Guidelines for migration of concurrent software with lock synchronization primitives to lock-free software

In the multicore era, in which we are currently living, software consists of dependent threads that execute concurrently. Concurrent threads interact over shared memory. If threads do not synchronize their operations on shared memory, the outcome could be invalid or overwritten data.

Common synchronization primitives are locks. Besides locks, there exist more complex synchronization primitives. Non-locking primitives use atomic operations and complex data structures to achieve synchronization. Advantages of non-locking synchronization are avoidance of deadlocks and possible increase in performance.

However, non-locking synchronization is not so common. Development using non-locking synchronization might prove to be more complex. This thesis aim at investigating challenges of non-locking synchronization.

Student should perform "re-synchronization" of an existing concurrent, lock based, software (e.g. Splash-2) using non-locking synchronization. Student should identify locks, using existing tools, and design a methodology for replacing them with appropriate non-locking construct. The outcome of the thesis will be a concurrent software, changed to use only non-locking synchronization, a guideline for transferring locks to non-locking synchronization primitives and a list of common pitfalls that the student faces during the transformation.

Requirements:

- Strong C/C++, JAVA development skills.
- Ability to work alone, self-initiative, self-motivation, research-oriented mind.

Beneficial, but not mandatory:

- Understanding of multi-threaded software.
- Understanding of compilers.
- Experience with LLVM compiler infrastructure.

M.Sc Jasmin Jahić
Engineer at Embedded Software Engineering
Fraunhofer IESE
Fraunhofer-Platz 1 | 67663 Kaiserslautern
Tel: +49 631 / 6800-2252 | Fax: +49 631 / 6800-9 2252
Mail: jasmin.jahic@iese.fraunhofer.de