DOE Award # FC02-07ER25812 Final Technical Report
Center for Technology for Advanced Scientific Component Software (TASCS)
University of Maryland
PI: Alan Sussman

Progress Summary

A new version of InterComm, version 2.0, was created and then released in September 2009 (http://www.cs.umd.edu/projects/hpsl/chaos/ResearchAreas/ic/), with subsequent updates, that provides all the functionality of previous versions for coupling components in parallel. InterComm 2.0 provides the convenient programming model that separates information about the programs to be run for a coupled application and how to couple between programs, using an externally specified XML Job Description (XJD). In addition, InterComm 2.0 provides dynamic timestamp matching on each connection, to further decouple simulation components. In that model, each component specifies a timestamp for each export or import operation, and InterComm finds the best match for each import, based on a user-specified policy in the XJD file for that connection. This capability is very important in connecting simulation components that use different time steps, so that a custom time step matching algorithm does not have to be implemented. The matching can be specified through the InterComm XJD file for a particular coupling, with InterComm performing the matching according to the policy specified for that coupling.

Correct SIDL (Scientific Interface Description Language) interfaces for the InterComm 2.0 API were built and compiled with the LLNL Babel compiler. We also built several examples with multiple parallel components, and ran them in the CCaffeine framework. In one scenario, each parallel component is run in a separate CCaffeine instance, and makes calls to the InterComm components to export or import data to a parallel component in a different instance of the framework. The connection between the CCaffeine instances is specified in an InterComm XJD file, so in this scenario InterComm is used to move data between parallel (or sequential) components running in different CCaffeine instances. However, we have also designed a second scenario to run multiple parallel components within a single CCaffeine instance, even though a more comprehensive solution really requires a distributed and parallel CCA framework to provide the full InterComm functionality (which was intended to be part of the work of the TASCS MCMD working group, which we have participated in, but was never completed because of the early termination of the TASCS funding).

We succeeded in building several CCA components that both encapsulate the basic InterComm functionality and that run within a single CCaffeine instance with parallel components. In addition to a set of three components that implement the basic InterComm data transfer capabilities, the single framework approach requires an additional *Manager* component that we implemented, to control startup and set up connections between the individual parallel (or sequential) application components that will use the InterComm component data transfer services. One key issue that had to be addressed, in addition to design and implementation of the Manager component, was thread safety issues within CCaffeine. CCaffeine runs all components on a single host machine as threads within one process. Since InterComm uses PVM as its underlying data transport mechanism (support for MPI as the communication mechanism between components via the LLNL

PnMPI framework, https://computation-rnd.llnl.gov/pnmpi/, was completed after the end of the TASCS project), and PVM is *not* thread-safe, the solution is that the Manager component ensures that different components run on disjoint subsets of processes. As stated earlier, a CCA framework implementation that supports both parallel and distributed components would be a more flexible solution to this problem, but is beyond the scope of this project.

We worked with space scientists through the NSF Center for Integrated Space Weather Modeling (*http:www.bu.edu/cism*). The space weather modelers are using InterComm as a standalone runtime library, but were educated about the benefits of a CCA environment, and showed interest in using CCA tools and techniques. However, that work was never performed because of the early termination of the TASCS project.

We also worked with Manish Parashar of Rutgers University on incorporating InterComm and CCA functionality into the support libraries for the SciDAC Center for Plasma Edge Simulation (CPES) codes, but that work required significant modifications to both the Rutgers and Maryland software tools, and was not completed by the end of the TASCS project.

Students Supported

One graduate student each year of the project

Students Graduated

Shang-chieh Wu, Ph.D. Computer Science, 2008
C.H. Afzal, M.S. Computer Science, 2009

Publications and Talks

Publications

- C.H. Afzal, "High Performance Communication Between Parallel Components with CCA and CCaffeine", masters degree paper, University of Maryland, August 2009.
- S.-C. Wu, "Flexible and Efficient Control of Data Transfers for Loosely Coupled Components", Ph.D. dissertation, University of Maryland, May 2008.
- A. Sussman. "Building Complex Coupled Physical Simulations on the Grid with InterComm", *Engineering with Computers* special issue on frameworks for scalable scientific and engineering applications, Vol. 22, 2007.
- J.S.-C. Wu and A. Sussman. "Taking Advantage of Collective Operation Semantics for Loosely Coupled Simulations", *Proceedings of the 21st International Parallel & Distributed Processing Symposium (IPDPS 2007)*, March 2007.

Web Site

For the InterComm project:
http://www.cs.umd.edu/projects/hpsl/chaos/ResearchAreas/ic/
For the InterComm CCA components:
http://www.cs.umd.edu/projects/hpsl/chaos/ResearchAreas/ic/dist/dist-cca.htm
The Maryland work is included in the web site for the overall TASCS center at
http://tascs-scidac.org/