TITRE: Synchronization and Concurrent programming
EQUIPE/THEME: Combinatoire et Algorithmique / Algorithmique distribuée
DIRECTEURS: Nicolas Hanusse, Alessia Milani
COURRIELS: nicolas.hanusse@labri.fr, milani@labri.fr
MOTS-CLES: programmation concurrente, mémoire transactionnelle, algorithmique
DIRECTEURS HABILITES: Nicolas Hanusse
DESCRIPTION du SUJET:

Transactional memory (TM) has been proposed as a new programming paradigm to support efficient multithread synchronization and to simplify concurrent programming on multicore architectures. Using the well-known abstraction of a transaction, the programmer can specify that a block of code should execute atomically: if any operation in the transaction takes place, they all do, and if so, they appear as one indivisible operation to other threads. The transactional memory is in charge of implementing atomicity.

An implementation of software transactional memory (STM) specifies representations for transactions and data items using base objects in the hardware. It also provides algorithms that apply primitive operations to these base objects, to enable processes to perform transactions.

The transactional memory paradigm will be an acceptable substitute to the well known lock-based synchronization, when efficient STM implementations will be designed. A recent technique to not pay the overhead of executing transactional code is to use small transactions to make shared data private and to access them not transactionally. This is similar to acquire a lock and then operate on the data in mutual exclusion. Unfortunately, existing transactional memories do not provide any guarantees when transactions interfere with not transactional code. Observe that this is also a key problem to allow transactional code to be integrated in legacy programs.

Additionally to its interest in pushing performance, the above approach provides a good direction to define lock-based critical sections in term of transactions, and thus to provide a uniform consistency model. But, despite the common sense, it has being showed that replacing lock-based critical sections with transactions is not always safe.

The aim of this Phd thesis is to understand the relation between lock-based synchronization and transactional synchronization. This is done to support the integration between transactional code and legacy code, to provide more performant STMs and to derive a consistency model to define the correct executions of code that may contain both locks and transactions. Moreover, from a theoretical point of view, this is important to clarify if transactional memory is a good substitue of lock-based synchronization which is the de facto synchronization paradigm.