REMIX

Online Detection and Repair of Cache Contention for the JVM

<u>Ariel Eizenberg,</u>

Joseph Devietti

Penn Engineering

Shiliang Hu,

Gilles Pokam

intel

What's wrong with this code?

```
public 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);
}}
```

313ms

What’s wrong with this code?

```
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);
}}
```

151ms

What happened?

What happened?

- Version A: 313ms

What happened?

- Version A: 313ms

- Version B: 151ms

What happened?

- Version A: 313ms

- Version B: 151ms

- More surprises!

GC, Class Loader, …

Managed Languages

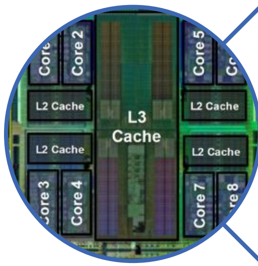
Runtime has increased control over execution
⇒ opportunities for dynamic optimization

Less programmer control over memory
⇒ more vulnerable to performance bugs

REMIX

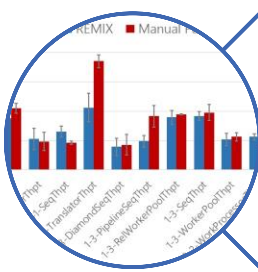
- First system to automatically **detect** and **repair** cache contention in **managed languages**
- Uses **hardware performance** counters to detect cache contention
- **Automatically** repairs cache contention bugs, **significantly simplifying** programmers job
- Implemented as a **GC-like pass**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

Outline



Cache Coherency

The Remix system

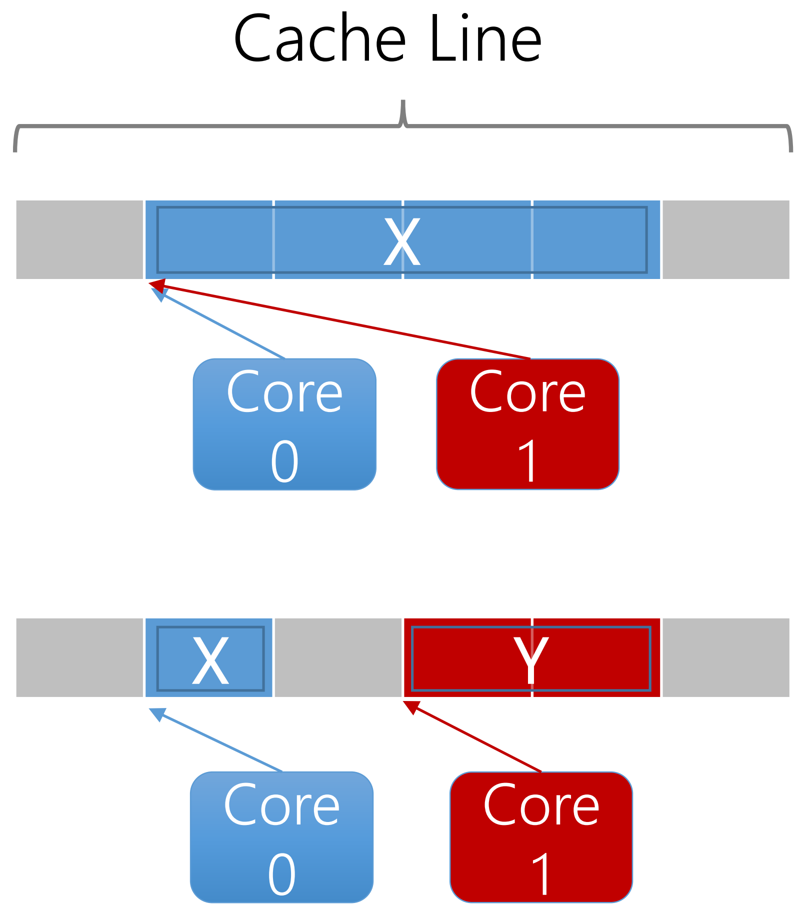


Evaluation

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

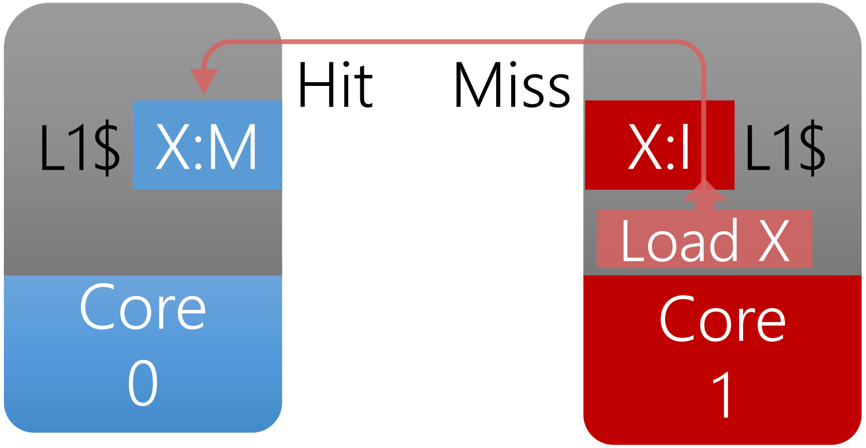
Different Bytes
False Sharing

Intel PEBS Events

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

PEBS HitM Events

- "**Hit-M**odified" - A cache miss due to a cache line in *Modified* state on a different core



Related Work

- Sheriff [Liu and Berger, OOPSLA 2011]
- Plastic [Nanavati et al., EuroSys 2013]
- Laser [Luo et al., HPCA 2016]

detect & repair unmanaged languages

- Cheetah [Liu and Liu, CGO 2016]
- Predator [Liu et. al., PPoPP 2014]
- Intel vTune Amplifier XE

detection only

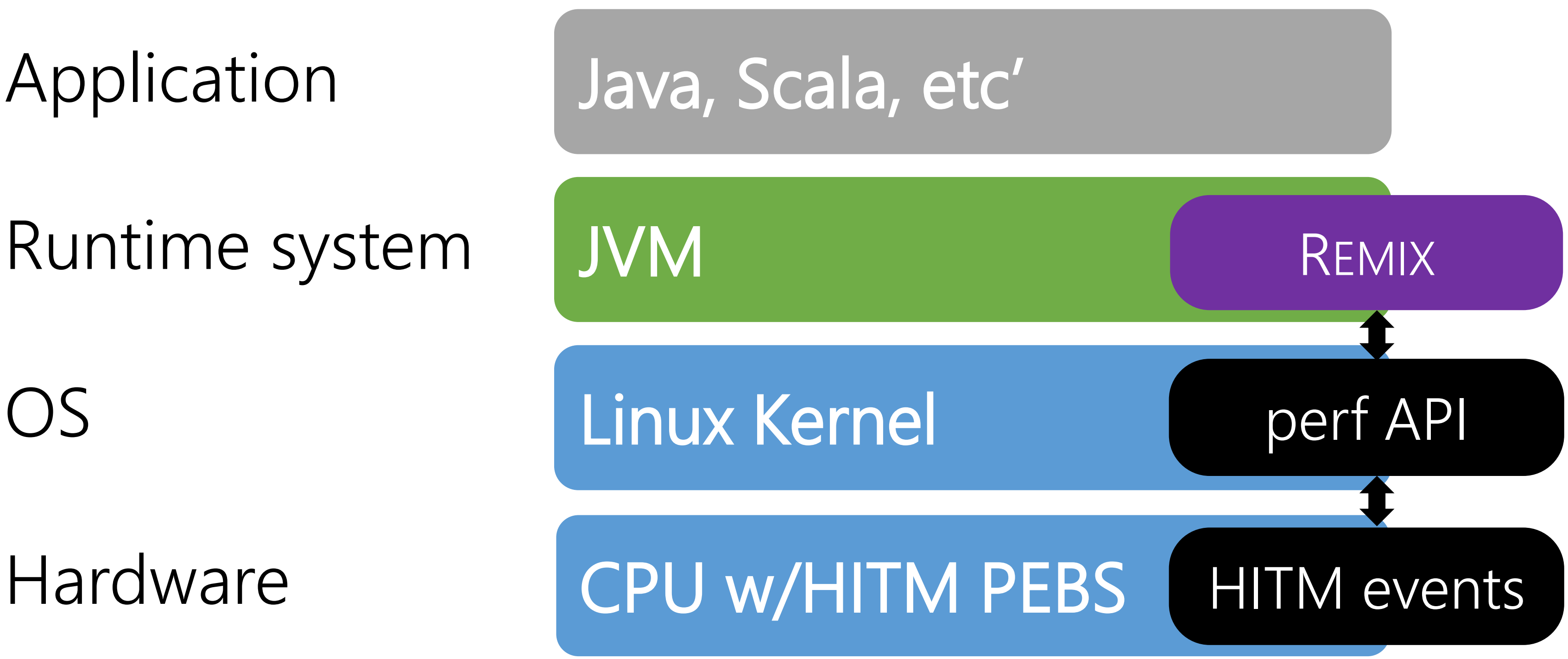
- Oracle Java8 @Contended

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

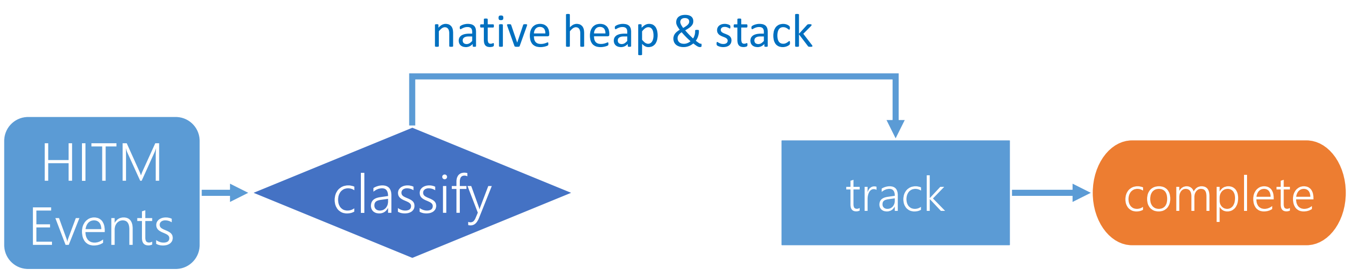


Detection

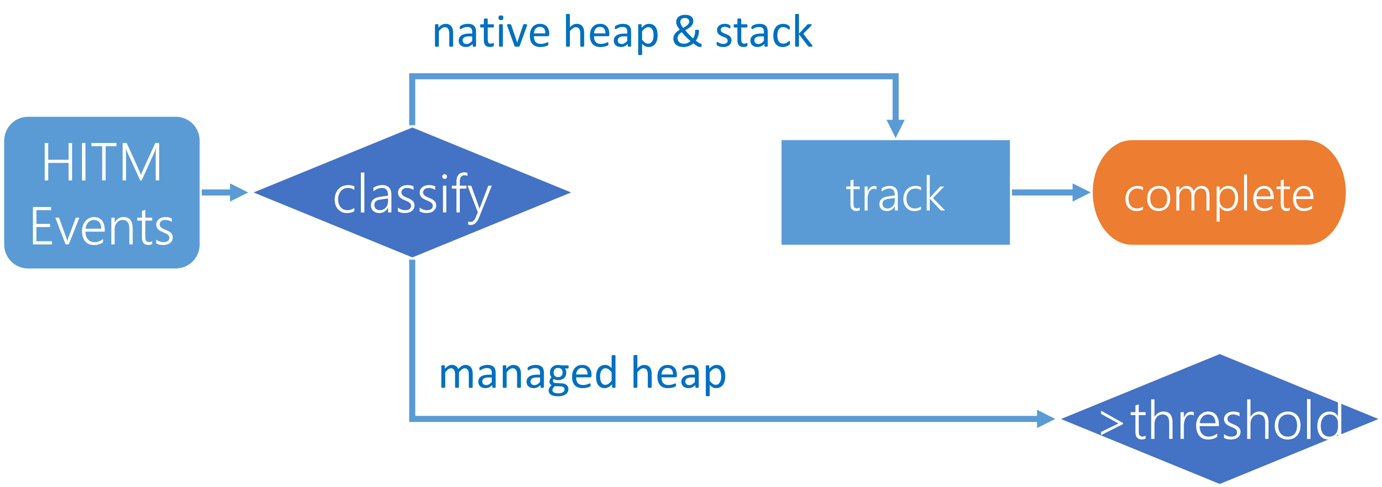
Detection

HITM Events

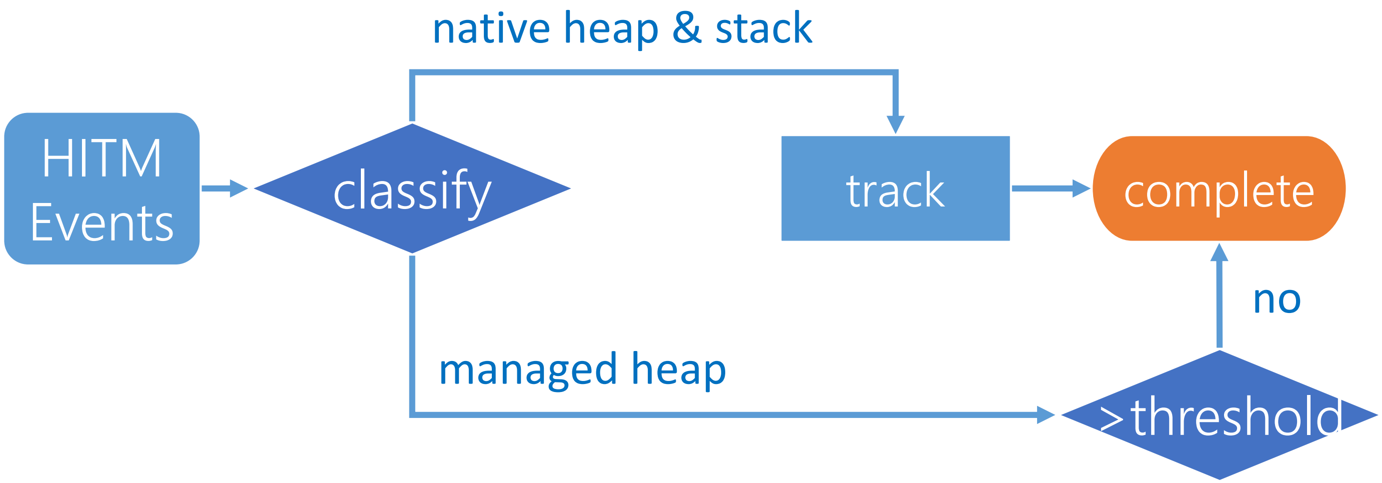
Detection



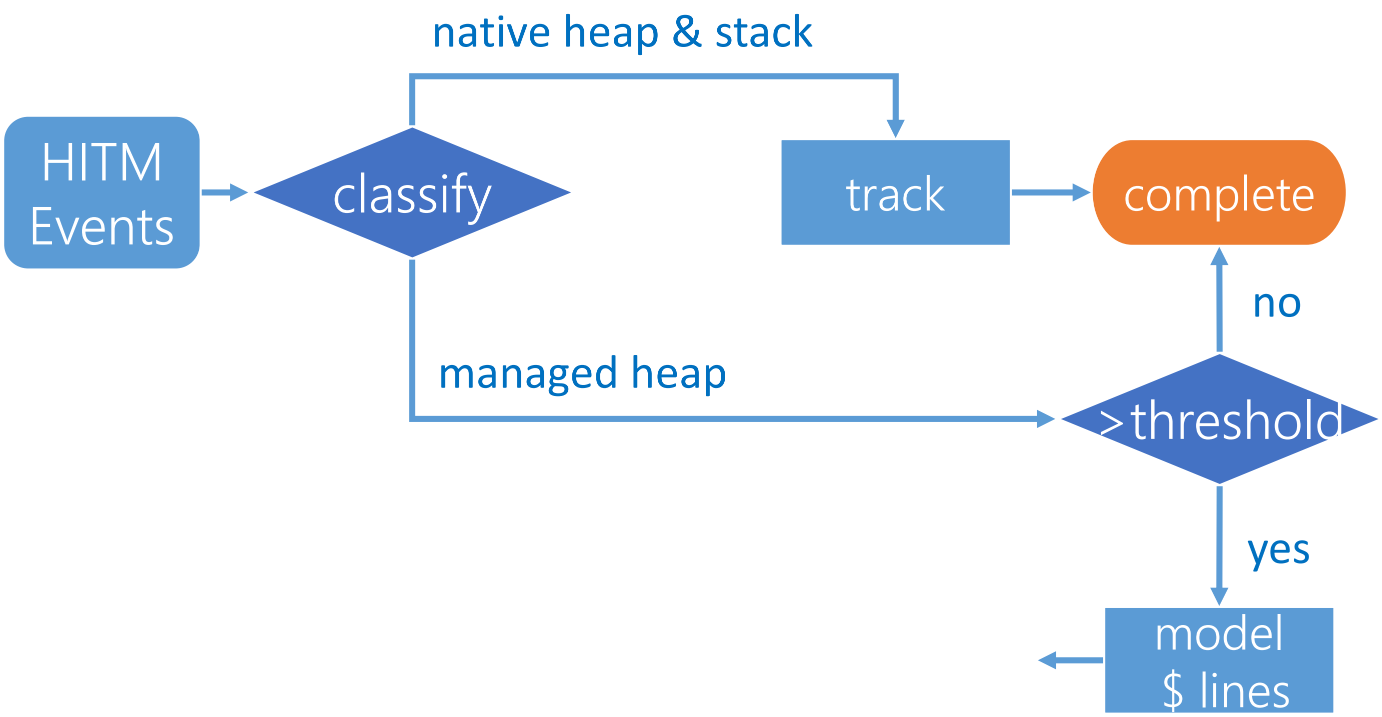
Detection


native heap & stack
HITM
Events
classify
track
complete
managed heap
>threshold

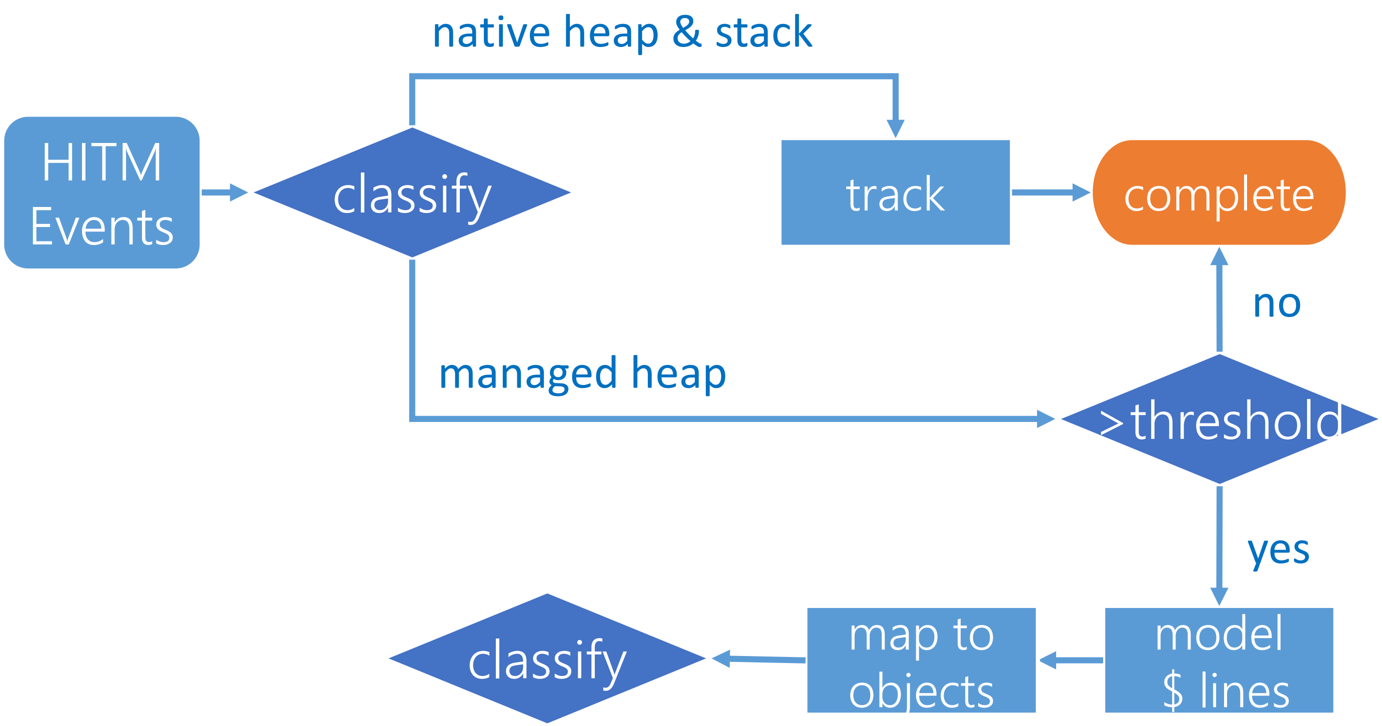
Detection


native heap & stack
HITM Events
classify
track
complete
no
managed heap
>threshold

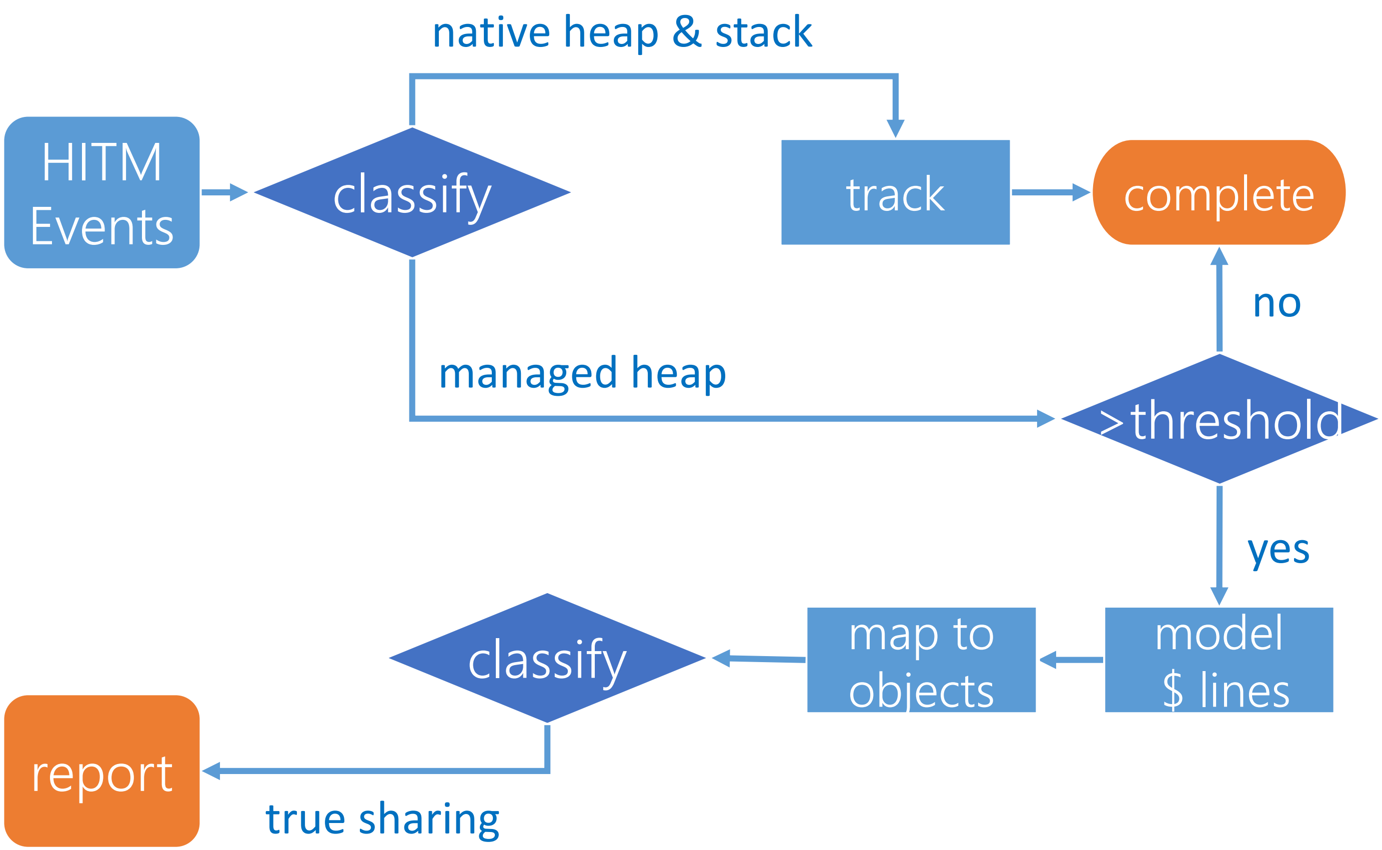
Detection


native heap & stack
HITM
Events
classify
track
complete
no
managed heap
>threshold
yes
model
$ lines

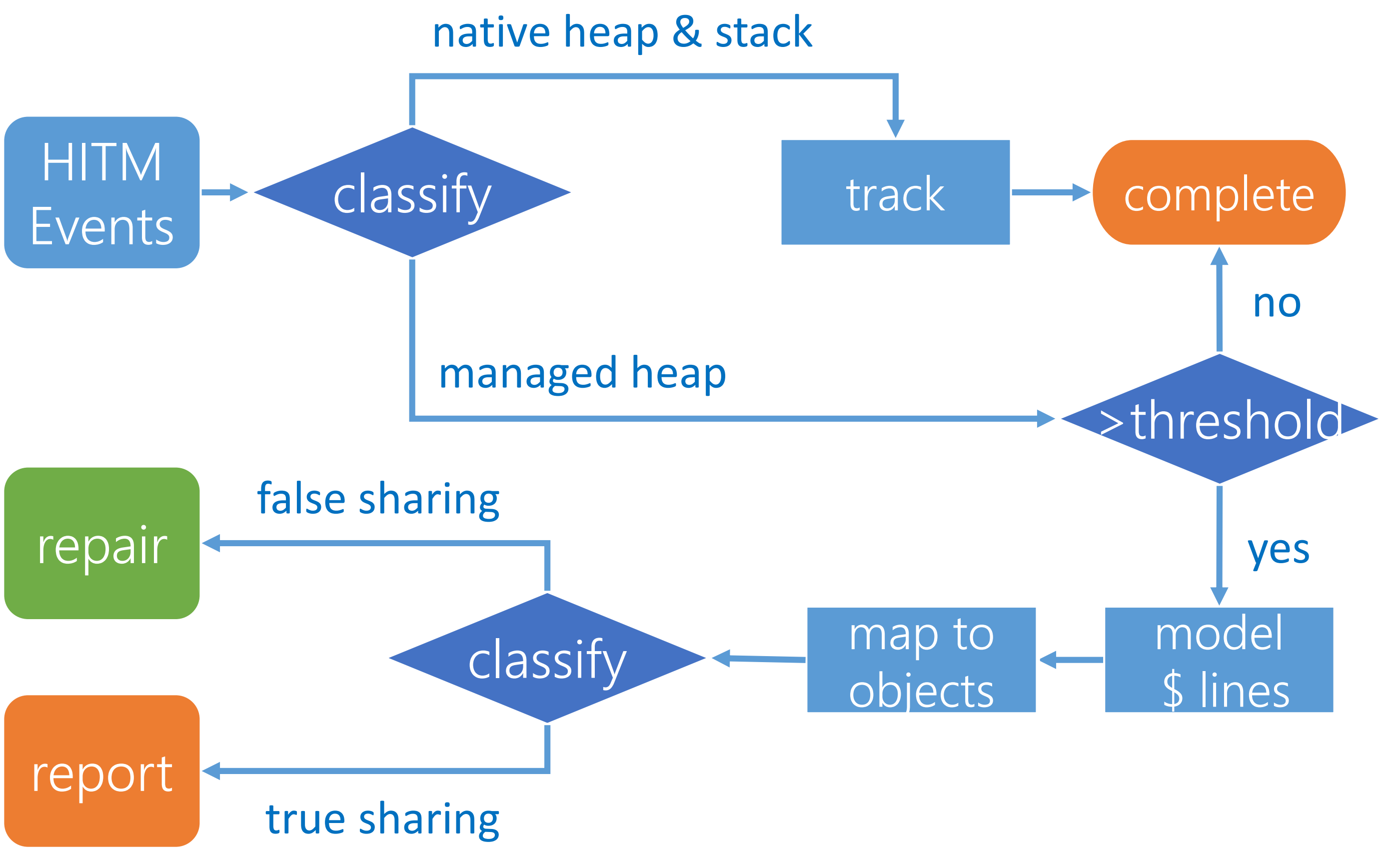
Detection


native heap & stack
HITM
Events
classify
track
complete
no
managed heap
>threshold
yes
classify
map to
objects
model
$ lines

Detection


native heap & stack
HITM
Events
classify
track
complete
no
managed heap
>threshold
yes
classify
map to
objects
model
$ lines
report
true sharing

Detection


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

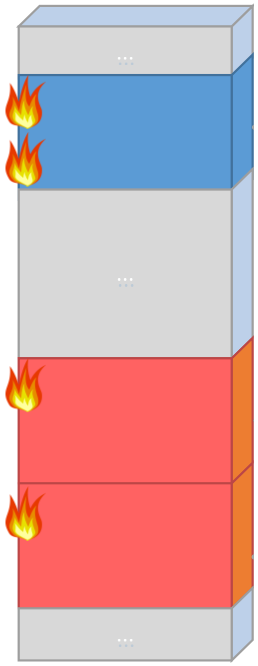
Cache Line Modelling

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

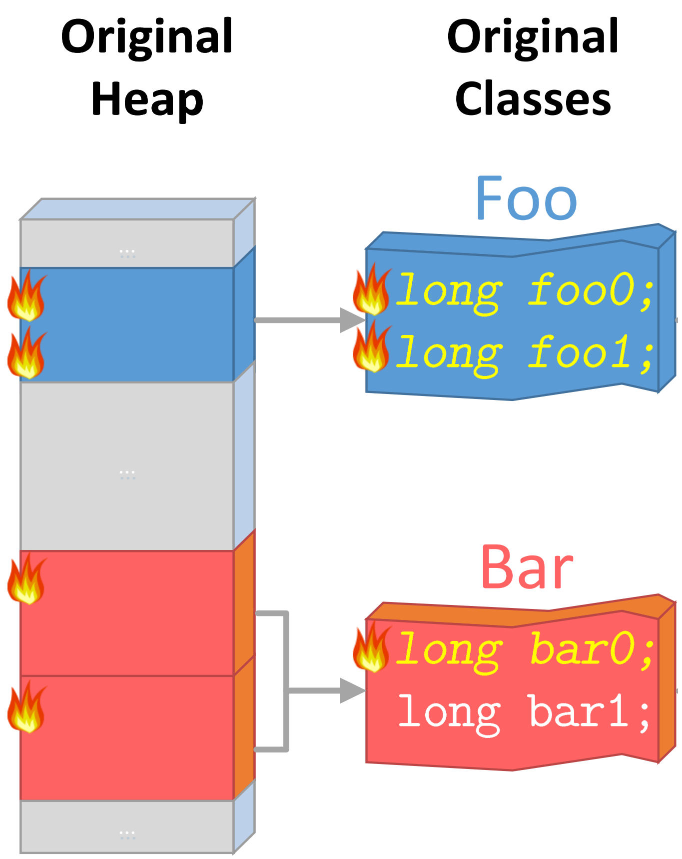
Top Level Flow

Top Level Flow

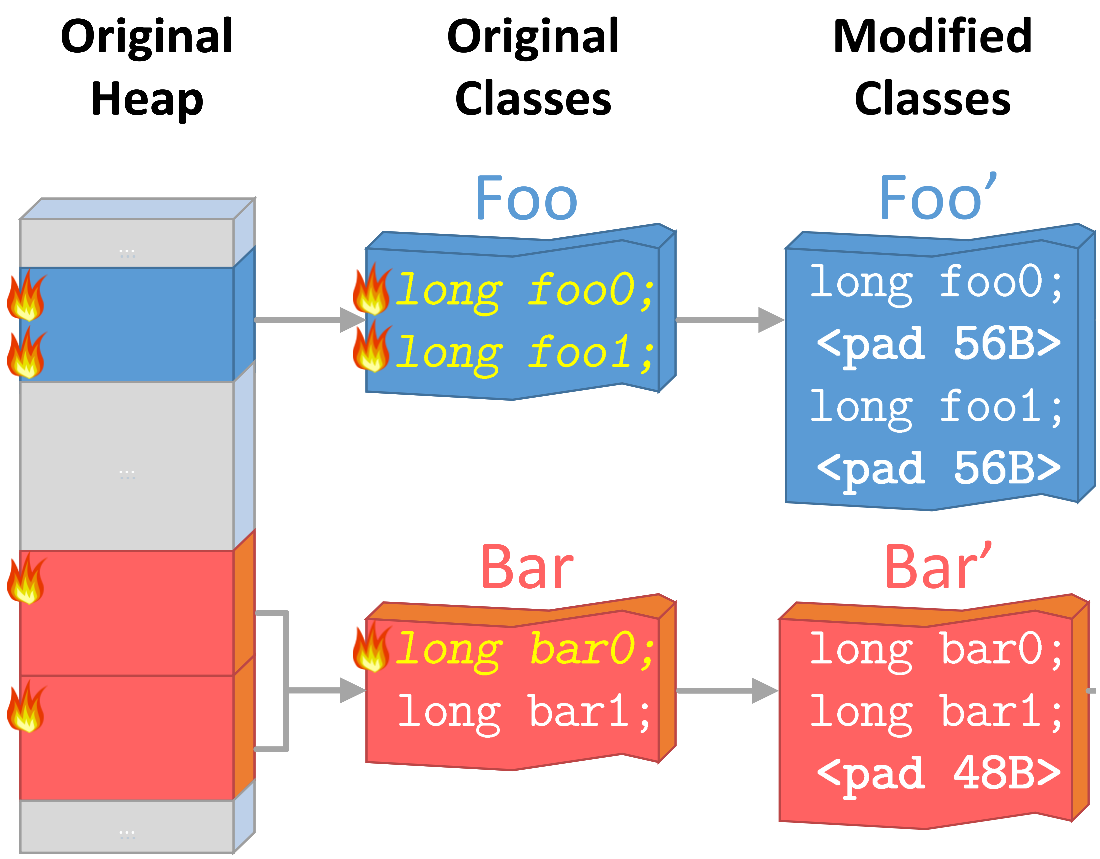
Original Heap



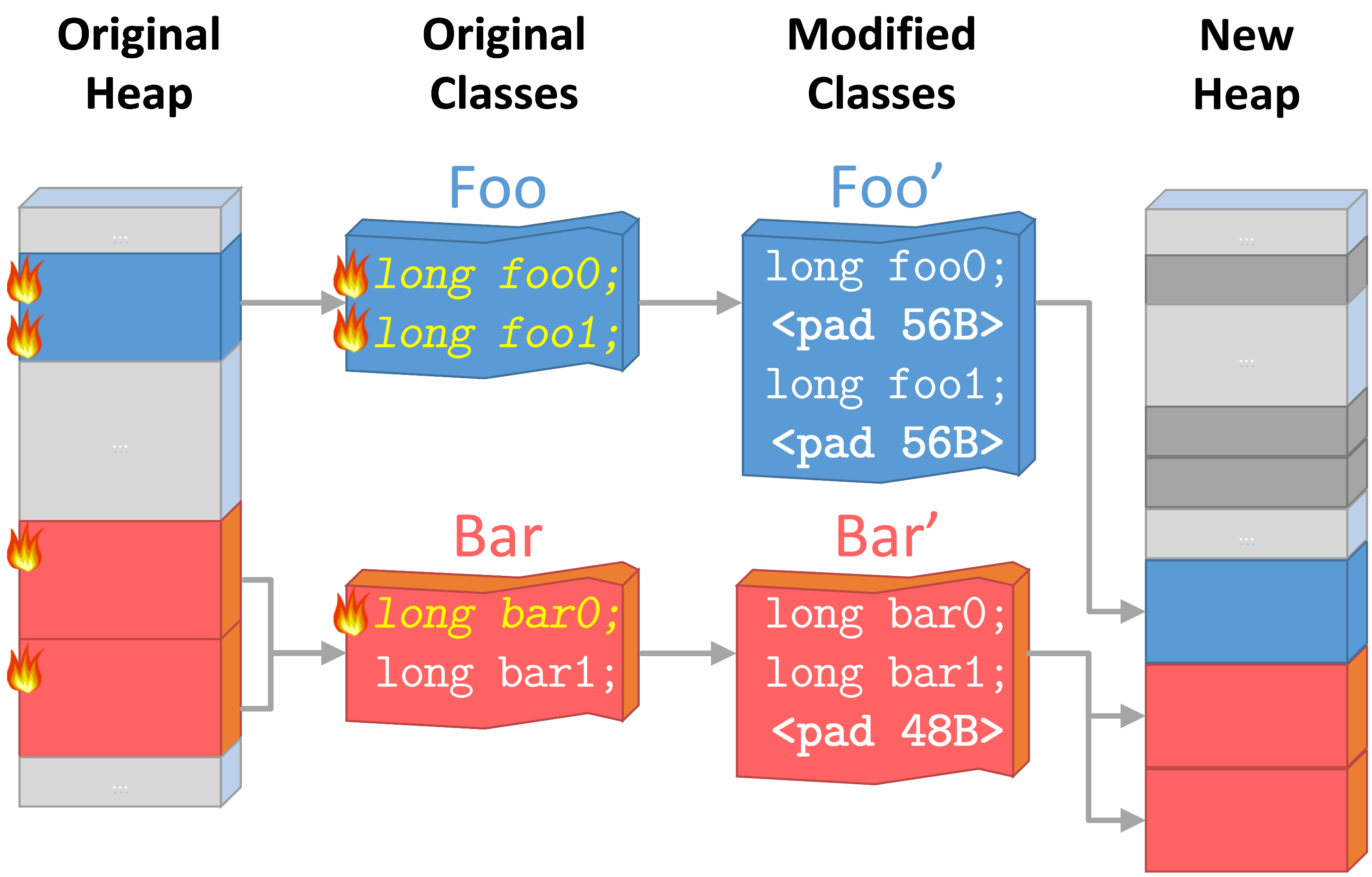
Top Level Flow



Top Level Flow



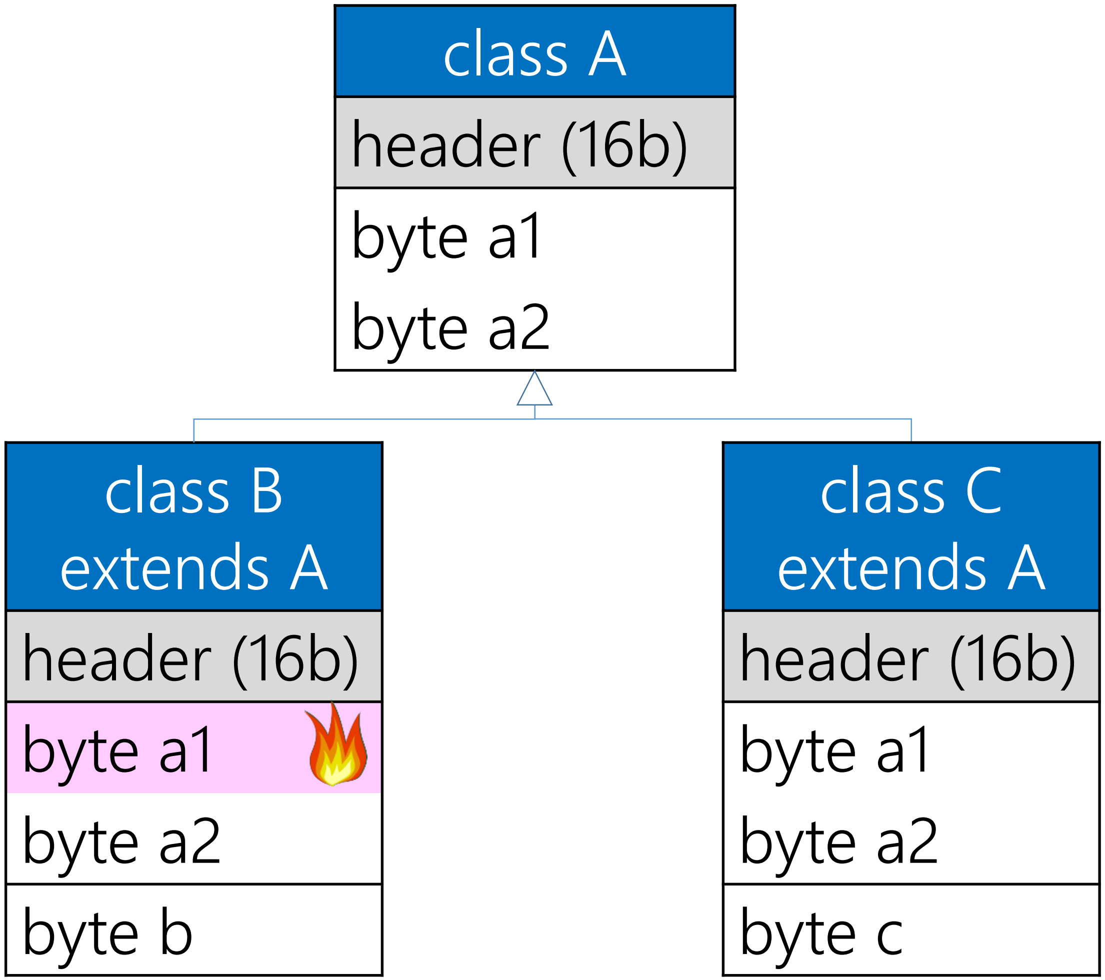
Top Level Flow



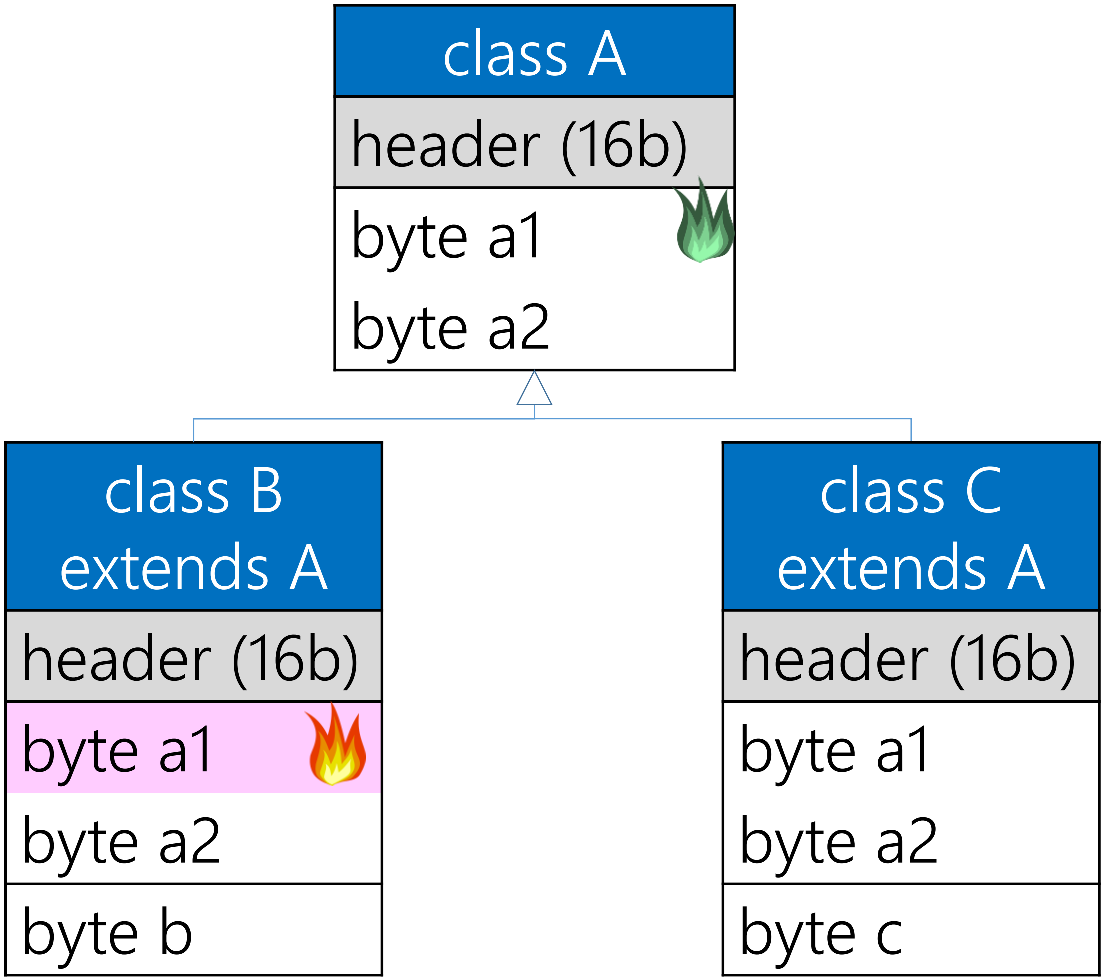
Padding

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

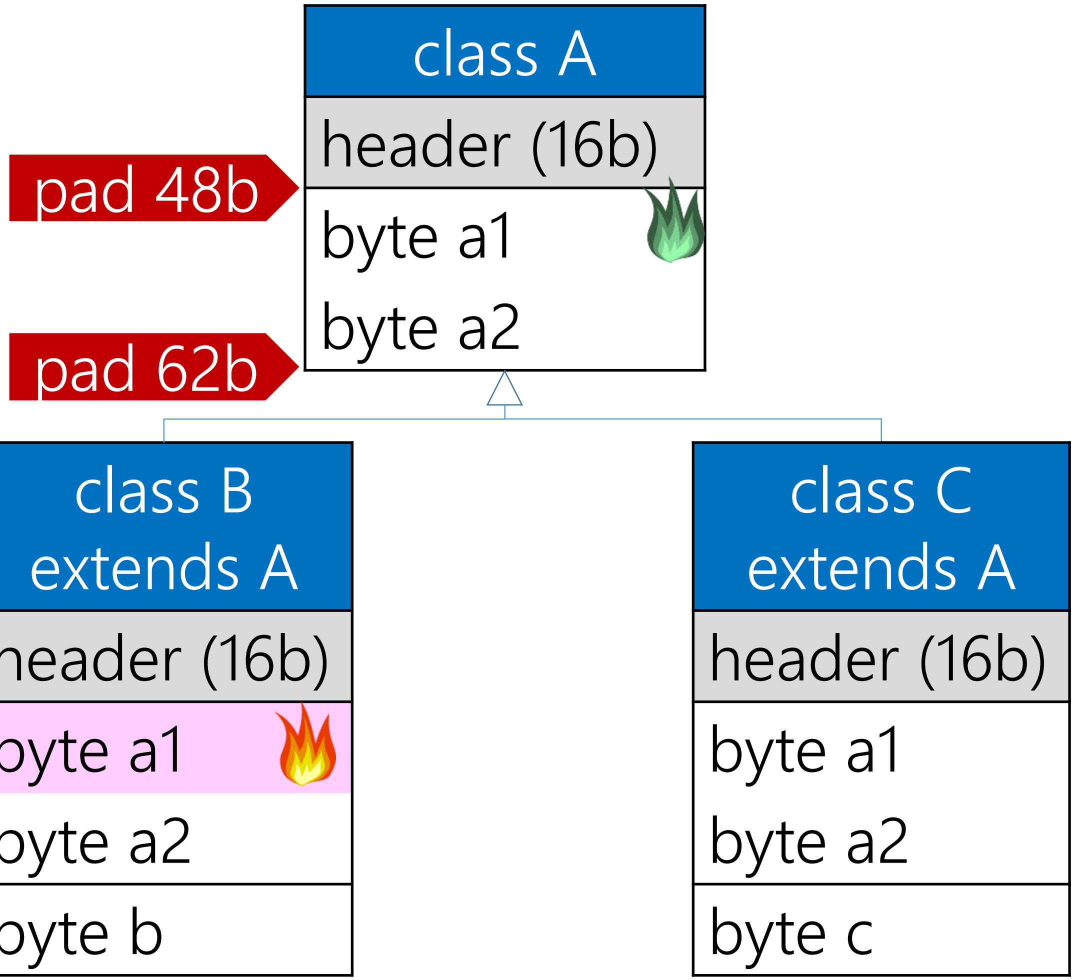
Padding - Inheritance



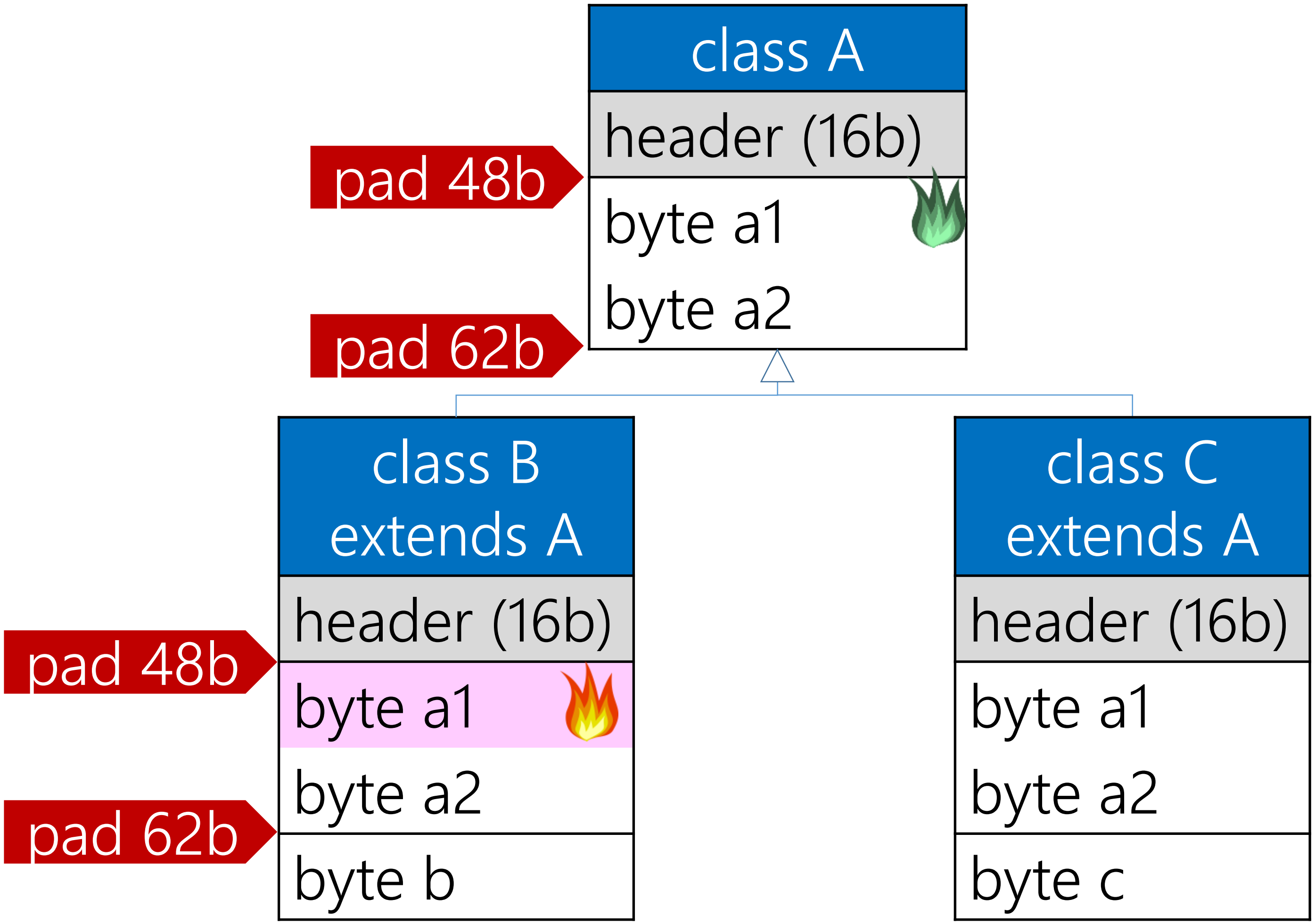
Padding - Inheritance



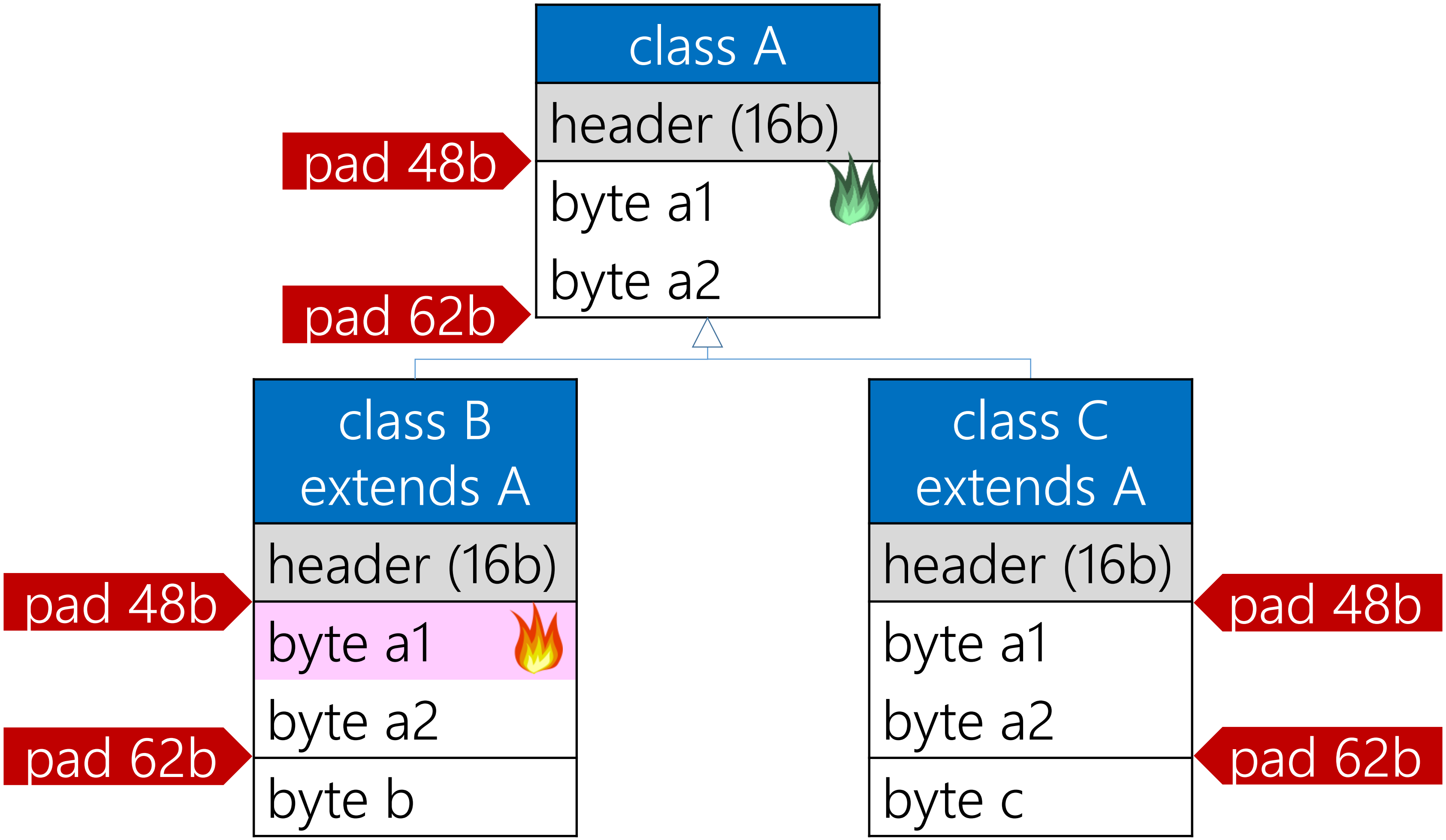
Padding - Inheritance



Padding - Inheritance



Padding - Inheritance



Repair

- Trace all strong+weak roots in the system
- Traverse heap and find targeted instances
  - Live $\Rightarrow$ Relocate & pad, store forwarding pointer
  - Dead $\Rightarrow$ 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

```
class Sequence {
  protected volatile long value;
}
```

value

Disruptor & Spring Reactor

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

56 bytes

```
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?

Disruptor & Spring Reactor

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

56 bytes

```
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?

value	a-g	i-o

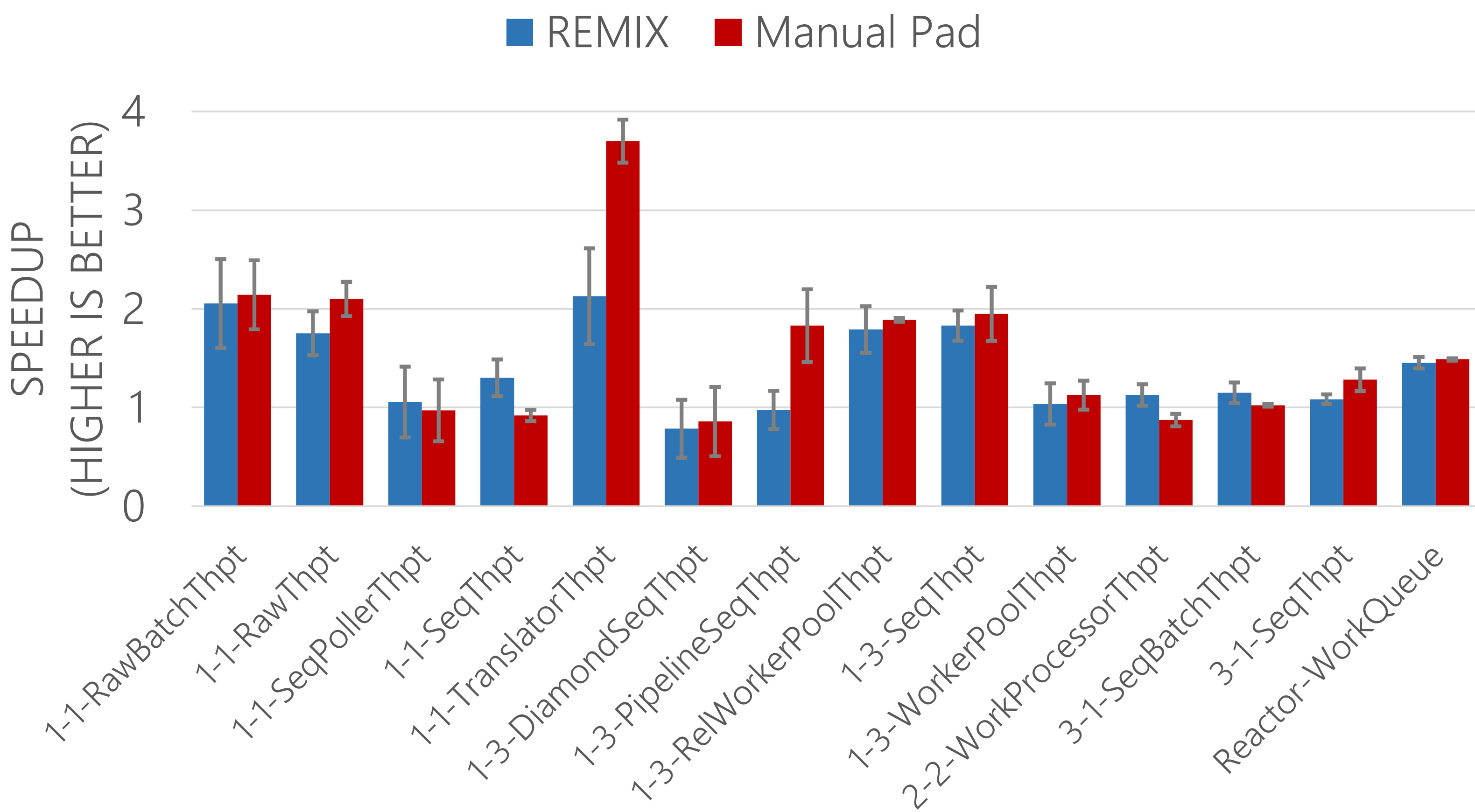
  - No! – Class loader optimizes aggressively, lays out *value* to before *a*.

Disruptor & Spring Reactor

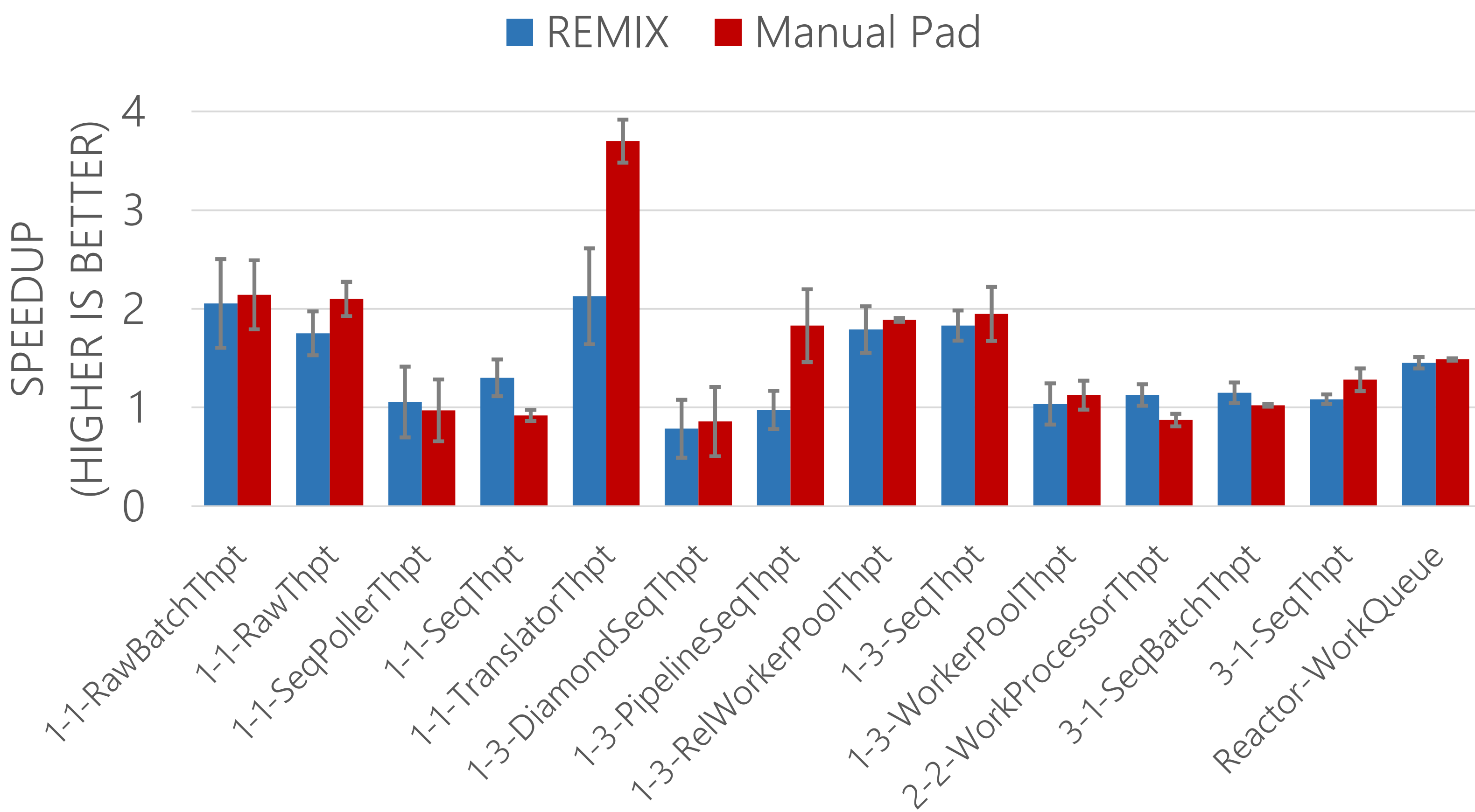
- A complex hierarchy forces field order

```
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
}
```

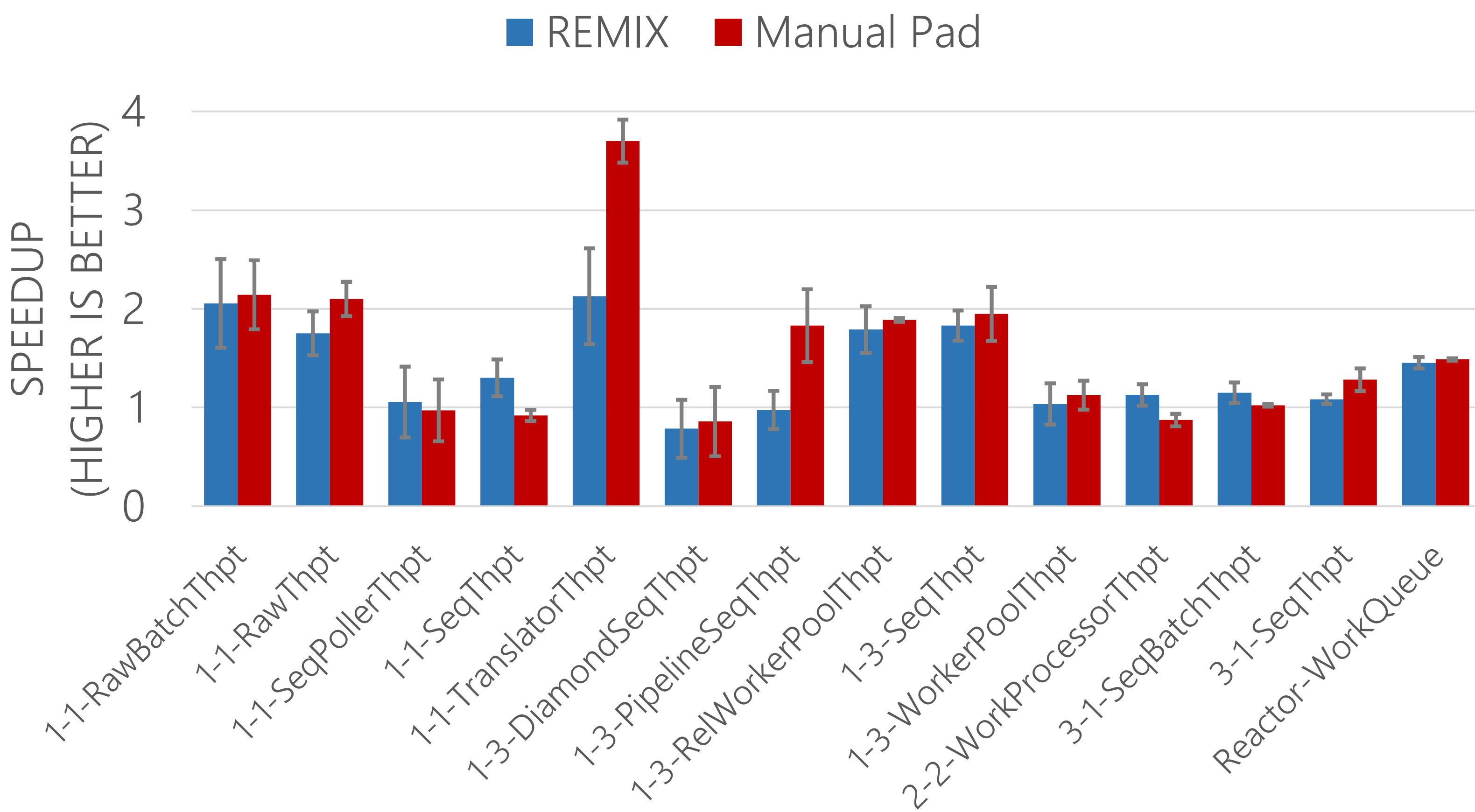
Disruptor + Spring Reactor



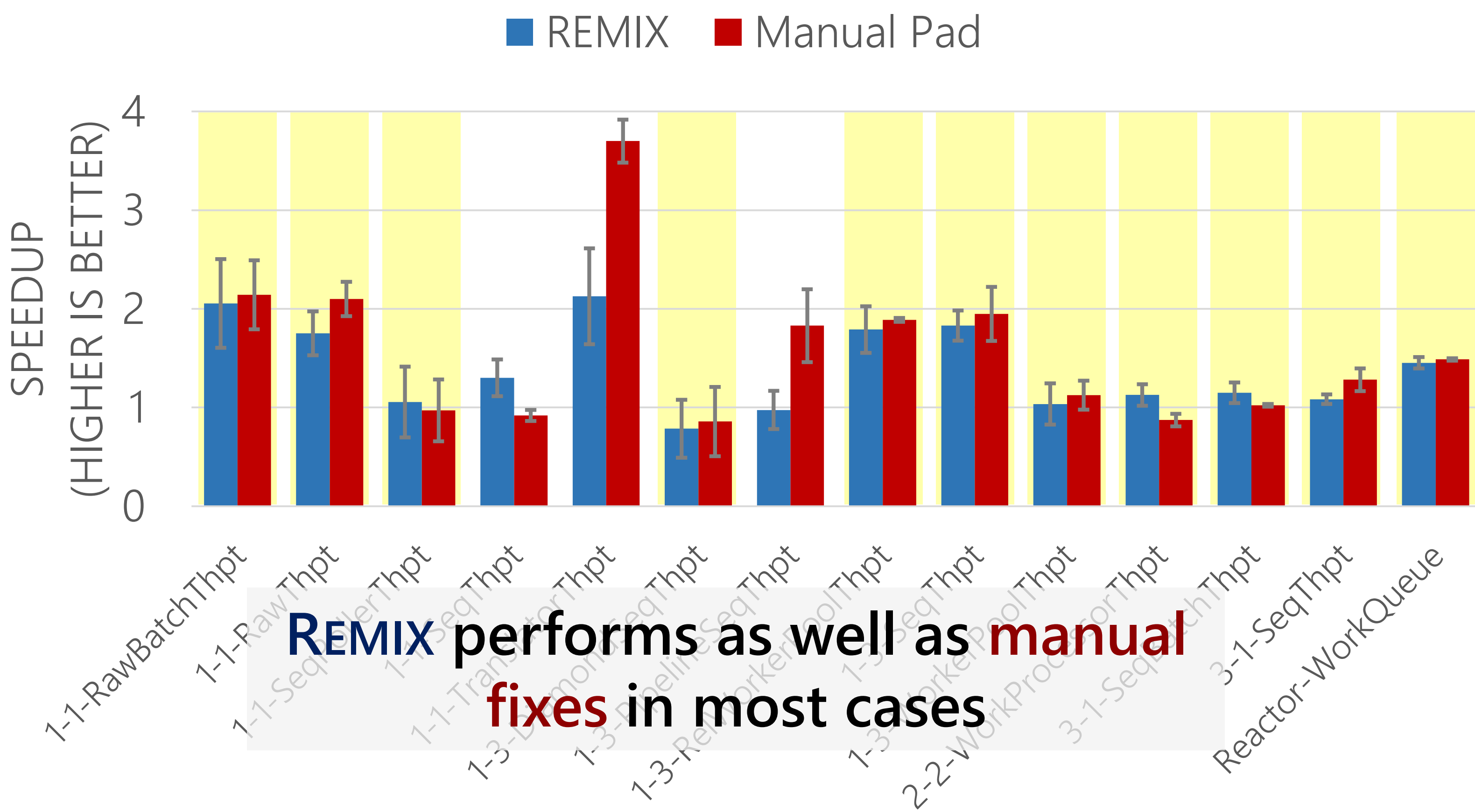
Disruptor + Spring Reactor



Disruptor + Spring Reactor

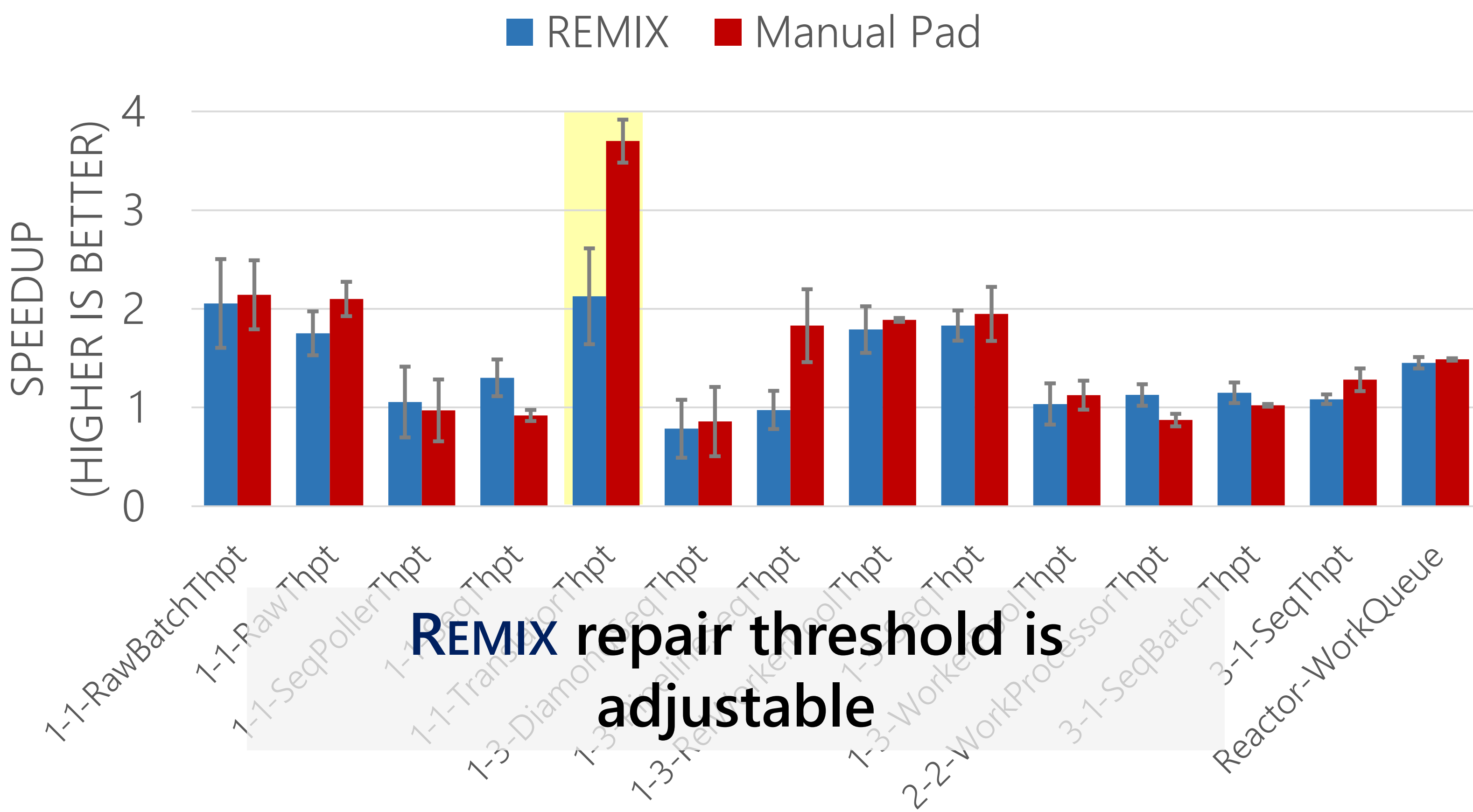


Disruptor + Spring Reactor



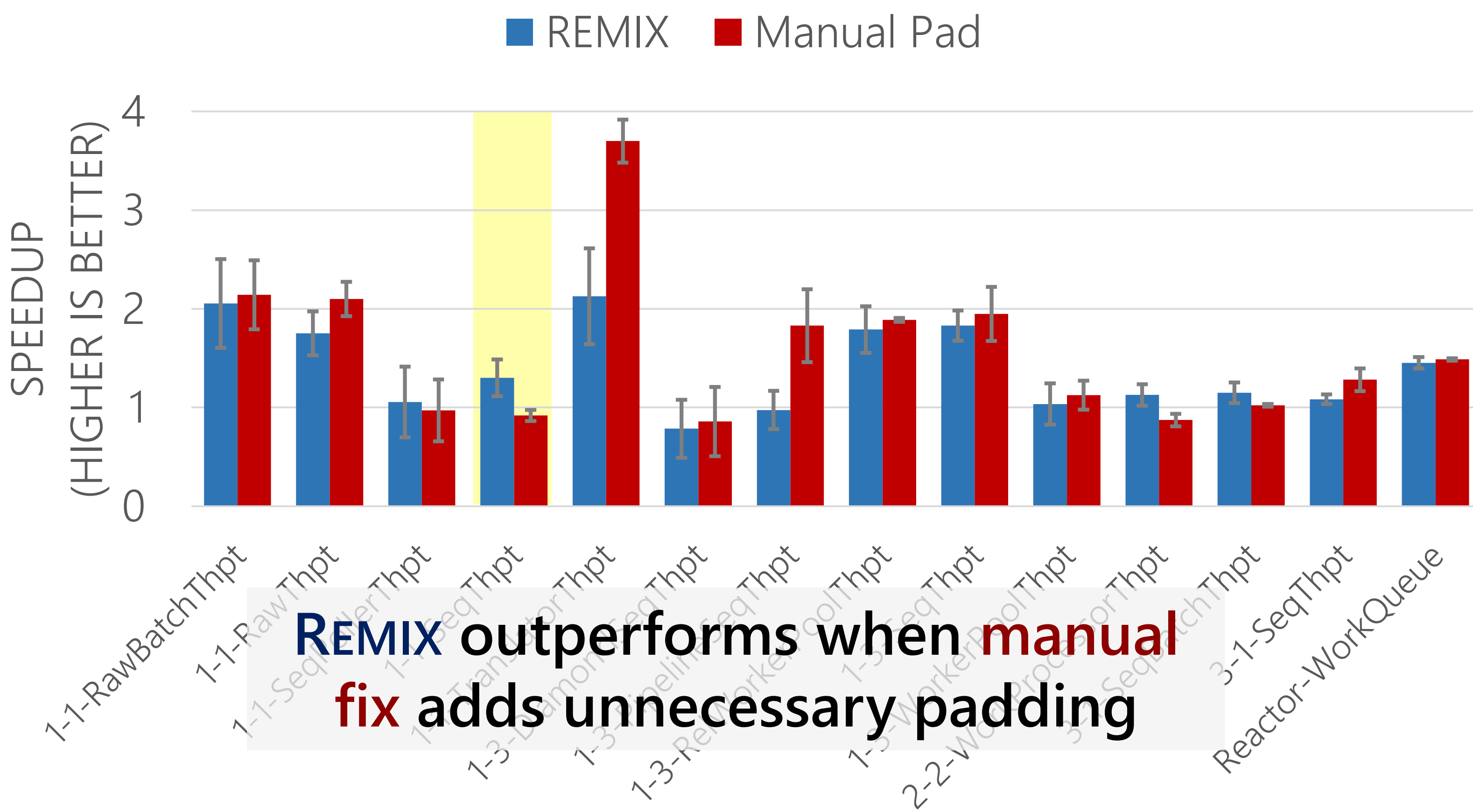
REMIX performs as well as manual fixes in most cases

Disruptor + Spring Reactor



REMIX repair threshold is adjustable

Disruptor + Spring Reactor



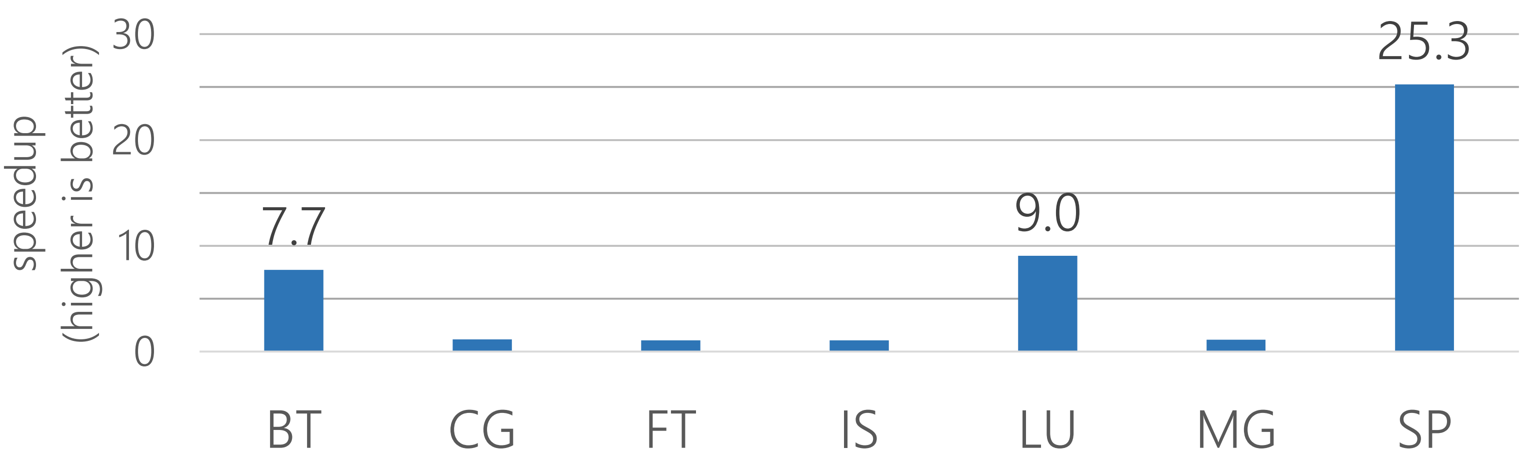
REMIX outperforms when manual fix adds unnecessary padding

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

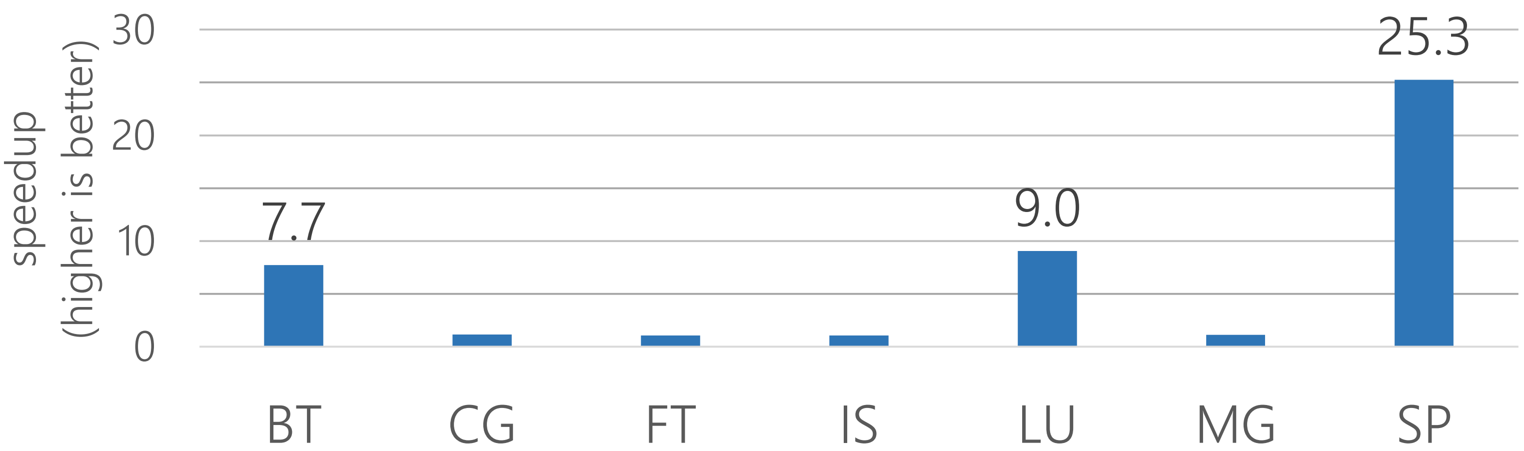
NAS Parallel Benchmarks

- 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



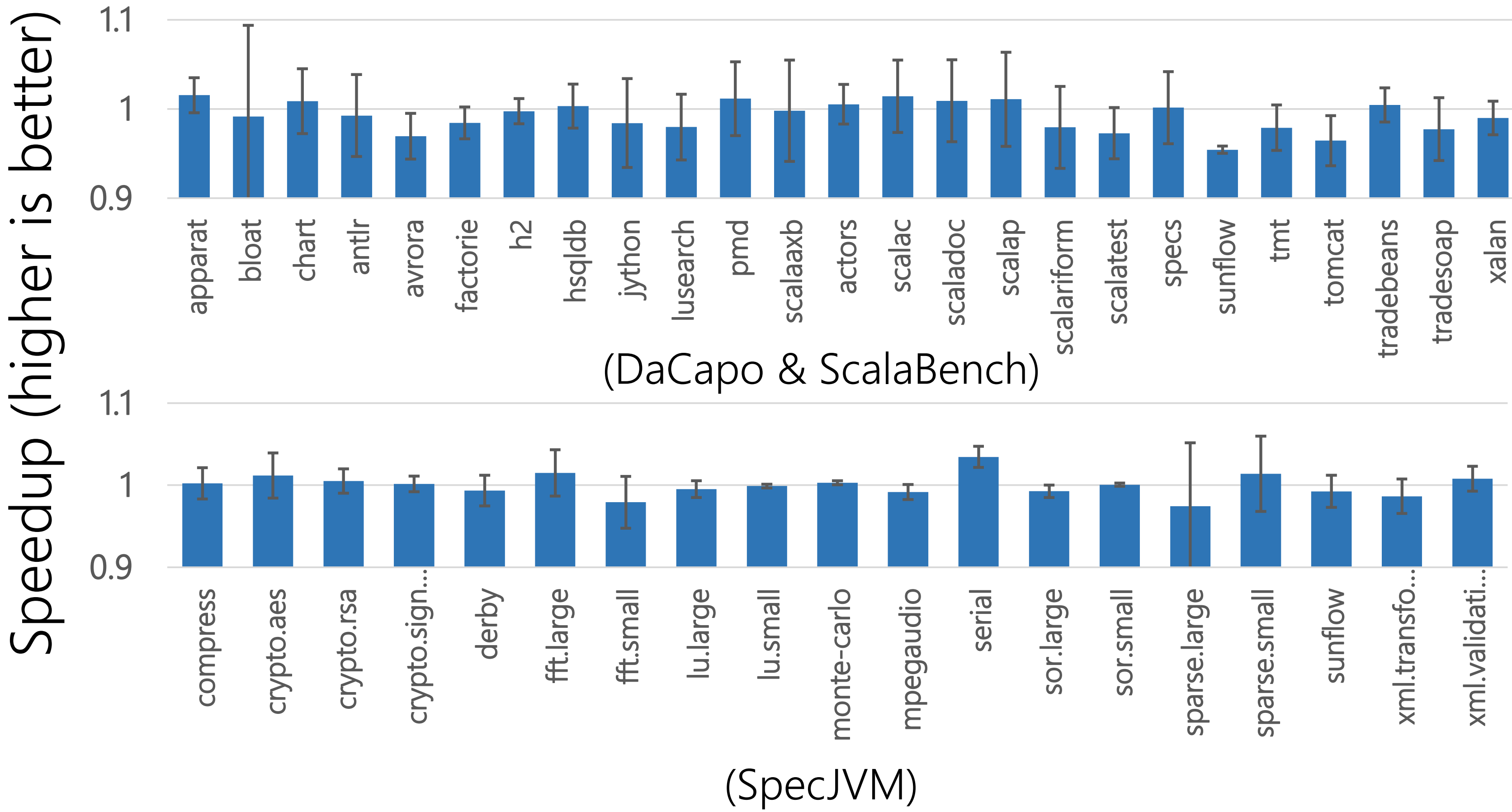
NAS Parallel Benchmarks

- 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

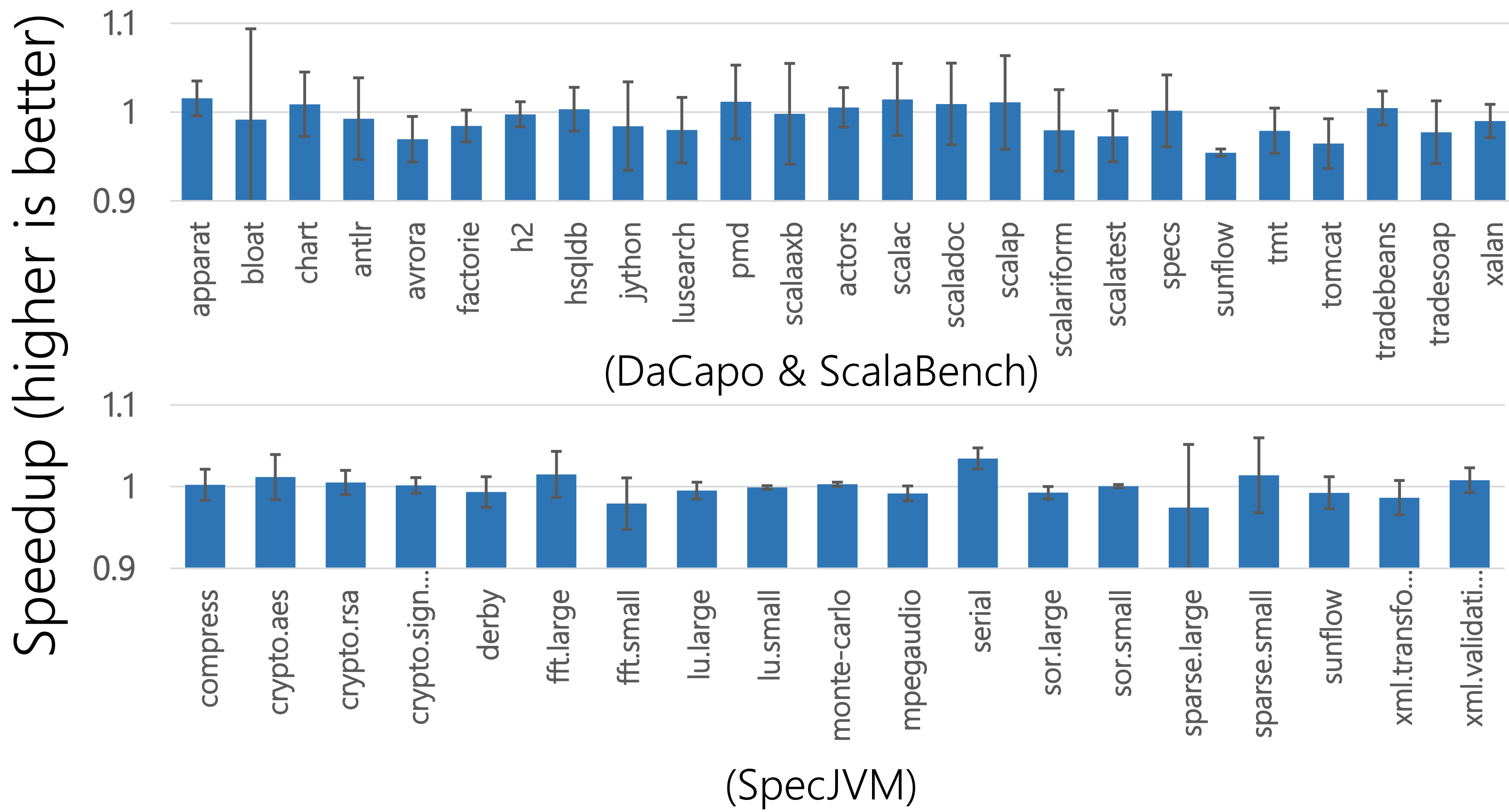


No Contention == No Impact

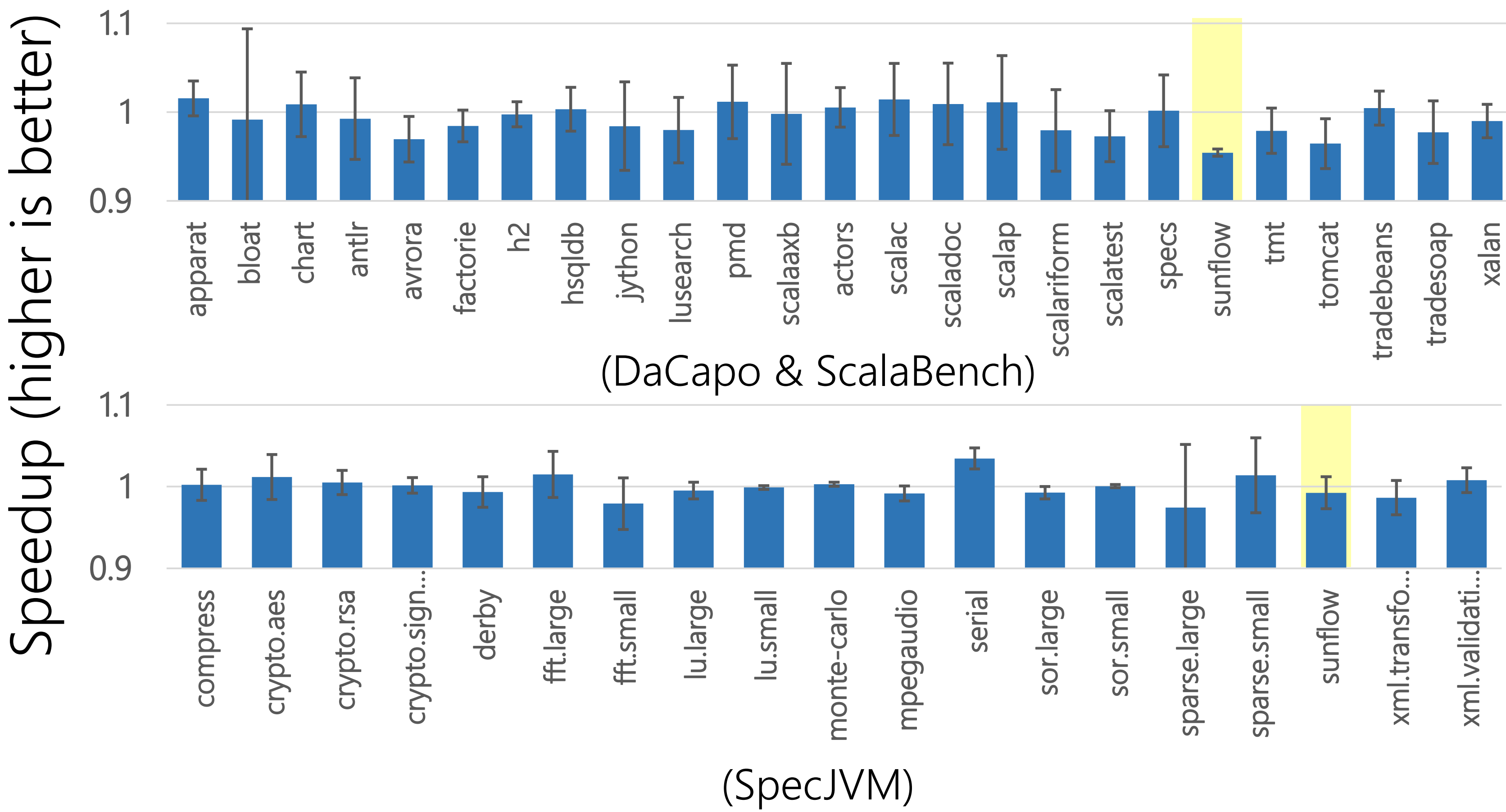
No Contention == No Impact



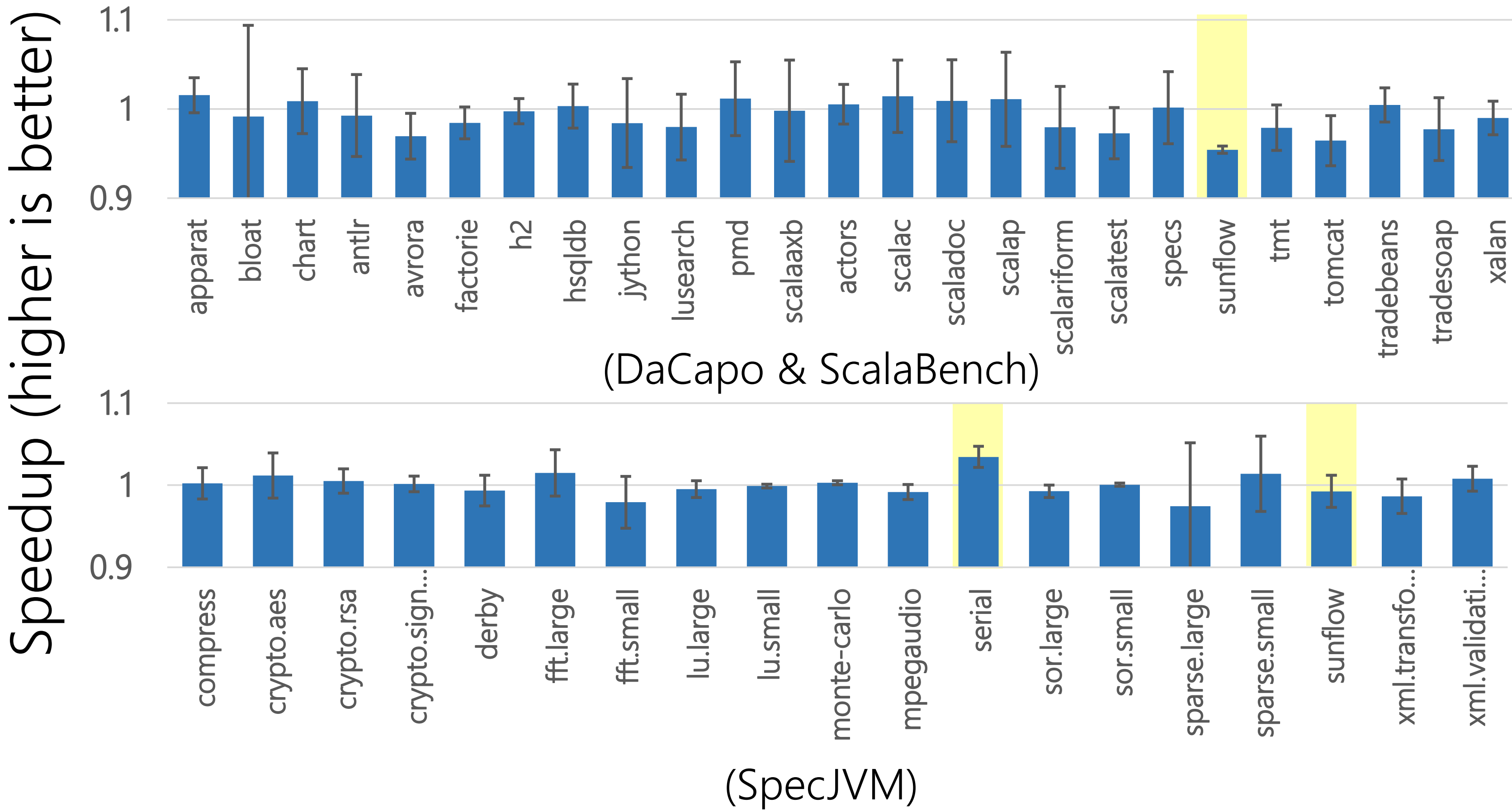
No Contention == No Impact



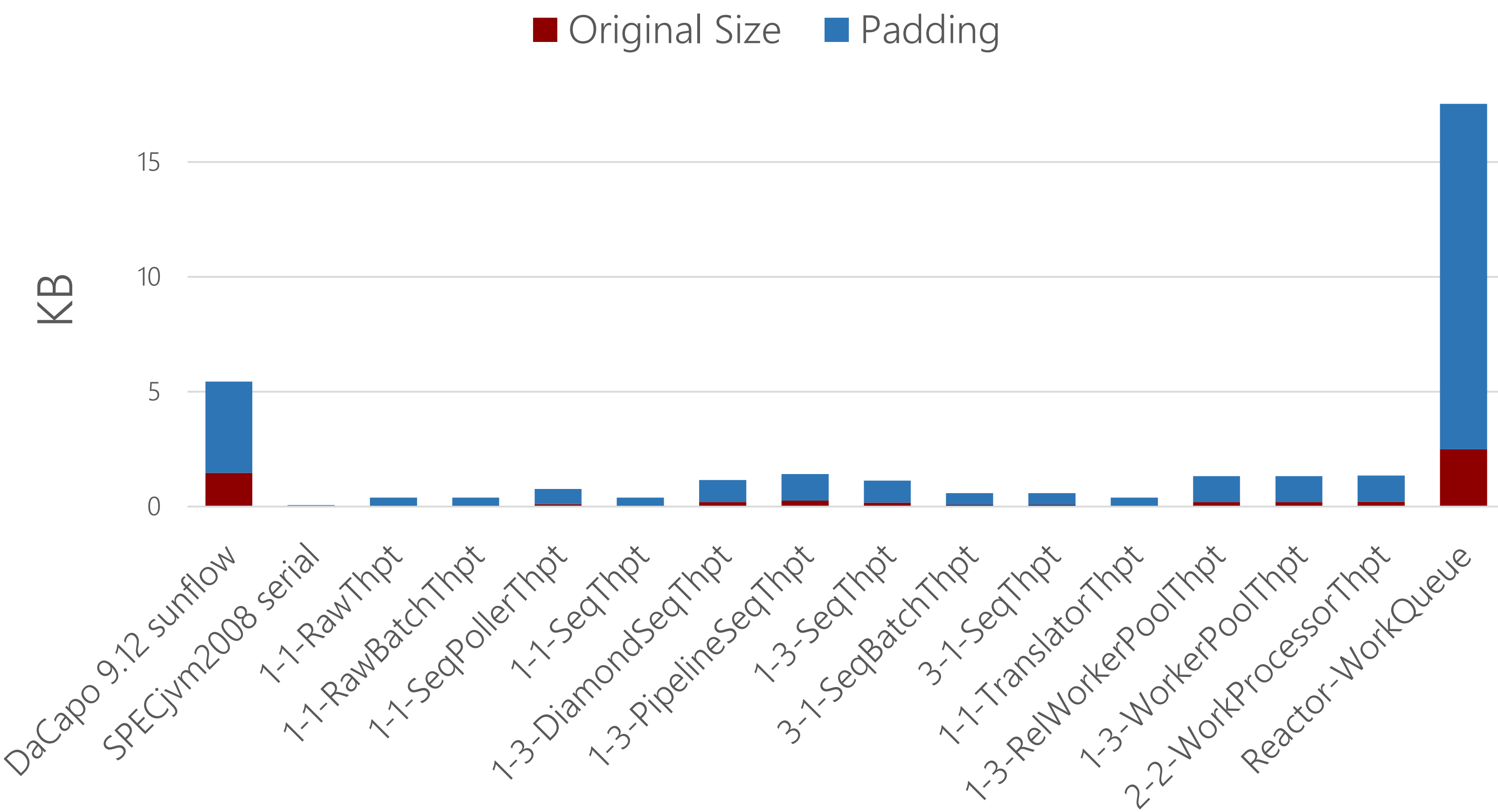
No Contention == No Impact



No Contention == No Impact

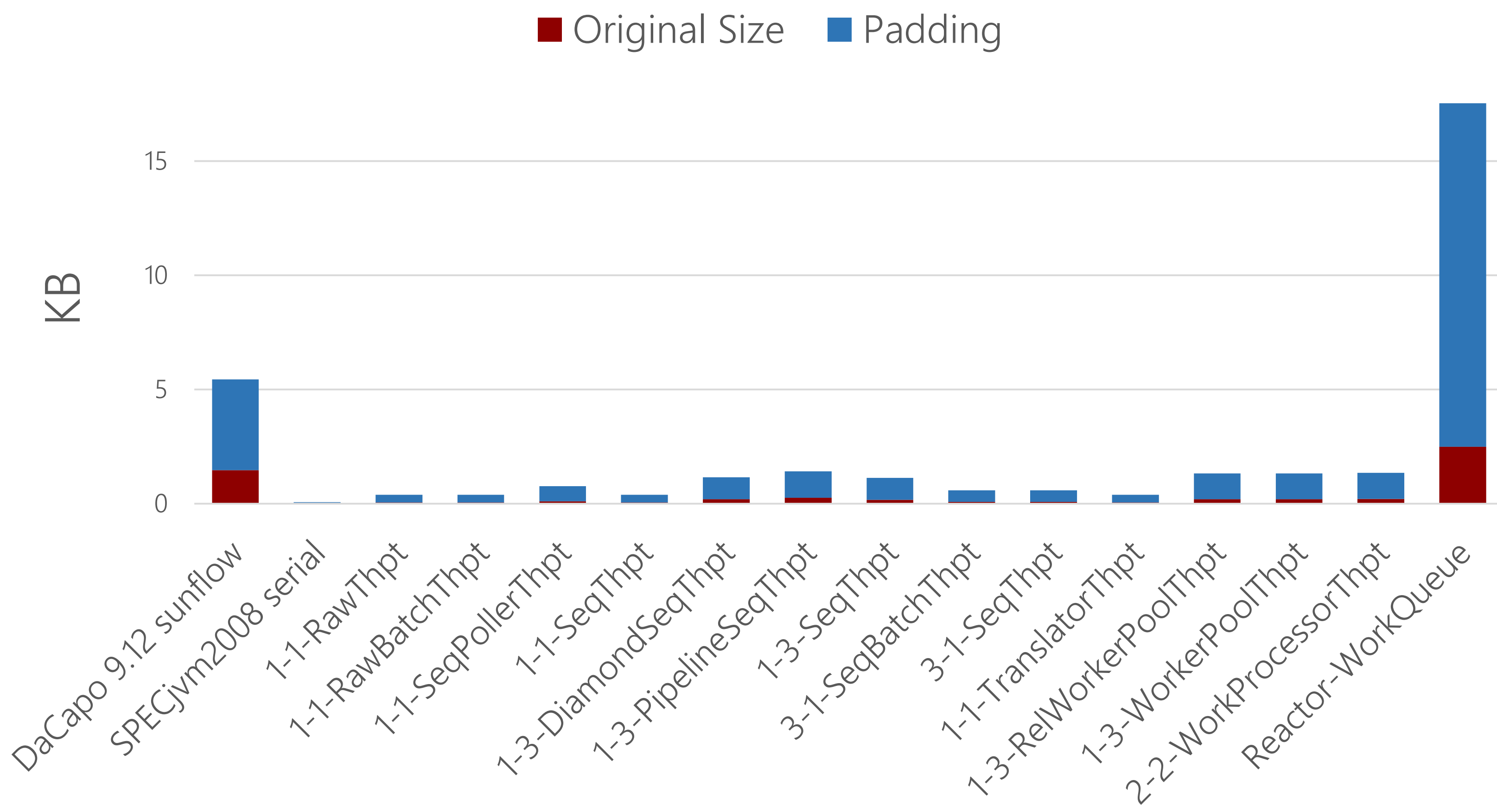


Padding



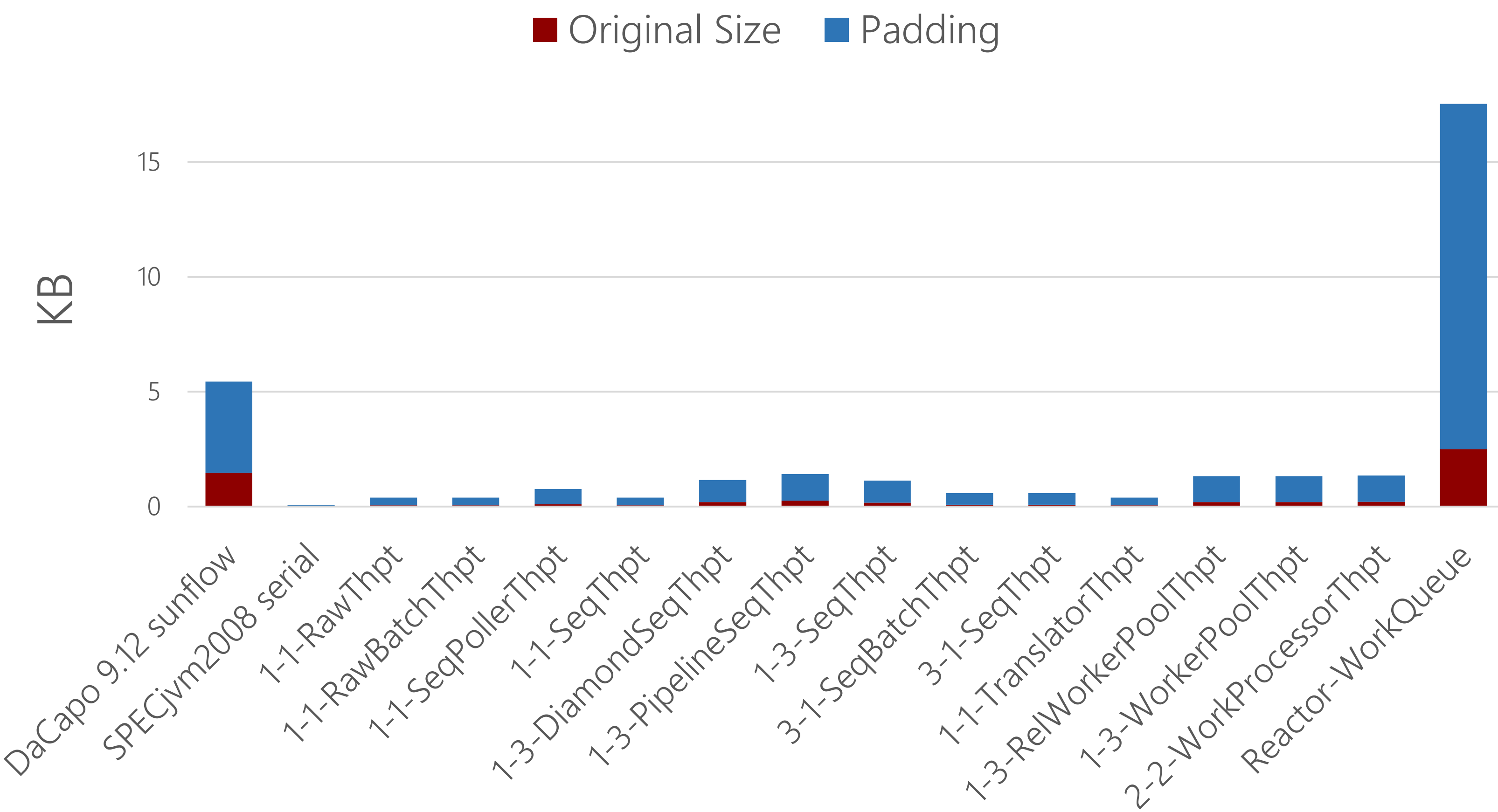
Original Size
Padding
KB
15
10
5
0
DaCapo 9.12 sunflow
SPECjvm2008 serial
1-1-RawThpt
1-1-RawBatchThpt
1-1-SeqPollerThpt
1-1-SeqThpt
1-3-DiamondSeqThpt
1-3-PipelineSeqThpt
1-3-SeqThpt
3-1-SeqBatchThpt
3-1-SeqThpt
1-1-TranslatorThpt
1-3-RelWorkerPoolThpt
1-3-WorkerPoolThpt
2-2-WorkProcessorThpt
Reactor-WorkQueue

Padding



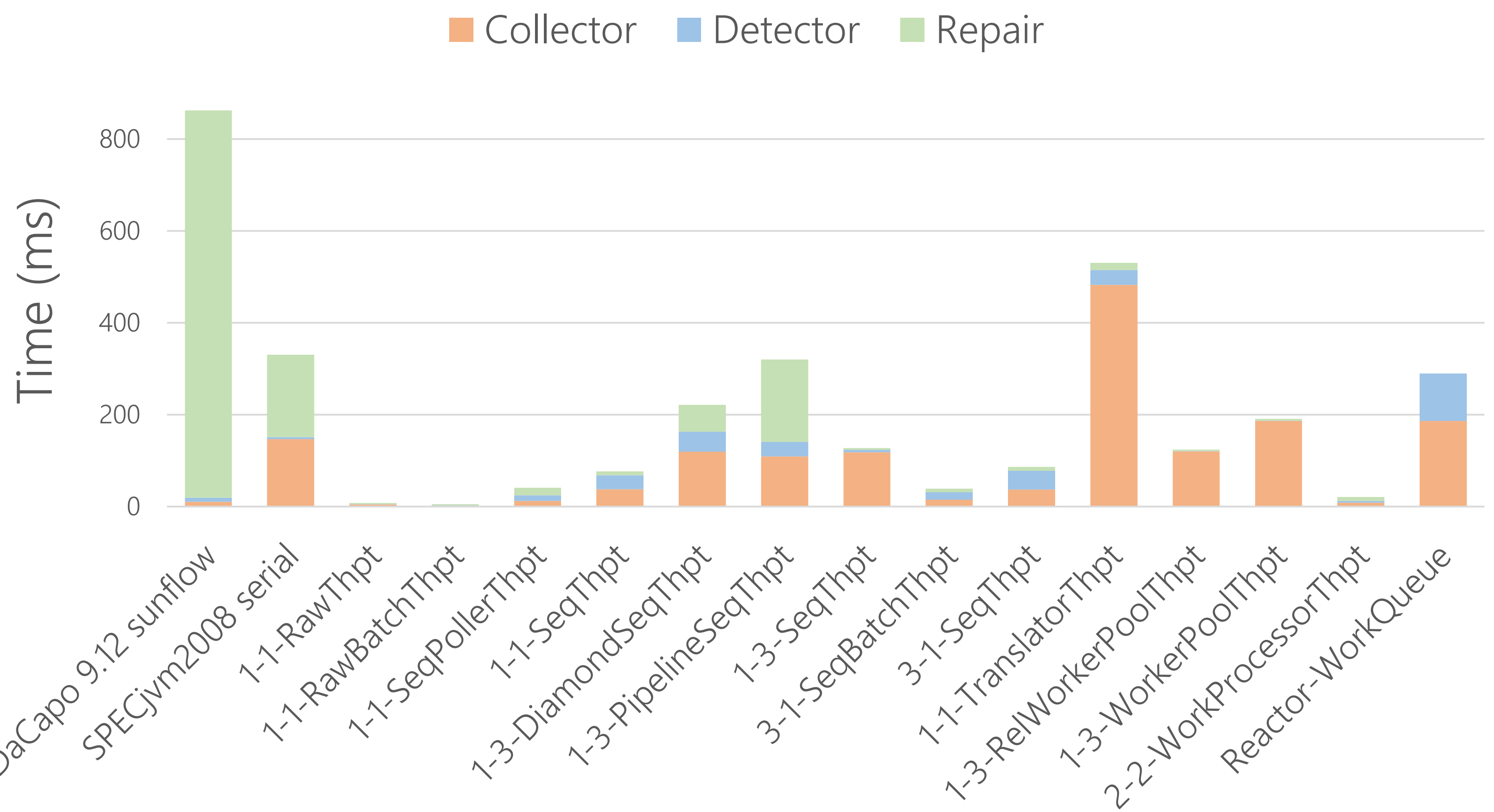
Original Size
Padding
KB
15
10
5
0
DaCapo 9.12 sunflow
SPECjvm2008 serial
1-1-RawThpt
1-1-RawBatchThpt
1-1-SeqPollerThpt
1-1-SeqThpt
1-3-DiamondSeqThpt
1-3-PipelineSeqThpt
1-3-SeqThpt
3-1-SeqBatchThpt
3-1-SeqThpt
1-1-TranslatorThpt
1-3-RelWorkerPoolThpt
1-3-WorkerPoolThpt
2-2-WorkProcessorThpt
Reactor-WorkQueue

Padding

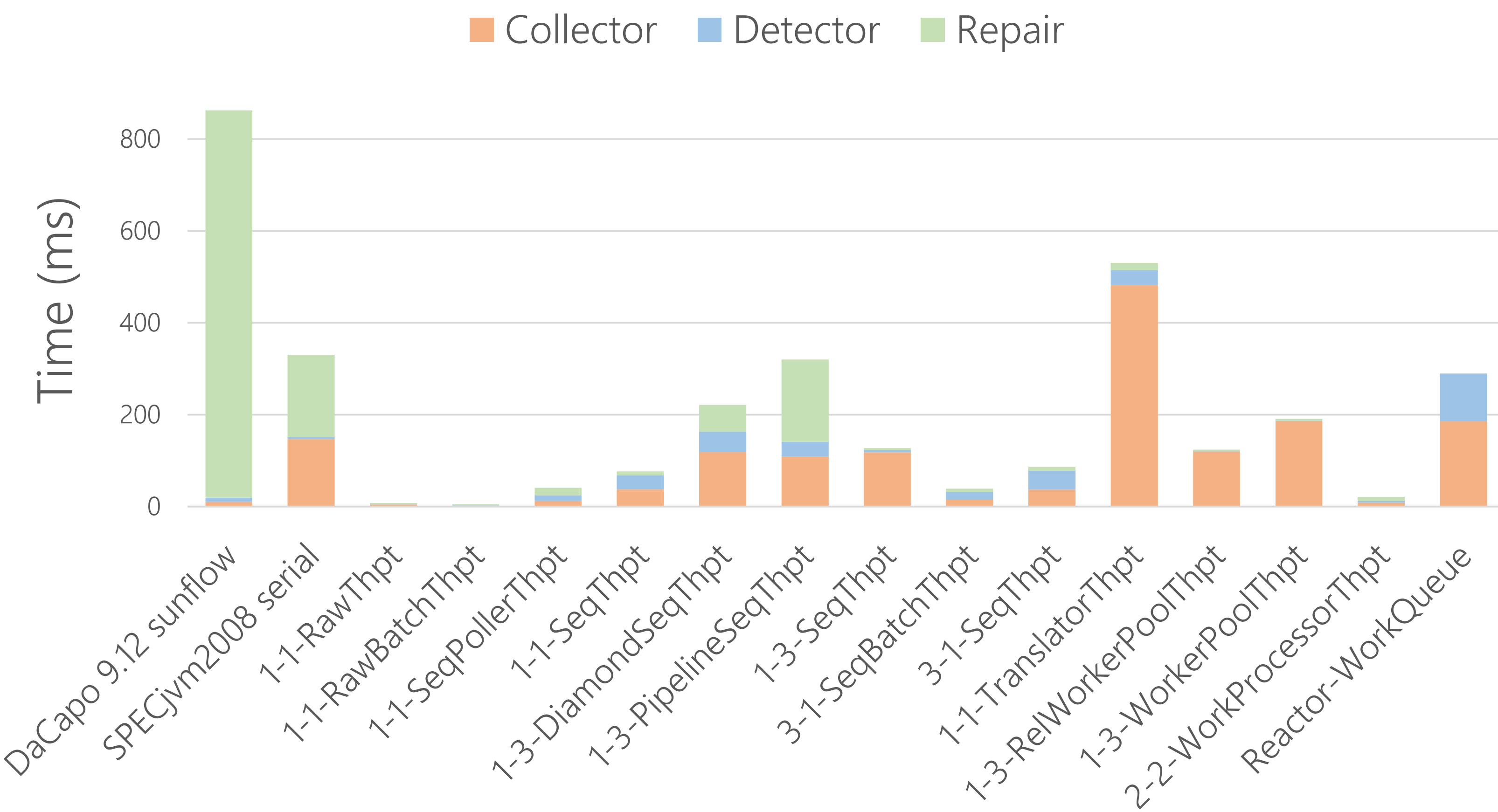


Original Size
Padding
KB
15
10
5
0
DaCapo 9.12 sunflow
SPECjvm2008 serial
1-1-RawThpt
1-1-RawBatchThpt
1-1-SeqPollerThpt
1-1-SeqThpt
1-3-DiamondSeqThpt
1-3-PipelineSeqThpt
1-3-SeqThpt
3-1-SeqBatchThpt
3-1-SeqThpt
1-1-TranslatorThpt
1-3-RelWorkerPoolThpt
1-3-WorkerPoolThpt
2-2-WorkProcessorThpt
Reactor-WorkQueue

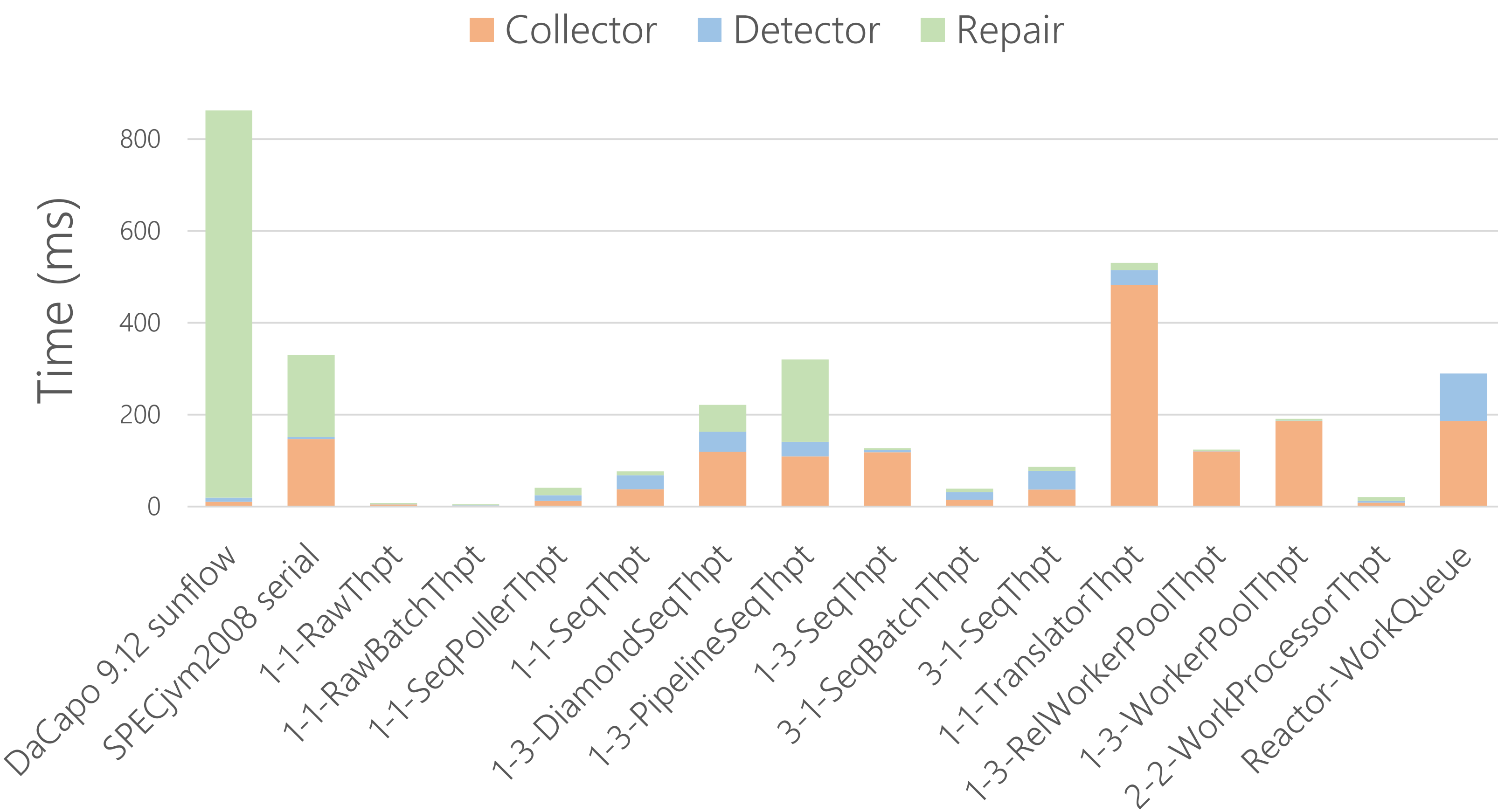
Performance



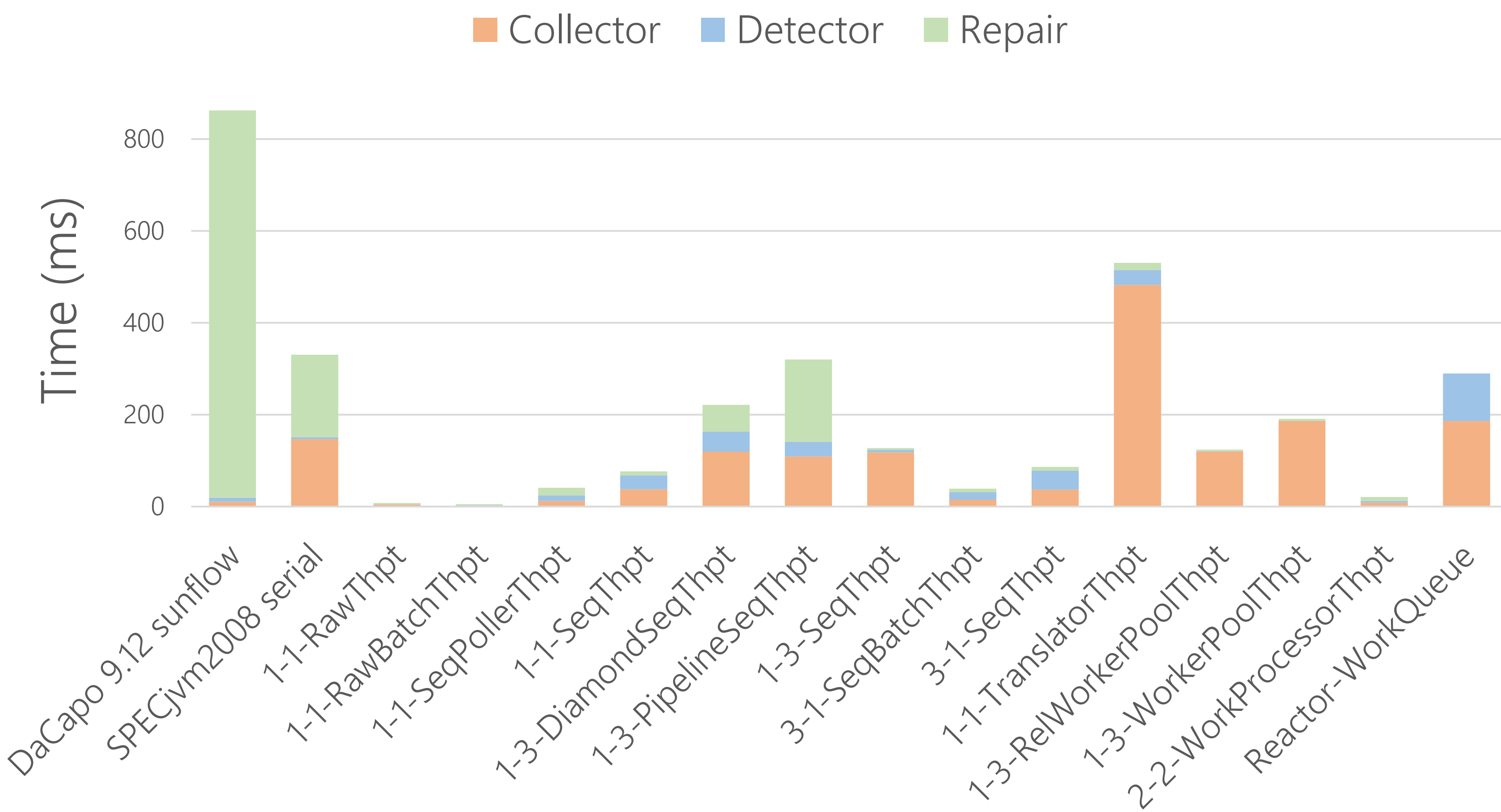
Performance



Performance



Performance

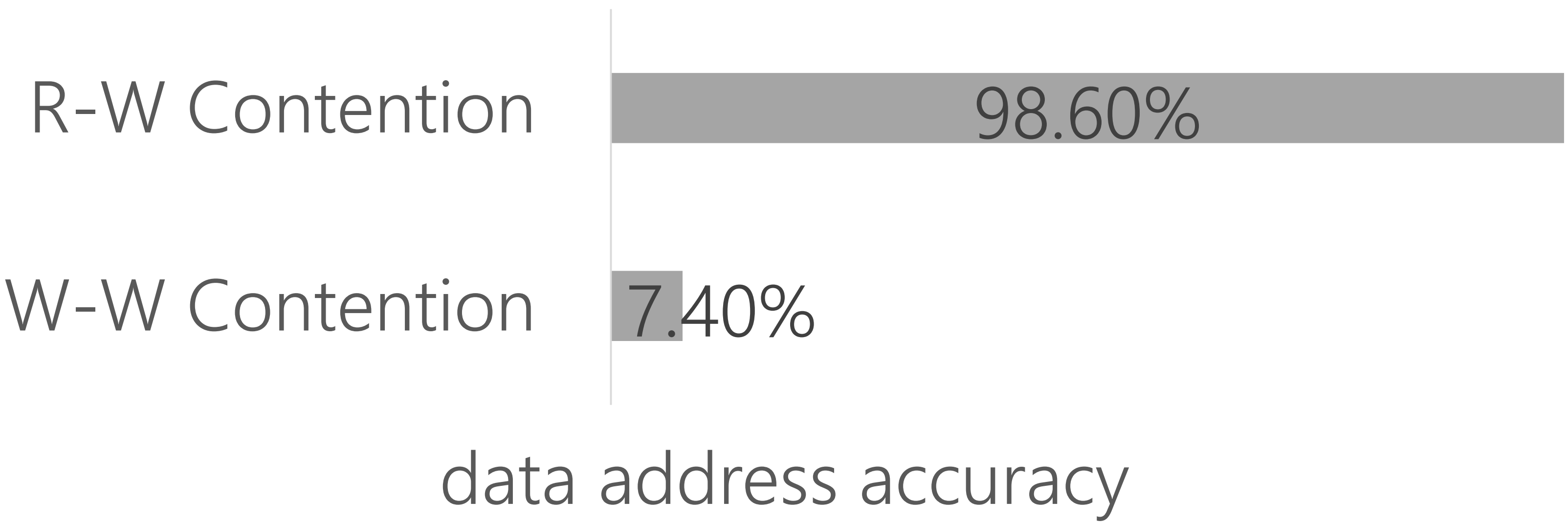


Conclusions

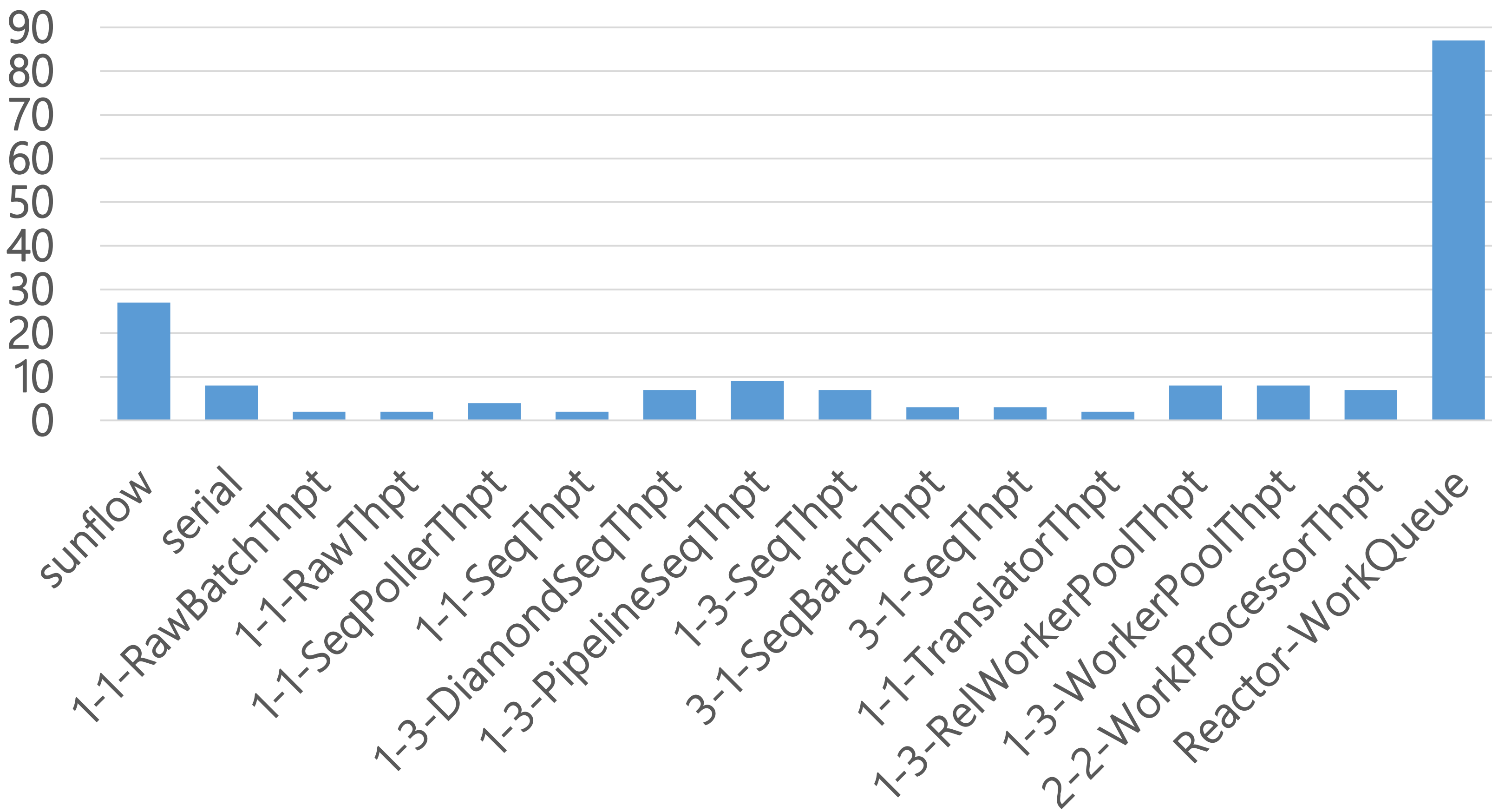
- Cache contention afflicts managed languages
- Dynamic information opens up new avenues for optimization
- Performance counters are a gold mine
- REMIX can simplify programmers' lives by **automatically fixing** false sharing bugs
- https://github.com/upenn-acg/REMIX (GPLv2)

Q&A

PEBS HITM Event Accuracy



Objects Moved



sun.misc.Unsafe

- Tracks unsafe access, prevents padding

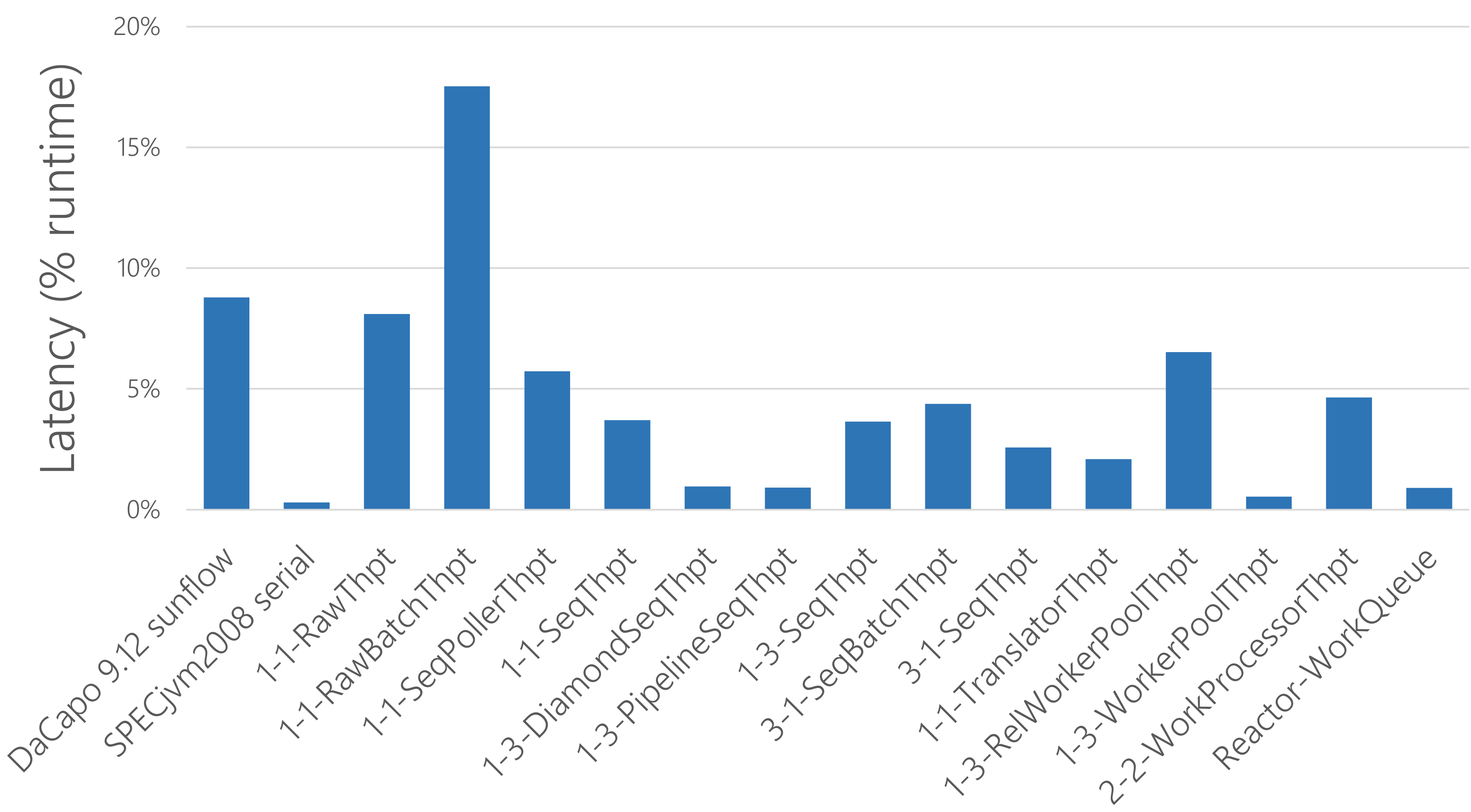
```
FLD_OFFS = U.objectFieldOffset(k.getDeclaredField("fld"));
Unsafe.putLong(o, FLD_OFFS, value);
```

- REMIX Extended **Unsafe** :

```
U.registerFieldOffset(k.getDeclaredField("fld"),
                      k.getDeclaredField("FLD_OFFS"));
Unsafe.putLong(o, FLD_OFFS, value);
```

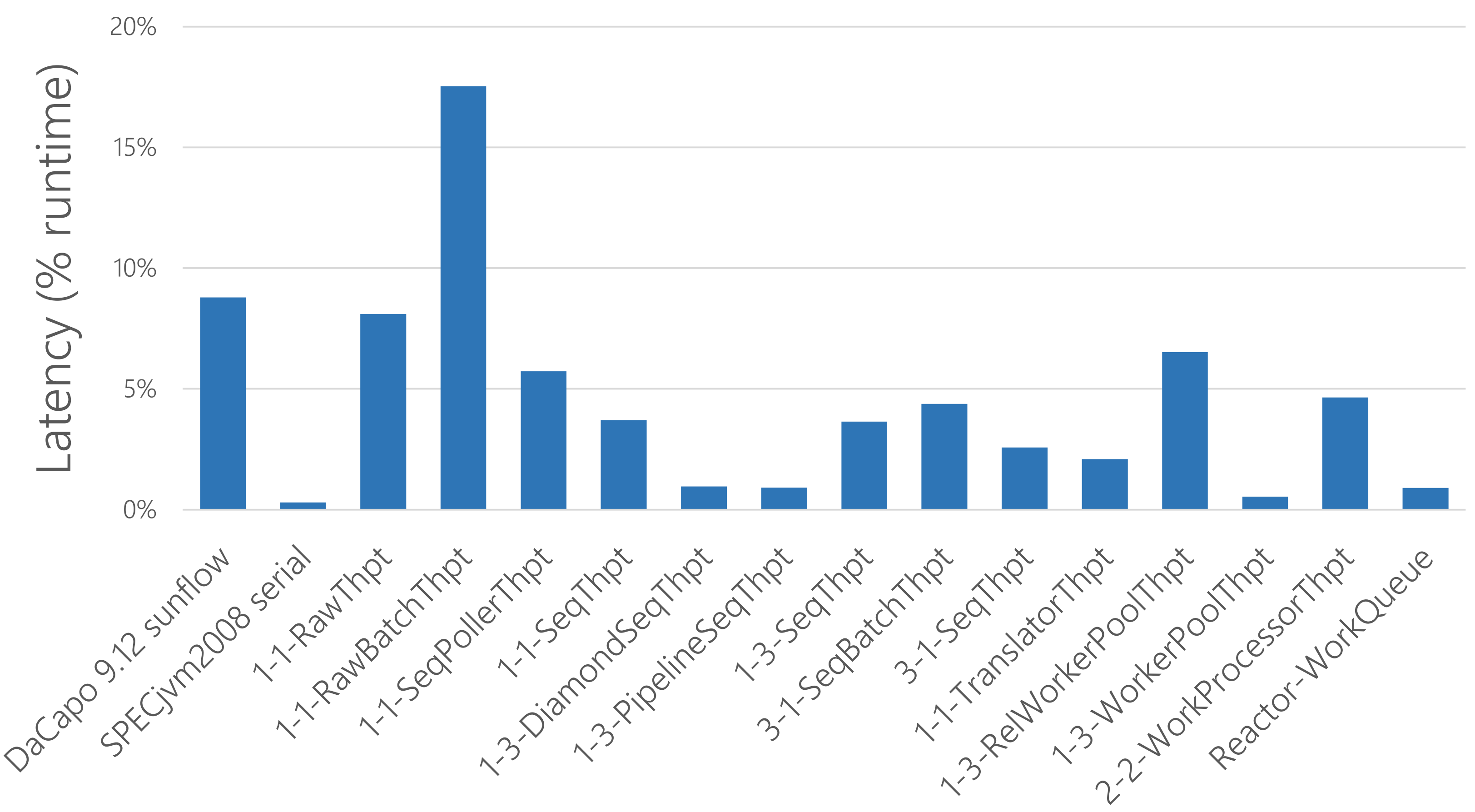
- Just as fast, enables padding

Time-to-fix



Latency (% runtime)
20%
15%
10%
5%
0%
DaCapo 9.12 sunflow
SPECjvm2008 serial
1-1-RawThpt
1-1-RawBatchThpt
1-1-SeqPollerThpt
1-1-SeqThpt
1-3-DiamondSeqThpt
1-3-PipelineSeqThpt
1-3-SeqThpt
3-1-SeqBatchThpt
3-1-SeqThpt
1-1-TranslatorThpt
1-3-RelWorkerPoolThpt
1-3-WorkerPoolThpt
2-2-WorkProcessorThpt
Reactor-WorkQueue

Time-to-fix

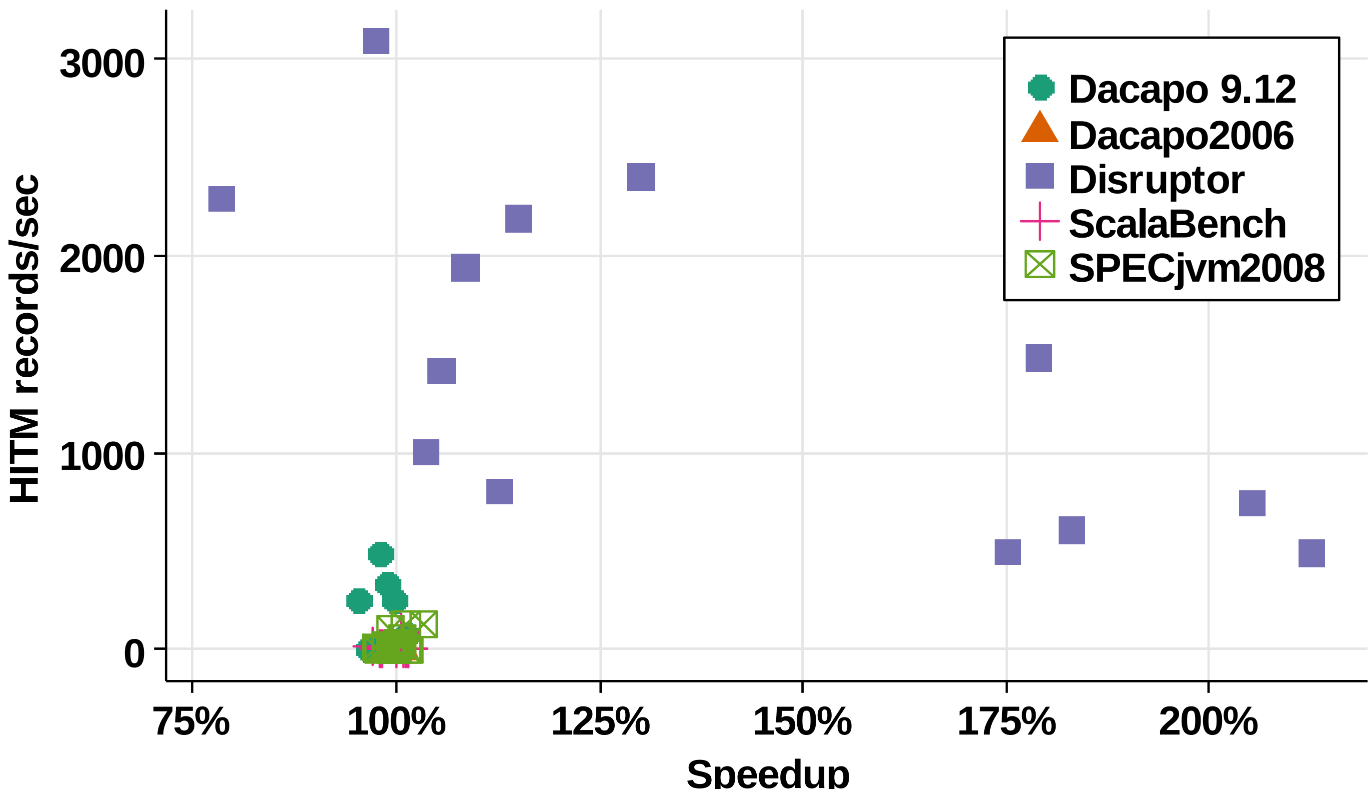


Latency (% runtime)
20%
15%
10%
5%
0%
DaCapo 9.12 sunflow
SPECjvm2008 serial
1-1-RawThpt
1-1-RawBatchThpt
1-1-SeqPollerThpt
1-1-SeqThpt
1-3-DiamondSeqThpt
1-3-PipelineSeqThpt
1-3-SeqThpt
3-1-SeqBatchThpt
3-1-SeqThpt
1-1-TranslatorThpt
1-3-RelWorkerPoolThpt
1-3-WorkerPoolThpt
2-2-WorkProcessorThpt
Reactor-WorkQueue

Performance

Benchmark(s)	Classes Loaded	Processing (ms)	Heap Size (MB)
DaCapo Sunflow	1879	1.47	854
Disruptor	700-900	0.5-1.3	40-85
SpringReactor WorkQueue	1617	0.748	41
SpecJVM serial	1498	1.21	1603

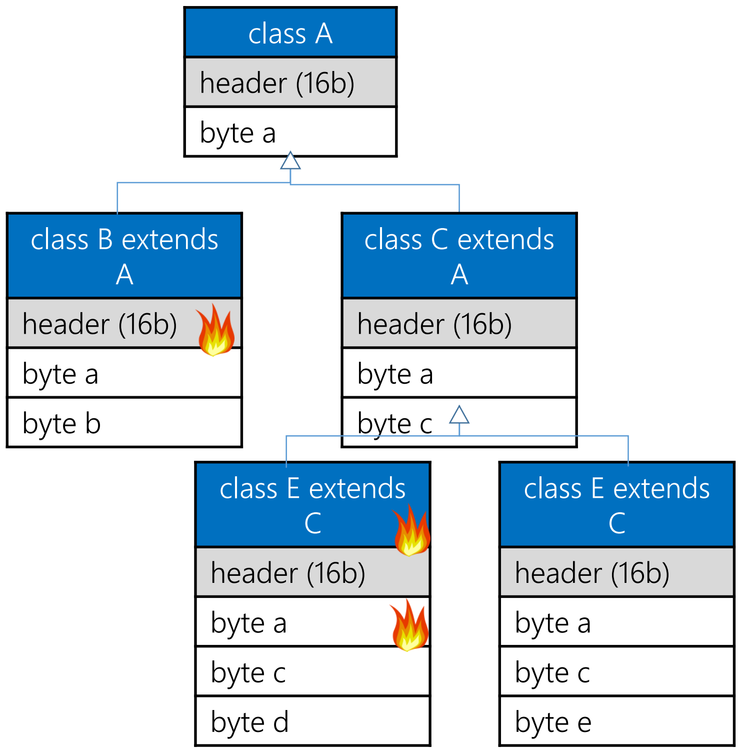
REMIX Speedup vs HITM rate



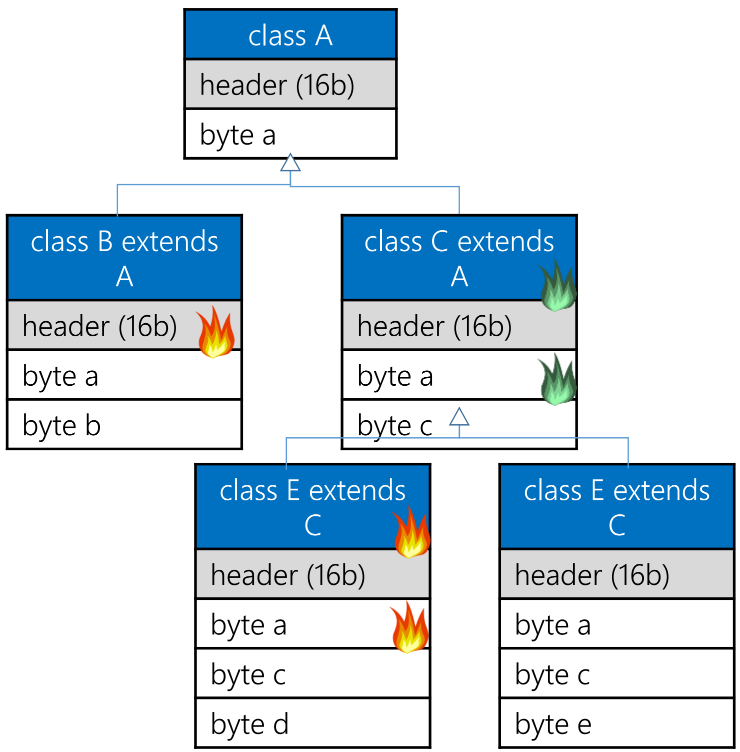
Size-Preserving-Klass

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

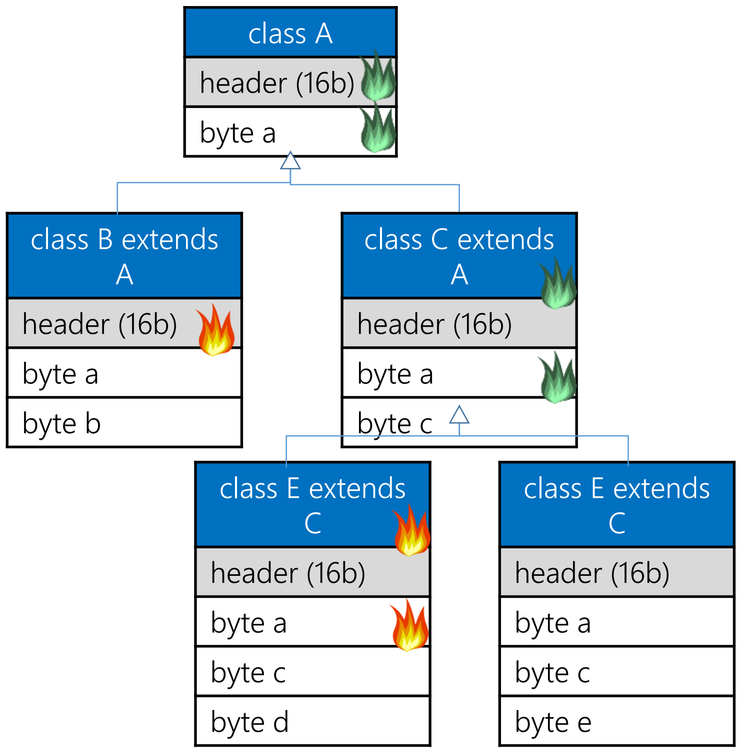
Padding - Inheritance



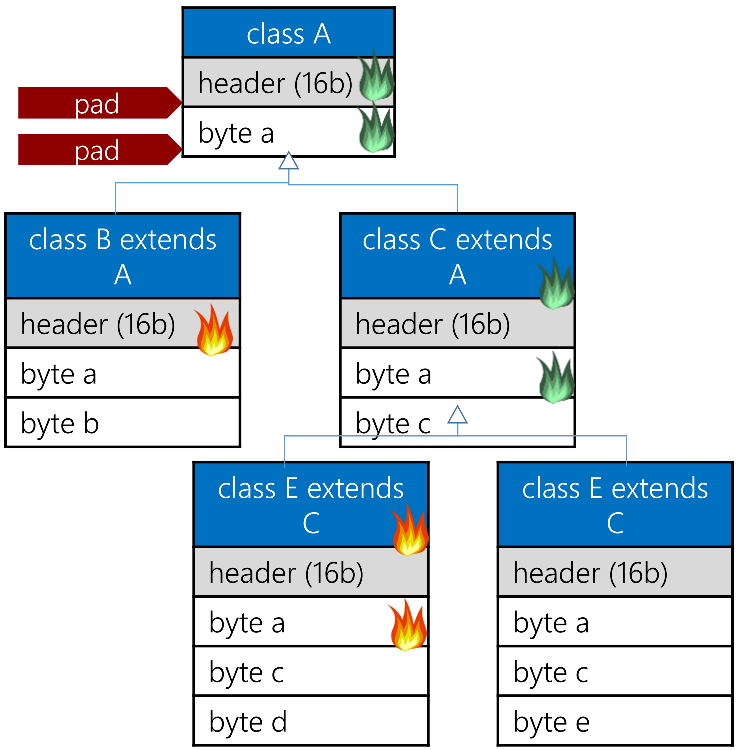
Padding - Inheritance



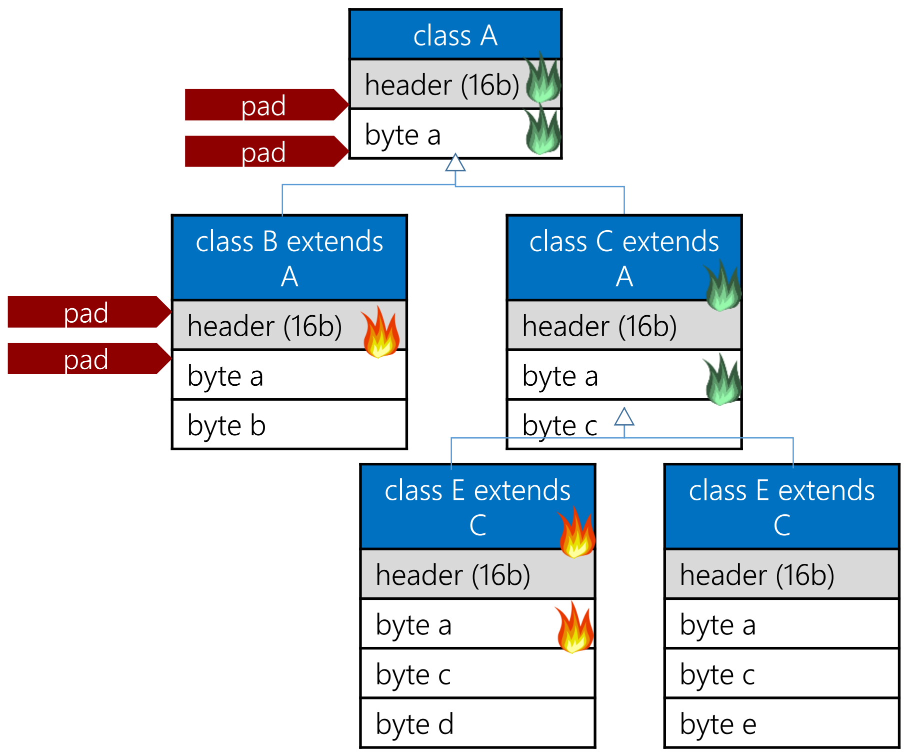
Padding - Inheritance



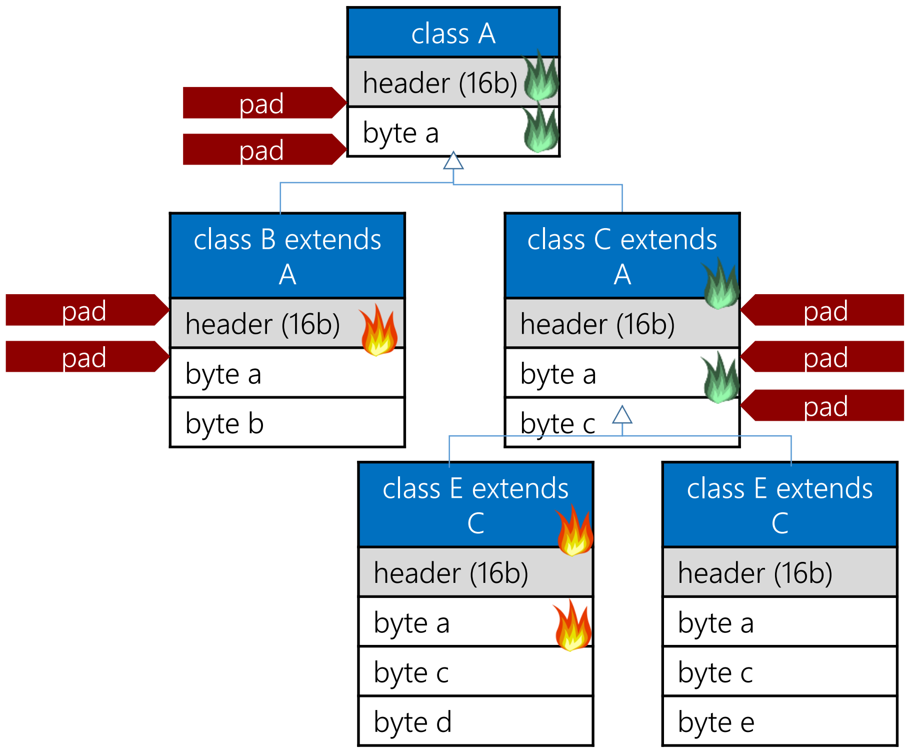
Padding - Inheritance



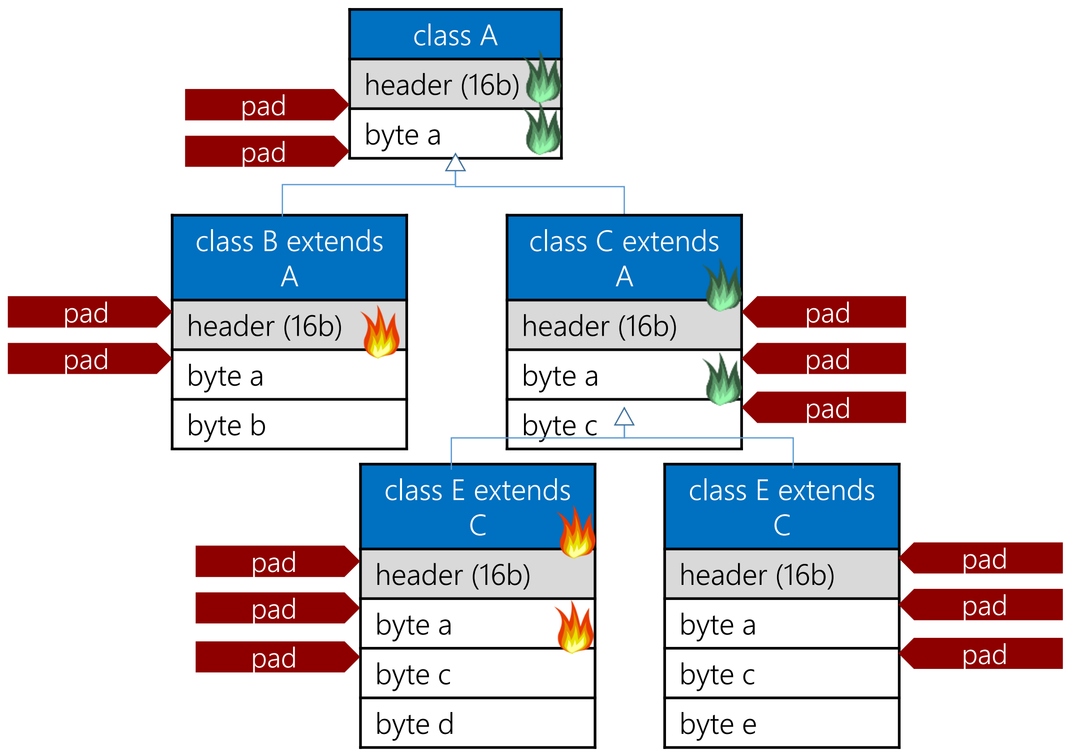
Padding - Inheritance



Padding - Inheritance

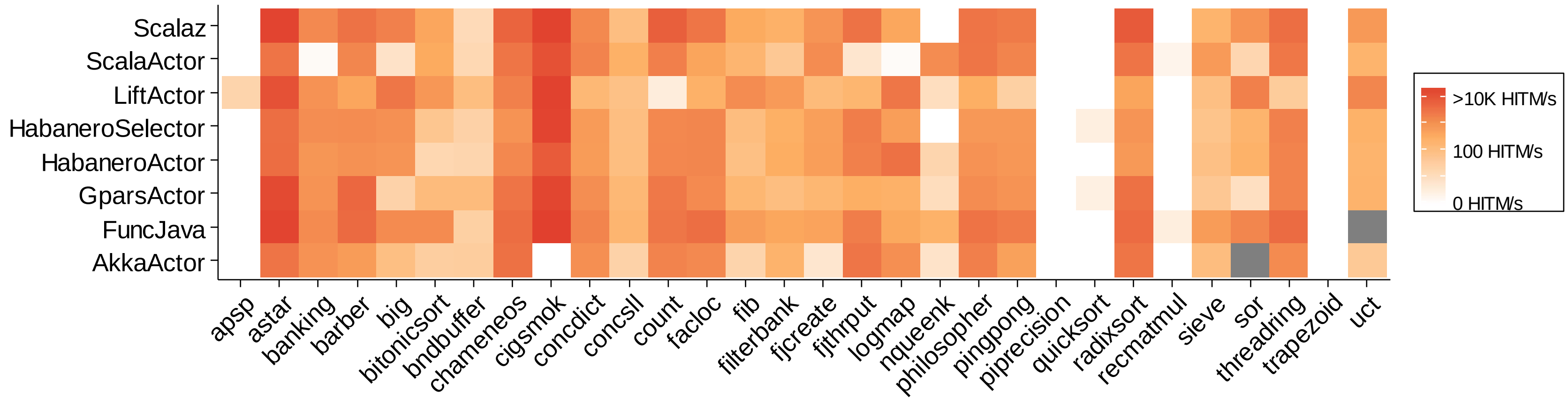


Padding - Inheritance



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