ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Prof. R. Wattenhofer

BA/SA/MA/Group/Lab:

Shared Counting

Consider the problem of implementing a counter object that is shared amongst several processes. The processes communicate and coordinate with each other by reading data from and writing data to a shared memory. In a single step, a process can read a shared memory register, write it, or conditionally update it using the compare-and-swap instruction. The counter object itself supports an increment operation that increments the value of the counter and a read operation that returns the current value of the counter.

We want the counter to be linearizable, which basically means that if a process increments the counter, then the increment is included in a read operation afterwards from any other process. A simple implementation would be to repeatedly try writing the incremented value into a shared register using the compare-and-swap instruction until it succeeds. The problem with this approach is that a process may end up trying forever in the worst case. So, we also want the counter operations to finish in a bounded time even in the worst case. It is possible to solve this problem so that the counter operations take $O(\log n)$ time.

Surprisingly, not everything is solved for this basic problem. For example, one of the problems with the above solution is that the counts only increase and there is no support for decrement. Also, there is no support for re-initialization. The goal of the thesis is to extend the above solution to support operations such as modulo-counting, decrement operation and intialization operation. We also have other interesting directions to start with but your ideas are also welcome!

Requirements: Interest in designing and analyzing concurrent algorithms.

Interested? Write an e-mail or just drop by for a chat!

Contacts

- Pankaj Khanchandani: pankaj.khanchandani@tik.ee.ethz.ch, ETZ G60.1