

High-Performance Scientific Component Research: Accomplishments and Future Directions

SciDAC Center for Component Technology for Terascale Simulation Software (CCTSS)

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

Rob Armstrong, Sandia National Laboratories, Lead PI

rob@sandia.gov

1 Introduction and Background

The Common Component Architecture (CCA) Forum began in 1998 as a grass-roots effort to fundamentally change the status quo for developing and using scientific software. The Forum is founded on the recognition that burgeoning complexity in large-scale computations increasingly hampers scientific progress, and on the vision that component based software engineering (CBSE) can be adapted to effectively mitigate this trend. This vision was not without controversy. Before the CCA, there was little precedent in CBSE for factors important in high-performance scientific computing, including parallelism in the single program multiple data (SPMD) style, suitably efficient inter-component communication, support for Fortran, support for a variety of high-performance architectures (including one-of-a-kind, leadership-class machines), and a tenable migration strategy for large legacy codes. With SciDAC funding since 2001, we have demonstrated the efficacy of our approach in a wide range of computational science domains, grown a diverse community of users, and furthered scientific research.

Here we briefly describe the CCA effort and the structure of the CCTSS project. We then present some of our accomplishments under SciDAC sponsorship and the opportunities we see to further enhance component-based high-performance scientific computing. Finally, we offer several suggestions for modifications to the SciDAC program that would facilitate our vision of *bringing component technology to high-performance computing*.

The Case for Components in High-Performance Scientific Computing. To date, software for high-end scientific computing has been primarily the province of a relatively small number of “hero” programmers. However, as both science and computer technology have advanced, the complexity of both the scientific problems and the computers on which the simulations are performed have grown tremendously, necessarily resulting in more complex software. If this advancement is to continue, the increasingly untenable “hero” programmer approach must be replaced by more collaborative software development. Moreover, the need to simulate multiple physical phenomena at multiple spatio-temporal scales makes software development increasingly interdisciplinary, intensifying the challenges of collaboration.

CBSE has emerged in business and Internet software as an important approach to address software complexity. Component approaches focus on carefully defining the interactions among different parts, or components, of a software system. Apart from these specific interactions, components are kept isolated from each other by a component environment. Components provide a means of structuring software that benefits both the developers and users of software. This approach lowers the barrier to developer participation and can simplify collaborations among research teams by allowing each group to focus on their interests and expertise. CBSE aids in creating software that is more readily reused across applications, and it enables the

automation of complicated tasks. Components can also help the performance and portability of applications through compositional optimization and other approaches.

CCA in the SciDAC Program. The Common Component Architecture effort is the embodiment of a long-range program of research and development into the formulation, roles, and use of component technologies in high-performance scientific computing. This effort fits well into the SciDAC program, which has a complementary goal of fundamentally changing the way computational science is done. Through world-class *computer* science research, we develop tools and techniques that help scientists carry out world-class *computational* science research.

The SciDAC ISIC program has provided the CCA with an unusual opportunity, by allowing us to formulate a comprehensive effort to bring the CCA through its formative stages. Our work covers four areas, which combine innovative research in computer science and useful implementations of those ideas:

- Formal specification of the Common Component Architecture and a reference implementation¹ of the associated environment and tools, allowing researchers to *use* the CCA in their software development.
- Creation of a freely available component “toolkit” of broad utility in scientific computing, to facilitate adoption of the CCA.
- Development of techniques and tools for the parallel data exchange required to couple previously separate applications into integrated multiscale and multiphysics simulations.
- User outreach and applications integration. This work includes educational activities, such as talks and tutorials, as well as interactions with users to aid in their evaluation and adoption of the CCA. Focused CCTTSS efforts in global climate modeling and quantum chemistry provide the opportunity to study the use of component technology in two scientific areas of particular interest.

Our work under current SciDAC sponsorship is establishing a basic, robust capability for CBSE in scientific software development. We look forward to extending this work to provide component-based applications with a richer, more sophisticated, and more dynamic environment, so that we can help computational scientists take full advantage of the next generation of leadership-class computers.

2 CCTTSS Accomplishments

We have improved the approach and productivity of HPC applications. CCA technology is intended to change the way scientific software is developed and used, and to improve the productivity of both developers and users. Early-adopters of the CCA have realized these benefits. For example, the SciDAC Center for Reacting Flow Science (CFRFS; H. Najm, PI) built an extensive simulation capability using CCA components. Recently, they found the CCA approach greatly increased their productivity when incorporating cutting-edge discretization ideas into their code.²

Emphasizing the importance of community-defined, domain-specific “common” software interfaces to increase the sharing and reuse of scientific codes is a hallmark of CBSE. The CCA is a catalyst for such efforts in several application domains (fusion, climate, chemistry, and combustion). The Scientific Interface Definition Language (SIDL) is being used for interface standardization in areas such as meshing software (SciDAC TSTT center; J. Glimm, PI), linear and nonlinear equations solvers (SciDAC TOPS center; D. Keyes, PI), and computational chemistry.

We have enabled the merging of software from disparate groups into large-scale scientific simulations. Another key benefit of component technology is that it facilitates the development of new appli-

¹B. Allan, et al., “The CCA Core Specification in a Distributed Memory SPMD Framework”, *Concurrency and Computation: Practice and Experience*, 14(5) pp. 1–23, 2002.

²S. Lefantzi, et al., “A Component-based Toolkit for Simulating Reacting Flows with High Order Spatial Discretizations on Structured Adaptively Refined Meshes,” *Progress in Computational Fluid Dynamics*, 2005 (to appear).

cations from codes that have been independently developed by different groups. For example, the Center for Reacting Flow Science has used the CCA to incorporate the GrACE adaptive mesh refinement library, CVODE integration package, TAU performance toolkit, and legacy Sandia code into sophisticated combustion simulations.^{3,4} In addition, quantum chemists at Pacific Northwest and Sandia National Laboratories, in collaboration with applied mathematicians at Argonne, have used the CCA to create a tool for molecular geometry optimization from the NWChem and MPQC parallel computational chemistry packages, the TAO optimization library, and the linear algebra capabilities of the Global Array and PETSc libraries.⁵ Efforts in the global climate modeling community have also used the CCA to integrate models of different physical processes into coupled simulations.⁶

We have developed reusable scientific components and the tools with which to use them. In addition to developing simple component examples and hands-on exercises as part of CCA tutorial materials, we are growing a CCA toolkit of components that is based on widely used software packages, including: ARMCi (one-sided messaging), CUMULVS (visualization and parallel data redistribution), CVODE (integrators), DRA (parallel I/O), Epetra (sparse linear solvers), Global Arrays (parallel programming), GrACE (structured adaptive meshes), netCDF and parallel netCDF (input/output), TAO (optimization), TAU (performance measurement), and TOPS (linear and nonlinear solvers). Many of these components are used in the scientific applications mentioned above. Toolkit components were publically released in August 2003 and November 2004. We continue to add components and make them more accessible to CCA users.

We have created an effective HPC component model, reference implementations, and a language interoperability tool. CCA technology is being used or evaluated in a wide range of application areas, including: biomedical engineering, climate modeling, combustion simulation, computational chemistry, fusion energy, large-scale visualization, materials science and nanoscience, mesoscale severe storm prediction, real-time and near-real-time data collection and processing, testing and modeling of military vehicles, and underground radionuclide transport. Our language interoperability tool, Babel,⁷ makes CCA components interoperable across languages *and* CCA frameworks. Numerous studies have demonstrated that the overheads of the CCA environment are small and easily amortized in typical scientific applications.

We have achieved recognition and influence in the HPC and CS communities by bringing state-of-the-art computer science research into multidisciplinary scientific applications. CCTTSS members have published over 60 documents since the beginning of the SciDAC program in 2001, including invited papers and works that received Best Paper awards at conferences. CCTTSS members have given over 41 presentations (we have not cataloged our presentations in detail, so this is significantly under-reported) and have been invited to give keynote addresses at workshops and conferences. The CCA is also influencing the next generation of computational scientists: CCA software and methodology have been incorporated into university courses, and more than 20 graduate students and 8 post-docs have participated in CCA-related work at CCTTSS institutions.

The CCA language interoperability requirement has affected the course of future Fortran standards. The Chasm array-interoperability API used in Babel was accepted by the J3 standard committee to be part of the next Fortran standard following Fortran 2003. The design of the CCA has also influenced the Business Process Execution Language (BPEL), a web services work-flow specification language used by

³S. Lefantzi, et al., "Using the Common Component Architecture to Design High Performance Scientific Simulation Codes," Proceedings of the International Parallel and Distributed Processing Symposium, April 2003, Nice, France

⁴J. Ray, et al., "Performance Measurement and Modeling of Component Applications in a High Performance Computing Environment: A Case Study," 18th International Parallel and Distributed Computing Symposium, April 26-30, 2004, Santa Fe, NM.

⁵J. Kenny, et al., "Component-Based Integration of Chemistry and Optimization Software," Journal of Computational Chemistry, 25(14) pp. 1717-1725, 2004.

⁶J. Larson, et al., "Components, the Common Component Architecture, and the Climate/Weather/Ocean Community," Proceedings of the 84th American Meteorological Society Annual Meeting, 11-15 January, 2004, Seattle, WA.

⁷Dahlgren, et al., "Babel User's Guide", Lawrence Livermore National Laboratory, Jan. 2005.

Microsoft, IBM, BEA, SAP, and other major corporations. Finally, companies including Cray, Fluent, IBM, MSC.Software, and Tech-X have expressed interest in or are using CCA.

3 Future Research Directions in Component-Based Scientific Computing

While we have made significant progress under the SciDAC program, much remains to be done to realize the full potential of component technology in high-performance scientific computing. In this section, we describe some of the key elements of our near-term vision for the CCA.

The CCA Forum must continue to work with domain scientists to strengthen the benefits that component technology brings to their research software. There is no doubt that coming generations of computers will be more complex and diverse, that the software running on them will increase in sophistication and fidelity, and that groups developing and running the software will be larger and increasingly interdisciplinary. These are the same issues that motivated the development of component technology, and we foresee an *increasing* need for the CCA. Our work includes education and direct interaction with users, support for and enhancement of the CCA environment, and making the components and the component approach more accessible to users. Component technology in the context of high-performance scientific computing is still fairly early in its development. It offers a rich array of opportunities for computer science research that could further strengthen the benefits component technology brings to large-scale scientific computing.

We plan to continue to examine, refine, and enhance the component architecture to make it easier to use and more powerful. One major area of opportunity involves the specification of components. The current *syntactic* specification, using SIDL, can be extended to capture more of the *semantics* of component behavior. For example, increasing the expressiveness of component specifications (the “metadata” available about them) makes it possible to catch certain types of errors automatically when a software component is used incorrectly and to warn the user when a component is used outside the bounds for which it was designed and tested.⁸

A second major area of research for the CCA will be developing generalized methodologies and tools for simulations that couple multiple kinds and scales of physics. There is a trend in computational science research toward simulations that require federations of codes representing multiple kinds and scales of physics. This work will extend CCTTSS research in parallel data redistribution and parallel remote method invocation,⁹ which constitute some of the fundamental building blocks to support generalized coupling. This work will also be informed by experiences in new research initiatives, such as fusion simulation and multiscale mathematics.

In addition, we must leverage the unique capabilities of component technology to inspire new CS research directions. For example, the CCA provides a dynamic model for components, allowing them to be connected and disconnected *during* execution. This model allows an application to monitor and adapt itself by swapping components for others. This approach, called *computational quality of service*, can benefit numerical,¹⁰ performance,^{4,11} and other aspects of software. Forthcoming petascale computers will require higher levels of parallelism than are currently available. Enhanced component specifications can provide copious information that parallel runtime environments could exploit to provide the utmost performance.

⁸T. Dahlgren and P. Devanbu, “Adaptable Assertion Checking for Scientific Software Components,” Proc. of the 1st International Workshop on Software Engineering for High Performance Computing System Applications, Edinburgh, Scotland, 2004.

⁹F. Bertrand, et al., “Data Redistribution and Remote Method Invocation in Parallel Component Architectures,” Proc. of IPDPS 2005, to appear (best paper award).

¹⁰P. Hovland, et al., “A Quality-of-Service Architecture for High-Performance Numerical Components,” Proceedings of the Workshop on QoS in Component-Based Software Engineering, Toulouse, France, June 20, 2003.

¹¹B. Norris, et al., “Computational Quality of Service for Scientific Components,” Proceedings of the International Symposium on Component-Based Software Engineering (CBSE7), Edinburgh, Scotland, 2004

Moreover, the development and use of new runtime environments could be simplified by integrating them with component frameworks.

4 Taking the SciDAC Program to the Next Level

The SciDAC program has been successful in bringing together teams of researchers to produce new world-class science. Many researchers (including us) speak effusively about the way the SciDAC program has changed the way they do research. But the program may not yet have realized the full vision originally articulated. SciDAC researchers have achieved much, but in large measure they do it as small groups of hero programmers rather than as integrated communities. We feel that an emphasis on community building would be a valuable addition to a SciDAC follow-on. Below we offer specific suggestions and impacts with this theme.

Enable research on a scale that engages an entire scientific community. The current SciDAC program brings together groups of researchers to tackle large-scale scientific problems. It is unreasonable to suppose that the SciDAC program would ever be funded at a level capable of directly supporting entire communities, but by emphasizing scientific problems of appropriate magnitude *and* a community-oriented approach to the software, it is possible to engage even unfunded researchers in the scientific challenge. As in open-source software development, SciDAC-supported researchers should be encouraged to make their simulation software available to the larger scientific community, which will encourage those outside of the SciDAC program to do likewise.

A community-based approach to the development of scientific codes offers tremendous opportunities to leverage the efforts of others, while allowing researchers to focus even more strongly on the elements of the problems of interest to them. However, this approach also poses significant challenges. Developing an effective software community requires an investment in the software – defining key functionality, agreeing on interfaces and behavior, and adapting existing software. Such investments will pay off for the community and for DOE as scientific challenges are addressed more quickly and easily.

Encourage and support domain scientists to work closely with computer scientists and applied mathematicians. Our experience is that many of the domain scientists involved in the current SciDAC program feel that new scientific results are their most important deliverable. As a result, many projects take a short-range view of their software, focusing on what is needed to achieve the next science result and often neglecting software investments that would pay off over a longer time frame. Increasing the importance and visibility of the software itself as a valuable product of the research, along with the science it enables, will encourage domain scientists to collaborate more extensively with mathematics and computer science researchers to make those investments in their software.

At the same time, both sides need financial support for these kinds of interactions, so that they do not come “at the cost of the science.” Unfortunately, opportunities for collaboration can be hard to predict at the time a long-term project is proposed. We suggest separate, modest funding be available (with a relatively short lead time) to projects that find specific opportunities for collaboration. Funded collaborations should be focused and involve high levels of interaction among a few specific people in the collaborating groups.

Provide stronger support for collaboration, especially for community-based software development. While many tools facilitate collaborations among geographically separated researchers, SciDAC projects are left on their own to discover, deploy, use, and maintain them. It would be more economical and productive for collaboration servers to be centrally supported, providing all SciDAC projects with uniform electronic collaboration capability. It might even be beneficial to employ an “evangelist”, tasked with helping groups unfamiliar with the newly developed tools to understand how to take advantage of them.