What complexity for TM performance?

Aleksandar Dragojević
TM performance

- Outperform sequential code
  - with low thread counts
- Scalable
  - improve with more threads
Performance studies

STAMP: Stanford Transactional Applications for Multi-Processing
Cao Minh Chi, Chung JaeWoong, Kozyrakis Christos, Olukotun Kunle
Performance studies

Software Transactional Memory: Why is it only a Research Toy?
Calin Cascaval, Colin Blundell, Maged M. Michael, Harold W. Cain, Peng Wu, Stefanie Chiras, Siddhartha Chatterjee
Performance studies

Why STM can be more than a Research Toy?
Aleksandar Dragojević, Pascal Felber, Vincent Gramoli, Rachid Guerraoui
(not published)
Current performance

Vacation

![Graph showing current performance with speedup on the y-axis and threads on the x-axis, indicating no change in speedup with increasing threads.](image)
Current performance

Vacation

Speedup

Threads

IBM ht
Current performance

Vacation

SwissTM default workload

SwissTM

IBM ht

Speedup

Threads
Current performance

Genome

SwissTM-ME

Speedup

Threads

0 1 2 4 8 16

5 10
Current performance

Genome

Speedup

SwissTM-ME

SwissTM-CE

Threads

1 2 4 8 16

0 5 10
Current performance

Genome

Speedup

SwissTM-ME

SwissTM-CE

SwissTM-MT

Threads

1

2

4

8

16

0

5

10
Current performance

Genome

Speedup

Threads

SwissTM-ME

SwissTM-CE

SwissTM-MT

SwissTM-CT
## Current performance

<table>
<thead>
<tr>
<th></th>
<th>SPARC (64)</th>
<th>x86 (16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SwissTM-ME</td>
<td>17/17</td>
<td>16/17</td>
</tr>
<tr>
<td>SwissTM-CE</td>
<td>-</td>
<td>13/14</td>
</tr>
<tr>
<td>SwissTM-MT</td>
<td>17/17</td>
<td>13/17</td>
</tr>
<tr>
<td>SwissTM-CT</td>
<td>-</td>
<td>11/14</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>87/96 (90%)</strong></td>
<td><strong>87/96 (90%)</strong></td>
</tr>
</tbody>
</table>
Open questions

• What is really going on?
  • significant difference

• What should we do?
  • given application - use TM or not?
Rules of a thumb

• Low contention workloads
  • concurrent threads access disjoint data
  • mostly read accesses
• High ratio of non-tx accesses
• ...

Rules of a thumb

• Low contention workloads
  • concurrent threads access disjoint data
  • mostly read accesses
• High ratio of non-tx accesses
• ...

How to define this precisely?
Current theory

- Worst case performance:

Guerraoui, Kapałka 2007:
Every progressive, single-version TM that uses invisible reads has the time complexity of $\Omega(k)$ where $k = |Obj|$.
Current theory

• Worst case performance:

Guerraoui, Kapałka 2007:
Every progressive, single-version TM that uses invisible reads has the time complexity of $\Omega(k)$ where $k = |Obj|$.

We want common case
Analytical approach

- Model TM as analytical function

\[ \text{speedup} = f(n) \]

- \( f \) captures:
  - workload
  - machine
  - STM
Analytical approach

• Model TM as analytical function

\[
\text{speedup} = f(n)
\]

• \( f \) captures:
  • workload
  • machine
  • STM

Work in progress
Predict performance

- Extrapolate
  - use TM for an application or not?
  - buy more CPUs or not?
- Interpolate
  - schedule optimal number of threads
Function choice

• Ideally
  • based on workload characteristics

• Currently
  • measure speedups
  • approximate
Approximation example

Intruder

Speedup vs. Threads

Speedup measured
Approximation example

Intruder

Threads

Speedup

Measured

57%
Approximation example

![Graph showing speedup vs. threads]

- Intruder
- Speedup
- Threads

Measured speedup for 16 threads is 31% higher than for 1.
Approximation example

![Graph showing speedup vs. threads with an intruder speedup of 24 and a measured speedup of 8% at 24 threads]
Function choice

- Polynomial function
  \[ f(n) = a_3 n^3 + a_2 n^2 + a_1 n + a_0 \]

- Rational function
  \[ f(n) = \frac{a_3 n^3 + a_2 n^2 + a_1 n + a_0}{b_3 n^3 + b_2 n^2 + b_1 n + b_0} \]
Function choice

• **Weierstrass approximation theorem:** Any continuous function on a closed and bounded interval can be uniformly approximated on that interval by polynomials to any degree of accuracy.
Add more cores?

SSCA2

Speedup

Threads

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
Add more cores?

SSCA2

Speedup

Threads
Add more cores?

SSCA2

Speedup

Threads
Add more cores?

SSCA2

Speedup vs. Threads graph

- Speedup values: 0.8, 0.88, 1.2, 1.4, 1.5
- Threads range: 1 to 31

The graph shows the speedup of SSCA2 as the number of threads increases. The speedup increases initially with the addition of more threads, reaching a peak around 1.4, and then plateaus as more threads are added, suggesting diminishing returns or a threshold effect.
Schedule

Intruder

Speedup

Threads
Schedule

Intruder

Speedup vs. Threads

![Graph showing the relationship between Speedup and Threads, with a peak at 22 threads.]
Schedule

Intruder

Threads

Speedup

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63

1, 2, 4, 8, 17

64

22
Intruder Speedup

Schedule

Intruder

Threads

Speedup

1, 2, 4, 8, 17, 64

20

22

16
Schedule

Intruder

Threads

Speedup

1, 2, 4, 8, 17, 20, 23, 64
Thank you
Questions, comments?