

Lecture 19: Transactional Memories III

Papers:

- Colorama: Architectural Support for Data-Centric Synchronization, HPCA'07, Illinois
- Bulk Disambiguation of Speculative Threads in Multiprocessors, ISCA'06, Illinois
- Unbounded Transactional Memory, HPCA'05, MIT
- An Integrated Hardware-Software Approach to Flexible Transactional Memory, ISCA'07, Rochester

TCC Vs. LogTM

- Versioning:
 - TCC takes advantage of caches to maintain versions; requires write-thru at commit (write-back will require a separate buffer to keep track of the old value)
 - LogTM almost always makes a copy (hopefully in a write buffer, with spills into cache), allows: writeback policy, transaction working set to spill out of cache

TCC Vs. LogTM

- Conflict detection:
 - TCC implements lazy conflict detection: aborted transactions need not broadcast their writes; more wasted work for transactions that abort; easier to implement conflict resolution (since one of the transactions will go on to commit)
 - LogTM implements eager conflict detection: writes by aborted transactions are also “seen”; conflicting transactions have to just wait, not abort; conflicts may lead to deadlock/livelock and sw resolution

TCC Vs. LogTM

- Complexity / Scalability:
 - TCC is conceptually simpler as it is focused on the common case (in my opinion); parallel commits are largely distributed (except the TID vendor)
 - LogTM can handle large transactions (introduces log management, sticky states, etc.); built upon a directory-based protocol and hence mostly distributed (conflicts need resolution by a central agent if they persist)
 - Which one leads to more messages?

Colorama

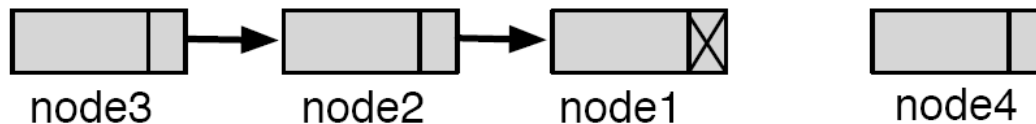
- Novel architecture built upon an underlying TM system
- Hypothesis: it is easier for the programmer to identify regions of data that must be handled by a single thread in a consistent manner (data centric synchronization – DCS) rather than regions of code that must execute atomically
- The former requires *local* reasoning, while the latter requires *non-local* reasoning
- Hence, programmer assigns colors to data structures and the hardware introduces TM-begin/end around accesses to such elements

Examples

- Color assigns a new color to an address range
- Colorprop adds another element to an existing color
- Need not insert TM-begin/end in each function that deals with these structures
- Easy for static data, harder for dynamically created data structures

Functions to manipulate the linked list:

```
insert_node(), delete_node(), traverse_list()
```



```
color(&node1, sizeof(node1), RED)
colorprop(&node2, sizeof(node2), &node1)
...
```

Figure 1. Example of linked-list manipulation.

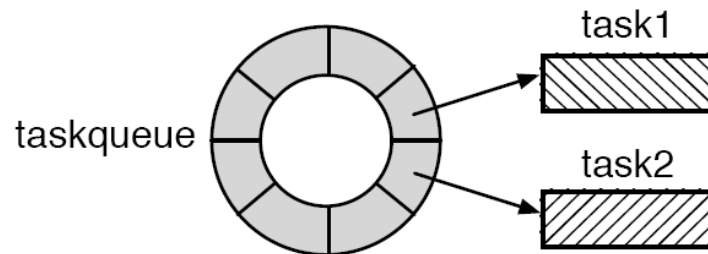
Examples

- Task queue: each task points to a bucket of data; the data in a bucket is read/updated before/after the task has been queued
- The task queue data structures must be updated in a consistent manner
- The updates to data in a bucket need not be co-ordinated with other updates

Functions to manipulate the task queue:

```
get_task(), put_task(), is_empty(),  
add_to_waiters(), is_everyone_waiting()
```

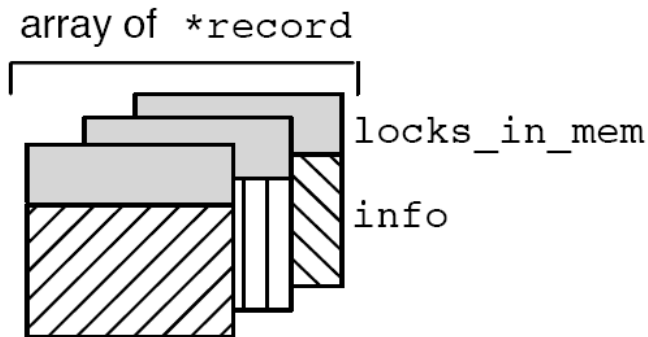
empty
 head
 tail
 num_waiters



```
color(&task1, sizeof(task1), GREEN)  
color(&task2, sizeof(task2), BLUE)  
color(&taskqueue, sizeof(taskqueue), RED)  
colorprop(&empty, sizeof(empty), &taskqueue)  
...
```

Examples

- MySQL data structure: an array of records
- In lock-based code, an element (`locks_in_mem`) in each record is protected by a single global lock, the remaining elements (`info`) in each record are protected by per-record local locks
- The `locks_in_mem` elements are all colored the same; the `info` in each record is assigned a per-record color



```
for(i=0; i < MAXREC, i++) {  
    color(&record[i]->locks_in_mem, ptrsize, RED)  
    color(&record[i]->info, infosize, RED+i+1)  
}
```


Entry/Exit Points

- The TM-begin/end should be placed around a series of accesses to elements of a single color
- TM-begin is introduced on the first access
- TM-end is introduced when that subroutine ends
- The programmer has to be aware of this exit policy to write correct programs (personal opinion: even DCS has to be code aware and reasoning is still somewhat non-local)

Interpreting the Exit Policy

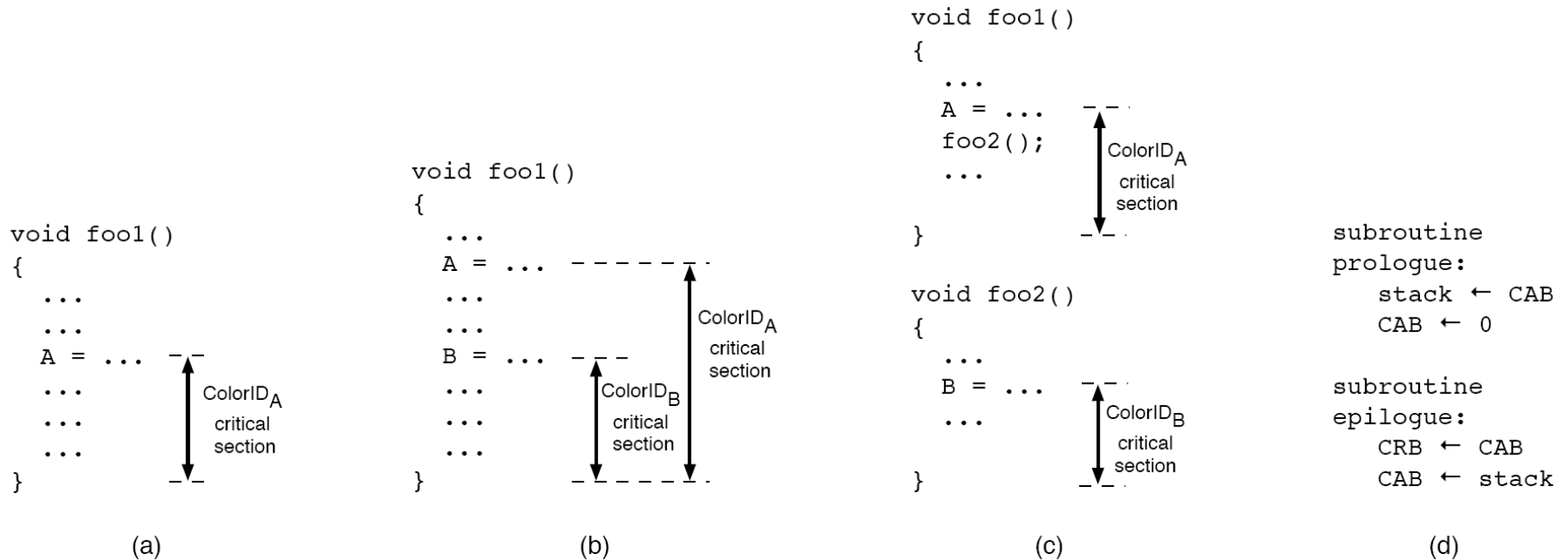


Figure 5. Illustration of the policy chosen in this paper to exit critical sections in Colorama and its implementation.

Interpreting the Exit Policy

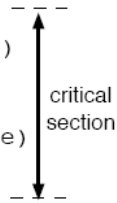
```
void htUpdate()
{
  ...
  lock(L)
  value = readHash(htPtr, key)
  value++
  writeHash(htPtr, key, value)
  unlock(L)
  ...
}
```

(a) Lock-based code

```
void htUpdate()
{
  ...
  value = readHash(htPtr, key)
  value++
  writeHash(htPtr, key, value)
  ...
}
```

(b) Colorama code

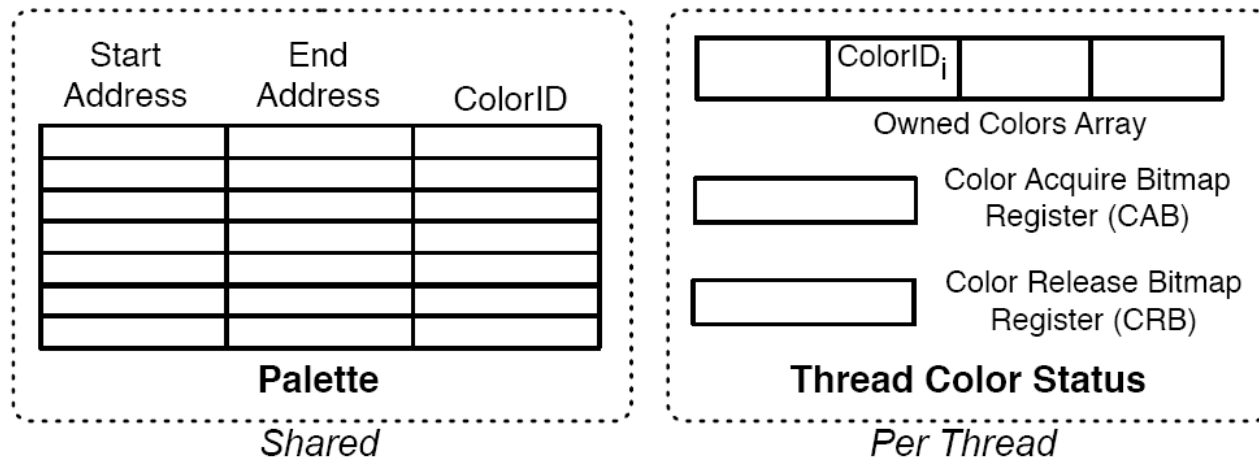
```
void htUpdate()
{
  ...
  colorcheck htPtr
  value = readHash(htPtr, key)
  value++
  colorcheck htPtr
  writeHash(htPtr, key, value)
  ...
}
```



(c) Colorama code with colorcheck

Hardware Requirements

- Hardware maintains a table of assigned colors and address ranges; a TM-begin is triggered every time a new color is accessed; the CAB and CRB help track the TM-ends that must be triggered on subroutine return



Bulk Disambiguation

- In a TM (or TLS) system, write addresses are broadcast at commit time; other nodes snoop and abort themselves if any of the write addresses match an address in their read set
- The process can be simplified by just sending a signature and comparing signatures to detect conflicts
- The process may lead to false conflicts; worsens performance, does not compromise correctness

Bloom Filters

- Bloom filters indicate set membership; if the bits corresponding to an address are set, the address may or may not belong to the set; if any of the bits is zero, the address is not part of the set

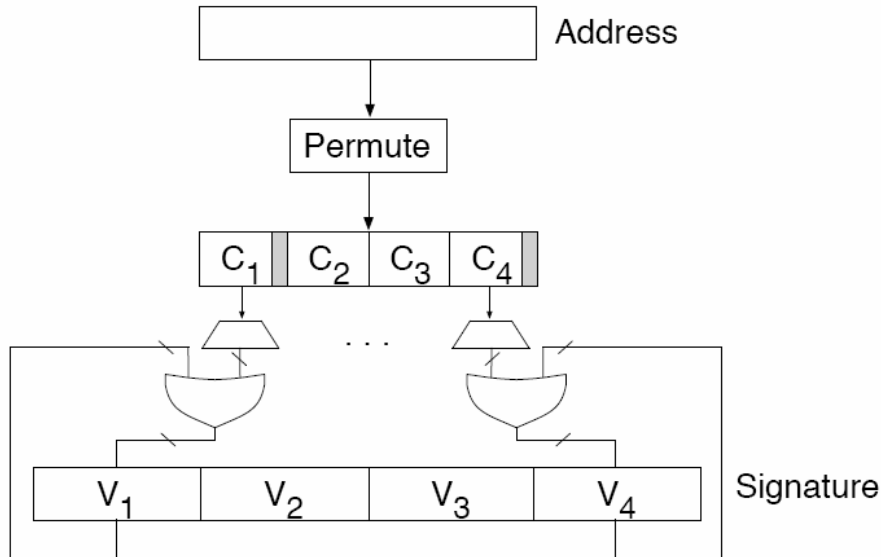


Figure 2. Adding an address to a signature.

- Set intersection is simply a bit-wise AND
- If any of the V_i fields is zero, the intersection yielded a null set
- If one of the C_i fields is the cache set index, on a conflict, we can invalidate the corresponding cache sets

Signature Results

Appl	Transaction Properties			False Positives	
	Rd Set Size (L)	Wr Set Size (L)	Dep Set Size (L)	Sq (%)	False Inv/Com (Avg)
cb	73.6	26.9	1.4	20.0	0.6
jgrt	67.1	22.1	1.3	22.1	0.2
lu	81.7	27.3	1.3	12.8	0.7
mc	51.6	17.6	1.9	9.8	0.1
moldyn	70.2	25.1	1.3	10.7	0.4
series	86.9	25.9	1.1	13.7	0.1
sjbb2k	41.6	11.2	1.4	7.7	0.1
Avg	67.5	22.3	1.4	13.8	0.3

UTM (Unbounded) and LTM (Large)

- Based on the premise that large transactions must also be handled gracefully (as long as the overheads are not too much when handling small transactions)
- UTM: complex design with logs and linked lists (complex because transactions/logs as large as virtual memory are being allowed)
- LTM: transactions/logs can be as large as physical memory and a transaction cannot persist beyond a context timeslice

LTM Architecture

- Does not use logs (lazy versioning): the cache maintains speculative write state; if the cache overflows, an overflow bit is set and the evicted cache line is placed in a hash table in memory
- If there is a cache miss and the overflow bit is set, memory must be looked up to find the data
- In essence, it gives the illusion that a really large cache exists and part of this cache is actually in memory; if the hash table fills up, the transaction is aborted and re-started after the OS allocates a larger hash table
- Employs eager conflict detection

Taxonomy

Table 1: A Transactional Memory (TM) taxonomy

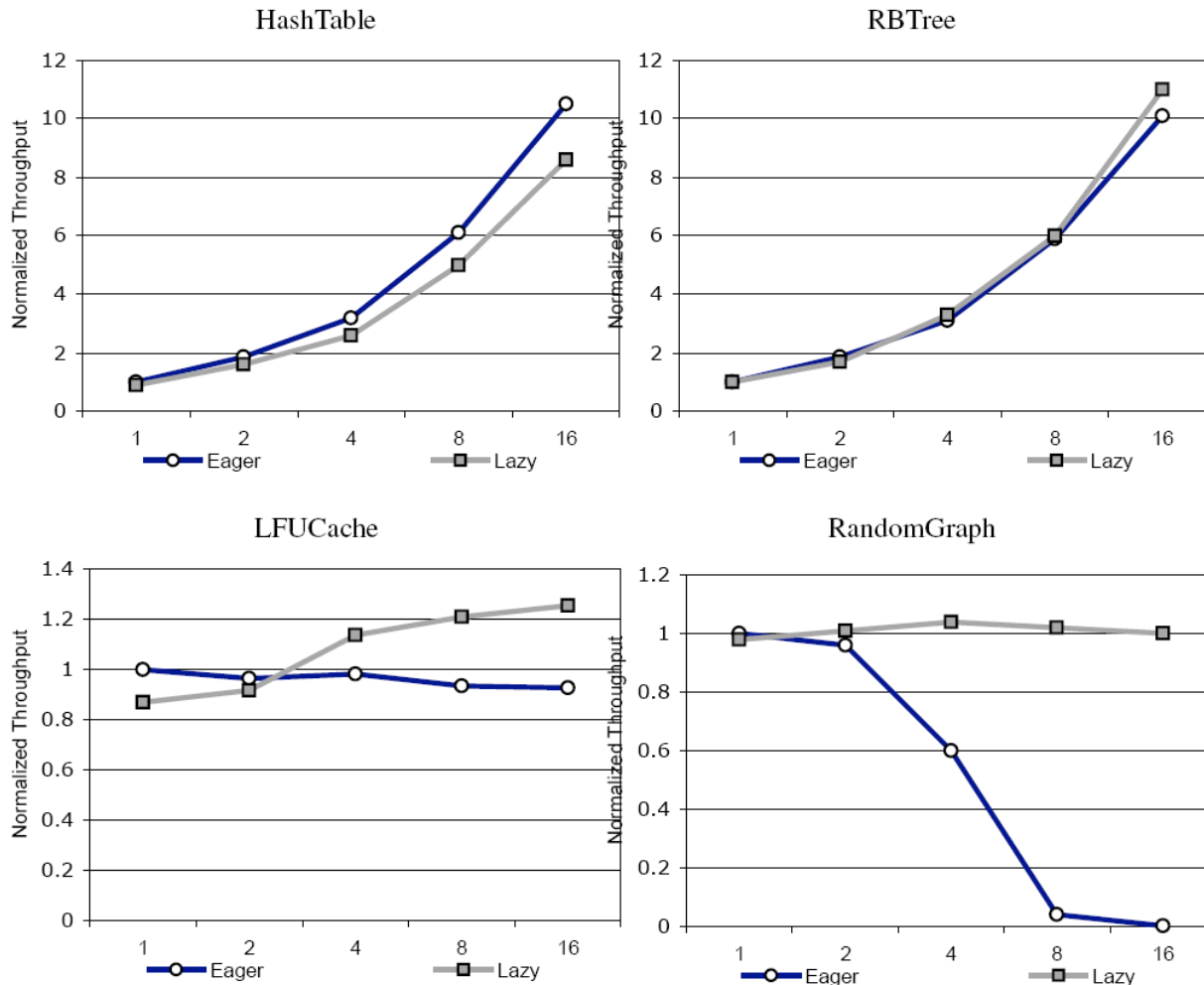
		Version Management	
		Lazy	Eager
Conflict	Lazy	OCC DBMSs [16] Stanford TCC [11]	none
	Eager	MIT LTM [2] Intel/Brown VTM [25] (on cache conflicts)	CCC DBMSs [6] MIT UTM [2] <i>LogTM [new]</i>

Rochester TM

- Hardware assists that speed up a software TM implementation
- Software TM imposes high overheads and policies (lazy/eager) can significantly impact performance
- Software-managed logs and software-managed checks for conflicts at commit time
- A hardware assist such as *alert-on-update* causes a cache coherence action to trigger a software handler to deal with the conflict rightaway

Policies in STM

- Lazy incurs more book-keeping overhead
- Eager incurs more conflicts and aborts



Title

- Bullet