The Technology and Infrastructure That Enable the CCA

CCTTSS Leads: Rob Armstrong (SNL, PI), Jim Kohl (ORNL, co-PI and Lead for Parallel Data Redistribution), Gary Kumfert (LLNL, co-PI and Lead for Frameworks); **Chasm:** Craig Rasmussen (LANL, co-PI), Matt Sottile (LANL); **Babel:** Tamara Dahlgren (LLNL), Tom Epperly (LLNL), Gary Kumfert (LLNL); **Ccaffeine:** Ben Allan (SNL), Rob Armstrong (SNL); **XCAT:** Dennis Gannon (Indiana Univ., co-PI); **SCIRun:** Steven Parker (Univ. Utah, co-PI); **MxN:** David Bernholdt (ORNL, co-PI), Randy Bramley (Indiana Univ.), Jim Kohl (ORNL), Jay Larson (ANL), Steven Parker (Univ. Utah)

Summary

The Center for Component Technology for Terascale Simulation Software (CCTTSS) faces unique technical challenges in bringing the benefits of component technology – well established in commercial computing – to the scientific computing community. Component technology broadens the scope of software's use, thereby reducing the need for tedious software customization or rewrites and increasing computational scientists' productivity.

The very nature of terascale computing is beyond the scope of commercial component systems. CCTTSS must address multiple architectures and operating systems (unlike Microsoft COM), multiple languages (unlike Java Beans), high performance in process interconnects (unlike CORBA), and support for massive parallelism, Fortran77/90/95, and Grid connectivity (unlike all the above). The CCTTSS is simultaneously conducting novel research and producing practical end-user tools in the areas of language interoperability, parallel data redistribution, and frameworks.

Language Interoperability

SciDAC applications are often written in Fortran90/95, whereas for SciDAC math and CS libraries C/C++ is more common. The technical difficulties in connecting the two often preclude effective sharing. We currently have two tools that cover the depth and breadth of the language interoperability problem: Chasm and Babel.

Chasm is an automatic wrapping tool to connect Fortran90/95 and C++. It generates wrappings implicitly by scanning existing source code. Chasm also has advanced capabilities in dealing with Fortran90 array descriptors – a particularly pernicious barrier to language interoperability.

Babel is an IDL based tool that currently connects C, C++, Fortran77/90/95, Python, and Java. Babel's approach is more scalable in the number of languages that it can support, but requires an explicit listing of what to wrap expressed in SIDL (Scientific Interface Definition Language). The CCA specification itself is written exclusively in SIDL.

These two tools with their differing approaches complement each other effectively. Chasm is applying its Fortran90 and C++ automation techniques to produce SIDL. Babel is adopting Chasm's Fortran90 array infrastructure to augment its own Fortran90 bindings.

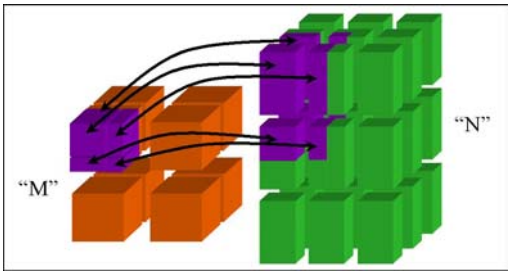

Fig 1. Parallel data redistributed between 4 and 9 processors

MxN Parallel Data Redistribution

A common barrier to scientific software reuse is a disparate underlying model for how data is distributed in memory by different codes. This problem is exacerbated when combining massively parallel codes and even more so when the two codes are separated on differing numbers of processors. This most general case is referred to as the "MxN problem" and illustrated in the figure above.

Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

In traditional software development, such external coupling of parallel data arrays has been a painful and ad hoc activity. The CCA will make generalized parallel model coupling a reality. MxN tools provide the basic functions for exchanging parallel data, including registering the data, creating connections with parallel "communication schedules", and then actually transferring parallel data elements among components.

MxN tools are currently implemented as components, built upon existing tools for interacting with parallel simulations. The CUMULVS system for interactive visualization, computational steering, and application fault-tolerance, has been wrapped as one component. The PAWS model coupling system has been wrapped as another.

SciDAC funding of the CCTTSS has motivated – and CCA components have enabled – these two disparate communication models from CUMULVS and PAWS to be generalized into a common interface specification that subsumes the capabilities of both systems. To incorporate a variety of structured and unstructured mesh support, this research has progressed in cooperation with the CCA scientific data working group. Extensions are planned for particle data, adaptive meshes, and sparse arrays.

Frameworks

CCA frameworks serve two purposes: (1) to simplify the task of composing applications from CCA components, and (2) to provide for and enforce the behavior ensuring interoperability that is laid out by the CCA specification. Frameworks manage components: their creation, connection, and eventual cleanup. The CCA specification not only admits parallel components for high-performance computing, but also allows CCA components to be distributed across remote Grid-enabled computers.

The CCTTSS supports three frameworks, each with its own specialties and purpose. Linkages between these frameworks provide a full complement of features to work with numerous application scenarios. Ccaffeine is the production CCTTSS framework best positioned for most terascale simulations. SCIRun and XCAT are research vehicles that are being used to prototype new ideas and push the CCA architecture in new directions.

The Ccaffeine framework focuses on high-performance parallel components and provides a simple, single-threaded environment that uses MPI or PVM for interprocess communication. The XCAT framework is both CCA and Grid compliant, emphasizing distributed components on the Grid. These two frameworks function in a complementary fashion, such that Ccaffeine can be used to compose and run a parallel high-performance application, and XCAT can provide the connectivity from CCA to the rest of the planet to allow execution as a Grid resource. A combustion application demonstrated at SC02 exploited precisely this capability. SCIRun is a third framework that combines aspects of the other two and is evolving to provide connections to other component-based systems. It supports parallel components like Ccaffeine but regards them as a single distributed component, and it also supports distributed components similar to XCAT.

Tying It All Together: An Example

An ambitious future CCA capability and example of continued CCTTSS development is support for Parallel Remote Method Invocations (PRMI), or the coordination of method invocations among parallel components. Babel is incorporating serial RMI capabilities from XCAT and SCIRun, thereby empowering Ccaffeine with distributed capabilities as well. The CCA specification will expand to include interconnect standards, assuring framework interoperability across the Grid. MxN will integrate its parallel data exchange capabilities from a component into the framework. Then, the framework being able to transparently handle parallel data redistribution without the explicit coordination of the user becomes a possibility.

Contact Information:
Rob Armstrong (PI)
Phone: (925) 294-2470
Email: rob@sandia.gov
http://www.cca-forum.org

Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.