A Locality-Aware Software Transactional Memory

Patrick Marlier, Anita Sobe, Pierre Sutra
University of Neuchâtel
first.last@unine.ch

The advent of chip level multiprocessing in commodity hardware has pushed applications to be more and more parallel in order to leverage the increase of computational power. However, the art of concurrent programming is known to be a difficult task, and new paradigms are required to help the programmer. Among those paradigms, software transactional memory (STM) is widely considered as a promising direction. The two key factors contributing to the popularity of STM are its simplicity and that STM is at heart a non-blocking technique.

The engine that orchestrates concurrent transactions run by the application, i.e., the concurrency manager, is one of the core aspects of a STM implementation design. A large number of concurrency manager implementations exists, ranging from pessimistic lock-based implementations to completely optimistic ones, with, or without, multi-version support. Because application workloads exhibit in general a high degree of parallelism, these designs tend to favor optimistic concurrency control. In particular, a widely accepted approach consists in executing tentatively the read operations and validating them on the course of the transaction execution to enforce consistency.

One of the very first STM design validates the read set at each step of the transaction, resulting in a quadratic-time validation complexity. More advanced techniques employ time-based validation. In a nutshell, this approach uses a global clock to track causality relations between transactions. When a transaction starts its execution, it retrieves a starting timestamp from the global clock, and re-validates its read set only if it encounters an object storing a higher timestamp.

In every time-based STM, the critical path of a transaction contains at least two global operations. Global operations are expensive in multi-core/multi-processors architecture, due to the synchronization wall. In particular, computer designs that exploit data locality with multiple cache levels, such as non-uniform memory architectures (NUMA), have to reduce the amount of global operations to improve application performance.

A few solutions exist to relieve time-based STM implementation from the usage of a global clock. However, to the best of our knowledge, they either increase storage cost, execute a high number of invalidations, or ensure weak consistency criteria. This talk addresses these shortcomings with a novel flexible STM design. Our implementation supports invisible reads, lazy snapshots, and it can be tailored to leverage data locality (from disjoint access parallelism to NUMA-awareness). Our design aims at maximizing the native workload parallelism, while leveraging locality of computer architectures to reduce the number of invalidations. Several experiments assess that our locality-aware design is in favorable cases close to the optimum, and that in the other ones, its performance are similar to existing STM solutions.