Software Transactional Memory:

Worst Case Execution Time Analysis

T. Sarni, A. Queudet, P. Valduriez

Paris, October 2009
Outline of the talk

1. Introduction

2. Scheduling of transactions

3. Our contribution

4. Conclusion
Concurrent real-time tasks accessing multiple shared resources on multiprocessor systems

- Soft real-time tasks
  - Task 1
  - Task 2

- Main memory
  - shared data1
  - shared data2

Using Transactional Memory concept

Transaction
Why transactions instead of locks?

- Eases programming and avoids problems like deadlocks, priority inversion ...

```
Init Semaphores : S1, S2

P(S1)
P(S2)
V(S1)
V(S2)

do{
    <instructions>
}
until transaction_commit ()
```

Transaction

Lock-based method

Context
Why transactions instead of locks?

- Eases programming and avoids problems like deadlocks, priority inversion ...
- Locks reduce throughput in multiprocessor systems
Why transactions instead of locks?

- Eases programming and avoids problems like deadlocks, priority inversion ...
- Locks reduce throughput in multiprocessor systems

Transactions-based access

Rollbacked if a conflict occurs
Why transactions instead of locks?

- Eases programming and avoids problems like deadlocks, priority inversion...
- Locks reduce throughput in multiprocessor systems

Transactional Memory (TM)

- The term TM is introduced by Herlihy *et al*. 93

- Interest renewed with advent of multicore systems

  - Hardware-based (HTM)
  - **Software-based** (STM)
  - Hybrid-based (HyTM)
Context

Issue

Software Transactional Memory (user space)

Tasks scheduling (kernel space)

Hardware layer - Multicore

Sporadic Task \( i = (r_i, C_i, P_i, D_i) \)

CPUs

STM scheduler (Ex. FSTM, DSTM...)

Real-Time Scheduler (Ex. P-EDF, Pfair...)

Transaction 2

Transaction 1

Task 3

Task 2

Task 1
**Issue**

**Context**

Transaction 2  Transaction 1  STM scheduler
(Ex. FSTM, DSTM...)

Task 3  Task 2  Task 1  Real-Time Scheduler
(Ex. P-EDF, Pfair...)

CPUs

Software Transactional Memory
(user space)

Tasks scheduling
(kernel space)

Hardware layer - Multicore

Task

Period

time

Transaction is part of job
Outline of the talk

1. Introduction
2. Scheduling of transactions
3. Our contribution
4. Conclusion
2. Scheduling of transactions

- Transaction 2
- Transaction 1
- Task 3
- Task 2
- Task 1
- STM scheduler (Ex. FSTM, DSTM...)
- Real-Time Scheduler (Ex. P-EDF, Pfair...)
- Software Transactional Memory (user space)
- Tasks scheduling (kernel space)
- Hardware layer - Multicore

CPUs
Classification

Transaction progress guarantees

• Wait-free
  - Strongest guarantee for transactions to make progress
  - Hard real-time

• Lock-free
  - Ensures that at least one transaction is making progress
  - Soft real-time

• Obstruction-free
  - Weakest guarantee for transaction to make progress
  - Non real-time
Outline of the talk

1. Introduction

2. Scheduling of transactions

3. Our contribution

4. Conclusion
3. Our contribution

Software Transactional Memory
(user space)

Tasks scheduling
(kernel space)

Hardware layer - Multicore

CPUs

Transaction 2
Transaction 1

STM scheduler
FSTM

Real-Time Scheduler
(Ex. P-EDF, Pfair...)

Task 3
Task 2
Task 1

Transaction 2
Transaction 1
Which parts cause execution time variation?

- Intuitively: The number of transaction retries (rollback times)

But STMs are usually based upon a dynamic memory allocator...

Impact of task scheduling on the execution time of transactions?

- Evaluation of rollback and objects allocation times under: P-EDF, G-EDF, Pfair
Rollback times

- Measuring the rollback time ratio for the whole set of tasks
- Global mean

WCET jitters

- Observed WCET jitter over 10 experiments
3. Our contribution

Performance metrics (example)

Execution time variation (WCET jitters) per operation and per experiment

Task i

1 1 1 1 2 3

Rollback times

1 2 2 3

Legend

Red-black tree operations:

1 Lookup
2 Update
3 Remove

Retry (=rollback)
3. Our contribution

Experimental context

Real-Time Operating System

- We choose LITMUS$^{RT}$ (developed at University of North Carolina)
- LITMUS$^{RT}$ is a Linux-based kernel
  - Implements real-time multiprocessor scheduling policies

Software Transactional Memory

- Enhanced Fraser's STM for real-time [Sarni et al. RTCSA'09]
- Based on a lock-free algorithm

Experimental context

- Using a x86 Dual-core processor ($m = 2$)
- Under Ubuntu 8.04
- $N$ tasks ($C_i = 20ms$, $P_i = (N * C_i) / m$, $D_i = P_i$) => heavy loads (in order to obtain a high contention for shared resources accesses)
3. Our contribution

Experimental context

Dlmalloc (Linux allocator)

- Allocation $O(M/m)$  \( M \): Maximum memory size, \( m \): Largest allocated block
- Deallocation $O(1)$

TLSF (Two-Level Segregate Fit) [Masmano et al. 04]

- Dynamic memory allocator
- Based on an algorithm with constant cost $\Theta(1)$
- Therefore, reasonable use in real-time context
3. Our contribution

Experiment results

Rollback times

![Rollback times graphs for Linux, G-EDF, Pfair, and P-EDF]
3. Our contribution

Experiment results

WCET Jitters

Using classical malloc (P-EDF)

Using TLSF (P-EDF)

Zoom (TLSF)
Outline of the talk

1. Introduction
2. Scheduling of transactions
3. Our contribution
4. Conclusion
5. Conclusion

Based on a real platform and for soft real-time constraints:

- In STM and for real applications, the rollback times are not the main cause of the execution time variation,

- A good memory allocator must be provided to bound the execution time of transactions

TO DO: Future work

Study of the interaction between the real-time scheduler of transactions and that of tasks
Questions?