Software Transactional Memory with Autonomic Management Techniques

Naweiluo Zhou – *Grenoble University/INRIA, France*
Gwenael Delaval –*Grenoble University,France*
Eric Rutten –*INRIA,France*
Jean-Francois Mehaut –*Grenoble University/CEA,France*

Naweiluo.Zhou@inria.fr

January 21st, 2015

Overview

1.Introduction to Parallel Program

Multi-core Processor

- Multi-core processors are everywhere, more parallelisms/concurrency levels give higher performance?
- Many threads execute concurrently. Threads share data. More threads maybe more conflict!

Synchronization VS Computation

A high concurrency level may decline computing time, but increase synchronization time. How to handle the trade-off between synchronization and computation?

1.Introduction to Parallel Program

Locks

A traditional way for synchronization. But:

- Deadlocks, vulnerability to failures, faults...
- Difficult to detect deadlocks
- Hard to figure out the interaction among concurrent operations

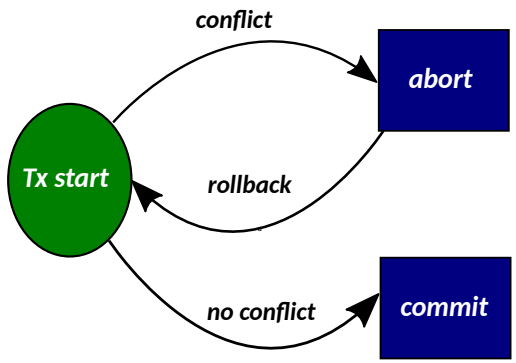
Transactional Memory

Lock-free, therefore no deadlocks! But really? Any problems? Why transactional memory does not become the dominating memory system?

2.Transactional Memory

Concepts

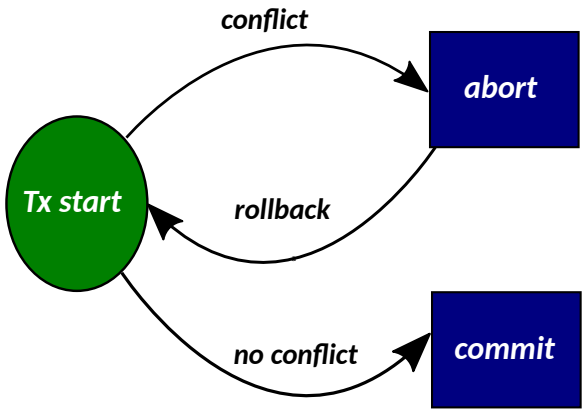
- Basic ideas are from **database systems**.
- Shared variables are wrapped by **transactions** (atomic blocks)
- concurrent accesses are performed inside transactions
- Transactions are executed speculatively and can either commit or abort. no other intermediate status



2.Transactional Memory

Three concepts

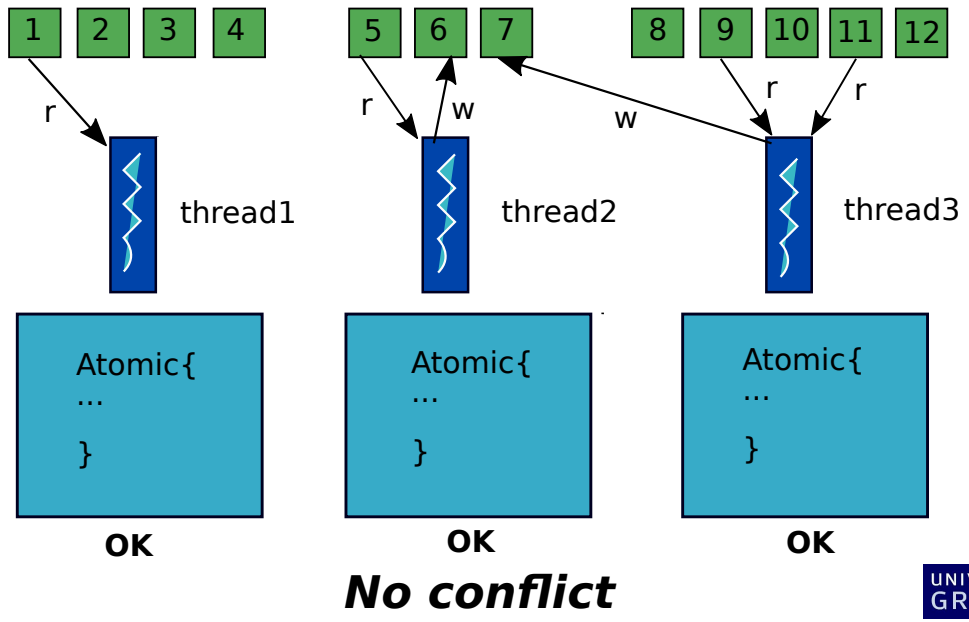
1. Commit: a transaction succeeds—changes are made
2. Abort: a transaction has a conflict — changes are discarded
3. Rollback: re-execute the aborted transactions



2.Transactional Memory

Example

consider three threads read/write data to the objects of different memory locations. Access occur inside transactions



2.Transactional Memory

Example

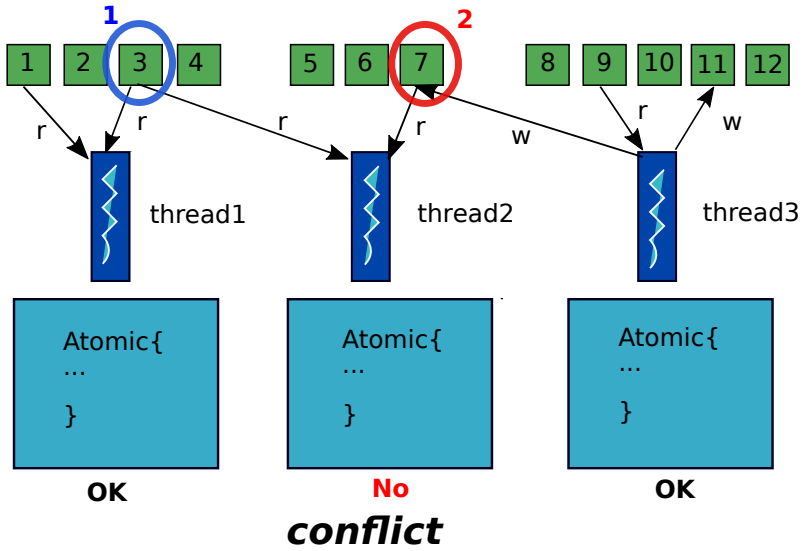
consider three threads read/write data from/to the objects of different memory locations. Access occur inside transactions

Case1

thread1 reads object3
thread2 reads object3

Case2

thread2 reads object7
thread3 reads object7



2.Transactional Memory

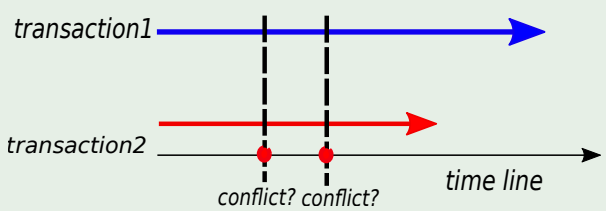
When to dectect the conflict:

Different TM systems have different choices...

It is hard to tell which one is better...

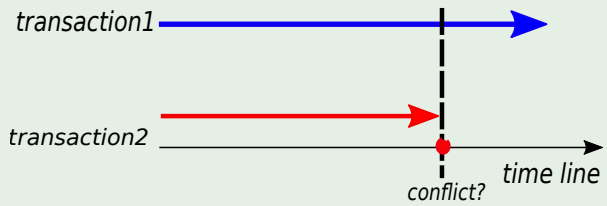
Eager

detect every memory access



Lazy

detect at commit time



2.Transactional Memory

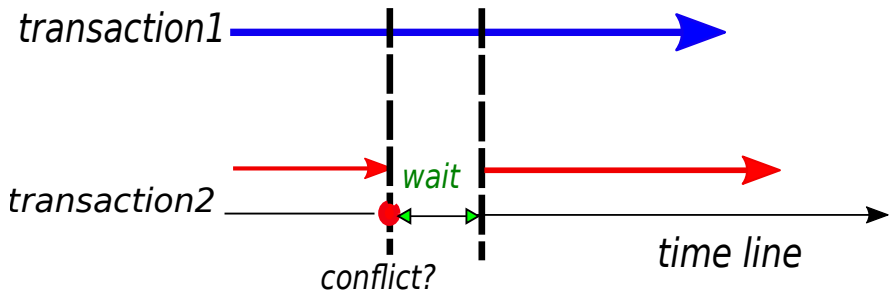
How to solve the conflict–contention manager mainly are based on three policies: **Aggressive**, **Suicide**, **Backoff**. It is hard to tell which one is better...

Three policies:

- Aggressive: I detect conflicts, I commit my work, others abort
- Suicide: I detect the conflict, I kill my work, others continue
- Backoff: I detect the conflict, I wait then I continue

The policy is a parameter which can be controlled later.

Backoff policy:



2.Transactional Memory

Locks vs TM on synchronization cost

Locks

- time wastes on acquiring locks every access
- time wastes on waiting for acquiring locks

TM

- time wastes on abort and then re-execute
- time wastes on checking conflicts (can be low though)

2.Transactional Memory —Implementation of TM

- Software Transactional Memory (STM)
  Implements all the transactional semantics in software (e.g. TinySTM, SwissTM...)
- Hardware Transactional Memory (HTM)
  Implements all the transactional semantics in hardware
- Hybrid Transactional Memory (HyTM)
  STM accelerated by hardware

2.Software Transactional Memory

Main STM issues:

- Too many parameters need to manage
  e.g.CM policy, concurrency level...
- An application behaviour changes online, parameters set statically
- ...

> —*Can we have everything adaptive and automatic?*
> —*Yes, building an autonomic control system!*

2.Autonomic Management

A system is regarded as an autonomic control system if it has one of the features:

- **Self-configuration:** a new component learns the system configurations
- **Self-optimization:** seek to improve performance & effeciency
- **Self-healing:** recover from failures
- **Self-protection:** defend against attacks

2.Autonomic Management Techniques

Elements of a feedback control loop:

1. Managed element: any software or hardware resource
2. Autonomic manager— a software component: monitor, plan, knowledge
3. Sensor: collect information
4. Effector: carry out changes

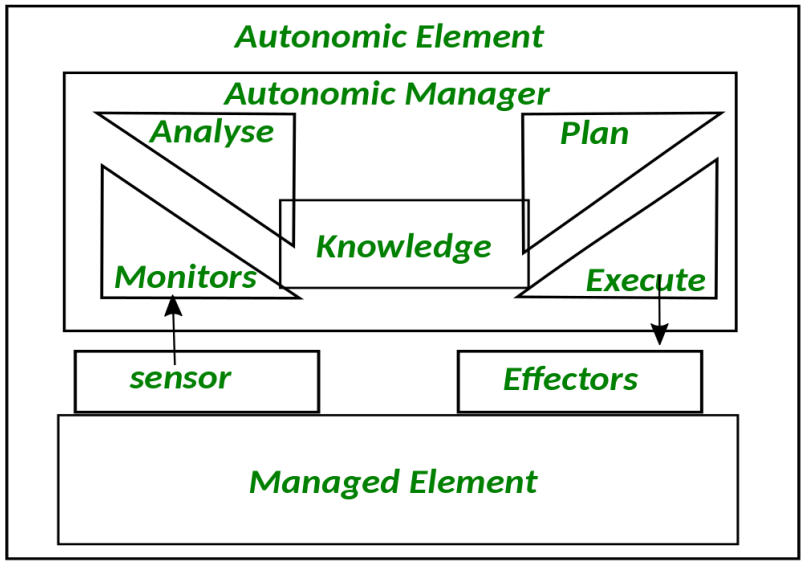


Figure : A feedback control loop

2.Autonomic Management Techniques

Components of the autonomic manager:

1. Monitor: sampling
2. Analyser:
3. Knowledge
4. Plan: use the knowledge of the system to do computation
5. Execute: make changes

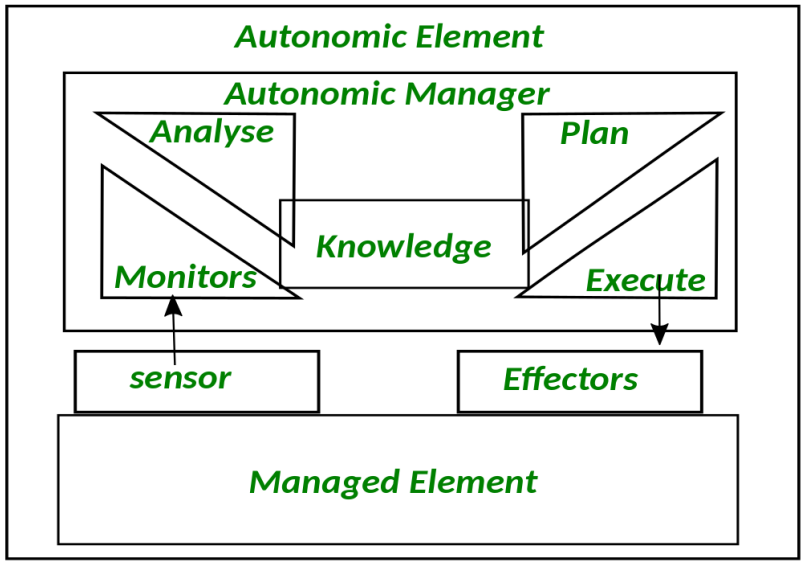


Figure : A feedback control loop

2.Autonomic Management Techniques

Components of the autonomic manager:

1. Managed elements: TM applications, STM system kernel
2. Controllers: CM controller, concurrency level controller, coordinating controller
3. Sensor: statistic collector–sampling
4. Effector: Parameters set in STM

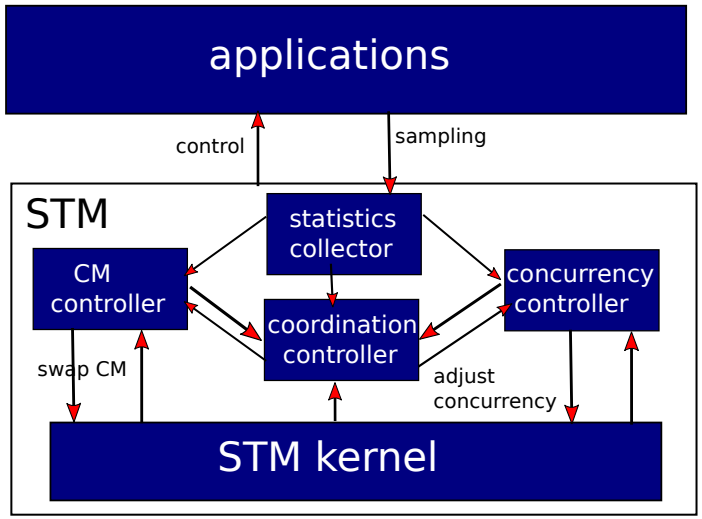


Figure : Feedback control loops on STM

Autonomic Management Techniques

We use a control approach to achieve autonomic management

- Control techniques for supervising computing systems in this project
- Feedback control loops
- Using reactive synchronous language: design safe autonomic loops

Bibiliography

The thesis extends the work of Xin AN [1], Macio Bastos Castro [2],Soguy Mak Karé Gueye [3]

1. An, Xin. "High Level Design and Control of Adaptive Multiprocessor Systems-on-Chip." PhD diss., Grenoble University, 2013.
2. Castro, Marcio Bastos. "Improving the Performance of Transactional Memory Applications on Multicores: A Machine Learning-Based Approach. " PhD diss., Universit de Grenoble, 2012
3. Gueye, Soguy Mak Karé. Modular coordination of autonomic managers by discrete control. PhD diss., Universit de Grenoble, 2014 (French)

Other related works:

3. D. Rughetti, P. Di Sanzo, B. Ciciani, and F. Quaglia. Machine learning-based self-adjusting concurrency in software transactional memory systems. In Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on, pages 278285, Aug 2012.
4. 4. M. Ansari, C. Kotselidis, K. Jarvis, M. Lujn, C. Kirkham, and I. Watson. Advanced concurrency control for transactional memory using transaction commit rate. In Proceedings of the 14th International Euro-Par Conference on Parallel Processing, Euro-Par 08, pages 719728, Berlin, Heidelberg, 2008. Springer-Verlag.

Experimental Platform

Hardware

SMP machine (12 Intel core, 64G memory) & SMP machine (192 Intel cores, 756G memory)

Software

- **STM:** tinySTM...
- **Control**: heptagon
- **Benchmark**: Eigenbench, STAMP
- **Programming languages:** C/C++, heptagon (reactive language)

3.Approaches on STM Management

Questions we will answer:

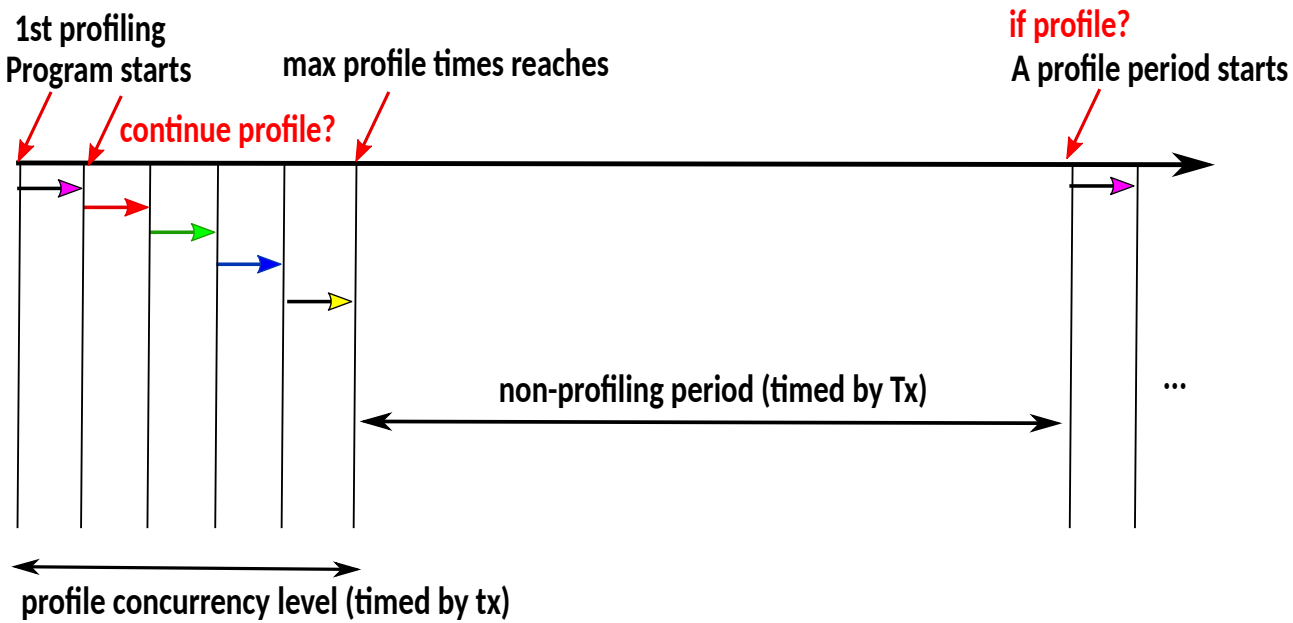
- Is the performance of TM applications affected by different configurations and parameters? How much?
- How can we deal with different TM applications?
- Are there possible feedback control loops?

Problems we will solve

- Optimize configurations at runtime
- As little changes as possible to the TM applications

3.Approaches on STM Management

A profile example



3.Approaches on STM Management

A feedback control loop on concurrency level:

1. control objective: high throughput–more commits per sec..
2. sensor: commits, aborts.
3. effector: change concurrency level (inc. or dec., how many), profile or not.
4. controller: analyse commits and aborts, make decision.
5. managed element: STM & benchmarks.

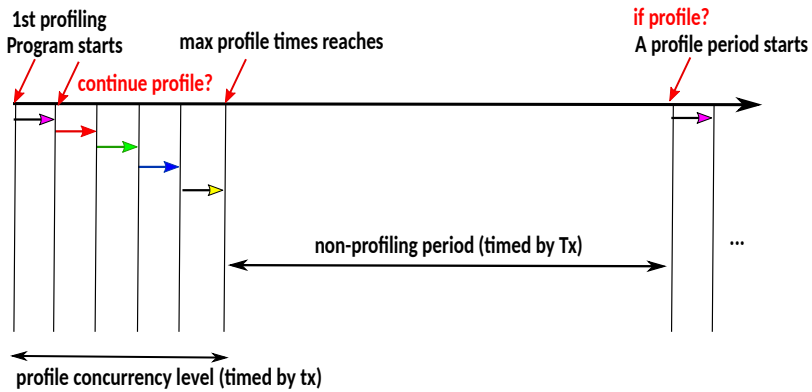


Figure : A profiling example

3.Approaches on STM Management

A feedback control loop on concurrency level:

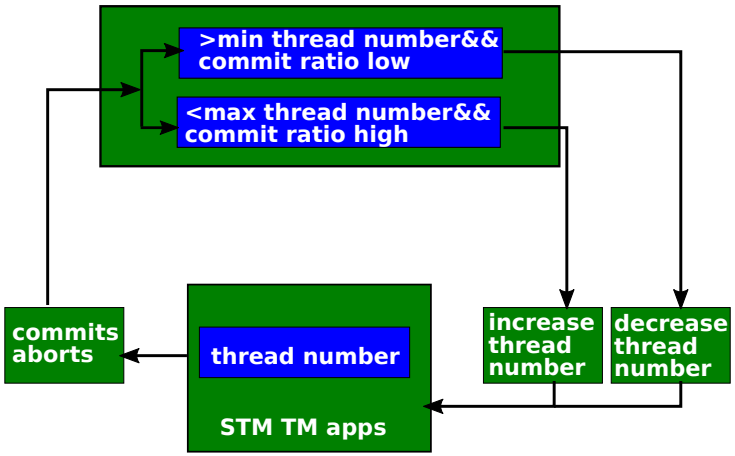


Figure : A simple feedback control loop on concurrency level

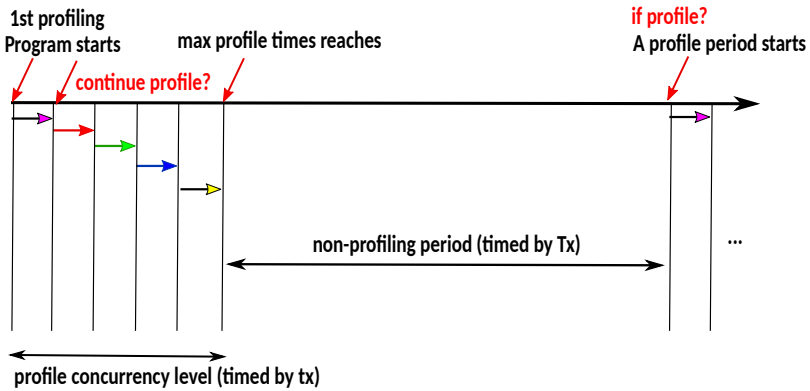


Figure : A profiling example

3.Approaches on STM Management

STM:

- Dynamic Contention Management
- Dynamic Concurrency Adaptation
- Dynamic Thread mapping policies

Control Method

- Identify autonomic loops for STM
- Implement feedback control loops with STM
- Control coordinating of different loops

Integration

Integration of STM and Control Approach

4.Conclusion

- Parallel program needs to manage the trade-off between computing time and synchronization time
- Transactional memory emerges as a promising way to overcome the shortcomings of traditional lock-based system
- Using feedback control loops to better manage STM together with its applications at runtime

5.Ongoing Work

- Integrate feedback control loops into STM system
- Integrate thread mapping polices together with concurrency level adjustment
- More monitoring inputs
- New contention manager Policies
- New profiling strategies
- ...

Questions?