

## Semester Thesis Speeding up programs (on multi-core systems)

### Motivation and Informal Description

Not only computer games addicts have big hopes in the computing power of multicore systems allowing for more realistic, more complex and simply better games. In fact everybody using IT will benefit in one or the other way from them. Due to this wide ranging impact of the "Multicore Revolution" even a tiny step forward in this research domain, might influence any person using a PC or PDA.

These days a huge junk of the available software is not able to make full use of these mighty processors. One of the reasons being that programs are written to be executed sequentially. Any software engineer, who is familiar with multi threading, knows the pitfalls of parallel execution of code - deadlocks, livelocks etc. and knows how time-consuming it can be to write efficient code using locks. Thus the idea of transactions has been brought from database systems to programming [4].

Essentially a transaction wraps a section of code where shared resources (usually objects) are accessed. In case two transactions demand the same resource, one of them can be aborted, freeing its resources and performing a rollback to undo all changes it has done. Transactional memory has been a "hot" topic in the last few years. Its interconnection with many different fields such as distributed computing, database systems and concurrent programming make it a fascinating subject. Despite the great effort by researchers there are many open issues left to investigate. For instance, creating a faster or more usable software transactional memory system, which might become embedded in the next Java, C# distribution or whatsoever programming language.

### Offered thesis

The student can pursue his/her own interests, though we offer the following thesis: The goal is to implement new ideas by writing a new or adapting an existing software transactional memory library such as [2, 3, 1]. The student should have good programming skills and an interest to dig deeply into the field of transactional memory.

Interested? Please contact us for more details!

### Contact

- Johannes Schneider: [schneider@tik.ee.ethz.ch](mailto:schneider@tik.ee.ethz.ch), ETZ G61.3, phone 044 632 47 76
- Roger Wattenhofer: [wattenhofer@tik.ee.ethz.ch](mailto:wattenhofer@tik.ee.ethz.ch), ETZ G63, phone 044 632 63 12

## References

- [1] R. S. Group. Rochester Software Transactional Memory. In <http://www.cs.rochester.edu/research/synchronization/rstm/>, 2006.
- [2] M. Herlihy. SXM: C# Software Transactional Memory . In <http://www.cs.brown.edu/~mph/SXM/>, 2005.
- [3] M. Herlihy, V. Luchangco, and M. Moir. A flexible framework for implementing software transactional memory. In *Proceedings of the OOPSLA Conference*, 2006.
- [4] N. Shavit and D. Touitou. Software transactional memory. *Distributed Computing*, 10, 1997.

