Structured Distributed Parallel Programming

A Tutorial

PACT-2005, Saturday, 9/17/05

Lei Pan
NASA Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109-8099
lei.pan@jpl.nasa.gov

Abstract: As cluster computers are becoming less expensive and increasingly available, our ability to program this new generation of supercomputers is lagging behind demand. Achieving scalability and ease of programming simultaneously is a grand challenge. The state of the art for implementing large-scale high-performance scientific and engineering applications has been message passing (MP), despite the fact that MP is widely considered hard to use. Studies reveal that the two statements at the heart of MP – `Send()` and `Recv()` – change the code like the `goto` statements, thus making structured programming problematic. This reminds us of the software crisis of the 1960s that was eventually ended by the suggestion from Dijkstra to abolish the use of `goto` statements. This tutorial introduces a new methodology, called *Navigational Programming* (NavP), that is based on the use of self-migrating threads. NavP is philosophically different because it describes the quantity of interest – a distributed computation along with the small data it requires – following the movement of its locus. In contrast, all other approaches use the SPMD (single program multiple data) view to describe computations at stationary locations. Program transformations in NavP involves the insertion of migration statements (`hop()`), which, unlike the `Send()` and `Recv()`, does not change code structure. NavP puts an end to the use of "`goto`" statements and it holds the promise of structured distributed parallel programming. Performance studies show that for several important and non-trivial real-world applications, NavP programs are as fast as or faster than the corresponding MPI programs. The goal of this tutorial is two folds: (1) To present NavP to the researchers in related fields and invite discussions and new ideas. (2) To show scientists and engineers with real-world examples how efficient, scalable, and elegant NavP programs can be developed in a incremental fashion.

Level: Intermediate (assumes basic knowledge of distributed parallel computing but no experience with navigational programming).

Required experience: Knowledge of principles of distributed computing; Some experience with distributed parallel programming using MPI or other paradigms; Basic knowledge of matrix computations; Working knowledge of a programming language such as C.

Expected audience: This tutorial is for the researchers in the field of high-performance distributed systems and methodologies, the scientists and engineers who have large-scale applications to be parallelized, and the students who want to learn distributed parallel programming.

Speaker's profile: The speaker is with The Applied Cluster Computing Technologies Group at NASA Jet Propulsion Laboratory, California Institute of Technology. His research areas include self-migrating threads in high-performance parallel distributed programming, parallelizing compiler techniques, parallelization of scientific and engineering applications, analysis and modeling using finite element and boundary element methods, direct and iterative numerical solution techniques, multigrid methods, and software engineering for large-scale computer-aided engineering (CAE) systems. He is the original developer of the concept and methodology of NavP and has presented this new

paradigm in numerous international forums.