Parallelization Agent for Legacy Codes

Suraj C. Kothari and Simanta Mitra

Department of Computer Science
Iowa State University
Ames, IA 50011
Email: kothari(mitra)@cs.iastate.edu
Phone: (515)-294-7212
FAX: (515)-294-0258

(July 30, 1996)

Acknowledgements: We are grateful to several colleagues at Iowa State University, Ames Laboratory, and Argonne National Laboratory. Without their help and encouragement, this project would not have evolved to its present form. This work was partially supported by the Center for Advanced Technology Development at Iowa State University.
Parallelization Agent Prototype (PA-I96)

Key Features:

- *Parallelization of explicit finite difference* codes; generates machine independent SPMD code for distributed memory machines.
- *High-level knowledge* about the finite difference method is built into the tool and that knowledge is used to generate efficient parallel code.
- *Structured Parallelization Process* with two phases: (i) user assisted preprocessing of the sequential code and (ii) automatic parallelization.
- *Graphical interface* for user interaction during the preprocessing phase.
- *Minimization of communication overhead* through automatic identification of sync/exchange points and automatic reduction of the number of sync/exchange points.
- *SPMD* code is produced for parallel computing platforms with distributed memory.
- *Graphical displays of key computational characteristics* to help the user’s understanding of complex codes and their parallelization.
- *Link between sequential and parallel versions* to support debugging and future evolution of the code.

Demonstration of the capability of PA-I96 by using it to parallelize MM5:

The MM5, a meteorological code, contains about 100,000 lines with hundreds of variables. The manual parallelization of the non-hydrostatic version of MM5 took three and half years. Using the PA-I96, the parallelization can be done in two weeks.

We have created a short demonstration based on *solve1.f*, a key routine from MM5. The call processing capability of PA-I96 is demonstrated by processing subroutines calls.

To evaluate the prototype, the code generated by PA-I96 is compared with the parallel code produced by expert programmers.
Parallelization Agent framework is being developed for parallelizing many legacy codes in science and engineering. The framework also provides a logical view of the code which is useful for maintaining and reusing legacy codes in a machine independent way.

**Basic Tenets:**

1. Develop a tool for parallelizing codes based on *finite difference*, *boundary element* and *finite element* methods. Parallelization of many widely used engineering and scientific codes can be addressed by focusing on these three numerical methods.

2. New approach to parallelization: *high-level knowledge* of the numerical method is built into the tool to serve as the *roadmap*, first for preprocessing of the sequential code and later for parallelizing the code. This approach avoids some of the intractable problems encountered by general purpose parallelization tools.

3. Use a *structured process* for preprocessing of sequential code and its subsequent automatic parallelization.

4. Use a mathematical paradigm to reason about parallel mappings. A parallel mapping is represented by a pair of functions *PLACE* (for processor allocation) and *STEP* (for parallel time steps) defined on the set of computations.
The PA technology shares a common philosophy with expert systems: the general problem may be intractable, but within a well defined application domain the problem can be solved by making use of the high-level knowledge about the problem domain. As the expert systems have proved to be very useful so also we believe that the PA technology will be useful. By focusing on the finite difference, boundary element and finite element methods, the parallelization problem can be addressed for a large and important class of applications commonly encountered in science and engineering.

Specific Benefits:

- Considerable amount of time and effort will be saved large codes are to be parallelized.
- It will be possible to capitalize on the enormous investments in legacy codes. This large software base of sequential codes can be reused on parallel computers.
- Many hard-to-debug errors associated with manual parallelization will be avoided by using automation.
- The preprocessing phase in Parallelization Agent will not only facilitate parallelization but it will also be useful for maintaining the sequential code.
- A new paradigm for creating parallel codes will emerge. Programmers will write the sequential code according the standard enforced by the Parallelization Agent and simply use the Parallelization Agent to create the parallel code from the sequential code.
- The preprocessing technology from Parallelization Agent can be used to improve performance of HPF compilers by preprocessing the code before feeding it to the compiler.
- Parallelization Agent through the graphical displays can provide a logical view of the code that will enable computational scientists to use higher order reasoning as opposed to line-by line analysis of the code. While viewing the demo, computational scientist were especially excited about this feature. They felt that the logical view of the code will greatly enhance their ability to deal with large and complex codes.
These displays provide a logical view of solve1.f. This 1500-line key routine of MM5 is executed at every time step and it calls many other subroutines.

**Display 1 (Block-level view of the code):**

The Parallelization Agent determines the sync/exchange points and groups the entire code into five blocks numbered 1 to 5. The code within each block can be viewed by clicking on a specific block.

**Display 2 (View Communication screen):**

By clicking the *stencils button* in Display 1, the *View Communication screen* appears. Communication is needed initially and also at the four synch/exchange points between the code blocks. Communication at a specific synch exchange point can be observed by clicking the corresponding number from the View Communication screen.

**Display 3 (Stencils to show Communication):**

By clicking on number 3 in Display 2, we can see the communication at the synch/exchange points between code blocks 3 and 4. In this case, there are six different communication patterns. For example, the first pattern shows that each cell receives from its north and south neighbors the variables: *qca, qnia, qia*, and *qra*.

**An interaction based on the logical view**

The logical view provided by the Parallelization Agent helps the scientist to deal with the complex code. For example, the scientist can selectively activate the subroutines called during *solve1.f*. By observing the communication patterns, scientists can relate the code to their understanding of the physical phenomena. This sample display was created by activating only a subset of the routines and when it was shown to a group of atmospheric scientists, they correctly guessed that the advection is not incorporated.