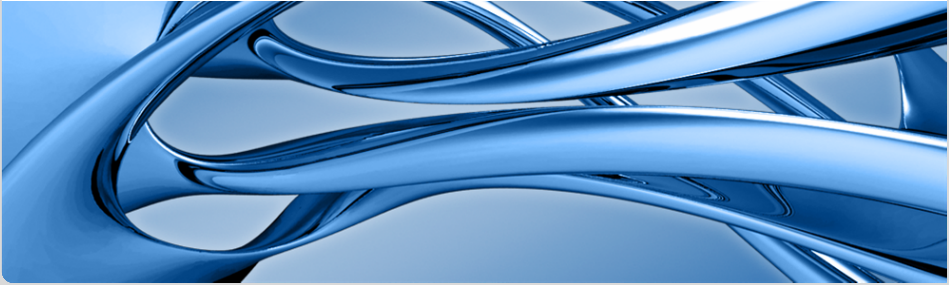
Karlsruhe Institute of Technology

Exploiting program phases in an FPGA-based Hybrid Transactional Memory system

Diploma thesis talk, Philipp Kirchhofer

Supervisor: Prof. Dr. rer. nat. Wolfgang Karl
Advisor: Dr.-Ing. Martin Schindewolf

Chair for Computer Architecture and Parallel Processing, Institute of Computer Engineering, Karlsruhe Institute of Technology



Outline

- Motivation
- Design of an adaptive TM system
- Experimental results
- Summary
- Outlook

Motivation

- Concurrent programming is required to get full performance on state-of-the-art multi-core architectures
- Commonly used techniques:
    - **Locks**: Error prone and difficult
    - **Read-Copy-Update**: Usable only in corner cases
    - **Transactional Memory**: Promising approach, easy to use, but difficult to optimize

Motivation

Short selection of related work

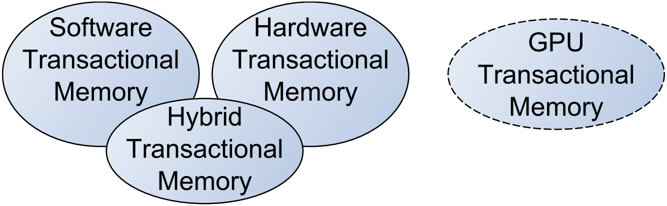

Software Transactional Memory
Hardware Transactional Memory
Hybrid Transactional Memory
GPU Transactional Memory

Figure : Various types of Transactional Memory

- Payer et al. [1] show that adaptivity is viable for STM.
- Lev et al. [2] describe a method for STM <-> HTM mode switching.
- Felber et al. [3] describe an adaptive technique for tuning STM.
- Arcas et al. [4] describe the HybridTM profiling framework that is used in this diploma thesis.

Motivation

The case for Transactional Memory adaptivity

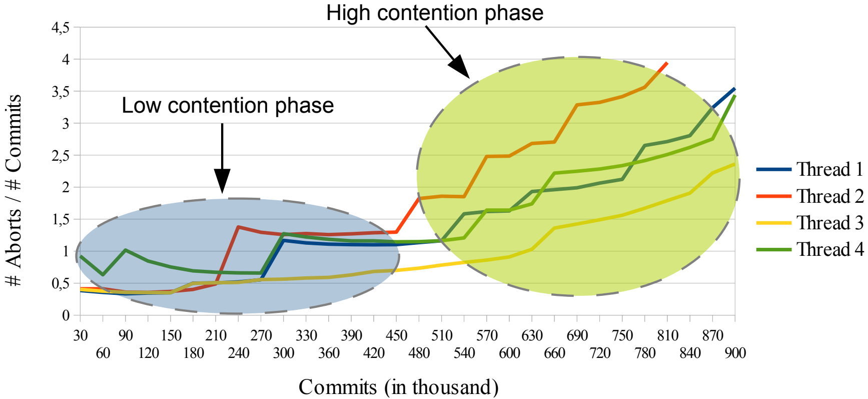


Figure : Program phases in STAMP Intruder benchmark

Motivation

This Diploma thesis improves Hybrid Transactional Memory by showing how to:

- Exploit changing program phases
- Provide adaptivity for a TM implementation
- Automatically optimize program performance

Design of an adaptive TM system I

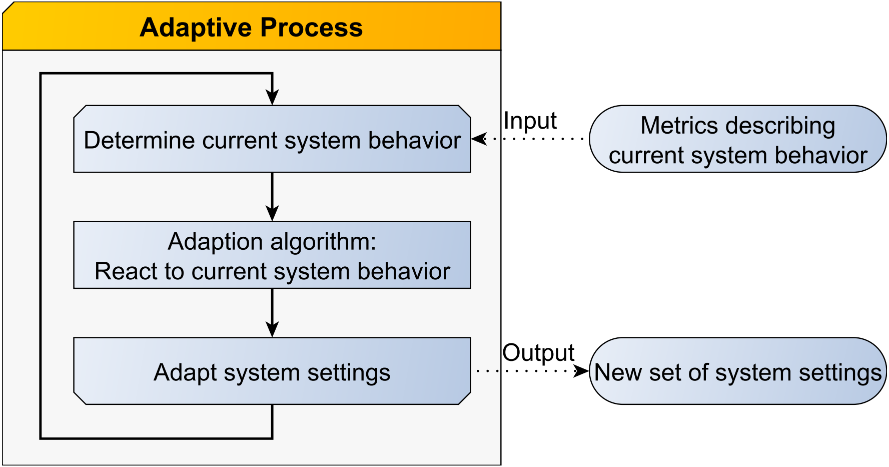

Adaptive Process
Determine current system behavior
Adaption algorithm:
React to current system behavior
Adapt system settings
Input
Metrics describing current system behavior
Output
New set of system settings

Figure : An adaptive process

Design of an adaptive TM system II

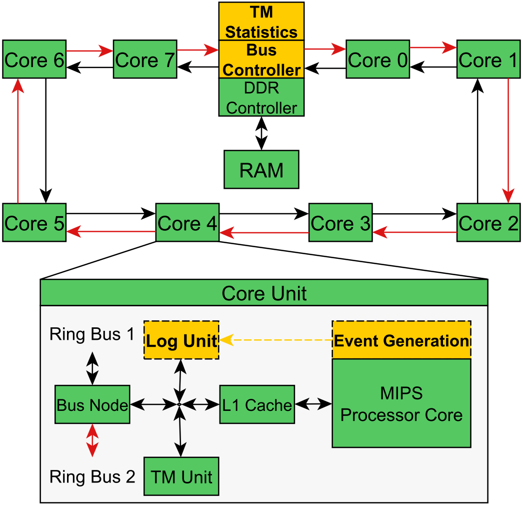


Figure : Tracing framework gets information about system behavior

Design of an adaptive TM system III

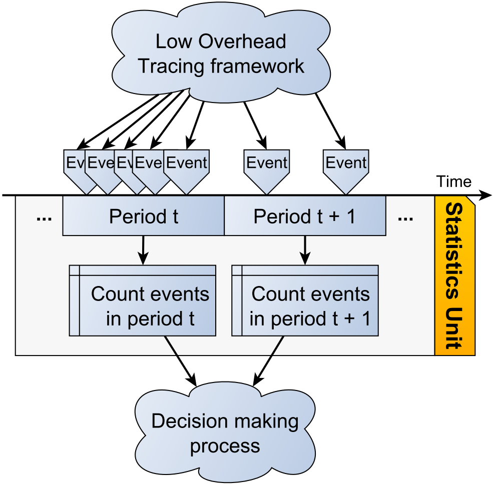


Figure : Statistics unit supplies metrics

Design of an adaptive TM system IV

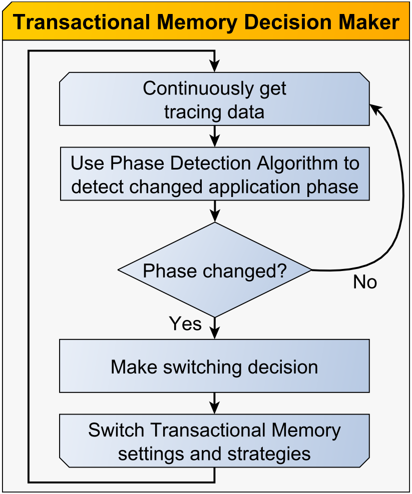

Transactional Memory Decision Maker
Continuously get tracing data
Use Phase Detection Algorithm to detect changed application phase
Phase changed?
No
Yes
Make switching decision
Switch Transactional Memory settings and strategies

Figure : Decision maker selects set of TM settings

Design of an adaptive TM system V

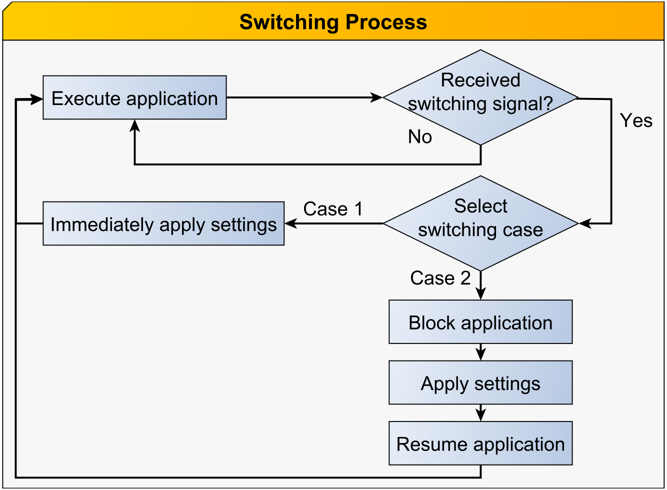
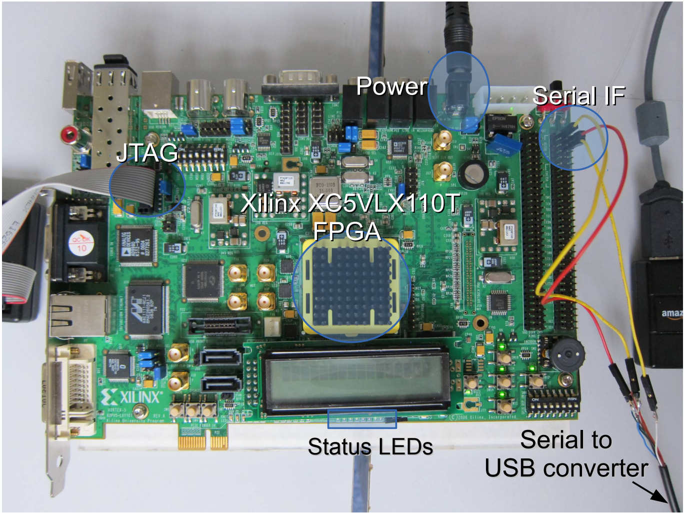


Figure : Switching process applies new set of TM settings

Experimental results

Porting to the Xilinx XUPV5 board



Experimental results

Choosing a set of promising Transactional Memory settings

- Data versioning strategies
  - Write-back
  - Write-through
- Conflict detection strategies
  - Early
  - Lazy

Combine in a reasonable way:

- Write-back using encounter-time locking (WB-ETL)
- Write-back using commit-time locking (WB-CTL)
- Write-through using encounter-time locking (WT)

Experimental results

Parameterizing the decision maker I

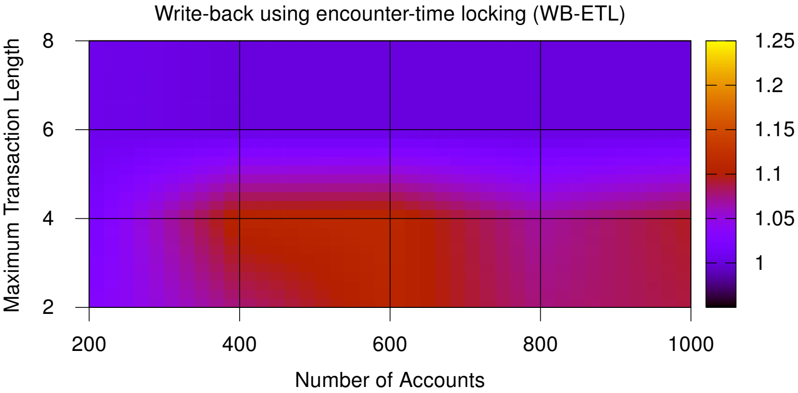


Figure : tm-bank runtime, relative to best performing strategy (interpolation 10x)

Experimental results

Parameterizing the decision maker II

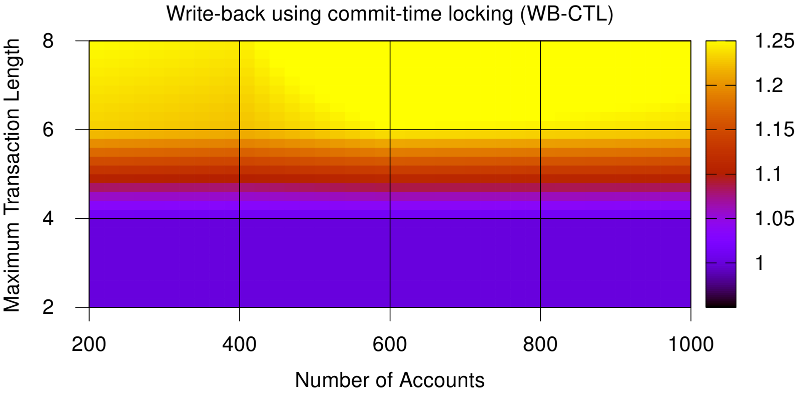


Figure : tm-bank runtime, relative to best performing strategy (interpolation 10x)

Experimental results

Parameterizing the decision maker III

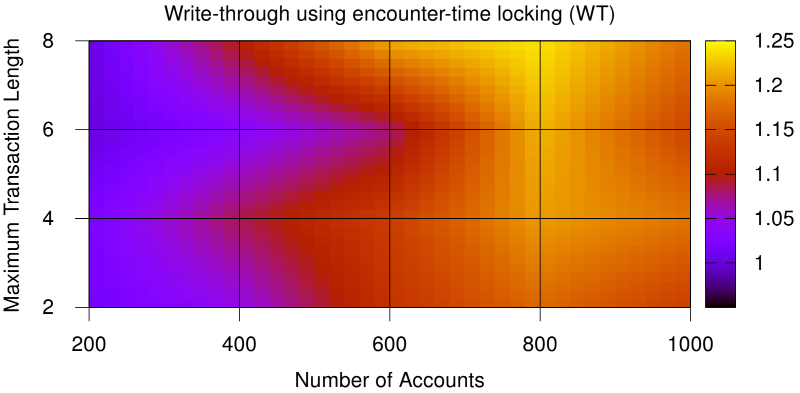
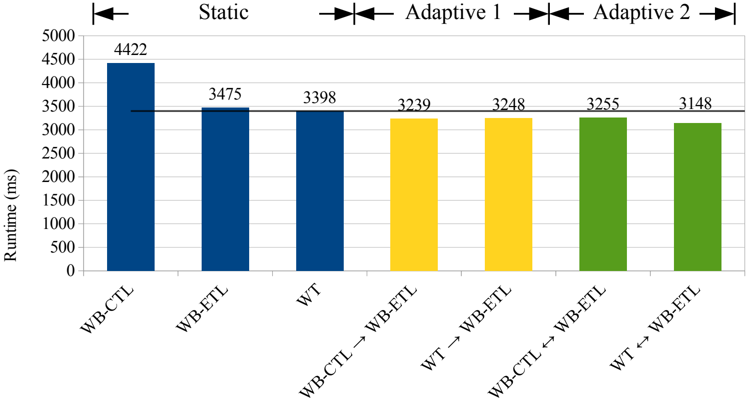


Figure : tm-bank runtime, relative to best performing strategy (interpolation 10x)

Experimental results

Static versus dynamic strategies (STAMP Intruder benchmark)



Relative improvement of up to 7 % when compared to the best static policy.
The improvement relative to the other static strategies is up to 28 %.

Summary at a glance

- Ported HybridTM implementation to XUPV5 board
- Shown systematic approach enabling the dynamic adaption of strategies and settings in a HybridTM system
- Chosen approach increases TM program performance by detecting and exploiting program phases automatically
- Strategies chosen by visually analyzing special benchmark program, Performance of STAMP Intruder improved

Outlook

- Evaluate other phase detection algorithms
- Exploit phased behavior in other STAMP programs
- Evaluate impact of HybridTM multi-processing
- Assess feasibility of adaptivity for HTM unit by using FPGA reconfiguration capabilities

Bibliography I

- Mathias Payer and Thomas R. Gross.
  Performance Evaluation of Adaptivity in Software Transactional Memory.
  In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, ISPASS '11, pages 165–174. IEEE Computer Society, 2011.

- Yossi Lev, Mark Moir, and Dan Nussbaum.
  PhTM: Phased Transactional Memory.
  Second ACM SIGPLAN Workshop on Transactional Computing, 2007.

Bibliography II

Pascal Felber, Christof Fetzer, and Torvald Riegel.
Dynamic Performance Tuning of Word-Based Software Transactional Memory.
In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, PPoPP '08, pages 237–246. ACM, 2008.

Oriol Arcas, Philipp Kirchhofer, Nehir Sonmez, Martin Schindewolf, Osman S. Unsal, Wolfgang Karl, and Adrian Cristal.
A Low-Overhead Profiling and Visualization Framework for Hybrid Transactional Memory.
In *Proceedings of the 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, FCCM '12, pages 1–8. IEEE Computer Society, 2012.

Additional slides

Karlsruhe Institute of Technology

Additional slides

Design of an adaptive TM system II

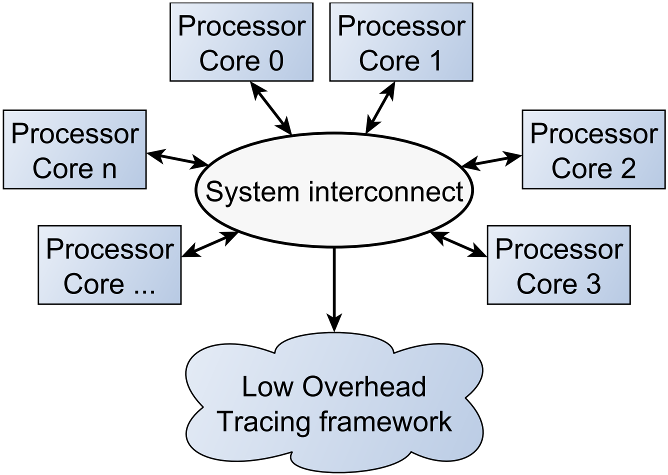
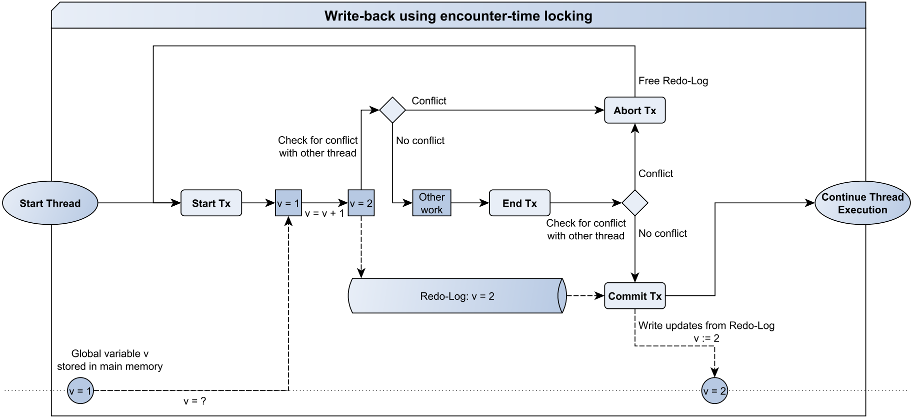


Figure : Tracing framework gets information about system behavior

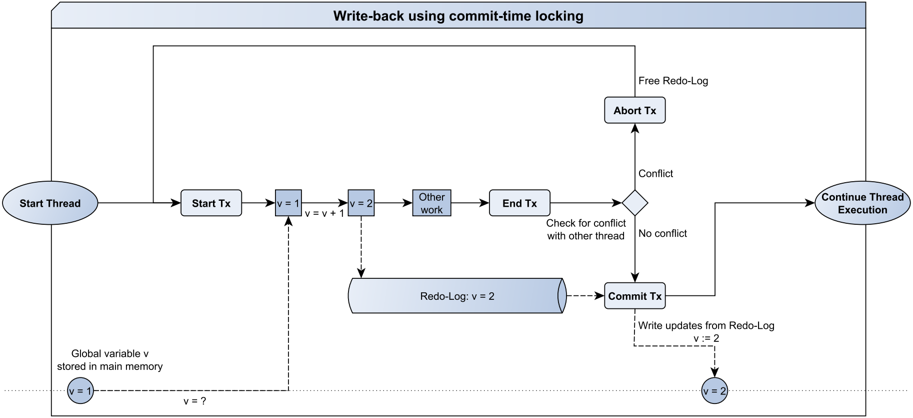
Design of an adaptive TM system II

Write-back using encounter-time locking


Write-back using encounter-time locking
Start Thread
Start Tx
v = 1
v = v + 1
v = 2
Check for conflict with other thread
Conflict
No conflict
Abort Tx
Free Redo-Log
Other work
End Tx
Check for conflict with other thread
Conflict
No conflict
Redo-Log: v = 2
Commit Tx
Write updates from Redo-Log v := 2
Continue Thread Execution
Global variable v stored in main memory
v = 1
v = ?
v = 2

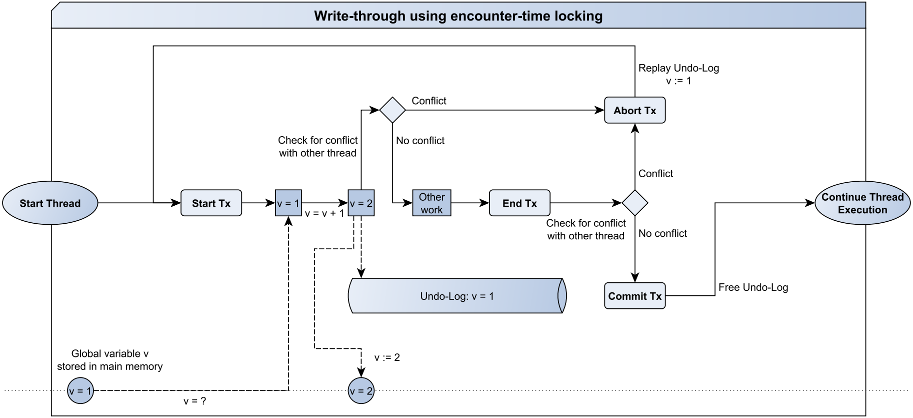
Design of an adaptive TM system II

Write-back using commit-time locking


Write-back using commit-time locking
Free Redo-Log
Abort Tx
Conflict
Start Thread
Start Tx
v = 1
v = v + 1
v = 2
Other work
End Tx
Check for conflict with other thread
No conflict
Continue Thread Execution
Redo-Log: v = 2
Commit Tx
Write updates from Redo-Log
v := 2
Global variable v stored in main memory
v = 1
v = ?
v = 2

Design of an adaptive TM system II

Write-through using encounter-time locking


Write-through using encounter-time locking
Start Thread
Start Tx
v = 1
v = v + 1
v = 2
Check for conflict with other thread
Conflict
No conflict
Other work
End Tx
Abort Tx
Replay Undo-Log v := 1
Conflict
Check for conflict with other thread
No conflict
Commit Tx
Free Undo-Log
Continue Thread Execution
Undo-Log: v = 1
Global variable v stored in main memory
v = 1
v = ?
v := 2
v = 2