Workshop on Transitioning to MultiCore (TMC 2011)

Caitlin Sadowski Jaeheon Yi
University of California at Santa Cruz
{supertri, jaeheon}@cs.ucsc.edu

Abstract

Multicore programming is both prevalent and difficult. Industry programmers deal with large amounts of legacy code and are increasingly relying on multithreading to provide scalability. For legacy systems, it may not be possible to change this programming model. The Transitioning to MultiCore (TMC) workshop is focused on tools and systems for parallel programming that are interoperable with legacy code, that minimize the annotation burden for developers, and match well with current industry practice. We solicit industry experience reports about working or unworkable examples of such tools or systems, as well as research reports.

Categories and Subject Descriptors D.1.3 [*Programming Techniques*]: Concurrent Programming–*Parallel Programming*; D.2.2 [*Software Engineering*]: Design Tools and Techniques

General Terms Design, Reliability, Human Factors

Keywords Tools, Systems, Usability, Legacy Programs

1. Background

In the early 2000s, we hit a power wall; the energy output of a chip with increased processor speed has become untenable [1]. Today, all major chip manufacturers have switched to producing computers that contain more than one CPU [12]; parallel programming has rapidly moved from a special-purpose technique to standard practice in writing scalable programs. Taking advantage of parallel processors often entails using concurrent software, where multiple threads work simultaneously. However, concurrent software suffers from concurrency-specific errors, such as data races, atomicity violations, determinism violations, and deadlocks [4, 8, 9]. Achieving parallel performance is also difficult. In fact, in previous studies which compared parallel programming models or techniques a large subset of the participants in different groups did not successfully complete a correct solution that exhibited *any* speedup (e.g. [5, 10]). Furthermore, a large survey on current development practices found that a large portion of developers have to regularly deal with multithreaded code [6].

2. Main Theme and Goals

It is clear from the above discussion that multicore programming is both prevalent and difficult. To address that difficulty, numerous programming models and systems have been proposed, including transactional memory [7, 11], revisions [3], and type systems [2]. However, industry programmers face large amounts of legacy code, and so it may not always be feasible to change the programming model.

The TMC workshop is focused on tools and systems for parallel programming that are interoperable with legacy code, that minimize the annotation burden for developers, and match well with current industry practice. We solicit industry experience reports about working or unworkable examples of such tools or systems, as well as research reports. The topics for these reports may include:

- Surveys or empirical studies focused at measuring the current state of practice for multicore programming in industry
- Field studies identifying barriers and benefits to using existing tools
- Analysis tools focused on correctness, performance, or understandability analysis of existing programs
- New programming models which are interoperable with legacy multithreaded systems

We aim to bring together industry developers and researchers who are interested in improving the current transition to multicore.

3. Participant Preparation

We accept shorter, two to four page *experience reports* focused on experiences with scalable systems used in industry, or problems with existing systems. We also accept longer, four to six page *research reports* focused on development

SPLASH'11 Companion, October 22–27, 2011, Portland, Oregon, USA.
ACM 978-1-4503-0940-0/11/10.

of new systems, tools, or ideas in the multicore space. Additionally, we accept two page *position papers* focused on proposals for improving existing systems or tools. Although this is a small new workshop, we care about reviewers returning high-quality paper reviews and have picked our program committee accordingly.

4. Activities and Format

We plan to spend the morning on paper presentations. Each presentation slot will be approximately 10-15 minutes long, followed by a five minute question period.

We plan to start the afternoon with a panel presentation, moderated by the workshop organizers, comprising a mix of industry and academic panelists who can describe some of the challenges experienced with transitioning to multicore. The goal of this panel will be to highlight issues that may not be obvious within the research community.

Afterwards, we will facilitate a group discussion about what workshop participants feel are the largest issues raised in the workshop, and any issues they feel are not adequately addressed by current research literature. First we will break into focus groups (containing approximately 4-5 people per group) for about 30-45 minutes. These groups sessions will serve as a networking event so that participants will ideally make some new connections at the workshop. We aim to have at least one industry representative in each group. Each group will come together with 2-3 specific points for future or ongoing research which we will collate on the projected screen.

5. Organizers

Caitlin Sadowski (University of California at Santa Cruz)

Jaeheon Yi (University of California at Santa Cruz)

6. Program Committee

Michael Bond (Ohio State University)

Rachel Brill (IBM Haifa Research Lab)

Sebastian Burckhardt (Microsoft Research)

Joe Devietti (University of Washington)

Eitan Farchi (IBM Haifa Research Lab)

Benedict Gaster (AMD)

Ganesh Gopalakrishnan (University of Utah)

Shan Lu (University of Wisconsin - Madison)

Shankar Pasupathy (NetApp)

Neha Rungta (NASA Ames Research Center)

Koushik Sen (University of California, Berkeley)

Konstantin Serebryany (Google)

Stephen Toub (Microsoft, Parallel Computing Platform)

References

[1] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, and K. Yelick. A view of the parallel computing landscape. *Communications of the ACM*, 52(10):56–67, 2009.

[2] R. L. Bocchino, Jr., V. S. Adve, D. Dig, S. Adve, S. Heumann, R. Komuravelli, J. Overbey, P. Simmons, H. Sung, and M. Vakilian. A type and effect system for Deterministic Parallel Java. Technical Report UIUCDCS-R-2009-3032, Department of Computer Science, University of Illinois at Urbana-Champaign, 2009.

[3] S. Burckhardt, A. Baldassion, and D. Leijen. Concurrent programming with revisions and isolation types. In *Symposium on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, 2010.

[4] S. Choi and E. Lewis. A study of common pitfalls in simple multi-threaded programs. *ACM SIGCSE Bulletin*, 32(1):329, 2000.

[5] K. Ebcioglu, V. Sarkar, T. El-Ghazawi, J. Urbanic, and P. Center. An experiment in measuring the productivity of three parallel programming languages. In *Workshop on Productivity and Performance in High-End Computing (P-PHEC)*, 2006.

[6] P. Godefroid and N. Nagappan. Concurrency at Microsoft: An exploratory survey. In *Workshop on Exploiting Concurrency Efficiently and Correctly*, 2008.

[7] J. R. Larus and R. Rajwar. *Transactional Memory*. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2006.

[8] E. A. Lee. The problem with threads. *Computer*, 39(5):33–42, 2006.

[9] S. Lu, S. Park, E. Seo, and Y. Zhou. Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. *SIGPLAN Notices*, 43(3):329–339, 2008.

[10] M. Luff. Empirically investigating parallel programming paradigms: A null result. In *Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU)*, 2009.

[11] C. Rossbach, O. Hofmann, and E. Witchel. Is transactional programming actually easier? In *Symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2010.

[12] H. Sutter. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobbs Journal*, 30(3):16–20, 2005.