UCRL-TR-235910

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Petascale Simulation Initiative Tech Base: FY2007 Final Report

J. May, R. Chen, D. Jefferson, J. Leek, I. Kaplan, J. Tannahill

October 29, 2007

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Petascale Simulation Initiative Tech Base: FY2007 Final Report

John May (PI) Robert Chen David Jefferson
James Leek Ian Kaplan John Tannahill

1 Introduction

The Petascale Simulation Initiative began as an LDRD project in the middle of Fiscal Year 2004. The goal of the project was to develop techniques to allow large-scale scientific simulation applications to better exploit the massive parallelism that will come with computers running at petaflops per second. One of the major products of this work was the design and prototype implementation of a programming model and a runtime system that lets applications extend data-parallel applications to use task parallelism. By adopting task parallelism, applications can use processing resources more flexibly, exploit multiple forms of parallelism, and support more sophisticated multiscale and multiphysics models.

Our programming model was originally called the Symponents Architecture but is now known as Cooperative Parallelism, and the runtime software that supports it is called Coop. (However, we sometimes refer to the programming model as Coop for brevity.) We have documented the programming model and runtime system in a submitted conference paper [1]. This report focuses on the specific accomplishments of the Cooperative Parallelism project (as we now call it) under Tech Base funding in FY2007.

Development and implementation of the model under LDRD funding alone proceeded to the point of demonstrating a large-scale materials modeling application using Coop on more than 1300 processors by the end of FY2006. Beginning in FY2007, the project received funding from both LDRD and the Computation Directorate Tech Base program. Later in the year, after the three-year term of the LDRD funding ended, the ASC program supported the project with additional funds.

The goal of the Tech Base effort was to bring Coop from a prototype to a production-ready system that a variety of LLNL users could work with. Specifically, the major tasks that we planned for the project were:

1. "Port SARS [former name of the Coop runtime system] to another LLNL platform, probably Thunder or Peloton (depending on when Peloton becomes available)."

2. “Improve SARS’s robustness and ease-of-use, and develop user documentation.”

3. “Work with LLNL code teams to help them determine how Symponents could benefit their applications.”

The original funding request was $296,000 for the year, and we eventually received $252,000. The remainder of this report describes our efforts and accomplishments for each of the goals listed above.

2 Porting the runtime system

Because the first large-scale tests of Coop were done on the MCR platform, and that machine was scheduled to be retired early in FY2007, it was apparent that Coop needed to work on other large-scale platforms so that we could continue tests on thousands of processors.

The Coop runtime system was ported successfully to both Thunder and the Peloton platforms (Atlas and Zeus). (We also completed a port of Coop to AIX platforms under separate funding.) The basic functionality was complete by the end of December 2006, but in the course of the work, we discovered some performance problems on Thunder related to copying arrays of double-precision values during remote method invocation (RMI). This was causing our test application to slow down unacceptably. With the assistance of LC staff, we eventually found and fixed the problem.

During the shake-down period for Atlas, when it was available to selected users for running large-scale tests, we conducted further scaling studies using our materials modeling application. We were able to run with as many as 4,000 processors on that system. However, various problems with both the system (in its prerelease state) and the Coop software itself prevented larger-scale runs and the gathering of detailed performance data.

A lingering issue that interfered with several of our large test runs had to do with an implementation detail in various MPI libraries. When an MPI call needs to wait for data from another process, it can either relinquish the processor and allow it to be used by other processes, or it can check for data repeatedly without releasing control of the processor. The first option is known as “blocking” and the second as “polling” or “spinning.” An MPI implementation that spins interferes with the normal operation of many Coop-based applications because it prevents other processes and threads on the same processor from making progress while one process is waiting for data. Many MPI implementations allow the user to choose between blocking and spinning through an environment variable setting. However, the default choice is often poorly documented, and several times when we have run on new platforms, our tests were hampered by MPI implementations that polled. To prevent further occurrences of the problem, we developed a self-contained test program that identifies whether an MPI implementation blocks or spins. Users can run this program before launching a large simulation to verify that MPI will behave properly. Running this test is especially valuable when preparing for large-scale runs, since it helps avoid the situation where a user has to wait a long time to receive a large allocation of processors, then begins the run only to have the application hang because of a spinning MPI implementation.

3 Robustness, ease-of-use, and documentation

Concurrently with porting Coop to new platforms, we identified many areas in the code for improvement. Our tasks included regularizing names, simplifying and improving the build system, improving the automated procedures for building and testing the code each night, running the Klocworks source-code checker and fixing the issues it found, and setting up a bug-tracking system. No single change made a dramatic difference to the stability of the software, and a few scalability issues remain, but it is safe to say that Coop is much more robust and well-tested now than it was at the beginning of the project.

Although we did a significant amount of work on documentation, the task is not yet finished. In the course of training new project members in the use and implementation of Coop, we have developed a set of briefing slides and how-to guides on Coop internals and on our development and testing procedures. We have also written a technical paper on Cooperative Parallelism for submission to a conference under LDRD funding. However, we have not yet produced a comprehensive user guide for writing programs with Coop. The source code itself is written so that the Doxygen documentation generator can read its structured comments and create an initial set of function and class descriptions. We augmented these comments substantially over the past year, so that our automatically-generated documentation is largely complete. However, we still need to complement this with a higher-level description of the strategy and process that programmers should use in developing Coop applications. Work continues on this effort, and we expect to finish in a few months.

4 Working with LLNL code teams

During FY2007, we met with a wide range of potential users and other interested parties, both inside and outside LLNL, to discuss how they might work with Cooperative Parallelism. A partial list of these contacts includes:

- Phase-field modeling project (Jim Belak, Fred Streitz, and others)
- ALE-AMR project (Alice Koniges)
- Cheetah (Larry Fried)
- ASC code teams (Burl Hall, Brian Pudliner, Brian McCandless, Rob Neely, and others)
- PSUADE (Charles Tong)

- DUNE (Bill Bateson)
- Joint Munitions Program (Rose McCallen, Bruce Watkins, and others)
- Argonne National Lab MPICH Team (Rusty Lusk, Bill Gropp, and Rajeev Thakur)
- CCA Forum and Multiple-Component, Multiple Data programming model working group (David Bernholdt, Jarek Nieplocha, and others)

All the contacts listed above have involved multiple discussions over a period of time. The work with the LLNL-based groups has been done partly in connection with our ASC and LDRD funding. (Unlike a specific development task, conversations with prospective collaborators are difficult to ascribe to a particular funding source.) We also continue to collaborate with team members from the original LDRD effort who use Coop to do materials modeling.

So far, three of these contacts have led to FY2008 funding:

- The ASC program is supporting about 1.25 FTE of effort to wrap certain calculations in codes of interest as separate components that can be called through RMI.
- The DUNE project is providing about 0.2 FTE of funding to investigate a Coop-based architecture and conduct prototyping studies.
- The Cheetah project is providing about 0.3 FTE of funding to investigate the use of Coop in various aspects of its equation-of-state calculations.

We are continuing to pursue further opportunities for funding through the Joint Munitions Program.

5 Conclusions

The Computation Directorate's support of the Cooperative Parallelism project through Tech Base funding has helped the project make a successful transition from LDRD research to an effort that is supported by three separate programs. The system is now under investigation by these programmatic partners, although it is not yet in production use. Nevertheless, Coop would not likely be in a position to receive this current level of support if the software were still in its original prototype form and if we had not been able to demonstrate portability and a significant level of scalability and robustness.

References

[1] Jefferson, D. R., Barton, N. R., Becker, R., Hornung, R. D., Knap, J., Kumfert, G., Leek, J. R., May, J., Miller, P. J., and Tannahill, J. Overview of the cooperative parallel programming model. Submitted to *IPDPS 2008*, UCRL-CONF-230029., 2007.