REMIX
Online Detection and Repair of Cache Contention for the JVM

Ariel Eizenberg, Joseph Devietti
Shiliang Hu, Gilles Pokam

Penn Engineering
intel
What’s wrong with this code?

```java
class Test extends AtomicLong implements Runnable {
    static void main(final String[] args) {
        Test test1 = new Test();
        Test test2 = new Test();
        Thread t1 = new Thread(test1);
        Thread t2 = new Thread(test2);
        t1.start();t2.start();
        t1.join();t2.join();
    }
    public void run() {
        for(int i = 0; i < 100000000; ++i) set(i);
    }
}
```
What’s wrong with this code?

```java
public class Test extends AtomicLong implements Runnable {
    static void main(final String[] args) {
        Test test1 = new Test();
        Thread t1 = new Thread(test1);
        Test test2 = new Test();
        Thread t2 = new Thread(test2);
        t1.start(); t2.start();
        t1.join(); t2.join();
    }
    public void run() {
        for(int i = 0; i < 100000000; ++i) set(i);
    }
}
```
What happened?
What happened?

• Version A: 313ms
What happened?

• Version A: 313ms

• Version B: 151ms
What happened?

- Version A: 313ms
  ![Diagram of Version A]

- Version B: 151ms
  ![Diagram of Version B]

- More surprises!
  ![Diagram showing additional issues]

GC, Class Loader, ...
Managed Languages

Runtime has increased control over execution ⇒ opportunities for dynamic optimization

Less programmer control over memory ⇒ more vulnerable to performance bugs
REMIX

• First system to automatically detect and repair cache contention in managed languages
• Uses hardware performance counters to detect cache contention
• Automatically repairs cache contention bugs, significantly simplifying programmers job
• Implemented as a GC-like pass within the HotSpot JVM
Outline

Cache Coherency

The Remix system

Evaluation
Cache Coherency

• Multicore processors implement a cache coherence protocol to keep private caches in sync.
• Operates on whole cache lines (usually 64 bytes).
• Cache lines have three key states:
  • Read Shared
  • Write Exclusive
  • Invalid
True vs False sharing

Same Bytes
True Sharing

Different Bytes
False Sharing

Cache Line

Core 0

Core 1

Core 0

Core 1
Intel PEBS Events

• PEBS – Precise Event-Based Sampling
• Available in recent Intel multiprocessors
• Log detailed information about architectural events
PEBS HitM Events

• “Hit-Modified” - A cache miss due to a cache line in Modified state on a different core
## Related Work

- **Sheriff** [Liu and Berger, OOPSLA 2011] - detect & repair unmanaged languages
- **Plastic** [Nanavati et al., EuroSys 2013]
- **Laser** [Luo et al., HPCA 2016]
- **Cheetah** [Liu and Liu, CGO 2016]
- **Predator** [Liu et. al., PPoPP 2014] - detection only
- **Intel vTune Amplifier XE**
- **Oracle Java8 @Contended** - manual repair
REMIX

• A modified version of the Oracle HotSpot JVM
• Detects & repairs cache contention bugs at runtime
• Works with all JVM languages, no source code modification required
• Provides performance matching hand-optimization
REMIX System Overview

Application: Java, Scala, etc'
Runtime system: JVM
OS: Linux Kernel
Hardware: CPU w/HITM PEBS

REMIX

Detection
Repair
Performance
Detection
Detection
Detection

- HITM Events
  - classify
  - native heap & stack
  - track
  - complete
Detection

HITM Events

classify

native heap & stack

managed heap

track

complete

>threshold

no
Detection

HITM Events → classify

native heap & stack

managed heap

track

> threshold

complete

no

yes

model $ lines
Detection

HITM Events -> classify

native heap & stack -> track

managed heap -> >threshold

classify -> map to objects

model $ lines

complete

no

yes
Detection

HITM Events → classify

native heap & stack

managed heap

track → complete

> threshold

no

yes

classify

map to objects

model $ lines

true sharing

report
Detection

- HITM Events
  - classify
    - native heap & stack
    - managed heap
  - track
    - complete
      - no
      - yes
  - model
    - map to objects
    - model \$ lines
  - false sharing
  - true sharing
  - repair
  - report

Summary | Background | REMIX | Detection | Repair | Performance
Cache Line Modelling

- Cache lines are modelled with 64-bit bitmaps
- HITM event $\Rightarrow$ set the \textbf{address} bit, count hit
- Multiple bits set $\Rightarrow$ potential false sharing
- Repair is cheaper than more complete modelling!
- Repair when counter exceeds threshold
Top Level Flow
Top Level Flow

Original Heap

...
Top Level Flow

Original Heap

Original Classes

Foo

long foo0;
long foo1;

Bar

long bar0;
long bar1;
Top Level Flow

Original Heap

Original Classes

Modified Classes

Foo

long foo0;
long foo1;

Bar

long bar0;
long bar1;

Foo'

long foo0;
<pad 56B>
long foo1;
<pad 56B>

Bar'

long bar0;
long bar1;
<pad 48B>
Padding

- Instance data
- Instance size
- Field list
- OOPMap (reference block map)
- Constant-pool cache
Padding - Inheritance

class A
header (16b)
byte a1
byte a2

class B extends A
header (16b)
byte a1
byte a2
byte b

class C extends A
header (16b)
byte a1
byte a2
byte c

Summary Background REMIX Detection Repair Performance
Padding - Inheritance

class A
- header (16b)
- byte a1
- byte a2

class B extends A
- header (16b)
- byte a1
- byte a2
- byte b

class C extends A
- header (16b)
- byte a1
- byte a2
- byte c
Padding - Inheritance

- Class A:
  - Header (16b)
  - Byte a1
  - Byte a2

- Class B (extends A):
  - Header (16b)
  - Byte a1
  - Byte a2
  - Byte b

- Class C (extends A):
  - Header (16b)
  - Byte a1
  - Byte a2
  - Byte c

Summary
Background
REMIX
Detection
Repair
Performance
Padding - Inheritance

class A
header (16b)
byte a1
byte a2

class B extends A
header (16b)
byte a1
byte a2
byte b

class C extends A
header (16b)
byte a1
byte a2
byte c
Padding - Inheritance

- class A
  - header (16b)
  - byte a1
  - byte a2

- class B
  - header (16b)
  - byte a1
  - byte a2
  - byte b

- class C
  - header (16b)
  - byte a1
  - byte a2
  - byte c

Image highlights:
- Padding: 48b and 62b
- Class B and C extending class A
- Summary, Background, REMIX, Detection, Repair, Performance
Repair

• Trace all strong+weak roots in the system
• Traverse heap and find targeted instances
  • Live ⇒ Relocate & pad, store forwarding pointer
  • Dead ⇒ Fix size mismatch
• Adjust all pointers to forwarded objects
• Deoptimize all relevant stack frames
Disruptor & Spring Reactor

• Both are libraries for high-speed inter-thread message passing
Disruptor & Spring Reactor

• Both are libraries for high-speed inter-thread message passing

```java
class Sequence {
    protected volatile long value;
}
```
Disruptor & Spring Reactor

• Both are libraries for high-speed inter-thread message passing

```java
class Sequence {
    protected long a, b, c, d, e, f, g; // pad before
    protected volatile long value;
    protected long i, j, k, l, m, n, o; // pad after
}
```

• Is this enough?

56 bytes
Disruptor & Spring Reactor

• Both are libraries for high-speed inter-thread message passing

```java
class Sequence {
    protected long a, b, c, d, e, f, g; // pad before
    protected volatile long value;
    protected long i, j, k, l, m, n, o; // pad after
}
```

• Is this enough?

• No! – Class loader optimizes aggressively, lays out `value` to before `a`. 

56 bytes
Disruptor & Spring Reactor

• A complex hierarchy forces field order

```java
class LhsPadding {
    protected long a, b, c, d, e, f, g;
}
class Value extends LhsPadding {
    protected volatile long value;
}
class RhsPadding extends Value {
    protected long i, j, k, l, m, n, o;
}
class Sequence extends RhsPadding {
    // actual work
}
```
Disruptor + Spring Reactor

SPEEDUP (HIGHER IS BETTER)

REMIX Manual Pad

1-1-RawBatchThpt 1-1-RawThpt 1-1-SeqPollerThpt 1-1-TranslatorThpt 1-3-DiamondSeqThpt 1-3-PipelineSeqThpt 1-3-RelWorkerPoolThpt 1-3-WorkerPoolThpt 3-1-SeqBatchThpt 3-1-SeqThpt Reactor-WorkQueue
Disruptor + Spring Reactor
Disruptor + Spring Reactor

![Graph showing SPEEDUP (HIGHER IS BETTER) for various scenarios.](image)

Legend:
- REMIX
- Manual Pad

21 Summary Background REMIX Detection Repair Performance
Disruptor + Spring Reactor

REMIX performs as well as manual fixes in most cases
Disruptor + Spring Reactor

REMIX repair threshold is adjustable
Disruptor + Spring Reactor

**REMIX** outperforms when **manual fix** adds unnecessary padding.
NAS Parallel Benchmarks

- **REMIX** reveals true-sharing running NAS suite
  - HITM from true-sharing on JIT counters
  - JIT-ing fixes contention
  - NAS workloads exceed JIT size limit, not JIT-ed
- **REMIX** detects this and forces compilation
NAS Parallel Benchmarks

- **REMIX** reveals true-sharing running NAS suite
  - HITM from true-sharing on JIT counters
  - JIT-ing fixes contention
  - NAS workloads exceed JIT size limit, not JIT-ed
- **REMIX** detects this and forces compilation
NAS Parallel Benchmarks

- REMIX reveals true-sharing running NAS suite
  - HITM from true-sharing on JIT counters
  - JIT-ing fixes contention
  - NAS workloads exceed JIT size limit, not JIT-ed
- REMIX detects this and forces compilation

Summary

<table>
<thead>
<tr>
<th>Speedup (higher is better)</th>
<th>BT</th>
<th>CG</th>
<th>FT</th>
<th>IS</th>
<th>LU</th>
<th>MG</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7.7</td>
<td></td>
<td></td>
<td></td>
<td>9.0</td>
<td></td>
<td>25.3</td>
</tr>
</tbody>
</table>
No Contention == No Impact
No Contention == No Impact

(DaCapo & ScalaBench)

(SpecJVM)
No Contention == No Impact

(DaCapo & ScalaBench)

(SpecJVM)
No Contention == No Impact

(DaCapo & ScalaBench)

(SpecJVM)
No Contention == No Impact

![Speedup Chart](chart.png)

(DaCapo & ScalaBench)

(SpecJVM)
Padding

- Original Size
- Padding

KB

DaCapo 9.12 sunflow
SPECjvm2008 serial
1-1-RawThpt
1-1-RawBatchThpt
1-1-SeqPollerThpt
1-3-DiamondSeqThpt
1-3-PipelineSeqThpt
1-3-SeqThpt
3-1-SeqBatchThpt
3-1-SeqThpt
3-1-TranslatorThpt
1-3-RelWorkerPoolThpt
1-3-WorkerPoolThpt
2-2-WorkProcessorThpt
Reactor-WorkQueue

Summary → Background → REMIX → Detection → Repair → Performance
Padding

- DaCapo 9.12: 5 KB
- SPECjvm2008: 15 KB
- 1-1-Raw: 2 KB
- 1-1-Batch: 1 KB
- 1-1-Seq: 1 KB
- 1-3-Diamond: 2 KB
- 1-3-Pipeline: 1 KB
- 3-1-Seq: 1 KB
- 3-1-Batch: 1 KB
- 1-1-Translator: 1 KB
- 1-3-RelWorker: 1 KB
- 1-3-Worker: 1 KB
- 2-2-WorkProcessor: 1 KB
- Reactor-WorkQueue: 15 KB

- Original Size
- Padding
Padding

- **Original Size**
- **Padding**

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Original Size</th>
<th>Padding</th>
</tr>
</thead>
<tbody>
<tr>
<td>DaCapo 9.12 sunflow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPECjvm2008 serial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-1-RawThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-1-RawBatchThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-1-SeqPollerThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-3-DiamondSeqThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-3-PipelineSeqThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-3-SeqThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-1-SeqBatchThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-1-SeqThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-1-TranslatorThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-3-RelWorkerPoolThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-3-WorkerPoolThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-2-WorkProcessorThpt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reactor-WorkQueue</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**KB**

- 0
- 5
- 10
- 15

**REMIX**

- Detection
- Repair
- Performance
Performance
Performance

The graph illustrates the performance of different benchmarks with respect to time (ms). The benchmarks include DaCapo 9.12 sunflow, SPECjvm2008 serial, and various other implementations. The graph shows the breakdown of time spent on Collector, Detector, and Repair tasks. For example, DaCapo 9.12 sunflow serial shows a significant time spent on Collector and Detector tasks, with a smaller portion on Repair.
Conclusions

• Cache contention afflicts managed languages
• Dynamic information opens up new avenues for optimization
• Performance counters are a gold mine
• REMIX can simplify programmers’ lives by automatically fixing false sharing bugs
• [https://github.com/upenn-acg/REMIX](https://github.com/upenn-acg/REMIX) (GPLv2)
Q&A
PEBS HITM Event Accuracy

R-W Contention: 98.60%

W-W Contention: 7.40%

data address accuracy
Objects Moved
sun.misc.Unsafe

- Tracks unsafe access, prevents padding
  
  ```java
  FLD_OFFS = U.objectFieldOffset(k.getDeclaredField("fld"));
  Unsafe.putLong(o, FLD_OFFS, value);
  ```

- **REMIX Extended Unsafe**:
  
  ```java
  U.registerFieldOffset(k.getDeclaredField("fld"),
                     k.getDeclaredField("FLD_OFFS"));
  Unsafe.putLong(o, FLD_OFFS, value);
  ```

- Just as fast, enables padding
Time-to-fix
## Performance

<table>
<thead>
<tr>
<th>Benchmark(s)</th>
<th>Classes Loaded</th>
<th>Processing (ms)</th>
<th>Heap Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DaCapo Sunflow</td>
<td>1879</td>
<td>1.47</td>
<td>854</td>
</tr>
<tr>
<td>Disruptor</td>
<td>700-900</td>
<td>0.5-1.3</td>
<td>40-85</td>
</tr>
<tr>
<td>SpringReactor WorkQueue</td>
<td>1617</td>
<td>0.748</td>
<td>41</td>
</tr>
<tr>
<td>SpecJVM serial</td>
<td>1498</td>
<td>1.21</td>
<td>1603</td>
</tr>
</tbody>
</table>
REMIX Speedup vs HITM rate
Size-Preserving-Klass

- Instances are stored contiguously in memory
- Object sizes not stored directly in the heap
  - Calculated through the \textit{klass} pointer in the object header
- Padding breaks this assumption
  - Object left behind at old location has wrong size!
- Solution – create a ghost \textit{klass} with the original size
  - Modify dead instances to point to this ghost
- Take care not to traverse padded objects in heap until \textit{klass} size is updated
Padding - Inheritance

- **Class A**: header (16b), byte a
- **Class B** extends A: header (16b), byte a, byte b
- **Class C** extends A: header (16b), byte a, byte c
- **Class E** extends C: header (16b), byte a, byte c, byte d
- **Class E** extends C: header (16b), byte a, byte c, byte e
Padding - Inheritance

class A
- header (16b)
- byte a

class B extends A
- header (16b)
- byte a
- byte b

class C extends A
- header (16b)
- byte a
- byte c

class E extends C
- header (16b)
- byte a
- byte c
- byte d

class E extends C
- header (16b)
- byte a
- byte c
- byte e
Padding - Inheritance

class A
header (16b)
byte a

class B extends A
header (16b)
byte a
byte b

class C extends A
header (16b)
byte a
byte c

class E extends C
header (16b)
byte a
byte c
byte d

class E extends C
header (16b)
byte a
byte c
byte e
Padding - Inheritance

class A
header (16b)
byte a

class B extends A
header (16b)
byte a
byte b

class C extends A
header (16b)
byte a
byte c

class E extends C
header (16b)
byte a
byte c
byte d

class E extends C
header (16b)
byte a
byte c
byte e
Padding - Inheritance

```
class A
header (16b)
byte a

class B extends A
header (16b)
byte a
byte b

class C extends A
header (16b)
byte a
byte c

class E extends C
header (16b)
byte a
byte c
byte d

class E extends C
header (16b)
byte a
byte c
byte e
```
Padding - Inheritance

class A
   header (16b)
   byte a

class B extends A
   header (16b)
   byte a
   byte b

class C extends A
   header (16b)
   byte a
   byte c

class E extends C
   header (16b)
   byte a
   byte c
   byte d

class E extends C
   header (16b)
   byte a
   byte c
   byte e
Padding - Inheritance

class A
header (16b)
byte a

class B extends A
header (16b)
byte a
byte b

class C extends A
header (16b)
byte a
byte c

class E extends C
header (16b)
byte a
byte c
byte d

class E extends C
header (16b)
byte a
byte c
byte e

Summary
Background
REMIX
Detection
Repair
Performance
Savina Actors Benchmark

- Evaluates actor libraries across 30 benchmarks
- REMIX detects false and true sharing bugs in libraries & benchmarks:
Virtualization

• Although Xen supports `perf`, it does not yet support virtualization of HW PEBS events.