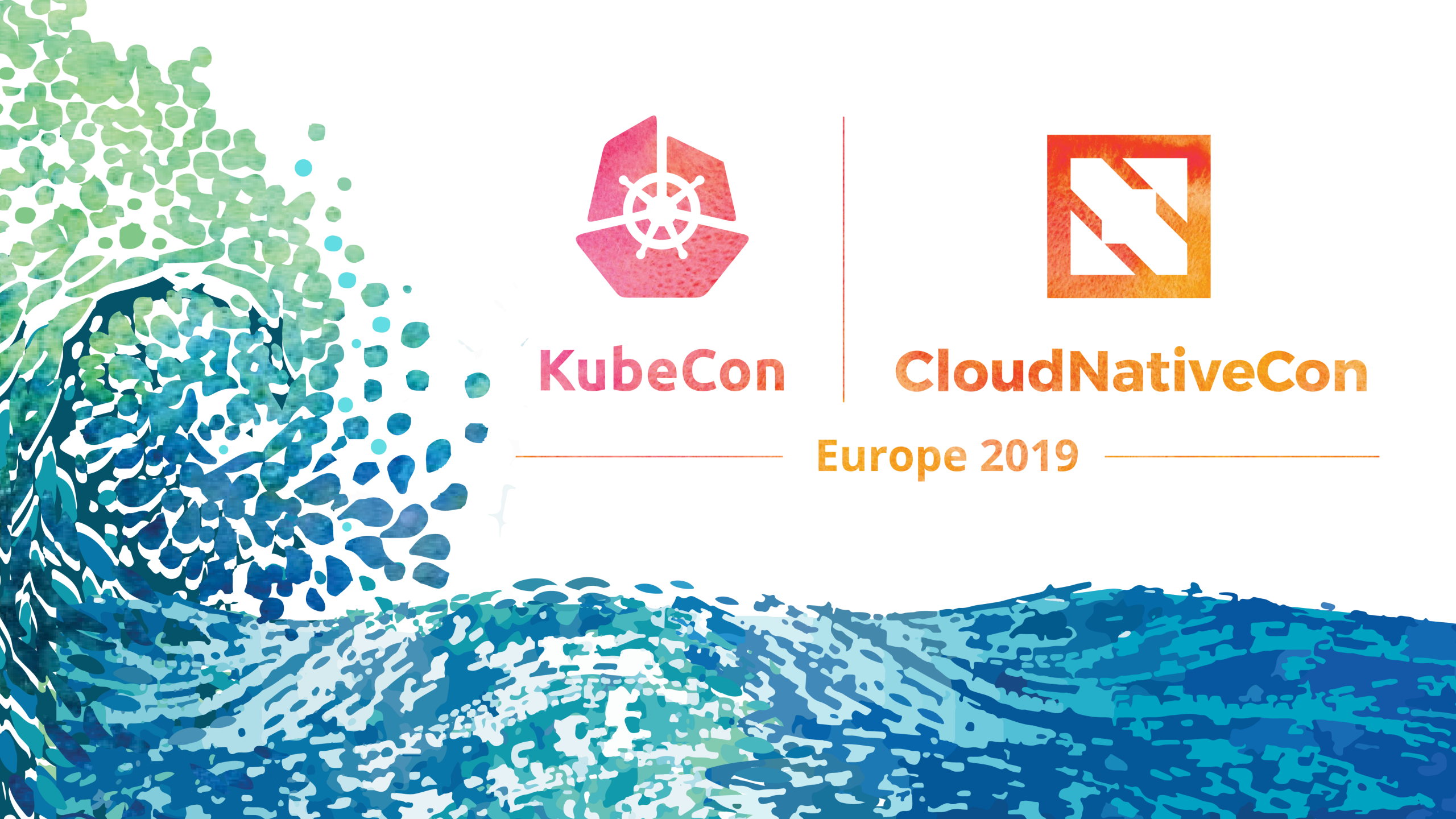

KubeCon
CloudNativeCon
Europe 2019

KubeCon | CloudNativeCon Europe 2019

What's the Performance Overhead?

Answering the Biggest Question in Tracing

Gabriela Soria

Former Outreachy intern for CNCF

About me – Outreachy

Outreachy provides three-month paid internships to work in Free and Open Source Software (FOSS) for under-represented people.

OPENTRACING

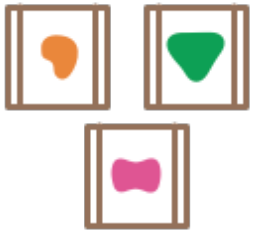
Agenda

- Introduction
- Benchmark tests
  - Scope and considerations
  - JMH Benchmark application example
  - Results
  - Conclusions
- Next steps
- Lessons learned

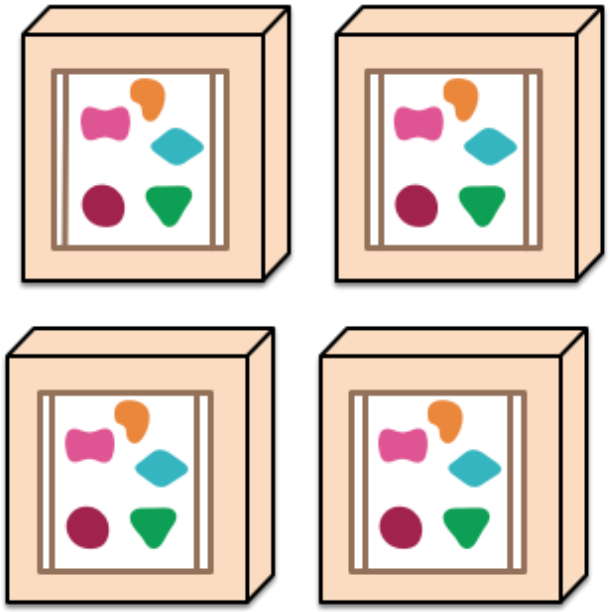
Introduction – Microservices

A monolithic application puts all its functionality into a single process...

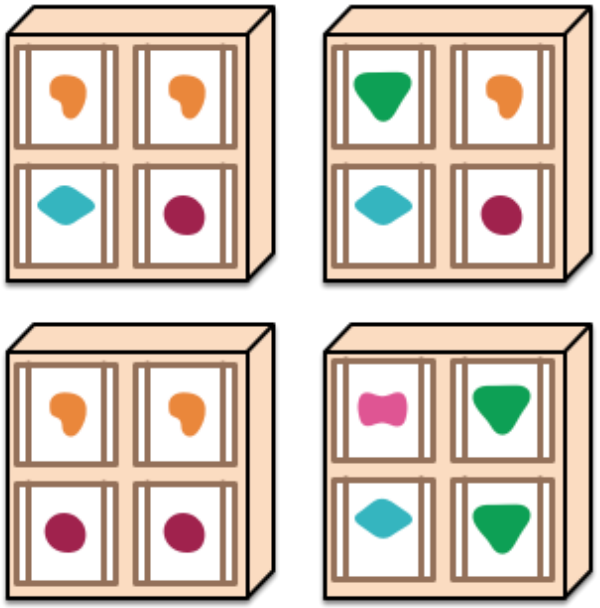
A microservices architecture puts each element of functionality into a separate service...



... and scales by replicating the monolith on multiple servers



... and scales by distributing these services across servers, replicating as needed.



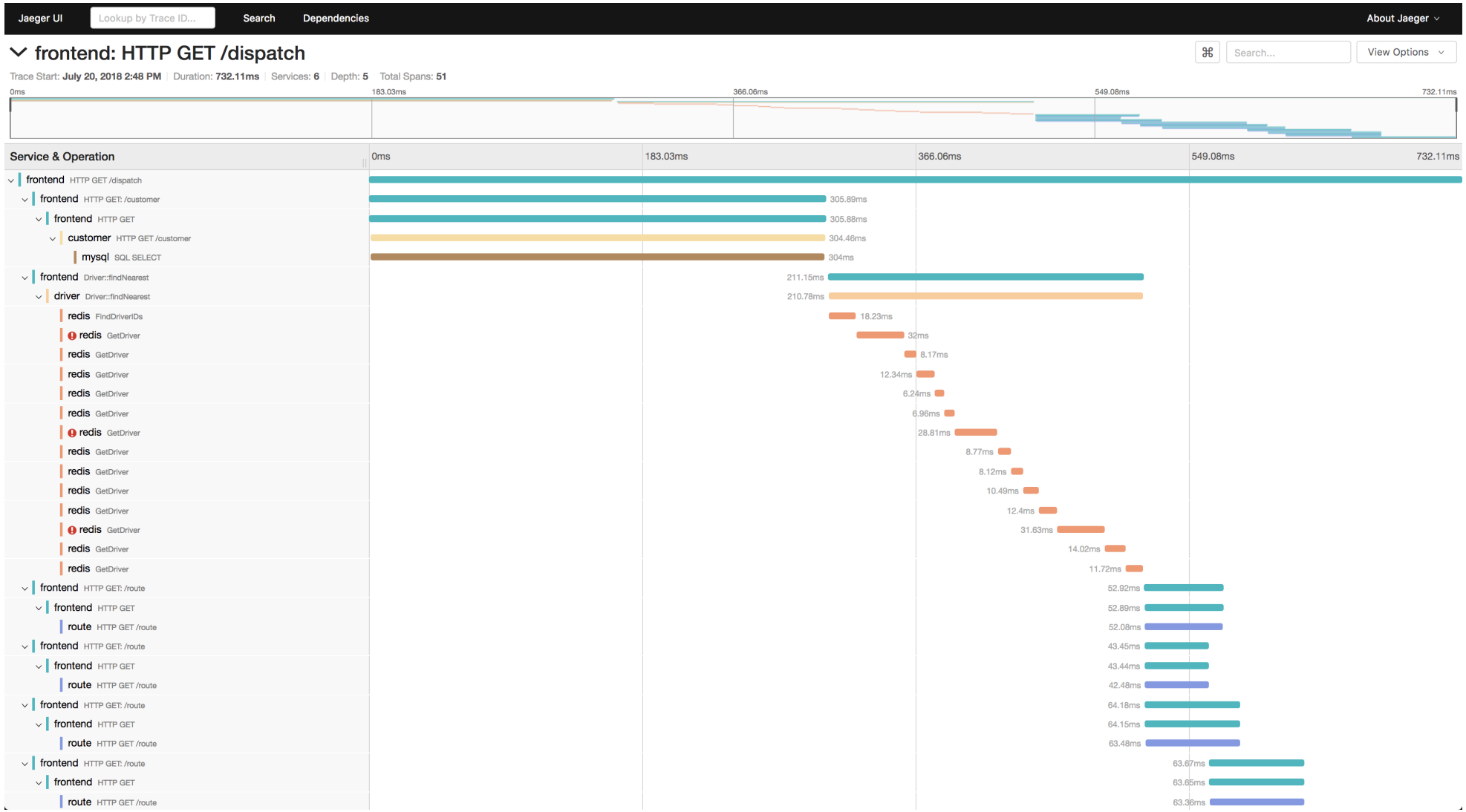
Source: https://martinfowler.com/articles/microservices.html

Distributed tracing – Opentracing

Distributed tracing profiles and monitor applications, especially those built using a microservices architecture.

OpenTracing is working towards creating more standardized APIs and instrumentation for distributed tracing.

Jaeger



@gabrielasoriag

Source: https://www.jaegertracing.io/docs/1.11/

Scope and considerations

- We used Java Microbenchmark Harness (JMH)
- We used JMH Visualizer to present the results
- Tested Opentracing **Java** API only
- We tried to avoid network calls (in the scenarios that was possible)

JMH Benchmark Application Example

- Create a JMH application:

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

JMH Benchmark Application Example


```
public class BenchmarkPetclinicSampleTime extends BenchmarkPetclinicBase {

    @Benchmark
    @BenchmarkMode(Mode.SampleTime)
    public Owner noInstrumentation(StateVariablesNoInstrumentation state) {
        return findPetOwnerById(state);
    }

    @Benchmark
    @BenchmarkMode(Mode.SampleTime)
    public Owner noopTracer(StateVariablesNoopTracer state) {
        return findPetOwnerById(state);
    }

    @Benchmark
    @BenchmarkMode(Mode.SampleTime)
    public Owner jaegerTracer(StateVariablesJaeger state) {
        return findPetOwnerById(state);
    }
....
```

JMH Benchmark Application Example


```
public class BenchmarkPetclinicBase {

  public Owner findPetOwnerById(StateVariables state) {
    return state.petcontroller.findOwner(1);
  }
  @State(Scope.Benchmark)
  public static class StateVariables {
    public PetController petcontroller;
    public ConfigurableApplicationContext c;

    @TearDown(Level.Iteration)
    public void shutdownContext() {
      c.close();
    }

    public void initApplication() {
      c = SpringApplication.run(PetClinicApplication.class);
      petcontroller = c.getBean(PetController.class);
    }
  }
```

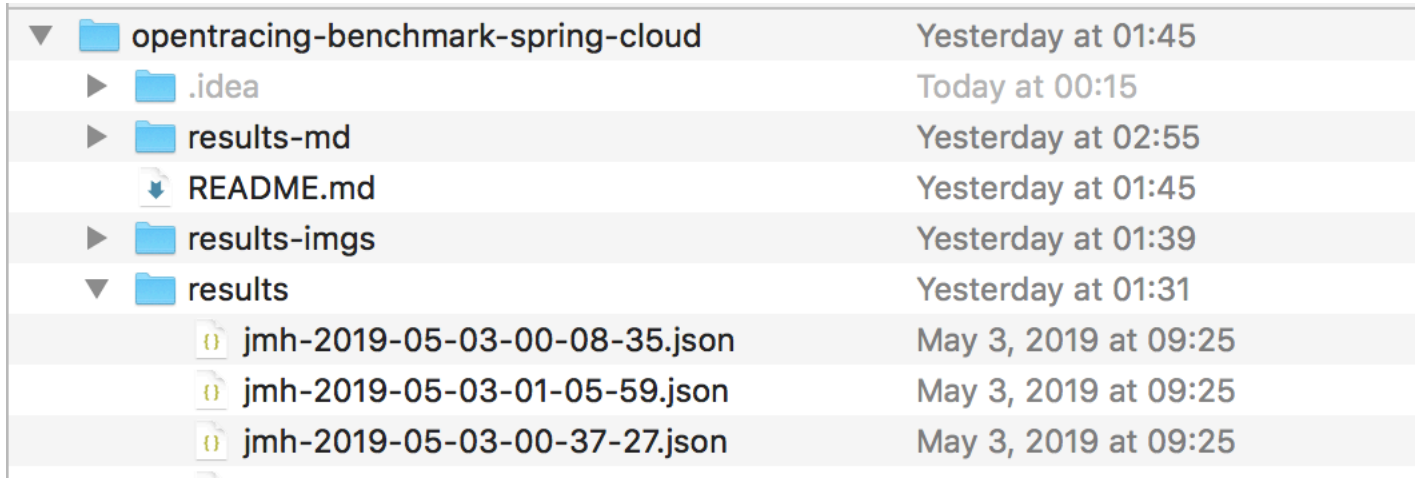
JMH Benchmark Application Example


```
public static class StateVariablesNoInstrumentation extends StateVariables {
    @Setup(Level.Iteration)
    public void doSetup() {
        System.setProperty("tracerresolver.disabled", Boolean.TRUE.toString());
        initApplication();
    }
}

public static class StateVariablesJaeger extends StateVariables {
    @Setup(Level.Iteration)
    public void doSetup() {

        System.setProperty(AbstractEnvironment.ACTIVE_PROFILES_PROPERTY_NAME,
TracerImplementation.JAEGERTRACER);
        initApplication();
    }
}
```

JMH Benchmark Application Example

- Run the tests:

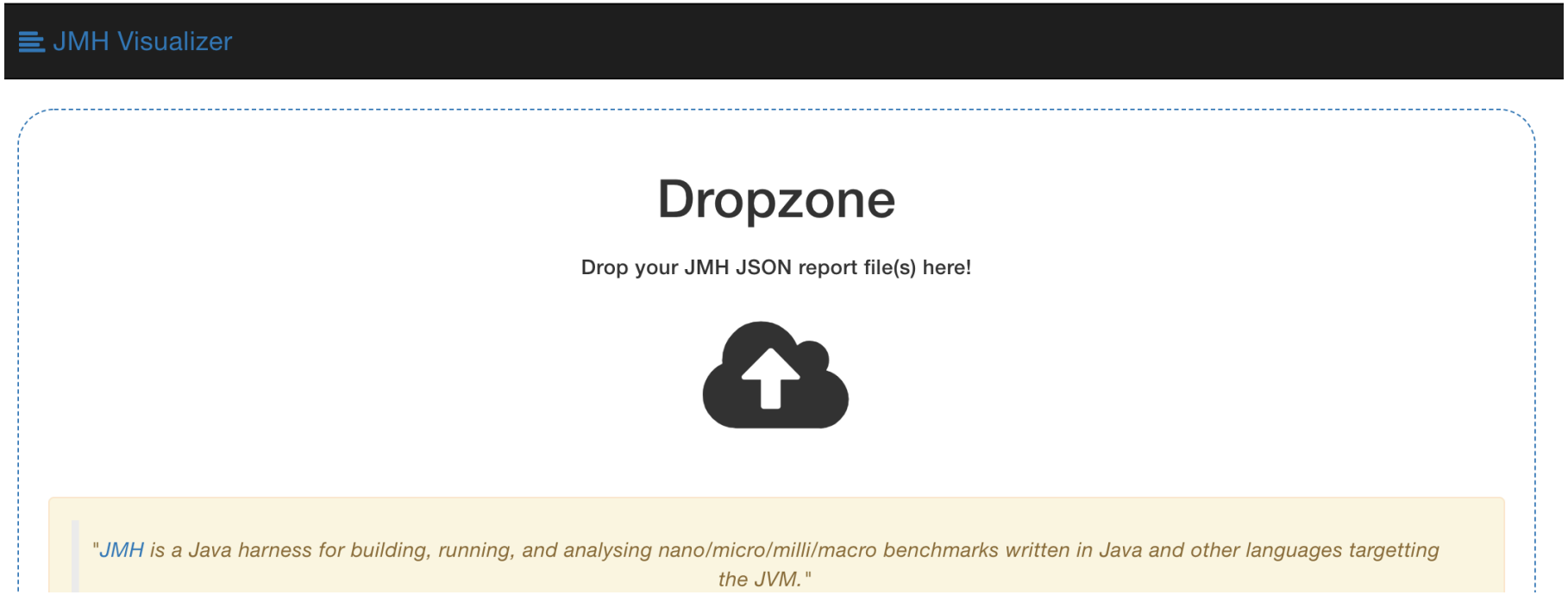
```
mvn clean install
java -jar target/benchmarks.jar
```



JMH Benchmark Application Example

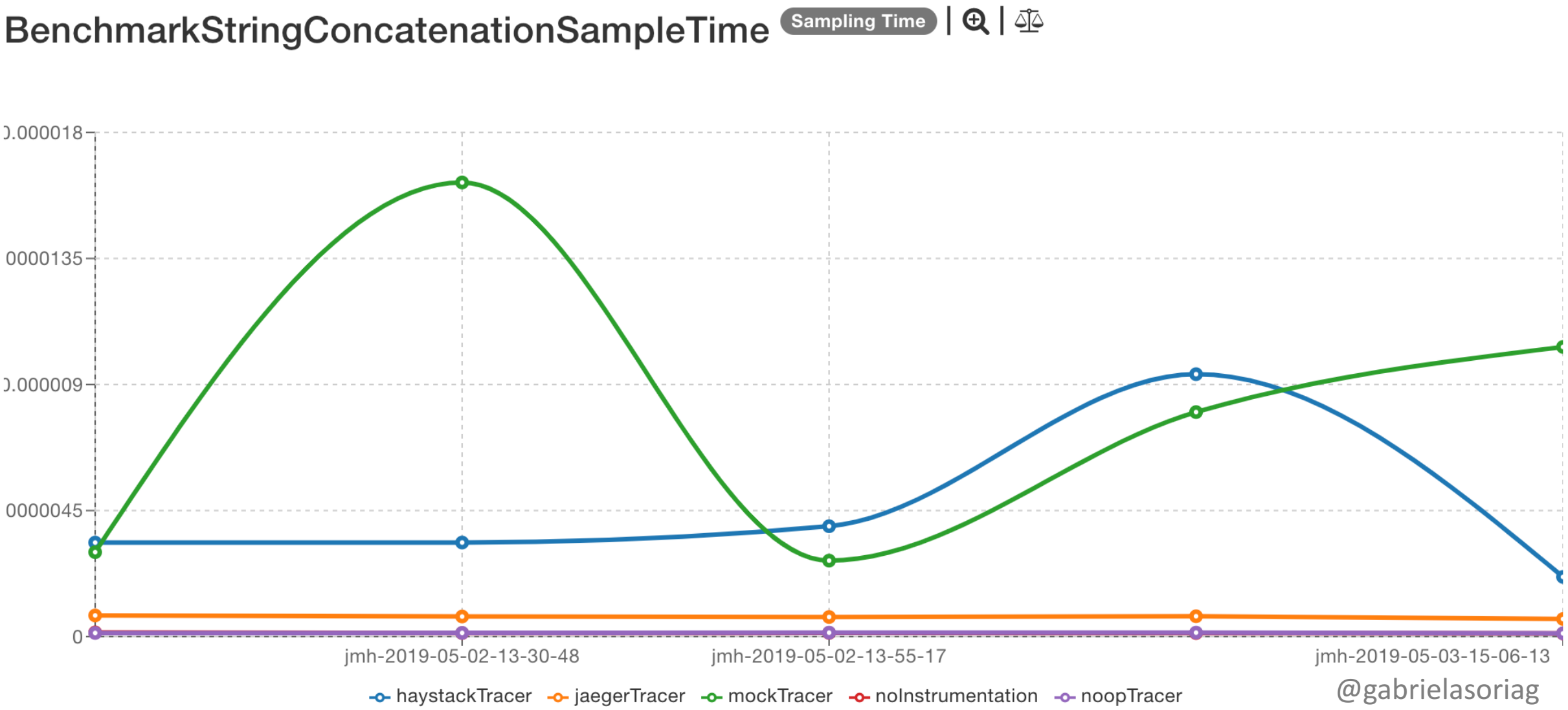
- Visualize the results

  http://jmh.morethan.io/



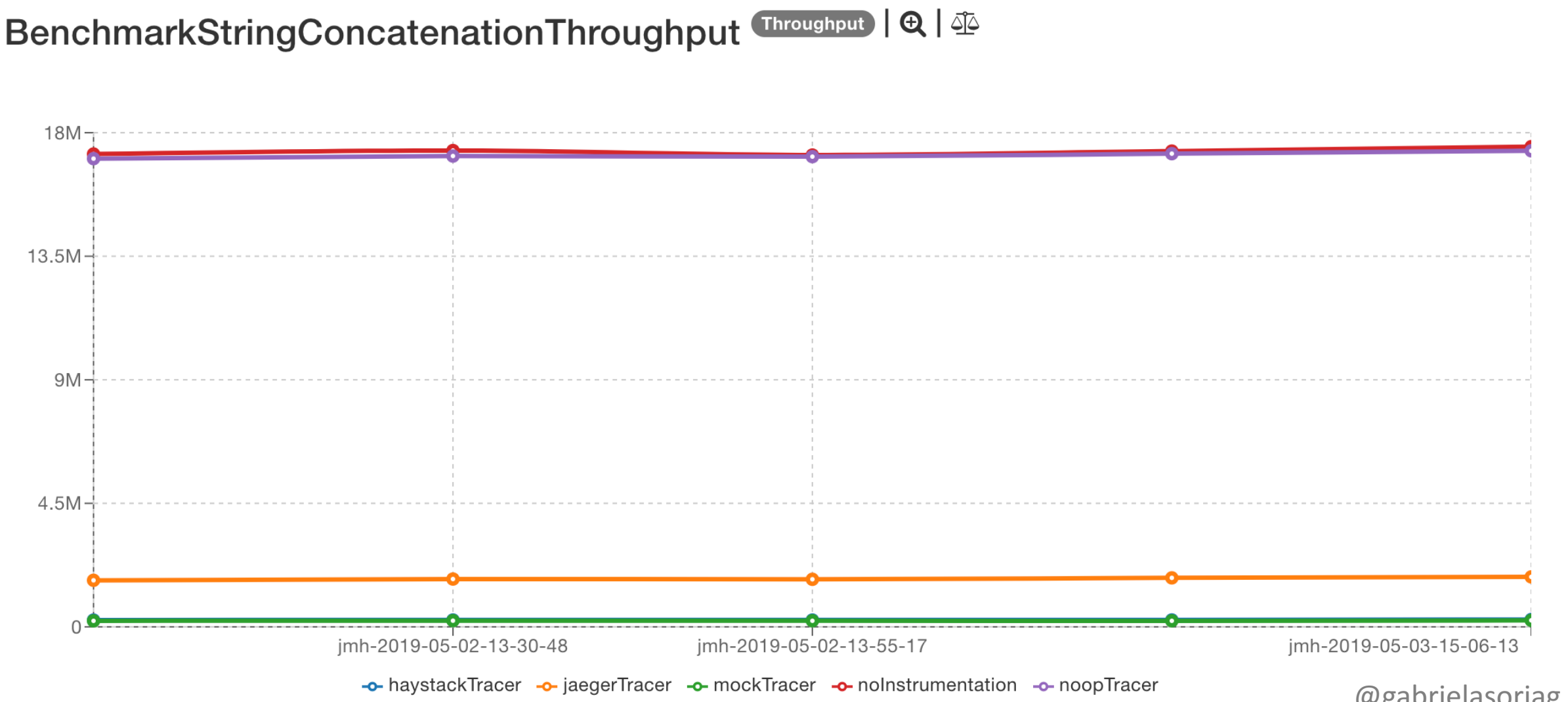
Results – Simple Java (Sample time)

KubeCon CloudNativeCon Europe 2019



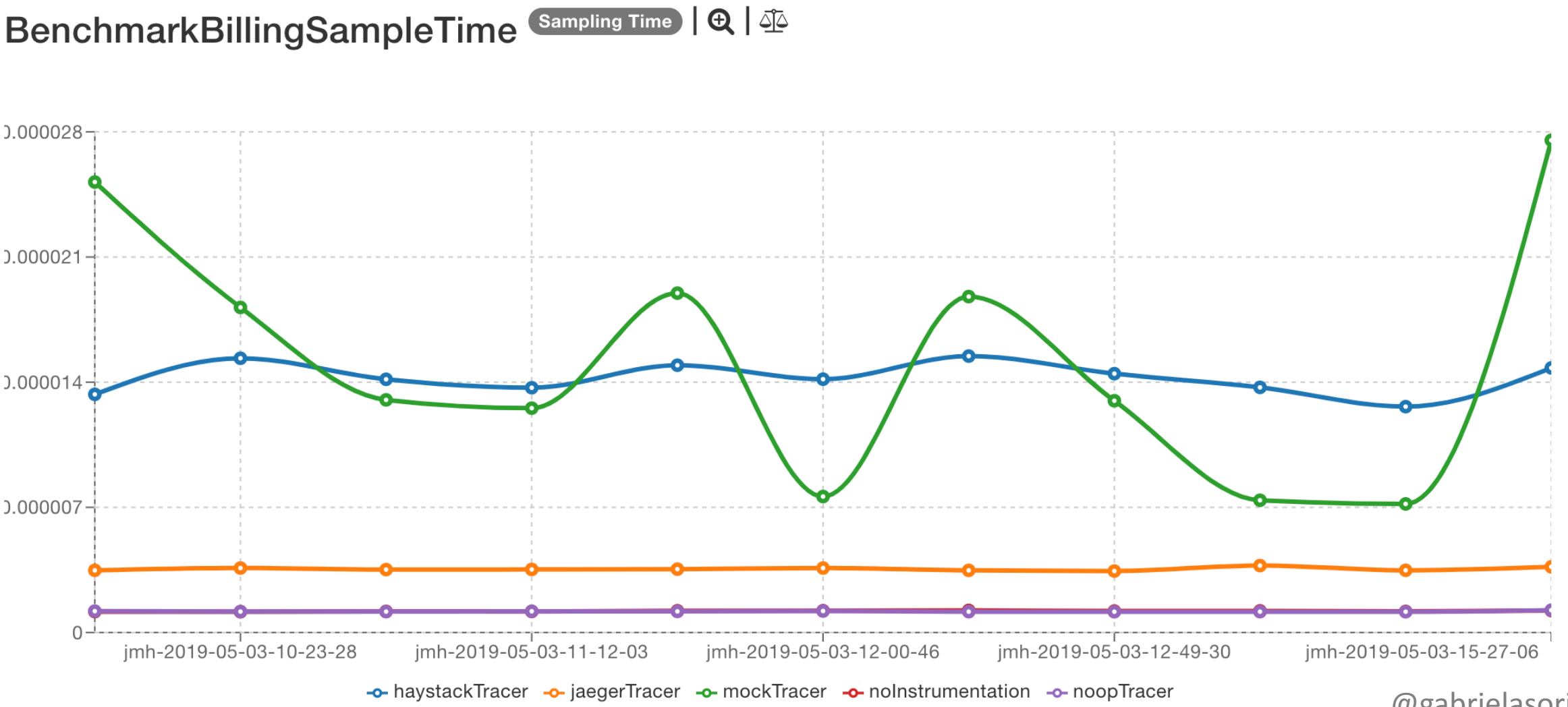
@gabrielasoriag

Results – Simple Java (Throughput)



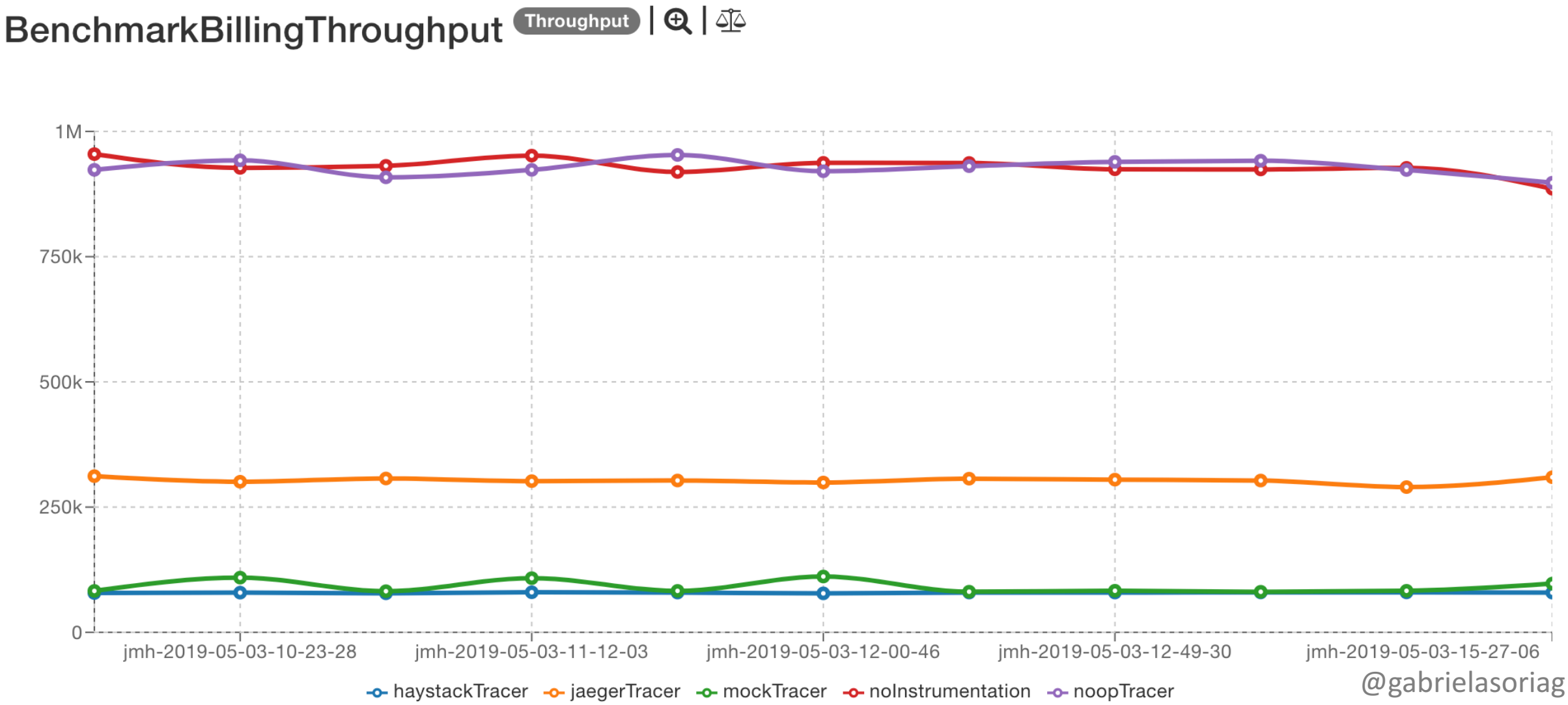
Results – Spring boot (Sample time)

KubeCon CloudNativeCon Europe 2019

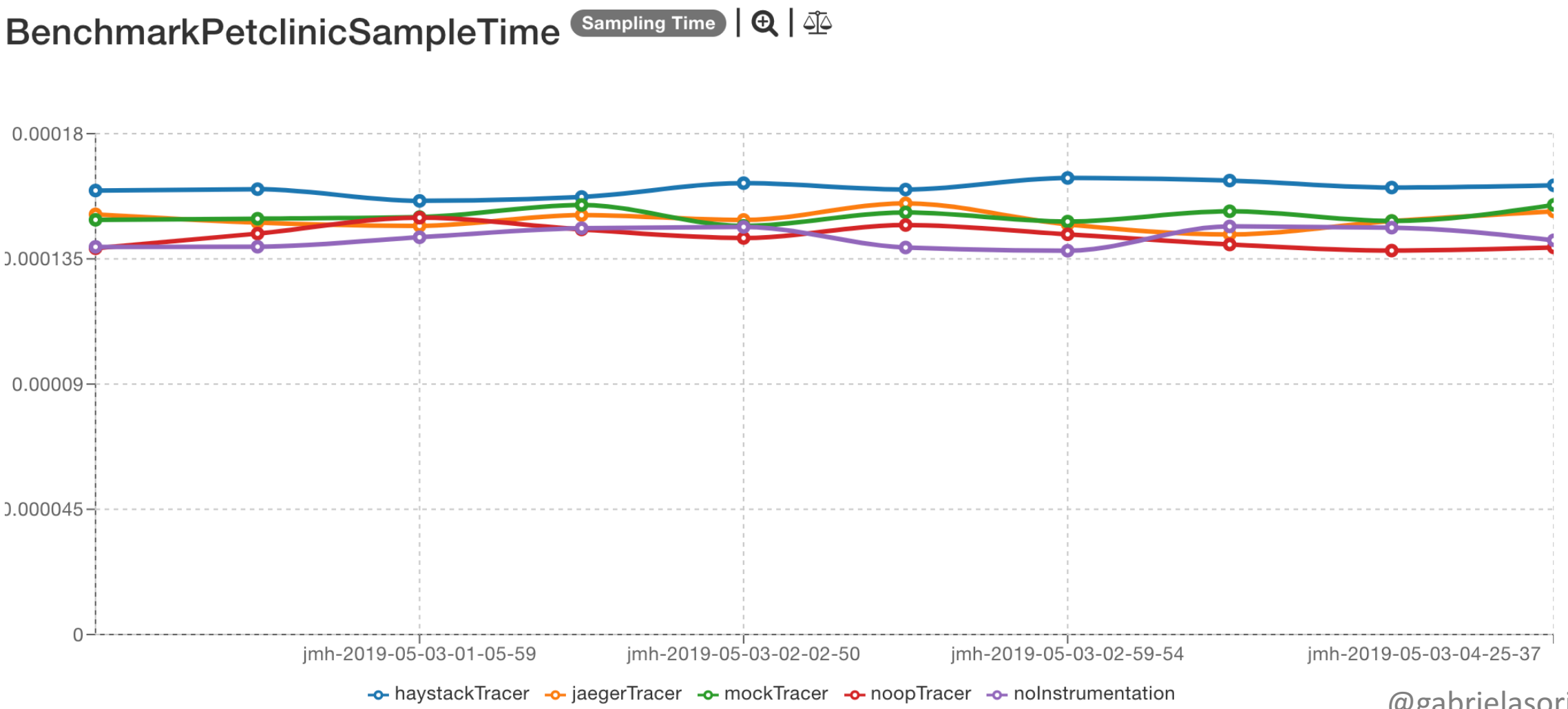


@gabrielasoriag

Results – Spring boot (Throughput)

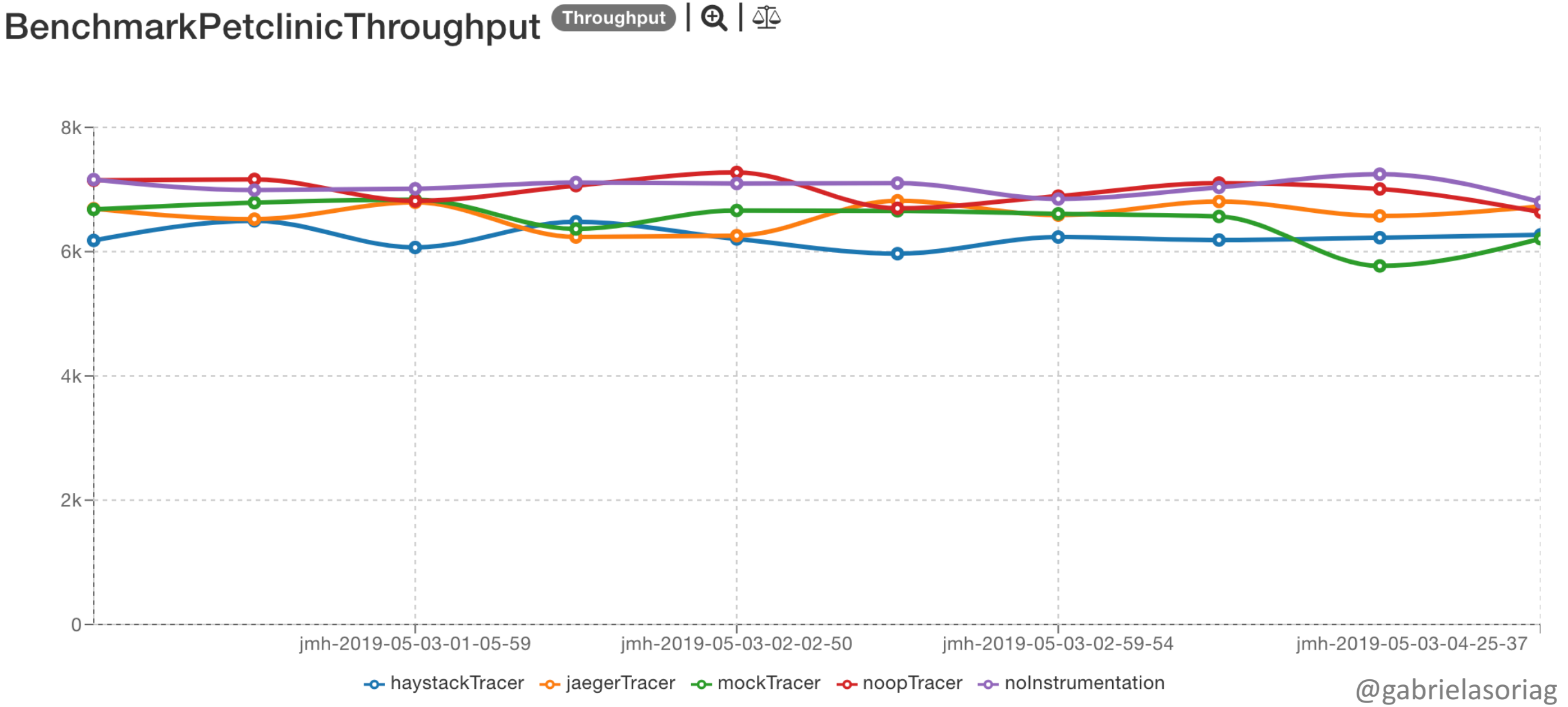


Results – Spring Cloud (Sample time)



Results – Spring Cloud (Throughput)

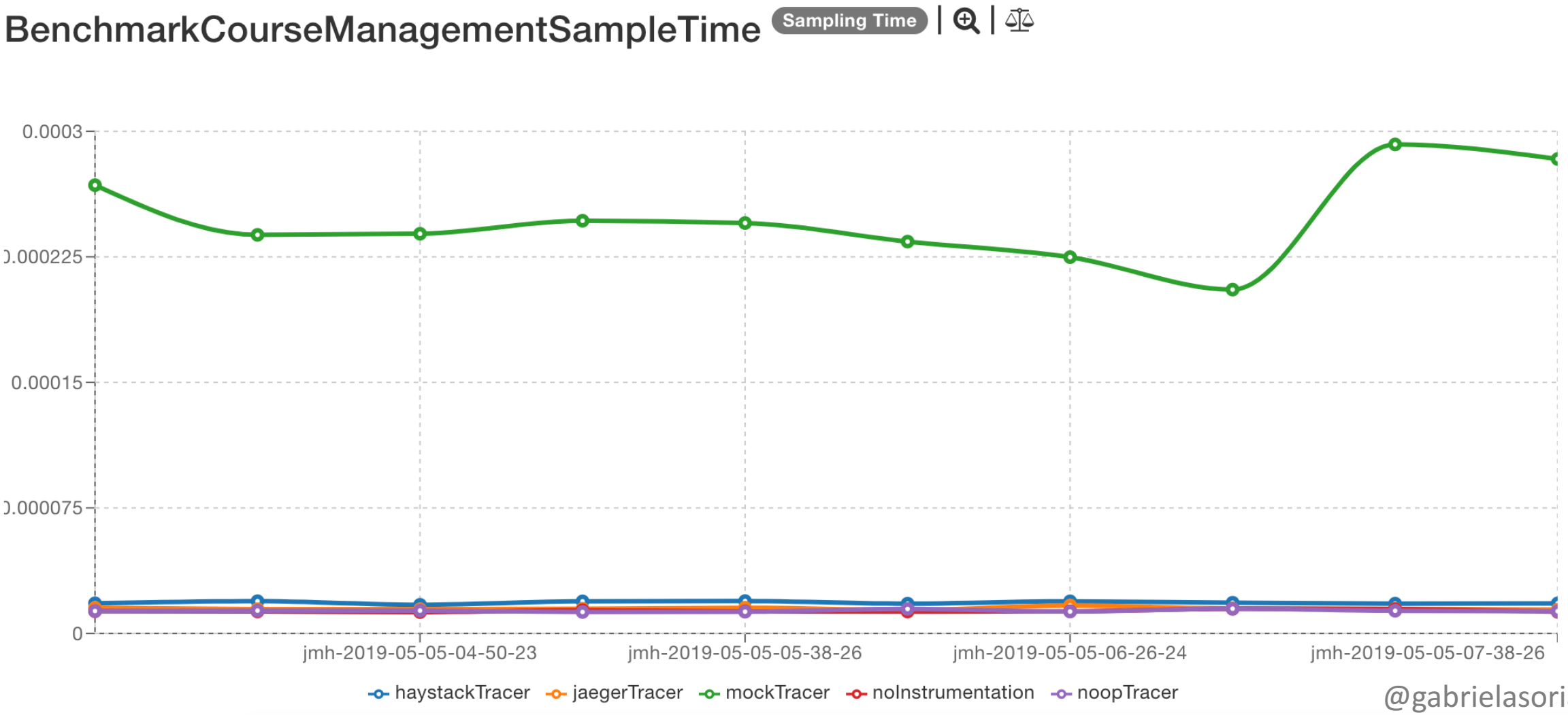
KubeCon CloudNativeCon Europe 2019



@gabrielasoriag

Results – JDBC (Sample time)

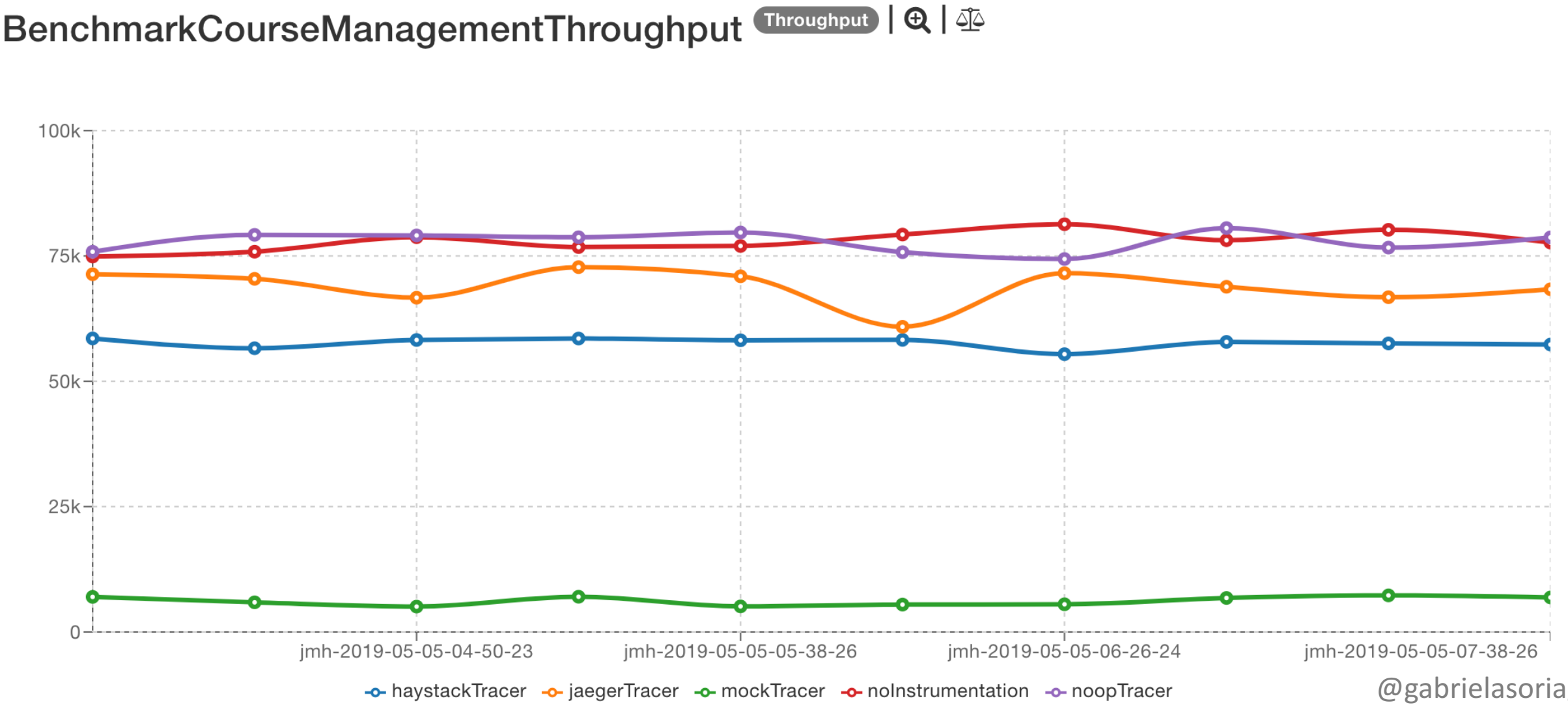
KubeCon CloudNativeCon Europe 2019



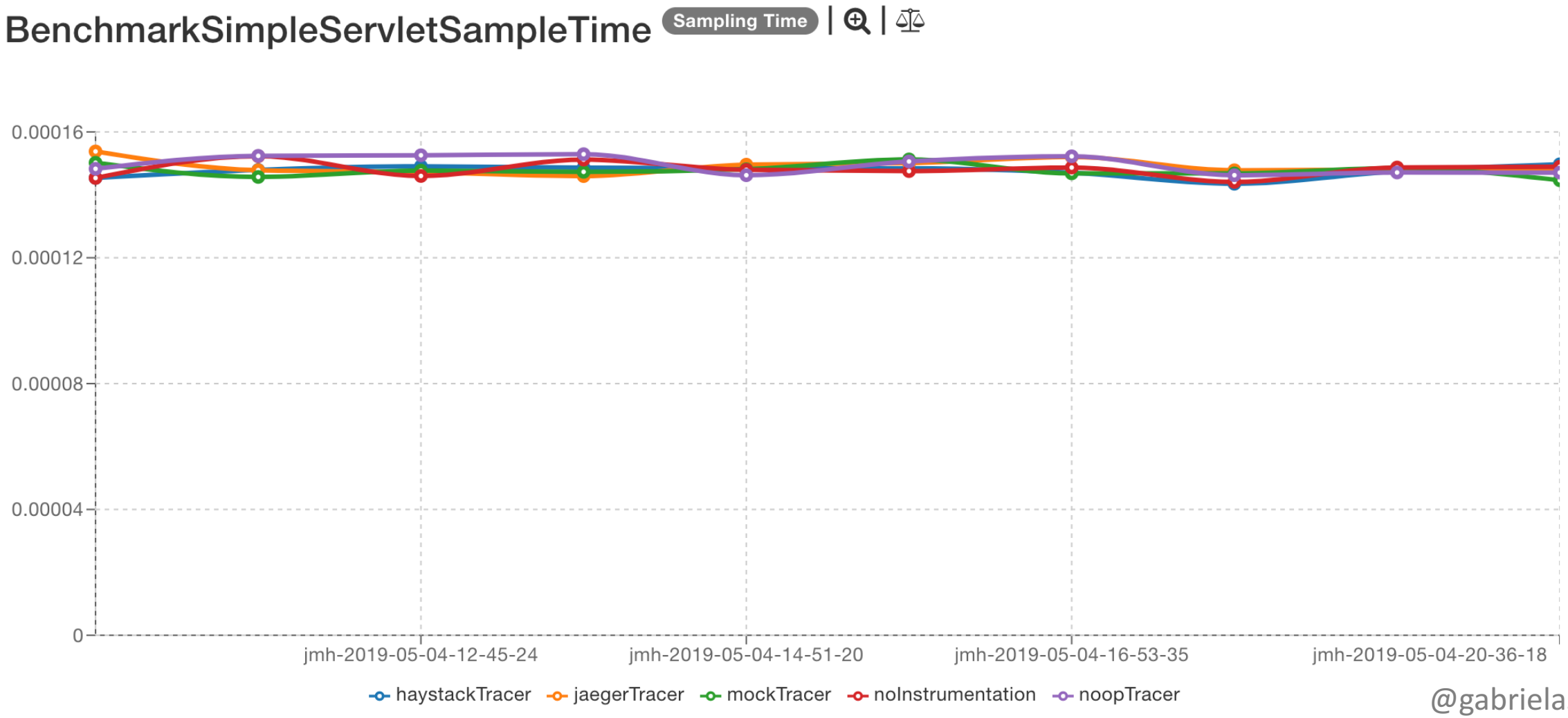
@gabrielasoriag

Results – JDBC (Throughput)

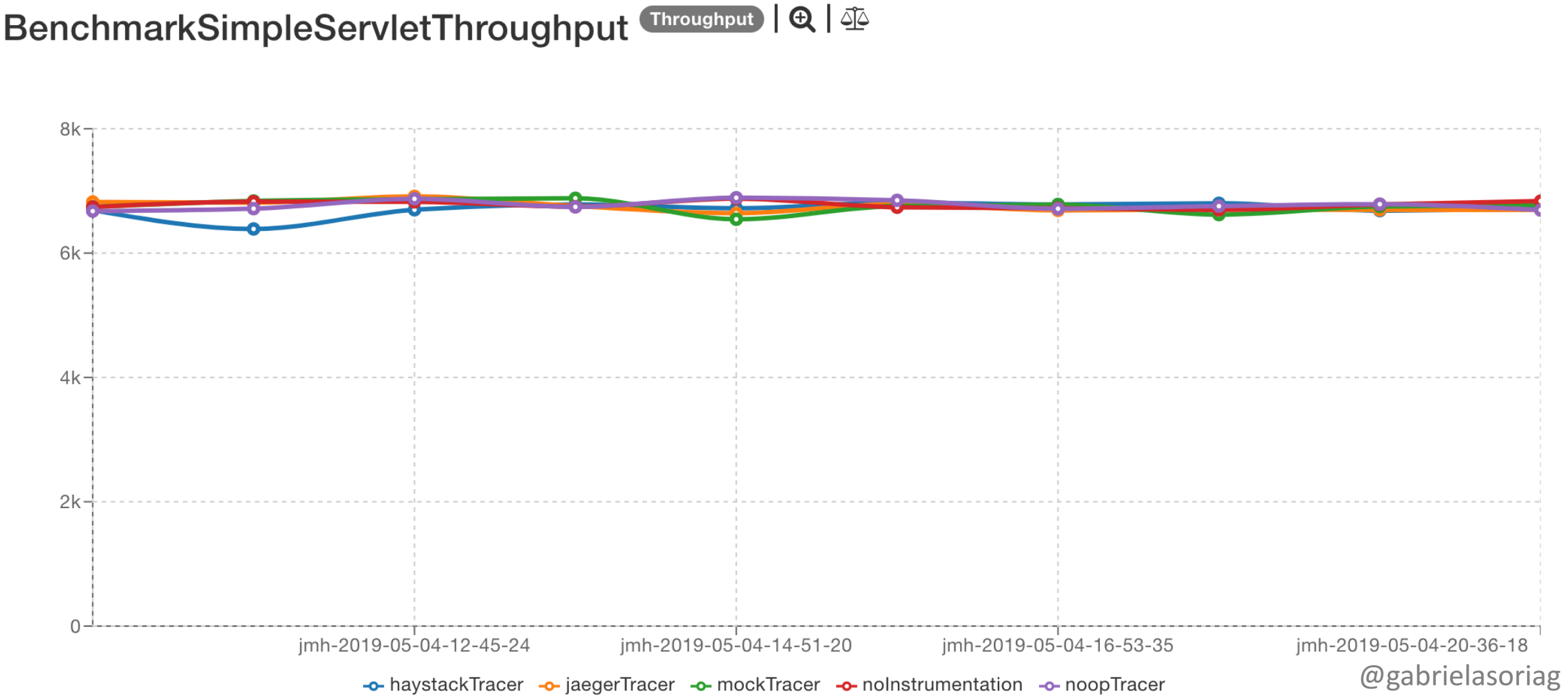
KubeCon CloudNativeCon Europe 2019



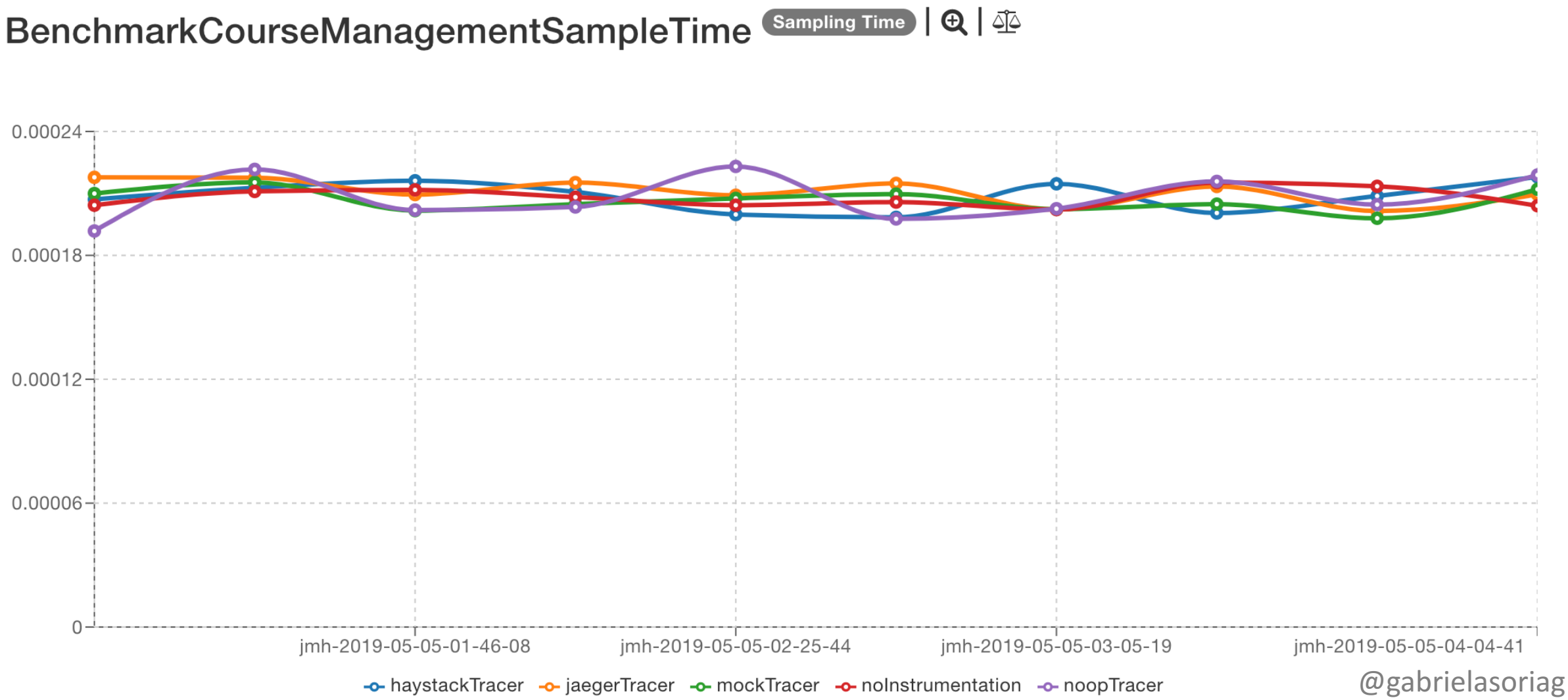
Results – Servlet Filter (Sample time)



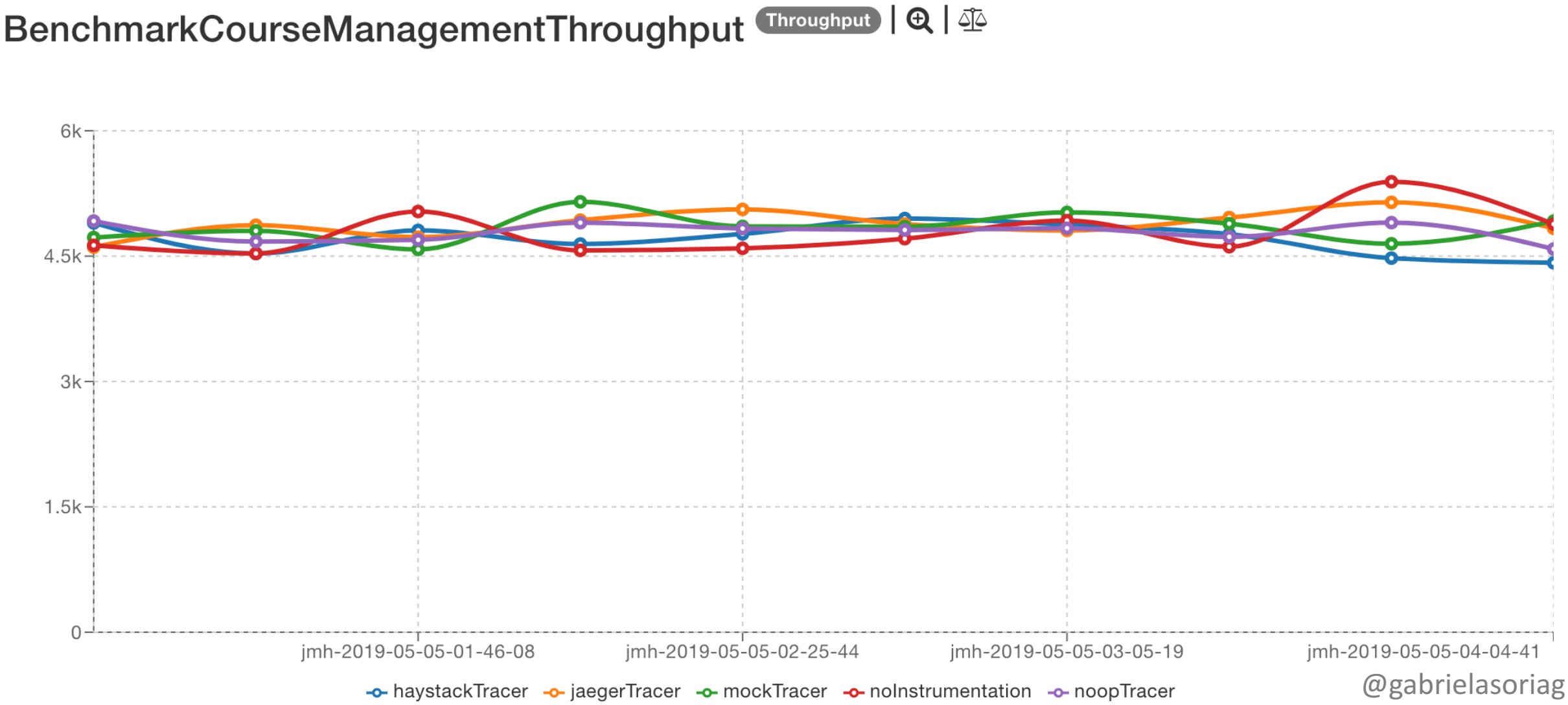
Results – Servlet Filter (Throughput)



Results – JAX-RS (Sample time)



Results – JAX-RS (Throughput)



Conclusions

- In simple java scenarios the throughput **decreases ~90%** and the sample time **increases ~440%**
- In scenarios that include calls through the framework (spring boot, spring cloud, JDBC), on average, the throughput **decreases 12%** and the sample time **increases 14%**.
- In the scenarios with **client calls through HTTP** (Servlet Filter, JAX-RS), the metrics show **no evidence of overhead**, as the deltas of throughput and sample time are not representative.

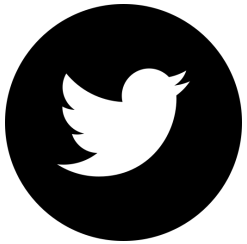
Next steps

- Benchmark tests for Jaeger in different scenarios:
  - gRPC vs. Thrift
  - Agent (UDP) vs. Collector (gRPC)
  - Different backend config
- Re-run the tests after merging OpenTracing and OpenCensus

Lessons learned

- Performance tests are tricky, the constant review of the code helps to improve the tests.
- You should ask yourself constantly if the benchmark test is measuring what you **really** want to measure.
- JMH
  - The method of the tests should return a value.
  - The garbage collector influences the results.
  - JMH performs dead code elimination (NoopTracer)

Keep in touch :)



@gabrielasoriag

Opentracing Java Benchmarks