RecPlay: a Fully Integrated Practical Record/Replay System

MICHIEL RONSSE and KOEN DE BOSSCHERE
Universiteit Gent, Belgium

This paper presents a practical solution for the cyclic debugging of nondeterministic parallel programs. The solution consists of a combination of record/replay with automatic on-the-fly data race detection. This combination enables us to limit the record phase to the more efficient recording of the synchronization operations, while deferring the time consuming data race detection to the replay phase. As the record phase is highly efficient, there is no need to switch it off, hereby eliminating the possibility of Heisenbugs because tracing can be left on all the time. This paper describes an implementation of the tools needed to support RecPlay. We believe that RecPlay is currently the most advanced and practical tool available for debugging parallel programs.

Categories and Subject Descriptors: D.1.3 [**Programming Techniques**]: Concurrent Programming – *parallel programming*; D.2.5 [**Software Engineering**]: Testing and Debugging – *debugging aids; monitors; tracing*; D.4.1 [**Operating Systems**]: Process Management – *concurrency; deadlock; multiprocessing/multiprogramming; mutual exclusion; synchronization*

General Terms: Algorithms, Experimentation, Reliability

Additional Key Words and Phrases: Binary code modification, multithreaded programming, race detection

1. INTRODUCTION

Cyclic debugging assumes that a program execution can be faithfully re-executed any number of times. Therefore, nondeterministic parallel programs are hard to debug by means of cyclic debugging because subsequent executions with identical input are not guaranteed to have the same behavior. Known sources of nondeterminism are: certain system calls (like `random()` and `gettimeofday()`), interrupts, traps, signals, non-initialized variables, dangling pointers and finally —for parallel programs— unsynchronized accesses to shared memory.

There exist effective and fairly efficient ways to remove the sources of nondeterminism that occur in sequential programs: the output of nondeterministic system calls can be traced; the nondeterminism caused by interrupts, traps and signals can be dealt with by using the technique of Interrupt Replay [**?**]; non-initialized vari-

Michiel Ronsse is supported by a grant from the Flemish Institute for the Promotion of the Scientific-Technological Research in the Industry (IWT). Koen De Bosschere is a research associate with the Fund for Scientific Research – Flanders.
Address: Department of Electronics and Information Systems, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium