



**HAL**  
open science

## Sequential Execution of Parallel Programs :). Threads Should not Play Dice. (Keynote Talk)

Luis Ceze

► **To cite this version:**

Luis Ceze. Sequential Execution of Parallel Programs :). Threads Should not Play Dice. (Keynote Talk). Pespma 2010 - Workshop on Parallel Execution of Sequential Programs on Multi-core Architecture, Jun 2010, Saint Malo, France. inria-00494298

**HAL Id: inria-00494298**

**<https://hal.inria.fr/inria-00494298>**

Submitted on 22 Jun 2010

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Sequential Execution of Parallel Programs :). Threads Should not Play Dice.

Luis Ceze  
University of Washington

## **ABSTRACT**

Current multicore systems are nondeterministic. Each time they execute a multithreaded application, even if supplied with the same input, they can produce a different output. This frustrates debugging, limits the ability to properly test multithreaded code and hinders fault-tolerant scenarios.

In this talk I will present fully deterministic shared memory multiprocessing (DMP). The behavior of an arbitrary multithreaded program on a DMP system is only a function of its inputs. I will explore multiple deterministic execution strategies with different performance, complexity and scalability overhead. Previous approaches to coping with nondeterminism in multithreaded programs have focused on replay, useful only for debugging. In contrast, while DMP systems are directly useful for debugging, we argue that parallel programs should execute deterministically in the field as well. I will end this talk with an overview of recent efforts in improving safety of multithreaded programs.