EigenBench: A Simple Exploration Tool for Orthogonal TM Characteristics

Pervasive Parallelism Laboratory, Stanford University

Sungpack Hong
Tayo Oguntebi
Jared Casper
Nathan Bronson
Christos Kozyrakis
Kunle Olukotun

IISWC’10, 2010-Dec-02
Outline

- Yet Another Benchmark for TM?
- Orthogonal Characteristics
- EigenBench
- Orthogonal Analysis
- Application Behavior
TM Benchmarks

- Transactional Memory (TM)
  - Significant number of TM proposals
    : Hardware TM, Software TM, Hybrid TM ...
  - How do we evaluate them?

- Conventional TM Benchmarks
  - Application benchmark (STAMP, ...) [Cao Minh et al, IISWC’08]
    - Realistic
  - Synthetic benchmarks (STMBench7, ...) [Guerraoui et al, Eurosys’07]
    - Easy to configure and parametrize.
    - Do they reflect realistic application behavior?
      (e.g.) SwissTM outperformed TL2 2x~5x in synthetic bench, but only 20~90% in STAMPs. [Dragojevic et al, PLDI’09]
Conventional Synthetic Benchmarks

- Synthetic Benchmarks (cont’d)
  - Typically based on shared data-structure access (e.g. red-black tree)
  - Degree of freedom for exploration?

(Example)

L% lookups, U% updates

L2 % lookups, U2% updates

Conflicts? or Number of writes?

Transaction Length? Conflicts?

IISWC’10, 2010-Dec-02
Knobs Wanted

- Want to observe each TM characteristic, separately
- .... But what are the TM characteristics?
  - People mean different things with one term.

ex> “Large Transactions”
  - Many TX reads & writes? (STM barrier overhead)
  - Many different addresses? (HTM overflow)
  - Many (non-tm) instructions inside TX? (rollback overhead)

- We propose eight orthogonal TM characteristics.
TM Characteristics (1/2)

- **Translation Length**
  - Number of *Transactional* read, write

- **Pollution (0.0 ~ 1.0)**
  - \( \frac{WR}{RD + WR} \)

- **Locality (0.0 ~ 1.0)**
  - Prob \{Repeated Address in transaction\}

- **Working-Set Size**
  - Size of memory address region frequently used in application
TM Characteristics (2/2)

- **Contention (0.0 ~ 1.0)**
  - Prob \{Conflict of a transaction\}

- **Concurrency**
  - Number of concurrent threads

- **Predominance (0.0 ~ 1.0)**
  - Fraction of *transactional* access
    - [ ] / ( [ ] + [ ] + [ ] )

- **Density (0.0 ~ 1.0)**
  - Fraction of non-tm instr in TX  
    (complementary)
    - [ ] / ( [ ] + [ ] )
### How do characteristics affect performance?

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>HTM</th>
<th>STM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx Length</td>
<td>Overflow</td>
<td>TX-Barrier overhead</td>
</tr>
<tr>
<td>Pollution</td>
<td>Overflow; conflict detection</td>
<td>Write-buffer manage; conflict detection</td>
</tr>
<tr>
<td>Locality</td>
<td>Overflow</td>
<td>Write-buffer searching</td>
</tr>
<tr>
<td>Working-Set Size</td>
<td>Conflict detection; cache miss latency</td>
<td></td>
</tr>
<tr>
<td>Conflict</td>
<td>Conflict detection</td>
<td></td>
</tr>
<tr>
<td>Concurrency</td>
<td>Scalability</td>
<td></td>
</tr>
<tr>
<td>Density</td>
<td>Cost of re-execution</td>
<td></td>
</tr>
<tr>
<td>Predom.</td>
<td>TM impact on overall performance</td>
<td></td>
</tr>
</tbody>
</table>

(*) Write-set size = (TX Length) * (Pollution) * (1 - Locality)
EigenBench

- How to explore each characteristic one by one?
- EigenBench – a simple exploration tool
EigenBench (Cont’d)

- Implementation is very simple (randomized memory accesses)
- EigenBench can induce each TM characteristic orthogonally.

![Table with equations for various characteristics]

Detailed explanation available in the paper
Orthogonal Analysis: How-to

- **Our approach**
  - Start from a *typical* transaction; explore each characteristic.
  - Non-conflicting transactions ⇒ *overhead*
  - Conflicting transactions ⇒ *detection precision*

- **Example Analysis**
  - TL2 vs. SwissTM
  - Default Transaction;
    - Length: 100, Pollution: 0.1,
    - **Conflict: 0.0**, Working-set: 256kB (per thread),
    - Locality: 0.0, Predom: 1.0,
    - Density: 1.0, Concurrency: 8
Orthogonal Analysis: Results(1)

Unprotected: Performance Upper-bound (No TM protection)

Transaction Length

Speedup (8 thread)

TM overhead

Very short or long transactions

Pollution

Pollution

Write-only TXs

Fast Performance drop (p.s. both performance eventually drops)

IISWC’10, 2010-Dec-02
Orthogonal Analysis: Results (2)

Speedup (8 thread)

Working-Set Size

Cache Effect (Last-Level Cache Overflow)

SwissTM suffers more from cache pressure

Unprotected □ SwissTM ● TL2
Orthogonal Analysis: Results (3)

Conflicts

- Unprotected
- SwissTM
- TL2

Speedup (8 thread)

Cache-Line Migration Effect

- More rollbacks, but better performance

Estimated Conflicts

- High conflicting region (But are we interested?)

$IISWC’10$, 2010-Dec-02

Analysis
Pathology Generation

- TM Pathology [Bobba et al, ISCA 2007]
  - memory access patterns causing low performance
  - Can we generate pathologies from EigenBench? Yes

Friendly Fire (Eager)

Two TXs violating each other

Starving Elder (Lazy)

Short TXs preventing long TX’s progress

TX trace via timestamp
Application Characteristics

- Questions
  - What are TM characteristics of real applications?
  - Can we explain application performance via TM characteristics?

- Example Study: STAMP applications mimicry
  - To demonstrate relationship between characteristics and application performance
  - Instrumentation/Profiling ➔ statistics for TM characteristics ➔ Replay with EigenBench
STAMP Application (1)

Observations
- Different distributions of characteristics
- Single average may not be enough ➔ Mix of discrete characteristics.

Genome

Long-tailed distribution

TX Length

Speedup

% Rollback

IISWC’10, 2010-Dec-02
STAMP Application (2)

Vacation (Low)

- Normal distribution
- Wide memory access

Graphs showing frequency and speedup with various memory access patterns.
STAMP Application (3)

- Long TX Low Density
- Short TX Large Working-Set
- Short TX High Conflicts

IISWC’10, 2010-Dec-02
EigenBench Use-cases

How to use EigenBench?

1. Orthogonal Analysis
   : Length, Working-Set, Pollution, Conflicts, Concurrency, (locality, density, predom)
   - Non-conflicting
   - Conflicting
2. Explain application behavior
3. (Optional) Check if it can survive pathologies.
Summary

- Orthogonal TM Characteristics
- EigenBench
- Orthogonal Analysis
- Application Performance Explanation

- Subsidiary Lessons for STM designers
  - Cache effect should be considered
  - Trade-off barrier overhead vs. conflict resolution
    - Restart penalty can be small

- Download: http://ppl.stanford.edu/eigenbench
- E-mail: eigenbench_manager@lists.stanford.edu