Thesis Proposal

Sreeram Sadasivam
M.Sc Distributed Software Systems
TU Darmstadt, Germany
sreeram.sadasivam@stud.tu-darmstadt.de

Overview

Concurrency bugs which are often resident in multi-threaded programs with shared memory designs are difficulty to find and reproduce. Deterministic multi-threading(DMT) is one such scheme indicated to resolve the above difficulty. But, DMT presents the challenge of having fixed scheduling constraints. However, currently there are no such techniques that allow to control the schedule of a multi-threaded program on a fine-grained level, i.e, on the level of single memory accesses. Thus, leaving the possibility to perform a verification of the execution program before run-time.

In the existing design, we have a thread scheduler and an verification engine. The verification engine primarily focusses on instrumenting the user code and realizing memory accesses made by various user threads. The set of safe schedules are provided by the verification engine for the given user program. The generated execution pattern is later realized with the thread scheduler, when the user program is executed. The scheduler thread is realized as another thread coupled within the framework environment. Isolating the thread as a seperate module presents the possibility of reducing execution time overhead and verification delay. The proposed design can be realized by having the thread scheduler as a Loadable Kernel Module.

Such a design would be benchmarked on various thread conditional scenarios such reader-writer problems, Peterson solution, Lamport solution. These programs enforce the verification of correctness in multi-threaded environment. And also with Indexer and LastZero benchmarking programs. The evaluation is performed on the overhead exerted by the transition to the LKM module. And additional comparison of execution overhead generated by the rival designs PARROT, CORE-DET are also considered for the above mentioned benchmarking programs. The evaluation is scaled from the use of thread count pertaining to the core count, to the use of scaled up version of thread count overshadowing the core count. Thus, creating a possibility of false sharing situations and various other potential execution overhead conditions.