Worst Case Response Time for Real-Time Software Transactional Memory

Yuanfeng Wen  
Dept. of Computer Science  
University of Houston  
TX77004, USA  
wyf@cs.uh.edu

Albert M.K. Cheng  
Dept. of Computer Science  
University of Houston  
TX77004, USA  
cheng@cs.uh.edu

Chaitanya Belwal  
Dept. of Computer Science  
University of Houston  
TX77004, USA  
cbelwal@cs.uh.edu

ABSTRACT
Software transactional memory (STM) is used to control access to shared resources. The worst case response time (WCRT) estimation is different from the classical preemptive or nonpreemptive model due to its abort-restart nature. In this paper, we derive a WCRT for a STM system using lazy conflict detection. A WCRT example is also given.

Categories and Subject Descriptors
D.1.3 [PROGRAMMING TECHNIQUES]: Concurrent Programming

General Terms
Algorithms, Performance

Keywords
Software transactional memory, WCRT

1. INTRODUCTION
Software transactional memory (STM) is used to control access to shared resources in memory. In [1], RTTM is proposed which implements an abstract model for transactions with bounded response time. There are two ways to detect a conflict, i.e., eager conflict detection (ECD) and lazy conflict detection (LCD) [2, 3]. ECD detects a conflict early on, while LCD detects that at the time of committing. LCD is easier to implement; therefore, it is often used. In [4], a necessary and sufficient scheduling condition for 2-task STM using LCD is proposed. This is also the first paper which formally derives a WCRT analysis on STM systems. However, there is a strong constraint in [4]. Due to the nondeterminism in release scenario in task sets having more than two tasks, the WCRT has not been studied before. Sufﬁcient scheduling conditions can also be obtained if WCRT is known.

In this paper, we first outline the limitation of the scheduling condition in [4] as a motivation in Section II. Section III proposes a general WCRT analysis for a task set of 2 tasks. WCRT for a task set of $n$ tasks is discussed in Section IV. Section V gives a WCRT example for $n$-task set.

2. MOTIVATION
In [4], a sufficient schedulability condition for 2-task sets is proposed.

Theorem 1. [4] If a 2-task set $\Gamma_2 = \{T_1, T_2\}$ satisfies the necessary scheduling conditions, it is guaranteed to be schedulable when the total utilization factor $U$ of this task set is at most 0.5. Or, the sufficient utilization bound of $\Gamma_2$ is $U \leq 0.5$.

The assumptions in this theorem are: 1) $T_1 \leq T_2 \leq 2 \cdot T_1$, 2) $U \leq 0.5$. Under these assumptions, we have the fact that each instance of $T_1$ can be preempted by at most once. That is why the WCRT of $T_1$ is $2 \cdot C_1 + C_2$ as described in Lemma 2 of [4].

However, when $T_1 > T_2$, Theorem 1 does not hold. Here is a counter-example. Let $C_1 = 4, T_1 = 18, C_2 = 1, T_2 = 6$ and tasks are released asynchronously, that is, $T_1$ is released at $t = 0$ and $T_2$ is released at $t = 1$. Then the first instance of $T_1$ will miss its deadline. This paper will determine the WCRT for this case.

The notations and formal definitions are as follows:
• a task set $\Gamma_n = \{T_1, T_2, ..., T_n\}$ is a set of $n$ periodic tasks; $\Gamma_n$ is also called a $n$-task set;
• the priority of $T_k \in \Gamma_n$ is the positive integer $k$, where a higher number implies higher priority;
• $T_k$ is the arrival time between two successive jobs;
• $C_k$ is the execution time;
• $D_k$ is the relative deadline, in this paper, $D_k = T_k$;
• STM-LCD is the execution model of STM system which runs on a uniprocessor system and implements LCD.

3. WCRT FOR 2-TASK SET
In this section, we will first discuss the WCRT for a 2-task set.

Theorem 2. If tasks in a 2-task set $\Gamma_2 = \{T_1, T_2\}$ under STM-LCD are released asynchronously, a sufficient scheduling condition is:
1) $C_2 \leq T_2$;
2) $\left\lfloor \frac{T_2}{2} \right\rfloor (C_1 + C_2) + C_1 \leq T_1$. Or the worst case response time of $T_1$ is $\left\lfloor \frac{T_2}{2} \right\rfloor (C_1 + C_2) + C_1 \leq T_1$.

Proof. Obviously, $T_2$ is schedulable when $C_2 \leq T_2$.
Let $T_2$ be released at time $h$. To achieve the WCRT, the condition $0 < h \leq C_1 - 1$ should be according to Lemma2.
in [4]. Since there are $\lceil \frac{T_{n-1}}{T_2} \rceil$ instances of $\tau_2$, the instance of $\tau_1$ can be preempted by $\tau_2$ at most $\lceil \frac{T_{n-1}}{T_2} \rceil$ times, then the WCRT of $\tau_1$ will be $\lceil \frac{T_{n-1}}{T_2} \rceil |(C_1 + C_2) + C_1|$, which is a monotonically decreasing function of $h$. When $h = 1$, the WCRT of $\tau_1 = \lceil \frac{T_{n-1}}{T_2} \rceil |(C_1 + C_2) + C_1|$ is at a maximum. So as long as $\tau_1 = \lceil \frac{T_{n-1}}{T_2} \rceil |(C_1 + C_2) + C_1| \leq T_1$, $\tau_1$ is schedulable.

Therefore, $\lceil \frac{T_{n-1}}{T_2} \rceil |(C_1 + C_2) + C_1| \leq T_1$ and $C_2 \leq T_2$ is the sufficient schedulability condition.

In the previous example, if let $T_1 = 19$, the task set is certainly schedulable, because:

- $C_2 \leq T_2$ holds;
- $\lceil \frac{T_{n-1}}{T_2} \rceil |(C_1 + C_2) + C_1| = \lceil \frac{19}{2} \rceil (4 + 1) + 4 = 19 \leq T_1$ holds;

Notice that, Theorem 2 is valid for asynchronous models, which requires the release offset of $\tau_2$ cannot be 0. If tasks are released synchronously, the WCRT is $\lceil \frac{T_{n-1}}{T_2} \rceil |(C_1 + C_2)|$.

4. WCRT FOR N-TASK SET

Generally, we can calculate the WCRT for a $n$-task set, $\Gamma_n = \{\tau_1, \ldots, \tau_n\}$, $n \geq 2$, in this way.

**Theorem 3.** In a $n$-task, $\Gamma_n = \{\tau_1, \ldots, \tau_n\}$, $n \geq 2$, the WCRT of $\tau_i$ is: $C_i + \lceil \frac{T_{n-1}}{T_{i+1}} \rceil \cdot (C_i + C_{i+1}) + \lceil \frac{T_{n-1}}{T_{i+2}} \rceil \cdot (\max\{C_i, C_{i+1}\} + C_{i+2}) + \ldots + \lceil \frac{T_{n-1}}{T_{n-2}} \rceil \cdot (\max\{C_i, C_{i+1}, \ldots, C_{n-1}\} + C_n)$, where $i = 1, 2, 3, \ldots, n$; $h_i$ is the release offset for $\tau_i$.

**Proof.** In $[0, T_i]$, there are $\lceil \frac{T_{n-1}}{T_2} \rceil$ instances of $\tau_j$, $j > i$. Since each instance of a higher priority task can induce abort costs to only a single instance of one lower priority task, to achieve the WCRT, each instance of $\tau_j$ should preempt a task instance with largest abort cost, or largest execution time, which is $\max\{C_i, C_{i+2}, \ldots, C_{j-1}\}$. Then to finish all instances of $\tau_j$, the worst case execution time is $\lceil \frac{T_{n-1}}{T_2} \rceil \cdot (\max\{C_i, C_{i+2}, \ldots, C_{j-1}\} + C_j), j > i$. $\tau_i$ only needs $C_i$ to execute itself, but it has to wait until tasks of higher priorities to finish. Summing them up proves the theorem.

5. N-TASK WCRT EXAMPLE

One question about this theorem is whether there is a $n$-task set that can achieve the WCRT. In fact, the $n$-task WCRT example can be constructed in a recursive way.

First, the two tasks with highest priority, $\tau_n$ and $\tau_{n-1}$ are constructed as following. Let $C_{n-1} = 5kT_2 = C_n + 8k$, $T_{n-1} = 2T_n$, $T_n$ starts at $t$, and $T_2$ starts at $t + k$. The guidelines to design the example are: 1) for simplicity, $T_{n-1}$ is a multiply of $T_n$; 2) if $T_{n-1} = pT_n$, $\tau_{n-1}$ should be preempted by $\tau_n$ for exactly $p$ times; 3) there should be an idle interval left for lower priority tasks. As shown in the example in Figure 1, each instance of $\tau_{n-1}$ is preempted by instances of $\tau_n$, and there is only an interval of length $k$ idle.

Then $\tau_{n-1}$ and $\tau_n$ are considered as a single task $\tau'_{n-1}$, with $T'_{n-1} = T_{n-1}$, and $C'_{n-1} = T_{n-1} - k$.

Using the same guideline, we construct $\tau_{n-2}$ according to $\tau'_{n-1}$. In details, $C'_{n-2} = 5kT'_2 = C'_{n-1} + 8k', T'_{n-2} = 2T'_n, T'_n$ starts at $t', T'_2$ starts at $t' + k'$. Considering that $C'_{n-1} = T_{n-1} - k$, the relationship between $k'$ and $k$ is $k' = \frac{k}{2}$. Figure 1 also shows how to construct $\tau_{n-2}$ and $\tau'_{n-1}$.

Similarly, we can construct $\tau_{n-3}, \ldots, \tau_2, \tau_1$. Notice that, in order to make sure that a higher priority task always preempts a lower priority task with the longest execution time or the largest penalty, we assume that $C_{n-1} \geq C_{n-2} \geq \ldots \geq C_1$.

Here is a specific example of $n = 3$. Let $k = 8, C_3 = 1$, then $T_3 = 65, C_2 = 5k = 40, T_2 = 2 \cdot T_3 = 130, C_1 = 5 \cdot k/8 = 5$. $T_1 = 260$.

The utilization of this 3-task set is only 34%, but the processor is busy almost at all the time.

6. CONCLUSIONS

In this paper, we derived a WCRT for the general $n$-task set running on a software transactional memory using lazy confliction detection. A WCRT example for $n$-task set is constructed in a recursive way.

7. ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under Award Nos. 0720856 and 1219082.

8. REFERENCES


