

LockSTM: Mixing Locks and Transactional Memory

Nicholas Kuilema

Parallel problem

- New computers are multi-core
- Parallel programming is hard
 - Data races
 - Deadlock

Security Implications

- Unpredictable behavior
- Out of bounds reads/writes
- Bad pointer dereferences
- Difficult to exploit
- Crash is most likely result

Atomic Sections

- Concept to allow easier programming
- A block of code that runs as if in isolation
- Enforced with locks or transactions

Using Locks

- Acquire at the start, release at end
 - One global lock
 - Lock per memory location
- Automatic inference - LockPick
 - Creates colors (locks) for each section

Using a Software Transactional Memory (STM)

- Let all sections run simultaneously
- Log memory reads/writes
- At section completion, detect conflicts
- If no conflicts, continue execution
- otherwise, undo all the actions and re-run

Drawbacks

Locks

- Limits parallelism
- Not composable

STM

- Logging is slow
- Conflicts cause slowdown
- Can't STM every section because some actions can't be undone (non-revocable)

Hybrid Approach

- Use locks for non-revocable sections
- Use STM for all other sections
- Use locks to make sure STMed sections don't conflict with non-STMed sections

3 atomic section types

1. Non-revocable sections

Has non-revocable code

2. Hybrid sections

Shares memory accesses with
non-revocable sections

3. Pure STM sections

No sharing with non-revocable sections

Atomicity enforcement

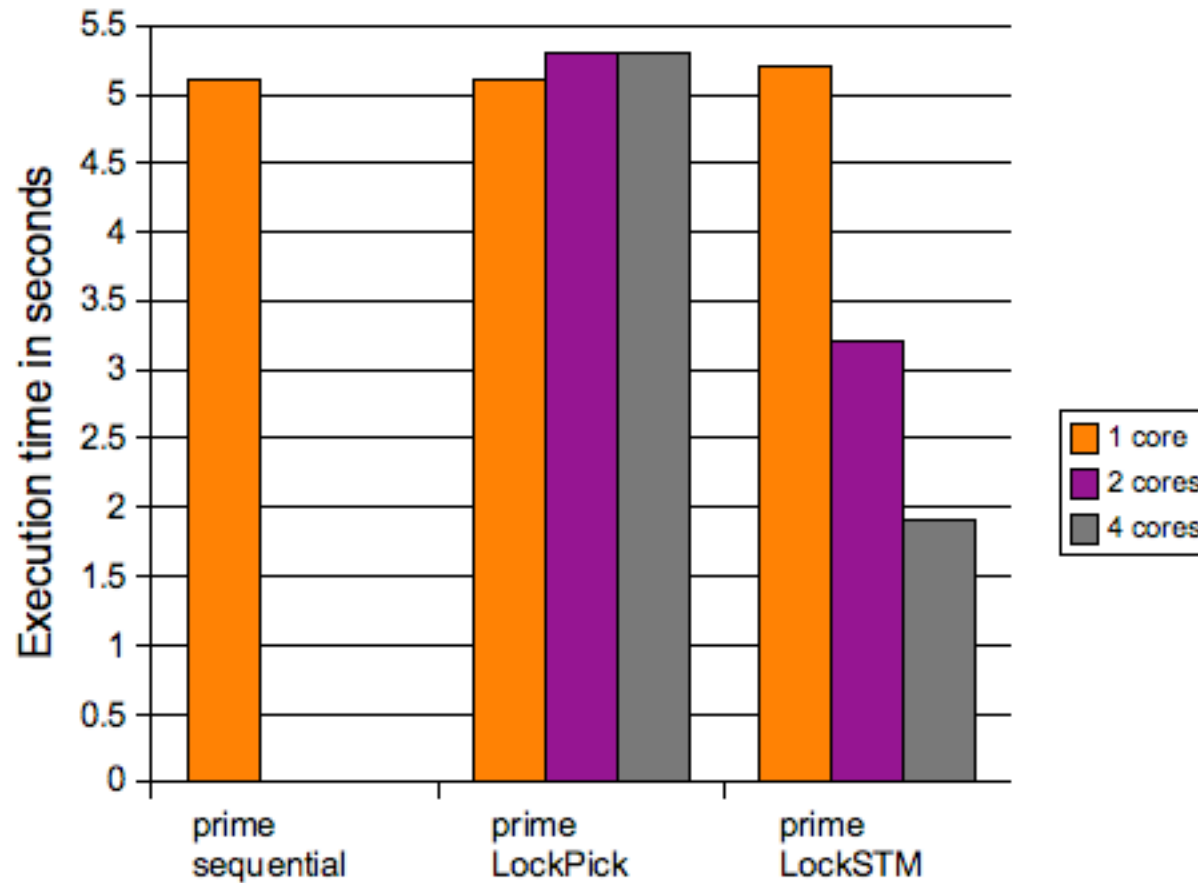
Non-revocable	<code>write_lock(l)</code>
Hybrid	<code>read_lock(l)</code> <code>STM_Start()</code>
Pure STM	<code>STM_Start()</code>

Implementation

- 1000 lines of OCaml as CIL modules
- Built off of LockSmith
- Source to source translation
- Ran experiments

Motivating Application

prime



Stanford Transactional Applications for Multi-Processing (STAMP)

		LockPick				LockSTM			
	Lines of code	Analysis Time (s)	Atomic Sections	Shared Variables	Colors	Non-Revocable sections	Hybrid sections	Pure STM sections	Colors
bayes	9150	49.9	15	51	1	11	4	0	1
genome	5675	18.0	5	40	1	4	1	0	1
kmeans	2043	0.7	3	13	3	0	0	3	0
labyrinth	5224	7.2	3	42	2	2	0	1	1
vacation	6970	6.9	3	16	1	3	0	0	1

Figure 7. LockPick and LockSTM analysis on STAMP

STAMP performance

Program	1 core	2 core	4 core
bayes	45.5		
bayes LockPick	44.7	55.5	51.7
bayes LockSTM	44.8	51.7	52.4
bayes TL2	42.9	52.8	43.2
genome	5.8		
genome LockPick	5.83	5.78	6.23
genome LockSTM	5.62	7.14	7.01
genome TL2	7.88	4.54	2.82
kmeans	16.3		
kmeans LockPick	18.5	11	7.7
kmeans LockSTM	34.6	27.1	16.2
kmeans TL2	40.1	22.5	14.3
labyrinth	3.58		
labyrinth LockPick	3.54	3.55	3.55
labyrinth LockSTM	3.58	3.6	3.63
labyrinth TL2	5.41	2.92	2.16
vacation	23.3		
vacation LockPick	23.6	42.9	45.5
vacation LockSTM	23.5	45.6	46.7
vacation TL2	98.4	69.4	56.2

Figure 8. STAMP runtimes in seconds

Conclusion

- Allows STM to be used on all applications
- Acceptable performance
- Future work, run on more applications to get a better understanding of performance