

Java for Real-Time Systems

Brian Neely

What is a real-time system?

- Reactive system
 - Bounded response time to events
- Soft real-time:
 - Jitter is OK
- Hard real-time:
 - Jitter == failure
- Throughput \neq Performance
 - We care about worst-case latency

Why should we care?

- Real-time systems are everywhere!
 - Planes, Trains and Automobiles
- Increasing complexity
 - Network connectivity, GUIs
- Shorter design cycle
- C is (almost) all we have



Java is Good and Bad

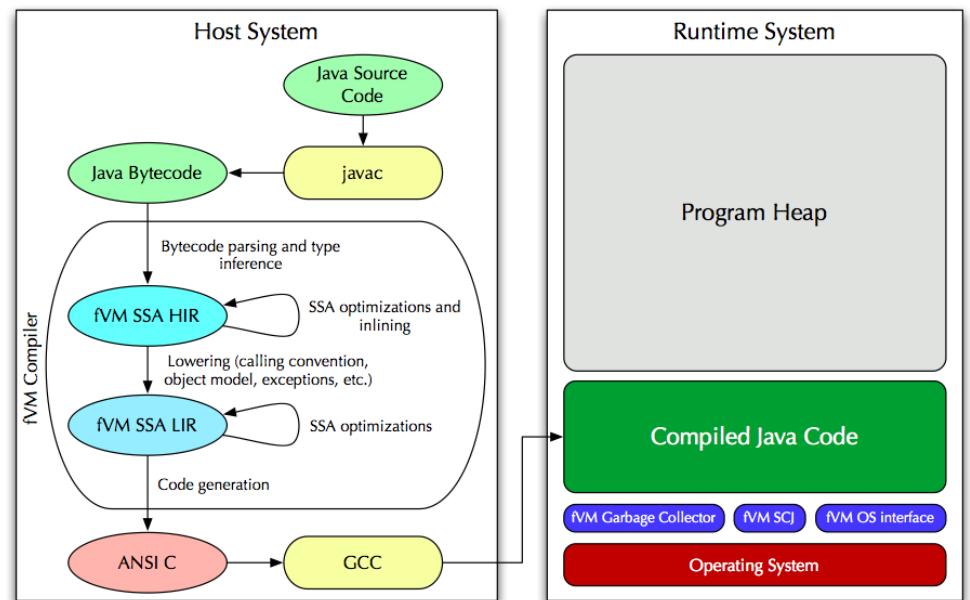
- No guarantees about timeliness
 - Interpretation
 - On-the-fly compilation
 - Dynamic dispatch
 - Stop-the-world garbage collection
- But affords great productivity
 - Modularity
 - Expressiveness
 - Safety

How do we make it work?

- A RT Java system must:
 - Support all or defined subset of the Java language (obvious)
 - Guarantee timeliness of RT operations
 - Provide deterministic, non-interfering garbage collection

The Fiji VM

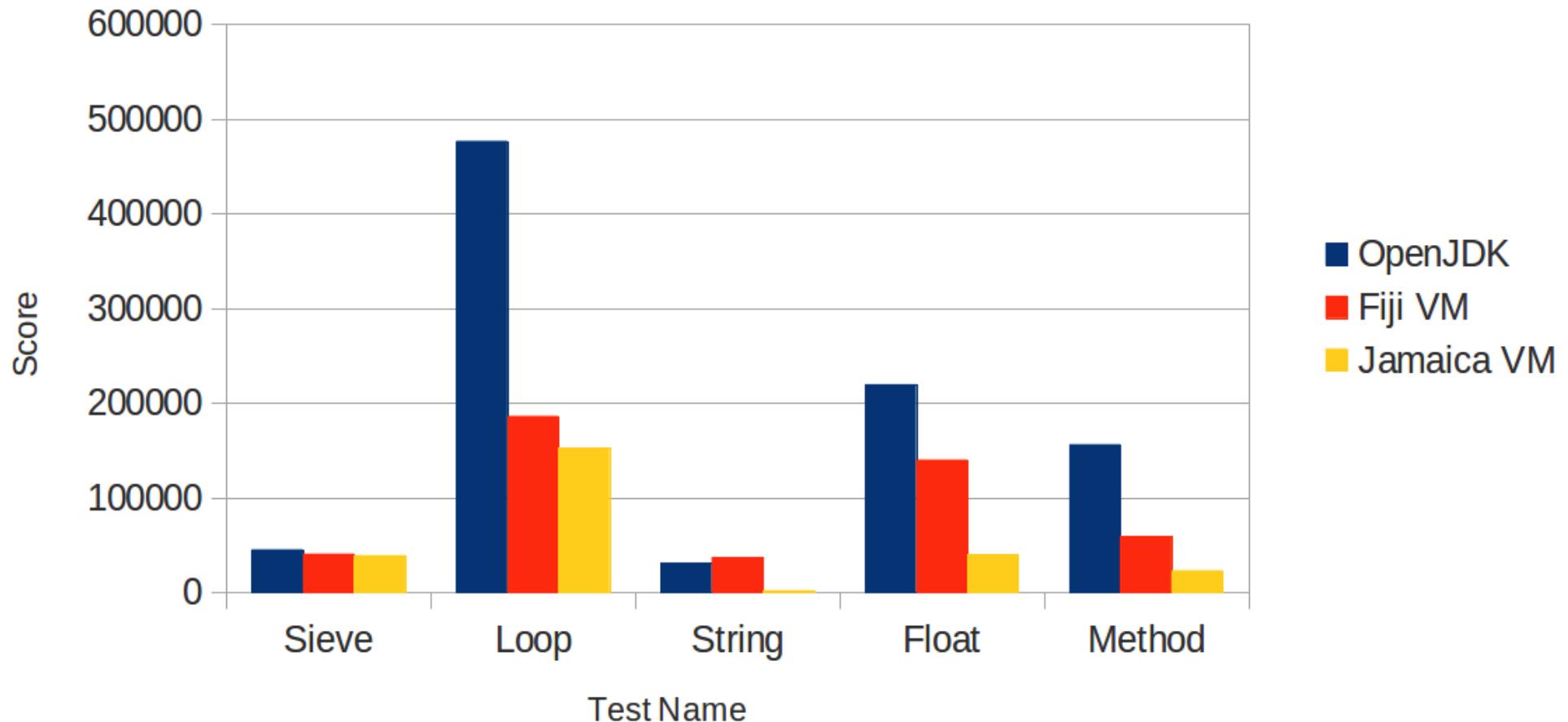
- A real-time Java VM from Purdue / Fiji Systems, Inc.
- Minimize timing variability:
 - Compile/optimize ahead-of-time
- Don't stop-the-world:
 - Stop for root scanning



Benchmarking Fiji

CaffeineMark

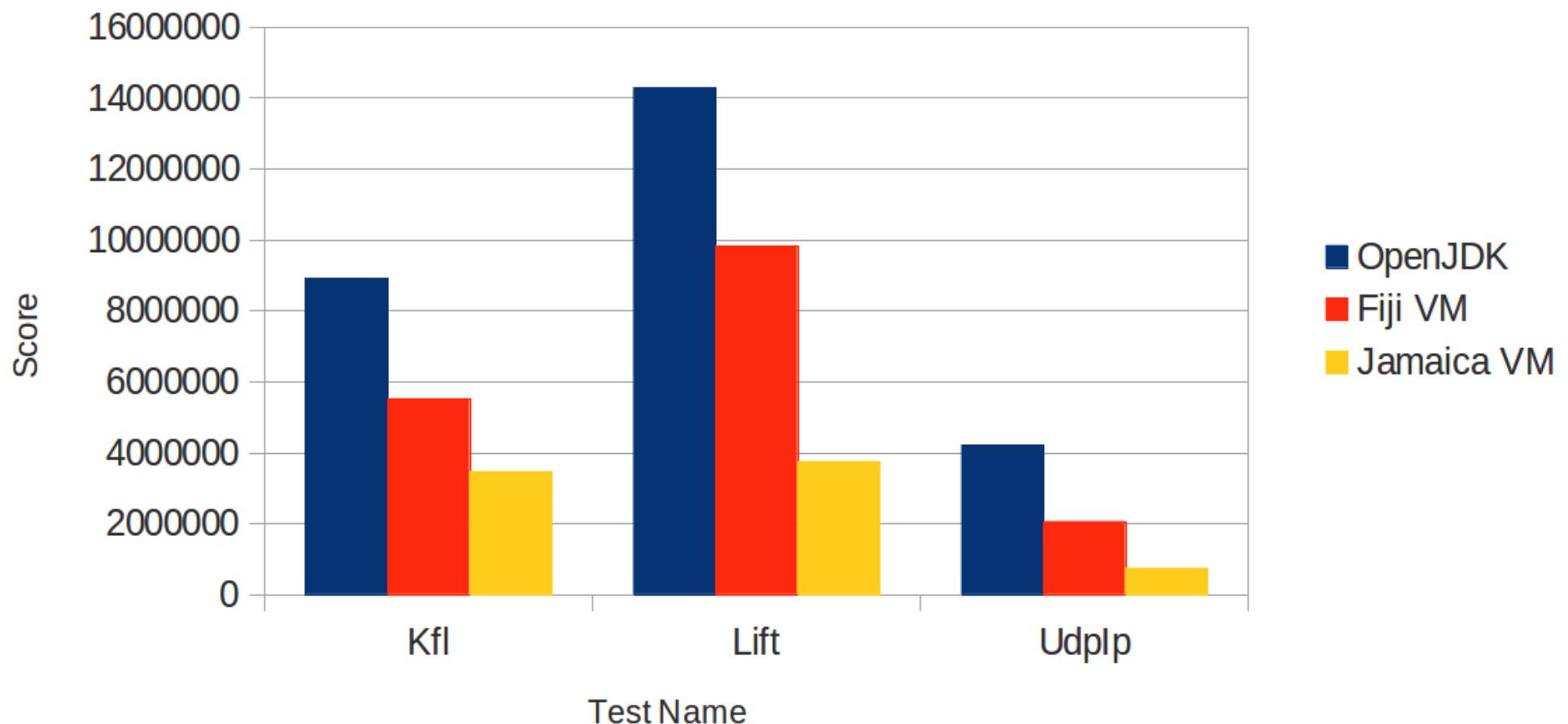
CaffeineMark Benchmark Results



Benchmarking Fiji

JemBench

JemBench Application Benchmark Results



Benchmarks are Misleading

- Capture the steady-state
 - We care about worst-case
- Measure throughput (iterations/second)
 - Again, we care about worst-case
- Execute same thing over and over
 - Is this realistic for a real-time system?

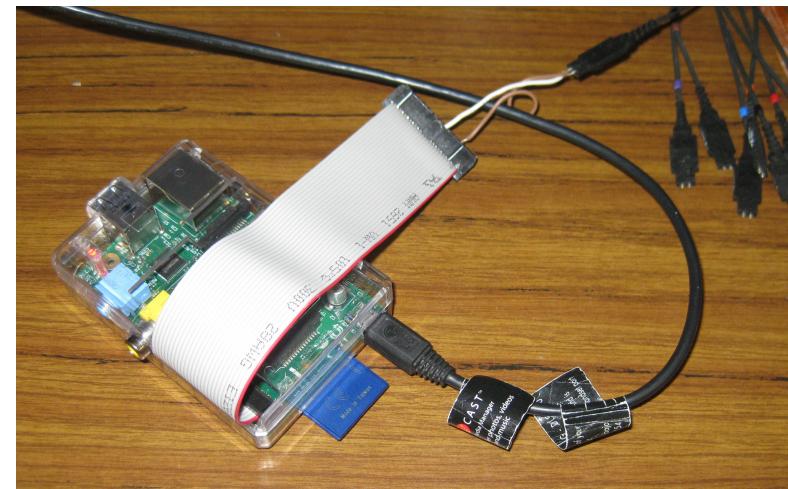
A Simple Real-time Benchmark

- Generate a 500 Hz pulse in software
- Measure the actual period of the pulse
- Why?
 - Simple, but measures timeliness of *every* event
 - Used for servo motors, PWM motors, etc.

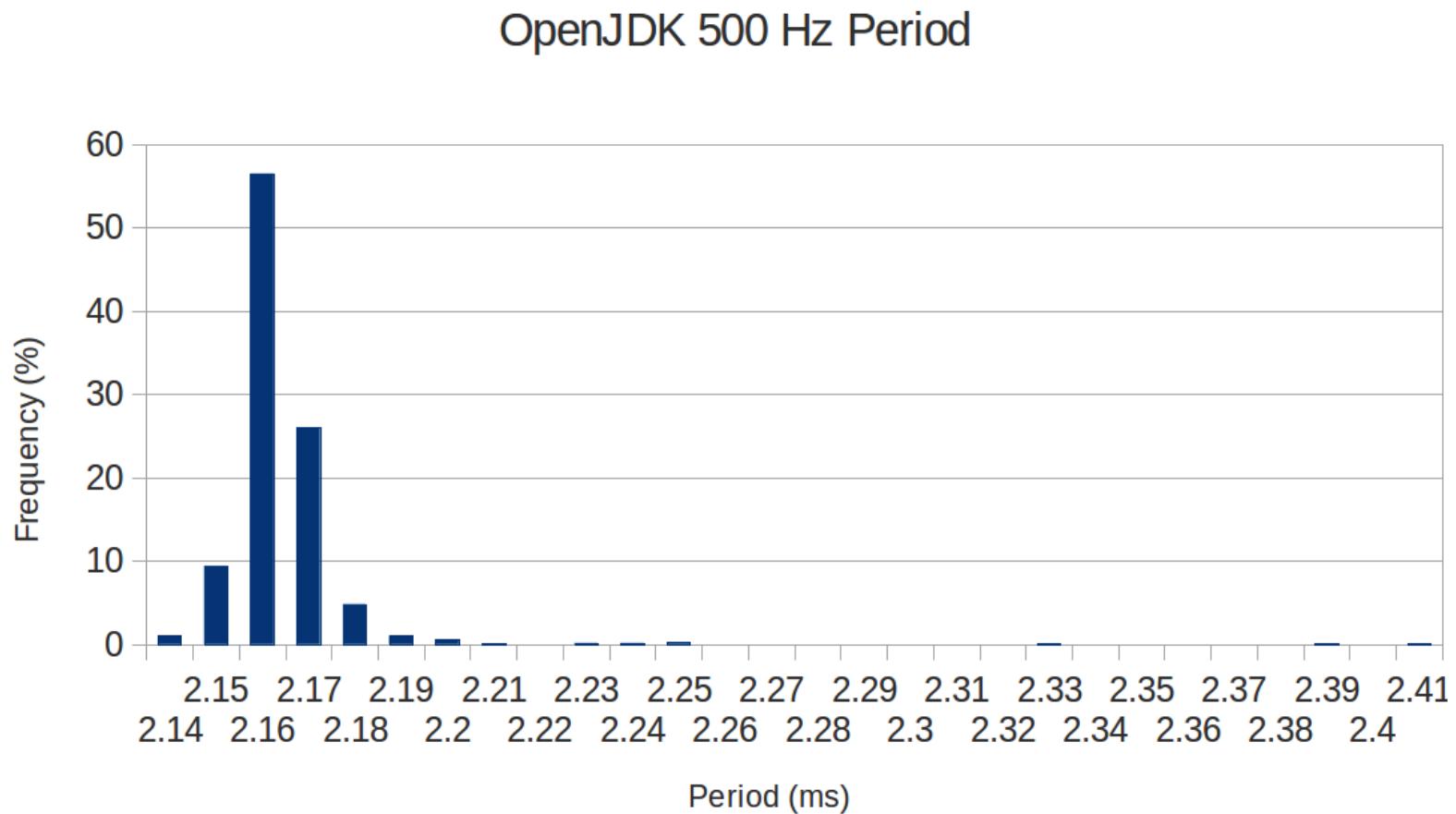


A Simple Real-time Benchmark

- Compare OpenJDK and Fiji VM for three scenarios:
 - 500 Hz Pulse, no system stress
 - 500 Hz Pulse, GC stress
 - 500 Hz Pulse, dynamic dispatch stress
- Equipment: Raspberry Pi, TLA 700 Logic Analyzer



500 Hz, No Stress Test



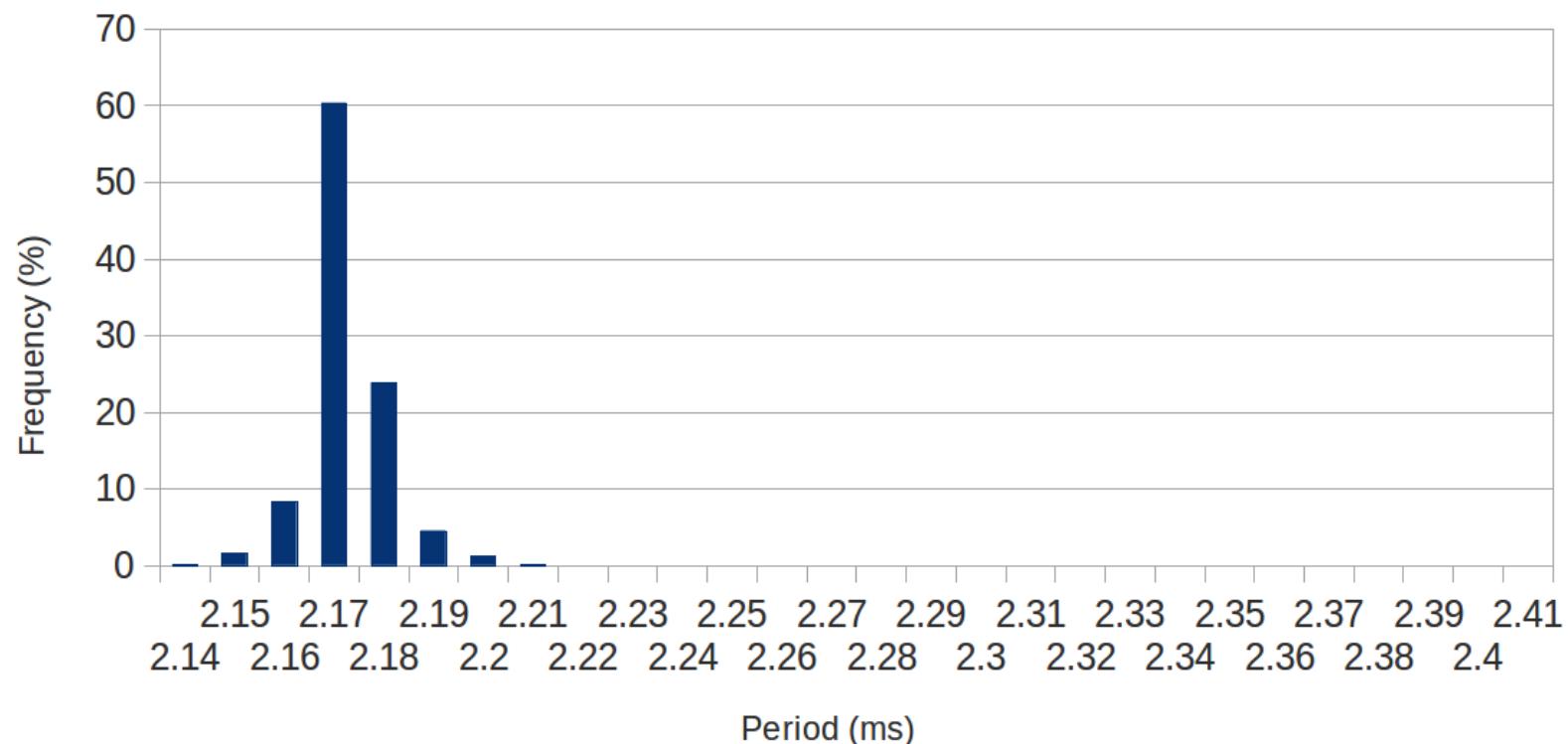
Avg: 2.17ms

Max/Min: 2.41 / 2.15ms

Stddev: 0.014

500 Hz, No Stress Test

Fiji VM 500 Hz Period



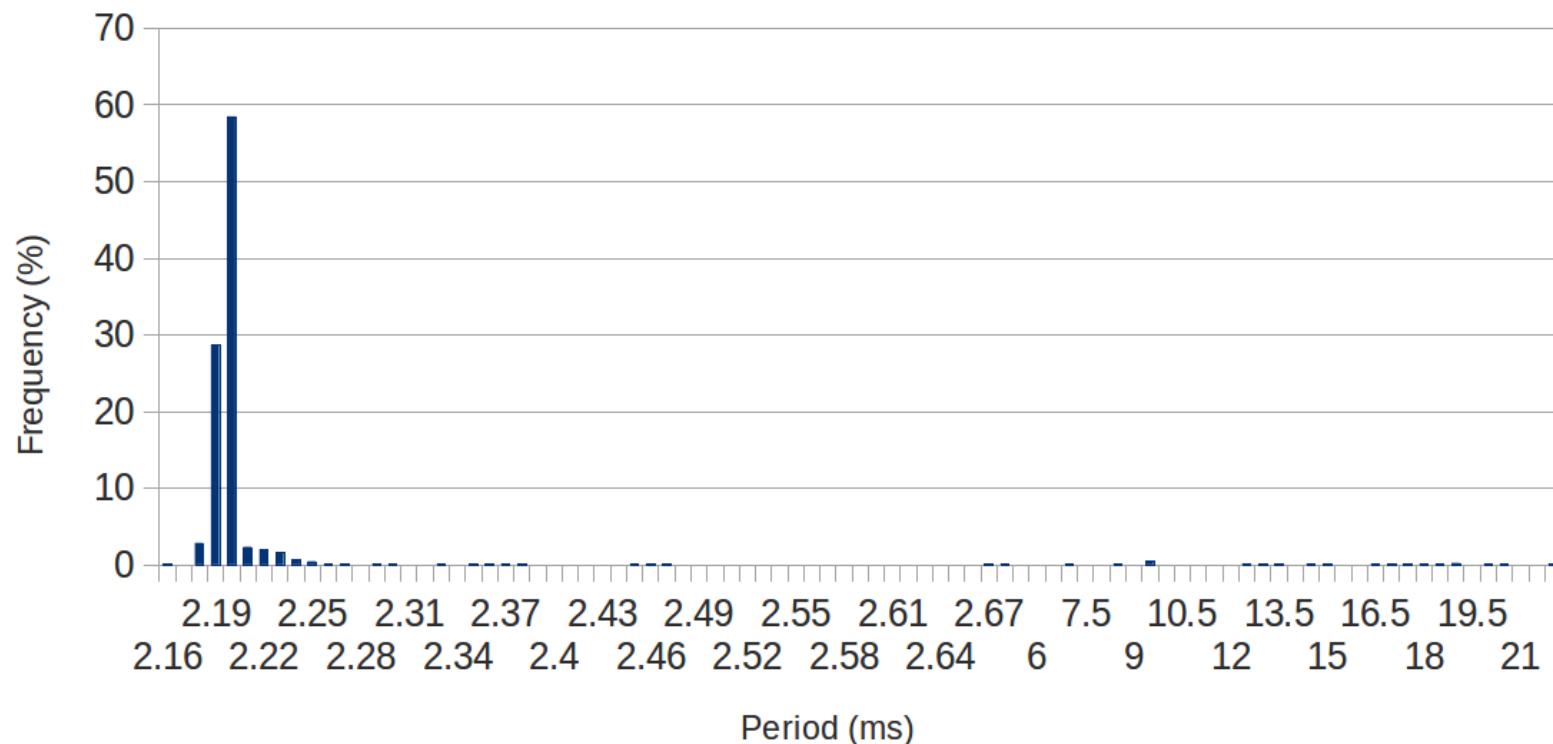
Avg: 2.18ms

Max/Min: 2.21 / 2.15ms

Stddev: 0.008

500 Hz, GC Stress Test

OpenJDK 500 Hz Period, GC Stress



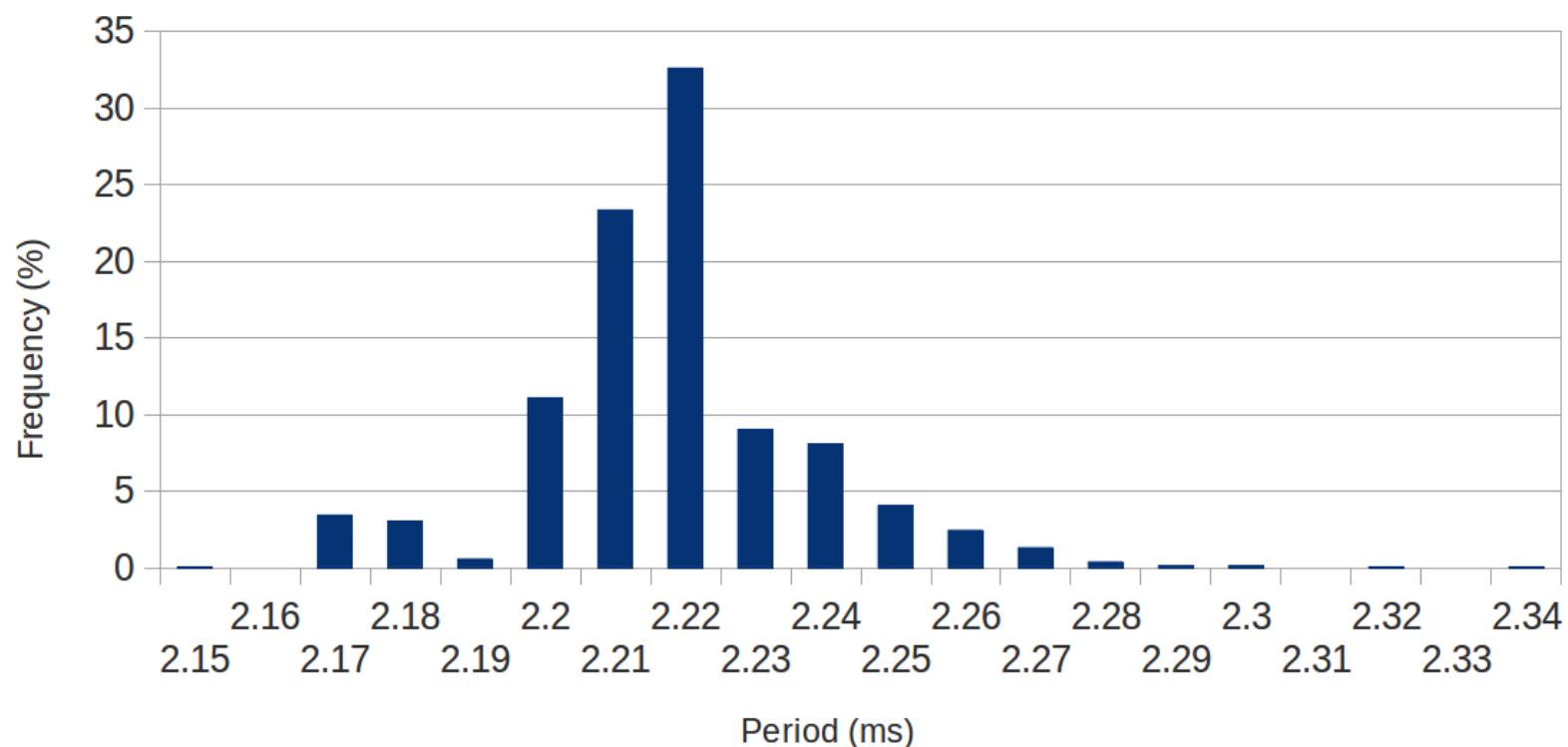
Avg: 2.45ms

Max/Min: 22.35 / 2.17ms

Stddev: 1.83

500 Hz, GC Stress Test

Fiji VM 500 Hz Period, GC Stress



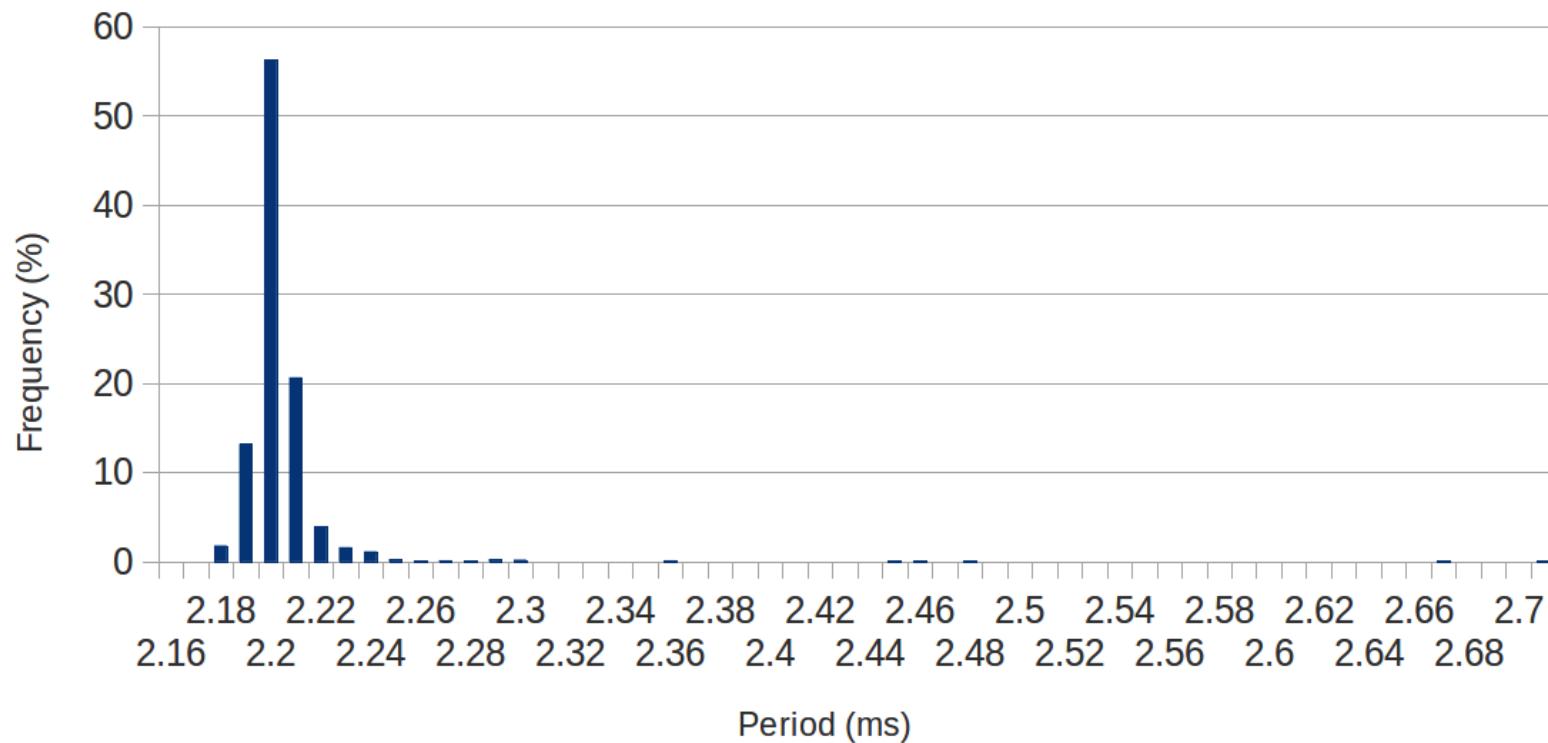
Avg: 2.22ms

Max/Min: 2.35 / 2.16ms

Stddev: 0.02

500 Hz, DD Stress Test

OpenJDK 500 Hz Period, Dyn Dispatch Stress



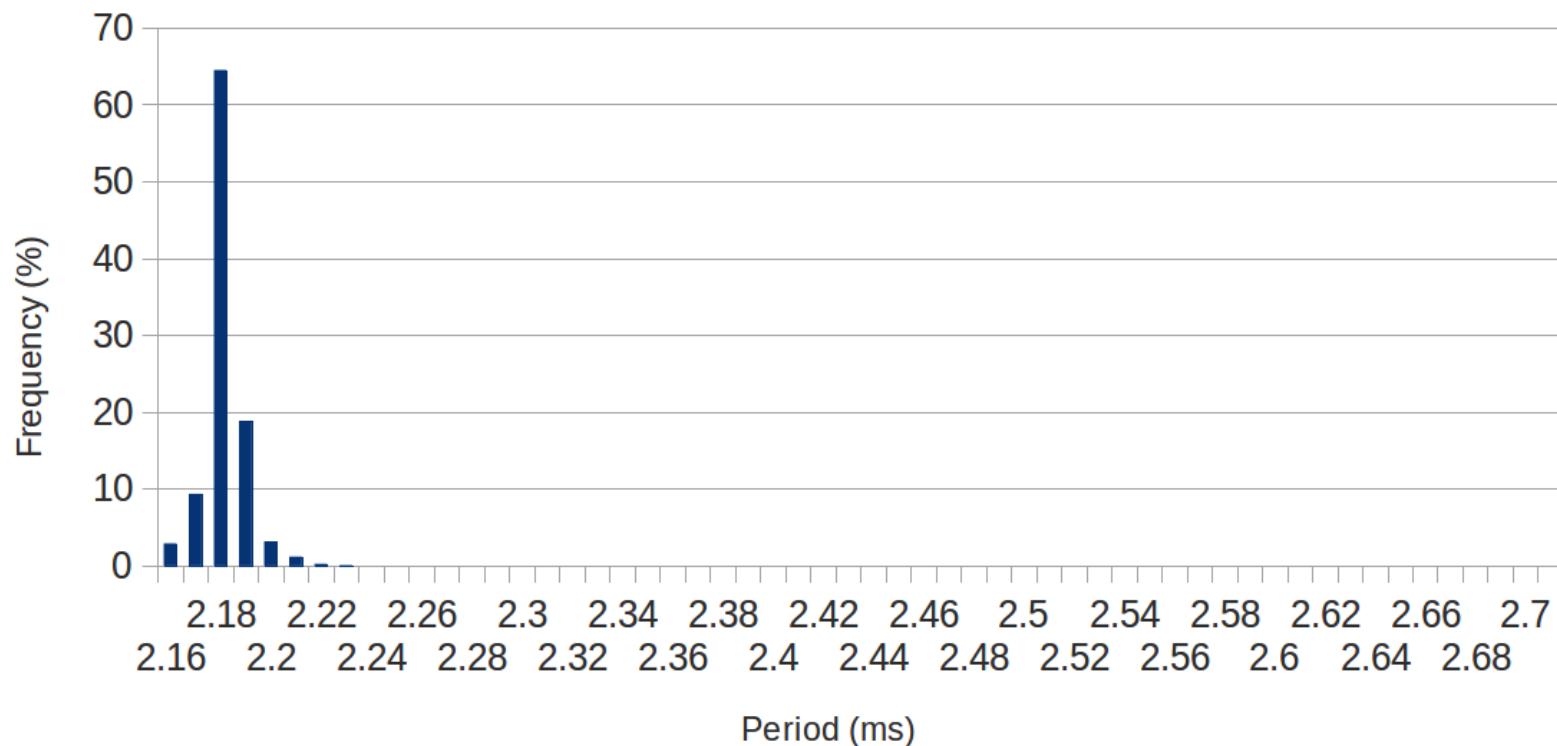
Avg: 2.21ms

Max/Min: 2.71 / 2.18ms

Stddev: 0.03

500 Hz, DD Stress Test

Fiji VM 500 Hz Period, Dyn Dispatch Stress



Avg: 2.19ms

Max/Min: 2.24 / 2.17ms

Stddev: 0.008

Going Beyond 500 Hz

- Can we do 96kHz?
- Why?
 - Sample audio from A/D converter
 - Filter it, transmit over network
 - In a single Java application
- Need a Real-time OS
 - Linux won't cut it:
 - Avg: 102us
 - Max: 186us

(Not) Going Beyond 500 Hz

- RTEMS
 - Fiji allows for Java timer interrupts:

```
final Timer t=new Timer();
t.fireAfter(10 /* ticks */,new Runnable{
    public void run() {
        ...
        t.fireAfter(10, this); // next iteration
    }
});
```

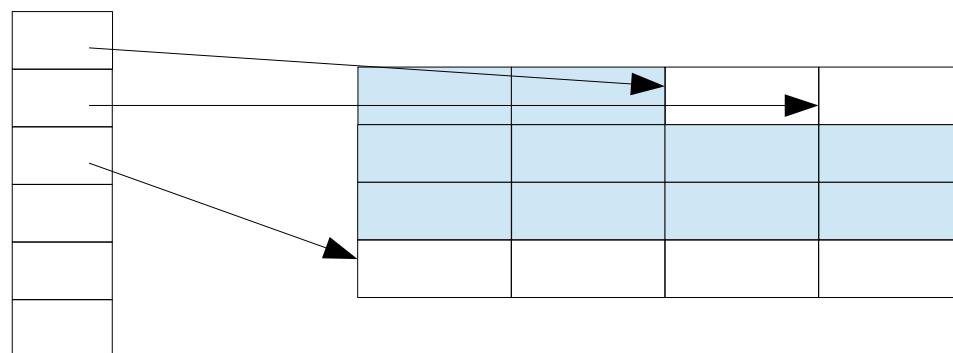
- But
 - Hard real-time support not available in Fiji academic release
 - Had to resort to `thread.sleep()`
 - Fastest sleep: 10ms

Conclusion

- Fiji allows:
 - Integration of hard real-time code into a Java app
 - Running Java on bare-metal
- But:
 - No class loading / reflection
 - Memory footprint: 8.7 MB
 - Doesn't solve all of your timing problems:
 - Bus contention, OS interrupts, etc.
 - Still need to know what you're doing!

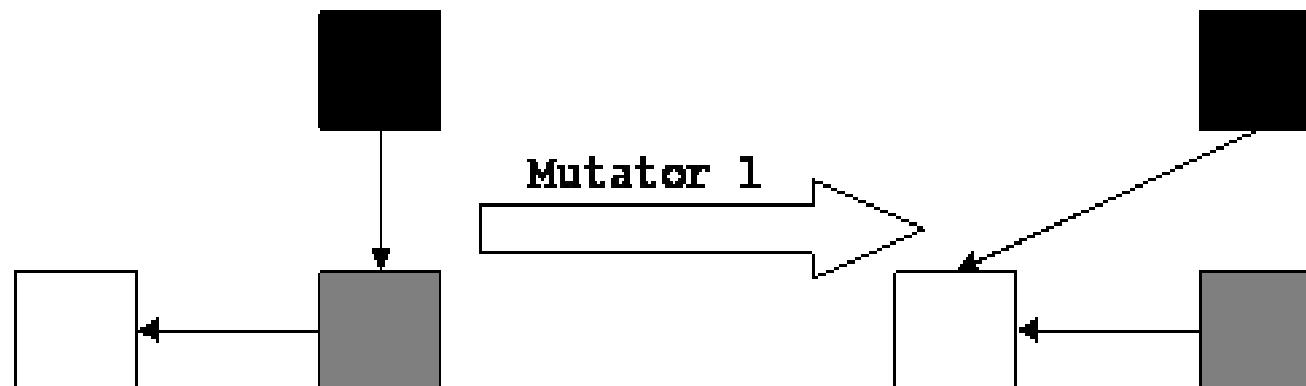
Garbage Collection

- Changes garbage collection strategy:
 - 70% of heap: Fragmented, mark-region collector
 - 30% of heap: Object *spines*, copy-collected
 - Minimizes fragmentation, allows constant-time access
 - Only stop mutator for root scanning
 - Run collector as lower-priority thread



Incremental Garbage Collection

- Dijkstra Write Barrier
 - Tricolor marking:
 - Black: Marked and children visited
 - Grey: Marked but children not visited
 - White: Not reachable
 - Mutator colors target grey on pointer change
 - Collector runs until no grey objects



Supported Java Features

- Most of standard library: `java.*`
 - Either Fiji Core Library or GNU Classpath
- `javax.safetycritical.*`
 - Safety-critical spec. DO-178B
- `javax.realtime.*`
 - RTSJ implementation
- `com.fiji.fivm.*`
 - Fiji-specific RT extensions
- `com.fiji.mvm.*`
 - Fiji API for time/space partitioning on single OS img