Overhead Comparison of OpenTelemetry, inspectIT and Kieker

David Georg Reichelt[1] Stefan Kühne[1]
Wilhelm Hasselbring[2]

[1]Universität Leipzig, Universitätsrechenzentrum, Forschung und Entwicklung

[2]Christian-Albrechts-Universität zu Kiel, Software Engineering Group

9th of November 2021

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Why Overhead Comparison?

- Performance measurement creates overhead
  - Instrumentation
  - Measurement
  - Serialization

  $\Rightarrow$ Should be as low as possible

  $\Rightarrow$ Replicable measurement provided by MooBench

- Overhead is relevant
  - ... for comparison of monitoring tools

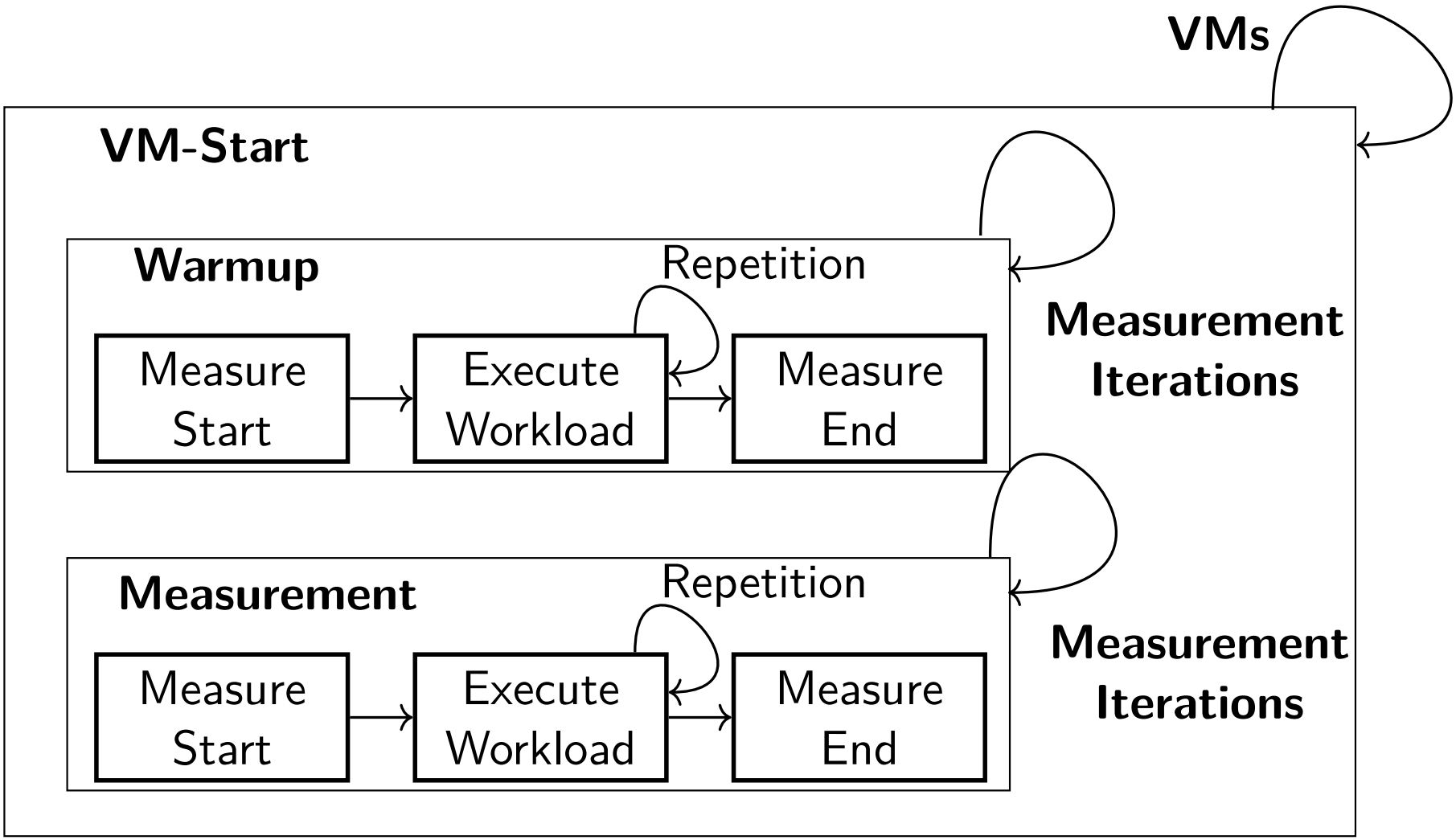
    $\Rightarrow$ this work
  - ... for further reduction of monitoring overhead for continuous root cause analysis in Peass

    $\Rightarrow$ ongoing research

Outline

MooBench

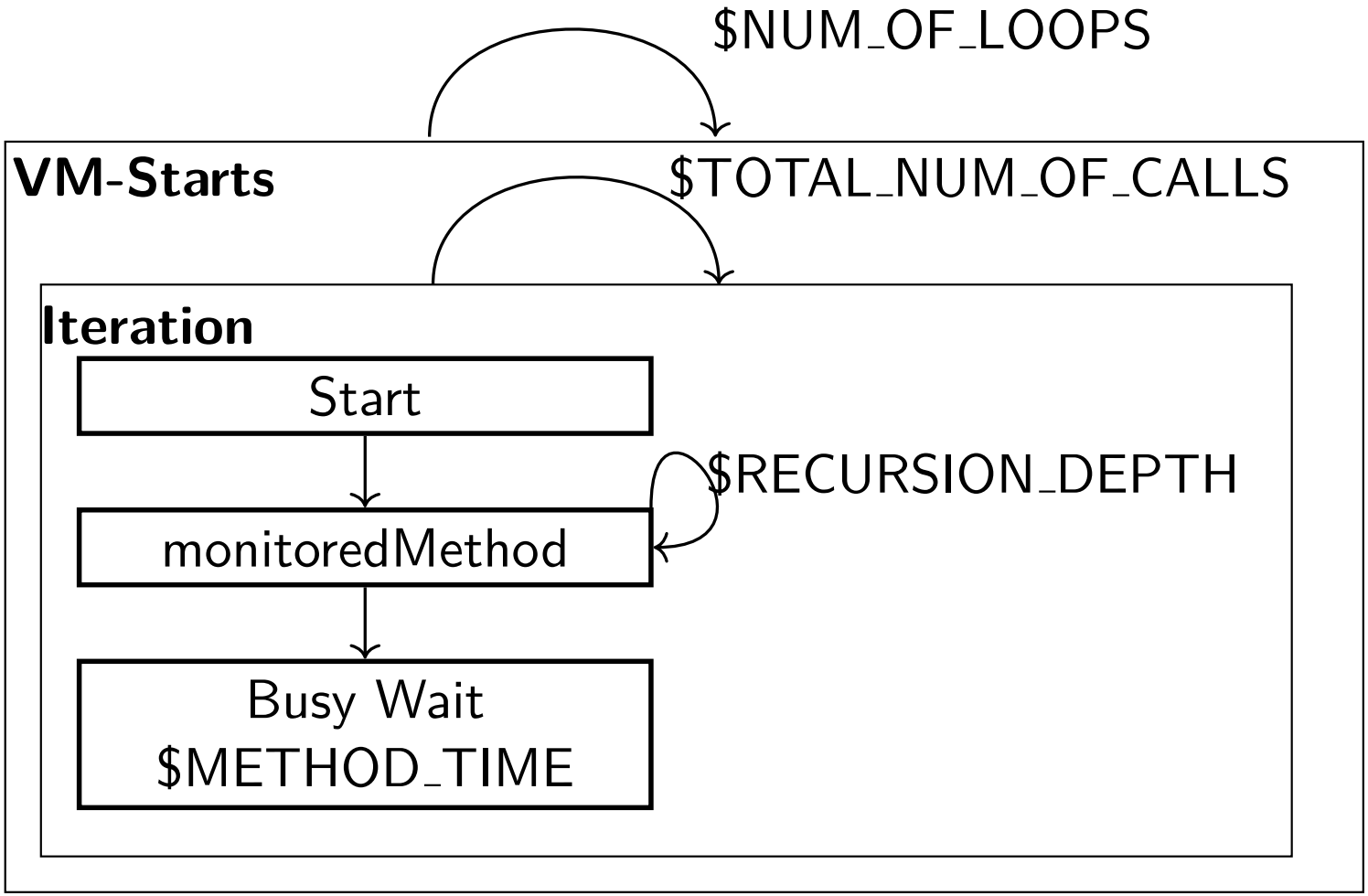
Performance Measurement in JVM



Performance Measurement in JVM

- Non-deterministic effects influence performance
  - Just-in-Time-Compilation
  - Garbage collection
  - Memory fragmentation
  - ...
- Measurement process
  - Warmup iterations
  - Measurement iterations
  - Repetition inside VMs
  - Analysis of values by statistical test, e.g. T-Test

MooBench (Measurement Process)



- Additional parametrisation by
  - SLEEP_TIME (Sleep time between VM starts, so system can cooldown)

MooBench (Variants)

- Baseline
- Regular instrumentation
- Deactivated monitoring
- Different monitoring configurations (e.g. different serialization)

Prior Work on MooBench

- Continuous measurement (Waller, Ehmke and Hasselbring, 2015)
- Testing of replicability (Knoche and Eichelberger, 2017; Knoche and Eichelberger, 2018)
- Effects of multithreading (Waller and Hasselbring, 2015)

Monitoring Frameworks

OpenTelemetry

- “Ubiquitous“ telemetry ⇒ Support of a many languages
- Supports variety of frameworks itself
- Different exporters (Zipkin, Prometheus, Jaeger)

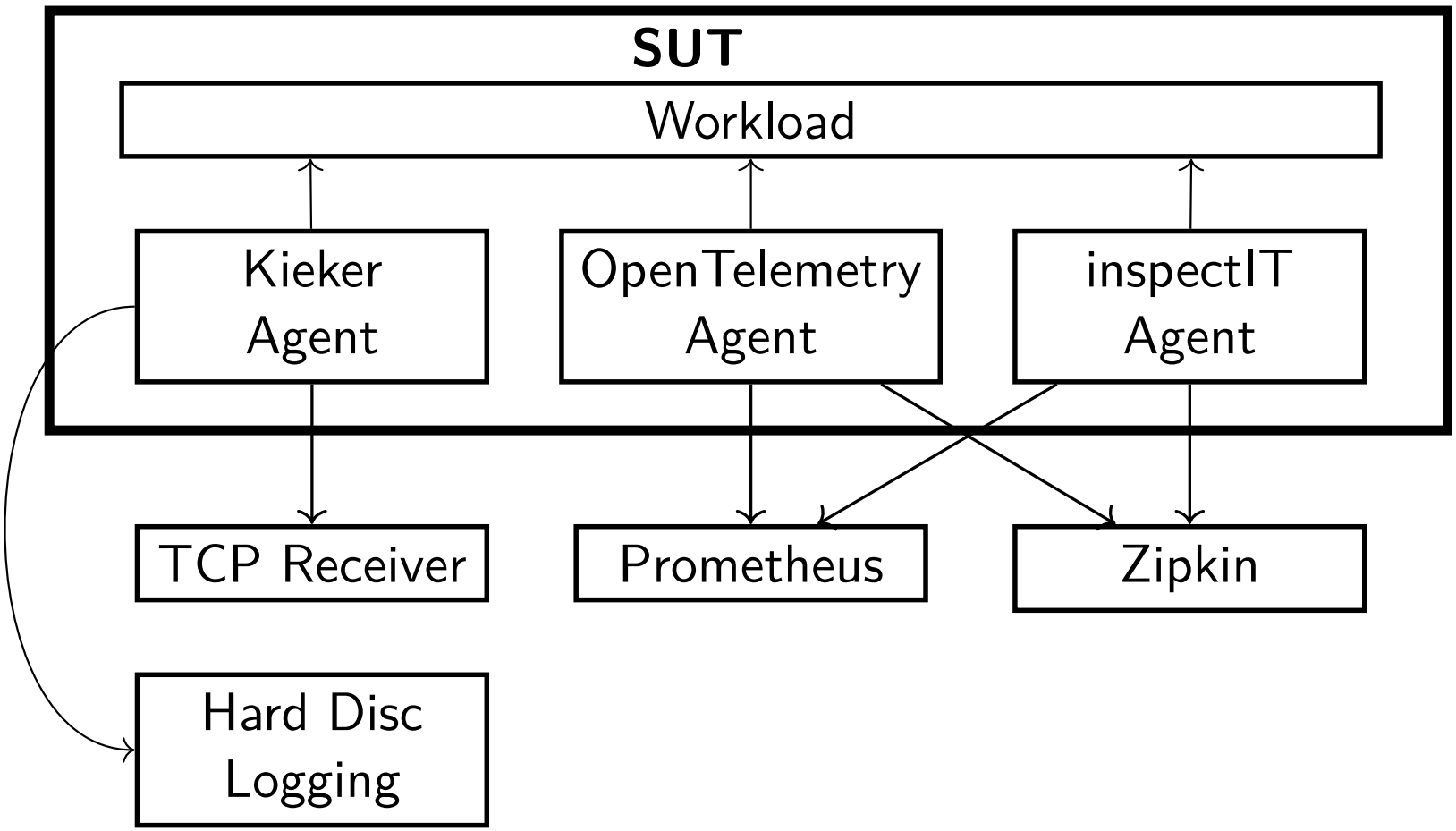
- Instrumentation through javaagent
- Configuration through command line, yaml file, ...

inspectIT

- “Zero-configuration“ Java agent for performance collection
- Supports variety of frameworks by usage of OpenCensus
- Different exporters (Zipkin, Prometheus, Jaeger)

- Configuration (through command line, yaml file, ...)
  - Scopes define measured methods
  - Rules define measurement metrics
  - Actions define processing on extracted data

Measurement with MooBench



Overhead Comparison

Setup

- OpenJDK 11.0.11
- Hardware
  - For replicability to older data: Raspberry Pi 4
  - Current desktop: i7-4770 CPU @ 3.40GHz with 16 GB RAM, running Ubuntu 20.04
- Workload sizes
  - Call tree depth 10 (default) for all configurations
  - Exponential growing call tree depth for TCP export

Call Tree Depth 10 (Kieker)

Variant	Raspberry Pi		i7-4770	
	95 % CI	σ	95 % CI	σ
Baseline	[1.5;1.5]	0.1	[0.057;0.058]	0.026
Kieker				
Deactivated Probe	[4.1;4.1]	7.5	[0.4;0.4]	7.1
DumpWriter	[51.9;52.0]	14.6	[8.5;8.5]	12.2
Logging (Text)	[743.3;799.4]	14315.8	[103.0;103.3]	56.4
Logging (Binary)	[59.8;87.8]	7149.4	[3.4;3.4]	15.8
TCP	[45.6;45.7]	14.6	[4.6;4.7]	10.4

Tabelle: Measurement Results for Kieker (in μs)

Call Tree Depth 10 (Kieker)

- Also deactivated probe has noticable overhead
- Regular text logging is very inefficient
- Fastest configuration for local processing: Binary Logging

Call Tree Depth 10 (default)

Variant	OpenTelemetry		
	Deactivated Probe	Zipkin	Prometheus
Pi 4 CI	[26.8;26.9]	[53.4;53.6]	[44.4;44.5]
σ	20.4	46.7	25.2
i7-4770 CI	[4.9;5.0]	[6.8;6.9]	[6.9;6.9]
σ	4.1	8.5	4.9
	inspectIT		
Pi 4 CI	[9.9;9.9]	[97.2;97.8]	[32.3;32.4]
σ	10.5	149.6	16.6
i7-4770 CI	[1.3;1.4]	[10.9;11.2]	[4.0;4.0]
σ	8.2	57.4	4.1

Tabelle: Measurement Results for OpenTelemetry and inspectIT (in μs)

Call Tree Depth 10 (default)

- OpenTelemetry has lower overhead for Zipkin trace (spans) export
- inspectIT has lower overhead for metrics export and deactivated probe
- Deactivated pobe overheads are significantly higher than in Kieker

Growing Call Tree Depth

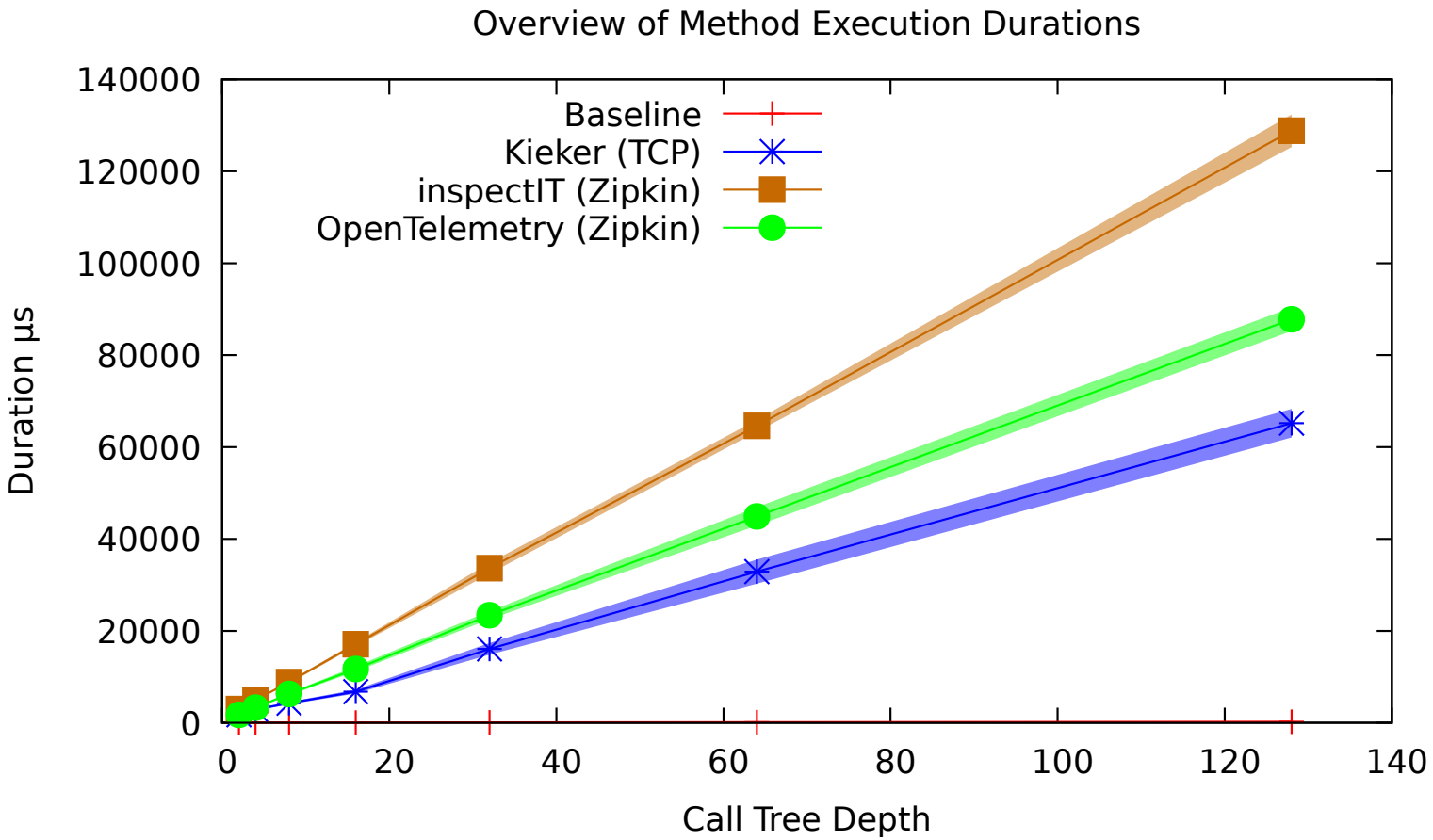


Abbildung: Overhead evolution with growing call tree depth

Growing Call Tree Depth

- Different writer configurations are not comparable
- Kieker currently does not support aggregated metrics export
- Only full trace export to Zipkin / TCP export comparable

Summary

Summary

- Monitoring overhead needs to be as low as possible
- MooBench compares Monitoring overhead of different frameworks and monitoring configurations

- MooBench was extended to support OpenTelemetry and inspectIT
- Measurement of traces by OpenTelemetry and inspectIT is slower than with Kieker

Outlook

- Benchmarking with more complex tree structure
- Comparison of overhead for different frameworks, e.g. Jersey
- Comparison of accuracy (How well does root cause analysis algorithm X perform?)