Further details about our approach in using CQoS infrastructure in quantum chemistry are available [1, 18].

CQoS in Combustion. We are also collaborating with the CFRFS project (Sec. A.1.6) to develop a meta-partitioner to be used to partition and load-balance parallel simulations of reactive flows on block-structured, adaptively-refined meshes. The numerical method adopted by CFRFS, a Strang-split time-integration conducted on such a mesh, results in spatially-uneven computational loads due to (1) the high concentration of grid points in the vicinity of the flame and (2) the difficulty in integrating the fast chemical kinetics within the flame, leading to the slow convergence of nonlinear solves, within the context of implicit integration. The aim of the combustion CQoS effort is to develop and configure a load-balancer such that the AMR mesh is appropriately partitioned and to develop, over time, a partitioner that accounts for spatially-varying per-cell computational load.

CFRFS researchers have provided detailed input on priorities for CQoS infrastructure, tested the prototype CQoS database capabilities [21], and developed application-specific metrics and control laws. In particular, this work addressed the problem of appropriately configuring a partitioner, which required development of metrics to quantify the characteristics of an AMR mesh [86, 87]. CFRFS researchers also investigated the sensitivity of partition quality on various partitioner parameters [88, 89] and how to pair mesh characteristics with their optimal partitioner parameter values by running four separate AMR simulations and saving the trace, i.e., the mesh hierarchy, after every re-grid operation [90]. These traces were then subjected to a battery of roughly 1000 partitioning tests, with various partitioner parameter settings, to gauge their quality. This work resulted in a simple rule-based decision engine that was then tested on the same traces to determine whether it could indeed select the optimal parameter values given the mesh.

The metapartitioner currently identifies the top ten parameter sets that are expected to provide an optimal partitioning, based on the database of approximately 1000 tests, and proceeds to test each empirically, to identify the best one [90]. Such an approach has been adopted because the initial sparse, evenly-distributed sampling of the six-dimensional parameter space was insufficient to identify the optimal parameter set. We will perform a more sophisticated and denser sampling of the parameter space, which will invariably require a more sophisticated data collection and analysis approach than flat text files, the initial method employed. In this regard, CFRFS researchers have investigated how a componentized database may be used within CFRFS data collection infrastructure [21].

Application Impact and Future Work. We will continue to augment the CQoS infrastructure with additional capabilities, with emphasis on the needs of the SciDAC quantum chemistry and combustion sim-

ulations. In later years we plan also to work directly with the fusion and accelerator projects to use CQoS tools to select and configure TOPS (non)linear solvers. More specifically, future work includes enhancing the CQoS infrastructure to support sophisticated analysis for (re)configuring algorithmic parameters and component instances during runtime, developing application-specific performance models, and incorporating the CQoS training phase into empirical experiment design. We will continue to be in close collaboration with TAU group to investigate performance analysis and knowledge mining approaches that help make decisions for runtime adaptation. Because we are developing the base CQoS infrastructure to be generally useful to high-performance scientific applications, we expect these tools to prove useful in application areas beyond those introduced above when the need arises for dynamically selecting and parameterizing algorithms/software during runtime.

4 The CCA Environment

Coordinator: Tom Epperly, LLNL

The CCA Environment thrust addresses two important aspects of the TASCS project. The first is our fundamental goal of bringing the CCA environment from a prototype stage to a production level of maturity and quality. The second is enhancing and extending the capabilities of the CCA's foundational technologies in response to the needs of other elements of TASCS, our collaborators, and the CCA community at large.

The CCA's fundamental technologies include the CCA specification itself, CCA framework implementations, and the Babel middleware. Babel provides high-performance language interoperability (currently supporting C, C++, Fortran, Java, and Python), remote method invocation (RMI), and interface contract (Sec. 3.2) capabilities, among others. Additional tools and technologies developed to make the CCA more accessible and user-friendly also become become critical parts of the CCA infrastructure for most users.

Work on the CCA Environment is organized around three types of activities. The first, *Core Tool Support and Maintenance* (Sec. 4.1), is the task of keeping the essential tools of the CCA environment working on diverse and ever-changing computing platforms and responding to user support requests. The second activity, *Enhancements* (Sec. 4.2), focuses on extending the CCA environment, while the third activity, *Usability* (Sec. 4.3), improves the usability and productivity of the CCA environment, primarily for the developers of component software.

While much of this work is aimed at the general CCA user base, some of the developments are motivated by the needs of specific collaborations, including the ComPASS (Sec. A.1.5), CFRFS (Sec. A.1.6), FACETS (Sec. A.1.7), and GWACCAMOLE (Sec. A.1.8) scientific applications, as well as the HPC (Sec. 3.1) and CQoS (Sec. 3.3) Initiatives. Some of the work has been carried out jointly with the Contractor (Sec. A.2.2), SDM (Sec. A.1.12), and Distributed Components (Sec. A.2.5) projects.

Since the beginning of the TASCS project, this work has been discussed in a variety of papers, presentations, and posters [3, 8, 9, 24–43].

4.1 Core Tool Support and Maintenance

Coordinator: Ben Allan, SNL
Participants: ANL, LLNL, ORNL, SNL

Collaborators. The TASCS participants in support and maintenance are Benjamin Allan, Rob Armstrong (SNL); Boyana Norris (ANL); Wael Elwasif, Jim Kohl (ORNL); Tom Epperly, Tammy Dahlgren, and Gary Kumfert (LLNL). The initial users of the CCA tools on leadership-class platforms will be the ComPASS (Sec. A.1.5), CFRFS (Sec. A.1.6), FACETS (Sec. A.1.7), and GWACCAMOLE (Sec. A.1.8) projects.

Motivation. It is crucial to the success of TASCS and the CCA that the core tools are available and well supported on computer platforms of interest, ranging from development machines (laptops and desktops) to large-scale capacity and leadership-class capability systems. The basic work of the Support activity is threefold: providing support and maintenance of the CCA tool chain, primarily in response to user-submitted issues; porting to new platforms; and improving documentation.

The commonly used CCA tool chain consists of the following tools, with their primary developers or maintainers in parenthesis:

- The CCA specification (SNL)
- The Babel middleware compiler (LLNL)
- The Ccaffeine parallel runtime framework and its GUI (SNL)
- The Bocca component development environment (ANL, SNL, ORNL)

Helpdesk and Bug Tracking. Bug tracking and helpdesk support is provided for the entire tool chain through web- and email-accessible bug tracking tools. A centralized CCA helpdesk[5] enables users to get the

[5] `http://www.cca-forum.org/help/`

Table 6: Platforms supported by the TASCS project.

Platform	Primary Site(s)	Lead Maintainer
Cray XT/Catamount	ORNL	*deprecated*
Cray XT/Compute Node Linux	ORNL, NERSC	Jim Kohl (ORNL)
IBM Blue Gene/P	ANL	Boyana Norris (ANL)
Redhat Linux clusters	IU, PNNL	
Redhat/Fedora Linux	many	
Debian/Ubuntu Linux	many	
Mac OSX	many	

Table 7: Development tools supported by the TASCS project.

Tool	Version(s)
GCC compilers	4.3 (C, C++, Fortran)
IBM XL compilers	9.x (C, C++), 11.x (Fortran)
Pathscale compilers	3.1
Intel compilers	9, 10
Portland Group compilers	6.x, 7.x, 8.0
Sun Java	1.4–1.6
Python	2.3–2.6
OpenMPI compiler wrappers	all versions
MPICH2 compiler wrappers	all versions

assistance they need without knowing which team is responsible for the issue at hand, though tool-specific trackers are also available for more experienced users. Issue resolution is performed by all tool developers, including ANL, LLNL, ORNL, and SNL team members.

Tool Maintenance and Porting. Users of CCA technologies employ a wide variety of platforms and software development tools, which are constantly evolving. The nature of the capabilities provided by CCA tools makes them sensitive to many small details of the platforms and software environments in which they function. Consequently, the CCA tools require continual attention (at varying levels). Table 6 lists the variety of platforms to which the CCA tool chain has been ported and maintained since the start of the TASCS project. Because of the special challenges associated with high-end systems, specific TASCS staff members are assigned to maintain these platforms. Table 7 gives an idea of the development tools that we support.

Challenges Supporting Specific Platforms. The majority of the effort of porting and maintaining the CCA tools has gone into the leadership-class systems (Cray XT and IBM Blue Gene). These platforms pose particular challenges for complex software infrastructures like the CCA because they typically offer stripped-down operating system functionality on compute nodes and rely on cross-compilation environments that often have insufficient support for configuration and build tools commonly used in open source software (e.g., GNU Autotools and Python distutils). Further, since these systems tend to suffer relatively frequent outages, porting work on them tends to proceed significantly more slowly than for more conventional environments. These challenges are exacerbated by computing center policies: (1) disk space and inode quotas pose extreme difficulties in maintaining local versions of supporting tools and libraries that are needed to compensate for outdated or missing tools, and (2) queue structures and disk policies make regular testing

difficult.

Dynamic linking is a specific capability that is *not* typically available on leadership-class systems but is central to the CCA on "normal" platforms. To remedy this difficulty, B. Allan (SNL) has produced an initial version of a tool to generate from a set of components a statically linked CCA application and a specification of how they should be wired together. Once the required code generation has been extended to cover all Babel-supported languages, the tool will be ready for general use.

A recurring request from a variety of potential CCA users is support for Microsoft Windows (more specifically, a native Windows port rather than the use of a UNIX emulation environment like Cygwin). Recently, G. Kumfert and T. Epperly (LLNL) visited Microsoft to explore the potential routes to a Windows port and the effort involved. They determined that while a port is feasible, the effort required would be substantial – certainly beyond currently available resources. Given this fact, our hope is to find a collaborating project with a strong enough interest in using the CCA in a native Windows environment that they are willing to invest some of their own resources in a Windows port.

Improving Packaging and Deployment. The Babel team has made a number of changes to reduce dependencies and simplify the build process for the tool. These include the incorporation of the (no longer supported) Chasm Fortran interoperability library into the Babel package, switching XML processors, and switching to using libtool for linking.

In addition, the overall approach to packaging and deployment of the CCA tool chain has recently been changed. The new approach leverages the Contractor build system developed by James Amundson (Fermilab, see Sec. A.2.2), which is focused on large software installations that require the coordinated configuration, building, and installation of numerous individual packages. The CCA tool chain build system was initially prototyped by summer intern D. Taylor and his mentor, B. Norris (ANL, see Sec. 6.4). CCA-Tools-Contractor provides integration of the existing configuration and build process for all tool-chain parts, automating the download, configuration, and build of CCA software and most of its prerequisites using simple command-line and optional graphical interfaces. After some refinement and lengthy testing, the CCA-Tools-Contractor system recently became the production packaging for the CCA tool chain.

Testing. As we realized the level of vigilance required to keep the CCA tool chain functioning properly in the face of constant evolution of the supporting software environments, especially on leadership-class platforms, we shifted our effort from the originally planned CCA specification conformance testing and test harnesses for component developers (Sec. 4.3) to implementing automatic "nightly build" and regression testing for the tool chain. The Babel team (LLNL) and others test Babel routinely on a wide variety of platforms using the Gantlet test harness that was co-developed with Babel. As noted above, routine testing on leadership-class systems can be very challenging. J. Kohl (ORNL) has developed routine testing infrastructure for the Cray XT (though different site policies require different procedures for NERSC and ORNL). B. Norris (ANL) is addressing this need for the IBM Blue Gene/P, but that environment is even more challenging, and additional effort will be required to make such testing routine there. We are also working to extend such testing to the entire tool chain, using the CCA tutorial as a primary test case. The new Contractor-based distribution of the tool chain is expected to facilitate this work.

Documentation. Technical documentation efforts have focused on extending the CCA tutorial materials (Sec. 6.2) to describe and demonstrate additional features of the CCA environment. The tutorial now includes:

- a set of very simple examples for solving single variable ordinary differential equations;
- a complex, MPI-based mini-toolkit for solving partial differential equations on orthogonal grids; and
- components for implementing and conformance testing the specifications for Event, MPI, BuilderService, and ServiceRegistry ports.

4.2 Enhancements

Coordinator: Tom Epperly, LLNL
Participants: BU, LLNL, ORNL, SNL, VSU

Collaborators. TASCS participants in this work include Ken Chiu, Madhu Govindaraju, Michael Head, Dai-Hee Kim, Xiang Gao, Ying Zhang (BU); Gary Kumfert, Tom Epperly (LLNL); Ben Allan (SNL); Steve Parker (University of Utah (UU)); and Kosta Damevski (UU/VSU). The TASCS HPC (Sec. 3.1) and CQoS (Sec. 3.3) Initiatives, as well as the ComPASS (Sec. A.1.5) and FACETS (Sec. A.1.7) applications, have provided specific motivations for some these activities. Some of the work has been performed jointly with the SDM (Sec. A.1.12) and Distributed Components (Sec. A.2.5) projects.

Motivation. The CCA environment must evolve to meet the needs of the user community. This work includes extending and refining the CCA specification itself, as well as the core tools associated with it.

The Common Component Architecture Review Board (CCARB). In the interest of ensuring a more robust specification, TASCS participant G. Kumfert (LLNL) proposed to the CCA Forum a new multi-step standardization process to be overseen by a Common Component Architecture Review Board, which represents both developers and users of the CCA. The Forum approved the proposal, and established the CCARB with Kumfert as its first chair, and TASCS participants B. Allan (SNL) and L. McInnes (ANL) among its members. On Kumfert's departure from LLNL (see Sec. 7), Allan was appointed chair and T. Epperly (LLNL) was added to the Board.

Extending the CCA Specification. A number of new services are making their way through the new standardization process:

- A publish/subscribe **Event Service** has a wide variety of uses within the CCA. The Event Service was originally motivated by the desire to support more general and flexible interactions between Graphical User Interface (GUI) frontends and CCA frameworks and applications. The Event Service has been employed to distribute and coordinate work in heterogeneous computing environments (Sec. 3.1.2), and we anticipate using it to make information about system faults available to CCA applications (Sec.3.1.3) and to support CQoS work (Sec. 3.3).

  Two implementations of the proposed Event Service specification have been developed: one by K. Damevski (UU/VSU), based on the event service built into the SciJump framework, and the other by the PNNL team in conjunction with their work on supporting heterogeneous computing environments (Sec. 3.1.2). The PNNL event service is notable in that it is a fully distributed implementation, using the Aggregate Remote Memory Copy Interface (ARMCI) for communications between parallel processes [8, 9].

  K. Damevski (VSU) chairs the CCARB's Event Service Working Group, which is responsible for moving the event service specification through the standards process. Recently, Damevski and G. Kumfert (LLNL) collected the existing event service implementations and began evaluating them for portability across CCA frameworks and functionality. Key stakeholders, including the CQoS team, are also reviewing the draft specification and implementations for functionality and performance as part of the standardization process.

- While the CCA specification itself is agnostic with respect to the parallel programming model used by components, the vast majority of parallel CCA components use MPI for communications. The **MPI Service** solves the problem of who is responsible for calling `MPI_Init` and supplies communicators to components that request them.

  B. Allan (SNL) chairs the CCARB's MPI Service Working Group, which has evaluated the common use cases and produced a draft specification. In addition, the group has two implementations of the

MPI Service specification as required by the CCARB's approval process. The draft is still under review but is expected to become part of the CCA specification this fiscal year.

A command line service, described in our proposal, has been made redundant by features of the MPI-2 standard, which are now widely enough implemented that we can rely upon them.

The next step beyond building independent individual frameworks is making them interoperate. Component applications should be able to transparently span multiple disjoint component frameworks with low overhead as compared to the same applications running within a single framework. Interoperable frameworks enable applications to take advantage of more resources, and to better match constituent parts to the underlying resources that best support them. The CCA specification does not prescribe a wire format for inter-component calls in distributed frameworks, thereby promoting considerable flexibility and customization for the framework developer. Anticipating a more intensive effort planned for the coming years of the project, the BU team has completed initial work in designing a specification for interoperability between CCA components of different frameworks. This work outlines five underlying component framework interoperability requirements, and three general approaches to addressing them [38].

RMI Enhancements. The BU team has focused thus far on the development of high-performance distributed middleware, a C++-based distributed framework, and multi-protocol support for Babel. This effort lays the groundwork for interoperability between distributed computing component frameworks and MPI-based parallel computing frameworks. Distributed middleware uses RMI to achieve parallel computing instead of MPI-based parallelism. The BU team extended XCAT, a C++-based distributed framework, with support for Babel to generate wrappers for distributed XCAT components. Furthermore, they added multi-protocol communication support to BabelRMI through the Proteus library [24]. The BabelRMI Proteus module is currently being designed to work with the SOAP communication protocol, which is widely used with Web services based applications. The BU team's work in these areas opens up new possibilities for interoperability among CCA components and Web services based applications.

The BU team presented several papers describing their recent work to improve XML parsing [25–27,39]. This work important for distributed CCA computing because of the extensive use of XML-based protocols for distributed services. In this work, the team studied how popular parallelization techniques used in SMP models can be applied and adapted for multi-core processors.

Tech-X researchers have implemented and characterized BabelRMI over the CORBA IIOP transport layer, showing how standards-based distributed communication protocols can be integrated into Babel with little performance penalty [24].

UU researchers Damevski, Zhang, and Parker have been exploring an approach to parallel remote method invocation in the CCA context [42].

Language Interoperability Enhancements. The LLNL Babel team worked with the Distributed Components Small Business Innovation Research (SBIR) project at Tech-X (Sec. A.2.5) to implement multi-language struct(ure) support in Babel, similar to the capabilities of native C/C++. With the advent of derived types in Fortran, structs are becoming common in scientific Application Programming Interface (API)'s to package related data in a single data structure. This work addresses the need for structs in multi-language interfaces. LLNL provided the overall design and the implementation of the C, C++, Java, and Python bindings, while the Tech-X SBIR project provided the Fortran2003 implementation. The goal is to be able to pass structs between C, C++, and Fortran 2003 without requiring any copies or function calls to access data members. This capability was described in a presentation and paper [28] and appeared in the Babel 1.4.0 release in October 2008.

One challenge we have encountered with the F2003 struct support in particular is that the Fortran compilers currently supported on Dept. of Energy (DOE)'s leadership-class systems (Portland Group and Pathscale) do not sufficiently support the "bind(c)" feature of the newest Fortran standard to allow Babel's F2003

struct binding to work. As struct support is very important to two of our science collaborators (ComPASS and FACETS), this circumstance is an obstacle to their use of Babel-based applications on such machines. We are investigating workarounds, as well as trying to get implementation time lines from the compiler vendors.

Summer intern M. Porche worked under the guidance of T. Epperly (LLNL) on a prototype of *Batooki*, a basic Babel toolkit of fundamental data structures. Most languages provide a basic library of common container classes and related algorithms. There is a similar need among Babel programmers, and Batooki is our work to fill that need. Porche demonstrated the feasibility of such a toolkit, but some additional work is required before Batooki can be released as part of Babel.

Extending the CCA Framework and Integration with Other Component Systems. The CCA framework is an important platform through which new capabilities can be provides for application software developers to use. In this context, K. Damevski and others at UU have been exploring the limits of scalability of the CCA framework (up to millions of distributed framework nodes) in a fault tolerant fashion [40, 41].

The CCA is not the only framework(-like) environment used in scientific computing. Integration of software across disparate frameworks becomes increasingly important as many computational scientists begin to undertake new coupled multiscale and multiphysics simulations (often based on existing codes, which might utilize their own framework environments), and to develop "end-to-end" solutions to automate more of the simulation and analysis workflows.

The BU team developed a novel decomposition scheme to incorporate CCA components in the Ccaffeine framework into the Kepler scientific workflow toolkit [3], developed by the Scientific Data Management (SDM) center (Sec. A.1.12). Scientific workflows are concerned with high-level orchestration of a time-decomposed simulation, while components are mainly used at a lower level to enable software modularity and reuse at a small performance cost. Combining these technologies in a seamless way leverages the benefits of both decomposition paradigms, resulting in more flexibility for scientific application design. This initial work showed how the benefits of a workflow approach can be combined with a components-based system to make a hybrid system with the benefits of both technologies. In addition to work with Kepler, the BU team developed a standards compliant approach to C++ reflection to support coupling in problem solving environment [29].

In collaboration with the SDM center, the UU/VSU team extended BU's earlier work with Kepler by developing a system with fine grained communication between the CCA framework and Kepler [30]. Where BU's approach required translation of Babel's SIDL into Web Service Definition Language (WSDL), the UU/VSU team simplified the design and improved performance by using "native" BabelRMI. The UU/VSU researchers designed three Kepler actors that are necessary for an initial hybrid application: one actor that is able to initialize the components, one actor that can control and begin the execution of the components, and one that is able to communicate status messages and results. They tested their design by executing and controlling a CCA tutorial application from Kepler. Having demonstrated interoperability between Kepler and the CCA, we are now looking for a suitable application to drive turning this into a production capability.

4.3 Usability

Coordinator: Matthew Sottile, UO
Participants: ANL, LLNL, ORNL, SNL, UO

Collaborators. The work described here has been performed by Boyana Norris (ANL); Wael Elwasif (ORNL); Ben Allan, Rob Armstrong (SNL); Geoff Hulette and Matt Sottile (UO). Note that in the original proposal, ANL did not plan to participate in the Usability activities, but that changed with the retooling of the effort to focus on Bocca and OnRamp (see below).

Motivation. The Usability activity is responsible for addressing critical issues that exist at the point where users of CCA interact with both the tools and the programming model. Usability of CCA tools involves

managing the complexity of the tool chain (largely due to the unavoidable complexity of multilingual code environments) and developing tools and approaches to make CCA users more productive in creating and using component software.

Retooling Usability. The usability effort initially focused on a concept known as "CCA-lite." During the SciDAC-1 CCTTSS project, it was already recognized that there was a significant learning curve for the CCA due to the complexity induced by SIDL and tools supporting full language interoperability. A "lite" version of the CCA specification was intended to reduce this complexity by limiting language interoperability support and eliminating support for dynamic loading of components from shared libraries. The intent of this second, pruned-down specification was to reduce the barrier to adoption with the intent that users of CCA-lite would eventually migrate to the full CCA specification. This approach had two drawbacks: the need to maintain a second standard and complete tool chain to support the lite approach and assist with the transition to full CCA, and potential resistance of CCA-lite users to move to the full CCA specification, thereby defeating the intent of the "lite" specification as an intermediate step.

With the advent of Bocca (Sec. 4.3.1), creating CCA components conformant to the original specification became simple enough to make the CCA lite option much less attractive. Moreover, we realized that Bocca could be leveraged to provide an even higher degree of automation for the migration of large legacy code bases to the CCA, namely, the OnRamp utilities (Sec.4.3.2).

Given the significance and impact of these two developments, as well as other priorities within the project (particularly on improving the portability and robustness of the tool chain), we have deferred additional usability tasks planned for the early years of the project, namely development of a component test harness and debugging support.

4.3.1 Bocca: Automated CCA Interfaces and Components

By the beginning of SciDAC-2 it was clear that while component-based language-interoperable software was attractive, the glue code and attendant build system necessary to support component oriented designs was overwhelming to most users. Bocca was conceived as a command-line tool that would create the skeletal structure for a component and its interfaces, including the entire build system necessary to compile the component as well as the Babel bindings and references required by the CCA specification. Bocca makes the creation of buildable components and their attendant interfaces as easy as giving them names (see Fig. 10). Because the build system comes with Bocca, these components are ready to load into a CCA framework. Of course, the components need implementation code before they will do something useful. The idea is that all of the glue code that makes a component is provided. Moreover, markers, in the form of comments, show application programmers precisely where to insert their code so that they can concentrate immediately and solely on the task of creating scientific software [34–36].

While Bocca provides functionality typically available in graphical software development environments, it cannot rely on having the project state continuously available in memory. Bocca's design is thus based on a lightweight graph-based representation of SIDL-based entities and their relationships (Fig. 11 contains an example graph for a simple Bocca project). Each command must load the entire project state, and a number of commands modify it and must store it, making efficient serialization and deserialization crucial. Bocca supports project creation and ongoing management, including various refactoring tasks (e.g., rename, remove, import, copy) on SIDL-based project entities (interfaces, ports, classes, components, enums).

Today Bocca provides numerous ways of importing implementation code and SIDL interfaces from external sources. Components can be created from SIDL/Babel code directly, while interfaces can be imported from existing SIDL files, and both can be imported from other CCA components.

As evidence of Bocca's remarkable success, the tool has virtually taken over the CCA world. Today, anyone contemplating making a CCA component will almost certainly use Bocca. Nearly every component in the CCA toolkit now uses Bocca for its maintenance and build system. The CCA tutorial has been

Shell command listing

```
$ bocca create project myProject
$ cd myProject
$ bocca create port myPort
$ bocca create component --uses=myPort@aPort --go=myGoPort Component1
$ bocca create component --provides=myPort@aPort  Component2
```

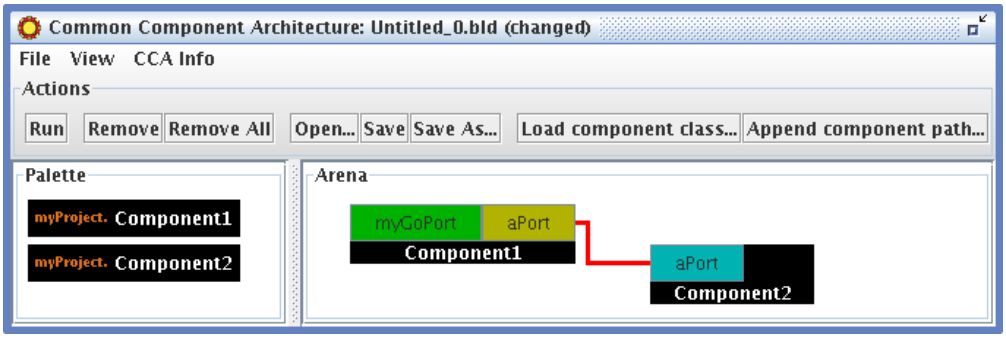

Figure 10: Bocca example creating simple, albeit empty, components. The shell commands at the top create the components in the Ccaffeine GUI. In this case, Bocca creates empty shell components that are nonetheless fully CCA-compliant. Subsequently, the user would need to add code to implement the desired functionality for the components.

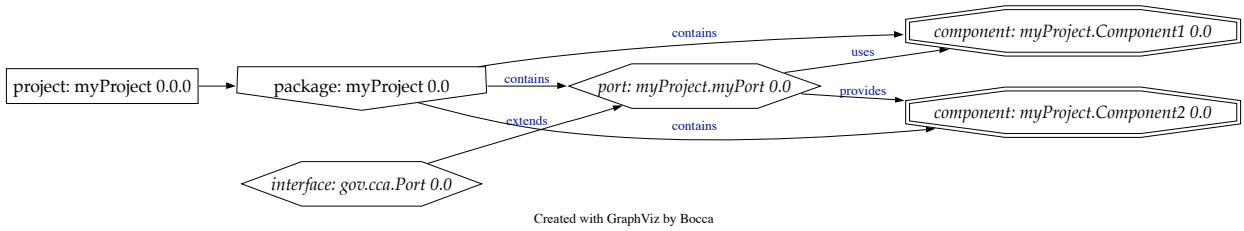

Figure 11: Graph representing the project state for a simple Bocca project.

rewritten to teach the Bocca style of component development exclusively. Most importantly, Bocca has greatly reduced the barrier to using component-based methods for scientific computing.

Bocca is itself readily extensible to accommodate different build systems and other capabilities. It can also serve as the basis on which to build higher levels of automation, such as the generation of component proxies (used by the TAU performance monitoring system, Sec. A.2.12), or the semi-automatic wrapping of legacy code into CCA components, as in OnRamp.

4.3.2 OnRamp

Significant effort in the Usability activity during this period has gone into OnRamp – the next step beyond Bocca. While Bocca autogenerates CCA componentry, creating all of the glue code necessary for participating in the CCA world, OnRamp provides a mechanism for converting legacy (i.e., non-component) code into a fully functional componentized version of the same application. Using developer-supplied directives that appear as comments in the original source, OnRamp constructs the desired CCA interfaces and binds the original code as the implementations of these interfaces. This process can be done piecewise to extract a particular fragment of an existing simulation, or the entire application can be converted to a component framework that preserves the same behavior as the original (Fig. 12).

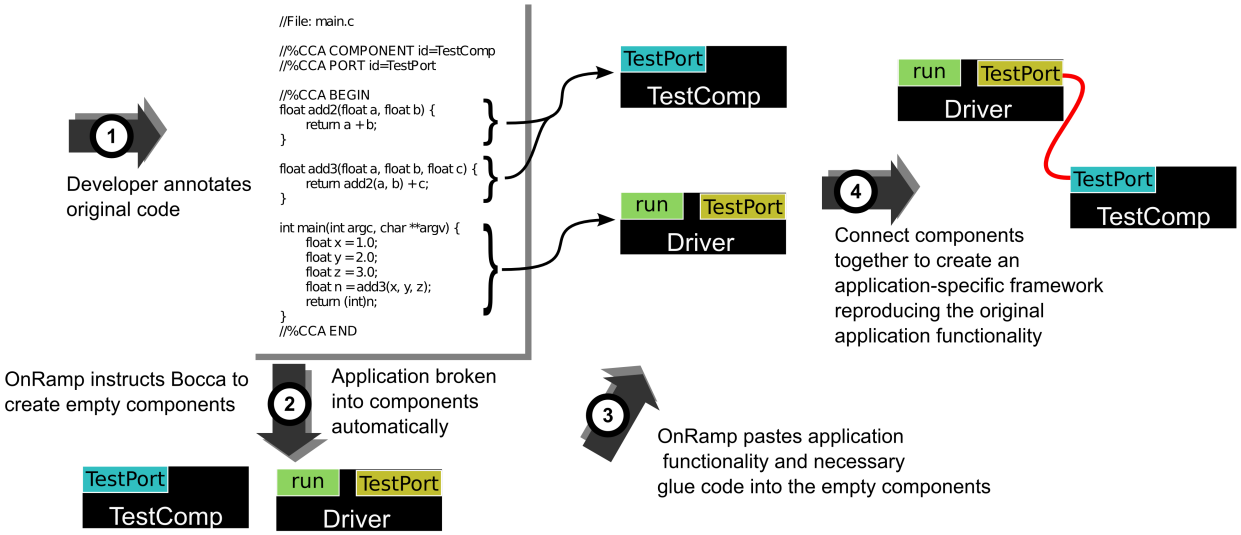

Figure 12: OnRamp processing an existing code into CCA components.

OnRamp is based on several components:

- commercial-grade compiler front-ends as provided by the UO Program Database Toolkit package for C, C++, and Fortran (77,90,95);
- the OnRamp code itself that uses the database representation generated by PDT to analyze the OnRamp annotations and static interface of the user code; and
- a set of OnRamp code generation routines for populating SIDL and Babel implementation files based on the results of the static analysis phase.

OnRamp needs to statically analyze both input code *and* annotations for two reasons. First, semantic guidance must be provided beyond that which can be inferred from the language alone. The type systems of common languages (such as C) do not contain sufficient information to properly describe complex data structures or even multidimensional arrays commonly needed by scientific simulations. Annotations are necessary to inform analysis tools of these types. Second, OnRamp must guide decomposition of code into meaningful and useful components. This work requires human intervention to decide how subroutines and types are related to an interface (i.e., Port), such as those that comprise a solver or mesh package.

OnRamp defines a concise annotation language to guide tools in determining the structure and semantics of the code beyond that which can be inferred from static syntax analysis alone. An example annotated subroutine is shown below.

```
/* %CCA PORT  id=VectorMath
        ARRAY elements=x length=len
        ARRAY elements=y length=len */
double dot(double *x, double *y, int len);
```

OnRamp performs code and annotation analysis sufficient to understand implementation dependencies and how implementations are decomposed into components. OnRamp relies heavily on Bocca to realize the components, glue code, and build system necessary to create components or a componentized application.

The OnRamp project is led by UO with contributions from ANL and SNL. The code repository and downloadable distribution for OnRamp is hosted at the SciDAC Outreach website[6]. We have produced a number of publications on OnRamp and related code wrapping efforts [31–33].

[6] `http://outreach.scidac.gov`

5 The CCA Toolkit

Coordinator: Rob Armstrong, SNL

Participants: ANL, IU, LLNL, ORNL, PNNL, SNL, Tech-X, UMD, VSU

Collaborators. Development of the CCA toolkit is is a broad collaboration among all TASCS participants, mathematicians in the ITAPS (Sec. A.1.2) and TOPS (Sec. A.1.13) math CETs, Masha Sosonkina at Ames (Sec. A.2.11), and the TAU group at UO (Sec. A.2.12).

Motivation. The aim of the CCA Toolkit is to create a repository of high-quality, high-performance components that enable scientific developers to perform rapid prototyping, numerical proofs of concept, and experimentation with readily available and easy-to-use components. These components also provide valuable plug-and-play capabilities to existing HPC applications.

Accomplishments. This work consists of several different elements: defining conventions and standards for components in the CCA Toolkit, so that their installations can coexist and be used together straightforwardly; the collection and publication of contributed components; and the development of new components for the toolkit. R. Armstrong (SNL) has led the first two elements of this effort, while a variety of groups, both within and outside TASCS, are developing and contributing components.

After evolving through several specifications for toolkit components, the development of Bocca (Sec. 4.3) has provided a very logical and straightforward set of conventions to follow. Toolkit contributors are not required to use Bocca, but should provide the equivalent `make` targets and installation procedures. As a reflection of how revolutionary Bocca has been to CCA component development, most of the Toolkit components now use it directly in their build systems and component generation. As a practical matter, R. Armstrong is providing direct assistance to external contributors in adapting their components to the Toolkit's conventions.

The CCA Component Collection. The CCA Toolkit web page, currently located on the developmental web site, `http://www.cca-forum.org/dev/software/components/`, to be made public soon, lists the currently available components; the SPARSKIT-CCA and TAU components are funded externally to TASCS.

- **Global Array Component** (PNNL) is a componentized version of the library.
- **InterComm** (UMD) is a framework that facilitates interoperation between two parallel codes. The InterComm component does the same for CCA components.
- **Parallel Cartesian Grid Component** (SNL) presents a fully parallel rectilinear grid and associated tools for solving PDEs on it.
- **TOPS Solver Component**s (ANL, Sec. A.1.13) provide high-level access to parallel (non)linear algebraic solvers for nonlinear PDEs; these components build upon PETSc, hypre, and SuperLU.
- **Linear Solver Interface (LISI)** (IU) [44–47] provides a lower-level linear solver interface, building on including PETSc, Trilinos, and SuperLU.
- **SPARSKIT-CCA** (Ames Lab, Sec. A.2.11) [44, 45] is a componentized version of the library.
- **TAO Numerical Optimization Components** (ANL) provide a component interface to the Toolkit for Advanced Optimization, which is part of TOPS (Sec. A.1.13).
- **TAU Components** (UO, Sec. A.2.12) for performance monitoring and analysis provide a componentized version of Tuning and Analysis Utilities.

In addition to this work, a number of groups within the project are in the process of developing components for the Toolkit.

- J. Billings at ORNL implemented a CCA-compliant version of the ITAPS mesh interface (Sec. A.1.2), which has been tested with a component implementation of the ITAPS test program. Componentization costs versus the native C implementation of the test program were measured to be 3.4%. Future testing will be in an application setting.
- N. Lemaster and D. Rouson at SNL are extending the Parallel Cartesian Grid component from 2D to 3D. In addition, a meta-partitioner is being crafted to allow any logical partitioning scheme for the grid to be employed as a component itself. The component will do the heavy lifting so that its processor/data decomposition matches the scheme.
- J. Larson and colleagues at ANL have been working to generalize their experience with coupling components and scientific applications such as FACETS (Sec. A.1.7) [48–52].
- The UMD team completed componentization of the InterComm data coupling package, which now includes a number of working examples that employ the Ccaffeine framework. They are also working with the CCA MCMD working group and framework developers on an approach for providing InterComm's full functionality within the CCA environment [53].
- K. Damevski (VSU) is adapting the SciJump framework's GUI into a general CCA GUI component, which will eventually replace the aging Ccafe-GUI that was developed specifically for the Ccaffeine framework. This work requires more intimate coupling to a CCA framework than the current CCA specification allows. The CCA Event Service (Sec. 4.2), also being developed by Damevski, will provide the fine-grain feedback necessary to make CCA frameworks interactive enough for a user-driven GUI.

Application Impact and Future Work. Toolkit development is always an outgrowth of other collaborations, including TASCS research initiatives (Sec. 3) and educational efforts in the CCA tutorial (Sec. 6). Two important components are expected in the near term: the CCA GUI component, replacing the antiquated and buggy Ccaffeine GUI (K. Damevski, VSU) and the ITAPS unstructured meshing tools (J. Billings, ORNL). This work will impact many CCA Toolkit users by providing a much-needed first-class user interface and an important mesh type that is not currently available.

An important complement to the development of actual *components* for the Toolkit is the definition of common interfaces that can provide interoperability across multiple libraries that offer similar capabilities. We encourage other SciDAC centers and applications teams to define common interfaces to their respective domains of expertise. CCA has an ongoing effort with ITAPS, TOPS, and QCSAP to aid in translating their advances into SIDL interfaces and CCA components.

6 User and Application Outreach and Support

Coordinator: David Bernholdt, ORNL

Participants: ANL, IU, LLNL, ORNL, PNNL, SNL, Tech-X

Because the long-range goal of the CCA effort is to fundamentally change the way high-performance scientific software is developed and used, outreach constitutes an important part of TASCS. This outreach takes several forms, ranging from work with specific application groups, to broader activities aimed at assisting and growing the number of knowledgeable CCA users, to general outreach to the computational science community, including publications, presentations, and educational activities.

6.1 Application Support

Appendix A describes more than two dozen external projects with which we have actively collaborated during this period. In each of these collaborations there has been a bidirectional intellectual exchange between the projects related to the use of component concepts and/or CCA tools. This is not merely a list of projects with which we have discussed possible collaborations, or projects that have adopted the CCA without interacting with us.

As mentioned in Sec. 1, these collaborations are very important to the way TASCS operates because they provide a testing grounds for our tools and ideas; they also provide feedback, which helps us to refine our work and often spurs new research ideas. The intensity and level of effort devoted to each collaboration varies from project to project. In many cases, TASCS team members are also direct participants in the collaborating project and provide a focal point for the collaborative work.

6.2 User Outreach and Support

Beyond direct collaborations with specific projects, we also make a significant effort to support a broader community in learning and using component concepts and CCA tools. Our primary vehicles for this work are tutorials and "coding camps." Coding camps are focused working sessions, typically several days to a week in duration, in which a group of CCA users and CCA developers meet in one room, akin to what some projects call "bring-your-own-code workshops." By co-locating users who have questions with experts capable of helping to solve problems, development can be greatly accelerated compared to typical email- or phone-based interactions.

Development and presentation of tutorial materials is a broad effort among numerous TASCS team members at ANL, LLNL, ORNL, SNL, and Tech-X, and our collaborators Tony Drummond (Lawrence Berkeley National Laboratory (LBNL)), and Alan Morris and Sameer Shende (UO). Tutorials presented during this period include:

- Ninth Workshop on the DOE Advanced Computational Software (ACTS) Collection, Oakland, California (2008)
- Simula Research Laboratory, Lysaker, Norway (2008)
- PARA 2008, Trondheim, Norway
- Supercomputing 2007, Reno, Nevada (two independent tutorials for Babel and the CCA)
- Eighth Workshop on the DOE Advanced Computational Software (ACTS) Collection, Berkeley, California (2007)
- SciDAC 2007, Boston, Massachusetts
- Seventh Workshop on the DOE Advanced Computational Software Collection, Berkeley, California (2006)

During this period, the tutorial was also significantly revamped, under the leadership of B. Allan and R. Armstrong (SNL), to incorporate Bocca as the primary development tool used by students and to add additional material based on a more complex PDE problem. Recent experience shows that both improvements are providing significant benefits in terms of the amount of progress students are able to make during

a hands-on session and the perceived relevance of the examples. We have also shifted the balance of the tutorial from an equal measure of lecture-style presentations and self-paced hands-on computer exercises to one with less lecture time and more hands-on time.

The project held coding camps in January 2008 and April 2007 as working sessions for contributors to the CCA Toolkit (Sec. 5). The 2007 coding camp also served as the initial "friendly user" release of the Bocca tool (Sec. 4.3).

6.3 Community Outreach

Outreach to the broader community includes a variety of scholarly activities as well as activities that support the CCA Forum.

Publications and Presentations. The TASCS team is actively involved in disseminating the results of our research and educating the community about CBSE. Appendix B provides a comprehensive list of the project's 70 publications and presentations to date. Citations for most of these appear in the body of this report, in conjunction with the description of the work, or in Appendix A in conjunction with the particular collaborations that gave rise to them. We have also produced a number of publications that provide more general overviews of the CCA and TASCS, often as outreach to specific scientific communities [6, 54–60], and some new ideas [61, 62].

Scientific Meetings. TASCS members are also active in the organization of scientific meetings to promote the exchange of ideas and experience with CBSD.

A focal point of these activities has been a series of workshops, which has been organized annually since 2005 at the initiative of R. Armstrong (SNL) and D. Bernholdt (ORNL). The workshops were originally known as CompFrame, and were a continuation of workshops organized by Aad van der Steen (U. Utrecht) several years earlier. Since 2006, the workshop has been co-organized with the European Union (EU) HPC Grid Programming Environments and Components (HPC-GECO) project, initially as HPC-GECO/CompFrame, and starting in 2008 as the Workshop on Component-Based High-Performance Computing (CBHPC). After the 2007 workshop, a Steering Committee was established to give the workshop series greater continuity independent of the TASCS and HPC-GECO projects. As past meeting chairs, Bernholdt and Armstrong serve terms on the initial Steering Committee. TASCS also hosts the CompFrame web site[7] on the CCA Forum collaboration servers (Sec. 6.5).

Other meeting organization activities by TASCS members include:

- At SIAM CSE 2009, Damian Rousson (SNL) organised an 8-speaker minisymposium titled "Multiphysics Modeling: Frameworks and Applications."
- David Bernholdt (ORNL) was co-chair of HPC-GECO/CompFrame 2007, with 16 speakers.
- For Para08, Ben Allan (SNL) co-organized a minisymposium on "HPC Software: Tools, Libraries and Frameworks" with ten speakers.

6.4 Education

Through its university participants and national lab internship programs, the TASCS project is actively working to instill in the next generation workforce an appreciation of and relevant experience with component-based high-performance computing.

CCA in the Classroom. All TASCS universities incorporate component-related material in one or more of the courses in their computer science curricula.

[7] `http://compframe.org`

Degrees Awarded. A number of students have obtained M.S. and Ph.D. degrees on topics related to TASCS:

- Tammy Dahlgren (LLNL), Ph.D., 2008, University of California Davis [23]
- Xiang Gao, M.S., 2008, Binghamton University
- Shang-chieh Wu, Ph.D., 2008, University of Maryland [63]
- Ashwin Swaminathan, M.S., 2007, University of Utah [64]
- Kostadin Damevski, Ph.D., 2006, University of Utah [65]

Summer Interns. The project has hosted a number of interns. Their technical contributions are covered in the sections referenced in the column "Description."

Year	Host Site	Name	School	Mentor	Description
2006	ANL	Andrea Berger	Clarion U.	Lois McInnes	Sec. 3.3
2006	ANL	Daniel Taylor	Edinboro U.	Boyana Norris	Sec. 4.1
2008	ANL	Bing Xie	Illinois Inst. of Tech.	Boyana Norris	Sec. 3.3
2008	LLNL	Monica Porche	Central State U.	Tom Epperly	Sec. 4.2
2008	ORNL	Henok Mikre	U. Tennessee	David Bernholdt	Sec. 6.3

6.5 Supporting the CCA Forum

The CCA Forum continues to exist as a community-based organization, separate from any particular research project such as TASCS. We consider this structure to be very important, and in fact as we were formulating the TASCS proposal, we simultaneously took steps within the Forum to formalize a management structure that previously had not been clearly defined. However, as the largest single CCA-related project, TASCS plays a very strong role in supporting the organization.

CCA Forum Meetings. Historically, TASCS (or its predecessor CCTTSS) participants have arranged and hosted approximately three quarters of all CCA Forum meetings. Since July 2006, ten of the last eleven quarterly Forum meetings have been hosted by TASCS sites. Financially, these meetings are supported by a combination of registration fees, intended to cover most of the costs, and project funds to make up the difference. TASCS leverages these meetings as, effectively, project meetings, as well as opportunities to engage frequently with collaborators and the larger CCA community. A standard agenda item at these meetings is a brief update on the recent activities of the TASCS project, and many of the contributed presentations are related to TASCS research. TASCS also supports the costs of providing remote access (teleconference and web sharing for presentations) to Forum meetings.

CCA Forum Collaboration Servers. TASCS provides a variety of collaboration services for the Forum, including:

- Mailing lists
- Web sites and wikis
- Code repositories (CVS and Subversion)
- Login access to a common computing environment

At the beginning of the TASCS project, we began migrating these services from old hardware provided by SNL and hosted by LBNL to newly-purchased servers at ORNL, where it was easier to cover IT department support costs from project funds. The migration has taken much longer than anticipated, in large part because the level of support we have been able to get from the IT department is much lower than we originally anticipated. This situation is largely due to ORNL IT staff being diverted to address the avalanche of cyber security requirements being imposed by DOE. At this point, code repositories still reside on the old LBNL servers (we expect to finish migrating them after this review is over), and all other services are hosted at ORNL.

While we had originally thought to establish our own SourceForge-like software development site focused on CCA-related projects, the challenges described above have caused us to rethink this plan. Recent discussions among leading developers of CCA software have suggested that SourceForge itself is considered sufficiently reliable, and with good information and cross-linking on the CCA Forum web site, there is little to be gained from a "private garden" for CCA software development, especially given the challenges we would face in deploying and supporting it without more IT assistance. Consequently, we recently began migrating several of our core tools to the newly-established `http://cca-forum.sourceforge.net` SourceForge project.

As an alternative to this deliverable, we have focused on developing an electronic document repository for CCA-related publications, presentations, and other materials, in order to address a long-standing complaint that the CCA effort does not do an adequate job of making its scholarly work widely accessible. The repository, `http://eprints.cca-forum.org`, is based on the freely available GNU EPrints software package (`http://www.eprints.org`). For us, the primary purpose of the document repository is to provide a comprehensive public bibliography of CCA-related publications and presentations. The EPrints package allows items to refer to external URLs (i.e., the journal's web site) or archive the document locally, as appropriate. The package is powerful and extremely customizable. The initial installation and customization of the EPrints package was done in the summer of 2008 by University of Tennessee undergraduate Henok Mikre during an internship with D. Bernholdt (ORNL). At this point, the repository has been populated with the bibliographic information for all publications of the CCTTSS and TASCS projects, and we are in the process of completing those entries to include links to or copies of the actual papers wherever possible. We are also collecting presentations, posters, and other materials, starting with TASCS. Once the repository contains the complete output of the two SciDAC main projects, we will begin a campaign to collect CCA-related publications from other projects as well.

7 Non-Technical Matters

During this period, the TASCS project has experienced a number of personnel changes that have significantly impacted the institutional make-up of the project as well as the composition of the leadership team. Fortunately, however, the new site leads are also long-standing participants in the CCA effort, so the disruptions have been minimal, and mostly associated with funding gaps, as discussed below. Within each institution, of course, there have been changes over time, primarily as students and postdocs come and go. Students having thesis or dissertation work related to TASCS, along with short-term student interns, are discussed in Sec. 6.3.

Matt Sottile at UO Succeeds Craig Rasmussen at Los Alamos National Laboratory (LANL). In April 2007, LANL co-PI Craig Rasmussen accepted additional management responsibilities within the lab and transferred his TASCS leadership responsibilities to Matt Sottile, who has been involved in the CCA effort since 2001. In September 2007, Sottile started a faculty position at UO, and with the concurrence of our DOE program managers, the LANL TASCS effort moved with him. Although Sottile was able to keep the Usability effort (Sec. 4.3) moving during the transition, the pace of work was necessarily slowed until the funding caught up with him in June 2008 and he was able to hire students to work on the project.

Kosta Damevski at VSU Succeeds Steve Parker at UU. In May 2008, Steve Parker left UU for a position at NVidia. At nearly the same time, Kosta Damevski, who had worked with Parker on TASCS, first as a Ph.D. student and then as a postdoc, accepted a faculty position at VSU. In consultation with our DOE program managers, we decided to replace Parker at UU with Damevski at VSU in TASCS. The paperwork to move the funds was submitted as quickly as possible after Damevski formally started his position in the Fall of 2008 and is still working its way through DOE. However, Damevski has managed to temporarily tap other funding sources for travel and has been able to be quite active in the project. Because his research interests are somewhat different from those of his mentor, we have made some minor adjustments to our future plans in response to this change. Those differences are discussed in Sec. 4.2 and Sec. 5.

Tom Epperly Succeeds Gary Kumfert as LLNL Co-PI. In October 2008, Gary Kumfert left LLNL for a position at Conviva, a small technology company; he was succeeded by Tom Epperly. Kumfert and Epperly have worked together on the CCA effort since March 2000, effectively jointly leading LLNL's participation since April 2003, so the transition within the TASCS leadership team went smoothly. However the ripple effects of Kumfert's departure are still being felt. Over the last six months, we have found replacements for Kumfert in the various roles he had, such as chair of the CCA Review Board, point of contact for various collaborations, etc. LLNL is in the process of finding a postdoc to join the project to spread the work load. (Due to LLNL's budget situation and related hiring constraints, it is not feasible to hire a regular staff member at this time.)

A External Collaborations

In this section we briefly summarize the external projects and groups with which we have collaborated during this period, including the TASCS institutions involved in these interactions.

A.1 SciDAC Projects

A.1.1 Applied Partial Differential Equations Center Enabling Technologies (APDEC)

- Project PI: Phil Colella (LBNL)
- Collaboration Point of Contact: Rob Armstrong (SNL)
- TASCS institution: SNL

Collaboration Summary: While a structured mesh component including Automated Mesh Refinement (AMR) exists primarily for the CFRFS combustion CCA application, there are a number of stability issues and functional limitations. Primarily driven by combustion scientists in the CFRFS project, work has begun on a more robust and feature rich component using Chombo, an AMR package provided by APDEC.

Collaboration Progress and Status: A Parallel Cartesian Grid component (non-AMR) already exists in the CCA Toolkit, and the CFRFS still uses an older AMR component based on the GrACE package from Rutgers University. As motivated by the needs of CFRFS applications, work on the new Chombo-based AMR component is proceeding using CFRFS resources and some consulting from CCA; the component is currently in the design stage.

A.1.2 Center for Interoperable Technologies for Advanced Petascale Simulations (ITAPS)

- Project PI: Lori Diachin (LLNL)
- Collaboration Point of Contact: Rob Armstrong (SNL)
- TASCS institutions: LLNL, ORNL, SNL

Collaboration Summary: The ITAPS CET is developing interoperable and interchangeable mesh, geometry, and field manipulation services that are of direct use to SciDAC applications. TASCS is teaming with ITAPS to develop CCA-compliant mesh components based on ITAPS implementations of iMesh.

Collaboration Progress and Status: The ITAPS center has created SIDL wrapper code that can be used with any ITAPS mesh implementation to provide support for all the languages that SIDL supports. TASCS researchers are collaborating with the ITAPS team to build CCA-compliant mesh components based on the MOAB implementation of iMesh. Tests are underway to understand the performance ramifications of using both the C- and SIDL-based iMesh binding for common access functions for mesh data.

The CCA is currently being used in the GWACCAMOLE project (Sec. A.1.8) for ground water applications that employ a home-grown, smoothed-particle hydrodynamics component for the data model. Discussions are underway about transitioning the applications to an unstructured mesh representation that would greatly benefit from this ITAPS mesh component. This work is a good example of how the CCA facilitates a collaboration between SciDAC applications and computer science/math centers.

A.1.3 Center for Scalable Application Development Software (CScADS)

- Project PI: John Mellor-Crummey (Rice University)
- Collaboration Point of Contact: Tom Epperly (LLNL)
- TASCS institutions: LLNL, ORNL

Collaboration Summary: In this collaboration we are working with John Mellor-Crummey's compiler group at Rice to develop component-focused optimizations to remove the computational overhead of the CCA. The runtime overhead from the CCA comes primarily from Babel, which increases the overhead of making a function call by 2 to 5 times a normal Fortran-to-Fortran function call depending on the arguments being passed. Often this overhead is negligible because of the amount of computing that occurs during

the function call, but in some cases, the function call overhead creates a noticeable performance decrease. For those cases, the Rice team is investigating optimizations that can decrease the overhead and improve performance.

Collaboration Progress and Status: Recently, the LLNL Babel team provided Mellor-Crummey's team with an example used by the ITAPS project (Sec A.1.2) to measure the impact of Babel's overhead on performance. In many ways, this test case represents a worst-case scenario because the amount of work performed per function call for low-level mesh operations is minimal. Jeffrey Sandoval performed some initial tests and identified several promising approaches to reduce the overhead, e.g., by removing malloc and free calls. This collaborative research project is ongoing.

A.1.4 Center for the Simulation of RF Wave Interactions with Magnetohydrodynamics (SWIM)

- Project PI: Don Batchelor (ORNL)
- Collaboration Point of Contact: David Bernholdt (ORNL)
- TASCS institutions: IU, ORNL

Collaboration Summary: SWIM is one of three SciDAC prototype projects for the anticipated Fusion Simulation Project. SWIM focuses on integrated modeling of radio-frequency waves and magnetohydrodynamics in magnetically confined plasmas. In order to provide a flexible simulation environment capable of supporting the extensive set of computational modules of interest, SWIM has drawn on component-based software development concepts and experience of the TASCS project to develop a custom component framework for the project. SWIM serves as a testing ground for several new TASCS developments, including MCMD programming (Sec. 3.1.1) and the CCA Event Service (Sec. 4.2), as well as a testbed to explore fault tolerance in component environments (Sec. 3.1.3), particularly in MCMD applications, as part of TASCS collaboration with the CIFTS project (Sec. A.2.4).

Collaboration Progress and Status: This active and ongoing collaboration is currently focused on revamping the SWIM Integrated Plasma Simulator (IPS) framework to support concurrent MCMD computing for the particular computational model in which each task is a separate parallel job (launched with `mpirun` or the equivalent). The IPS also features an event service modeled on the proposed CCA Event Service (simplified in that it only supports polling event delivery, not asynchronous delivery). The SWIM project is exploring an even lighter-weight event service interface that trades performance for simplicity and may prove useful at higher levels of applications, where performance of the event service is not a paramount concern. Depending on the experience of the SWIM developers with this interface, it might be proposed as a "simplified event service" interface for the CCA specification. The IPS has also provided the proof of principle for connectivity between the CCA Event Service and the CIFTS Fault Tolerance Backplane.

A.1.5 Community Petascale Project for Accelerator Science and Simulation (ComPASS)

- Project PI: Panagiotis Spentzouris (FNAL)
- Collaboration Point of Contact: Boyana Norris (ANL)
- TASCS institutions: ANL, Tech-X

Collaboration Summary: The ComPASS-TASCS collaboration is developing interoperable beam dynamics components based on Synergia2 (FNAL), MaryLie/Impact (LBNL), and TxPhysics (Tech-X Corporation). Our goals are to facilitate the interaction among multiple physics modules, such as space charge, electron cloud, and wakefield effects, as well as to provide easy access to scalable software under development by other SciDAC projects. TASCS members L. McInnes and B. Norris are also members of the COMPASS project.

Collaboration Progress and Status: Recently ComPASS and TASCS teamed with S. Muszala (Sec. A.2.1) to develop a prototype component electron cloud simulation that leverages high-level componentization of

Synergia, including SIDL-based wrappers for beam optics components (quadrupoles and drifts) built on MaryLie/Impact, as well as C++ and F90 particle store components based on Synergia2 and a new Synergia2 C++-based space charge component. This work also incorporates new SIDL-based components built on TxPhysics. Further details are in two joint papers [66, 67].

This work extensively employed Bocca, resulting in the rapid development of the new SIDL-based components as well as important feedback to Bocca developers about new features needed by ComPASS. In addition, ComPASS is a motivator for recent work in porting the CCA tools and their prerequisites to LCF machines, with the goal of automating the process as much as possible for future platforms. Also, a joint ComPASS-TASCS-TOPS collaboration to dynamically select and parametrize algebraic solvers in beam dynamics simulations motivates work in the CQoS Initiative (Sec. 3.3).

A.1.6 Computational Facility for Reacting Flow Science (CFRFS)

- Project PI: Habib Najm (SNL)
- Collaboration Point of Contact: Jaideep Ray (SNL)
- TASCS institutions: ANL, SNL

Collaboration Summary: As highlighted in Sec. 2, TASCS is helping CFRFS to design and implement a component toolkit for reacting flows. The focus is on enabling high-fidelity simulations of lab-scale flames using detailed chemical mechanisms. The low-Mach approximation of the Navier-Stokes equations is used to model momentum and energy transport. The toolkit performs its simulations on block-structured adaptive meshes and employs advanced numerical schemes, specifically fourth-order spatial discretizations and extended-stability, explicit time-integrators on block-structured mesh hierarchies.

The CFRFS toolkit is maturing. Laminar reacting flow calculations are becoming routine. Much of the CCA design (e.g., the components, ports) and CCA-related details (e.g., the framework) are considered by the investigators as infrastructure and only mentioned in passing in talks and papers. The principal investigator, Habib Najm, cites the modularity of CFRFS's component-oriented design as a key feature that enables the seamless incorporation of new code by all CFRFS researchers, including many short-term visitors of the Combustion Research Facility. Existing components can be easily used and modified, while new components can be added, without endangering the previous functionality, nor constraining future additions to the simulation. The CFRFS Toolkit today contains over 100 components; most simulations use about 30–40 of them at a time.

Collaboration Progress and Status: Building on earlier developments of components for the energy and species equations [68], an important aspect of the entire construction – the projection method for the momentum equations – was recently implemented [76] and fourth-order convergence demonstrated [77]; see Fig. 4. Scalability results were shown for up to approximately 500 processors.

CFRFS researchers also collaborate with TASCS in the CQoS Initiative (Sec. 3.3) to motivate and validate new tools that facilitate the dynamic composition and reconfiguration of components. CFRFS is using this infrastructure to develop a meta-partitioner to be used to partition and load-balance parallel simulations of reactive flows on block-structured, adaptively-refined meshes.

A.1.7 Framework Application for Core-Edge Transport Simulations (FACETS)

- Project PI: John Cary (Tech-X)
- Collaboration Point of Contact: Sveta Shasharina (Tech-X)
- TASCS institutions: ANL, IU, LLNL, Tech-X

Collaboration Summary: The FACETS (Framework Application for Core-Edge Transport Simulations) project [35] began in January 2007 with the goal of providing core to wall transport modeling of a tokamak fusion reactor. This work involves coupling previously separate computations for the core, edge, and wall regions and developing new codes. Such coupling is primarily through connection regions of lower

dimensionality. The project has started developing a component-based coupling framework to bring together models for each of these regions. Our collaboration fall into two categories: directly contributing to FACETS development and providing consulting and educational support to FACETS physicists.

Collaboration Progress and Status: The collaboration between FACETS and TASCS has been very active and successful. FACETS embraces the paradigm of component-based software engineering, and several researchers from the TASCS team are also members of FACETS.

FACETS uses Babel to couple legacy codes written in variants of Fortran and Python to the FACETS framework, which is written in C++. For example, transport modules such as glf23 and mmm95 were wrapped in SIDL and used in the first version of the FMCFM (Framework for Modernization and Componentization of Fusion Modules, developed at Tech-X in a different project), which is essential for FACETS core modeling. The tokamak edge is modeled in FACETS by the UEDGE code, which is also called through Babel bindings. To enable use of Babel on the LCFs, we provided a static Babel build for FACETS researchers. Our work with this project resulted in joint papers [28, 69], and the result of code integration enabled by Babel is visualized in Fig. 3.

The educational and consulting help to FACETS took the form of informal seminars and joint testing of TASCS software. Presentations about CCA components and Bocca were given at Tech-X Corporation, and the full CCA tool chain, including Bocca and OnRamp, is installed on multiple Tech-X computers. We have worked on integration of CCA tools with the FACETS build system and developed several svn repositories with working examples of CCA components and Bocca usage.

A.1.8 GroundWAter CCA MOdeling Library and Extensions (GWACCAMOLE)

- Project PI: Bruce Palmer (PNNL)
- Collaboration Point of Contact: Manojkumar Krishnan (PNNL)
- TASCS institutions: ORNL, PNNL, SNL

Collaboration Summary: The GWACCAMOLE (GroundWAter CCA MOdeling Library and Extensions) SciDAC SAP is building a component-based HPC framework for subsurface simulations using a hybrid approach to combine different physical models into a single coherent simulation. Existing parallel simulation tools will be decomposed into components for use within the CCA framework and will be extended with components providing interfaces between the different models. The underlying models for these interfacial components will be developed as part of the associated science application. The framework will also incorporate new numerical grid and solver technologies being developed by other SciDAC centers to provide a flexible framework for large-scale simulations of subsurface reactive flows.

Collaboration Progress and Status: We recently added a simple chemistry component to the GWACCAMOLE Smoothed Particle Hydrodynamics (SPH) framework. This work involved refactoring the STOMP (Sec. A.1.9) subsurface continuum simulation package into two components: a physics component and a mesh component, where the SIDL component interface can support both rectangular and unstructured meshes. Discussion is in progress with ITAPS (Sec. A.1.2) on incorporation of ITAPS unstructured mesh libraries. We also ported the SPH framework to the new Bocca project management infrastructure. We are beginning large-scale SPH simulations of diffusive transport in porous media, including a nearly complete port of the SPH framework to the NERSC Franklin machine. This project uses the MCMD model introduced in Sec. 3.1 to incorporate task parallelism among components.

A.1.9 Hybrid Numerical Methods for Multiscale Simulations of Subsurface Biogeochemical Processes

- Project PI: Tim Schiebe PNNL
- Collaboration Point of Contact: Manojkumar Krishnan (PNNL)
- TASCS institution: PNNL

Collaboration Summary: This SciDAC project is developing an integrated multiscale modeling framework with the capability of directly linking different process models at continuum, pore, and sub-pore scales. Current TASCS collaborations focus on adding MCMD capabilities to the Subsurface Transport Over Multiple Phases (STOMP) application for subsurface flow and transport. As discussed in Sec. 3.1, the MCMD implementation incorporates task parallelism between the components as well as data parallelism within the components. This work will provide a validation of the CCA *teams* model, while allowing the STOMP developers to conduct parameter studies using multiple instances of the parallel STOMP component to simulate flows through heterogeneous, porous media at Hanford.

Collaboration Progress and Status: PNNL has been developing a MCMD implementation for large-scale subsurface simulations. A key capability is the estimation of spatially variable subsurface geologic material properties using field observations of multivariate data types and sample sizes. A multimodal minimization methodology uses forward models of hydrologic, geochemical, and geophysical processes to estimate the desired properties.

A.1.10 Quantum Chemistry SAP

- Project PI: Mark Gordon (Ames Lab)
- Collaboration Point of Contact: Joe Kenny (SNL)
- TASCS institutions: ANL, PNNL, SNL

Collaboration Summary: As highlighted in Sec. 2 and shown in Fig. 5, TASCS supports the Quantum Chemistry SAP in their mission to develop interfaces among major quantum chemistry packages, creating a flexible, community-based software development environment. This application project employs CCA infrastructure to develop interoperable components based on three important computational chemistry codes: General Atomic and Molecular Electronic Structure System (GAMESS), the Massively Parallel Quantum Chemistry program (MPQC), and Northwest Chem (NWChem). Component technology allows not only interchangeability of high-level capabilities provided by participating codes but also the ability to mix and match low-level components providing such services as integral evaluation and the formation of operator matrices [79].

Collaboration Progress and Status: As early adopters of the CCA approach, the Quantum Chemistry SAP development team has settled into a development process and is using CCA software as a production tool, enabling the project to focus on meeting its scientific objectives. Recent scientific progress has included a CCA component-based hybrid quantum/classical simulation capability [80]. The application developers provide useful feedback as TASCS members further harden the CCA software stack for production use. Recently, a particularly active collaboration has been sustained between QCSAP developers and the TASCS CQoS Initiative (Sec. 3.3). This work has led to the design and implementation of prototype applications to more efficiently manage computational resources during complicated quantum chemistry calculations [1, 18].

A.1.11 Performance Engineering Research Institute (PERI)

- Project PI: Bob Lucas (ISI)
- Collaboration Point of Contact: Boyana Norris (ANL)
- TASCS institution: ANL

Collaboration Summary: The TASCS CQoS Initiative (Sec. 3.3) aims to systematically identify and improve component-based software by applying component-wise performance engineering techniques. This collaboration involves the instrumentation of codes, the collection of performance profiles of components, and the storage of that data in a performance database that makes optimization possibilities more clear to software designers.

Collaboration Progress and Status: PERI has defined a standard representation for application metadata, which is required by the CQoS Initiative. The analysis portions of the CQoS infrastructure classify performance results based on parameters that are part of the metadata. The initial implementation steps toward fully automated tuning of component applications leverage PERI performance measurement tools and are beginning to incorporate some of the infrastructure being developed by the PERI autotuning working group (of which B. Norris is a member).

A.1.12 Scientific Data Management Center (SDM)

- Project PI: Arie Shoshani (LBNL)
- Collaboration Point of Contact: Kosta Damevski (VSU)
- TASCS institutions: UU, VSU

Collaboration Summary: The focus of this collaboration is to have interoperability between the software architectures used by the TASCS and SDM centers in order to enable applications to use the benefits of both approaches. The SDM center uses scientific workflows that are concerned with high-level orchestration of time-decomposed simulations, while TASCS's components are mainly used at a lower level to enable software modularity and reuse. Combining these technologies in a seamless way leverages the benefits of both paradigms, resulting in more flexibility for scientific application design.

Collaboration Progress and Status: A proof of concept application that uses both scientific workflows and components has been designed and implemented. The next step is to identify a meaningful scientific application that would benefit from using both technologies.

A.1.13 Towards Optimal Petascale Simulations (TOPS)

- Project PI: David Keyes (Columbia University)
- Collaboration Point of Contact: Lois Curfman McInnes (ANL)
- TASCS institutions: ANL

Collaboration Summary: The focus of TASCS-TOPS collaboration is the development of scalable TOPS linear/nonlinear solver components, with emphasis on the needs of beam dynamics applications in the SciDAC ComPASS accelerator project (Sec. A.1.5) and core-edge plasma applications in the SciDAC FACETS fusion project (Sec. A.1.7). We also develop numerical optimization components based on TAO, which are employed in quantum chemistry simulatons [81] (Sec. A.1.10). The selection and parametrization of solver algorithms and implementations in long-running simulations motivate work in the SQV (Sec. 3.2) and CQoS (Sec. 3.3) Initiatives.

Collaboration Progress and Status: We have developed prototype high-level, language-independent components for the scalable solution of large linear and nonlinear algebraic systems arising from either structured or unstructured meshes. These components, compliant with the CCA and written using SIDL, can interface to underlying solvers provided by a large variety of libraries developed at various institutions, including hypre (LLNL), SuperLU (LBNL), and PETSc (ANL).

The common interfaces employed by TOPS solver components enable easy access to suites of independently developed algorithms and implementations. The challenge then becomes how, during runtime, to make the best choices for reliability, accuracy, and performance. We are currently extending TOPS components to incorporate new CQoS capabilities to facilitate appropriate choices for algorithms and parameters of TOPS linear and nonlinear solver components.

A.2 Other DOE Projects

A.2.1 Common Component Architecture for Electron Cloud Accelerator Simulations

- Project PI: Stefan Muszala (Tech-X)
- Collaboration Point of Contact: Boyana Norris (ANL)

- TASCS institutions: ANL, Tech-X

Collaboration Summary: The CCA Ecloud project models the Electron Cloud Effect (ECE) using CCA components. Electrons bouncing off of particle accelerator beam walls cause the wall to emit more electrons through secondary emission and eventually to build into a cloud. The ECE is important to particle accelerator simulations because the cloud causes the proton beam to degrade. The scope of this project involves wrapping and coupling TxPhysics and Synergia2, while addressing computational quality of service requirements (Sec. 3.3) through the use of TAU performance tools (Sec. A.2.12).

Collaboration Progress and Status: Recent work focused on developing SIDL-based components based on the TxPhysics package and using these in conjunction with Synergia2-based components developed by the ComPASS project (Sec. A.1.5) to create a new prototype CCA electron cloud application. Ionization routines from the TxPhysics library were wrapped to allow their use as function mapped components. Initial validation demonstrates that the component and non-component applications produce closely matching numbers of electrons.

Also, we recently tested OnRamp (Sec. 4.3) within TxPhysics (used by ComPASS codes Synergia and VORPAL) to demonstrate the ease of creating CCA components for existing physics codes and to compare Bocca-generated SIDL interfaces with the interfaces developed manually at Tech-X. The success of this effort will hopefully further promote CCA tools among physics application developers.

A.2.2 Contractor Meta-Build System

- Project PI: James Amundson (Fermilab)
- Collaboration Point of Contact: Boyana Norris (ANL)
- TASCS institution: ANL

Collaboration Summary: The Contractor meta-build system[8] was designed to manage complex processes involving configuration, building, and installation of software, such as when numerous interdependent packages must be configured and installed in a variety of environments.

Collaboration Progress and Status: As discussed in Sec. 4.1, the standard distribution of the CCA tools is now based on Contractor, and joint development of the CCA-Tools-Contractor capabilities continues.

A.2.3 Cooperative Programming (Co-Op)

- Project PI: John May (LLNL)
- Collaboration Point of Contact: Tom Epperly (LLNL)
- TASCS institution: LLNL

Collaboration Summary: A LLNL LDRD project developed the Cooperative Programming Model (Co-op) model that uses Babel RMI to manage collections of MPI jobs to perform large, multi-scale, multi-physics applications. This project considered an approach to petascale computing based on simultaneously running multiple parallel jobs. For example, an application scales well to 1000 processors, and 100 thousand-processor runs can be performed simultaneously using Co-op, a program has been created that scales to 100,000 processors. Initially, this work was applied to material modeling, where a continuum model was coupled with a fine-scale model that was automatically invoked when the continuum assumption broke down [91–93]. More recently, this work has also been applied to a discrete event modeling simulation for space situational awareness. This simulation models the systems for tracking satellites and known space junk orbiting earth.

Collaboration Progress and Status: The basic Co-op system is largely stable, and recent activities have focused on maintenance and new applications. In addition to materials modeling and space situational

[8] `http://home.fnal.gov/~amundson/contractor-www`

awareness, some other LLNL multi-physics applications have investigated using Co-op for special-purpose calculations.

A.2.4 Coordinated Infrastructure for Fault Tolerance of Systems (CIFTS)

- Project PI: Pete Beckman (ANL)
- Collaboration Point of Contact: David Bernholdt (ORNL)
- TASCS institution: ORNL

Collaboration Summary: As the project name suggests, CIFTS is developing an infrastructure to make fault information available throughout the software stack (from the device drivers and operating system to the applications) and to facilitate a coordinated response to faults when they occur. Central to the CIFTS project is the Fault Tolerance Backplane (FTB), an event service to disseminate fault information. As part of the TASCS Emerging HPC Initiative (Sec. 3.1), we are collaborating with CIFTS to connect the CCA environment to the FTB and explore how fault-related abstractions might be conveniently expressed in the component environment.

Collaboration Progress and Status: Initial exploration of fault tolerance in component environments is taking place primarily in collaboration with the SWIM fusion project (Sec. A.1.4), which is also making use of TASCS MCMD and Event Service concepts. After we gain some practical experience with fault tolerance for real simulations running in the SWIM IPS, we plan to extend these ideas into fully CCA-compliant capabilities.

A.2.5 Distributed CCA Components and Grid Services for Scientific Computing

- Project PI: Nanbor Wang (Tech-X)
- Collaboration Point of Contact: Sveta Shasharina (Tech-X)
- TASCS institutions: LLNL, Tech-X

Collaboration Summary: TASCS and Tech-X jointly worked on providing struct support for Fortran2003 in Babel and demonstrated using Babel Simple RMI to perform remote data analysis of the code VORPAL.

Collaboration Progress and Status: The project has succeeding in developing the struct support to be used in FACETS (Sec. A.1.7) to integrate transport modules into the FACETS framework for coupled core-edge-wall modeling of a tokamak fusion reactor.

A.2.6 High-Performance Mass Spectrometry Facility

- Project PI: Mikhail Belov (PNNL)
- Collaboration Point of Contact: Manojkumar Krishnan (PNNL)
- TASCS institution: PNNL

Collaboration Summary: The High-Performance Mass Spectrometry Facility[9] is teaming with the Emerging HPC Initiative (Sec. 3.1) to evaluate the use of a newly developed high-performance Event Service implementation, which the PNNL team recently developed to exploit efficient communication mechanisms commonly used on HPC platforms. We designed and implemented the Event Service using the Aggregate Remote Memory Copy Interface (ARMCI) as an underlying communication layer for this mechanism. Two alternative implementations were developed and evaluated on a Cray XD-1 platform.

Collaboration Progress and Status: The performance results demonstrated that event delivery latencies are low and that the Event Service is able to achieve high throughput levels. The Event Service has been used to distribute work and coordinate execution between CPU-only nodes and nodes with attached accelerators (e.g., FPGA, GPU) on a HPC system. In order to take advantage of the FPGA accelerators from a global system and application perspective, the CPU-only nodes can request processing by the nodes with attached

[9] http://www.emsl.pnl.gov/capabs/hpmsf.shtml

FPGAs. We applied this model to a computational proteomics application involving inverse Hadamard transform signal processing, where the FPGA accelerator achieved a 6x speed advantage over pure software processing. The data transfer time between CPU-only requester nodes and the nodes with attached FPGAs is minimal, compared to the signal processing computation. This application demonstrated the use of CCA-based hybrid computing strategies on a scalable platform [8, 9].

A.2.7 Nuclear Energy Advanced Modeling and Simulation (NEAMS)

- NEAMS Program Manager: Alex Larzelere (DOE)
- IBM-FOA Project PI: George Chiu (IBM)
- NEAMS-CT Project PI: Gil Weigand (ORNL)
- Collaboration Point of Contact: David Bernholdt (ORNL)
- TASCS institution: ORNL

Collaboration Summary: Nuclear Energy Advanced Modeling and Simulation (NEAMS) is a program in the DOE Office of Nuclear Energy with the goal of bringing state of the art high-performance computational science and engineering capabilities to revamp the way nuclear power systems are designed and developed. An important part of the NEAMS vision is a framework to provide a unified environment for the development, composition, and execution of a wide range of simulations modeling nuclear fuels, reactors, reprocessing, and waste forms. The NEAMS framework is expected to be based on component concepts, and the CCA is being considered as a possible infrastructure on which to develop the domain-specific framework.

Collaboration Progress and Status: Currently two active projects within the NEAMS program include design work on the NEAMS framework. The IBM-FOA project began in October 2008, and the NEAMS-CT effort was funded in March 2009, with the framework efforts in both led by David Bernholdt (ORNL). NEAMS is using model-driven system design (MDSD) and Unified Modeling Language (UML) methodology and tools to analyze the NEAMS problem domain, as well as major software systems used in the domain, and candidate infrastructure, including the CCA.

A.2.8 NWChem

- Project PI: Bert de Jong (PNNL)
- Collaboration Point of Contact: Manojkumar Krishnan (PNNL)
- TASCS institution: PNNL

Collaboration Summary: NWChem is a large (2.5 million lines of code) suite of computational chemistry algorithms that was developed based on multiple languages (Fortran, C, C++, Python) and programming models (MPI, Global Arrays). Although NWChem has been designed from scratch to work on massively parallel systems, until recently it was unable to effectively exploit variable degrees of parallelism available in the set of algorithms and methods it offers. As a result, the scalability of some important calculations was limited by the least scalable parts of the simulation.

Collaboration Progress and Status: We are using CCA in the context of computational chemistry to express and manage multiple levels of parallelism (MLP) through the use of processor groups [83]. As discussed in Sec. 3.1, we are using the MCMD approach to manage MLP. To exploit available hardware parallelism in petascale systems, exploitation of MLP is essential. This technique has been shown to increase granularity of computation and thus improve the overall scalability.

A.2.9 Polygraph

- Project PI: Bill Cannon and Doug Baxter (PNNL)
- Collaboration Point of Contact: Manojkumar Krishnan (PNNL)
- TASCS institution: PNNL

Collaboration Summary: We have studied the use of CCA components for creating portable interfaces for custom algorithms implemented on hardware accelerators (Sec. 3.1). These components can cleanly encapsulate the entire low-level, platform specific details of hardware accelerated implementations, while providing a high-level interface that is 100% compatible with components implementing software versions of the same algorithm. To demonstrate this concept, we have created a component-based version of an application in proteomics named Polygraph.

Collaboration Progress and Status: Polygraph, developed at PNNL, uses an innovative approach to extract mass spectra from experimental databases given a description of a candidate peptide. Polygraph utilizes a genetic algorithm approach to find the best matches for the candidate peptide sequence, which is a compute intensive process for long sequences on large databases. We developed an FPGA-accelerated version of a key computational kernel in Polygraph, and demonstrated the ease of integrating the FPGA-accelerated kernel into the rest of the software application using CCA component technology. We also built a CCA component of the original software version of the kernel and demonstrated seamless interoperability between the FPGA and software versions, with very low overhead compared to non-componentized, monolithic versions [7].

A.2.10 ROSE

- Project PI: Dan Quinlan (LLNL)
- Collaboration Point of Contact: Tom Epperly (LLNL)
- TASCS institution: LLNL

Collaboration Summary: The ROSE project uses Babel to link a graphical front-end written in Java to the ROSE parsing and analysis engine in C++ [70]. By linking these two sub-systems with Babel, the graphical front-end is able to visualize large-scale software systems and incorporate various pieces of information determined by ROSE analysis into a single view.

Collaboration Progress and Status: Currently, the visualization tool is not under active development, so the project does not require active support at this time.

A.2.11 SPARSKIT-CCA

- Project PI: Masha Sosonkina (Ames Laboratory)
- Collaboration Point of Contact: Lois Curfman McInnes (ANL)
- TASCS institutions: ANL

Collaboration Summary: SPARSKIT-CCA, a suite of CCA components for the parallel iterative solution of sparse linear systems, is part of the CCA Toolkit (Sec 5). Major components implement various parallel preconditioners and accelerators available in the original SPARSKIT, its modern extension, ITSOL, and the pARMS parallel solution library. The matrix storage format is very flexible and may be easily extended. The list of preconditioners and accelerators may also be easily extended due to the standard interfaces developed for SPARSKIT-CCA.

Collaboration Progress and Status: SPARSKIT, developed in the 90s by Yousef Saad at the University of Minnesota, is a basic toolkit for sequential sparse matrix computations and is widely used in scientific community. Written in Fortran 77 and having a cumbersome interface, it is considered a legacy code.

Our first objective is to enable its wider usage in modern applications and to facilitate further development of SPARSKIT. The component interfaces are defined according to CCA requirements. The SPARSKIT-CCA design features matrix-free interfaces, which enable its usage with a variety of matrix formats and iterative solution implementations of preconditioners and accelerators. In particular, we have integrated SPARSKIT-CCA, which has a medium-level interface granularity, with LISI interfaces [44, 45], which provide higher-level access to linear system solution methods and encapsulate both preconditioners

and accelerators within a joint interface. Our second goal – creation of parallel SPARSKIT components – is well underway. We have implemented a distributed matrix component, components for graph partitioners, and parallel preconditioners. Combining all these ingredients and careful testing constitute our forthcoming efforts.

A.2.12 Tuning Analysis and Utilities (TAU)

- Project PI: Allen Malony (UO)
- Collaboration Point of Contact: Li Li (ANL)
- TASCS institution: ANL

Collaboration Summary: The collaboration between the TAU group and TASCS has the goal of developing performance monitoring and analysis components, with emphasis on dynamic configuration and adaptation of computational components during the execution of a scientific application.

Collaboration Progress and Status: The collaboration is developing CCA-compliant performance monitoring (e.g., through automated performance proxy component generation) and analysis capabilities based on TAU as part of the CCA Toolkit (Sec 5). A second point of interaction is performing analysis and prediction based on the performance data to help make runtime decisions about component reconfiguration and adaptation [19, 22].

A highlight of recent work is the extension of PerfExplorer, a framework for parallel performance data mining and knowledge discovery, as motivated by the CQoS Initiative (Sec. 3.3) and the Quantum Chemistry Science Application Partnership (Sec. A.1.10). We integrated performance analysis capabilities of PerfExplorer into general CQoS infrastructure to classify performance and meta-information for quantum chemistry computations and then suggested appropriate configurations for new problem instances [1].

A.3 Other Sponsors

A.3.1 Center for Integrated Space Weather Modeling (CISM)

- Project PI:
- Collaboration Point of Contact: Alan Sussman (UMD)
- TASCS institution: UMD

Collaboration Summary: The Center for Integrated Space Weather Modeling (CISM) is an NSF Science and Technology Center focused on building a set of coupled physical models for space weather to fully model the Sun to Earth environment. The models are written in different programming languages, with some parallel (MPI or shared memory) and some sequential. Currently the physical coupling between the models uses the Maryland InterComm software, and the goal of the TASCS collaboration is to introduce the space weather modeling community, and the CISM participants more immediately, to the benefits and capabilities of a complete CCA environment.

Collaboration Progress and Status: We have designed, built, and tested SIDL interfaces for the InterComm API. We have built several examples with multiple parallel components and have run them using the Ccaffeine framework. Each parallel component runs in a separate Ccaffeine instance and makes calls to InterComm components to export or import data to a parallel component in a different instance of the framework. In the near future we will work with CISM space physicists to encapsulate one or more of their models as CCA components, so that they can determine if CCA technology can provide them with services not currently provided by InterComm or other model coupling tools they are currently using.

A.3.2 Chapel Language Development Team

- Project PI: Brad Chamberlain (Cray)
- Collaboration Point of Contact: Tom Epperly (LLNL)
- TASCS institutions: LLNL

Collaboration Summary: Brad Chamberlain (Cray) teamed with G. Kumfert and T. Epperly (LLNL) to consider the challenge of introducing High-Productivity Computing Systems (HPCS) computer languages into Babel. We developed a list of requirements and technical challenges that would need to be overcome. Extending Babel to incorporate Parallel Global Address Space (PGAS) languages is a major research challenge because parallelism in PGAS languages is implicit, and with the serial languages Babel already supports, parallelism is explicit. Ultimately, we authored a joint proposal to seek funding to work on this and submitted it to the DOE's call for proposals in the area of Petascale Tools Development.

Collaboration Progress and Status: The collaboration is awaiting a funding source.

A.3.3 Community Surface Dynamics Modeling System (CSDMS) Integration Facility

- Project PI: Scott Peckham (University of Colorado at Boulder)
- Collaboration Point of Contact: Boyana Norris (ANL)
- TASCS institution: ANL

Collaboration Summary: The CSDMS project is a community effort that deals with modeling the Earth's surface, specifically the dynamic interfaces between lithosphere, hydrosphere, cryosphere, and atmosphere. The goal of the Integration Facility is to provide component interfaces to the numerous existing (and future) models with the goal of enabling faster model development, easier access to and better usability of simulation codes, and coupling models in HPC environments in the future.

Collaboration Progress and Status: B. Norris has assisted The National Science Foundation (NSF)-funded Community Surface Dynamics Modeling System (CSDMS) Integration Facility in evaluating the suitability of CCA technology for providing an appropriate environment for multiple models and their implementations. At present CSDMS has adopted the CCA component model and integrated it with the Open Modelling Interface and Environment (OpenMI) standard [94] for environmental modeling [95]. Interactions with CSDMS have also resulted in improvements and extensions of CCA tool infrastructure, especially Bocca (Sec. 4.3.1).

A.3.4 HPC Application Software Consortium (HPC-ASC)

- Organizers: Merle Giles (NCSA), Robert Graybill (ISI)
- Collaboration Point of Contact: David Bernholdt (ORNL)
- TASCS institutions: LLNL, ORNL, SNL

Collaboration Summary: This nascent consortium is intended to address the needs of industrial HPC users for better scalability and interoperability in simulation tools developed (primarily) by commercial independent software vendors. These challenges are closely related to many of the issues the CCA Forum was created to address, but (at least initially) are more focused on particular code-coupling issues, and with the added complications of most of the software involved being proprietary, developed by for-profit entities.

Collaboration Progress and Status: TASCS project members G. Kumfert (LLNL), R. Armstrong (SNL) and D. Bernholdt (ORNL) have engaged with the HPC Application Software Consortium (HPC-ASC) leadership at several face-to-face meetings, as well as email exchanges and teleconferences. TASCS and the CCA Forum have offered their experience in organizing and running community-based standards organizations, as well as technical experience with component technology and related middleware in order to help the consortium make progress towards their goals. We are also interested in getting more insight into the multi-physics code-coupling problems they need to address in order to inform our own long-term interest in developing abstractions and tools to generally support solving such problems in component environments like the CCA. At present, after an initial flurry of activity, the consortium's progress seems to have slowed.

B TASCS Publications and Presentations

[1] Li Li, Joseph P. Kenny, Meng-Shiou Wu, Kevin Huck, Alexander Gaenko, Mark S. Gordon, Curtis L. Janssen, Lois Curfman McInnes, Hirotoshi Mori, Heather M. Netzloff, Boyana Norris, and Theresa L. Windus, Adaptive Application Composition in Quantum Chemistry, in *5th International Conference on the Quality of Software Architectures (QoSA 2009)*, pages 1–17, 2009, (in press).

[2] David E. Bernholdt, Software Architecture for High-Performance Scientific Computing: A Pragmatic Approach, course (8.5 hours lecture, 3 hours lab), CEA-EDF-INRIA Summer School: Design of High-Performance Scientific Applications, St. Remy les Chevreuse, France, 2006.

[3] Wei Lu, Kenneth Chiu, Satoshi Shirasuna, and Dennis Gannon, Hybrid Decomposition Scheme for Building Scientific Workflows, in *Proceedings of High Performance Computing Symposium (HPC 2007), Norfolk, Virginia, March 25–29*, pages 388 –394, ACM, 2007, Best paper award.

[4] David E. Bernholdt, The Common Component Architecture: Building Frameworks for Computational Science, keynote, International Conference on Modeling and Simulation in the Petroleum Industry, 2007.

[5] Lois Curfman McInnes, Tamara Dahlgren, Jarek Nieplocha, David Bernholdt, Benjamin A. Allan, Rob Armstrong, Daniel Chavarria, Wael Elwasif Ian Gorton, Manoj Krishan, Allen Malony, Boyana Norris, Jaideep Ray, and Sameer Shende, Research Initiatives for Plug-and-Play Scientific Computing, in David Keyes, editor, *SciDAC 2007, 24–28 June 2007, Boston, Massachusetts, USA*, volume 78 of *Journal of Physics: Conference Series*, page 012046, Institute of Physics, 2007.

[6] Steven Parker, Rob Armstrong, David Bernholdt, Tamara Dahlgren, Tom Epperly, Joseph Kenny, Manoj Krishnan, Gary Kumfert, Jay Larson, Lois Curfman McInnes, Jarek Nieplocha, Jaideep Ray, and Sveta Shasharina, Enabling Advanced Scientific Computing Software, *CTWatch Quarterly* **3** (2007), invited article,.

[7] Daniel Chavarria-Miranda, Jarek Nieplocha, and Ian D. Gorton, Hardware-Accelerated Components for Hybrid Computing Systems, in *Proceedings of the 2008 CompFrame/HPC-GECO Workshop on Component Based High Performance Computing*, ACM, 2008.

[8] Ian D. Gorton, Daniel Chavarria-Miranda, and Jarek Nieplocha, Design and Implementation of a High-Performance CCA Event Service, *Concurrency and Computation: Practice and Experience* (2009), in press.

[9] Ian Gorton, Daniel Chavarria-Miranda, Manojkumar Krishnan, and Jarek Nieplocha, A High-Performance Event Service for HPC Applications, in *SE-HPC '07: Proceedings of the 3rd International Workshop on Software Engineering for High Performance Computing Applications*, page 1, Washington, DC, USA, 2007, IEEE Computer Society, 5pp,.

[10] Tamara L. Dahlgren, Performance-Driven Interface Contract Enforcement for Scientific Components, in *Component-Based Software Engineering*, volume 4608 of *Lecture Notes in Computer Science*, pages 157–172, Berlin/Heidelberg, 2007, Springer.

[11] Tamara Dahlgren, David Bernholdt, and Lois Curfman McInnes, Gaining Confidence in Scientific Applications Through Executable Interface Contracts, in Rick Stevens, editor, *SciDAC 2008, 14-17 July 2008, Seattle, Washington, USA*, volume 125 of *Journal of Physics: Conference Series*, page 012086, Institute of Physics, 2008.

[12] T. Dahlgren, Interface Contracts for Scientific Components and Libraries, seminar, Mathematics and Computer Science Division Colloquium, Argonne National Laboratory, Argonne, IL, 2008.

[13] Li Li, Boyana Norris, Van Bui, Kevin Huck, Joseph P. Kenny, Lois Curfman McInnes, Heather Netzloff, and Meng-Shiou Wu, Database and Analysis Support for Automated Configuration of Scientific Applications, talk, SIAM Conference on Computational Science and Engineering, 2009.

[14] Li Li, Boyana Norris, and Lois Curfman McInnes, Database Components for Support of Computational Quality of Service for Scientific CCA Applications, talk, SIAM Conference on Parallel Processing for Scientific Computing (PP08), 2008.

[15] Boyana Norris, Enabling Adaptive Algorithms through Component-Based Software Engineering, seminar, RWTH Aachen, Aachen, Germany, 2007.

[16] Boyana Norris, Lois Curfman McInnes, Sanjukta Bhowmick, and Li Li, Adaptive Numerical Components for PDE-Based Simulations, *PAMM: Special Issue: Sixth International Congress on Industrial Applied Mathematics (ICIAM07) and GAMM Annual Meeting, Zurich 2007* **7**, 1140509 (2007).

[17] Jaideep Ray, Cosmin Safta, and Habib Najm, Designing Adaptive Mesh Simulators for Reacting Flows Using the Common Component Architecture, talk, SIAM Conference on Computational Science and Engineering, 2009.

[18] J. Kenny, Kevin Huck, L Li, L McInnes, H Netzloff, B Norris, M. Wu, A. Gaenko, and H. Mori, Computational Quality of Service (CQoS) in Quantum Chemistry, poster, 2008 ACM/IEEE conference on Supercomputing, 2008.

[19] Van Bui, Boyana Norris, Kevin Huck, Lois Curfman McInnes, Li Li, Oscar Hernandez, and Barbara Chapman, A Component Infrastructure for Performance and Power Modeling of Parallel Scientific Applications, in *Proceedings of the 2008 CompFrame/HPC-GECO Workshop on Component Based High Performance Computing*, ACM, 2008.

[20] Boyana Norris, Lois McInnes, Sanjukta Bhowmick, and Li Li, Adaptive Numerical Components for PDE-Based Simulations, in *Proceedings of ICIAM 2007, Zurich, Switzerland*, 2007.

[21] L Li, B Norris, H Johansson, L McInnes, and J Ray, Component Infrastructure for Managing Performance Data and Runtime Adaptation of Parallel Applications, in *Proceedings of PARA08 (9th International Workshop on State-of-the-Art in Scientific and Parallel Computing)*, 2008.

[22] Kevin Huck, Oscar Hernandez, Van Bui, Sunita Chandrasekaran, Barbara Chapman, Allen D. Malony, Lois Curfman McInnes, and Boyana Norris, Capturing Performance Knowledge for Automated Analysis, in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–10, Piscataway, NJ, USA, 2008, IEEE Press.

[23] Tamara L. Dahlgren, Performance-Driven Interface Contract Enforcement for Scientific Components, Technical Report CSE-2008-6 (Also LLNL UCRL-TH-235341), University of California, Davis, Davis, CA, 2008, Ph.D. Thesis.

[24] Nanbor Wang, Paul Hamill, Fang Liu, Steve Tramer, Roopa Pundaleeka, and Randall Bramley, Integrating Large-scale Distributed and Parallel HPC (DPHPC) Applications using a Component-Based Architecture, in *Proceedings of the 2008 CompFrame/HPC-GECO Workshop on Component Based High Performance Computing*, 2008.

[25] Yinfei Pan, Ying Zhang, and Kenneth Chiu, Simultaneous Transducers for Data-Parallel XML Parsing, in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, IEEE, 2008.

[26] Yinfei Pan, Ying Zhang, Kenneth Chiu, and Wei Lu, Parallel XML Parsing Using Meta-DFAs, in *E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, pages 237–244, Washington, DC, USA, 2007, IEEE Computer Society.

[27] Rajdeep Bhowmik, Chaitali Gupta, Madhusudhan Govindaraju, and Aneesh Aggarwal, Optimizing XML Processing for Grid Applications Using an Emulation Framework, in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–11, IEEE, 2008.

[28] S. Muszala, T. Epperly, and N. Wang, Babel Fortran 2003 Binding for Structured Data Types, in Jack Dongarra, Anne C. Elster, and Jerzy Wasniewski, editors, *Applied Parallel Computing: 9th International Conference, PARA 2008, Trondheim, Norway, 13-16 May, 2008*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2008, (in press).

[29] Tharaka Devadithya, Kenneth Chiu, and Wei Lu, C++ Reflection for High Performance Problem Solving Environments, in *Proceedings of High Performance Computing Symposium (HPC 2007), Norfolk, Virginia, March 25–29*, pages 435–440, ACM, 2007.

[30] Kostadin Damevski, Ayla Khan, and Steven Parker, Scientific Workflows and Components: Together at Last!, in *Proceedings of the 2008 CompFrame/HPC-GECO Workshop on Component Based High Performance Computing*, 2008.

[31] Wael R. Elwasif and Benjamin A. Allan, Incorporating Legacy Libraries in Modern HPC Component Enviornments, in *PARA08-9th International Workshop on State-of-the-Art in Scientific and Parallel Computing*, Trondheim, Norway, 2008, (in press).

[32] Geoffrey Hulette, Matthew J. Sottile, Benjamin A. Allan, and Robert C. Armstrong, Using CCA and Onramp to Generate an Application-specific Framework from a Monolithic Application, poster, 2008 ACM/IEEE conference on Supercomputing, 2008.

[33] Benjamin A. Allan and Boyana Norris, Automating SIDL-Based Development for New and Legacy Software, in *Proceedings of the 2008 CompFrame/HPC-GECO Workshop on Component Based High Performance Computing*, 2008.

[34] Benjamin A. Allan, Boyana Norris, Wael R. Elwasif, and Robert C. Armstrong, Managing Scientific Software Complexity with Bocca and CCA, *Scientific Programming* **16**, 315 (2008).

[35] Wael Elwasif, Boyana Norris, Benjamin A. Allan, and Robert Armstrong, Bocca: A Development Environment for HPC Components, in *Proceedings of HPC-GECO/CompFrame'07, October 21-22, 2007, Montreal, Québec, Canada*, pages 21–30, ACM, 2007.

[36] Benjamin A. Allan and Boyana Norris, Automating Multilanguage Development Processes for the High-Performance Software Life-Cycle, in *PARA 2008-9th International Workshop on State-of-the-Art in Scientific and Parallel Computing*, Trondheim, Norway, 2008, (in press).

[37] Steven Parker, Kostadin Damevski, Ayla Khan, Ashwin Deepak Swaminathan, and Chris Johnson, The SCIJump Framework for Parallel and Distributed Scientific Computing, in M. Parashar, X. Li, and S. Chandra, editors, *Advanced Computational Infrastructures for Parallel/Distributed Adaptive Applications*, Wiley Press, 2007.

[38] Madhusudhan Govindaraju, Michael J. Lewis, and Kenneth Chiu, Design and Implementation Issues for Distributed CCA Framework Interoperability, *Concurrency and Computation: Practice and Experience* **19**, 651 (2007).

[39] Rajdeep Bhowmik, Chaitali Gupta, Madhusudhan Govindaraju, and Aneesh Aggarwal, McGrid: Framework for Optimizing Grid Middleware on Multi-Core Processors, in *SOCP '07: Proceedings of the 2007 Workshop on Service-Oriented Computing Performance: Aspects, Issues, and Approaches*, pages 9–16, New York, NY, USA, 2007, ACM.

[40] Kostadin Damevski, Ashwin Deepak Swaminathan, and Steven Parker, CCALoop: Scalable Design of a Distributed Component Framework, in *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, pages 213–214, New York, NY, USA, 2007, ACM.

[41] Kostadin Damevski, Ashwin Deepak Swaminathan, and Steven Parker, Highly Scalable Distributed Component Framework for Scientific Computing, in *Proceedings of the 3rd International Conference on High Performance Computing and Communication*, Houston, Texas, 2007.

[42] Kostadin Damevski, Keming Zhang, and Steven Parker, Practical Parallel Remote Method Invocation for the Babel Compiler, in *Proceedings of the 1st Joint HPC-GECO/CompFrame Workshop*, Montreal, Canada, 2007, ACM.

[43] Nanbor Wang, Rooparani Pundaleeka, and Johan Carlsson, Distributed Components for Integrating Large-Scale HPC Applications, in *HPC-GECO/CompFrame 2007, 21–22 October, Montreal, Quebec, Canada*, 2007.

[44] Masha Sosonkina, Dane Coffey, Fang Liu, and Randall Bramley, Hierarchical Usability Levels for Sparse Linear System Solver Components, in *Proceedings of the 2008 CompFrame/HPC-GECO Workshop on Component Based High Performance Computing*, 2008.

[45] Masha Sosonkina, Fang Liu, and Randall Bramley, Usability Levels for Sparse Linear Algebra Components, *Concurrency and Computation: Practice and Experience* **20**, 1439 (2007).

[46] Fang Liu and Randall Bramley, CCA-LISI: On Designing A CCA Parallel Sparse Linear Solver Interface, in *21th International Parallel and Distributed Processing Symposium (IPDPS 2007), Proceedings, 26-30 March 2007, Long Beach, California, USA*, pages 1–10, IEEE, 2007.

[47] Fang Liu, Masha Sosonkina, and Randall Bramley, A HPC Sparse Solver Interface for Scalable Multilevel Methods, in *High Performance Computing and Simulation Symposium (HPC 2009) , Part of 2009 Spring Simulation Multiconference (SpringSim'09)*, 2009, (in press).

[48] Everest T. Ong, J. Walter Larson, Boyana Norris, Robert L. Jacob, Michael Tobis, and Michael Steder, A Multilingual Programming Model for Coupled Systems, *International Journal for Multiscale Computational Engineering* **6**, 39 (2008).

[49] J. Walter Larson and Boyana Norris, Component Specification for Parallel Coupling Infrastructure, in O. Gervasi and M. L. Gavrilova, editors, *Proceedings of the International Conference on Computational Science and its Applications (ICCSA 2007)*, volume 4707 of *Lecture Notes in Computer Science*, pages 56–68, Springer-Verlag, 2007.

[50] E. T. Ong, J. W. Larson, B. Norris, R. L. Jacob, M. Tobis, and M. Steder, Multilingual Interfaces for Coupling in Multiphysics and Multiscale Systems, in *Proceedings of the International Conference on Computational Science, Beijing, China, May 27–30, 2007, Proceedings, Part I*, volume 4487/2007 of *Lecture Notes in Computer Science*, pages 931–938, 2007.

[51] E. T. Ong, J. W. Larson, B. Norris, R. L. Jacob, M. Tobis, and M. Steder, A Multilingual Programming Model for Coupled Systems, *International Journal of Multiscale Computational Engineering* **6**, 39 (2008).

[52] J. W. Larson, Graphical Notation for Diagramming Coupling Workflows for Multiphysics and Multiscale Systems, in *Proceedings of the 9th International Conference on Computational Science (ICCS 09)*, Lecture Notes on Computer Science, Springer, 2009, (in press).

[53] Shang chieh Wu and Alan Sussman, Taking Advantage of Collective Operation Semantics for Loosely Coupled Simulations, in *Proceedings of the 21st International Parallel & Distributed Processing Symposium*, IEEE Computer Society Press, 2007.

[54] David E. Bernholdt, The Common Component Architecture: Building Frameworks for Computational Science, invited talk, MODEST-7c: Multi-Scale, Multi-Physics Software Frameworks (in MODEST and Elsewhere), 2006.

[55] David E. Bernholdt, The Common Component Architecture: Building Frameworks for Computational Science, invited talk, US-Japan Workshop on Integrated Simulation of Fusion Plasmas, 2007.

[56] David E. Bernholdt, The Role of Component Software Technology in Meeting the Challenge of Petascale Scientific Simulation, seminar, PetroBras CENPES (Research Center), Brazil, 2007.

[57] Valmor de Almeida, David E. Bernholdt, Doug Dechow, and Wael Elwasif, Integrated Simulation using the Common Component Architecture, poster, Computational Science and Engineering Conference (CESC) 2007, 2007.

[58] Benjamin A. Allan, Developments in the Common Component Architecture, in *International Conference on Computational Methods 2007*, Hiroshima, Japan, 2007.

[59] Benjamin A. Allan, Developments in the Common Component Architecture, in *International Workshop for Large-Scale Coupled Simulations*, Tokyo, Japan, 2007.

[60] S. Shasharina, N. Wang, S. Muazala, and R. Pundaleeka, Grid and Component Technologies in Physics Applications, in *International Conference on Accelerator and Large Experimental Physics Control Systems (ICALPCS) 2007*, 2007.

[61] David E. Bernholdt, Component Architectures in the Next Generation of Ultrascale Scientific Computing: Challenges and Opportunities, in *HPC-GECO/CompFrame 2007, 21–22 October, Montreal, Quebec, Canada*, 2007.

[62] Kostadin Damevski and Hui Chen, Automated Provenance Collection for CCA Component Assemblies, in *Proceedings of the 9th International Conference on Computational Science (ICCS 09)*, Lecture Notes on Computer Science, Springer, 2009, (in press).

[63] Shang chieh Wu, *Flexible and Efficient Control of Data Transfers for Loosely Coupled Components*, PhD thesis, University of Maryland, 2008.

[64] Ashwin Deepak Swaminathan, CCALoop - A Scalable Distributed Component Framework for Scientific Computing, Master's thesis, University of Utah, 2008.

[65] Kostadin Damevski, *Component Model Interoperability for Scientific Computing*, PhD thesis, University of Utah, 2006.

[66] Douglas R. Dechow, Boyana Norris, and James Amundson, The Common Component Architecture for Particle Accelerator Simulations, in *Proceedings of HPC-GECO/CompFrame'07, October 21-22, 2007, Montreal, Québec, Canada*, ACM, 2007.

[67] J. F. Amundson, D. Dechow, L. McInnes, B. Norris, P. Spentzouris, and P. Stoltz, Multiscale, multi-physics beam dynamics framework design and applications, in *Journal of Physics: Conference Series 125*, volume 125, 2008.

[68] H. N. Najm, J. Ray, M. Valorani, F. Creta, and D. A. Goussis, A Computational Facility for Reacting Flow Science, *Journal of Physics: Conference Series* **46**, 53 (2006).

[69] Svetlana Shasharina, John R. Cary, Ammar Hakim, Gregory R. Werner, Scott Kruger, and Alex Pletzer, FACETS: A Physics Driven Parallel Component Framework, in *Proceedings of the 2008 CompFrame/HPC-GECO Workshop on Component Based High Performance Computing*, 2008.

[70] Thomas Panas, Thomas Epperly, Daniel J. Quinlan, Andreas Saebjornsen, and Richard W. Vuduc, Communicating Software Architecture using a Unified Single-View Visualization, in *12th International Conference on Engineering of Complex Computer Systems (ICECCS 2007), 10-14 July 2007, Auckland, New Zealand*, pages 217–228, IEEE Computer Society, 2007.

C Additional Non-TASCS References

[71] U.S. Department of Energy, Archive of SciDAC Discovery Highlights, `http://www.scidac.gov/highlights/hilites_archive.html`, 2009.

[72] U.S. Department of Energy, Workshops on Scientific Computing at the Extreme Scale (climate, high energy physics, nuclear physics, fusion), `http://www.sc.doe.gov/ascr/WorkshopsConferences/WorkshopsConferences.html`, 2009.

[73] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, ACM Press, New York, 1999.

[74] Benjamin A. Allan, Robert Armstrong, David E. Bernholdt, Felipe Bertrand, Kenneth Chiu, Tamara L. Dahlgren, Kostadin Damevski, Wael R. Elwasif, Thomas G. W. Epperly, Madhusudhan Govindaraju, Daniel S. Katz, James A. Kohl, Manoj Krishnan, Gary Kumfert, J. Walter Larson, Sophia Lefantzi, Michael J. Lewis, Allen D. Malony, Lois C. McInnes, Jarek Nieplocha, Boyana Norris, Steven G. Parker, Jaideep Ray, Sameer Shende, Theresa L. Windus, and Shujia Zhou, A Component Architecture for High-Performance Scientific Computing, *Intl. J. High-Perf. Computing Appl.* **20**, 163 (2006).

[75] J. R. Cary, J. Candy, R. H. Cohen, S. Krasheninnikov, D. C. McCune, D. J. Estep, J. Larson, A. D. Malony, A. Pankin, P. H. Worley, J. A. Carlsson, A. H. Hakim, P. Hamill, S. Kruger, M. Miah, S. Muzsala, A. Pletzer, S. Shasharina, D. Wade-Stein, N. Wang, S. Balay, L. McInnes, H. Zhang, T. Casper, L. Diachin, T. Epperly, T. D. Rognlien, M. R. Fahey, J. Cobb, A. Morris, S. Shende, G. W. Hammett, K. Indireshkumar, D. Stotler, and A. Yu Pigarov, First Results from Core-Edge Parallel Composition in the FACETS Project, in *Journal of Physics: Conference Series 125*, 2008.

[76] Cosmin Safta, Jaideep Ray, and Habib N. Najm, Performance of a High-order Projection Scheme for AMR Computations of Chemically Reacting Flows, in *SIAM 12th International Conference on Numerical Combustion*, Monetrey, CA, 2008.

[77] Cosmin Safta, Jaideep Ray, and Habib N. Najm, A High-Order Projection Scheme for AMR Computations of Chemically Reacting Flows, in *SIAM Conference on Computational Science and Engineering*, Miami, FL, 2009.

[78] J. P. Kenny, C. L. Janssen, M. S. Gordon, M. Sosonkina, and T. L. Windus, A Component Approach to Collaborative Scientific Software Development: Tools and Techniques Utilized by the Quantum Chemistry Science Application Partnership, *Scientific Computing* **16**, 287 (2008).

[79] J. P. Kenny, C. L. Janssen, E. F. Valeev, and T. L. Windus, Components for Integral Evaluation in Quantum Chemistry, *J. Computat. Chem.* **29**, 562 (2008).

[80] T. P. Gulabani, M. Sosonkina, M. S. Gordona, C. L. Janssen, J. P. Kenny, H. Netzloff, and T. L. Windus, Development of High Performance Scientific Components for Interoperability of Computing Packages, in *High Performance Computing Symposium (HPC 2009), part of 2009 Spring Simulation Multiconference (SpringSim'09), San Diego, California, USA*, 2009.

[81] Joseph P. Kenny, Steven J. Benson, Yuri Alexeev, Jason Sarich, Curtis L. Janssen, Lois Curfman Mcinnes, Manojkumar Krishnan, Jarek Nieplocha, Elizabeth Jurrus, Carl Fahlstrom, and Theresa L. Windus, Component-Based Integration of Chemistry and Optimization Software, *J.Computat. Chem.* **24**, 1717 (2004).

[82] C. L. Janssen, J. P. Kenny, I. M. B. Nielsen, M. Krishnan, V. Gurumoorthi, E. F. Valeev, and T. L. Windus, Enabling New Capabilities and Insights from Quantum Chemistry by Using Component Architectures, *Journal of Physics: Conference Series* **46**, 220 (2006).

[83] Theresa Windus Manojkumar Krishnan, Yuri Alexeev and Jarek Nieplocha, Multilevel Parallelism in Computational Chemistry using Common Component Architecture and Global Arrays, in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, 2005.

[84] L. McInnes, J. Ray, R. Armstrong, T. Dahlgren, A. Malony, B. Norris, S. Shende, J. Kenny, and J. Steensland, Computational Quality of Service for Scientific CCA Applications: Composition, Substitution, and Reconfiguration, Technical Report ANL/MCS-P1326-0206, Argonne National Laboratory, 2006.

[85] Boyana Norris, Jaideep Ray, Rob Armstrong, Lois C. McInnes, David E. Bernholdt, Wael R. Elwasif, Allen D. Malony, and Sameer Shende, Computational Quality of Service for Scientific Components, in Ivica Crnkovic, Judith A. Stafford, Heinz W. Schmidt, and Kurt Wallnau, editors, *Proceedings of the International Symposium on Component-Based Software Engineering (CBSE7)*, volume 3054 of *Lecture Notes in Computer Science*, pages 264–271, Edinburgh, Scotland, 2004, Springer, (also available as Argonne preprint ANL/MCS-P1131-0304),.

[86] J. Steensland and J. Ray, A Partitioner-Centric Model for SAMR Partitioning Trade-Off Optimization: Part I, *International Journal of High Performance Computing Applications* **19**, 409 (2005).

[87] J. Steensland and J. Ray, A Partitioner-Centric Model for SAMR Partitioning Trade-Off Optimization: Part II, in *Proceedings of the 6th International Workshop on High Performance Scientific and Engineering Computing (HPSEC-04)*, 2004.

[88] H. Johansson and J. Steensland, A Performance Characterization of Load Balancing Algorithms for Parallel SAMR Applications, Technical Report Technical Report 2006-047, Department of Information Technology, Uppsala University, 2006.

[89] H. Johansson, Performance Characterization and Evaluation of Parallel PDE Solvers, Licenciate thesis, Department of Information Technology, Uppsala University, 2006.

[90] H. Johansson, Design and Implementation of a Dynamic and Adaptive Meta-Partitioner for Parallel SAMR Grid Hierarchies, Technical Report Technical Report 2008-017, Department of Information Technology, Uppsala University, 2008.

[91] J. Knap, N. R. Barton, R. D. Hornung, A. Arsenlis, R. Becker, and D. R. Jefferson, Adaptive Sampling in Hierarchical Simulation, *International Journal for Numerical Methods in Engineering* **76**, 572 (2008).

[92] Nathan R. Barton, Jaroslaw Knap, Athanasios Arsenlis, Richard Becker, Richard D. Hornung, and David R. Jefferson, Embedded Polycrystal Plasticity and Adaptive Sampling, *International Journal of Plasticity* **24**, 242 (2008).

[93] Joel V. Bernier, Nathan R. Barton, and Jaroslaw Knap, Polycrystal Plasticity Based Predictions of Strain Localization in Metal Forming, *Journal of Engineering Materials and Technology* **130**, 021020 (2008).

[94] OpenMI Home Page, 2009.

[95] Scott Peckham, CSDMS Handbook of Concepts and Protocols: A Guide for Code Contributors, 2009.