FBLT

Mohammed Elshambakey, Virginia Tech University
Binoy Ravindran, Virginia Tech University

NEEDS TO BE WRITTEN

Categories and Subject Descriptors: C.3 [**Special-Purpose and Application-based Systems**]: Real-time and embedded systems

General Terms: Design, Experimentation, Measurement

Additional Key Words and Phrases: Software transactional memory (STM), real-time contention manager

ACM Reference Format:
Elshambakey, M., Binoy, R. 2012. FBLT. ACM Trans. Embedd. Comput. Syst. , , Article ( ), 6 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Embedded systems sense physical processes and control their behavior, typically through feedback loops. Since physical processes are concurrent, computations that control them must also be concurrent, enabling them to process multiple streams of sensor input and control multiple actuators, all concurrently. Often, such computations need to concurrently read/write shared data objects. They must also process sensor input and react, while satisfying time constraints.

The de facto standard for concurrent programming is the threads abstraction, and the de facto synchronization abstraction is locks. Lock-based concurrency control has significant programmability, scalability, and composability challenges [Herlihy 2006]. Software transactional memory (STM) is an alternative synchronization model for shared memory data objects that promises to alleviate these difficulties. With STM, code that read/write shared objects is organized as transactions, which execute speculatively, while logging changes made to objects. Two transactions conflict if they access the same object and one access is a write. When that happens, a contention manager (CM) [Guerraoui et al. 2005] resolves the conflict by aborting one and allowing the other to commit, yielding (the illusion of) atomicity. Aborted transactions are restarted, after rolling back the changes. In addition to a simple programming model, STM provides performance comparable to locking and lock-free approaches (see an example STM system's performance in [Saha et al. 2006]), and is composable [Harris et al. 2008]. TM has been proposed in hardware, called HTM, and in software, called STM, with the usual tradeoffs: HTM has lesser overhead, but needs transactional support in hardware; STM is available on any hardware.

New Paper, Not an Extension of a Conference Paper.
Author's addresses: M. Elshambakey and B. Ravindran, ECE Department, Virginia Tech University, Blacksburg, VA 24060, USA.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Given STM's programmability, scalability, and composability advantages, it is a compelling concurrency control technique also for multicore embedded real-time software. However, this requires bounding transactional retries, as real-time threads, which subsume transactions, must satisfy time constraints. Retry bounds under STM are dependent on the CM policy at hand (analogous to the way thread response time bounds are OS scheduler-dependent).

Past real-time CM research has proposed resolving transactional contention using dynamic and fixed priorities of parent threads, resulting in EDF-based CM (ECM) and RMS-based CM (RCM), respectively [Fahmy and Ravindran 2011; El-Shambakey and Ravindran 2012b; 2012a]. In particular, [El-Shambakey and Ravindran 2012b] shows that ECM and RCM achieve higher schedulability – i.e., greater number of task sets meeting their time constraints – than lock-free synchronization only under some ranges for the maximum atomic section length. That range is significantly expanded with the LCM contention manager in [El-Shambakey and Ravindran 2012a], increasing the coverage of STM's timeliness superiority. However, these works restrict to *one* object access per transaction, which is a major limitation.

To allow multiple objects per transaction, Priority CM with Negative value and First access (PNF) was introduced [El-Shambakey 2012]. PNF can be used with global EDF (G-EDF) and global RMA (G-RMA) multicore real-time schedulers [Davis and Burns 2011]. PNF requires priori knowledge of accessed objects by each transaction. This is not suitable with dynamic software transactional memory [Herlihy et al. 2003].

We introduce First Blocked, Last Timestamp (FBLT) CM (Section **??**). Overview about previous contention managers and their limitations that led to FBLT is given in Section **??**. FBLT combines benefits of LCM and PNF. FBLT reduces retry cost of a single transaction due to another transaction. FBLT can handle multiple objects per transaction better than ECM, RCM and LCM. In contrast to PNF, FBLT does not require prior knowledge of accessed objects by each transaction. Implementation of FBLT is simpler than PNF. FBLT is a decentralized CM that does not use locks in implementation. We establish FBLT's retry and response time upper bounds with G-EDF and G-RMA (Section **??**) schedulers. We identify the conditions under which FBLT outperforms ECM (Section **??**), RCM (Section **??**), G-EDF/LCM (Section **??**), G-RMA/LCM (Section **??**), PNF (Section 2.1) and lock-free (Section 2.2). We implement FBLT and competitor CM techniques in the Rochester STM framework [Marathe et al.] and conduct experimental studies (Section **??**)

2. FBLT VS. OTHER SYNCHRONIZATION TECHNIQUES

2.1. FBLT vs. PNF

CLAIM 1.
Let $\rho_i^j(k)$ be the difference between sum of transactional lengths of all transactions in τ_j conflicting with s_i^k, and the maximum length transaction in τ_i. Schedulability of FBLT is better or equal to PNF's if maximum abort number of any preemptive transaction s_i^k is not greater than sum of maximum $m-1$ transactional lengths in all tasks subtracted from sum of all $\rho_i^j(k)$ of all jobs of τ_j interfering with τ_i.

PROOF.
By substituting $RC_A(T_i)$ and $RC_B(T_i)$ in (**??**) with (**??**) and (6.1) in [El-Shambakey 2012] respectively.

$$\sum_{\forall \tau_i} \frac{\sum_{\forall s_i^k \in s_i} \left(\delta_i^k len(s_i^k) + \sum_{s_{iz}^k \in \chi_i^k} len(s_{iz}^k) \right) + RC_{re}(T_i)}{T_i} \tag{1}$$

$$\leq \sum_{\forall\tau_i} \frac{\sum_{\forall\tau_j\in\gamma_i}\sum_{\theta\in\theta_i}\left(\left(\left\lceil\frac{T_i}{T_j}\right\rceil+1\right)\sum_{\forall\bar{s}_j^h(\theta)} len\left(\bar{s}_j^h(\theta)\right)\right)}{T_i}$$

$\bar{s}_j^h(\theta)$ can access multiple objects. $\bar{s}_j^h(\theta)$ is included only once for all objects accessed by it. $RC_{re}(T_i)$ is given by (6.8) in [El-Shambakey 2012] in case of G-EDF, and (6.10) in [El-Shambakey 2012] in case of G-RMA. Substituting $RC_{re}(T_i) = \sum_{\forall\tau_j\in\gamma_i}\left(\left\lceil\frac{T_i}{T_j}\right\rceil+1\right)s_{i_{max}}$, covers $RC_{re}(T_i)$ given by (6.8) and (6.10) in [El-Shambakey 2012] and maintains correctness of (1). If τ_j has no shared objects with τ_i, then release of any higher priority job τ_j^y will not abort any transaction in any job of τ_i. So, (1) holds if

$$\sum_{\forall\tau_i}\frac{\sum_{\forall s_i^k\in s_i}\left(\delta_i^k len(s_i^k)+\sum_{s_{iz}^k\in\chi_i^k} len(s_{iz}^k)\right)}{T_i}$$
$$\leq \sum_{\forall\tau_i}\frac{\sum_{\forall\tau_j\in\gamma_i}\left(\left\lceil\frac{T_i}{T_j}\right\rceil+1\right)\left(\left(\sum_{\forall\bar{s}_j^h(\Theta),\,\Theta\in\theta_i} len\left(\bar{s}_j^h(\theta)\right)\right)-s_{i_{max}}\right)}{T_i} \quad (2)$$

For each $s_i^k \in s_i$, there are a set of zero or more $\bar{s}_j^h(\Theta) \in \tau_j, \forall\tau_j \neq \tau_i$ that are conflicting with s_i^k. Assuming this set of conflicting transactions with s_i^k is denoted as $\eta_i^k(j) = \left\{\bar{s}_j^h(\Theta)\in\tau_j : (\Theta\in\theta_i)\wedge(\tau_j\neq\tau_i)\wedge\left(\bar{s}_j^h(\Theta)\notin\eta_i^l,\, l\neq k\right)\right\}$. The last condition $\bar{s}_j^h(\Theta)\notin\eta_i^l,\, l\neq k$ in definition of η_i^k ensures that common transactions $\bar{s}_j^h$ that can conflict with more than one transaction $s_i^k\in\tau_i$ are split among different $\eta_i^k(j),\ k=1,..,|s_i|$. This condition is necessary because in PNF, no two or more transactions of τ_i^x can be aborted by the same transaction of τ_j^h. Let γ_i^k be subset of γ_i that contains tasks with transactions conflicting directly with s_i^k. By substitution of $\eta_i^k(j)$ and γ_i^k in (2), (2) holds if for each s_i^k:

$$\delta_i^k + \sum_{s_{iz}^k\in\chi_i^k} len\left(\frac{s_{iz}^k}{s_i^k}\right)$$
$$\leq \sum_{\forall\tau_j\in\gamma_i^k}\left(\left\lceil\frac{T_i}{T_j}\right\rceil+1\right)\left(\left(\sum_{\forall\bar{s}_j^h(\Theta)\in\eta_i^k(j)} len\left(\frac{\bar{s}_j^h(\Theta)}{s_i^k}\right)\right)-len\left(\frac{s_{i_{max}}}{s_i^k}\right)\right) \quad (3)$$

Non-preemptive transactions preceding s_i^k in m_set can have direct or indirect conflict with s_i^k. Under PNF, transactions can only have direct conflict with s_i^k. So, s_{iz}^k on the left hand side of (3) is not necessarily included in $\bar{s}_j^h(\Theta)$ on the right hand side of (3). Let $\epsilon = \{s_{u_{max}} : (1\leq u\leq n)\wedge(s_{u1_{max}}\geq s_{u2_{max}},\ u1<u2)\}$, where n is number of tasks, and $s_{u_{max}}$ is maximum transactional length in any job of τ_u. Thus, ϵ is the set of maximum transactional lengths of all task in non-increasing order. Each $s_{u_{max}}\in\epsilon$ belongs to a distinct task. Thus, $\sum_{s_{iz}^k\in\chi_i^k} len\left(\frac{s_{iz}^k}{s_i^k}\right) \leq \sum_{u=1,\, s_{u_{max}}\in\epsilon}^{min(n,m)-1} s_{u_{max}}$. $\sum_{u=1,\, s_{u_{max}}\in\epsilon}^{min(n,m)-1} s_{u_{max}}$ is the sum of at most maximum $m-1$ transactional lengths of all tasks. $|\chi_i^k|\leq m-1$. Then (3) holds if

$$\delta_i^k \leq \left(\sum_{\forall\tau_j\in\gamma_i^k}\left(\left\lceil\frac{T_i}{T_j}\right\rceil+1\right) len\left(\frac{\left(\sum_{\forall\bar{s}_j^h(\Theta)\in\eta_i^k(j)} len\left(\bar{s}_j^h(\Theta)\right)\right)-s_{i_{max}}}{s_i^k}\right)\right)$$
$$-\sum_{u=1,\, s_{u_{max}}\in\epsilon}^{min(n,m)-1} s_{u_{max}}$$

As $\rho_i^j(k) = \left(\sum_{\forall \bar{s_j^h}(\Theta)\in\eta_i^k(j)} len\left(\bar{s_j^h}(\Theta)\right)\right) - s_{i_{max}}$, $\tau_j \in \gamma_i^k$, and $\left(\left\lceil \frac{T_i}{T_j}\right\rceil + 1\right)$ is total number of jobs of τ_j interfering with τ_i, Claim follows.

□

2.2. FBLT vs. Lock-free

CLAIM 2.
Under G-EDF and G-RMA, schedulability of FBLT is equal or better than lock-free's if $s_{max} \le r_{max}$. If transactions execute in FIFO order (i. e., $\delta_i^k = 0, \forall s_i^k$) and contention is high, s_{max} can be much larger than r_{max}.

PROOF.
Lock-free synchronization [Devi et al. 2006; El-Shambakey and Ravindran 2012b] accesses only one object. Thus, the number of accessed objects per transaction in FBLT is limited to one. This allows us to compare the schedulability of FBLT with the lock-free algorithm.

By substituting $RC_A(T_i)$ and $RC_B(T_i)$ in **(??)** with **(??)** and (6.17) in [El-Shambakey 2012] respectively.

$$\sum_{\forall\tau_i} \frac{\sum_{\forall s_i^k\in s_i}\left(\delta_i^k len(s_i^k) + \sum_{s_{iz}^k\in\chi_i^k} len(s_{iz}^k)\right) + RC_{re}(T_i)}{T_i} \le \sum_{\forall\tau_i} \frac{\left(\sum_{\forall\tau_j\in\gamma_i}\left(\left(\left\lceil\frac{T_i}{T_j}\right\rceil + 1\right)\beta_{i,j} r_{max}\right)\right) + RC_{re}(T_i)}{T_i} \tag{4}$$

where $\beta_{i,j}$ is the number of retry loops of τ_j that access the same objects as accessed by any retry loop of τ_i [Devi et al. 2006]. r_{max} is the maximum execution cost of a single iteration of any retry loop of any task [Devi et al. 2006]. For G-EDF(G-RMA), any job τ_i^x under FBLT has the same pattern of interference from higher priority jobs as ECM(RCM) respectively. $RC_{re}(T_i)$ for ECM, RCM and lock-free are given by Claims 25, 26 and 27 in [El-Shambakey 2012] respectively. $RC_{re}(T_i) = \left\lceil\frac{T_i}{T_j}\right\rceil s_{i_{max}}, \forall\tau_j \ne \tau_i$ covers $RC_{re}(T_i)$ for G-EDF/FBLT and G-RMA/FBLT. $RC_{re}(T_i) = \left\lceil\frac{T_i}{T_j}\right\rceil r_{i_{max}}, \forall\tau_j \ne \tau_i$ covers retry cost for G-EDF/lock-free and G-RMA/lock-free. (4) becomes

$$\sum_{\forall\tau_i} \frac{\sum_{\forall s_i^k\in s_i}\left(\delta_i^k len(s_i^k) + \sum_{s_{iz}^k\in\chi_i^k} len(s_{iz}^k)\right) + \sum_{\forall\tau_j\ne\tau_i}\left\lceil\frac{T_i}{T_j}\right\rceil s_{i_{max}}}{T_i} \le \sum_{\forall\tau_i} \frac{\left(\sum_{\forall\tau_j\in\gamma_i}\left(\left(\left\lceil\frac{T_i}{T_j}\right\rceil + 1\right)\beta_{i,j} r_{max}\right)\right) + \sum_{\forall\tau_j\ne\gamma_i}\left\lceil\frac{T_i}{T_j}\right\rceil r_{i_{max}}}{T_i} \tag{5}$$

Since $s_{max} \ge s_{i_{max}}$, $len(s_i^k)$, $len(s_{iz}^k)$, $\forall i, z, k$ and $r_{max} \ge r_{i_{max}}$ (5) holds if

$$\sum_{\forall\tau_i} \frac{\left(\left(\sum_{\forall s_i^k\in s_i}(\delta_i^k + |\chi_i^k|)\right) + \sum_{\forall\tau_j\ne\tau_i}\left\lceil\frac{T_i}{T_j}\right\rceil\right) s_{max}}{T_i} \le \sum_{\forall\tau_i} \frac{\left(\left(\sum_{\forall\tau_j\in\gamma_i}\left(\left(\left\lceil\frac{T_i}{T_j}\right\rceil + 1\right)\beta_{i,j}\right)\right) + \sum_{\forall\tau_j\ne\tau_i}\left\lceil\frac{T_i}{T_j}\right\rceil\right) r_{max}}{T_i} \tag{6}$$

(6) holds if for each τ_i

$$\left(\left(\sum_{\forall s_i^k\in s_i}\left(\delta_i^k + |\chi_i^k|\right)\right) + \sum_{\forall\tau_j\ne\tau_i}\left\lceil\frac{T_i}{T_j}\right\rceil\right) s_{max} \tag{7}$$

$$\leq \left(\left(\sum_{\forall \tau_j \in \gamma_i} \left(\left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1\right) \beta_{i,j}\right)\right) + \sum_{\forall \tau_j \neq \tau_i} \left\lceil \frac{T_i}{T_j} \right\rceil\right) r_{max}$$

$$\frac{s_{max}}{r_{max}} \leq \frac{\left(\sum_{\forall \tau_j \in \gamma_i} \left(\left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1\right) \beta_{i,j}\right)\right) + \sum_{\forall \tau_j \neq \tau_i} \left\lceil \frac{T_i}{T_j} \right\rceil}{\left(\sum_{\forall s_i^k \in s_i} \left(\delta_i^k + |\chi_i^k|\right)\right) + \sum_{\forall \tau_j \neq \tau_i} \left\lceil \frac{T_i}{T_j} \right\rceil} \tag{8}$$

It appears from (8) that as δ_i^k, as well as $|\chi_i^k|$, increases, then s_{max}/r_{max} decreases. So, to get the lower bound on s_{max}/r_{max}, let $\sum_{\forall s_i^k \in s_i} \left(\delta_i^k + |\chi_i^k|\right)$ reaches its maximum value. This maximum value is the total number of interfering transactions belonging to any job τ_j^l, $j \neq i$. Priority of τ_j^l can be higher or lower than current instance of τ_i. Beyond this maximum value, there will be no more transactions to conflict with s_i^k. So, higher values for any δ_i^k beyond maximum value will be ineffective. $\sum_{\forall s_i^k \in s_i} \left(\delta_i^k + |\chi_i^k|\right) \leq \sum_{\forall \tau_j \in \gamma_i} \left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1\right)$. Consequently, (8) will be

$$\frac{s_{max}}{r_{max}} \leq \frac{\left(\sum_{\forall \tau_j \in \gamma_i} \left(\left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1\right) \beta_{i,j}\right)\right) + \sum_{\forall \tau_j \neq \tau_i} \left\lceil \frac{T_i}{T_j} \right\rceil}{\left(\sum_{\forall \tau_j \in \gamma_i} \left(\left\lceil \frac{T_i}{T_j} \right\rceil + 1\right)\right) + \sum_{\forall \tau_j \neq \tau_i} \left\lceil \frac{T_i}{T_j} \right\rceil} \tag{9}$$

As we look for the lower bound on $\frac{s_{max}}{r_{max}}$, let $\beta_{i,j}$ assumes its minimum value. So, $\beta_{i,j} = 1$. (9) holds if $\frac{s_{max}}{r_{max}} \leq 1$.

Let $\delta_i^k(T_i) \rightarrow 0$ in (8). This means transactions approximately execute in their arrival order. Let $\beta_{i,j} \rightarrow \infty$, $\left\lceil \frac{T_i}{T_j} \right\rceil \rightarrow \infty$ in (8) . This means contention is high. Consequently, $\frac{s_{max}}{r_{max}} \rightarrow \infty$. So, if transactions execute in FIFO order and contention is high, s_{max} can be much larger than r_{max}. Claim follows.

□

REFERENCES

DAVIS, R. I. AND BURNS, A. 2011. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv. 43,* 4, 35:1–35:44.

DEVI, U., LEONTYEV, H., AND ANDERSON, J. 2006. Efficient synchronization under global edf scheduling on multiprocessors. In *18th Euromicro Conference on Real-Time Systems*. 10 pp. –84.

EL-SHAMBAKEY, M. 2012. Phd proposal. Ph.D. thesis, Virginia Tech.

EL-SHAMBAKEY, M. AND RAVINDRAN, B. 2012a. Stm concurrency control for embedded real-time software with tighter time bounds. In *Proceedings of the 49th Annual Design Automation Conference*. DAC '12. ACM, New York, NY, USA, 437–446.

EL-SHAMBAKEY, M. AND RAVINDRAN, B. 2012b. Stm concurrency control for multicore embedded real-time software: time bounds and tradeoffs. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. SAC '12. ACM, New York, NY, USA, 1602–1609.

FAHMY, S. AND RAVINDRAN, B. 2011. On stm concurrency control for multicore embedded real-time software. In *International Conference on Embedded Computer Systems*. SAMOS. 1 –8.

GUERRAOUI, R., HERLIHY, M., AND POCHON, B. 2005. Toward a theory of transactional contention managers. In *PODC*. 258–264.

HARRIS, T., MARLOW, S., JONES, S. P., AND HERLIHY, M. 2008. Composable memory transactions. *Commun. ACM 51*, 91–100.

HERLIHY, M. 2006. The art of multiprocessor programming. In *PODC*. 1–2.

HERLIHY, M., LUCHANGCO, V., MOIR, M., AND SCHERER, III, W. N. 2003. Software transactional memory for dynamic-sized data structures. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*. PODC '03. ACM, New York, NY, USA, 92–101.

MARATHE, V., SPEAR, M., HERIOT, C., ACHARYA, A., EISENSTAT, D., SCHERER III, W., AND SCOTT, M. Lowering the overhead of nonblocking software transactional memory. In *Workshop on Languages, Compilers, and Hardware Support for Transactional Computing*. TRANSACT.

SAHA, B., ADL-TABATABAI, A.-R., ET AL. 2006. McRT-STM: a high performance software transactional memory system for a multi-core runtime. In *PPoPP*. 187–197.

Received ; revised ; accepted