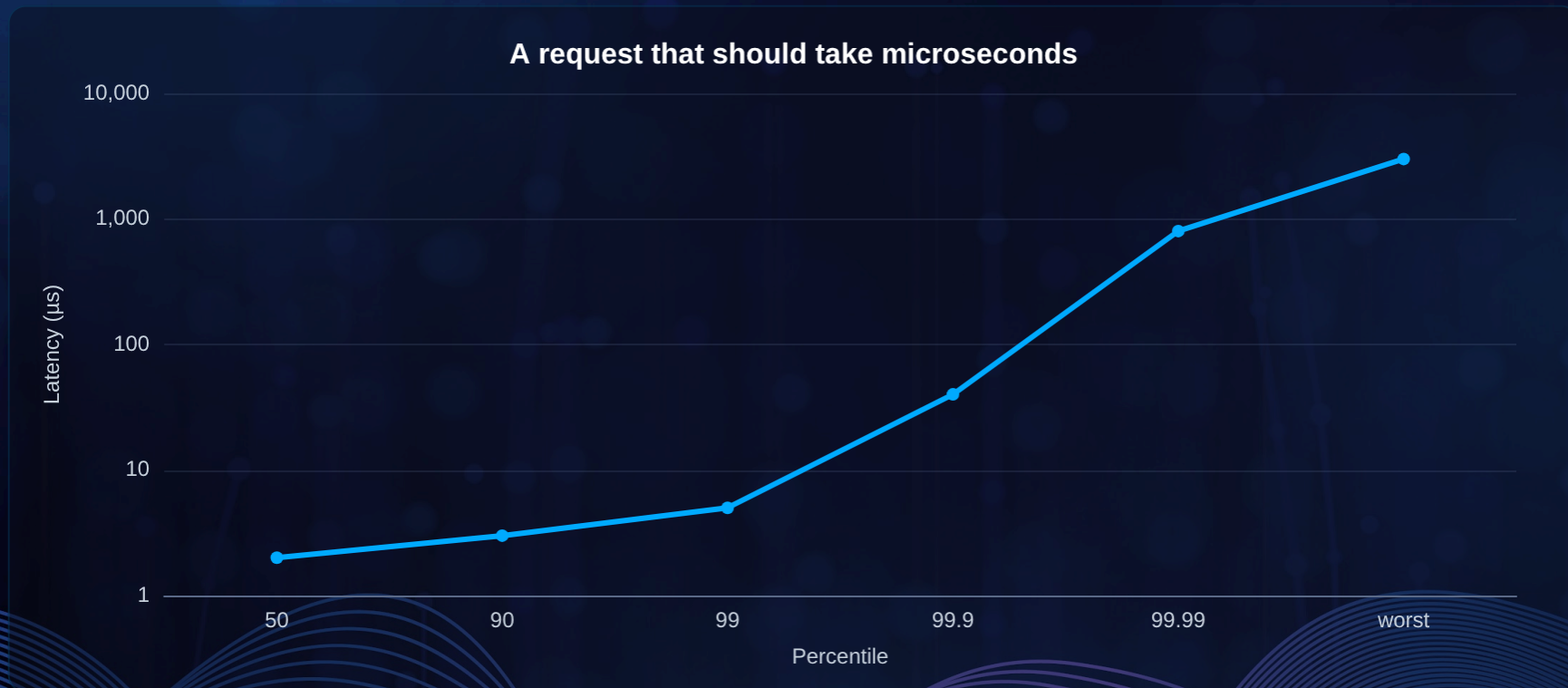


Close to the Metal

Modern low-latency development

Tom Dellmann · Director of Engineering

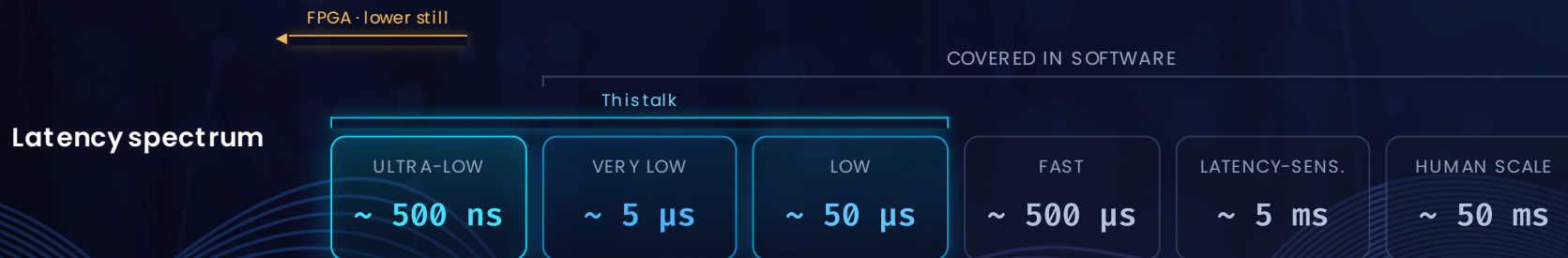
Averages can be deceiving



The matching engine order book

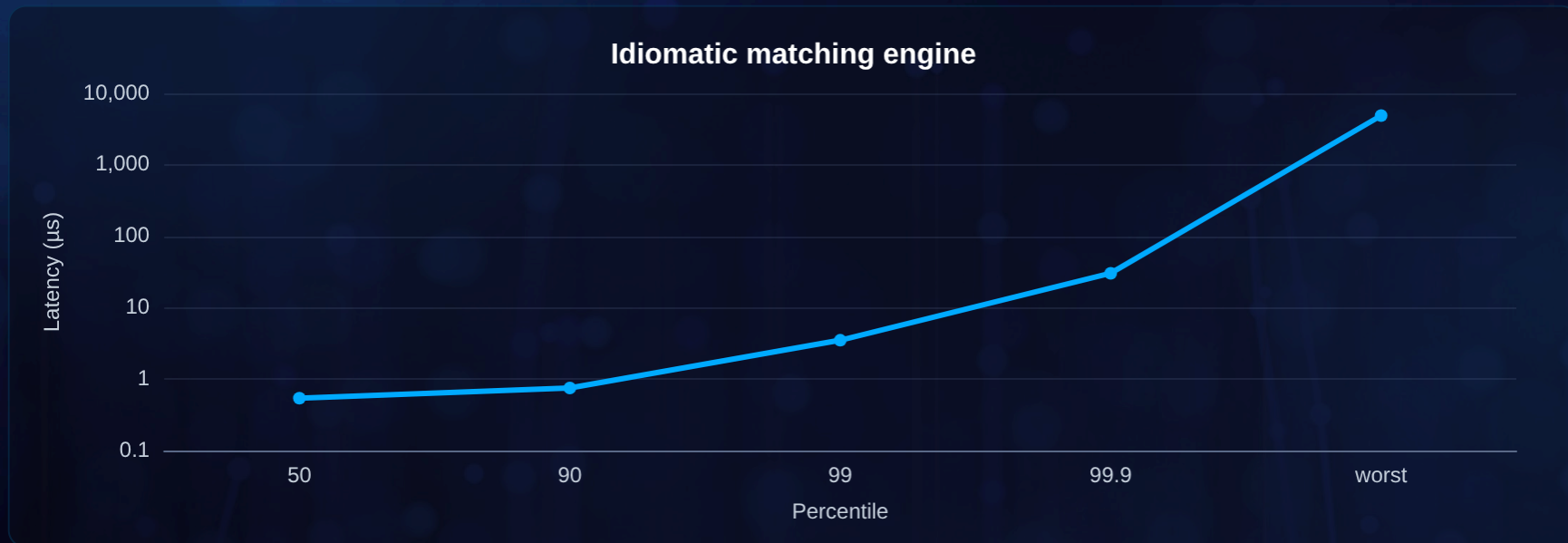


Experiment setup



How well does idiomatic Java code perform?

Idiomatic engine performance



The impact of data layout and garbage collection

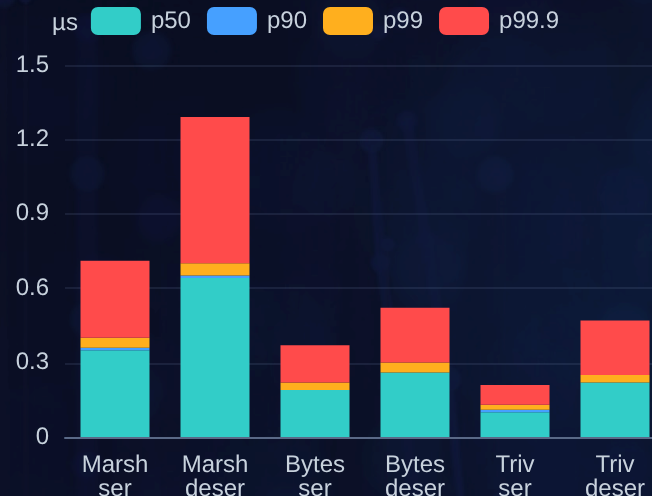
Serialisation trade-offs

```
class Order extends SelfDescribingMarshallable {  
    long price; int qty; int symbol;  
    Side side; TimeInForce timeInForce; // ...  
}
```

```
class Order extends BytesInBinaryMarshallable {  
    long price; int qty; int symbol;  
    Side side; TimeInForce timeInForce;  
    public void writeMarshallable(BytesOut<?> o) {  
        o.writeLong(price); o.writeInt(qty); // ...  
    }  
    public void readMarshallable(BytesIn<?> i) {  
        price = i.readLong(); qty = i.readInt(); // ...  
    }  
}
```

```
class Order extends BytesInBinaryMarshallable {  
    static final int[] R = BytesUtil.triviallyCopyableRange(C  
    static final int START = R[0], LENGTH = R[1] - R[0];  
    long price; int qty; int symbol;  
    byte side, timeInForce; // ...  
}
```

Latency (µs), stacked by percentile



What each flavour costs you

Flavour	Wire size	Self-describing	Readability	Portable	Serialise p50	Deserialise p50
Marshallable	205 B	yes	high, plain fields	yes, across languages	0.32 μ s	0.68 μ s
BytesMarshallable	146 B	no	medium, hand-written read/write	yes, fixed field order	0.19 μ s	0.26 μ s
TriviallyCopyable	148 B	no	low, raw memory block	no, same JVM and layout only	0.12 μ s	0.22 μ s

Fast but with what caveats?

```
class Order { long price; int qty; int symbol; Side side;
              long orderId; int seq;
              short version; byte status; byte priority; }
```

JDK 11

	OFF	SZ	TYPE	DESCRIPTION
	0	8		(object header)
-	12	4	int	Order.qty
-	16	8	long	Order.price
-	24	8	long	Order.orderId
-	32	4	int	Order.symbol
-	36	4	int	Order.seq
	40	2	short	Order.version
	42	1	byte	Order.status
	43	1	byte	Order.priority
	44	4	Side	Order.side

JDK 17

	OFF	SZ	TYPE	DESCRIPTION
	0	8		(object header)
+	12	4	int	Order.symbol
+	16	8	long	Order.orderId
+	24	8	long	Order.price
+	32	4	int	Order.seq
+	36	4	int	Order.qty
	40	2	short	Order.version
	42	1	byte	Order.status
	43	1	byte	Order.priority
	44	4	Side	Order.side

Going Zero GC

	byte[]	byte[]		byte[]	byte[]						java.lang.Objec..			
	java/util/..	java/lang/A..		java/util..	java/lang/Abstr..		java/util/Ar..	java/lan..			java/util/Array..			
			java.I..	java/lang..	java/lang/Strin..	java.lang...	java/lang/St..	java/lan..	soft..	jav..	ja..	ja..		software/chronicl..
java.lang..	java/lang/..	java/lang/S..		java/lang/StringBu..	net/openhft/chr..	java/lang/StringBuilder...	net/open..	java..	jav..	ja..	ja..		java...	software/chronicl..
java/lang/StringBuilde..	net/openhft..			net/openhft/chronicle/wire/LongConve..	net/openhft/chronicle/wire/LongCon..			java..	jav..	ja..	ja..	..	java/..	java/util/TreeMap..
net/openhft/chronicle/wire/LongConve..	software.chronicle.matchev..			software/chronicle/matchevolution/be..	software/chronicle/matchevolution/..			java..	software..	soft..	..		java/..	java/util/TreeMap.computel..
software/chronicle/matchevolution/be..	software/chronicle/matchevolution/matching/BookOrder.from							soft..	software/chroni..				software/chronicle/matchevolu..	
software/chronicle/matchevolution/matching/MatchingEngineImpl.addOrder														
software/chronicle/matchevolution/bench/QueueRoundTripBenchmark.consumerLoop														
software/chronicle/matchevolution/bench/QueueRoundTripBenchmark.lambda\$setUp\$0														
software/chronicle/matchevolution/bench/QueueRoundTripBenchmark\$\$Lambda\$0x000000003e130228.run														
java/lang/Thread.runWith														
java/lang/Thread.run														
[matcher-naive tid=668114]														
all														

matcher-zero-gc: no allocation samples to plot

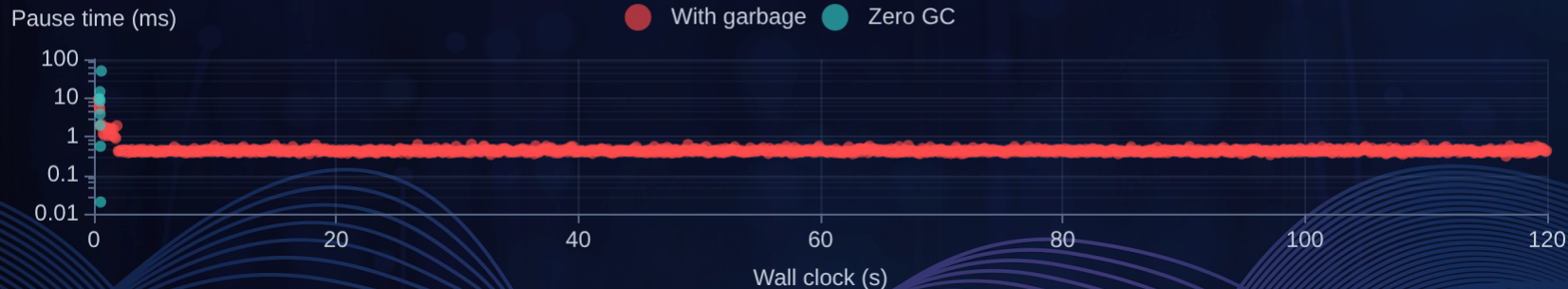
[matcher-zero-gc tid]
all

The power of Zero GC

Before vs after pooling (end-to-end over Chronicle Queue)



GC pauses over a 2-minute run (log scale)



Data structure selection impact

TreeMap



Array-indexed

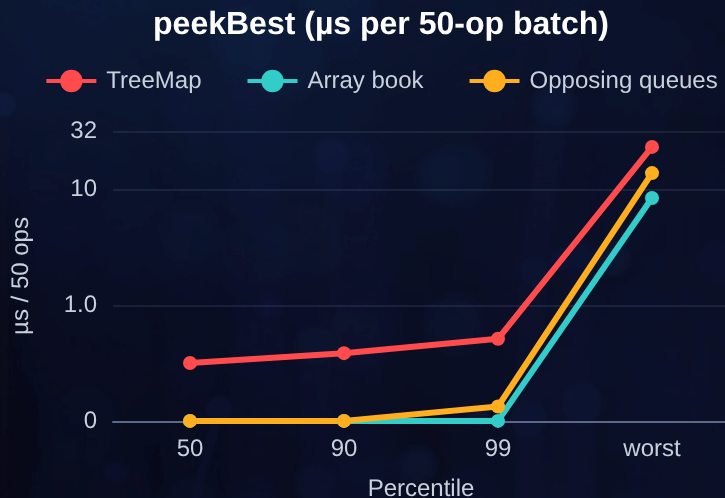
idx 0	100.00
idx 1	100.01
idx 2	100.02
idx 3	100.03

Opposing queues

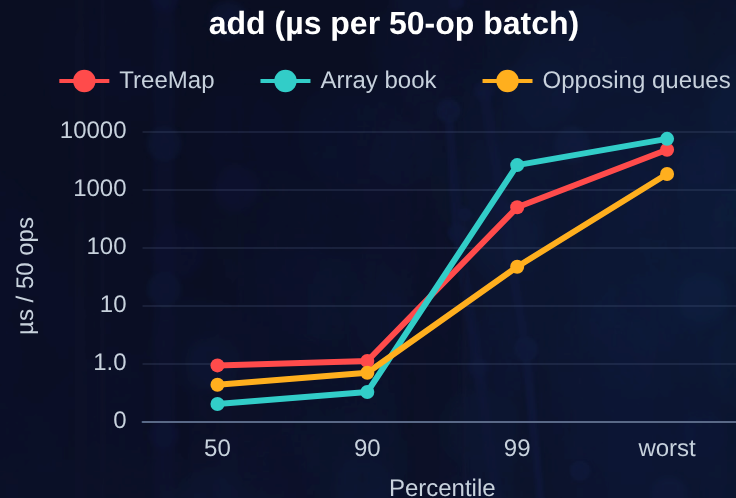


Memory access tradeoffs

Read: peekBest

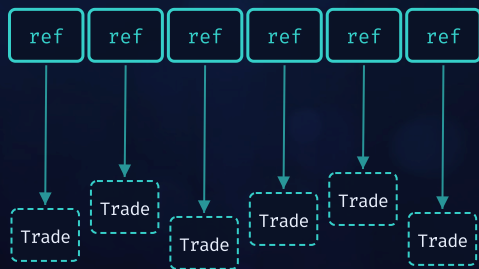


Write: add

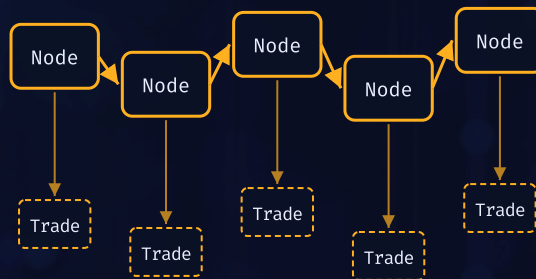


Memory layout of a list of trades

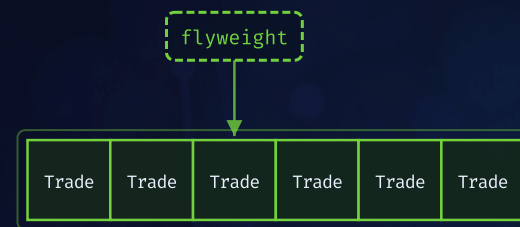
ArrayList<Trade>



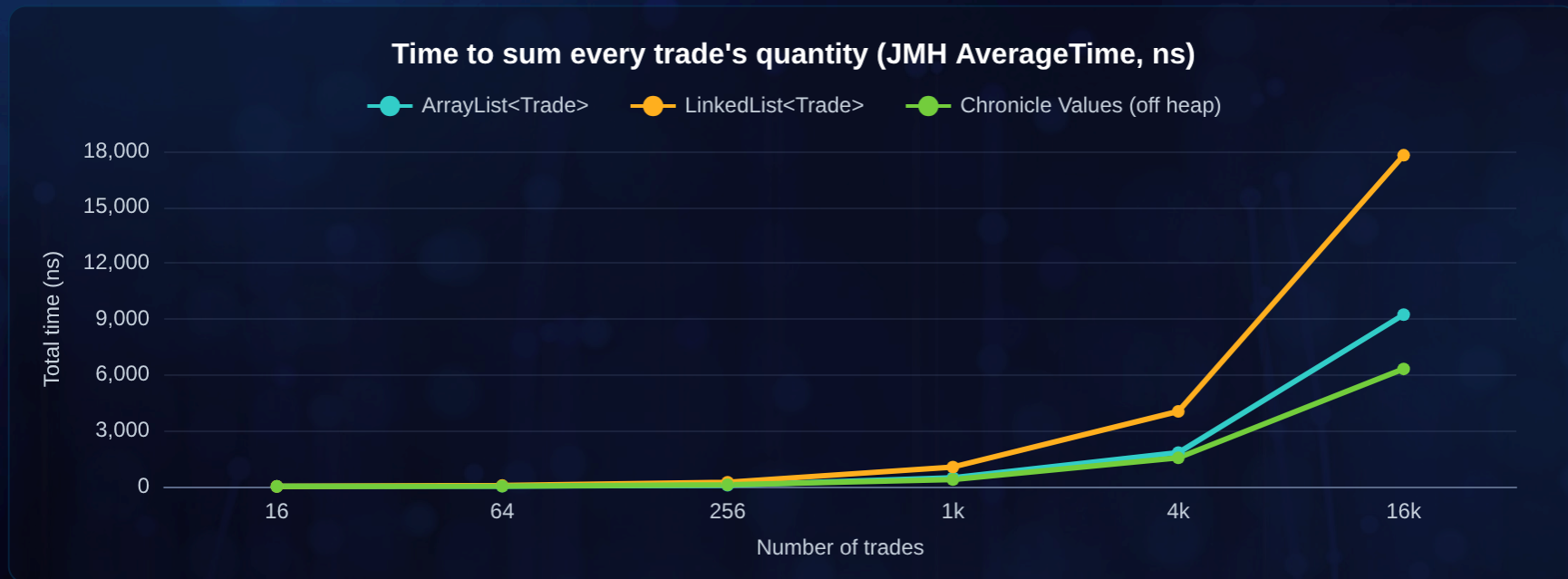
LinkedList<Trade>



Chronicle Values (off heap)



Throughput impact of data structure layout



Getting the most from the OS

Effectiveness of single threaded workloads



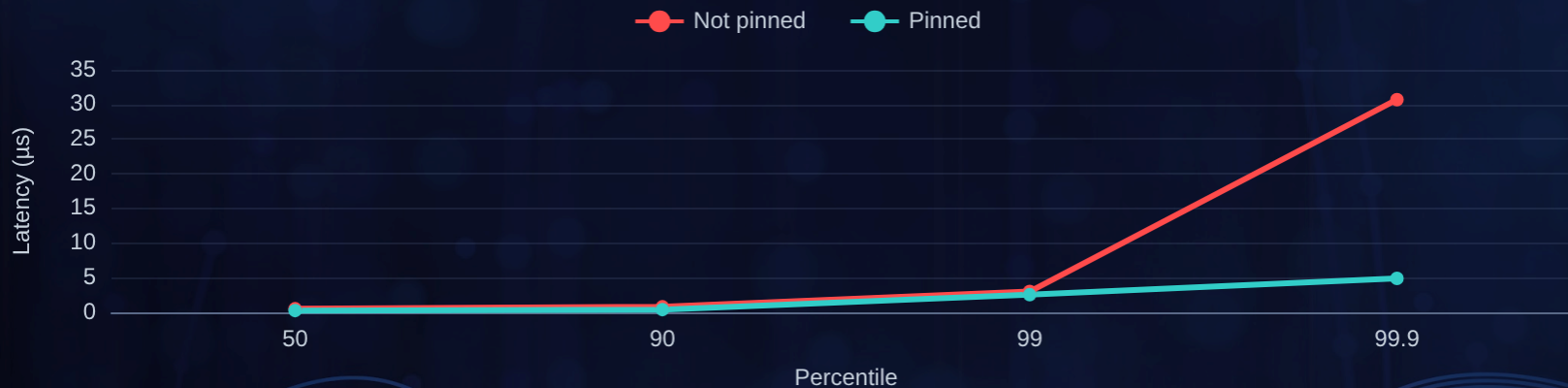
```
class MatcherHandler implements EventHandler {  
    public boolean action() {  
        Order o = inbound.poll();  
        // No work this tick  
        if (o == null) return false;  
        engine.match(o);  
        // Work done  
        return true;  
    }  
}
```

```
try (EventLoop loop = EventGroup.builder()  
    .withName("trade")  
    .withBinding("3")  
    .withPauser(Pauser.busy())  
    .build()) {  
    loop.addHandler(new MatcherHandler());  
    loop.addHandler(new ExecutionHandler());  
    loop.start();  
}
```

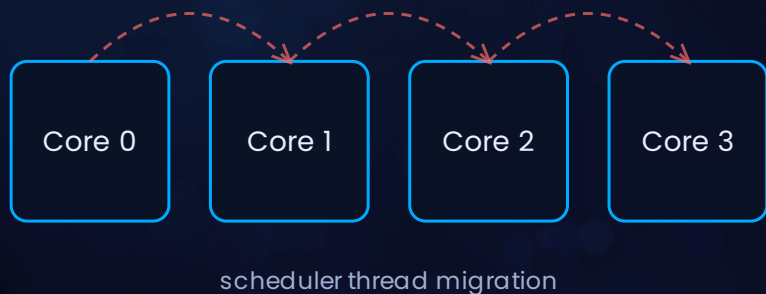
Stopping threads from wandering

```
// Pin the matcher thread to a dedicated core  
EventGroup.builder().withBinding("3"); // ...  
// Or on any threads  
AffinityLock.acquireCore(); // ...
```

Before vs after pinning

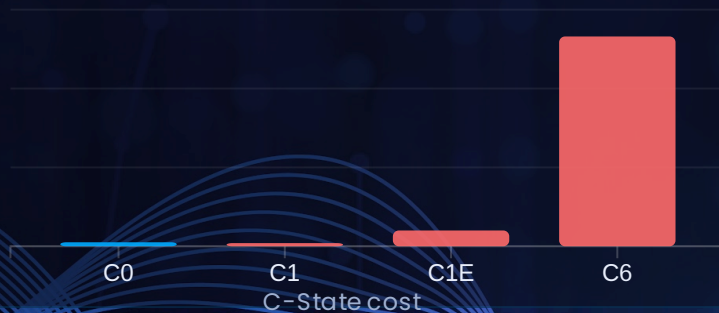


The cost of interruptions



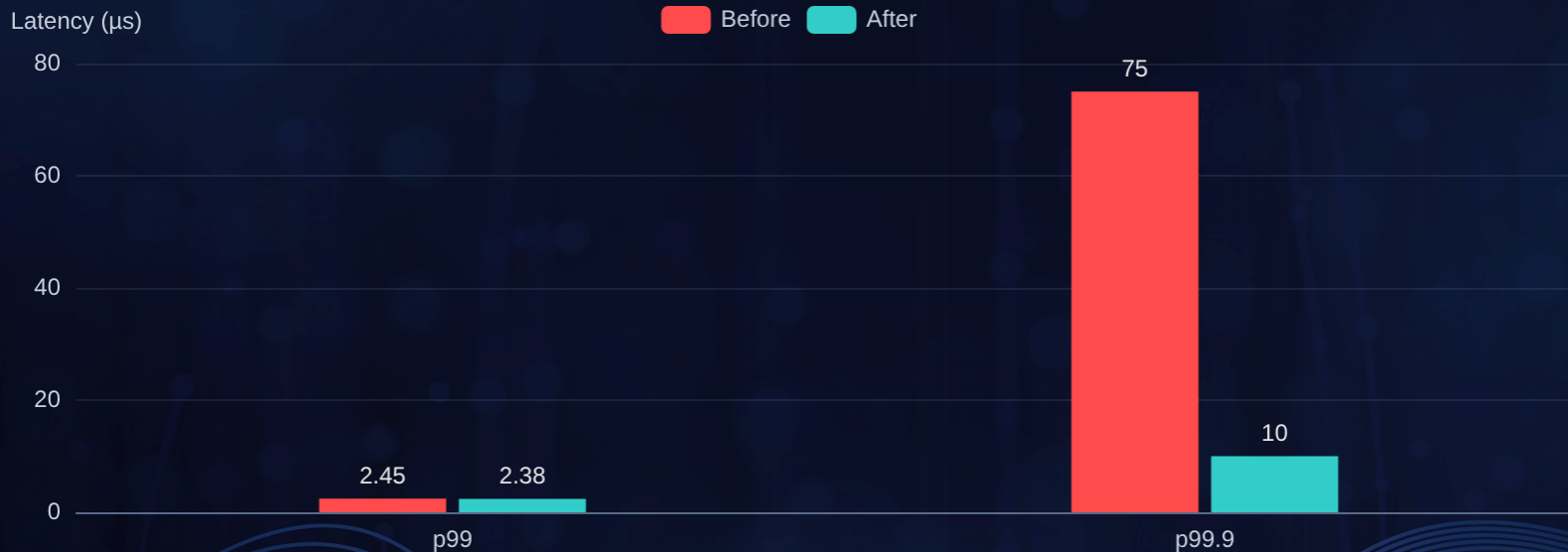
- CPU isolation
- SMT control
- IRQ steering
- C-states
- Power governor
- Kernel tuning

exit latency



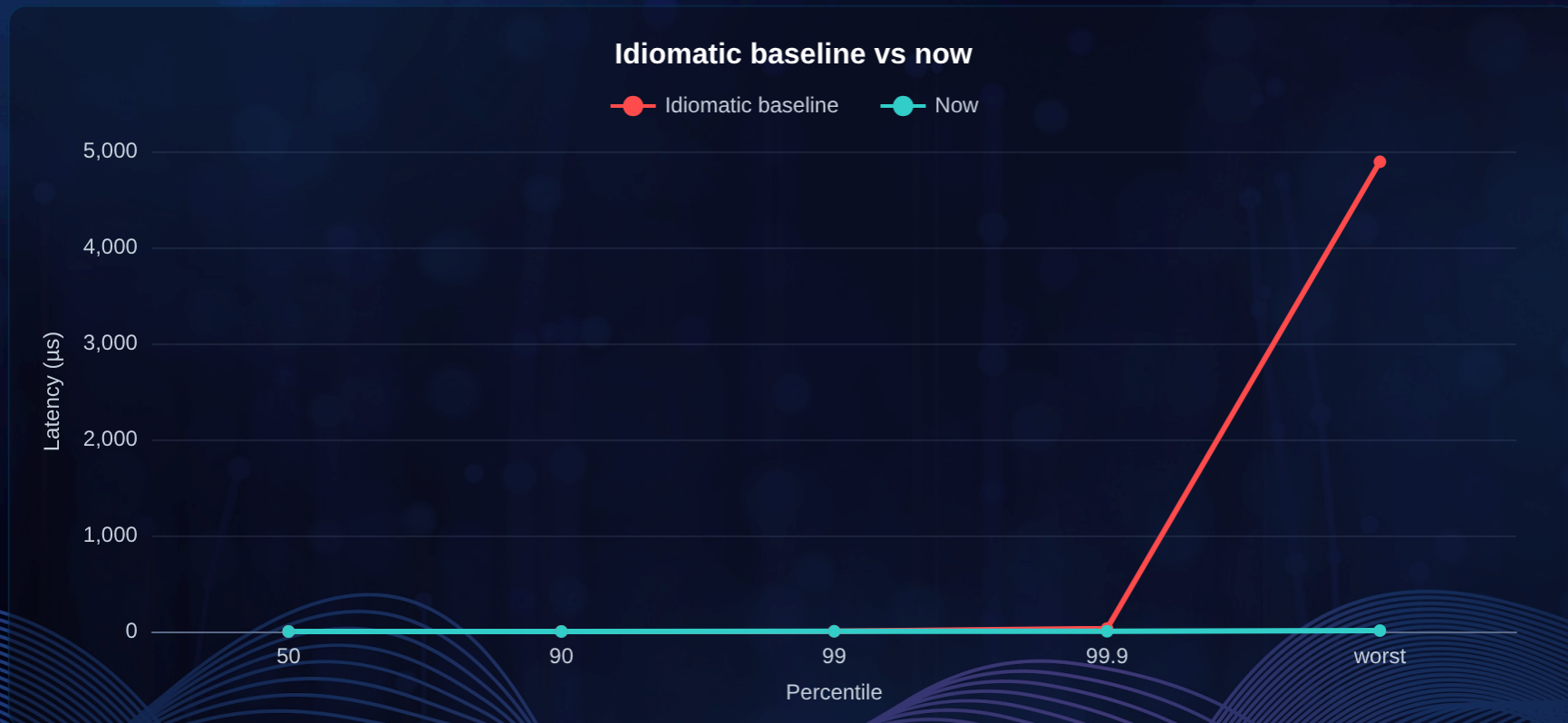
Tail latency before and after OS tuning

Before vs after OS tuning (tail latency)

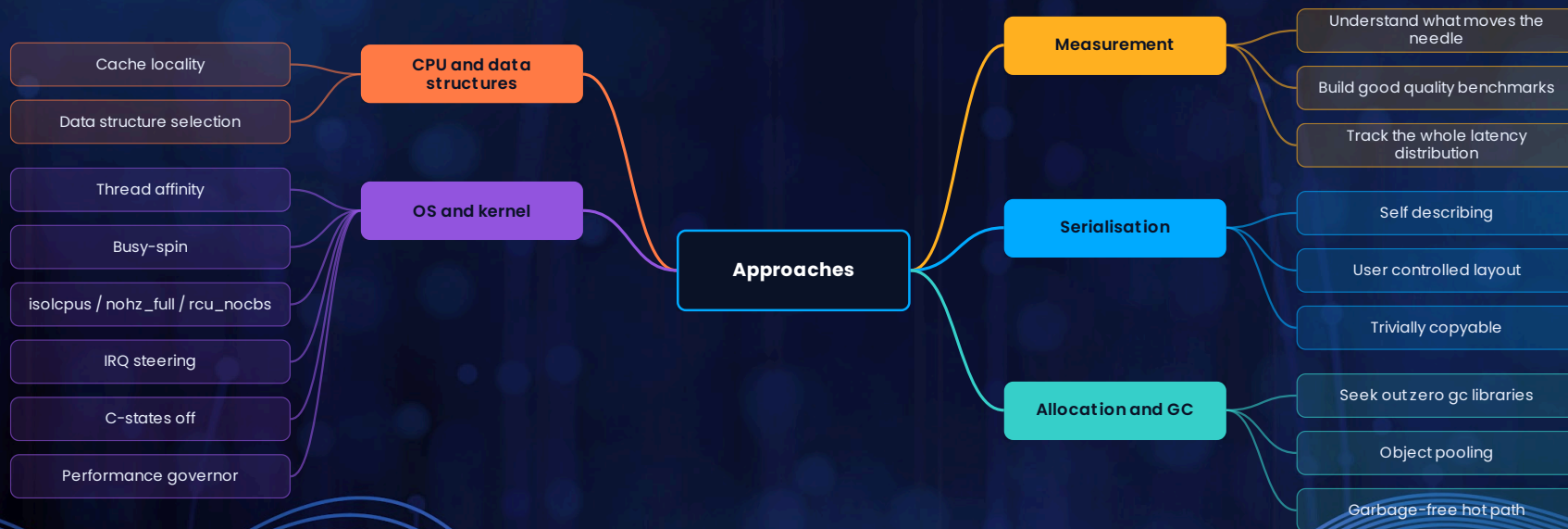


Where we got to

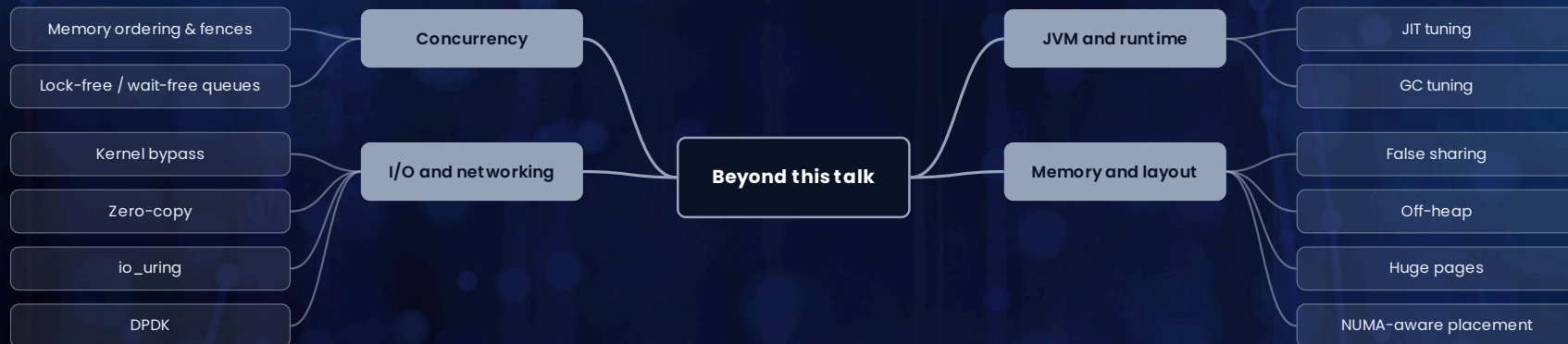
Where we started, where we are



Recap



Deeper optimisations...



Thanks

Close to the metal, one measured change at a time.

Tom Dellmann · Chronicle Software