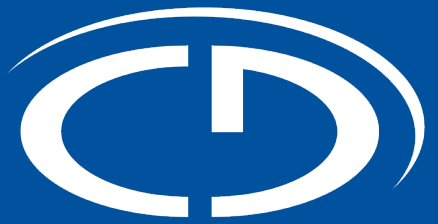


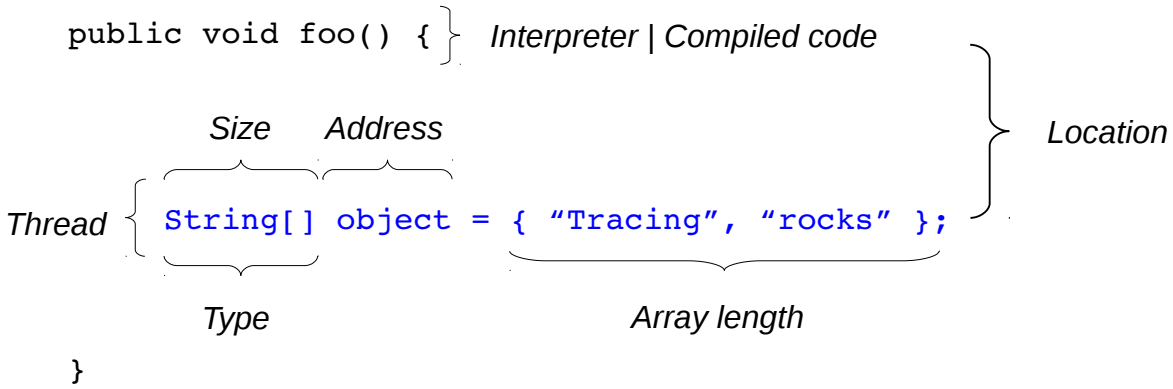
Accurate and Efficient Object Tracing for Java Applications

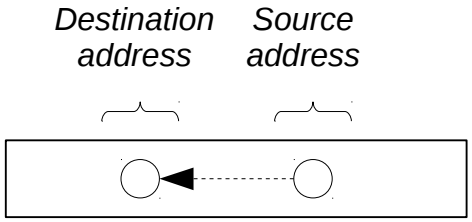
Philipp Lengauer
Verena Bitto

2015-02-02

What if ...

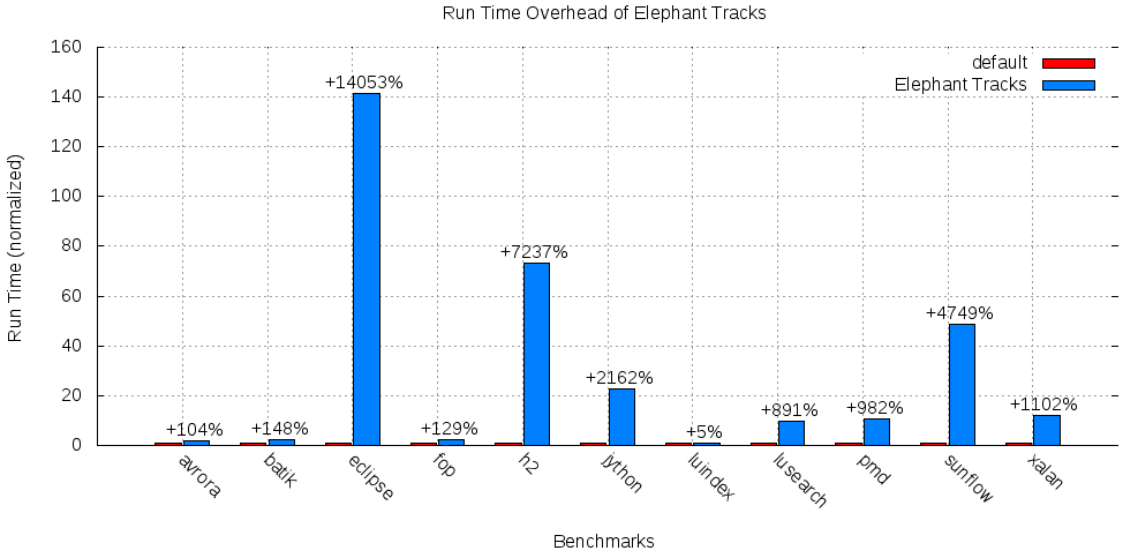
… we would know all there is to know about every object?

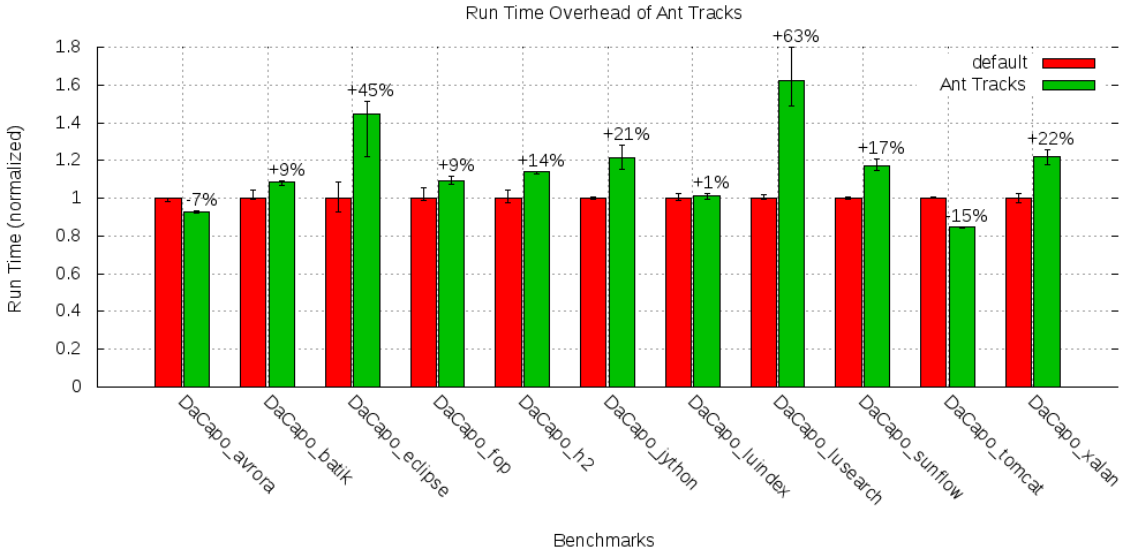




… then we could reproduce the entire heap for every point in time and do offline analysis!

State of the Art – An Unfair Comparison



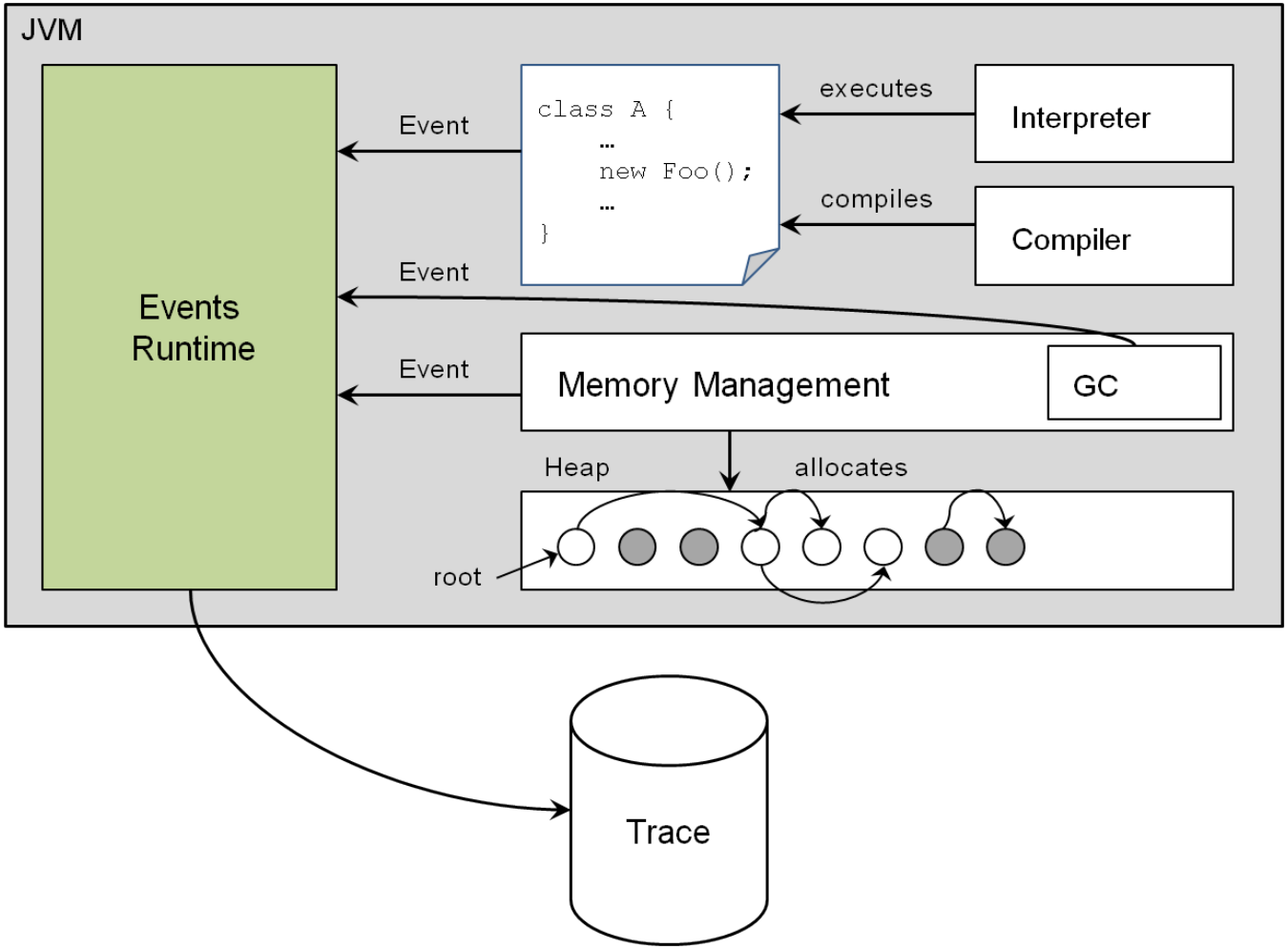


Instrumentation based
→ changes application behavior
→ introduces large overhead
→ unable to reconstruct heap in detail

avg(overhead) = 1054 %

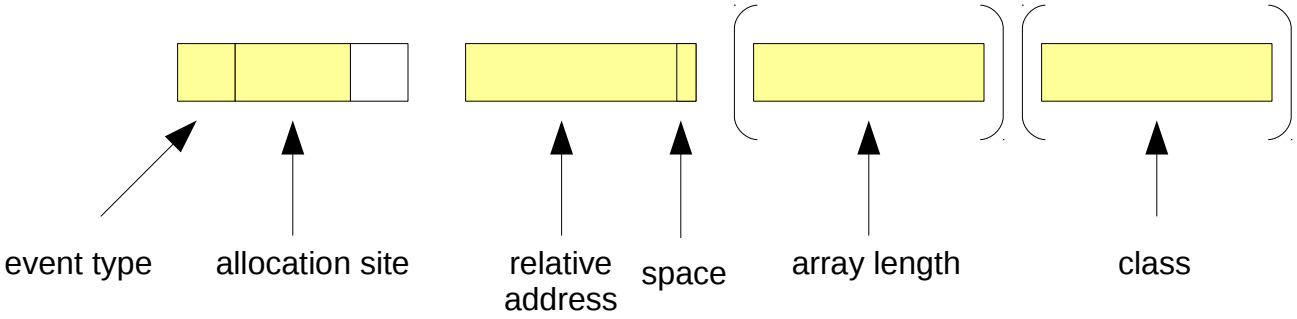
avg(overhead) = 14 % (7 %)

Approach


JVM
Events Runtime
Event
class A {
…
new Foo();
…
}
executes
Interpreter
compiles
Compiler
Event
Event
Memory Management
GC
Heap
allocates
root
Trace

Allocation Events

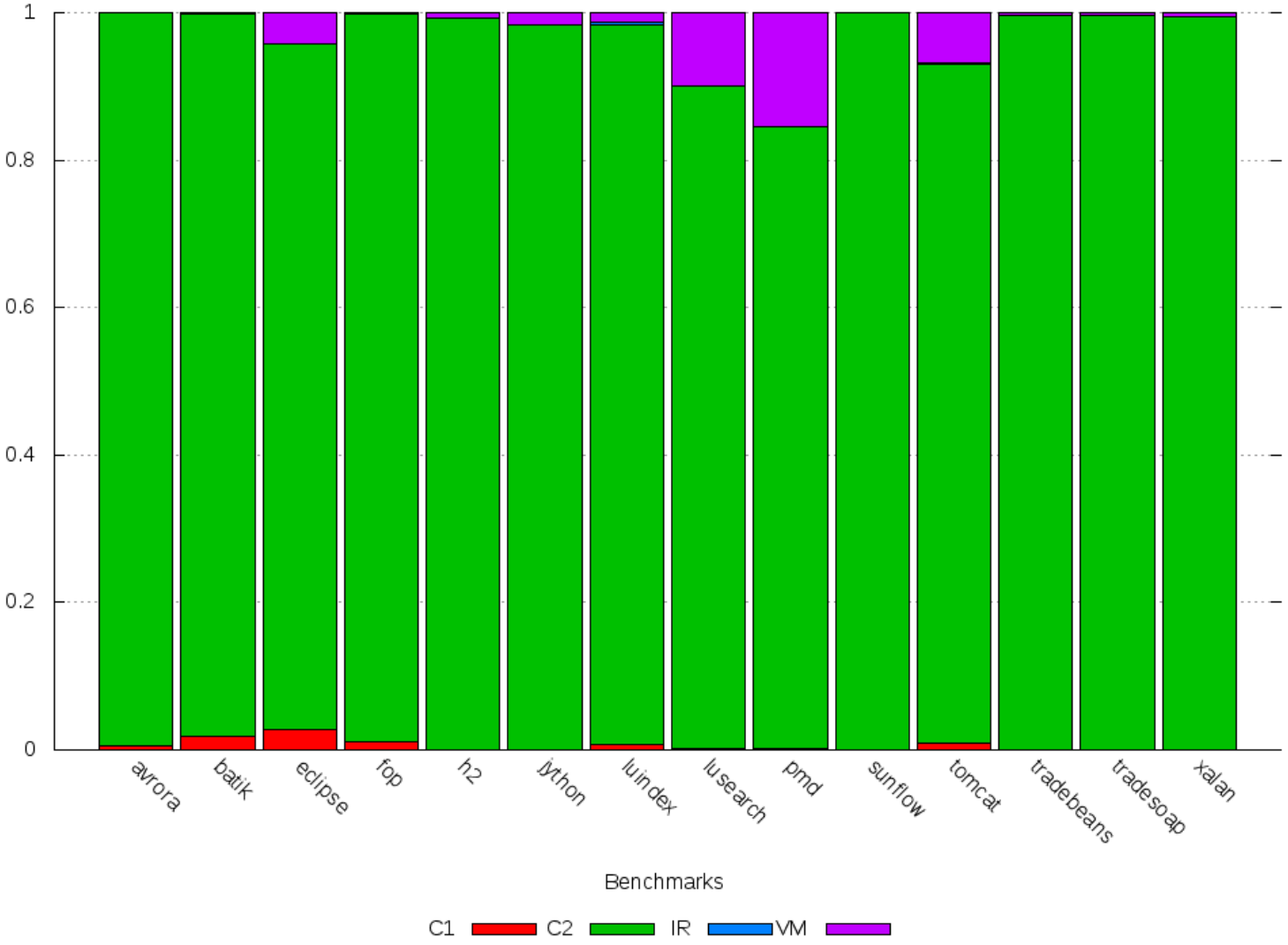
Slow allocation event



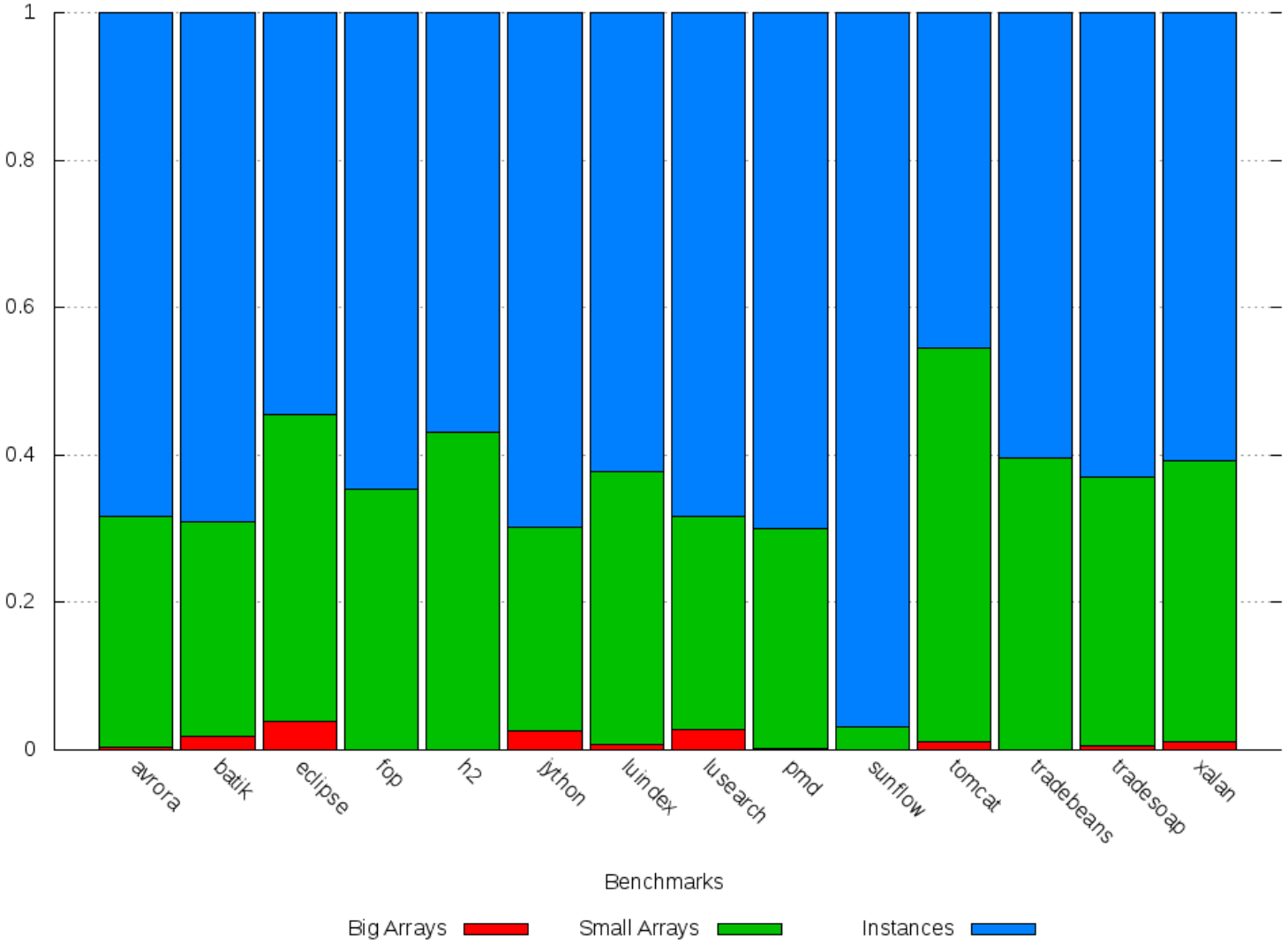
mode → event type
class → allocation site
size → class + length

→ 8 – 16 bytes per allocation

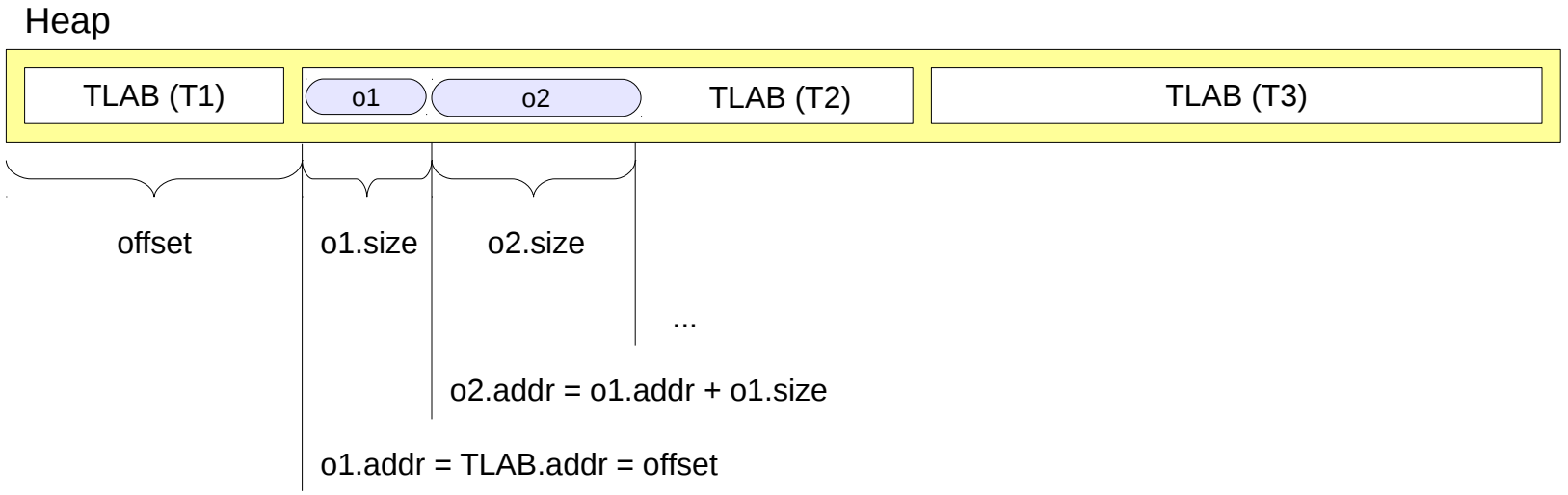
Allocating Subsystems



Object Types



Allocations in JVMs

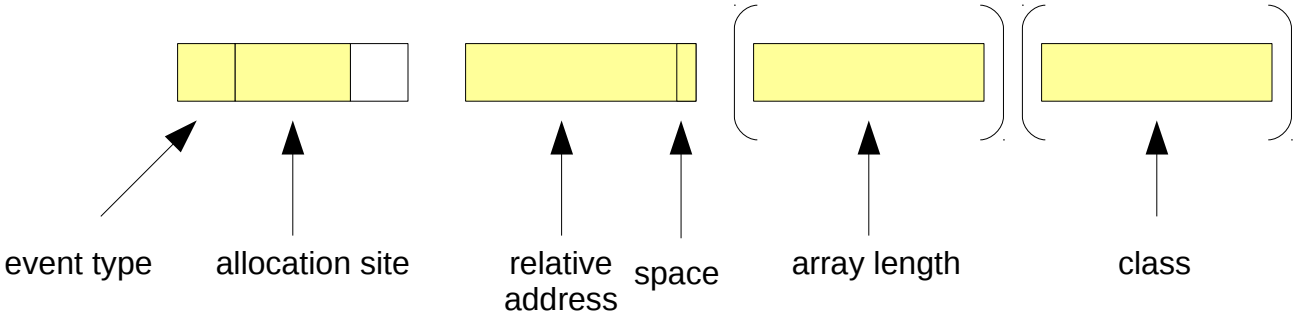


$$addr(o_n) = \begin{cases} addr(TLAB(o_n)) & \text{if } n = 1 \\ addr(o_{n-1}) + size(o_{n-1}) & \text{else} \end{cases}$$

Addresses of objects that are allocated into a TLAB are computable offline!

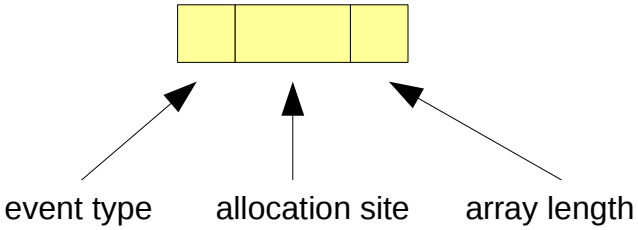
Allocation Events Revisited

Slow allocation event



mode → event type
class → allocation site
size → class + length

Fast allocation event



address → previous events + TLAB information

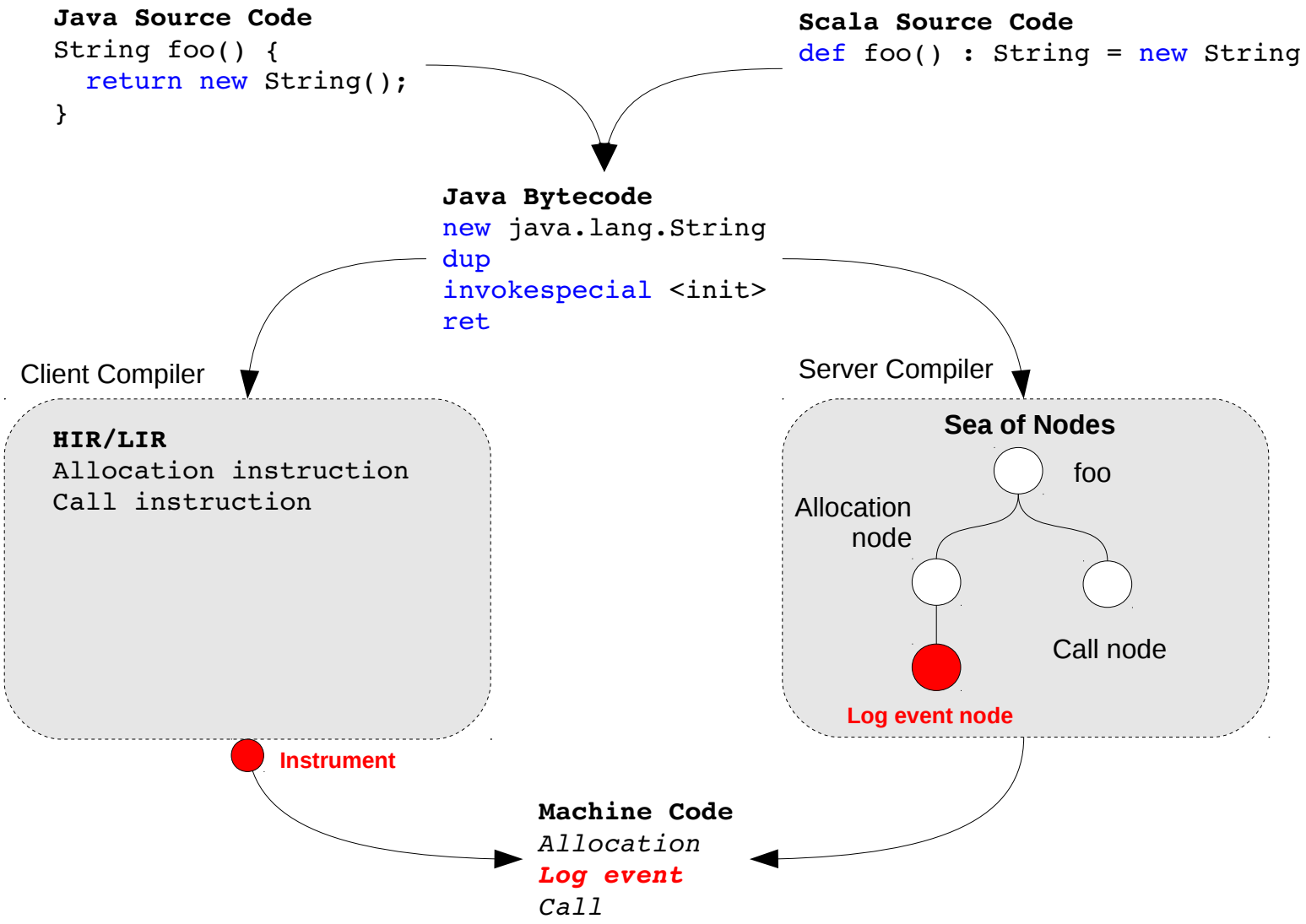
→ 4 bytes per allocation
→ **computable at compile-time (JIT)**

Firing a Fast Allocation Event

is as easy as ...

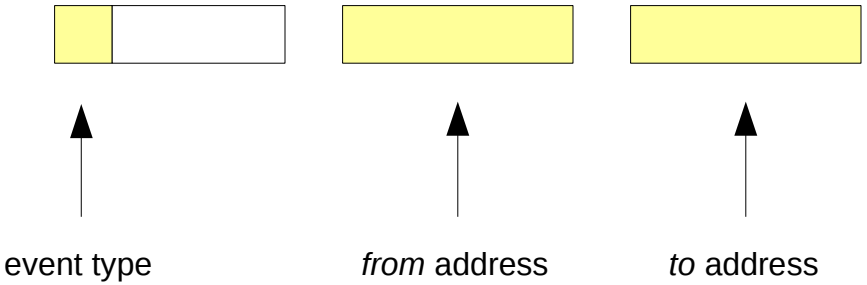
```
const int event = 0xABCDEF00;

if(buffer->top + 1 <= buffer->end) {
    *(buffer->top++) = event;
} else {
    fire_event_slow(event);
}
```

Instrumenting Allocation Sites



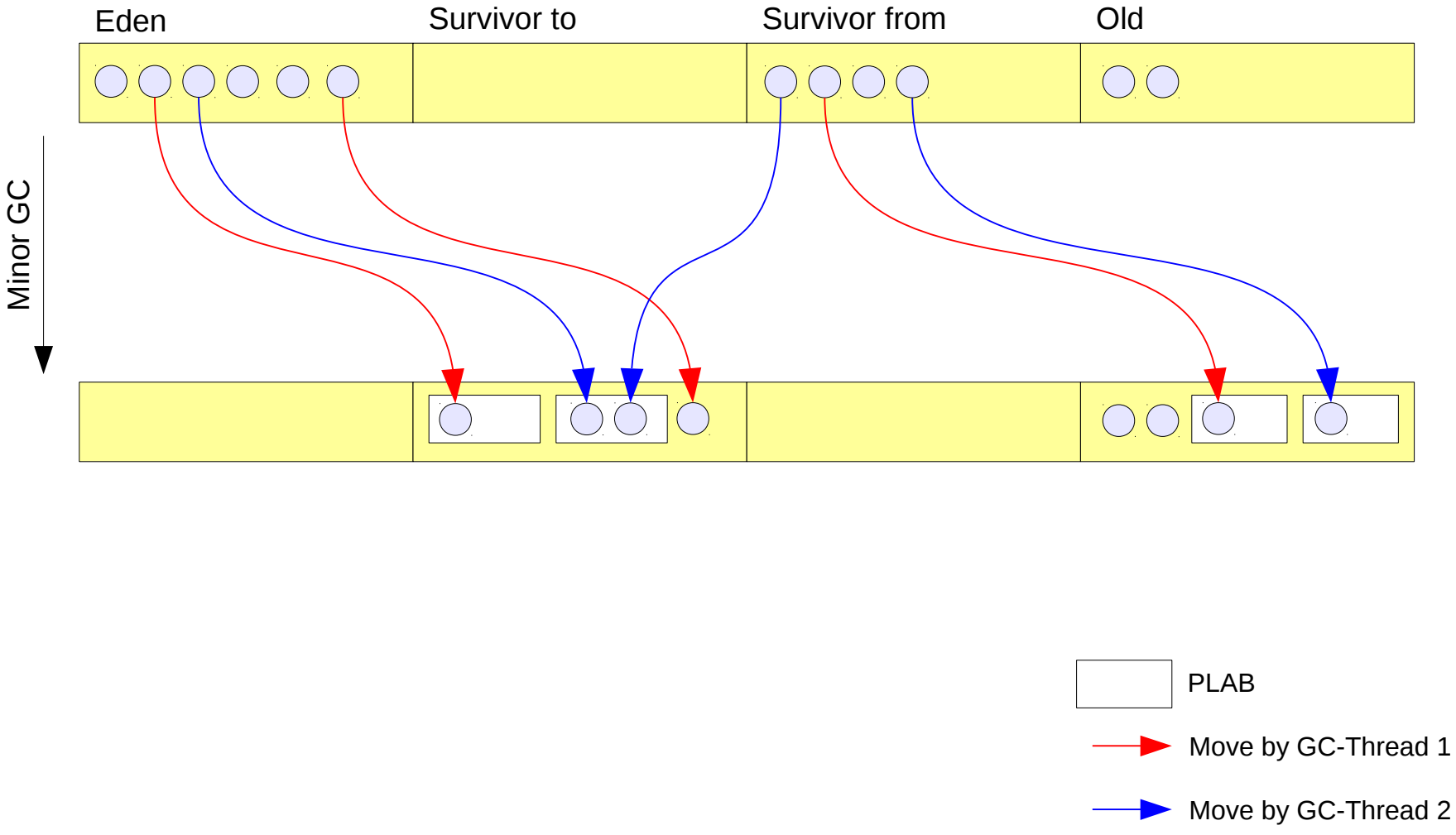
Move Events

Slow move event



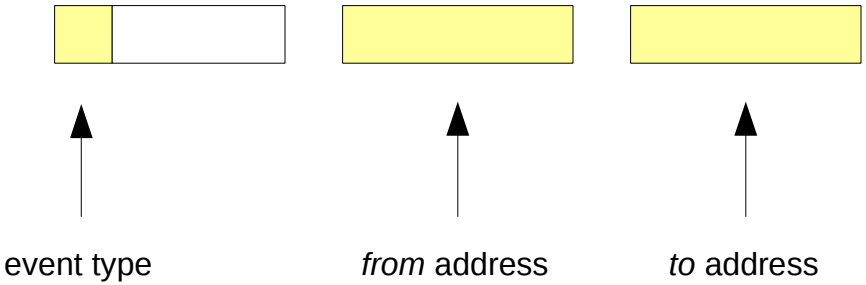
If there is no move event for an object in a collected space, it has been deallocated.

Minor GCs

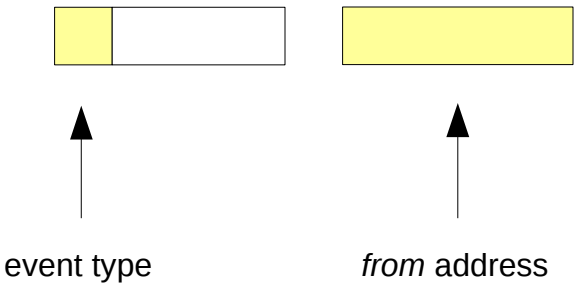

Eden
Survivor to
Survivor from
Old
Minor GC
PLAB
Move by GC-Thread 1
Move by GC-Thread 2

Move Events Revisited (for Minor GCs)

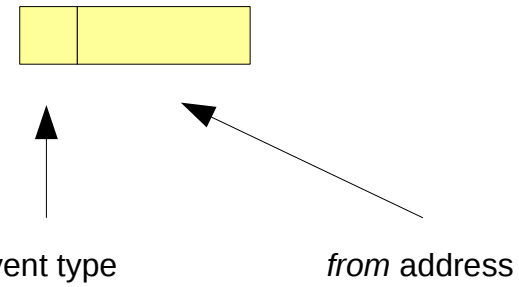
Slow move event



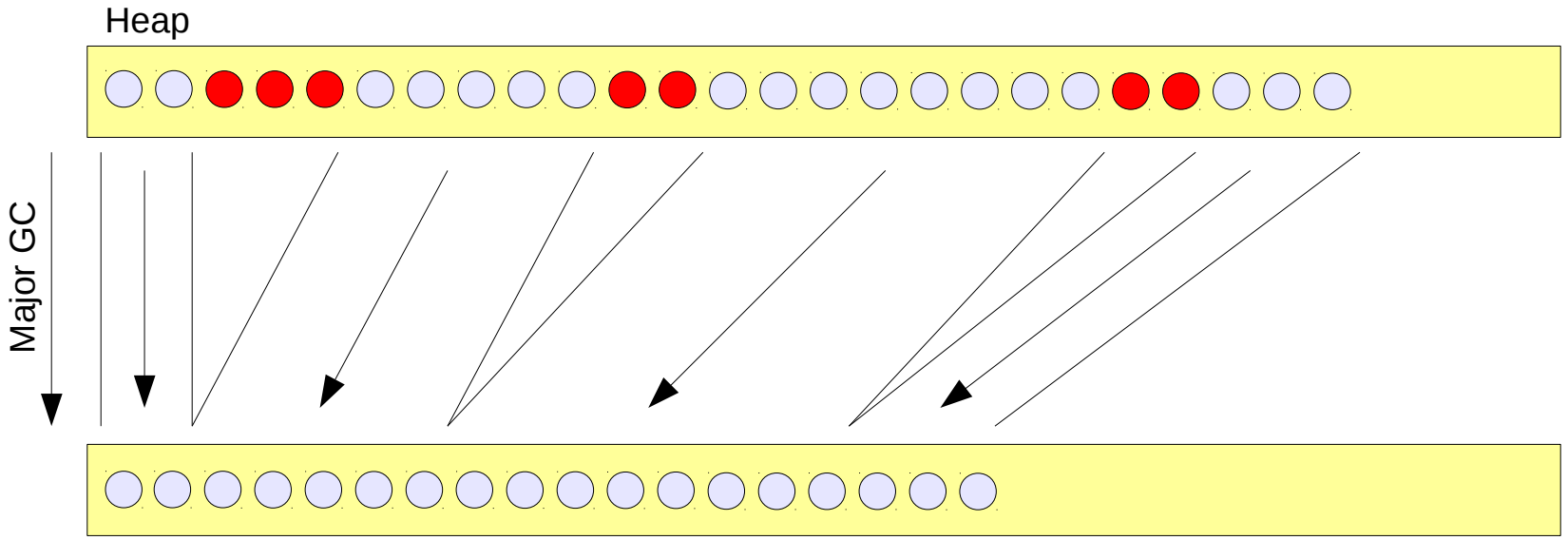
Fast move event



Fast narrow move event



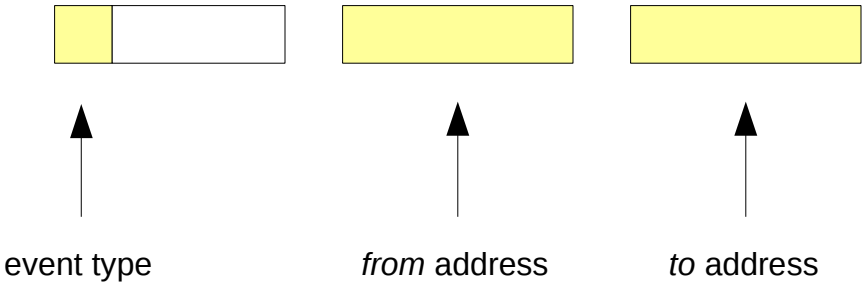
Major GCs



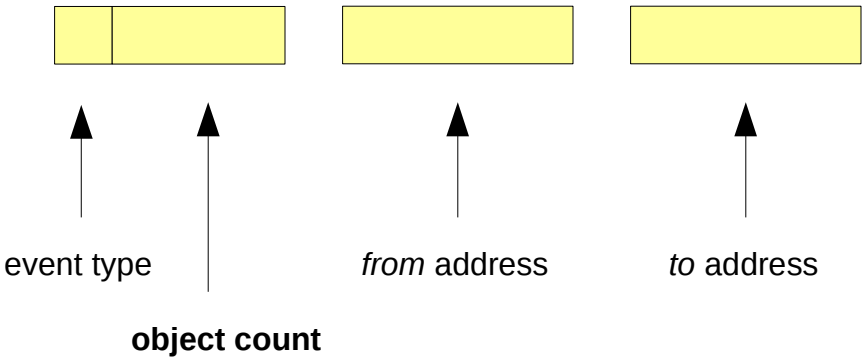
Claim: objects live and die in groups due to their sequential allocation

Move Events Revisited (for Major GCs)

Slow move event

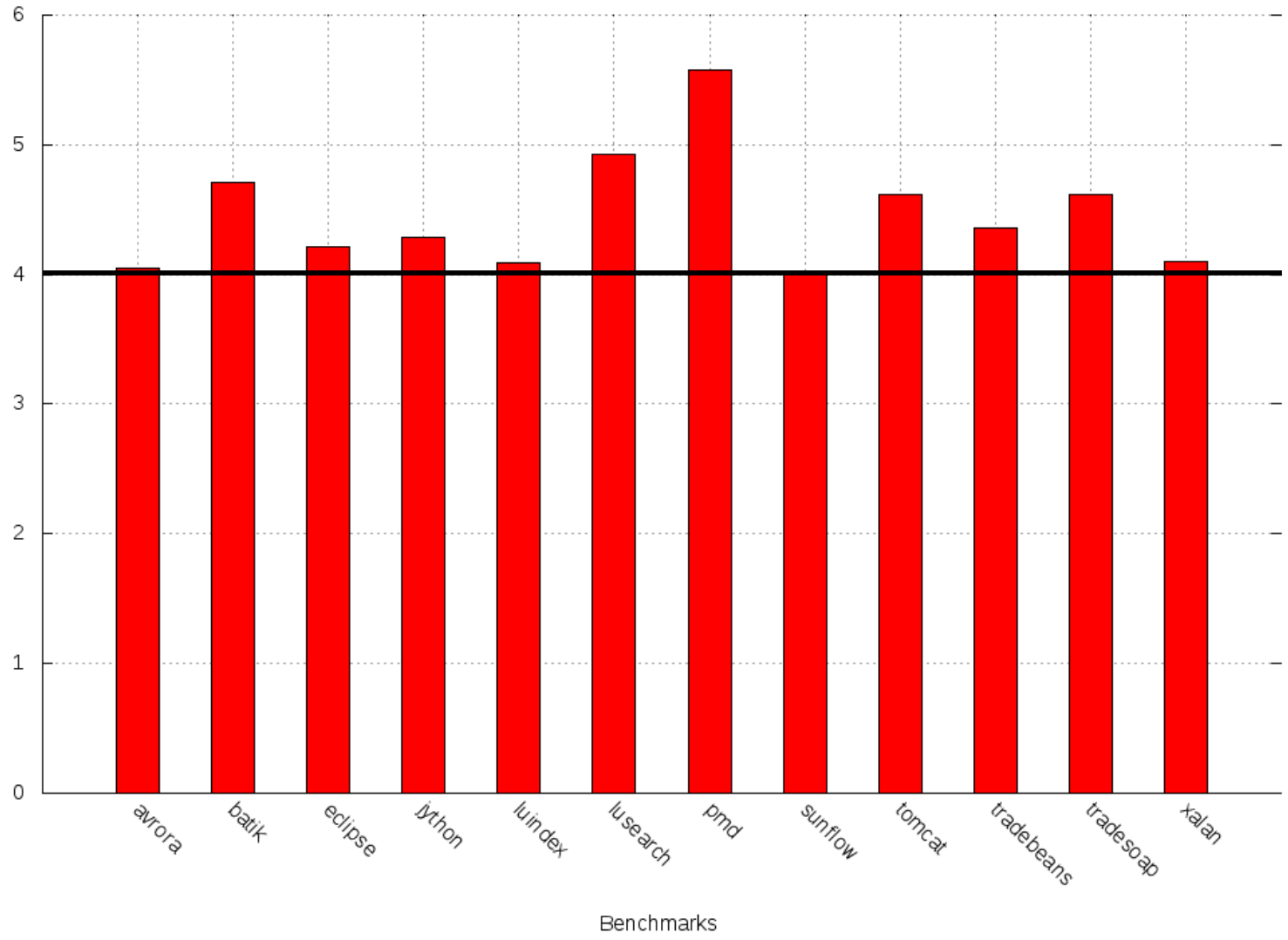


Region move event

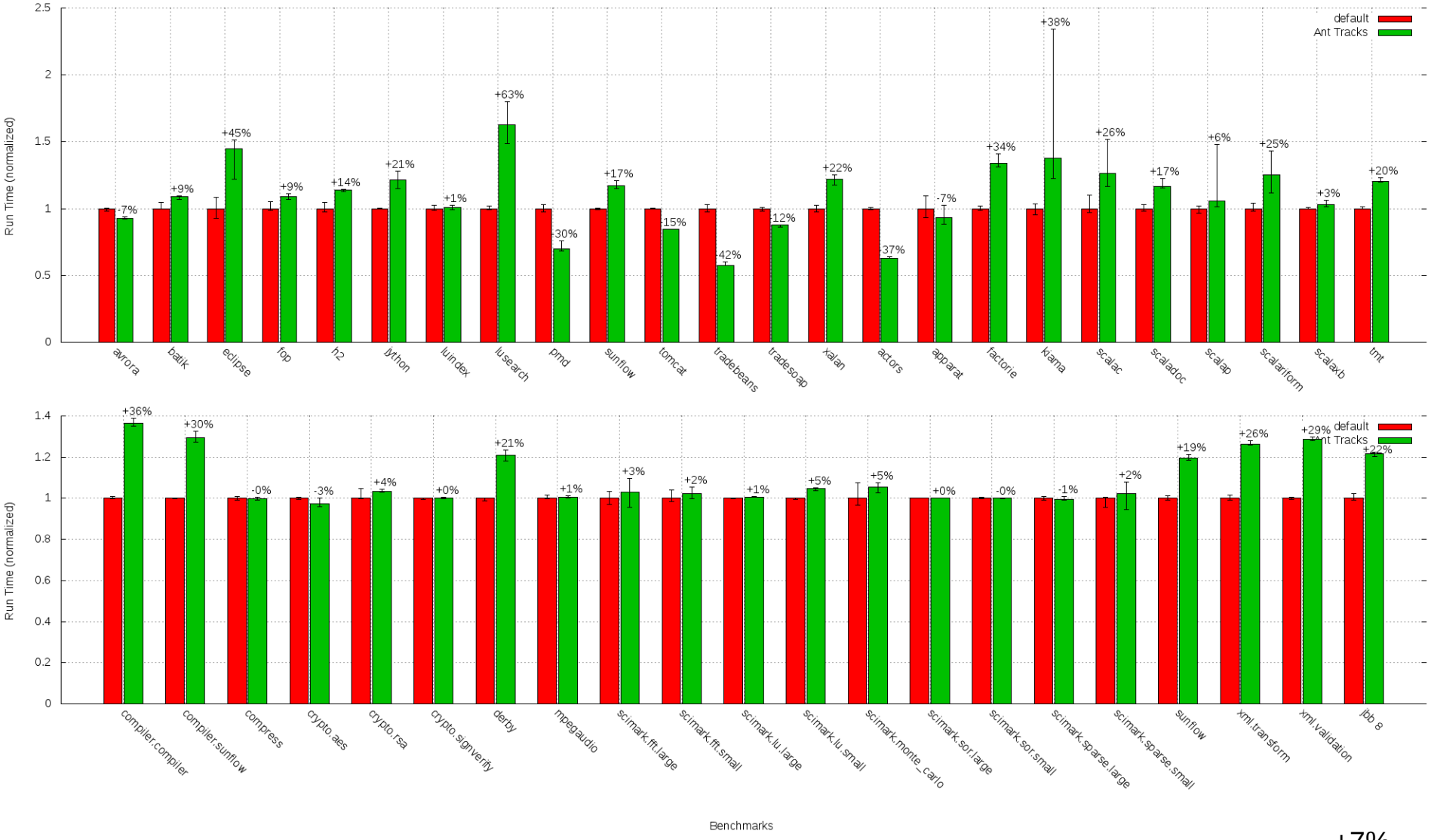


~ 312 objects per event (3.65Kb -> 12b)

Average Event Size [bytes]

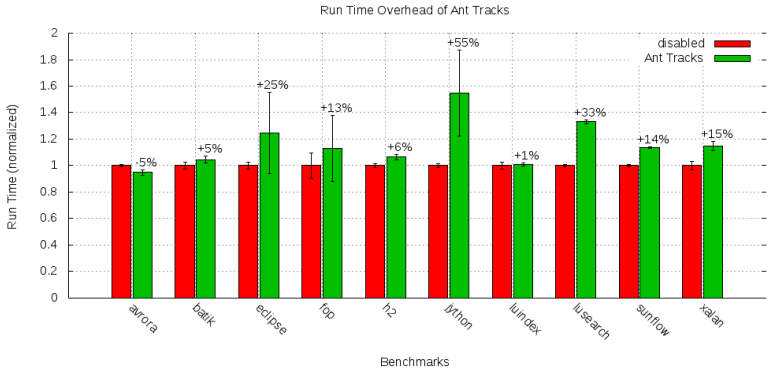


Overhead (DaCapo, DaCapo Scala, SPECjvm, SPECjbb)



+7%

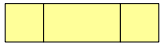
Conclusion



Very low overhead (7 %)

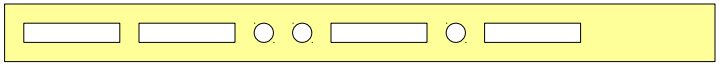
Slow event

Fast event



Compact event formats, ...

Heap

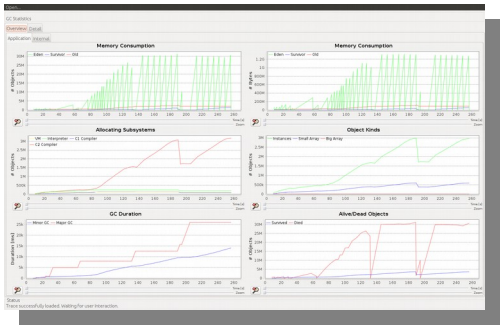


Trace

alloc
alloc
alloc
...

$$addr(o_n) = \begin{cases} addr(TLAB(o_n)) & \text{if } n = 1 \\ addr(o_{n-1}) + size(o_{n-1}) & \text{else} \end{cases}$$

Analyze trace and rebuild heap offline



Proof of concept – Ant Tracks

Q&A

?