LASER: Light, Accurate Sharing dEtection and Repair

Liang Luo, Akshitha Sriraman, Brooke Fugate, Shiliang Hu, Chris J Newburn, Gilles Pokam, Joseph Devietti
Multicore is Eating the World

+ Performance
+ Energy efficiency
- Performance bugs
Cache Contention Bugs

Contention for a single cash line

- Caused significant performance loss (Linux, MySQL, Boost)
- Architecture-specific
- Hard to find and debug
• Cache coherence keeps private caches in sync
• All protocols share 3 key states: Modified, Shared, Invalid

![Diagram showing cache coherence and protocol states]

- Core 0: Write X
- Core 1: Read X
Two Types of Contention

Same Bytes == True Sharing

Core 0
Write X

Core 1
Read X

Cache Line
Two Types of Contention

Different Bytes == False Sharing

Cache Line

Core 0  Write X

Core 1  Read Y
Related Work

- **Sheriff**
  [Liu and Berger, OOPSLA 2011]
- **Plastic**
  [Nanavati et al., EuroSys 2013]
- **Cheetah**
  [Liu and Liu, CGO 2016]
- **vTune Amplifier XE**
  [Intel]
HitM Events

• A fundamental part of both types of contention
  • A cache hit in a remote core’s cache in M state.
HITM Record Accuracy

• 160 simple programs with read/write and write/write true/false/no sharing
• Intel Core i7-4770K 3.4GHz Haswell 4-core processor
HITM Record Accuracy (1/2)

160 simple benchmarks

Counting only exact addresses as being correct
HITM Record Accuracy (2/2)

160 simple benchmarks

- Counting only exact PC as being correct
- Counting adjacent PCs as being correct
LASER

- **Light:** leverage Haswell h/w, no s/w or OS changes
- **Accurate:** low false positives and false negatives
- **Sharing:** detects both true and false sharing
- **dETection:** works as a profiling tool
- **Repair:** automatically repairs false sharing at runtime
LASER System Overview

User Level App

Detector Process

LASER Repair

App Process

Operating System

Linux Kernel

Driver

Hardware

Haswell or later Processors
Detection Algorithm

Cache Line Model

Disjoint

HITM

Store X+8, 4B

HITM

Load X, 2B

FALSE SHARING
Detection Algorithm

Aggregate
By Source
Code Line

Foo.c:23
Evaluating LASER Detection

- Intel Core i7-4770K 3.4GHz Haswell 4-core processor
- 33 workloads from Phoenix 1.0, Parsec 3.0 and Splash2X
LASER Detection Accuracy

Created a database of manually-validated cache contention bugs
  • 9 bugs total across 33 workloads
  • 4 new bugs discovered by LASER

<table>
<thead>
<tr>
<th></th>
<th>False Negative</th>
<th>False Positive</th>
<th>Runnable Benchmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASER</td>
<td>0</td>
<td>24</td>
<td>33/33</td>
</tr>
<tr>
<td>VTUNE</td>
<td>1</td>
<td>64</td>
<td>33/33</td>
</tr>
<tr>
<td>SHERIFF</td>
<td>3</td>
<td>4</td>
<td>12/33</td>
</tr>
</tbody>
</table>
Profiling Performance Comparison

**LASER** (1.01x) vs **VTune Amplifier XE 2015** (1.8x)
Repairing FS with SSB

- Needs Online False Sharing elimination!
  - Legacy programs – no access to source code.
  - Programs that need to be always running.
- Challenge is needing to rewrite the program without breaking it as it is running.
- Solve the problem of online FS repair with a Software Store Buffer (SSB). LASER repair tool is launched to attempt fix of FS.
  - Implemented with Intel Pin
Flush at synchronization points for TSO compliance and basic block end for performance.
The conventional hardware store buffer is not good enough for speedup.
Needs optimizations for better performance
  • May cause subtle memory consistency issues.
  • For good performance, requires coalescing store buffer. But coalescing violates TSO. E.g. Sheriff does not provide TSO compliance.
LASER’s TSO-Compliant, Coalescing SSB

- Instrument regions from Laser’s input by walking through the CFG.
- Coalescing, TSO compliant SSB made possible with Intel TSX.

Read X:
Result = SSB[X]
If (Result == null)
  Result = *X;
Return Result;

Write X, Val:
If (SSB[X] == null && SSB.full())
  Flush();
SSB[X] = Val;

Flush:
TSX_Begin_Transaction
Foreach pair in SSB
  *pair.memory = pair.value;
TSX_End_Transaction
Redo_TSX_Transaction_If_Fails
LASER FS Repair Performance

Automatic speedups of up to 19%
LASER profiling informs manual fixes of up to 17x
Conclusions

• Cache contention bugs undermine the promise of multicore
• LASER uses Intel’s Haswell platform for fast, precise contention detection and automatic false sharing repair
• Many opportunities to leverage Haswell’s sharing detection capabilities

Researchers Who Read This Paper Also Read

✓ REMIX: Online Detection and Repair of Cache Contention for the JVM
[PLDI 2016]