Real-Time Synchronization on Multicore Platforms

Department of Computer Science University of North Carolina at Chapel Hill August 2010

The Challenge

This project is motivated by two important developments that are expanding the range of real-time applications that designers wish to support. The first is the growing availability and use of commodity **multicore** and **multiprocessor** platforms. The second is the recent shift in the Linux community specifically, and the software industry generally, to introduce **real-time**-oriented features that enhance predictability in settings that have not been conventionally viewed as "real-time." Fueled by these developments, the need for "real-time execution" has moved beyond embedded devices into broader arenas, particularly arenas where multiprocessors will be used.

This realization has led to renewed interest in multiprocessor real-time systems, with considerable recent work directed at **scheduling** issues. Unfortunately, the equally-important topic of **synchronization**, in comparison, has been somewhat neglected. Indeed, the most influential work on this topic was done decades ago, at a time when multiprocessor real-time applications were deemed to be mostly of "academic" interest only.

The Approach

The objective of the proposed project is to re-visit the issue of real-time synchronization in light of the ongoing multicore revolution, with the goal of devising mechanisms that can be practically applied. The fundamental thesis of this project is "simplicity wins." Specifically, real-time synchronization mechanisms, though generally applicable, should be designed with common-case scenarios in mind, using simple techniques that can be reasonably analyzed. This begs the question: What is the common case? To determine this, trace data will be collected in this project concerning the synchronization behavior of a wide range of real-time applications. This data will be used to produce real-time synchronization benchmarks, which will guide development efforts. These efforts will consider: real-time multiprocessor locking protocols; techniques for supporting read-mostly synchronization; the use of transactional memory and non-blocking synchronization in real-time applications; and synchronization techniques that can be applied in settings where both real-time and non-real-time components co-exist. The synchronization mechanisms developed in this project will be implemented in a UNC-produced Linux extension called LITMUS^{RT} (**LI**nux **T**estbed for **MU**ltiprocessor **S**cheduling in **R**eal-**T**ime systems) and evaluated on a number of different multiprocessor/multicore test platforms in our lab.

Significance

To host real-time applications on multicore platforms, efficient multiprocessor real-time synchronization protocols are needed that have analyzable behaviors (from a real-time correctness standpoint). The development of such protocols is the focus of this project.

Project Members

James Anderson, professor
Sanjoy Baruah, professor

Research Sponsor

National Science Foundation

For More Information

Dr. James Anderson
Department of Computer Science
University of North Carolina at Chapel Hill
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175
Phone: (919) 962-1757
Fax: (919) 962-1799
E-mail: anderson@cs.unc.edu

http://www.cs.unc.edu/~anderson/projects/rtsynch.html