Microbenchmarking in Java

Ben Evans and James Gough

This Talk

- Who are we?
- Why Microbenchmarking is not for everyone
- The need for JMH
- Demo of JMH
- Importance of Statistics in Benchmarking

About Us

Ben

- Co-founder & Tech Fellow, jClarity
- Trainer and Author
- Surfer, whisky expert
- `@kittylyst`

jClarity
Performance Tuning via Machine Learning

O'REILLY

James

- Java(script) developer, teacher and author
- Works primarily in Technology Training
- Father, Hacker, aspiring whisky expert
- `@Jim__Gough`

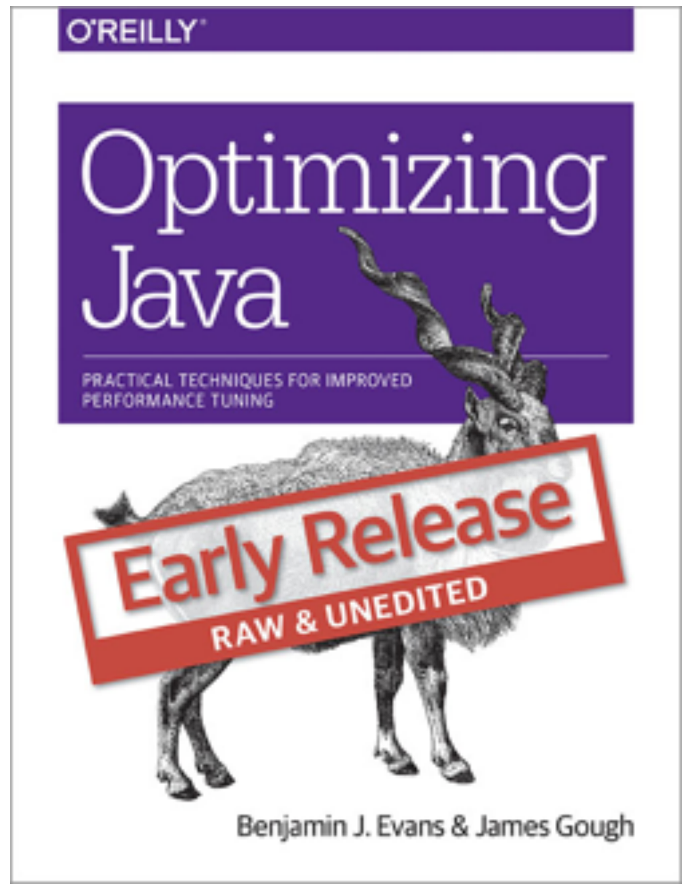
Bloomberg

Community

- Java Community Process Executive Committee
- London Java Community
  - Organising Team, AdoptAJSR
- **Ben:** Java Champion & JavaOne Rock Star Speaker

Writing

- **Ben**
  - Java in a Nutshell (6th Edition)
  - Introduction to Java 8
  - The Well-Grounded Java Developer
- **Ben and James**
  - Optimizing Java (forthcoming)


O'REILLY
Optimizing Java
PRACTICAL TECHNIQUES FOR IMPROVED PERFORMANCE TUNING
Early Release
RAW & UNEDITED
Benjamin J. Evans & James Gough

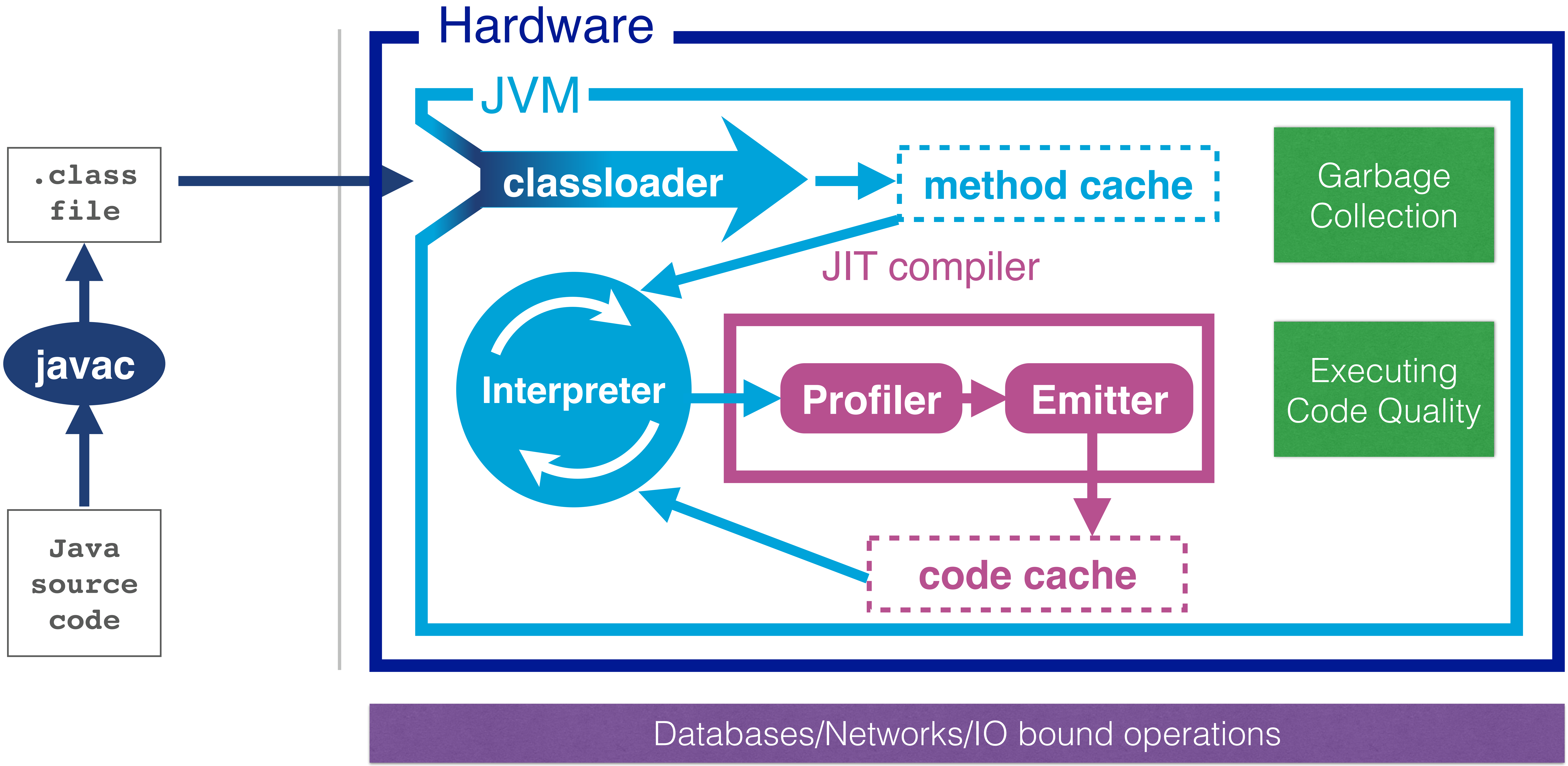
What Is Java Performance?

- “A measurement-driven approach to understanding an applications behaviour under load”
    - Note: Measurement-driven

- This sets us up for a clash between people & data

- Performance is a huge topic

Performance Landscape



Where is Microbenchmarking Relevant?

- General-purpose library code broad use cases
- Developer on OpenJDK or another Java platform implementation
- Extremely latency sensitive code
    - Low-latency trading

Microbenchmarking Frameworks

- Main methods with self invented timers
    - Time for the pub!

- Google Caliper
    - Not very active (last commit in Jan)
    - Struggled to avoid the JVM bear traps

Criterium (Clojure)

Criterium measures the computation time of an expression. It is designed to address some of the pitfalls of benchmarking, and benchmarking on the JVM in particular.

This includes:

- statistical processing of multiple evaluations
- inclusion of a warm-up period, designed to allow JIT to optimise code
- purging of gc before testing, to isolate timings from GC state prior to testing
- a final forced GC after testing to estimate impact of cleanup on the timing results

Microbenchmarking Frameworks

- JMH
  - Written by the authors of the JVM
  - Used to performance test parts of the JVM
  - Learned from others mistakes (hopefully)

Why JMH?

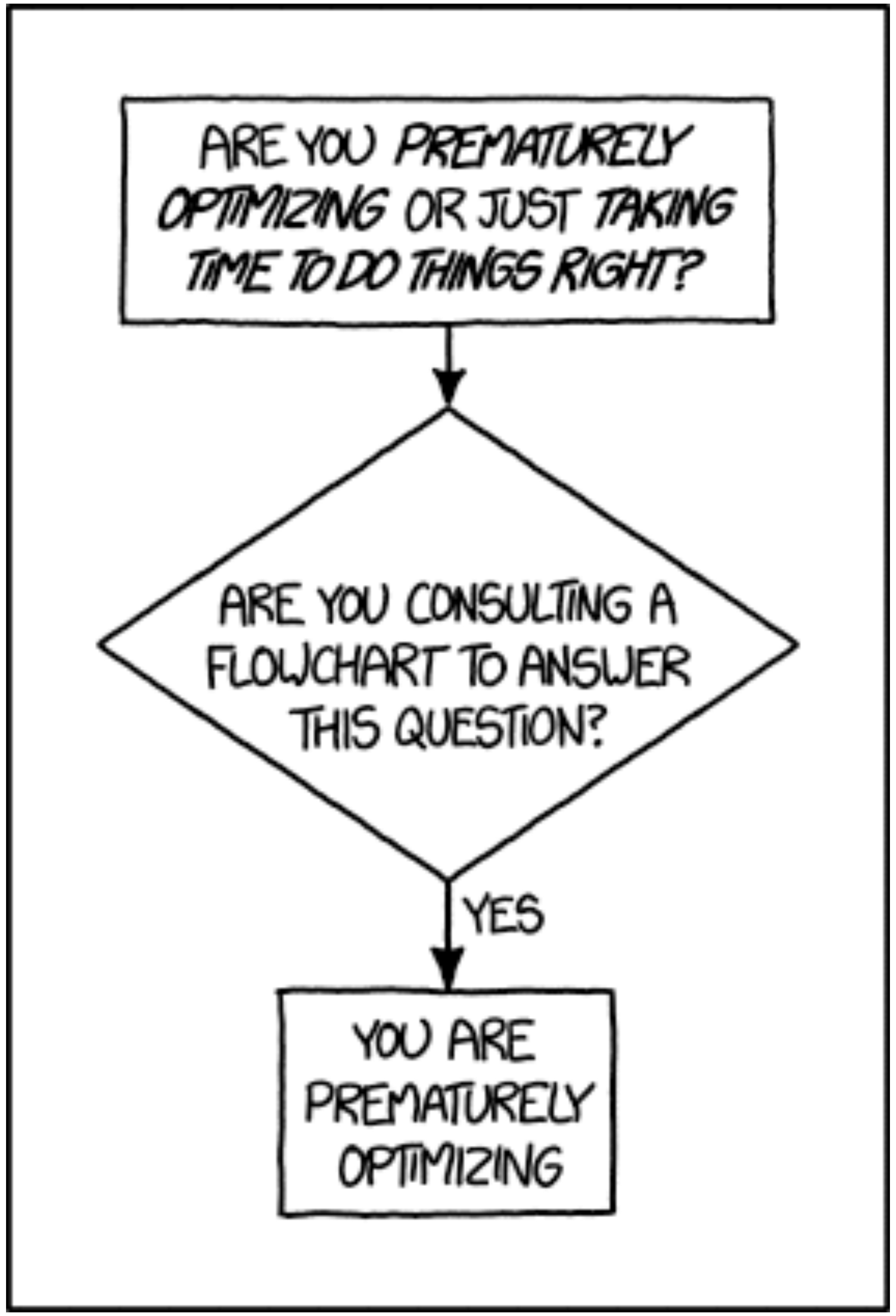
- Understands the JVM, because they wrote it
- Power Management Issues
  - Power management can cause poor benchmarks
  - JMH uses spin loops to ensure core is activated
- OS Scheduling Issues
  - Scheduling issues resolved by running process longer

Selecting and Executing Benchmarks

- Benchmark frameworks must be dynamic
  - Using reflection can introduce issues
  - Optimisations between test and benchmark
- JMH generates wrapper code to avoid this
  - Carefully avoiding JVM optimisations
- These complexities are hidden from the user
  - It’s hard enough to write that code

Common Benchmark Issues

- Problem1


ARE YOU PREMATURELY OPTIMIZING OR JUST TAKING TIME TO DO THINGS RIGHT?
ARE YOU CONSULTING A FLOWCHART TO ANSWER THIS QUESTION?
YES
YOU ARE PREMATURELY OPTIMIZING

http://xkcd.com/1691/

Optimising Away Benchmarks….

- A pitfall is part of the benchmark being optimised away
    - Easily happens when nothing is done with the result
- JMH provides an easy mechanism to prevent this
    - Must be efficient and avoid optimisation

Blackholes

```
public volatile int i1 = 1, i2 = 2;

public final void consume(int i) {

  if (i == i1 & i == i2) {

     // SHOULD NEVER HAPPEN

     nullBait.i1 = i;

    }

}
```

Blackholes

```
public int tlr = (int) System.nanoTime();

public final void consume(Object obj) {

   int tlr = (this.tlr = (this.tlr * 1664525 + 
1013904223));

    if ((tlr & tlrMask) == 0) {

     // SHOULD ALMOST NEVER HAPPEN IN MEASUREMENT

        this.obj1 = obj;

        this.tlrMask = (this.tlrMask << 1) + 1;

    }

}
```

Getting Started

```
$ mvn archetype:generate \
   -DinteractiveMode=false \
   -DarchetypeGroupId=org.openjdk.jmh \
   -DarchetypeArtifactId=jmh-java-benchmark-archetype \
   -DgroupId=org.sample \
   -DartifactId=test \
   -Dversion=1.0
```

Demo Time

Empirical Performance Analysis

- Cognitive Biases in Performance
- Review of statistics for the JVM

Why Measure?

- Humans are poor at guessing
  - Measurements can be subjective
    - Especially Time measurements

- We all have cognitive biases
  - Especially Confirmation Bias

Cognitive Bias - Definition

- Cognitive biases are psychological tendencies that cause the human brain to draw incorrect conclusions.

Cognitive Biases

- Confirmation Bias
- Reductionist Bias
- Action Bias (“Fog of War”)
- Anti-Risk Bias
- Hyperbolic Discounting
- Information-Gathering Bias

Probability-Specific Cognitive Biases

- Texas Sharpshooter Fallacy
- Clustering Illusion
- Disregarding Regression to the Mean
- Attention Bias
- Recency Bias

Why Measure?

- Developers tend to think along “golden paths” in code
  - Testers are trained to think down darker paths
- Modern systems are exceedingly complex
  - Lots of external meddlers
    - Virus scans, other apps, backups, the cleaner...

Humans are bad at spotting patterns

- Best tool against cognitive biases is data

- Need logging & monitoring
  - But also analysis
  - Data can overwhelm
  - Patterns aren’t always easy to spot by eye

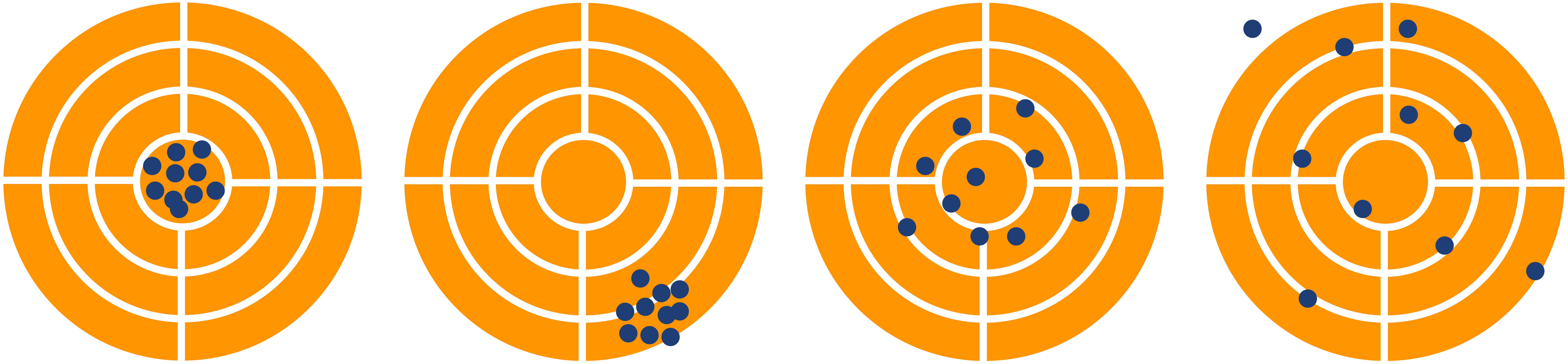
Measurement & Statistics

- Proper collection processes are needed
  - Too many outages are analysed via ad-hoc data
- Ensure sufficient logging
  - Can we retrace all the steps of an outage?

Statistical Data

- Treat our performance observables like experimental data
- Collect data
- Build distributions
- Account for and understand sources of error
  - Systematic Error (Accuracy)
  - Random Error (Precision)

Systematic and Random Error



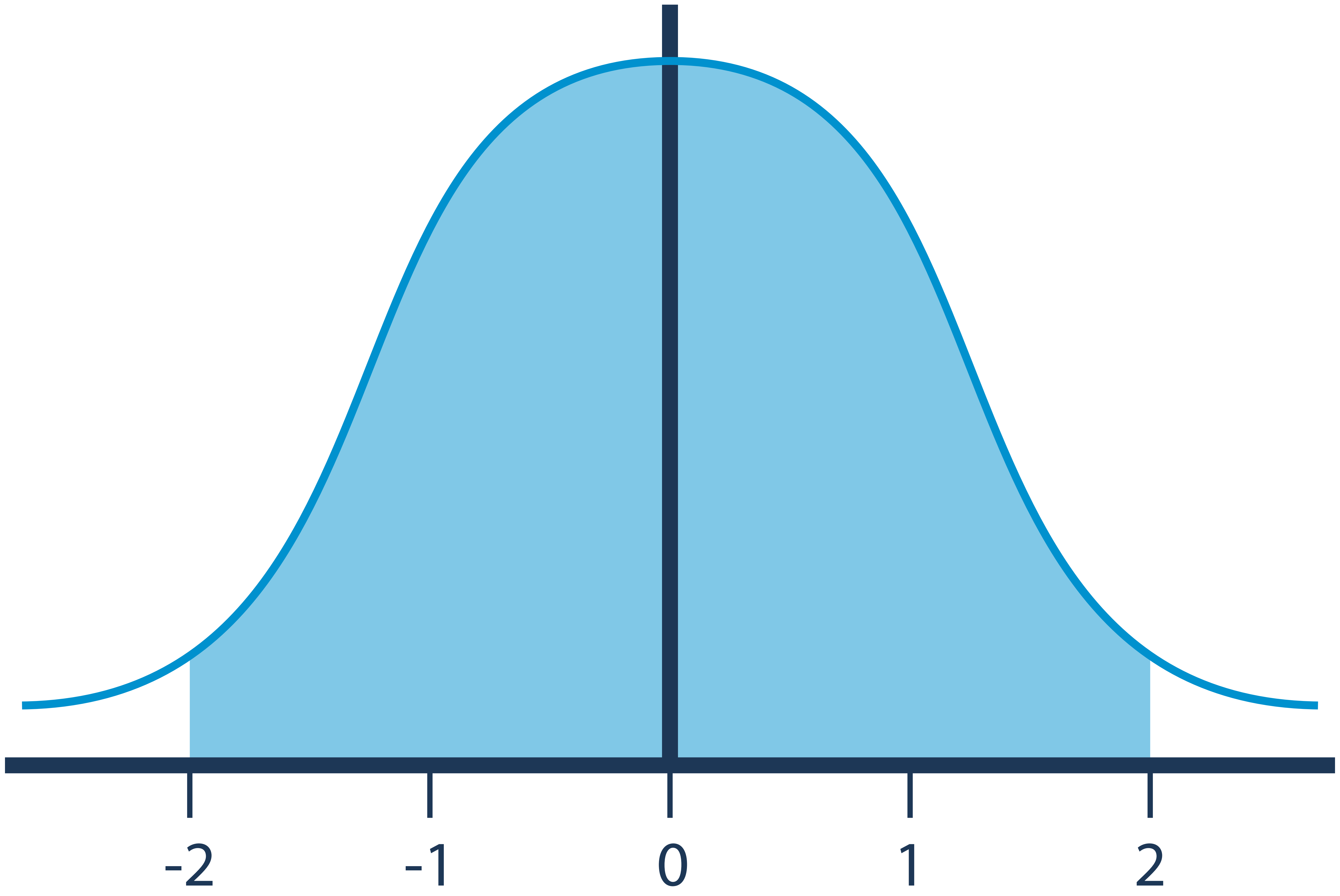
Know Basic Statistics

- Everyone should know:
    - Mean
    - Mode
    - Percentiles
    - Probability distributions

Know Basic Statistics

- Sometimes useful
  - Standard Deviation (be careful)
  - Significance Levels
  - Central Limit Theorem
  - p-values

Normal distributions



Non-Normal Statistics

- Real data often is not normally distributed

- JVM applications have a “hot path” where everything works
  - Deviations from the path add latency
  - Latency >> random error
  - Latency is never negative

- Gives rise to a “long tail” distribution
  - Technically, a specific kind of Gamma distribution

Non-Robust Statistics

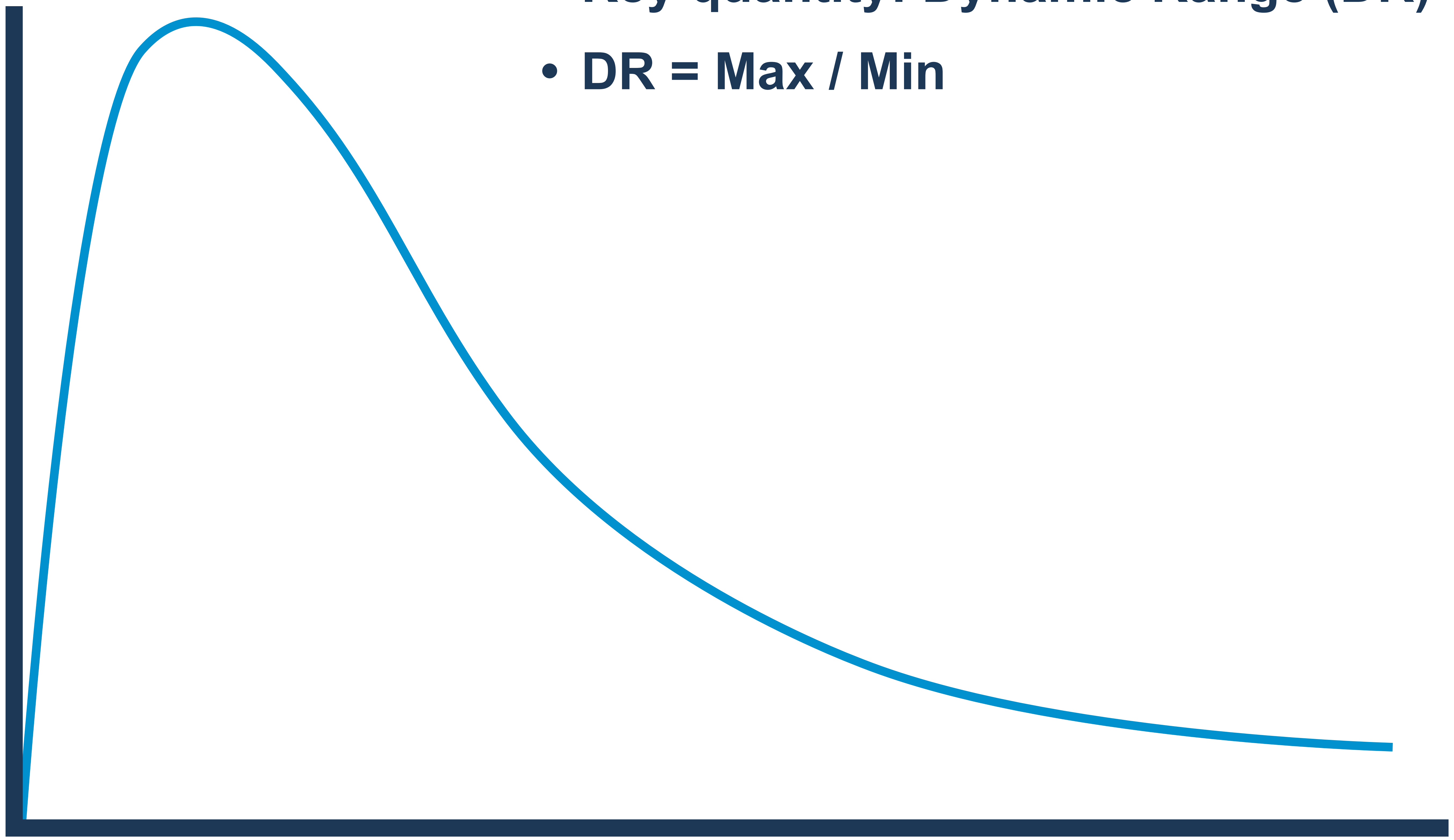
- Non-robust statistics simultaneously:
  - Bend to skew by outliers
  - Dilute the meaning of those outliers
- - from “Statistics for Software” by Mahmoud Hashemi (Paypal)

Non-Normal Statistics

- Normally-distributed statistics
  - Are easy and familiar to many
  - Aren’t much help for most software performance
    - Especially standard deviation

Gamma distribution

- Key quantity: Dynamic Range (DR)
- DR = Max / Min



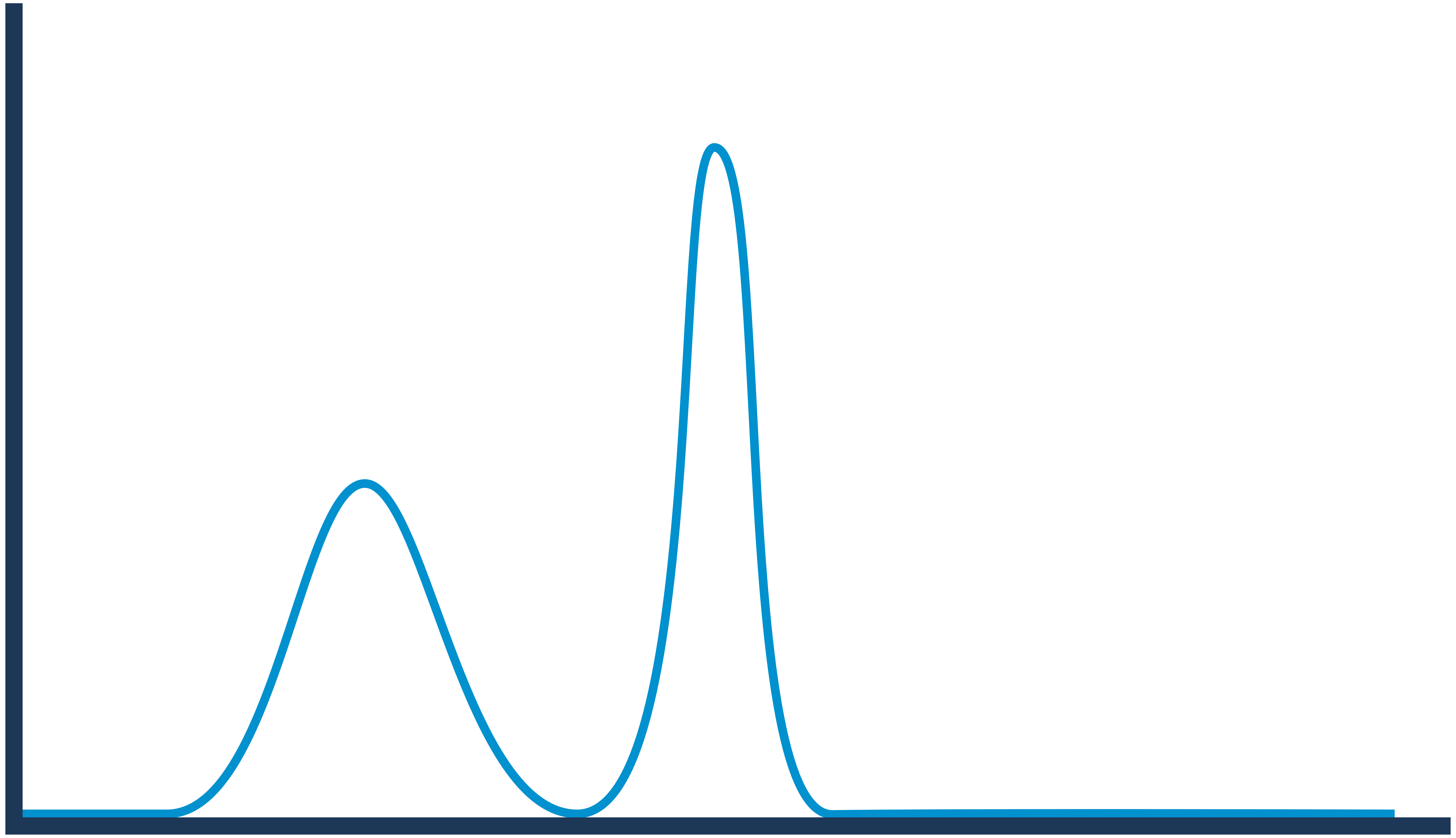
Long-tail Percentiles

- One useful technique is “long-tail percentiles”
  - Compensates for the high dynamic range

- Example
  - Getter method timing

```
50.0% level was 23 ns
90.0% level was 30 ns
99.0% level was 43 ns
99.9% level was 164 ns
99.99% level was 248 ns
99.999% level was 3,458 ns
99.9999% level was 17,463 ns
```

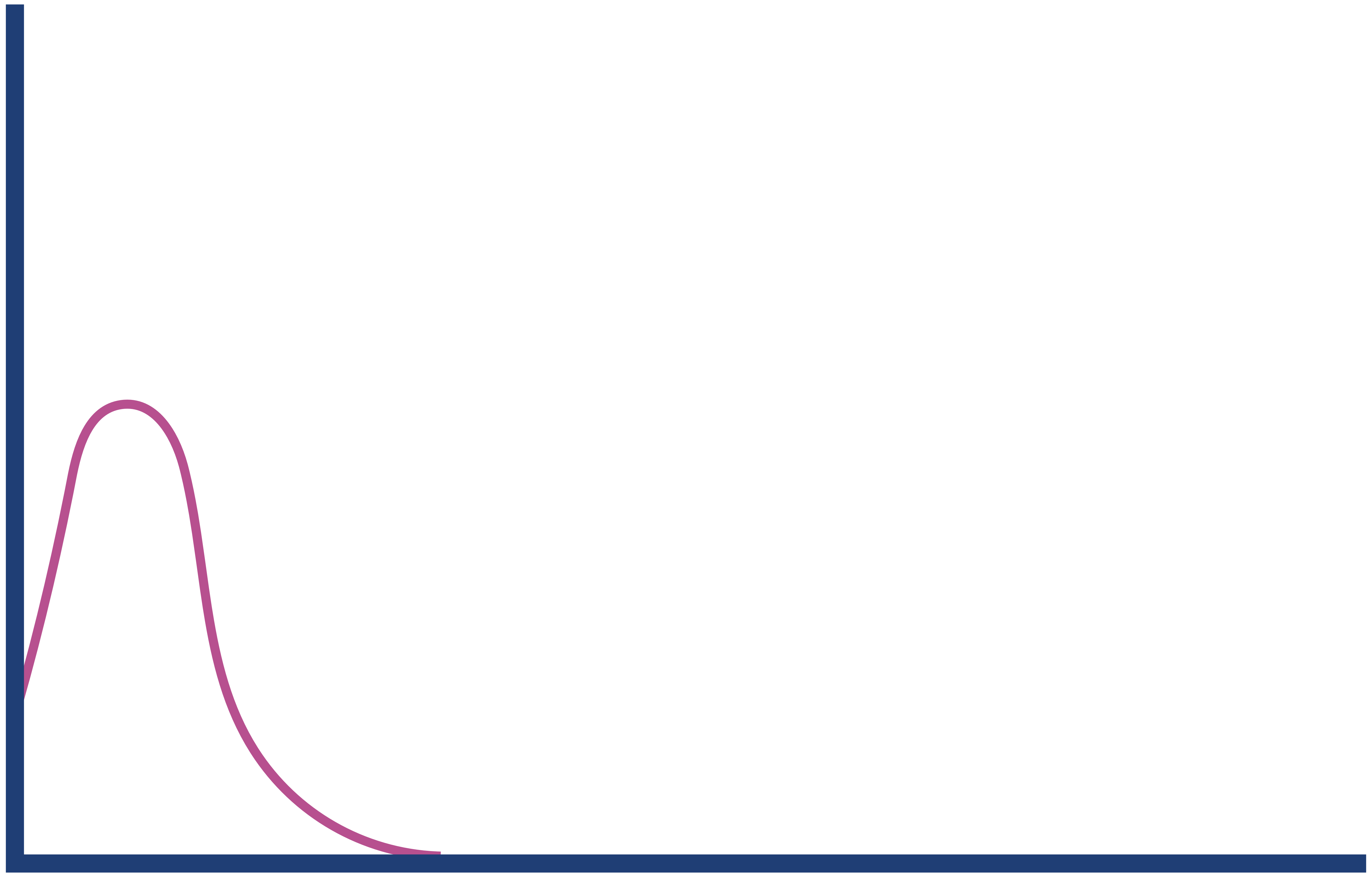
Bimodal distribution



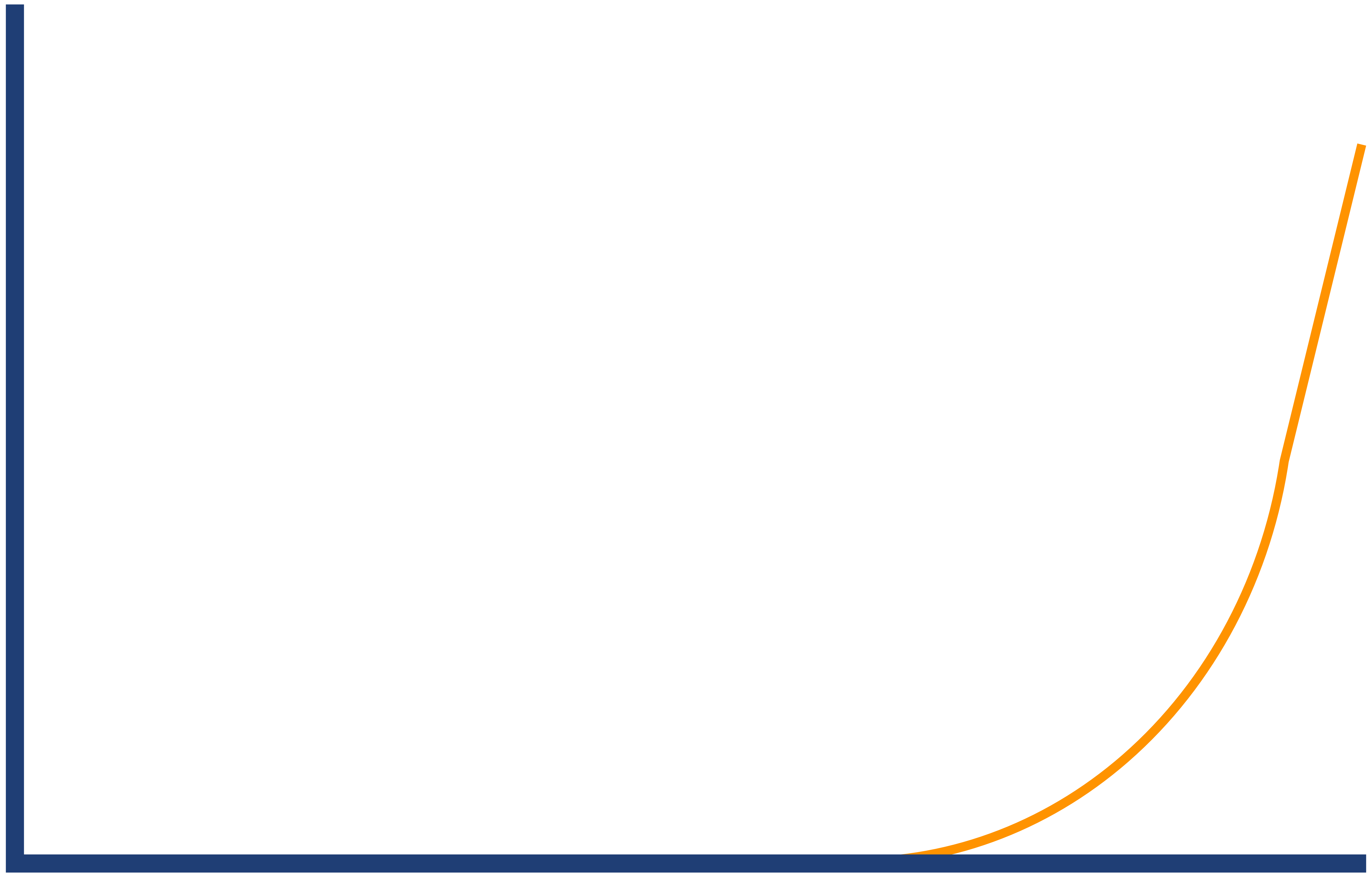
Different Outcomes Have Different Distributions

- Recall HTTP Response Codes
- 2XX (Success)
- 4XX (Client Error)
- 5XX (Server Error)

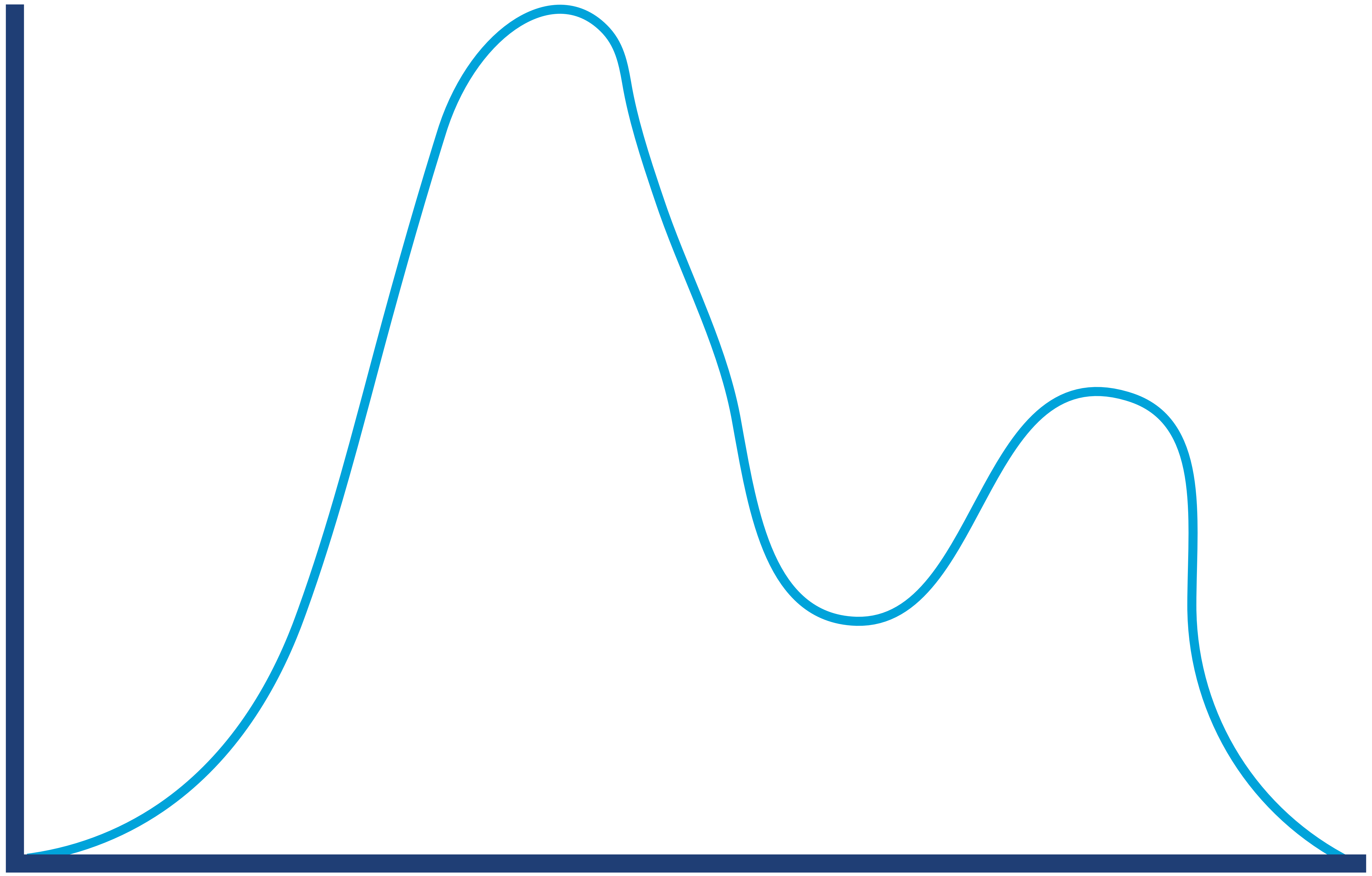
Client Error Response Times



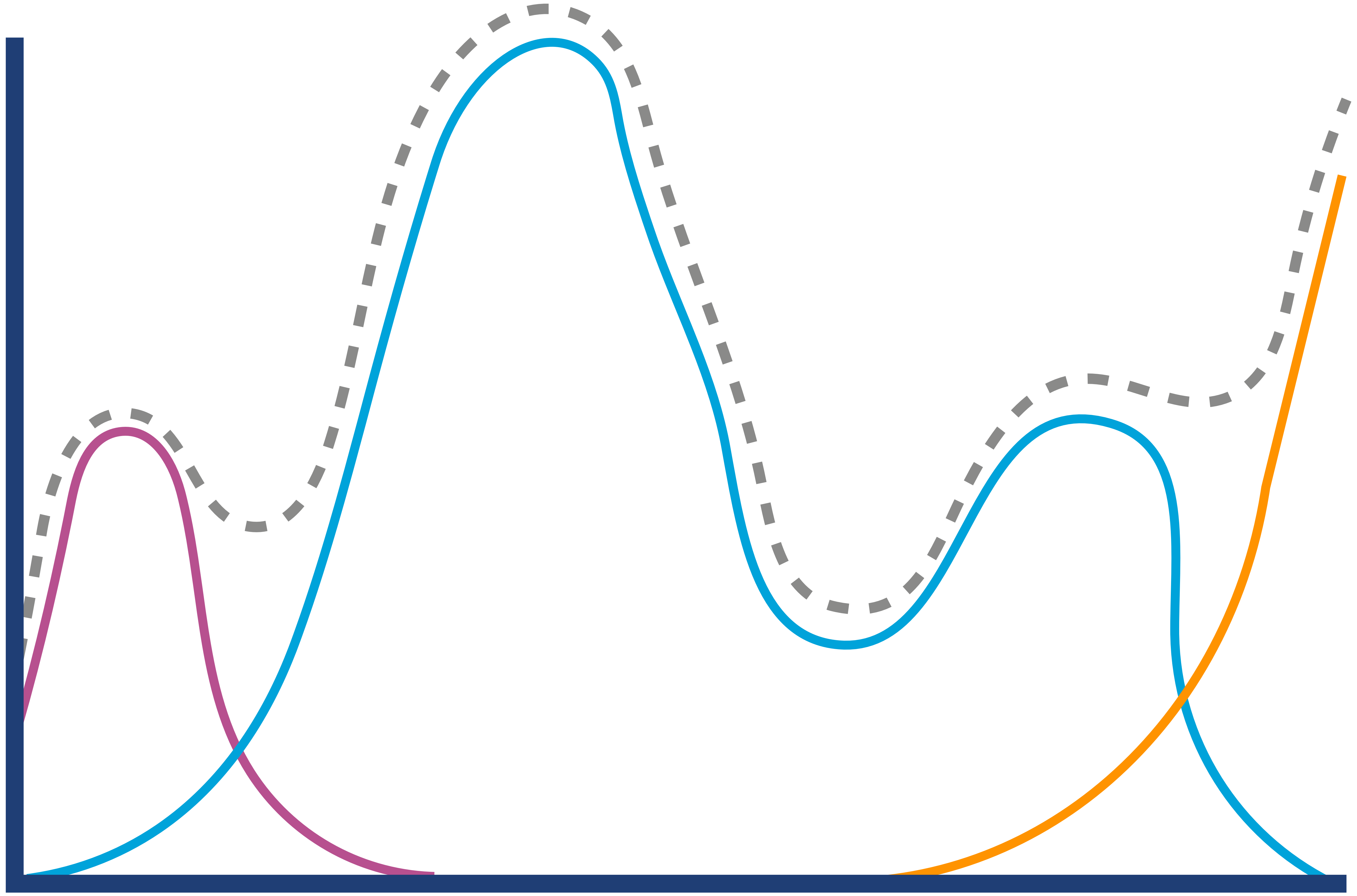
Server Error Response Times



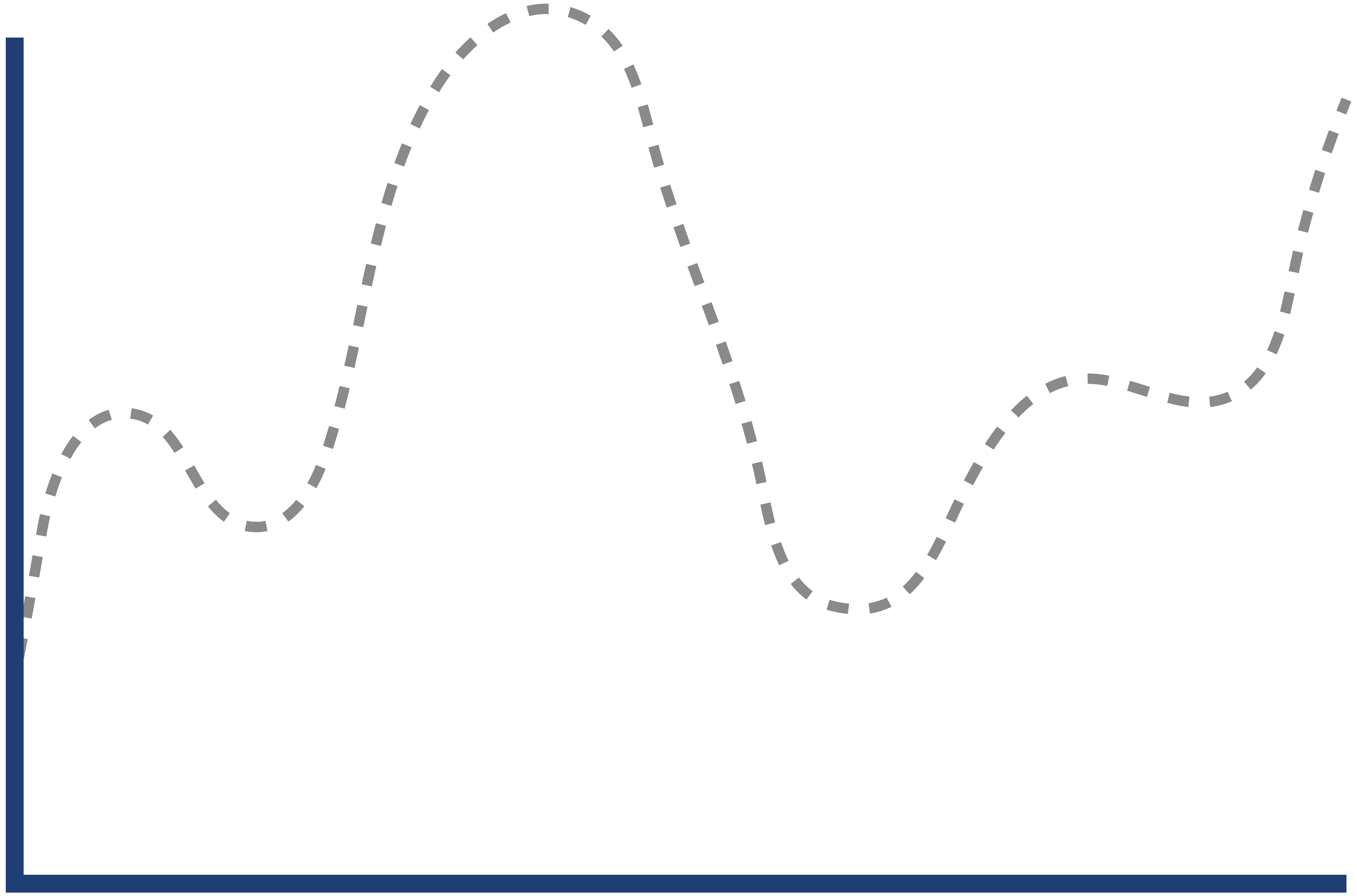
Success Response Times



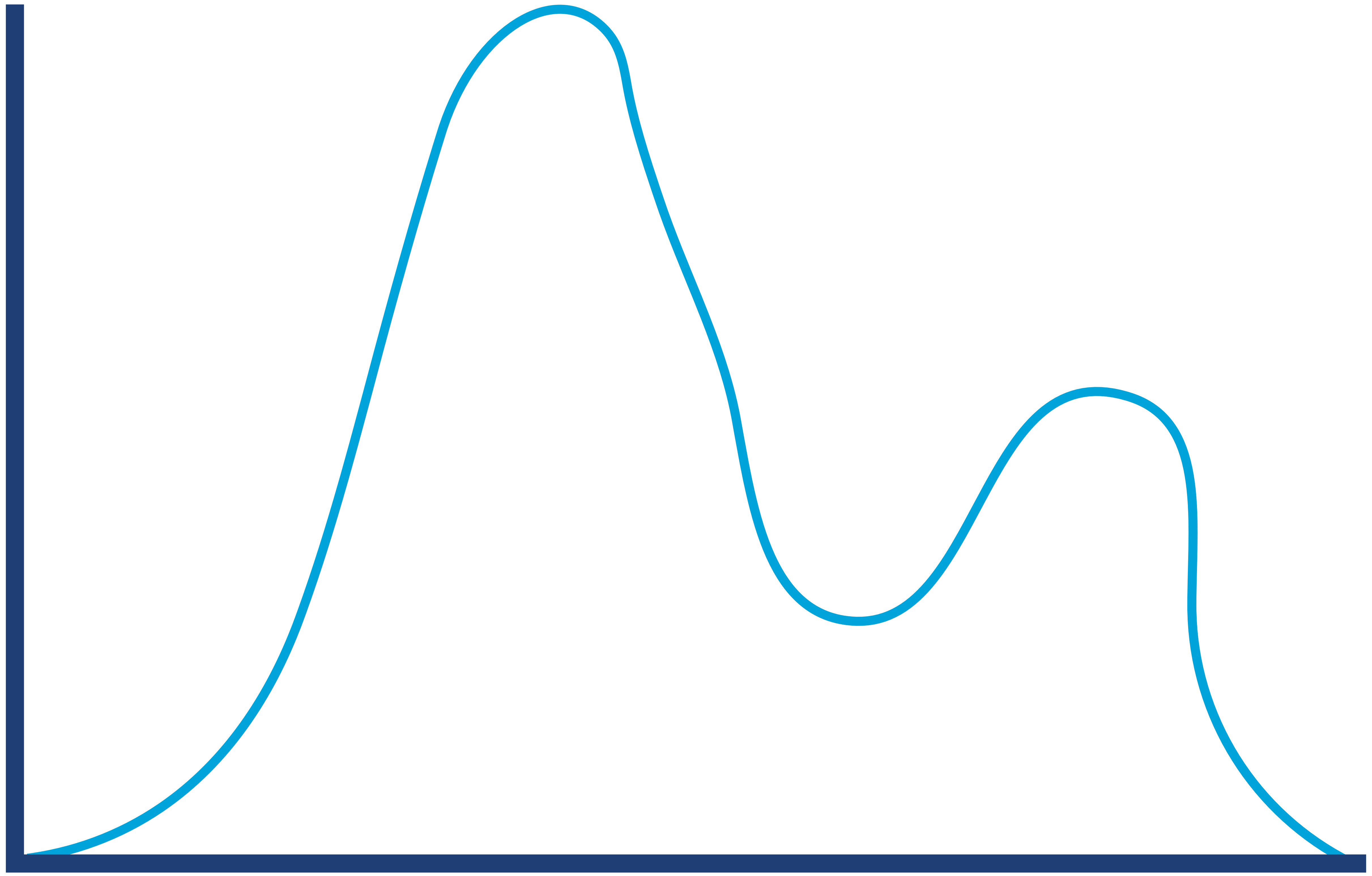
Combined Response Times



Hat or Elephant?



Subpopulations Within Success



Why is the JVM a Special Case?

“C++ implementations obey the zero-overhead principle:
What you don't use, you don’t pay for.
And further, what you do use, you couldn't hand code any better.”

- Bjarne Stroustrup

“Java is a blue-collar language. It’s not PhD thesis material but a language for a job.”

- James Gosling

THANK YOU

Products

jClarity Censum: The world's best GC log analysis tool

jClarity Illuminate: The learning performance problem finder

Community - www.meetup.com/londonjavacommunity

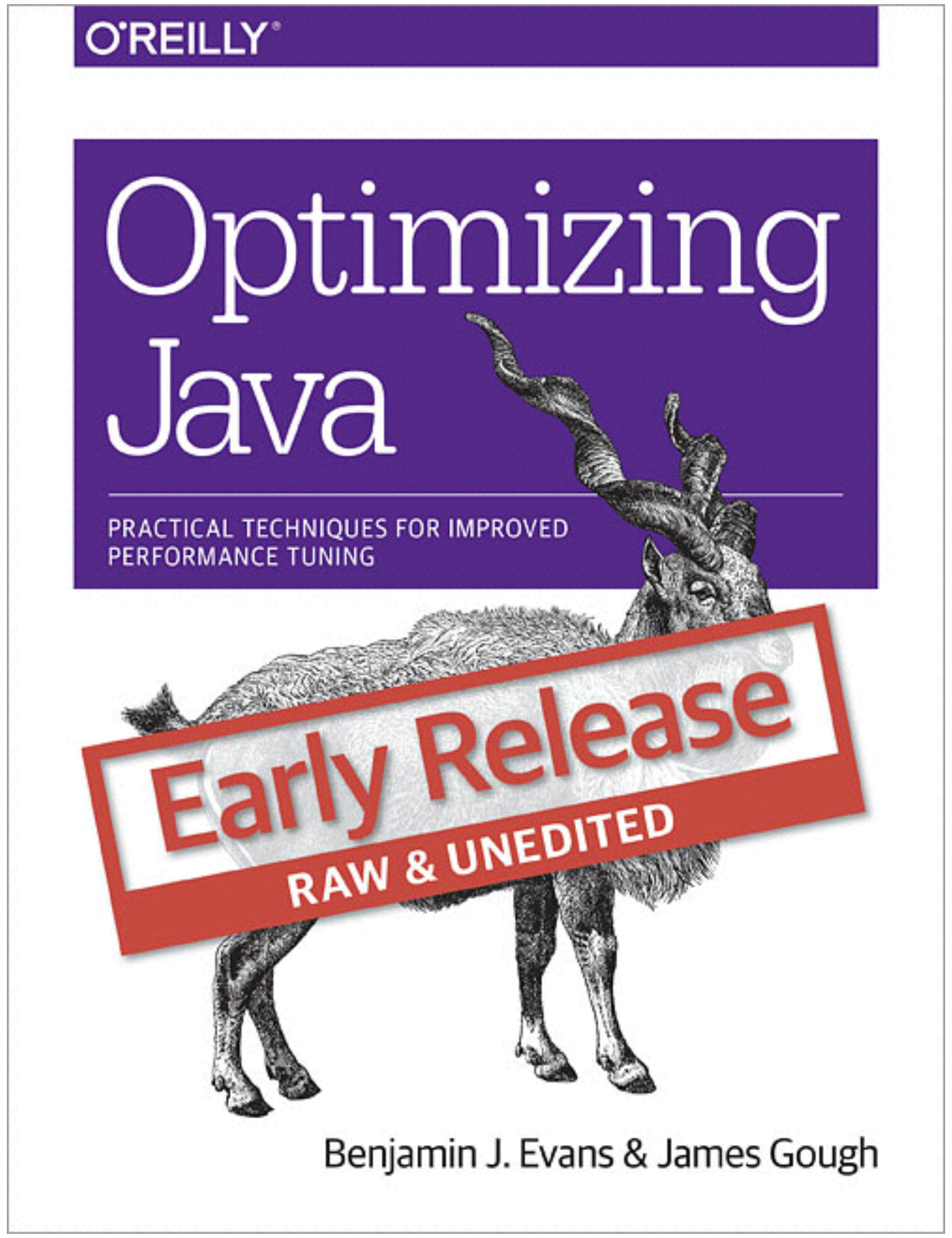
Email - ben@jclarity.com, jpgough@gmail.com

Books:

Java in a Nutshell (6th Edition) - O'Reilly

The Well-Grounded Java Developer - Manning

Forthcoming: Optimizing Java


O'REILLY
Optimizing Java
PRACTICAL TECHNIQUES FOR IMPROVED PERFORMANCE TUNING
Early Release
RAW & UNEDITED
Benjamin J. Evans & James Gough