

**The Center for Component Technology for Terascale
Software Simulation (CCTTSS) at Indiana University.
*Final Report.***

DE-FC02-01ER25492 A002

PI: Dennis Gannon
Co-PI: Randall Bramley
Department of Computer Science
Indiana University
Bloomington, Indiana
812 855-4510
gannon@cs.indiana.edu
bramley@cs.indiana.edu

1. Introduction

Indiana University-Bloomington (IUB) research in CCTTSS has concentrated on four areas:

1. **XCAT.** Composing applications where the components may be distributed across the wide area network. In collaboration and coordination with the University of Utah and SUNY Binghamton University, we have shown the CCA concept works very well in this context. We researched and developed a bridge between CCA components and emerging Grid computing standards, particularly web services. Building CCA interfaces to web services leverages emerging standards in location-independence of codes, access to distributed data management systems, and to on-line sensors and instruments.
2. **Proteus.** To support the bridge between CCA technology and the Web service world, it was essential that we examine the low level communication layer. As Web services are based on XML and the SOAP standard, it was essential that we consider the performance and implementation details of these protocols in the context of scientific computing.
3. **DCA.** The third major area of work involves the problem of connecting parallel components across a wide area network. For example one component may be running on a 1000 node machine at Oak Ridge while the other component may be running on a 789 node machine at Los Alamos. Connecting these two components together in parallel is what we call the MxN problem. We have had substantial success on this problem.
4. **Component Toolkit.** Our fourth area of work involves the help we have provided to the development of the first CCA component toolkit. We have created a single component interface for sparse linear solvers including PetSC, Trilinos and SPLib. We also developed a CCA interface for a parallel IO library.

IUB has also collaborated with the CCA Forum in developing and presenting CCA tutorials, and provided systems support and HPC cluster accounts in support of those tutorials. Several students have been encouraged by the work to seek internships at DOE labs, and two Ph.D. graduate students are now faculty at SUNY Binghamton, where they continue to collaborate with DOE researchers on projects. A total of five students have completed dissertations on CCA-related work, and another two will complete theirs in 2007.

2. Technology Results

XCAT

The current industry standard for composing software into large applications is based on Web Services standards being developed by IBM, Microsoft, HP, Sun through organizations like OASIS and W3C. While these standards have roots in ideas first

proposed by CCA, they are focused more on coupling services over the internet than on the CCA goal of coupling code modules in a single program for a massively parallel computer. One goal of the XCAT subproject was to turn the tables: what can CCA learn from all this progress in Web Services?

The XCAT3 framework provides an implementation of the CCA specification for distributed components on the Grid using web services. Making CCA components compatible with Grid standards enables access using protocols and clients shared by other useful Grid services, apart from adding desirable features such as multiple level naming and dynamic service introspection as specified by the Grid specifications. Furthermore, complex applications on the Grid can be conveniently composed by using composition techniques prescribed by CCA.

In XCAT3 every component is modeled as a set of Grid web services - every Provides port is a Grid web service, every Uses port is a Grid web service client.

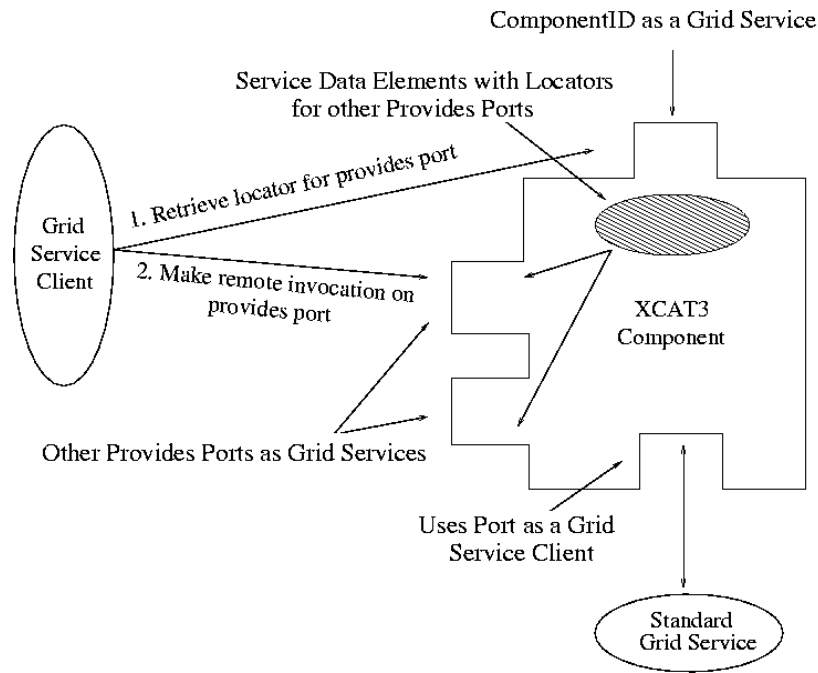


Figure 1. This illustration show how a standard Grid web service client can interact with XCAT3 components

We use Babel to generate the interfaces for the framework for compatibility with the CCA specification. In addition, XCAT3 has been used as a research vehicle to investigate (a) distributed user-level checkpointing for components, to be used for fault tolerance, and (b) migration of individual components to adapt to dynamic Grid environments. To this end, the framework provides APIs for component writers to load and store their states to stable storage, and algorithms that enable storing a consistent global state into stable storage for distributed checkpointing, and that enable storing individual component states into stable storage for migration purposes. This work

resulted in a Ph.D. thesis for Sriram Krishnan who is now employed at the San Diego Supercomputing Center.

To test XCAT3 in a real application we are working with a large collaboration of meteorologists and atmospheric scientists to study mesoscale storms such as tornadoes and hurricanes. We have constructed a component composition tool which allows the user to compose weather data analysis and simulation tools by connecting boxes with a standard “drop and drag” programming model.

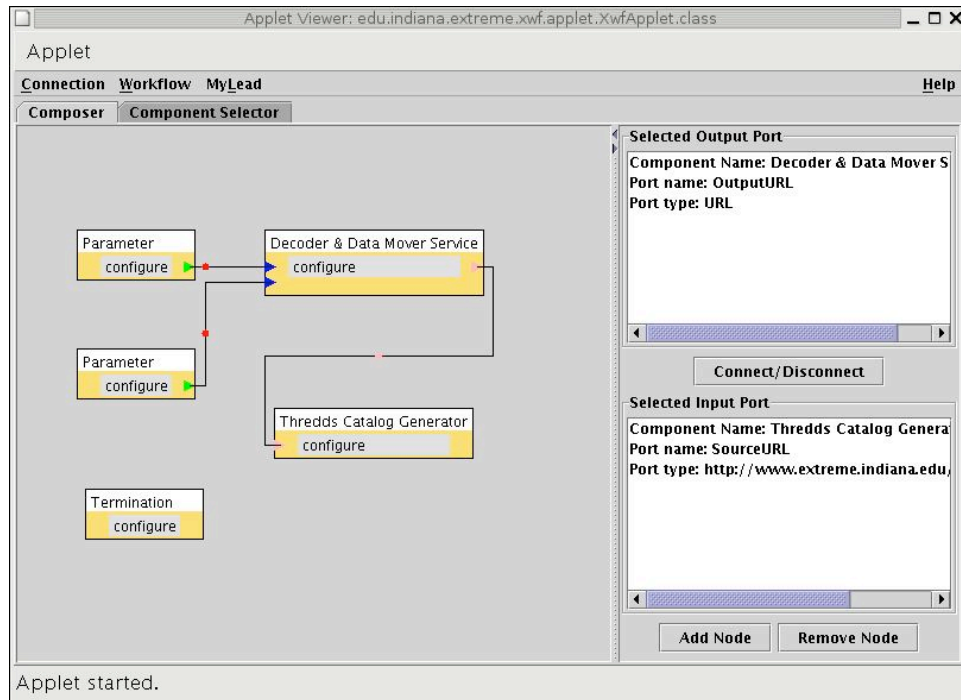


Figure 2. XCAT version of LEAD portal composer

The resulting composed application is run on a distributed Grid of compute and data servers. The result was demonstrated at SC 2004 in Pittsburgh.

Proteus

We have designed, prototyped and measured a multiprotocol, web-services invocation framework for CCA components. Development was divided into three fronts. The first front is a C++-based SOAP implementation with a dynamic invocation interface. The second front is providing a protocol-independent Web services invocation toolkit. The third front is on Babel/Proteus integration.

SOAP Implementations

Proteus can use an existing SOAP implementation, but we were not able to find a suitable existing implementation. The primary requirement is that the SOAP implementation have a dynamic invocation interface (DII). This allows Proteus to pass a service-

independent invocation object to the SOAP implementation, which can then directly invoke the service. Without a DII, stubs and skeletons for the SOAP implementation would have to be generated in a separate step, in addition to Proteus stubs and skeletons. The two leading contenders, gSOAP and Axis C++ both do not currently have a DII.

Thus, we have developed a SOAP implementation, and made significant progress on that front. We currently have a SOAP implementation that has demonstrated interoperability with XSOAP Java under Proteus for a subset of XML Schema types. We also worked on HTTP 1.1 improvements such as chunking and persistent connections.

WSIT

A number of independent but related factors have led us to the realization that we should factor out all Web services invocation classes into a Web services invocation toolkit. This includes the Proteus stub and skeleton design, the Proteus generic serialization framework, and our WSDL-to-Proteus-Invocation-Model code generator. This left us with three significant bodies of code. Proteus focused on the multiprotocol mediation aspects. XSOAP is the SOAP implementation. WSIT handles all invocation aspects. WSIT and XSOAP can be used independently without Proteus.

The primary motivation behind this refactoring is the realization that there is significant utility to a stand-alone SOAP implementation, for those situations where multiprotocol capability is not necessary. To achieve this independence in a practical manner, however, it is necessary that the invocation aspects of Proteus be factored out, so that these can be used with XSOAP without the other aspects of Proteus.

A secondary motivation is that other people may wish to use our WSIT with their own SOAP implementation (or other protocol implementation, such as a binary XML implementation). This refactoring involved modifying XSOAP to use Proteus invocation patterns and interfaces, work that is substantially complete.

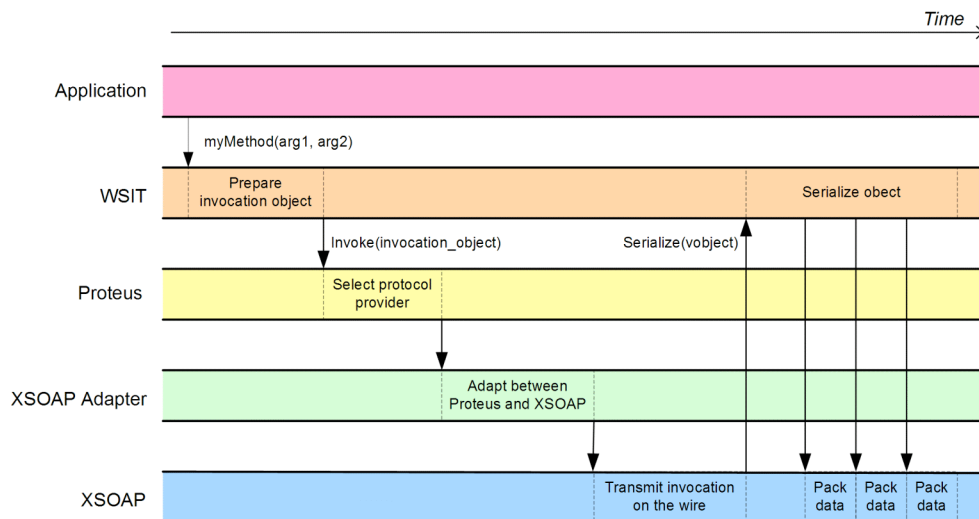


Figure 1 This diagram shows how an invocation progresses through the layers from an application down to the transport layer, and the relationships among WSIT, Proteus, and XSOAP.

Babel

We have also researched integrating Babel with Proteus. Babel provides language interoperability, but has only recently begun to explore distributed communication capabilities. By integrating Babel with Proteus, we provided a path for distributed framework interoperability.

We completed a prototype that demonstrated how Babel with Proteus can provide Babel with RMI capabilities (see Figure 2). This prototype uses hand-written glue code. We are now in the process of modifying Babel so that it will generate the necessary glue code automatically.

Legion

We have also researched a design for integrating Proteus into Legion/CCA. Since Legion/CCA does not use Babel, distributed interoperability with Legion requires that Legion use SOAP. Integrating with Proteus allows them to negotiate and then switch to an efficient shared binary protocol. An initial design and preliminary prototype have been carried out.

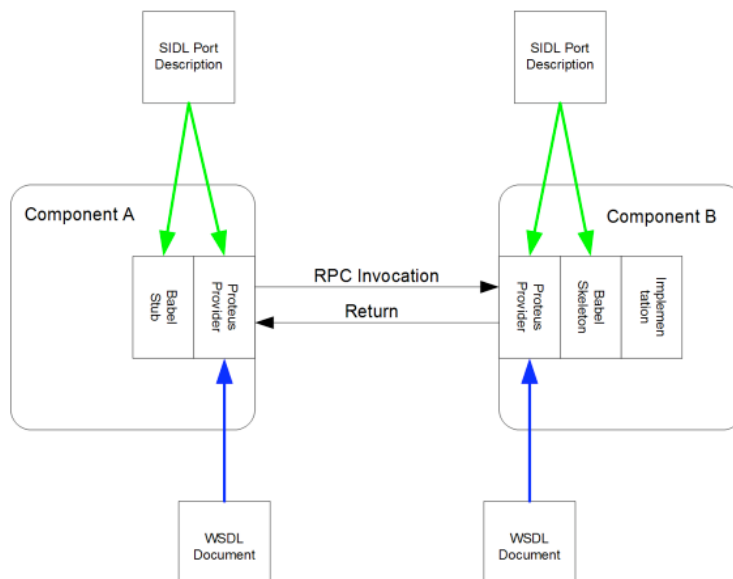


Illustration 2 Bridging between Babel and Proteus. Green lines are compile-time inputs, blue lines are run-time inputs. At run-time, only the binding and addressing information is used from the WSDL document.

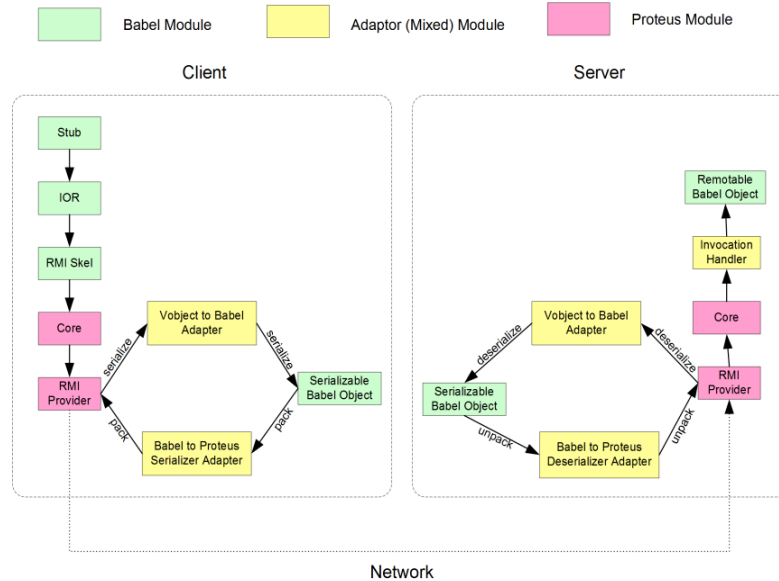


Figure 3. Proteus is a multiprotocol Remote Method Invocation (RMI) library designed to plugged into larger systems. Proteus has been integrated with the Babel language interoperability package at Lawrence Livermore National Laboratory. Babel RMI allows one process to call another process on a different machine. The RMI call proceeds through the Babel implementation layers to an RMI skeleton which packages the invocation in a form suitable for the Proteus multiprotocol library. The Proteus layer transmits the invocation across the network to the server. On the server side Proteus delivers the invocation to the server-side IOR, which then calls the actual user implementation.

Lessons Learned and Further Directions

Our experience with XCAT has provided valuable information about using distributed scientific applications within a CCA simulation. While the CCA model of directly coupling components extends to the distributed case, there are some fundamental differences in the nature of the computations that must be accounted for. First, in the wide-area distributed case security is a very big factor. The messages that travel between components should be encrypted and digitally signed. Authentication and Authorization is an important issue. To solve this problem we have incorporated web service security mechanisms into our system. In doing so we have discovered several important new ways to optimize security for our distributed applications. We have now published these

results in the security community and in the high performance distributed computing conference.

Second, we observe that in the distributed case, the world is more asynchronous and subject to failure. The trend in web services is to communicate by asynchronous messages rather than remote procedure call. Components have to be designed with much greater fault tolerance in mind. Consequently, components need to be as stateless as possible. One essential part of any distributed system is a mechanism that allows components to publish notification events that any other component can subscribe to.

A main effort for XCAT has been to show how to extend the concept of event notification to a “message bus” that can be made a standard part of the CCA programming model. This allows a component to be programmed with a slightly different semantics than is currently possible with CCA. Currently each component has one or more provides ports and one or more uses ports. An invocation of a provides port causes the component to respond. This response may involve an invocation of a uses port on another component. In a message bus model a component can be initialized to “subscribe” to a class of messages by “topic”. It may also publish new messages onto the bus with any topic. In this way one or more components may receive the messages published by other components. This allows components to publish events, such as error conditions or task completions, which other components may need to know about. It also allows for a type of performance monitoring that is not easily accomplished with standard approaches. At the same time it provides a form of decoupling to handle the more asynchronous character of distributed computations, and increased failure recovery opportunities in that a secondary component can respond to a published event if the primary intended recipient is disconnected or fails.

This can be done completely within the CCA model: Each component would have a subscription and a publish uses port and a event listener provides port. We have implemented event-based programming paradigm with several applications and it is now a critical component of our Mesoscale Weather Prediction System LEAD and the preliminary framework for fusion energy simulations in the CSWIM project. We continue to explore other concepts from the web services model that can be valuable in the CCA context.

DCA and the Distributed MxN Problem

Indiana University has researched the problem of connecting distributed parallel programs, with different components running on different sites – and possibly with different numbers of processes. The semantics of these MxN interactions is a novel problem in scientific computing, and IUB has developed the Distributed CCA Architecture (DCA) for exploring the issues of parallel remote method invocation within the CCA context. DCA uses MPI as a run-time system because it provides distributed synchronization and communication utilities, letting us concentrate on the semantics and performance of parallel-remote method invocation. In recent years this has led to a major

expansion in how CCA is perceived and used, and we have routinely connected test applications. Currently we are working with geologists, climate models and space systems to evaluate efficiencies and what kind of semantics are most useful for these applications.

DCA provides remote method invocation between parallel components. This form of invocation is designed to resemble that of collective calls in the Single-Program Multiple-Data model of parallel programming. This is possible thanks to a flexible design based on MPI communicator groups to define the sets of participating processes in both the source and the target components.

In addition, large data objects that span multiple processes can be defined and deployed at the target component at the time of the remote invocation. To do this, the DCA has defined a special kind of "parallel" argument that is subject to data redistribution. The DCA supports several types of layouts inspired by High Performance FORTRAN, such as block, cyclic, block-cyclic, and more. Parallel arguments in our prototype implementation are restricted to one-dimensional arrays. DCA is complementary to the distributed MxN approach used by Utah's SciRUN2 by allowing more complex coupling between components.

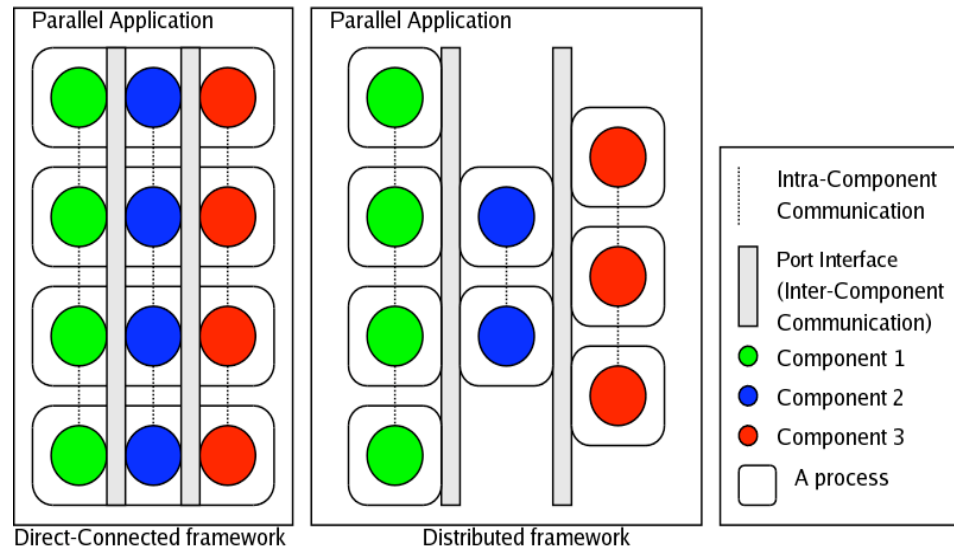


Figure 3. In the Direct-Connected framework, the coupling between components does not require network communication. The MxN problem requires coupling

An MPI-like interface provides a useful stepping stone for new users migrating to CCA, and has useful concepts of collectivity and communicator groups that can help make clearer the meaning of CCA invocations between components of different cardinalities. Redistribution of data in DCA is dynamic and “manual” (as opposed to using pre-cooked data arrangements). This is more work for the programmer, who must set the parameters for the redistribution when distributed parameters are used but is flexible and uses well

known MPI-like constructs. In the past year DCA has added

- A simplified SIDL parser that parses ports and the most argument types: 1D arrays, strings, parallel arguments. This has helped detect minor errors in the CCA SIDL specification which have been corrected this year by the Babel team.
- Basic framework implementation: BuilderServices, setServices, register/use ports, goPorts, etc. all implemented using MPI.
- MxN parallel remote method invocation ports.
- Automatic data redistribution for parallel arguments.
- DCA services not required by CCA: metadata management, logging, monitoring, and basic dynamic application steering.

The DCA uses MPI communicator groups to establish process participation during remote, collective port method calls: the component making the call can decide which processes participate. On the receiving side, however, all processes must participate: this is not a shortcoming of DCA, but is inherent in the uses-provides design pattern that CCA requires.

In the past year the DCA has been implemented and has been tested on the large-scale scheduling of scientific instruments on a low earth orbit satellite.

Note: an overview of the entire CCA research effort on the NxM problem won the “best software track paper” award at the 2005 International Parallel and Distributed Processing Symposium.

CCA Component Toolkit

A crucial part of the CCTTSS effort is building a toolkit of interfaces that are mapped to HPC codes. IU’s research for this includes parallel I/O, sparse linear solvers, and a metadata management system. Each of these is already a Ph.D. dissertation topic for a graduate student.

Parallel I/O is increasingly a bottleneck operation as the scale of both simulations, visualization, and analysis codes has grown. Currently most hard drives can write at the rate of 100 Mbyte/second, while data soon will be generated at the rate of petabytes per second. Parallelism is the only way to resolve this, and IU is building a parallel I/O library interface and implementation. MPI-2 provides parallel I/O, but the interface is complex including dozens of functions and fairly sophisticated ideas dealing with shared pointers, collective calls, etc. Many application users simply need to read and write distributed arrays, and in collaboration with ORNL and PNL we have defined a “distributed array descriptor” (DAD) interface that succinctly describes how an array is distributed across processors. By using a DAD object, a distributed array can be read or written using a simple function call with only 3-4 arguments. As with the DCA, an MPI-like idiom has been adopted to ease the transition for end-applications users. A parallel

I/O capability is present in Global Arrays (GA), and recently ANL has developed a parallel netCDF interface. GA requires that the array be allocated and used with the GA library, and for many existing applications converting all arrays to use the GA package is a large amount of work. The IU parallel I/O library is intended to work with any array that has a DAD, and so adding the capability to existing MPI programs is straightforward. In tests on a 128 node Opteron cluster, the I/O performance has only been limited by the speed of the interconnect network, not the hard drives.

Sparse linear solvers are an important kernel for most HPC codes, and the DOE has supported highly effective, reliable, and scalable libraries providing them. In collaboration with ANL and Sandia we have defined a single interface that allows users to exchange a Petsc solver for a Trilinos one, or to switch to using the sparse direct solver SuperLU. Currently using one or the other of those libraries alone tends to “lock-in” an application, and the new interface encourages rapid prototyping and experimentation from an application. Furthermore the solvers have complementary strengths and weaknesses so combining them in a single interface provides important utility. The Linear Solver Interface (LISI) is a CCA interface which has been developed in collaboration with other CCA researchers, presented both at the CCA forum and other conferences, and appears to be likely to become the single interface desired. It has been tested with Petsc, Trilinos, SPLIB, and Sparskit solvers and works well.

A third toolkit component is Obsidian, a set of components that lets users rapidly build a metadata management system for an application. Unlike most other metadata managers, Obsidian does not impose a particular schema and so allows flexibility in creating metadata management that is particularly efficient and useful for a given application. Obsidian includes utilities for creating a metadata database, maintaining logical relationships between objects, tracking data, and archiving it. This system has been used in a wide variety of applications over the past year, including job management in bioinformatics, x-ray crystallography at remote sites, clinical radiation therapy, and long term photometric studies in astronomy. In crystallography over 100 Gbytes of data has been managed in a single month, and the software has been distributed and used in five different laboratories both on and off campus. For bioinformatics a researcher needed to build a database of genome-genome and protein-protein similarities; this required over 50,000 parallel jobs to create and subsequently requires about 250 jobs a month to maintain (each time GeneBank adds a genome or protein entry, all pairwise interactions need to be computed with it and existing data). Handling such a large number of jobs running on different platforms and with different batch managers requires automated relaunch capability and complex decisions to determine whether or not a job has “failed”. Tracking this information and the subsequent data has been done with Obsidian. This year we have just started the collaboration with a fully automated (unattended) astronomy observatory. Over twelve years of observations have been accumulated and each night up to 150 new ones are generated. Astronomers are interested in mining this data to search for quasars and variable stars, and the first step is to build systematic data management into it. Obsidian is not a competitor with the OGSA-DAI interface or SDSC’s SRB/MCAT, instead constituting a higher-level architecture which can use those as components. We intend to extend Obsidian to use for combined simulations in

the fusion simulation project as part of the CSWIM effort.

3. People Supported

The people supported by this project include

- Dennis Gannon, PI
- Randall Bramley, PI
- Kenneth Chiu, Postdoc – now on the faculty at SUNY Binghamton University.
- Felipe Bertrand, Graduate Research Assistant. Ph.D. completed 2006
- Sriram Krishnan, Graduate Research Assistant. Ph.D. completed 2005 – now at San Diego Supercomputing Center.
- Alek Slominski, Graduate Research Assistant. Ph.D. expected 2007
- Yogesh Simmhan, Graduate Research Assistant. Ph.D. expected 2007
- Madhu Govindaraju, Ph.D. 2005 – now on the faculty of SUNY Binghamton University.
- Wei Liu, Graduate Research Assistant
- Yongquan (Cathy) Yuan, Graduate Research Assistant. Ph.D. 2006.
- Fang (Cherry) Liu, Graduate Research Assistant
- Yu (Maried) Ma, Graduate Research Assistant. Ph.D. 2006.
- Al Rossi, Graduate Research Assistant. M.S. in CS, 2004.
- Rachana Ananthakrishnan, Graduate Research Assistant. M.S. in CS, 2004
- Lavanya Ramakrishnan, Graduate Research Assistant. M.S. in CS, 2004
- Caroline Olariu, visiting Ph.D. student from University of Versailles, France
- Nicolas Rey-Cenvaz, visiting Ph.D. student from University of Versailles, France
- Raul Indurkar, Graduate Research Assistant. M.S. in CS, 2004

4. Publications.

1. Rachana Ananthakrishnan, Sriram Krishnan, Madhusudhan Govindaraju, Lavanya Ramakrishnan, Aleksander Slominski, Dennis Gannon, Grid Web Services and Application Factories,' Chapter 9 in "Grid Computing: Making the Global Infrastructure a Reality" Fox, Berman, Hey, Eds., Wiley 2003.
2. D. E. Bernholdt, B. A. Allan, R. Armstrong, F. Bertrand, K. Chiu, T. L. Dahlgren, K. Damevski, W.R. Elwasif, T. G. W. Epperly, M. Govindaraju, D. S. Katz, J. A. Kohl, M. Krishnan, G. Kumfert, J. W. Larson, S. Lefantzi, M. J. Lewis, A. D. Malony, L. C. McInnes, J. Nieplocha, B. Norris, S. G. Parker, J. Ray, S. Shende, T. L. Windus, S. Zhou. A Component Architecture for High-Performance Scientific Computing. Intl. J. High-Perf. Computing Appl., 2005 in ACTS Collection special issue.
3. Felipe Bertrand, David E. Bernholdt, Randall Bramley, Kostadin B. Damevski, James A. Kohl, Jay W. Larson, and Alan Sussman. Data redistribution and remote method invocation in parallel component architectures. In Proceedings of the 19th

- International Parallel and Distributed Processing Symposium: IPDPS 2005, 2005.
Best Paper Award.
4. Felipe Bertrand, Randall Bramley, Kostadin B. Damevski, James A. Kohl, David E. Bernholdt, Jay W. Larson, and Alan Sussman, Data Redistribution and Remote Method Invocation in Parallel Component Architectures, *Best Paper of the Software Track* by the International Parallel and Distributed Processing Symposium, Denver, CO, 4-8 April 2005.
 5. Felipe Bertrand, Yongquan Yuan and Kenneth Chiu and Randall Bramley, "An Approach to Parallel MxN Communication", Proceedings of the Los Alamos Computer Science Institute (LACSI) Symposium, Oct, 2003.
 6. Felipe Bertrand, Yongquan Yuan, Kenneth Chiu, Randall Bramley, An Approach to Parallel MxN Communication, International Journal of High Performance Computing Applications, volume 19, Number 4, Winter 2005, pp 399-407. (Item 5 above is a conference version of this paper.)
 7. Felipe Bertrand, Data Redistribution and Remote Method Invocation in Parallel Component Architectures, IUB Ph.D. dissertation, 2006.
 8. Felipe Bertrand, Randall Bramley, "DCA: A distributed CCA framework based on MPI", Proceedings of the 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS'04), IEEE Press pp. 80-89. 2004.
 9. Randall Bramley, Rob Armstrong, Lois McInnes, High-Performance Component Software Systems, Chapter in Frontiers of Parallel Processing For Scientific Computing, M. A. Heroux, P. Raghavan and H. D. Simon, editors, SIAM series: Software, Environments, and Tools, Dec 2005.
 10. Kenneth Chiu, Tharaka Devadithya, Wei Lu, and Aleksander Slominski. A binary XML for scientific applications. In e-Science 2005, 2005.
 11. Kenneth Chiu, "XBS: A Streaming Binary Serializer for High Performance Computing", Proceedings of the High Performance Computing Symposium 2004, April 2004
 12. Kenneth Chiu, Madhusudhan Govindaraju, and Dennis Gannon, "The Proteus Multiprotocol Library," In Proceedings of the 2002 Conference on Supercomputing, November 2002.
 13. Kenneth Chiu, An Architecture for Concurrent, Peer-to-Peer Components, IUB Ph.D. dissertation, 2001.
 14. L. Fang and D. Gannon, "XCAP - An Extensible Capability-based Authorization Infrastructure for Grids", 4th Annual PKI R&D Workshop: Multiple Paths to Trust, NIST Gaithersburg, MD April 19-21, 2005
 15. D. Gannon, J. Alameda, O. Chipara, M. Christie, V. Duple, L. Fang, M. Farrellee, S. Hampton, G. Kandaswamy, D. Kodeboyina, S. Krishnan, C. Moad, M. Pierce, B. Plale, A. Rossi, Y. Simmhan, A. Sarangi, A. Slominski, S. Shirasuna, T. Thomas, Building Grid Portal Applications from a Web-Service Component Architecture, Proceedings of the IEEE. Special Issue on Grid Technology, Vol. 93, no. 3, pp. 551-563, March 2005
 16. D. Gannon, S. Krishnan, L. Fang, G. Kandaswamy, Y. Simmhan, A. Slominski, On Building Parallel and Grid Applications: Component Technology and Distributed Services, Proceedings, Challenges of Large Applications in

- Distributed Environments (CLADE) In conjunction with the 13th International Symposium on High Performance Distributed Computing (HPDC-13), pp. 44-51, June, 2004. (to appear in a special issue of Cluster Computing 2005.)
17. D. Gannon, S. Krishnan, A. Slominski, G. Kandaswamy, L. Fang, "Building Applications from a Web Service based Component Architecture", in In V. Getov and T. Kielmann, editors, Component Models and Systems for Grid Applications. Proc. of the Workshop on Component Models and Systems for Grid Applications, June 26, 2004 held in Saint Malo, France. Springer, 2005, ISBN: 0-387-23351-2.
 18. D. Gannon, L. Fang, G. Kandaswamy, D. Kodeboyina, S. Krishnan, B. Plale, A. Slominski, "Building Grid Applications and Portals: An Approach Based on Components, Web Services and Workflow Tools", Invited Keynote Paper, EuroPar, Pisa Italy, August 2004.
 19. Dennis Gannon, Randall Bramley, Geoffrey Fox, Shava Smallen, Al Rossi, Rachana Ananthakrishnan, Felipe Bertrand, Ken Chiu, Matt Farrellee, Madhu Govindaraju, Sriram Krishnan, Lavanya Ramakrishnan, Yogesh Simmhan, Alek Slominski, Yu Ma, Caroline Olariu, Nicolas Rey-Cenvaz, "Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications," Journal of Cluster Computing, 5(3): 325-336 (2002)
 20. Dennis Gannon, "Software Component Technology for High Performance Parallel and Grid Computing", invited keynote paper. Euro-Par 2001 Springer Verlag. 2001
 21. Madhusudhan Govindaraju, Sriram Krishnan, Kenneth Chiu, Aleksander Slominski, Dennis Gannon, Randall Bramley, "Merging the CCA Component Model with the OGSF Framework", to appear, Proceedings of CCGrid-2003, Tokyo March 2003.
 22. S. Krishnan and D. Gannon, "Checkpoint and Restart for Distributed Components in XCAT3", the 5th IEEE/ACM International Workshop on Grid Computing, Pittsburgh, Nov. 8, 2004. pp. 281-288.
 23. Sriram Krishnan, Dennis Gannon, "XCAT3: A Framework for CCA Components as OGSA Services", HIPS 2004, 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments, IEEE Computer Society Press, April 26, 2004
 24. S. Krishnan, R. Bramley, M. Govindaraju, R. Indurkar, A. Slominski, D. Gannon, J. Alameda, D. Alkaire "The XCAT Science Portal," Proceedings SC2001, Nov. 2001, Denver. An extended version to appear in a special issue of the Journal of Scientific Computing, 2003.
 25. Sriram Krishnan, An Architecture for Checkpointing and Migration of Distributed Components on the Grid, IUB Ph.D. dissertation, 2004.
 26. Fang Liu, Randall Bramley, CCA-LISI: On Designing A CCA Parallel Sparse Linear Solver Interface, Proc. of the IEEE International Parallel and Distributed Processing Symposium, 2007.
 27. Wei Lu, Kenneth Chiu, and Dennis Gannon. Building a generic SOAP framework over binary XML. In The 15th IEEE International Symposium on High Performance Distributed Computing (HPDC-15), June 2006.

28. Wei Lu, Kenneth Chiu, and Yinfei Pan. A parallel approach to XML parsing. In The 7th IEEE/ACM International Conference on Grid Computing, Barcelona, September 2006.
29. Yu Ma and Randall Bramley, "A Composable Data Management Architecture for Scientific Applications", Proceedings of Challenges of Large Applications in Distributed Environments 2005, July.
30. Yu (Marie) Ma, A Composable Data Management Architecture for Scientific Applications, IUB Ph.D. dissertation, 2006.
31. Srinath Perera and Dennis Gannon. Web Service Extensions for Scientific Workflows. In HPDC2006 Workshop on Workflows in Support of Large-Scale Science (WORKS06), Paris, France, June 2006.
32. Lavanya Ramakrishnan, Helen Rehn, Jay Alameda, Rachana Ananthakrishnan, Madhusudhan Govindaraju, Aleksander Slominski, Kay Connelly, Von Welch, Dennis Gannon, Randall Bramley, Shawn Hampton, "An Authorization Framework for a Grid Based Component Architecture," Grid2002 Workshop at SC2002, Baltimore Oct. 2002.
33. Shirasuna S., Slominski A., Fang L., and Gannon D., Performance Comparison of Security Mechanisms for Grid Services, the 5th IEEE/ACM International Workshop on Grid Computing, Pittsburgh, Nov. 8, 2004. pp. 360-364.