# Position Paper and Brief Announcement: An Empirical Study to Compare Two Parallel Programming Models

Lorin Hochstein
University of Maryland
Computer Science Department
College Park, MD 20742
lorin@cs.umd.edu

Victor R. Basili
University of Maryland
Computer Science Department
College Park, MD 20742
basili@cs.umd.edu

**Categories and Subject Descriptors:** D.1.3 [Programming Techniques]: Concurrent Programming—*parallel programming*

**General Terms:** Human Factors, Experimentation, Languages

**Keywords:** MPI, XMT, message-passing, PRAM, empirical study, HPC, productivity, parallel programming, effort

## 1. ABSTRACT

While there are strong beliefs within the community about whether one particular parallel programming model is easier to use than another, there has been little research to analyze these claims empirically. Currently, the most popular paradigm is message-passing, as implemented by the MPI library [1]. However, MPI is considered to be difficult for developing programs, because it forces the programmer to work at a very low level of abstraction. One alternative parallel programming model is the PRAM model, which supports fine-grained parallelism and has a substantial history of algorithmic theory [2]. It is not possible to program current parallel machines using the PRAM model because modern architectures are not designed to support such a model efficiently. However, current trends towards multicore chips suggest that large-scale, fine-grained uniform-memory-access parallel machines may soon be feasible. XMT-C is an extension of the C language that supports parallel directives to provide a PRAM-like model to the programmer. A prototype compiler exists that generates code which runs on a simulator for an XMT architecture [3].

To better understand how much benefit a PRAM-like model could provide over a message-passing model, we conducted a feasibility study in an academic setting to compare the effort required to solve a particular problem. The questions under study were: can we measure the effort in developing a program using these two programming models and can we differentiate the amount of effort for each model?

The subjects participating in the study were divided up into two groups. One group solved a problem using the MPI library in either C,C++, or Fortran, and the other group solved the problem using XMT-C. The task was to write a function to multiply a sparse matrix with a dense vector.

To obtain subjects, we leveraged existing graduate-level parallel programming courses at two different universities: University of California, Santa Barbara (UCSB), and University of Maryland (UMD). At UCSB, the students solved the problem in MPI, and at UMD, the students solved the same problem in XMT-C. The focus of the UCSB class was on developing parallel programs to run on the current generation of architectures, and the students were taught MPI, as well as other models. The focus of the UMD class was parallel algorithms in the PRAM model, and the students were taught XMT-C. The students were assigned to treatment groups by class. The study was integrated into each class, as the problem was a required assignment in each class.

Subjects kept track of their effort with a self-reported effort log. We also collected automatic effort data by instrumenting the compilers, which recorded data at each compile. We computed three effort measures: self-reported, instrumented, and combined. Self-reported effort measures are based entirely on effort logs, instrumented effort measures are based entirely on timestamps from the instrumented compilers, and combined effort measures are based on compiler timestamps when the subject is working on the instrumented machine, and self-reported effort when the subject is working off the instrumented machine.

The results of this preliminary study answer both of our questions in the positive. In this case, on average, students required less effort to solve the problem using XMT-C compared to MPI. The reduction in mean effort was approximately 50% for all three measures, which was statistically significant at the level of $p < .05$ using a t-test.

This study demonstrates that the effect of programming model on effort can be directly measured through empirical studies with human subjects. While no single study can conclusively demonstrate the advantage of one programming model over another, a series of studies examining different models and different problems can provide insights into the relative strengths of parallel programming models in different contexts. The study described above is one of a series of such studies that we are currently conducting.


*SPAA'06,* July 30–August 2, 2006, Cambridge, Massachusetts, USA.
ACM 1-59593-452-9/06/0007.

## 2. REFERENCES

[1] J. Dongarra, S. Otto, M. Snir, and D. Walker. A message passing standard for MPP and workstations. *Communications of the ACM*, 39:84–90, July 1996.

[2] J. JaJa. *An Introduction to Parallel Algorithms.* Addison-Wesley Professional, March 1992.

[3] U. Vishkin, S. Dascal, E. Berkovich, and J. Nuzman. Explicit multi-threading (XMT) bridging models for instruction parallelism. In *10th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '98)*, 1998.