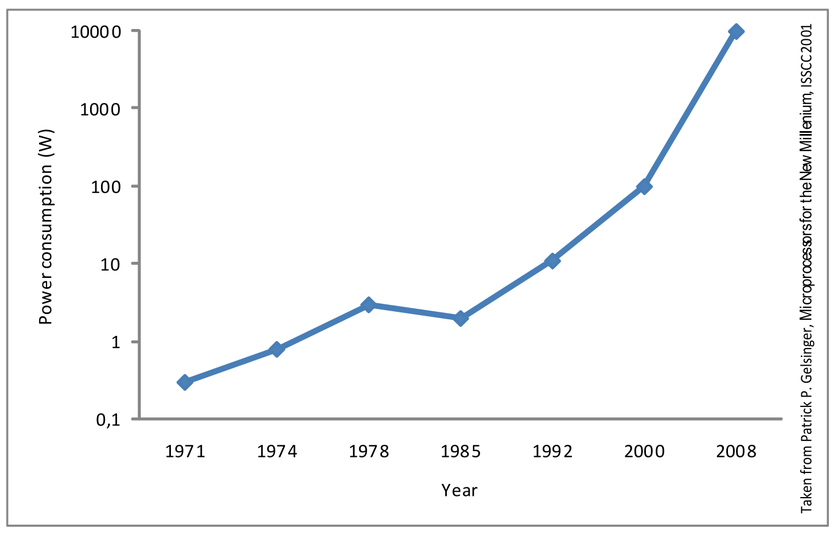
Non-blocking synchronization for multi-core processors

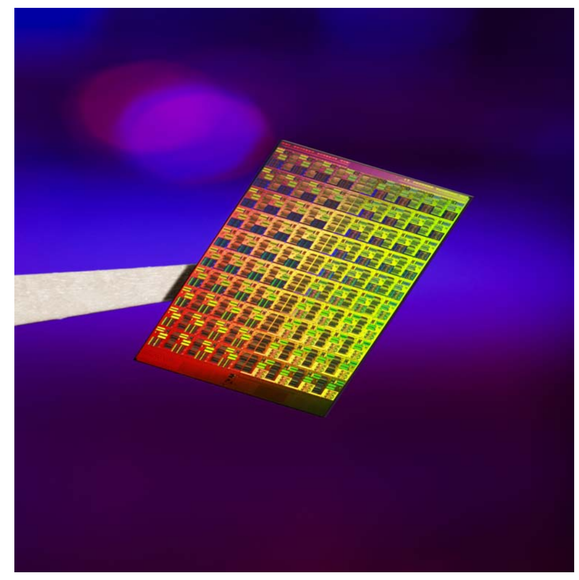
The problem with the ever increasing power consumption and heat dissipation in modern processors has moved the issue of **parallelism** into the forefront.



Predicted power consumption

Dual-core processors are now common-place and Intel has already presented a research processor with 80 cores.

Programs now have to be specially designed to run faster on **multi-core** processors. They need to be divided into tasks that can be performed in parallel on different cores. For these tasks to be able to **communicate** with each other they have to have access to safe and efficient **synchronization**.



80 core research processor

Locks

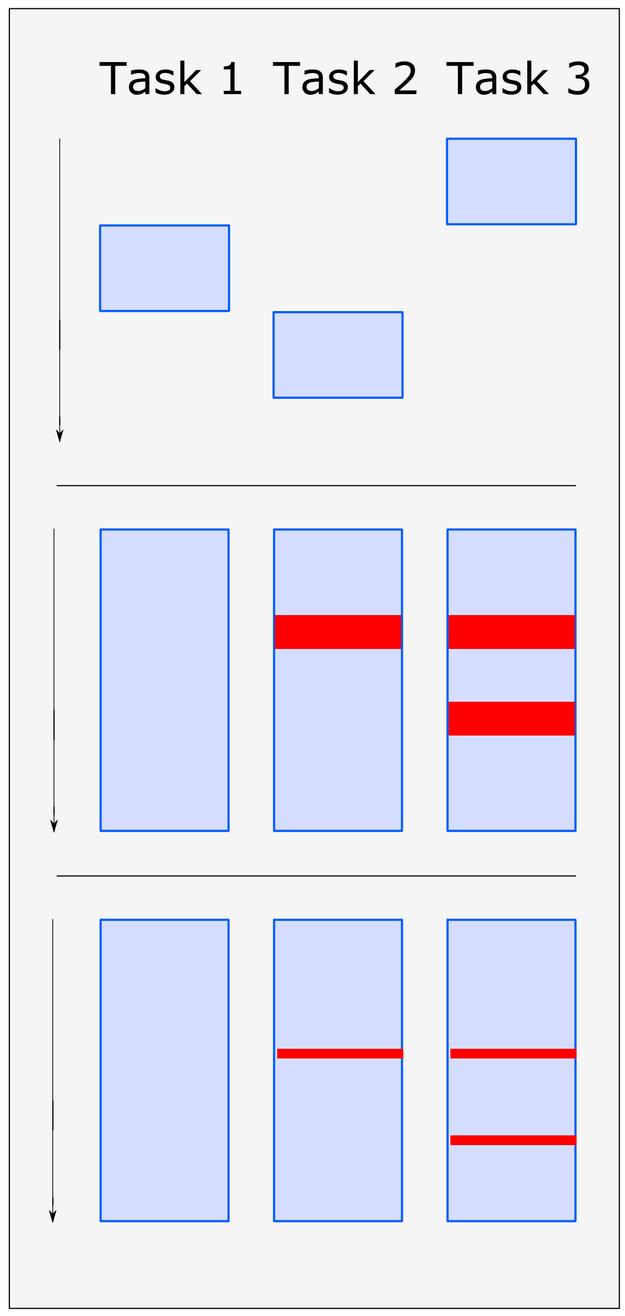
A *lock* provides a very simple form of synchronization and is one of the most commonly used methods of synchronization. The lock gives one task an exclusive access to a certain memory area. This can lead to **poor parallelization** and introduces the risk of deadlocks if the code is not designed correctly. It also makes objects hard to compose.

STM

Software transactional memory is a method that allows the programmer to mark sections of the code as atomic so that either all operations in the section are performed or none. This makes it **easy for programmers** to write concurrent but not necessarily efficient code.

Lock-free

Lock-free data structures allow tasks to communicate with each other in a **highly efficient** manner. They provide objects that can be easily composed, however they can be somewhat complicated to design.

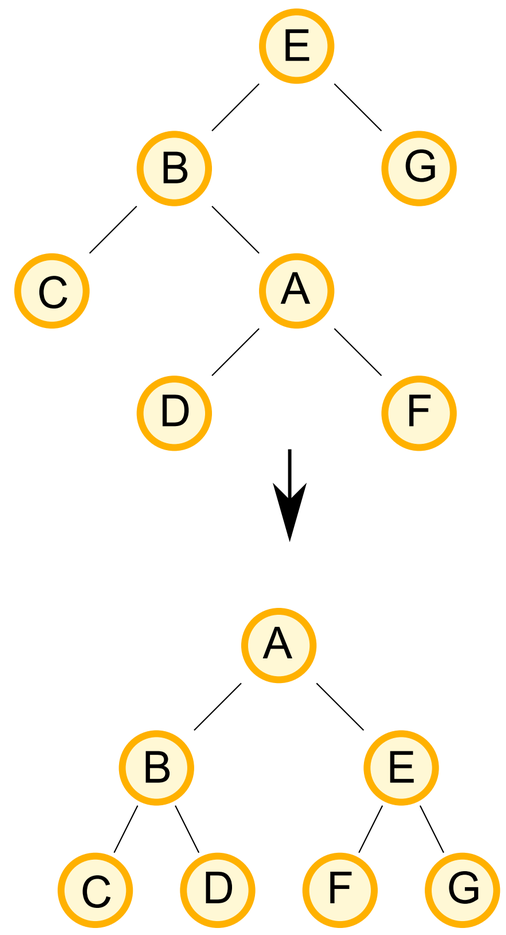


Splay tree

I'm currently trying to provide a parallel lock-free version of a splay tree.

A splay tree is a commonly used data structure. It is a self-balancing tree that gives better performance than any other tree when faced with a non-uniform sequence of searches. It behaves like a cache and gives faster access to more recently used data.

Challenges
- Frequently used data is gathered at the root which increases conflicts.
- Every access to the tree changes it which is expensive.
- No extra data should be added to the tree.



Daniel Cederman - cederman@cs.chalmers.se
Supervisor: Philippas Tsigas - tsigas@cs.chalmers.se
Distributed Computing and Systems - www.cs.chalmers.se/~dcs
Computer Science and Engineering
Chalmers University of Technology
Sweden

Microsoft® Research

CHALMERS

Distributed Computing and Systems
Chalmers university of technology