Detailed Plans
Exploiting Concurrency Efficiently and Correctly – $(EC)^2$
CAV 2008 Workshop
July 7-8, 2008, Princeton, NJ
http://www.cs.utah.edu/ec2

1 Introduction and Justification

Computing is poised at an inflection point due to the arrival of multicores. The report *Landscape of Parallel Computing Research: A View from Berkeley* (the "View from Berkeley report") characterizes at length how the landscape of computing — the nature of emerging applications, programming methods, and the hardware — are likely to be impacted by multicores. In a nutshell, multicores are inevitable because of two main reasons:

1. Extracting parallelism in sequential code through deeper pipelines and higher clock frequencies imposes unreasonable power demands.
2. The path to continuing the exponential curve of application speed-up through low-frequency/low-voltage chips that are programmed to run explicitly parallel code is, theoretically, wide open.

In other words, one cannot hope to increase performance per watt by simply porting applications to new and faster, but conventional (single core) processors. These facts have been observed even as early as 1993 (Lyon, Custom Integrated Circuits Conference): "The idea of using parallelism of slow processing units as a power-saving alternative to the single fast central processor that characterizes many of today's desktop and portable products is a good way to reduce system costs and increase performance per watt."

1.1 Why Not Existing Workshops?

There are innumerable workshops and symposia devoted to multicores, transactions, parallel libraries, memory models, compilation issues, and the impact on hardware. Why then this $(EC)^2$ workshop addressing the CAV audience? The reasons are several, discussed in the following paragraphs.

Impact on Programming and Correctness: The biggest impact of multicores is expected to be on the design and analysis of concurrent programs. To name a few issues to be addressed:

- Given the inherent bug propensity and non-compositionality (in the sense of building large-scale systems) of lock-based code, the use of locks must be confined to well justified situations, and for these situations, one must develop verification methods.
- Alternatives to locks (lock-free algorithms, transactions) must be adequately understood, with verification methods developed for them.
- Transactions must be semantically well-characterized to be useful in the real world (e.g., nesting, aborts, side effects).

- The plethora of libraries (e.g., MPI, OpenMP, Cuda, Cilk, Intel's Thread Building Blocks, Microsoft's Task Parallel Library), and languages (e.g., X10, FORTRESS, Cray's Cascade system and associated languages, and Partitioned Global Address Space languages such as UPC and Titanium) must be understood.
- One must begin tracking those languages and mechanisms that currently are, or are likely to become dominant (e.g., MPI, OpenMP). One must dig into their formal specifications and usage patterns at depth.
- Practitioners must set up challenge problems for formal methods researchers to gradually make progress along productive paths.

The above points are substantiated by the references mentioned already, and also by others in recent writings. This workshop can lead to the development of static analysis and run-time analysis methods for the classes of problems mentioned above.

Impact on Verification Methods: The arrival of multicores is expected to have a significant impact on verification methods. One must understand where model-building costs are justified, and how run-time methods are to be developed when they are not. With performance and correctness deeply intertwined, one must plan for debugging beginning at early points in the development cycle. As Ed Lee points out in his article *The Problem with Threads*, one must ensure determinacy and replayability to get a handle on the complexity of systems. These facts were emphasized many times in a recent HPC meeting at Microsoft, Redmond, by experienced developers.

In addition to these issues, there are also opportunities - such as to exploit parallelism to speed-up verification, as typified by Holzmann's recent efforts on several versions of Multicore SPIN, and various other parallel model checking methods published in the PDMC workshop series.

Impact on Hardware and Runtime Design: Many scientists have started observing the enormous importance of addressing shared memory models, lock-free algorithms, transactions, and how to address various notions of encapsulation in the area of transactions. Concurrent program semantics will be complicated by the presence of partitioned shared memory domains (e.g., as in IBM's Cell Processor), with weakly ordered 'put' and 'get' constructs (e.g., as in MPI's one-sided communication) that allow data transmission between the domains. Experts are already worried that the incorrect handling of weak memory models may lead to large-scale numerical simulations running without any apparent flaws, but nevertheless generating useless answers by virtue of having processed data that is "too old."

Education: All discussions in this area ultimately boil down to education. Programmers must be educated so that they avoid making costly mistakes. Students must be educated, with concurrency topics introduced with adequate motivations as well as rigor. The CAV community must be educated so that they start attacking fresh and important areas, instead of being stuck publishing on hackneyed topics. Language and library developers much be educated so that verifiability is taken into consideration in all design decisions. Books such as "Patterns for Parallel Programming" by Tim Mattson, Beverly Sanders, and Berna Massingill, and previous papers/blogs written by these authors must ideally become the background for at least some of the future CAV publications. Everyone in the

CAV community must realize important happenings, such as the massive scale at which PetaScale computing is being pursued, the huge volumes of data that will be generated, and formal methods to compress the data intelligently (e.g., by exploiting symmetries) and to extract useful answers. The prevalence of multicore processors at all scales - hand-held, desktop, game market, HPC, etc. - seem to offer existing CAV participants sufficient evolution paths through which their existing research skills can be made to apply to rapidly oncoming problems that require solutions to be developed on an urgent footing.

1.2 Comparisons with Recent Events

$(EC)^2$ does not have a prior occurrence, although it is inspired by previous workshops that the organizers have been involved with in the past. In particular, Gopalakrishnan co-organized (with John W. O'Leary of Intel) the TV06 workshop as part of FLoC'06 (www.cs.utah.edu/tv06) featuring multicores, transactions, memory models, and programming. This workshop was a success in many ways. However, many things could have been done differently. The contributed papers did not have the same level of quality as the invited talks. The situation of TV'06 amidst FLoC also meant that it was one of about 30-odd workshops, thus diluting its attendance (which was still very respectable). $(EC)^2$ is coming two years later, when many of the issues being discussed are even more important. Last but not least, our organizational plans (Section 3) address how $(EC)^2$ is being tailored to mesh with the CAV agenda.

2 Focal Themes

Since the CAV community has not had a "face to face" with a group of domain experts who are involved in advancing the state of practice in multicores and related issues, we would like to impart some focus to $(EC)^2$. Yet, we do not want to leave out related areas by focusing too much. In addition, CAV has been trying to restore some balance in its program; for instance, there has been a deliberate attempt to boost the number of papers in hardware verification. Keeping all this in mind, we propose four focal themes:

Hardware: How hardware is going to deal with coherent memory hierarchies, provide support for verification (e.g., run-time verification supported by 'hooks' into the virtualization layer), and performance monitoring, and provide support for higher level algorithms (e.g., transactions).

Weak memory models: How memory models are going to play a role in multicores. How programmers are to deal with weak memory models, both at the hardware and the language levels.

Parallel Libraries and Languages: How the community will deal with established and emerging libraries, and parallel languages in terms of verification. Dealing with concurrent programs where library function calls are the rule, and not the exception.

Mechanisms: Transactions, and other mechanisms to organize concurrency in principled ways.

These topics are all not "at par" in that some are broad and some are deep. This is a deliberate decision to ensure deeper traction on some specific issues that are squarely in the formal analysis area.

3 Workshop Plans and Desired Outcomes

We would like to organize a forum in which CAV attendees

- stand to learn the issues from the experts who are aware of the merits of formal analysis, but are not necessarily working full-time on formal methods,
- start thinking about developing methods that can help create correct and efficient parallel programs.

To reduce wasted organizational effort, ensure the highest quality, and free flow of ideas, the workshop plan is to have about six keynote speakers who will deliver lectures on the focal themes of this workshop (see Section 2). We will issue an open "Call for position papers" in early 2008. A booklet consisting of all the position papers will be distributed to the workshop participants. In case there are more position statements than slots for presentation, the organizers reserve the rights to select those position statements that best address the goals of the workshop. Overall, the plan is to have a highly interactive *two-day* workshop in which a lot of crucial exchanges of ideas will transpire.