Accelerating conflict-intensive applications

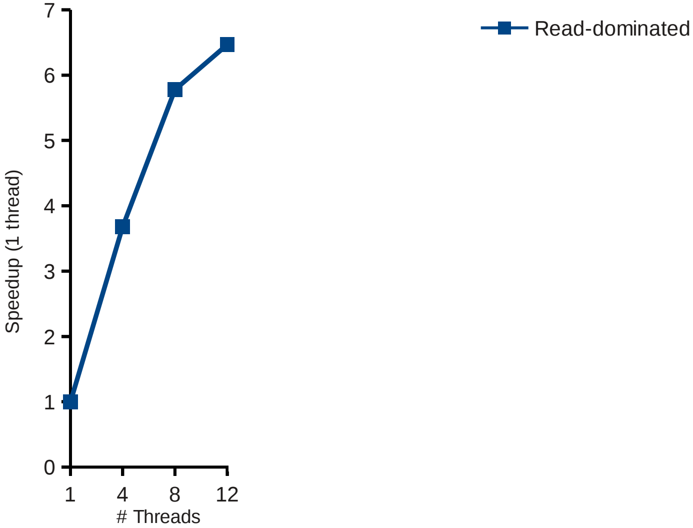
Hugo Rito, João Cachopo

INESC-ID
Instituto Superior Técnico, Portugal

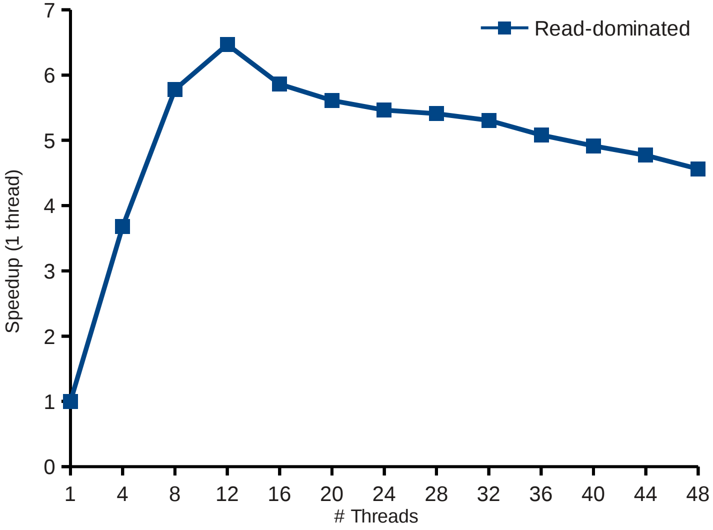
April 10, 2012

Overview

Overview

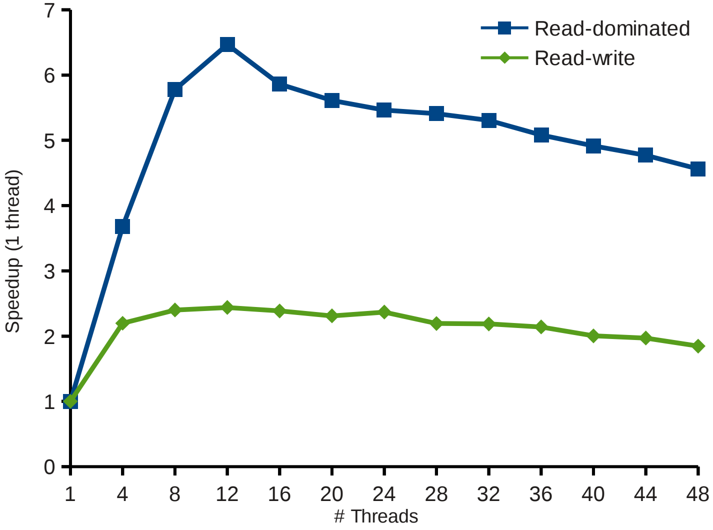


Overview



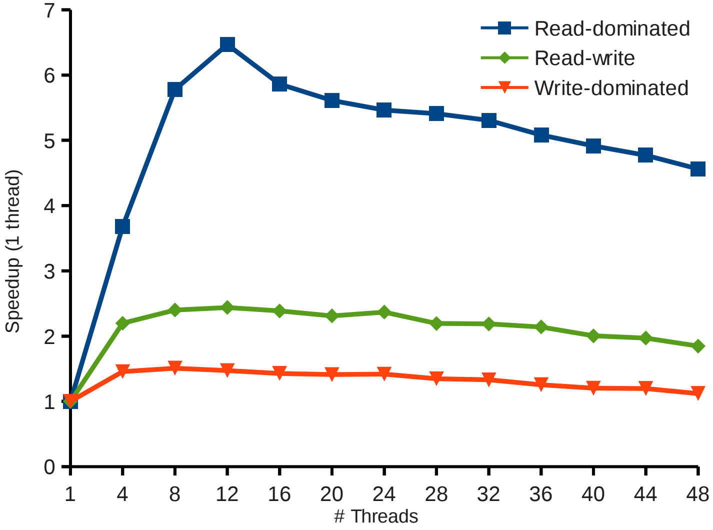
Read-dominated
Speedup (1 thread)
0
1
2
3
4
5
6
7
1
4
8
12
16
20
24
28
32
36
40
44
48
Threads

Overview



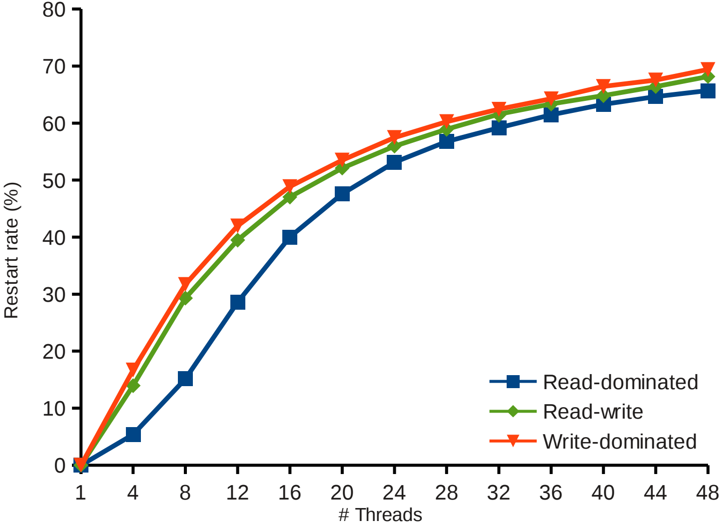
Read-dominated
Read-write
Speedup (1 thread)
Threads
0
1
2
3
4
5
6
7
1
4
8
12
16
20
24
28
32
36
40
44
48

Overview

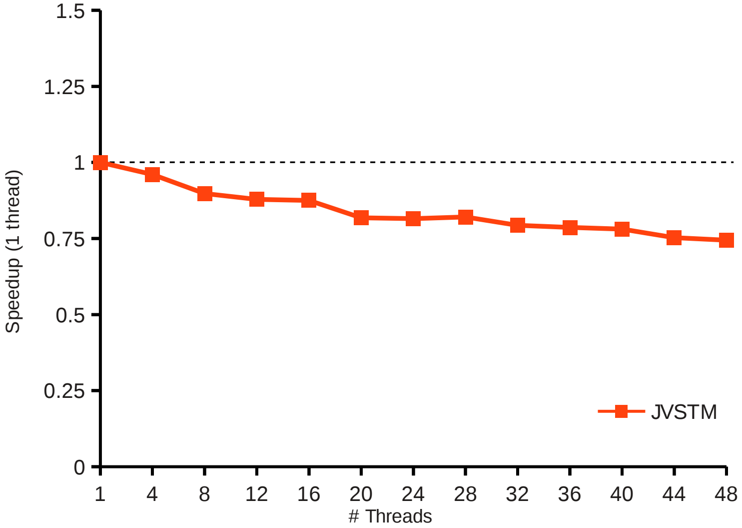


Read-dominated
Read-write
Write-dominated
Speedup (1 thread)
Threads
0
1
2
3
4
5
6
7
1
4
8
12
16
20
24
28
32
36
40
44
48

Overview


80
70
60
50
40
30
20
10
0
Restart rate (%)
1
4
8
12
16
20
24
28
32
36
40
44
48
Threads
Read-dominated
Read-write
Write-dominated

STMBench7: Write-dominated + Long-traversals



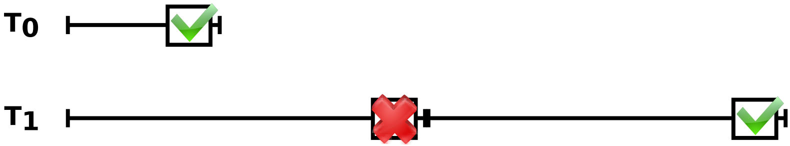
Problem

Single-threaded

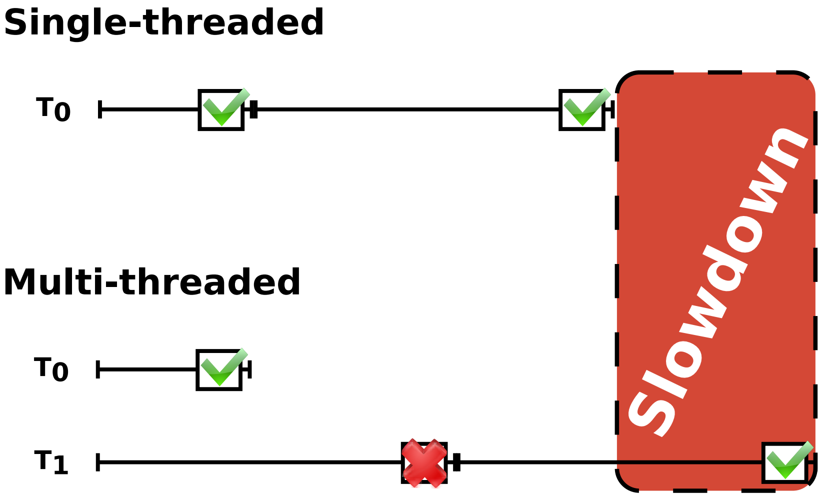
Problem

Single-threaded

Multi-threaded



Problem



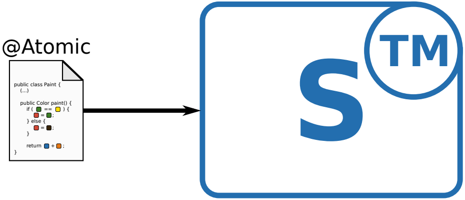
Problem

- Optimistic $\rightarrow$ Conflicts $\rightarrow$ Restarts

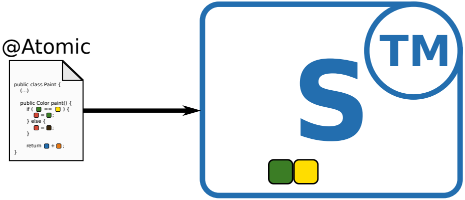
Problem

- Optimistic $\rightarrow$ Conflicts $\rightarrow$ Restarts
- Pessimistic $\rightarrow$ Limited concurrency $\rightarrow$!Solution

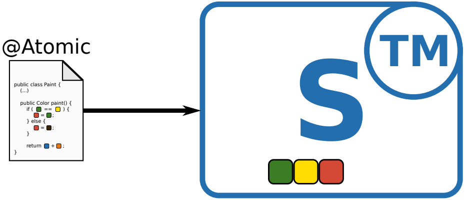
Transaction execution



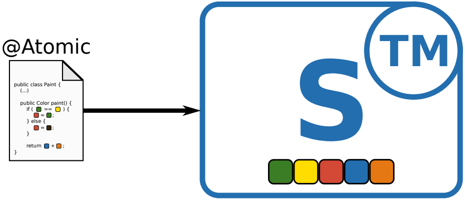
Transaction execution



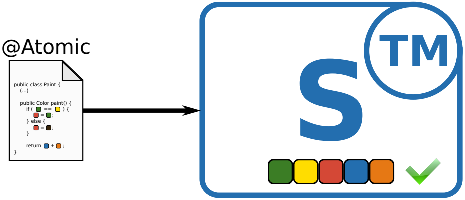
Transaction execution



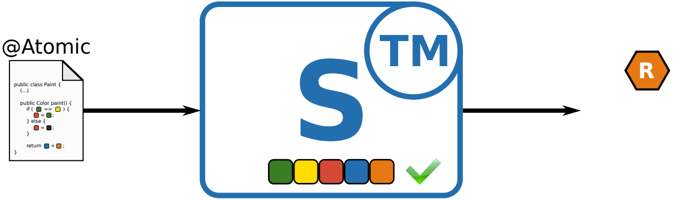
Transaction execution



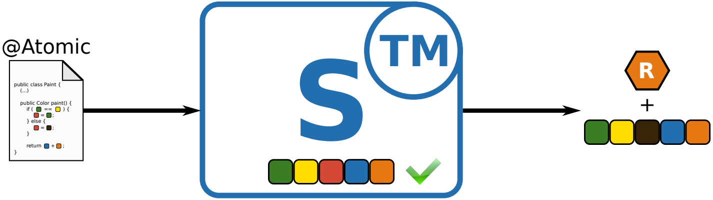
Transaction execution



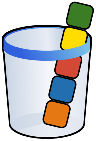
Transaction execution



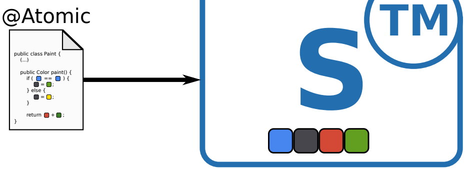
Transaction execution



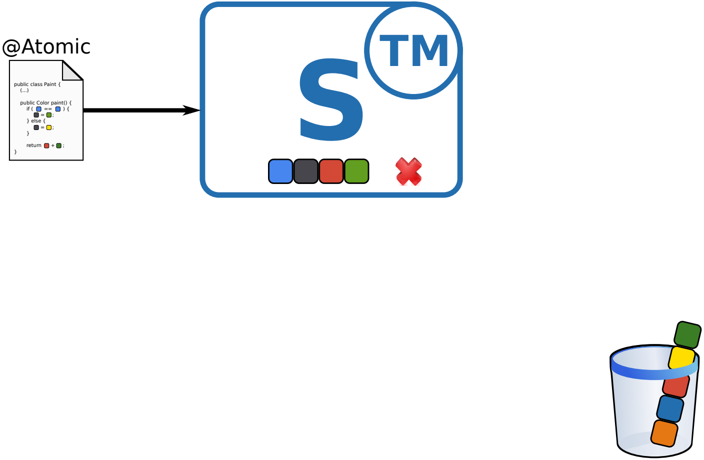
Transaction execution



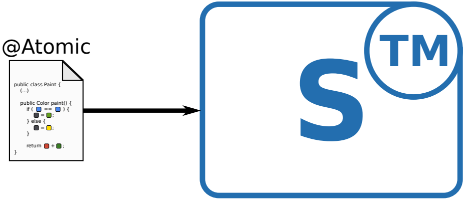
Transaction execution



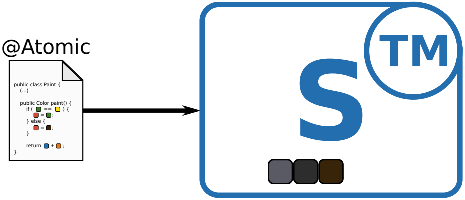
Transaction execution



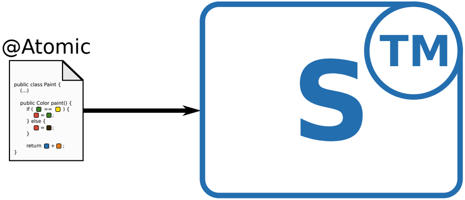
Transaction execution

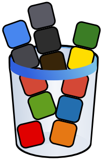


Transaction execution

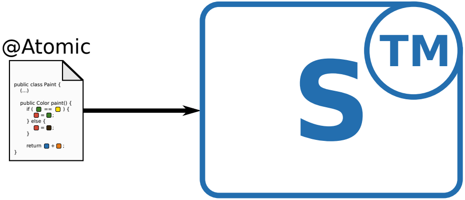


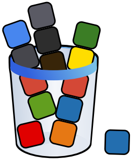
Transaction execution



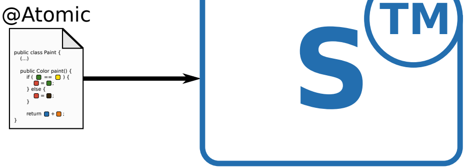


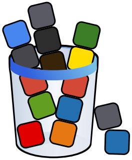
Transaction execution



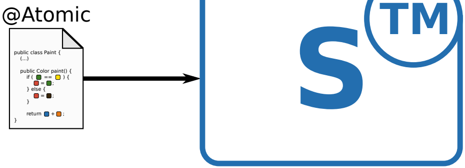


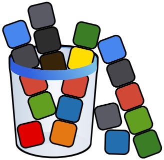
Transaction execution



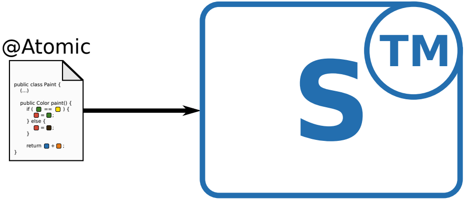


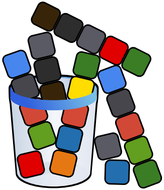
Transaction execution



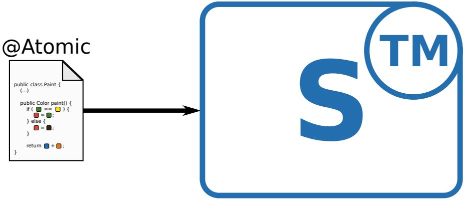


Transaction execution



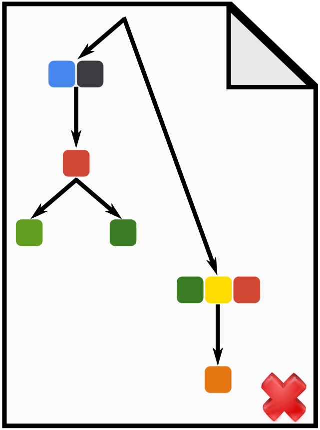


Transaction execution

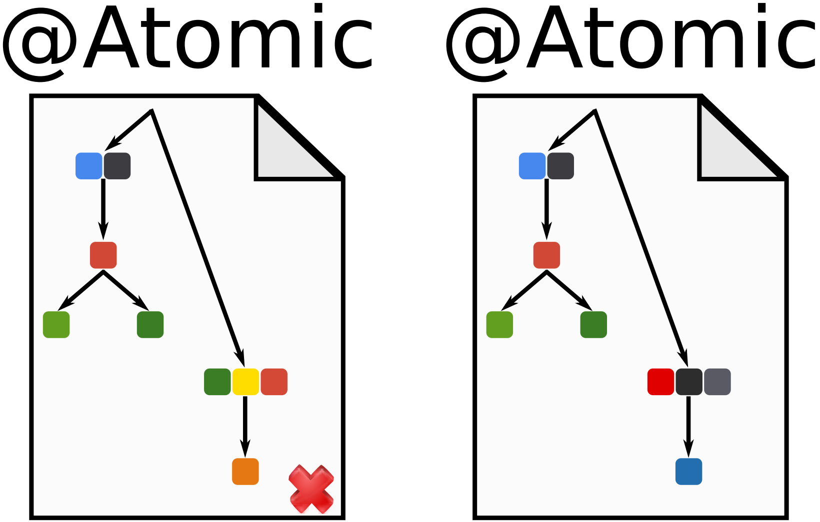


Similar (re)executions

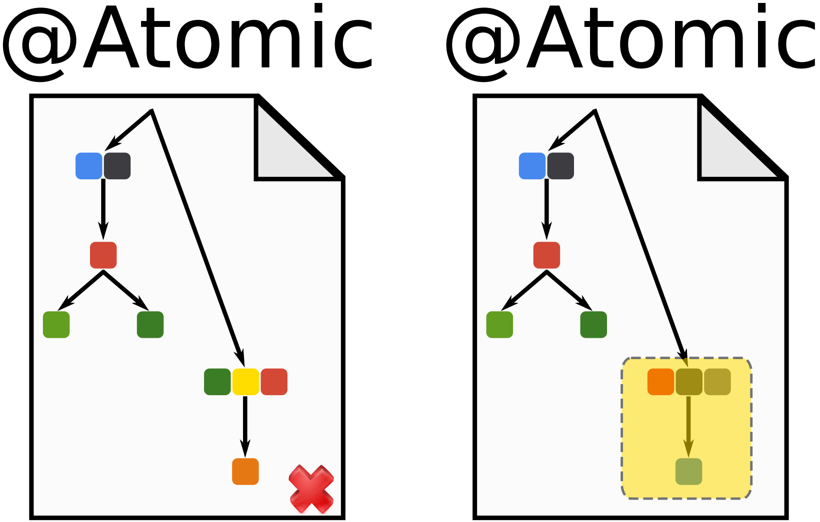
@Atomic



Similar (re)executions



Similar (re)executions



Green STM

Green STM

Green STM

Green STM

Recycle

(faster reexecutions)

Green STM

Recycle

(faster reexecutions)

Reuse

(faster executions)

Green STM

Recycle

(faster reexecutions)

Reduce

(overheads)

Reuse

(faster executions)

Memoization

Memoization ♻

♻ **also known as function caching**

Memoization

- Transparent

Memoization

- Transparent
- Benefits from reexecutions

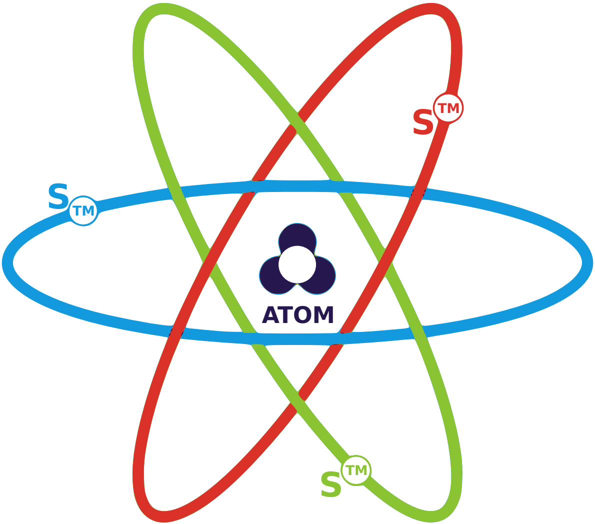
Memoization

- Transparent
- Benefits from reexecutions
- Free

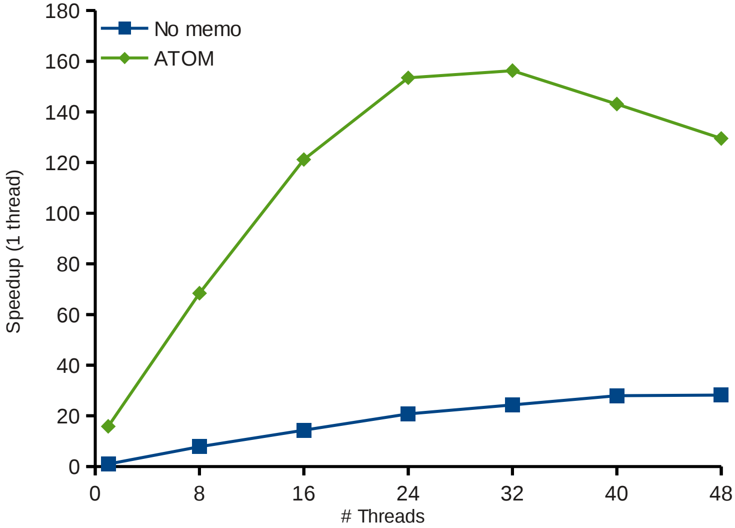
Memoization

- Transparent
- Benefits from reexecutions
- Free (almost)

Automatic **T**ransaction-**O**riented **M**emoization



STMBench7: Read-dominated

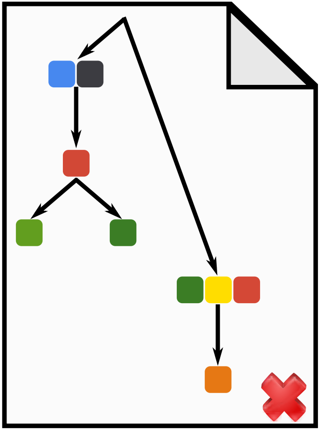


STM

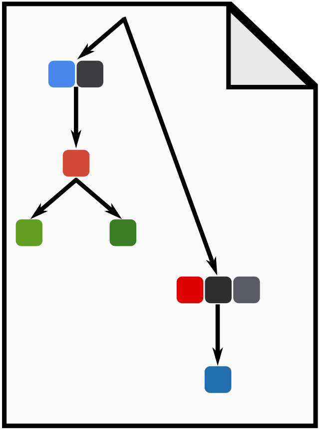
STM + Memo = Green STM

STM + Memo = Green STM

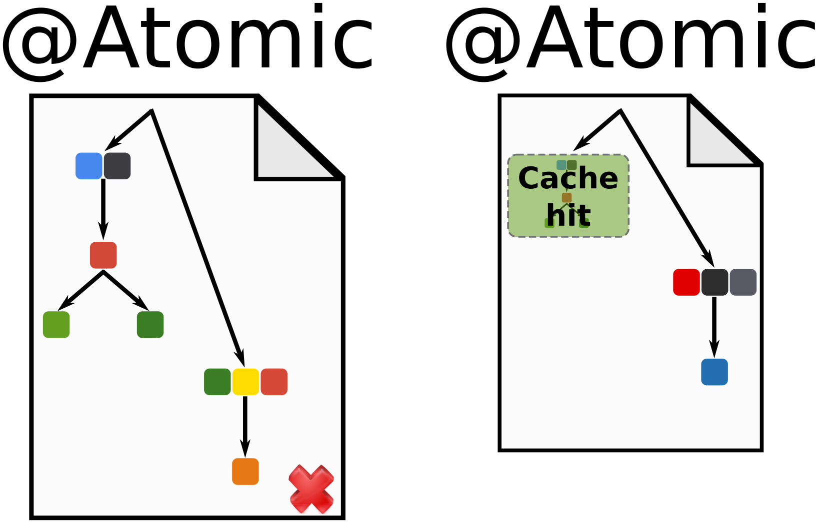
@Atomic



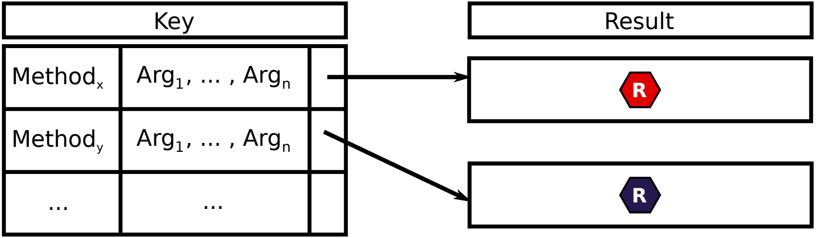
@Atomic



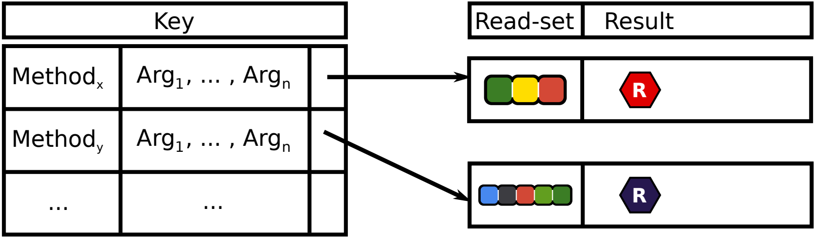
STM + Memo = Green STM



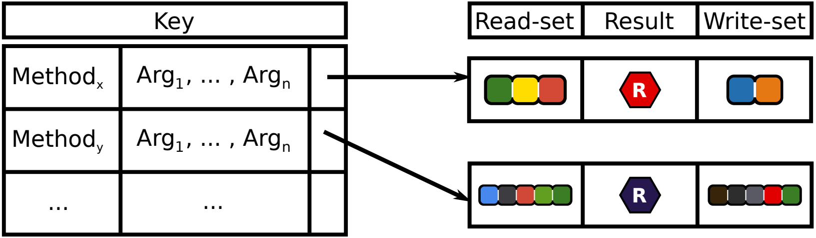
Per-transaction memo-cache



Per-transaction memo-cache



Per-transaction memo-cache



Green STM implementation

Recycle
(faster reexecutions)

Reduce
(overheads)

Reuse
(faster executions)

Green STM implementation

Per-transaction memo-cache

Reduce
(overheads)

Reuse
(faster executions)

Green STM implementation

Per-transaction memo-cache

Reduce
(overheads)

Central memo-cache

Green STM implementation

Per-transaction memo-cache

Sibling thread

Central memo-cache

Green STM implementation

Per-transaction memo-cache

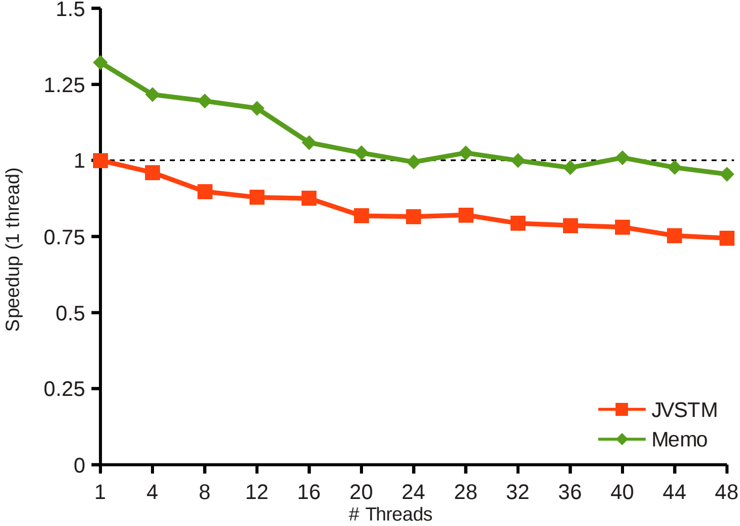
Sibling thread

Central memo-cache

Experimental setup

- STMBench7 benchmark
- 4xAMD Opteron 6168 (48 cores)
- 1 $\rightarrow$ 48 threads
- Total 1200 ops

STMBench7: Write-dominated + Long-traversals



Conclusions

- STMs collect lots of information
- Accelerate similar executions
- Memoization

Thank you,

QUESTIONS