

Digging parallelism out of a highly-conflicting workload

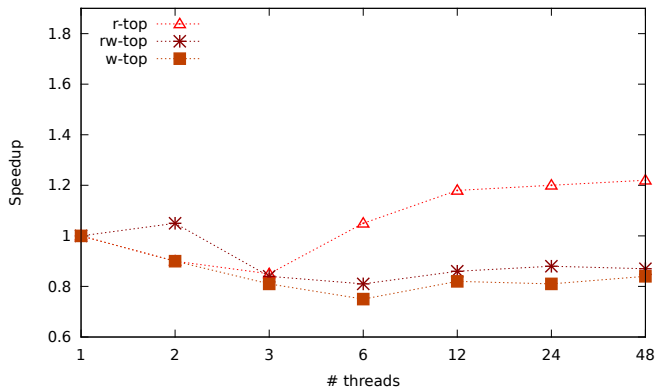
Nuno Diegues João Cachopo

nmdl, joao.cachopo@ist.utl.pt

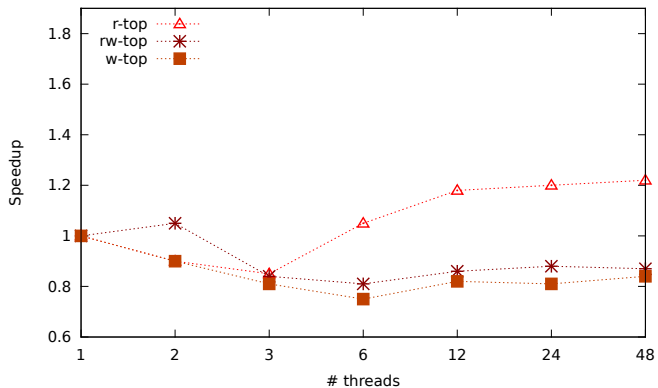
INESC-ID/Technical University of Lisbon

April 10, 2012

Problem

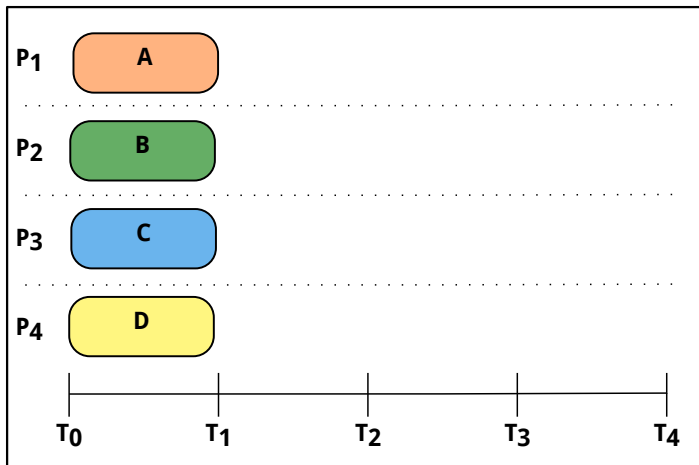


Problem

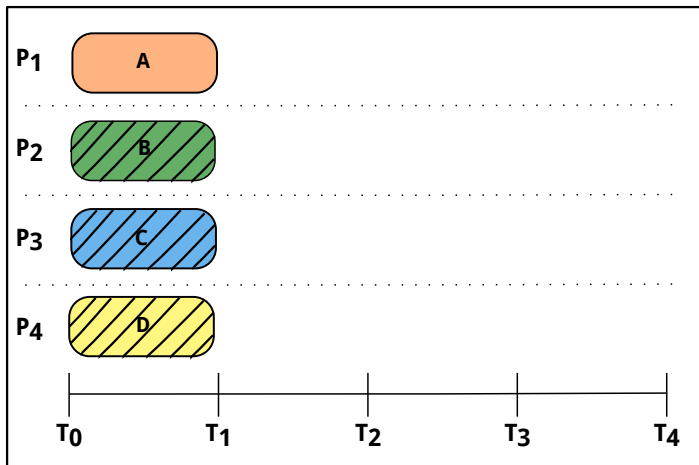


What is going wrong?

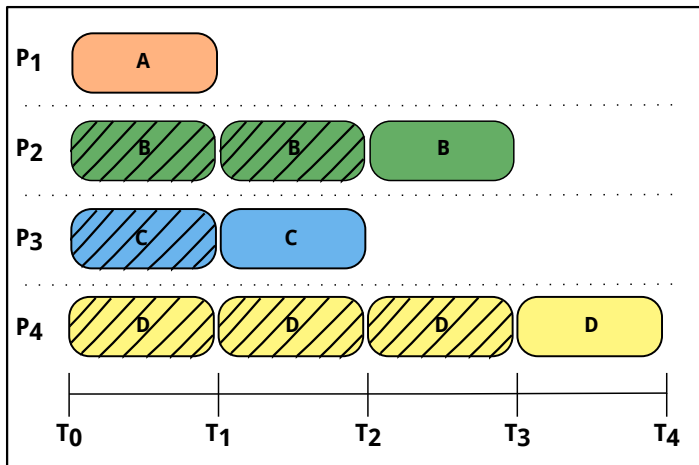
Problem



Problem



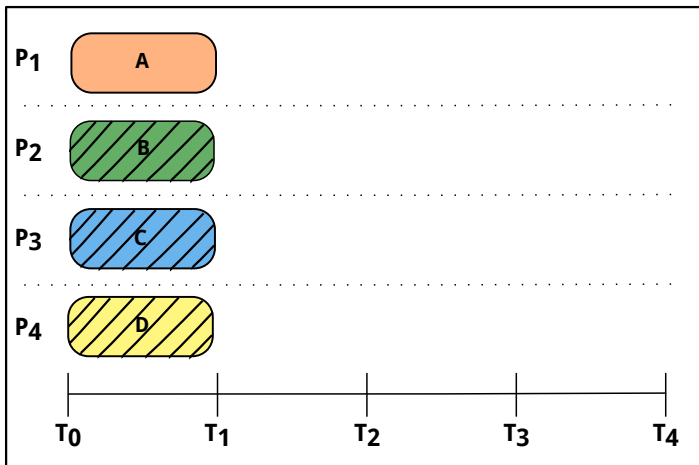
Problem



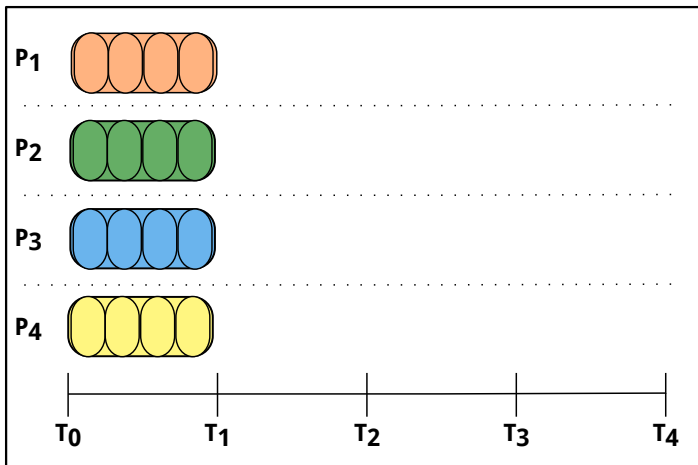
Optimistic assumption defrauded

Explore parallelism within transactions

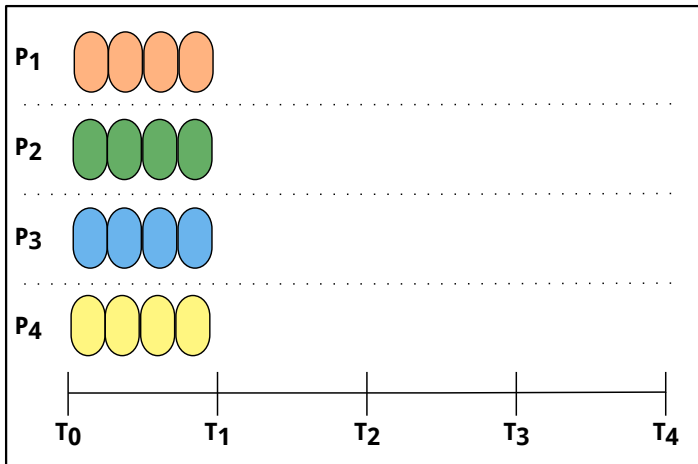
Solution



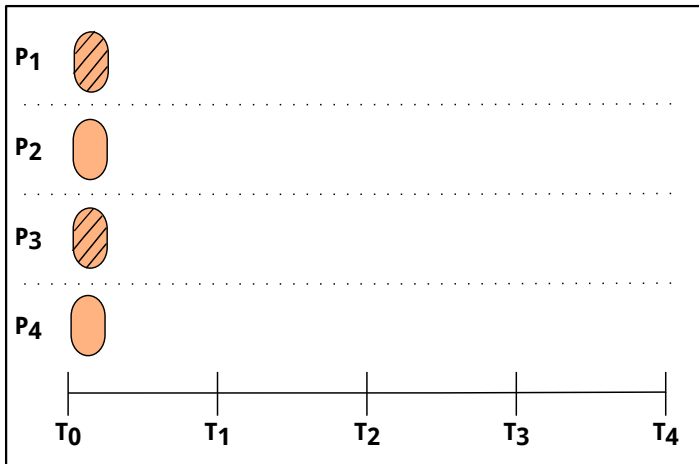
Solution



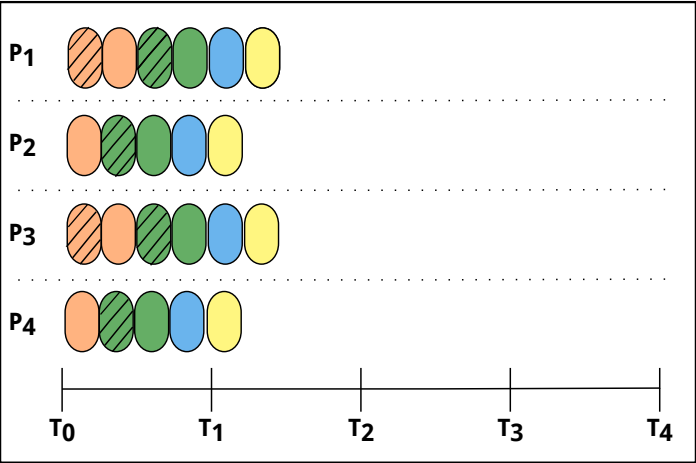
Solution



Solution



Solution



Outline

- 1 Parallel nesting

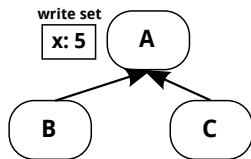
Outline

- ① Parallel nesting
- ② Transaction scheduling

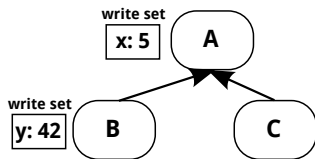
Naive parallel nesting



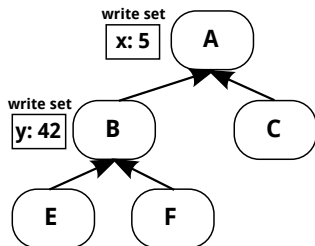
Naive parallel nesting



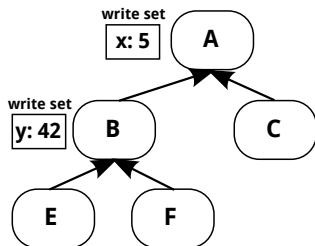
Naive parallel nesting



Naive parallel nesting

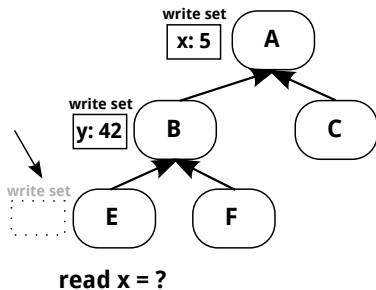


Naive parallel nesting

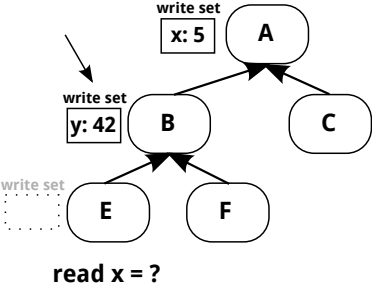


read x = ?

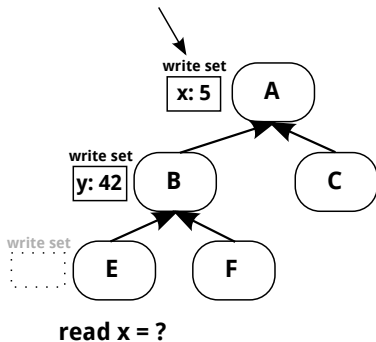
Naive parallel nesting



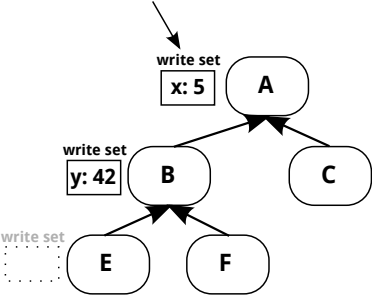
Naive parallel nesting



Naive parallel nesting

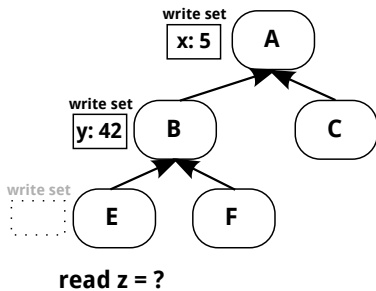


Naive parallel nesting

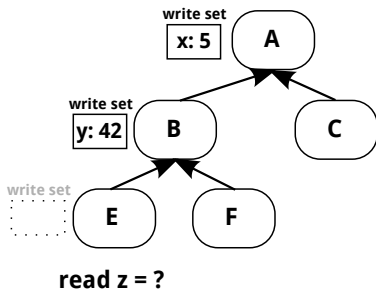


read x = 5 read-after-write!

Naive parallel nesting



Naive parallel nesting

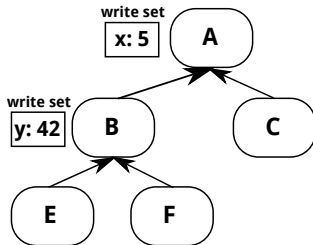


Read operation worst-case: $O(d)$

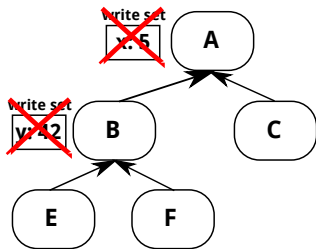
Read-after-writes in STMbench7, Lee-TM and Vacation

test	reads ($\times 10^3$)	raws ($\times 10^3$)	%
bench7-r-notrav	8000	31	0
bench7-rw-notrav	9000	34	0
bench7-w-notrav	5000	19	0
bench7-r	83000	45000	54
bench7-rw	109000	65000	59
bench7-w	127000	71000	56
lee-mainboard	507000	3	0
lee-memboard	281000	2	0
lee-sparselong	67000	0	0
lee-sparseshort	1000	0	0
vac-reservations	84000	0.2	0
vac-deletions	33000	600	1.8

Shared write-set per nesting tree

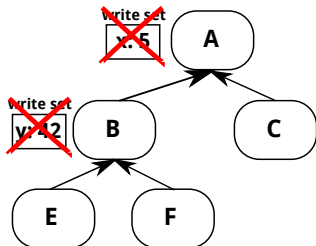


Shared write-set per nesting tree

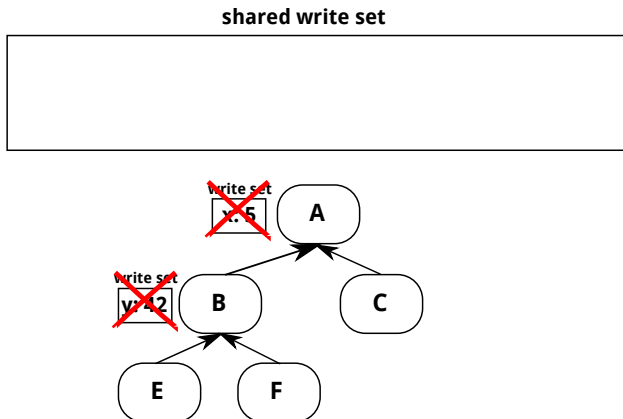


Shared write-set per nesting tree

shared write set

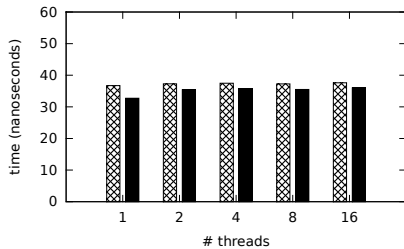
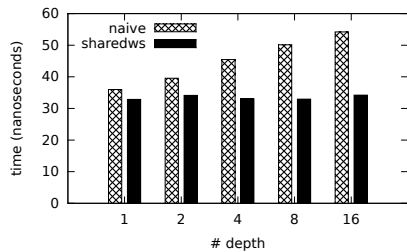


Shared write-set per nesting tree

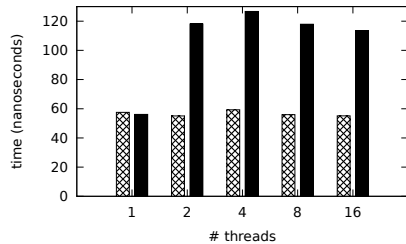
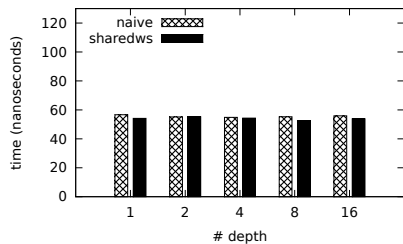


Nuno Diegues and João Cachopo. *Parallel nesting in a lock-free multi-version software transactional memory*. *TRANSACT*, 2012.

Profiling of read operations



Profiling of **write** operations

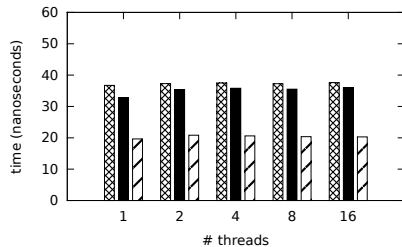
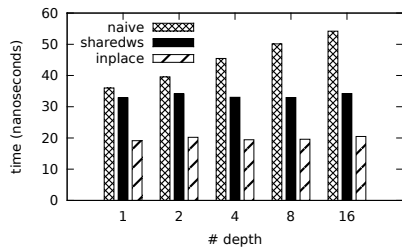


Writes in-place

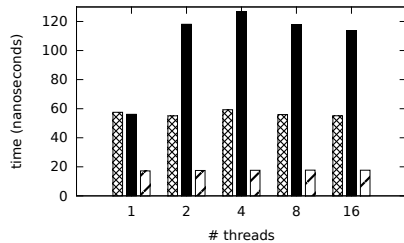
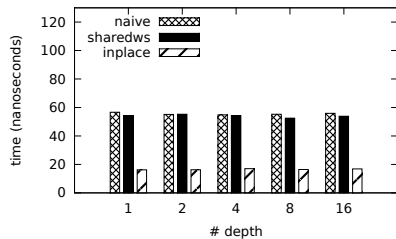
Inplace: What was improved?

- Cheaper writes
- Reads still independent of depth
- Reduced commit-time:
 $O(\#children) \ll O(size(writeSet) + size(readSet))$

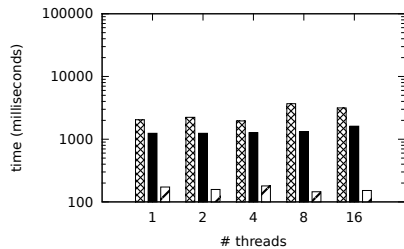
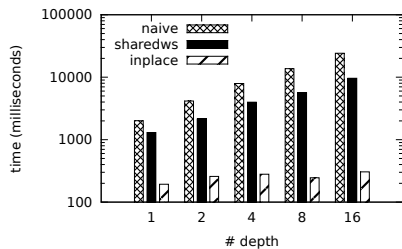
Profiling of read operations



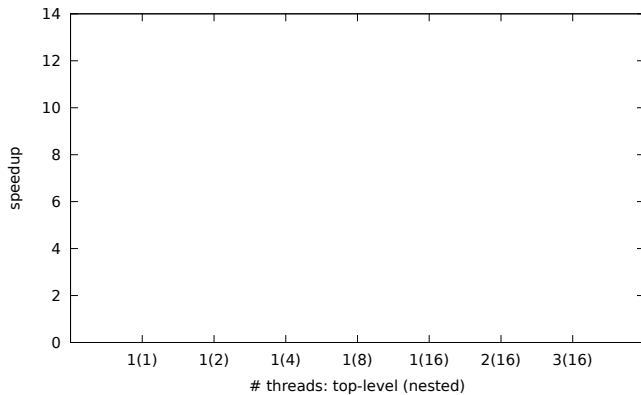
Profiling of **write** operations



Profiling of **commit** operations



Evaluation



Evaluation

- ① Vacation from STAMP
- ② STMBench7
- ③ Lee-TM

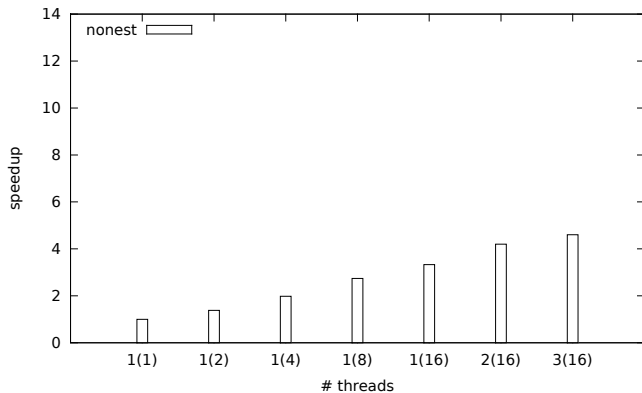
Evaluation

- ① Vacation from STAMP
- ② STMBench7

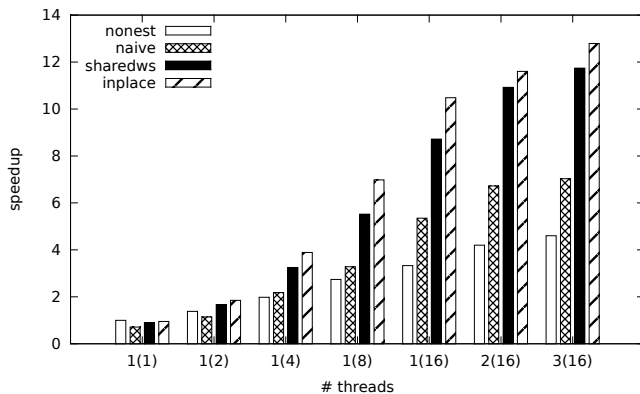
Evaluation

- ① Vacation from STAMP
- ② STMBench7

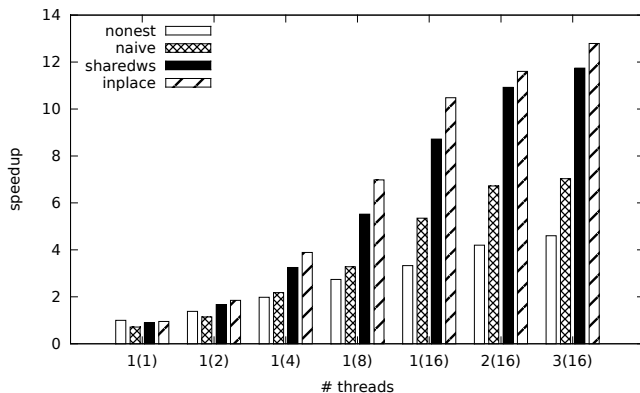
Vacation - **high** contention



Vacation - **high** contention

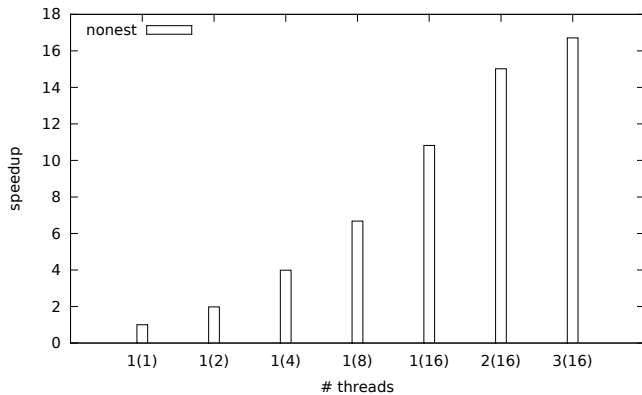


Vacation - **high** contention

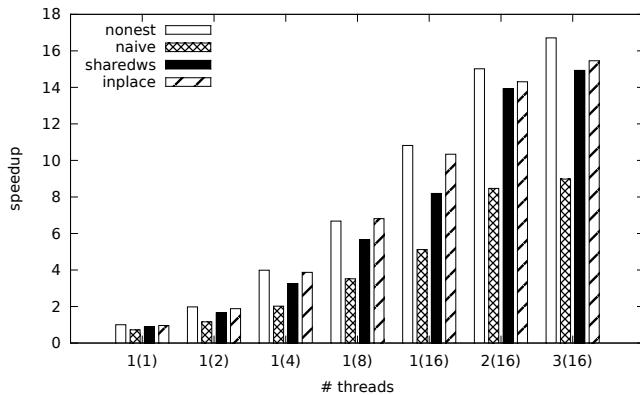


From 4.6 to 12.8 speedup

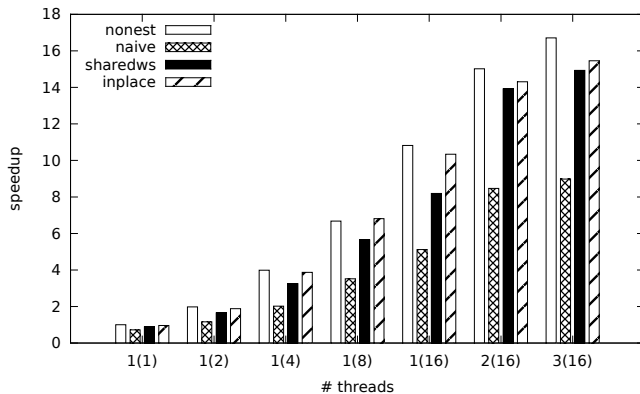
Vacation - **low** contention



Vacation - **low** contention



Vacation - **low** contention

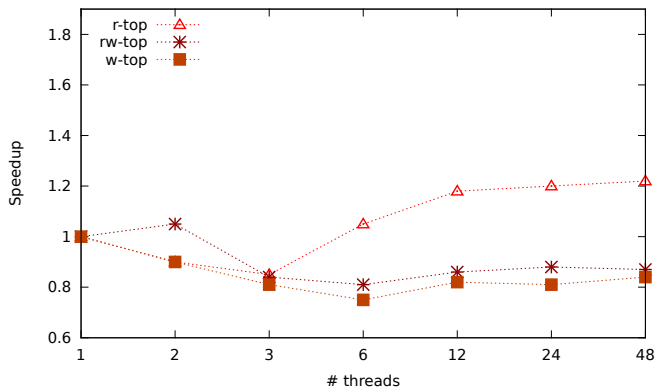


Overhead is visible

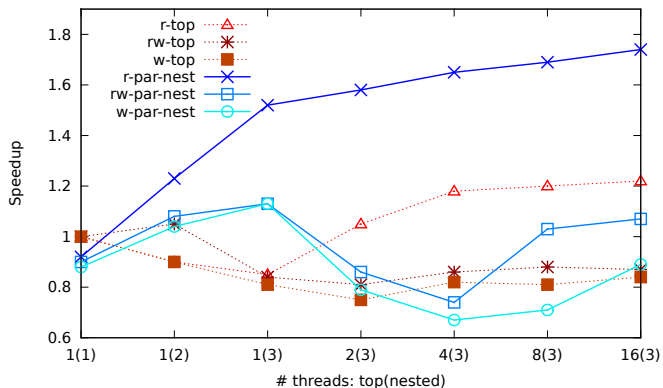
Evaluation

- ① Vacation from STAMP
- ② STMBench7

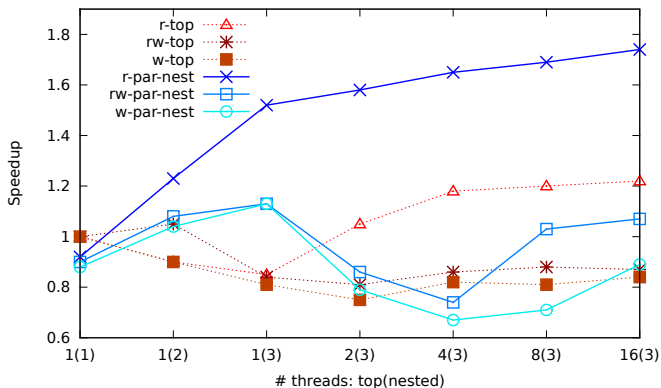
Parallel nesting in the wild



Parallel nesting in the wild



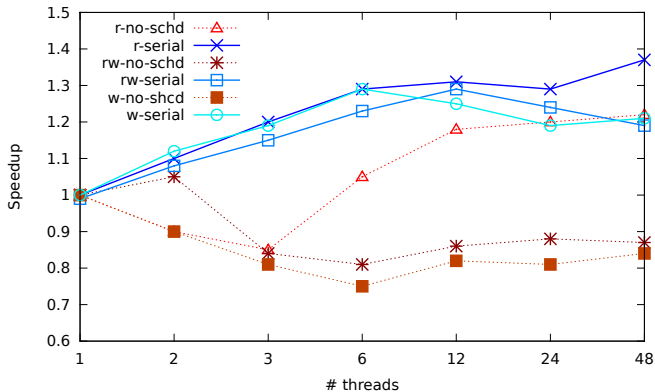
Parallel nesting in the wild



Missing one of the requirements in the motivation

Transaction Scheduling

Scheduling results - no nesting

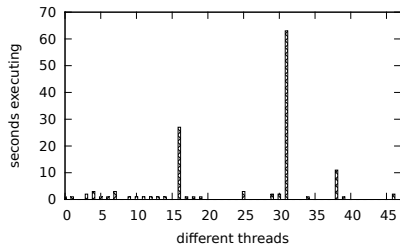
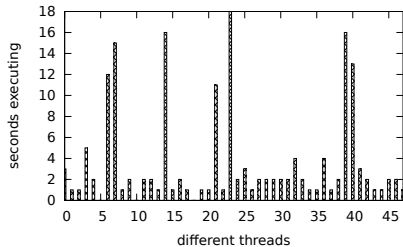


Read-dominated: 1.22 to **1.37**

Read-write: 0.87 to **1.19**

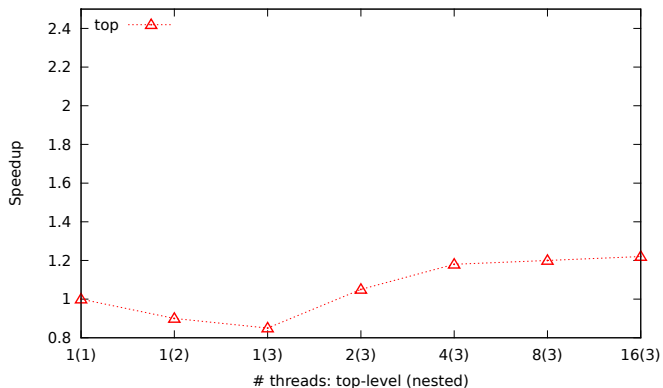
Write-dominated: 0.84 to **1.21**

Time spent computing by each thread

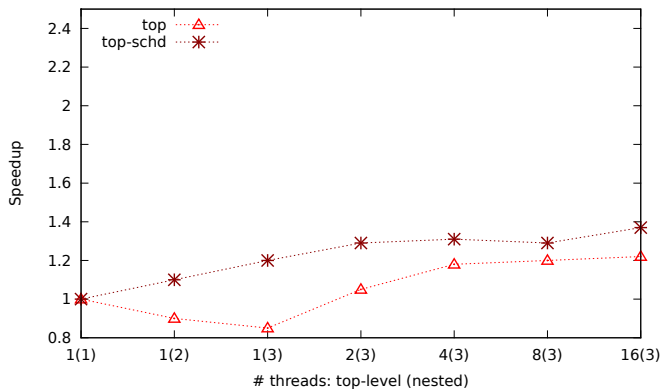


Side effect: processors are idle (rather than computing conflicting transactions)

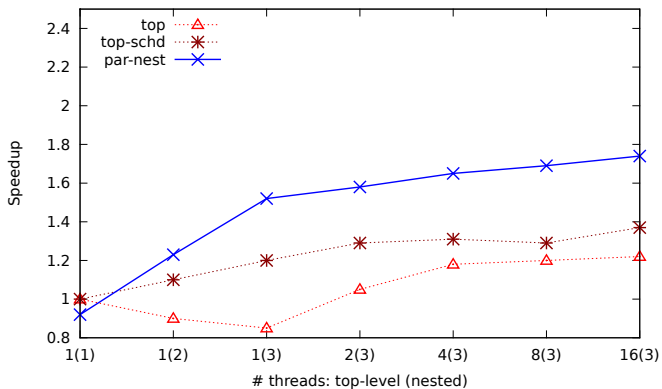
STMBench7 - read-dominated



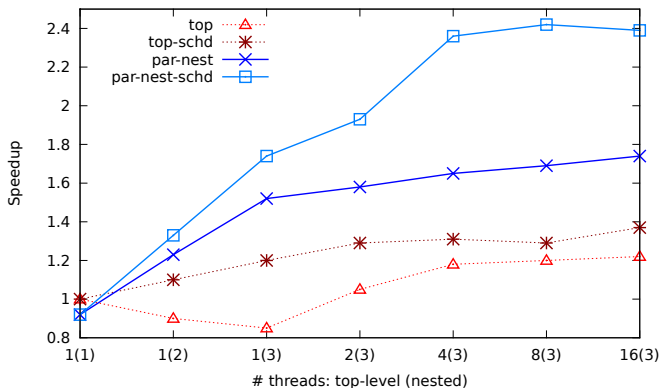
STMBench7 - read-dominated



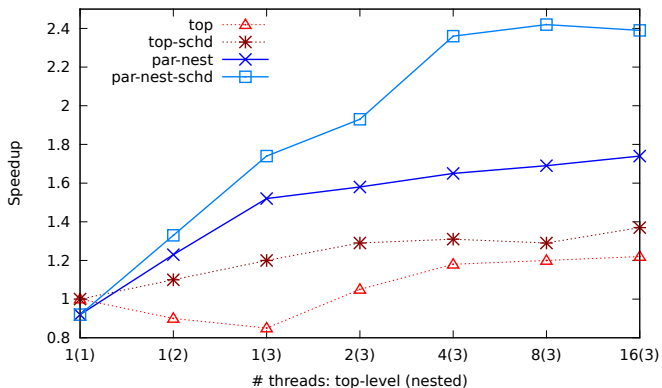
STMBench7 - read-dominated



STMBench7 - read-dominated

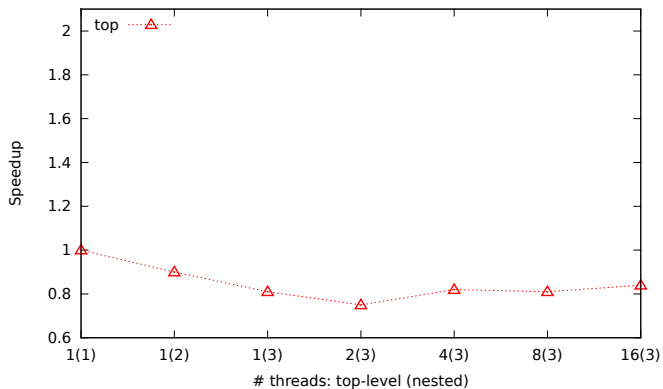


STMBench7 - read-dominated

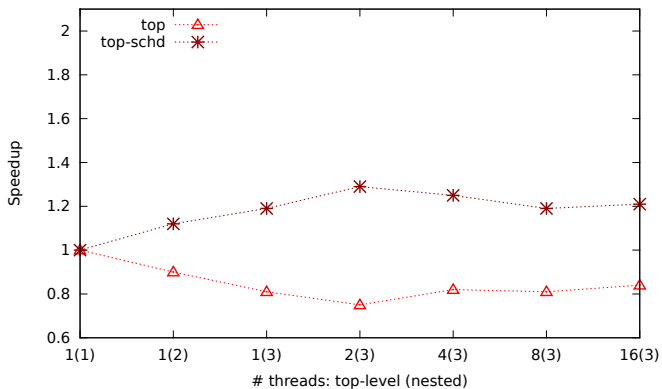


From 1.22 to 2.39 speedup.

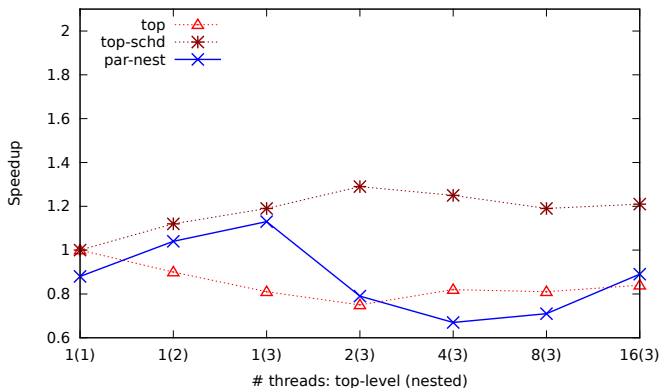
STMBench7 - write-dominated



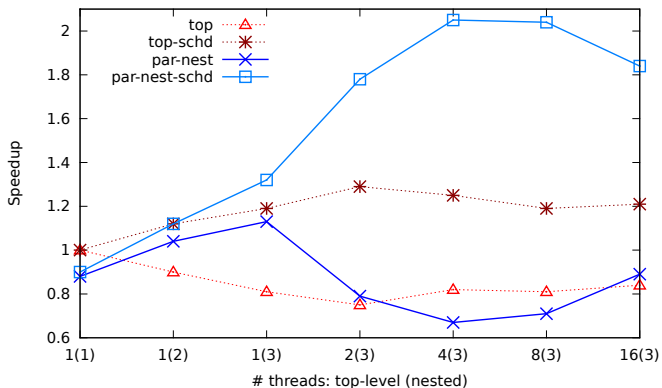
STMBench7 - write-dominated



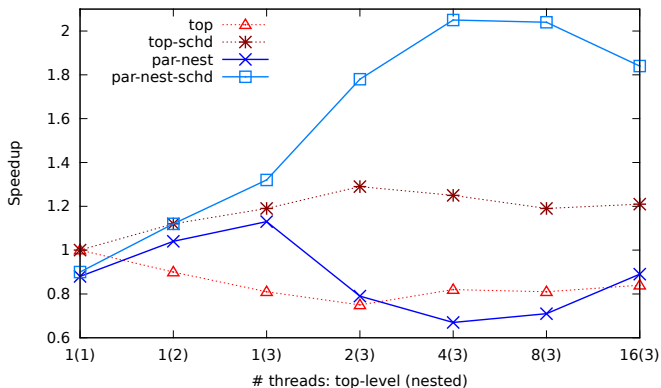
STMBench7 - write-dominated



STMBench7 - write-dominated



STMBench7 - write-dominated

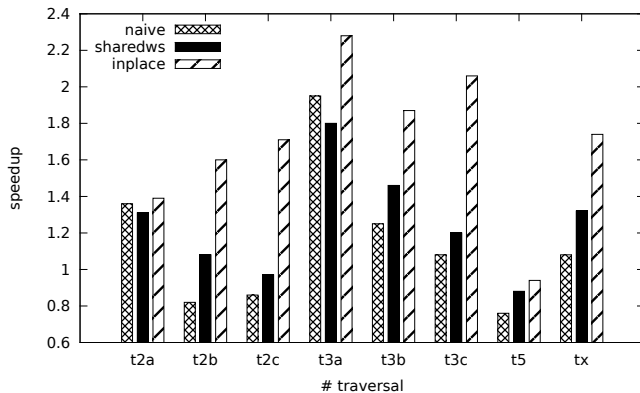


From 0.84 to 1.84 speedup.

Thank you

Questions?

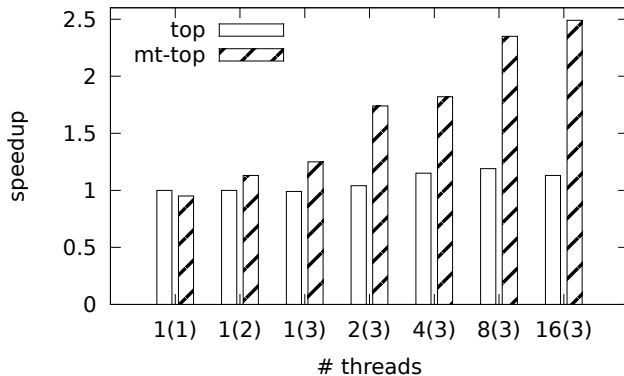
STMBench7



Embarrassingly parallelizable transaction

- More than one thread in the same transaction
- No validation required
- Accesses performed as if it was a top-level transaction

Lee-TM - mainboard



120% increase in performance