COMMUNITY
THE ASF CONFERENCE
CODE

Boosting Tomcat Performance with Virtual Threads

Han Li
Apache Tomcat PMC

Introductions

Han Li - lihan@apache.org

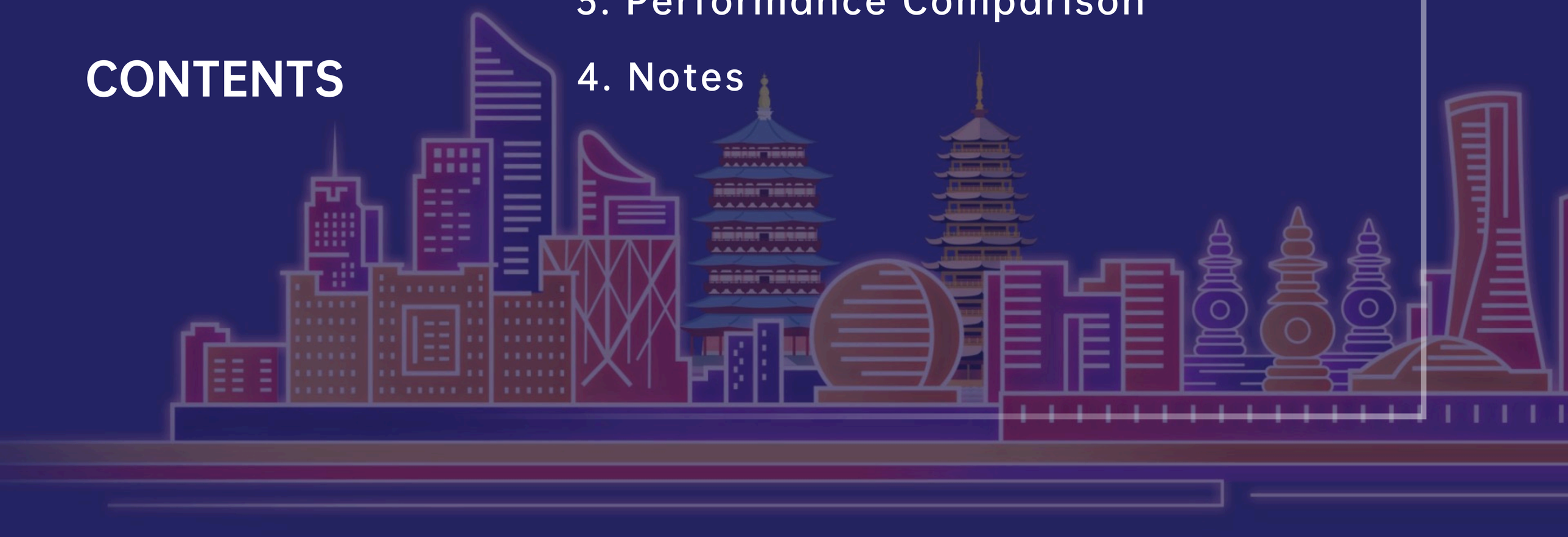
Apache Tomcat Committer, PMC member

Senior engineer at Du Xiaoman Financial（度小满金融）

COMMUNITY
THE ASF CONFERENCE
CODE

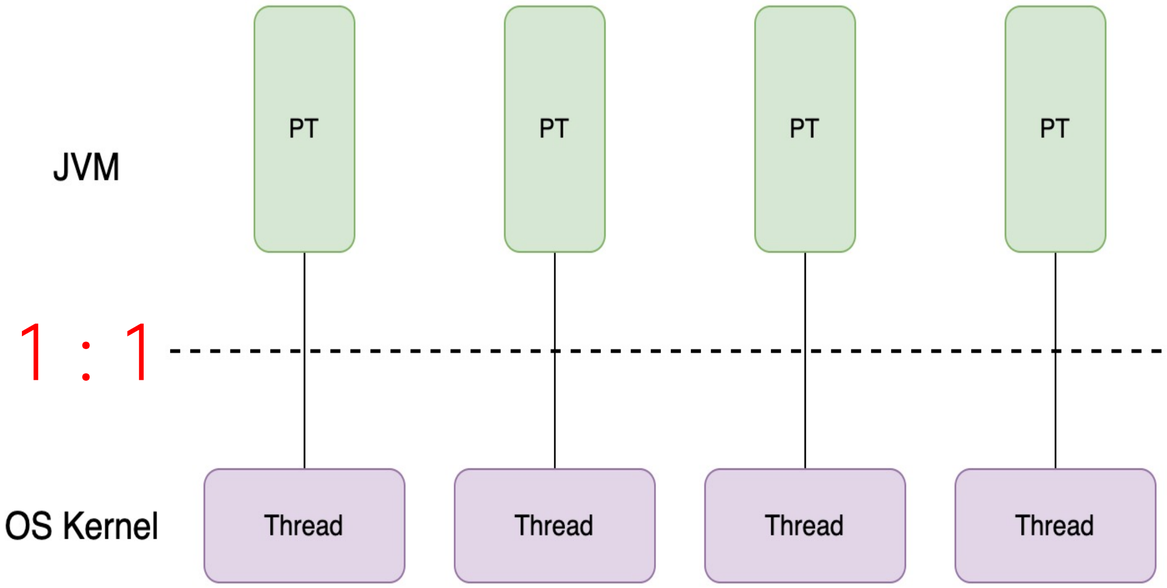
COMMUNITY
THE ASF CONFERENCE
CODE

CONTENTS



Platform Thread vs. Virtual Thread

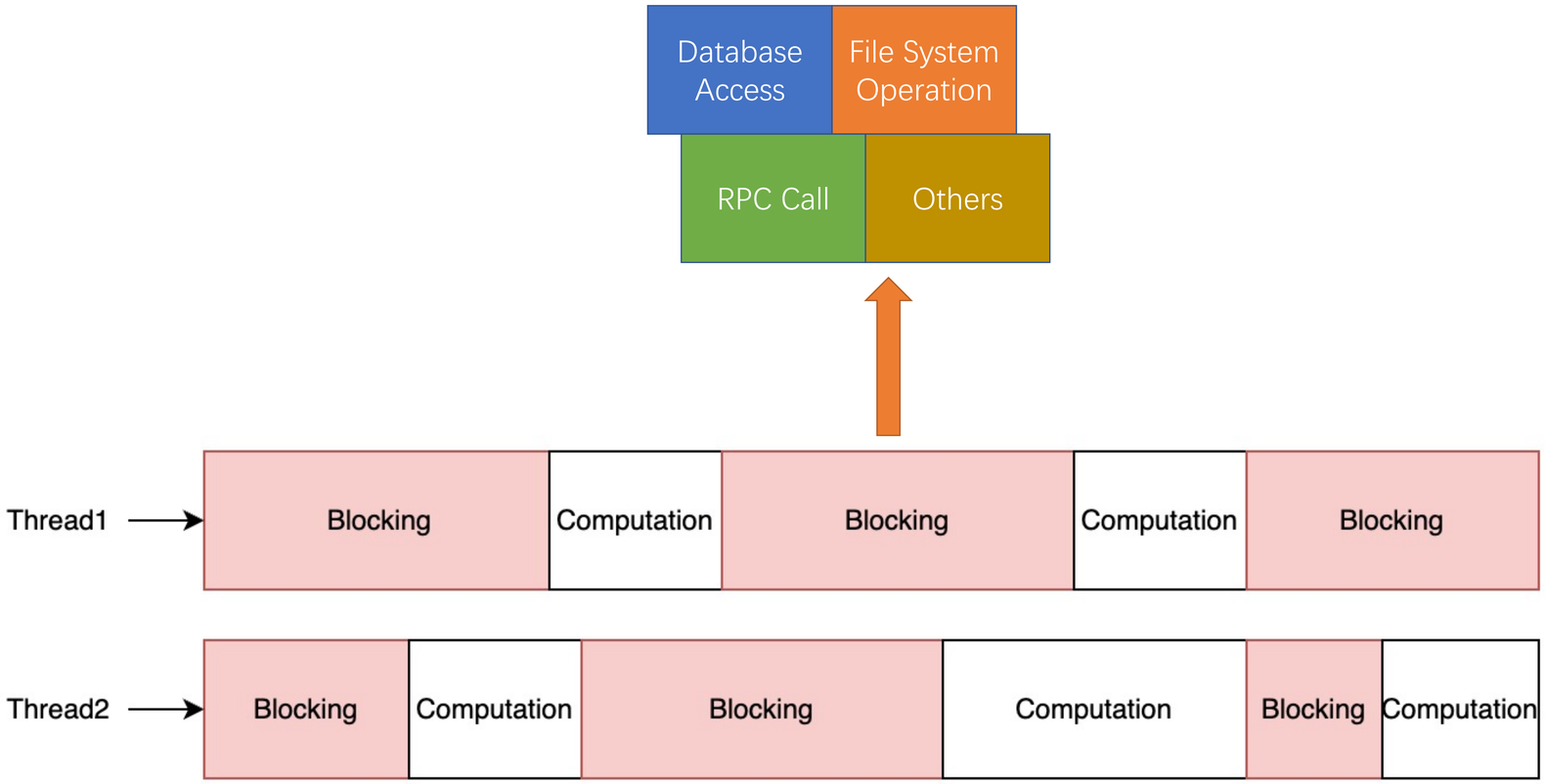
Platform Thread



- Java Platform Threads are OS Thread Wrappers
- High Resource Usage and Overhead
  (at least KB-level memory space)
- Limited Number

Platform Thread

Many Threads Blocked Waiting for Operations to Complete



Platform Thread

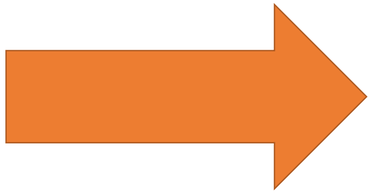
Common Solutions

Asynchronous Programming

CompletableFuture

Reactive Programming

Reactor、RxJava



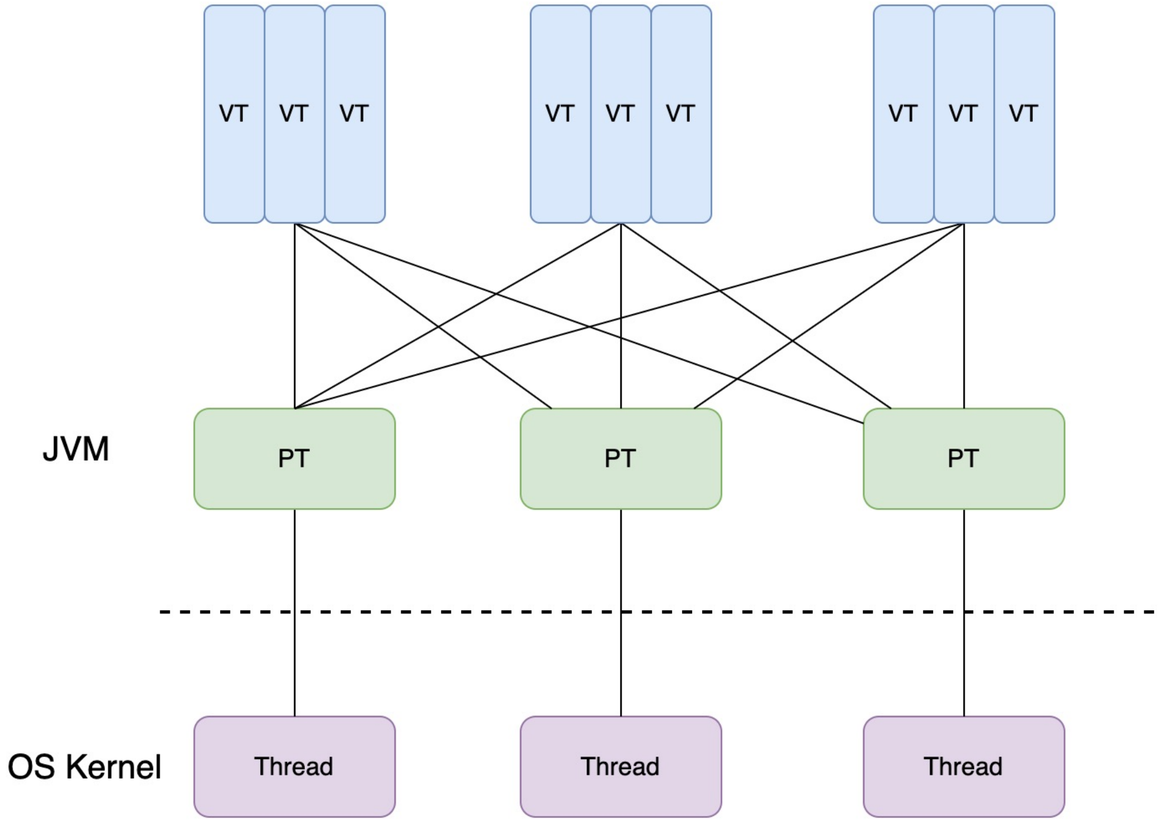
Pros:

- High throughput request handling
- Excellent resource utilization

Cons:

- High learning curve
- Different code style
- Difficult to understand
- Difficult to debug

Virtual Thread



- Not linked to OS Thread
- Light weight
  (byte-level memory space)
- Easy to use
  (Just like the previous usage with minimal change)
- Preserving the thread-per-request style
- Easy troubleshooting, debugging, and profiling

Virtual Thread

Usage

```java
Thread.startVirtualThread(() -> System.out.println("hello world"));

Thread.ofVirtual()
        .unstarted(() -> System.out.println("hello world"))
        .start();

Thread.ofVirtual()
        .factory()
        .newThread(() -> System.out.println("hello world"))
        .start();

try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
    executor.submit(() -> System.out.println("hello world"));
}
```

Application in Tomcat

Application in Tomcat

Standalone

```
<Executor
name="tomcatThreadPool"
className="org.apache.catalina.core.StandardVirtualThreadExecutor"
namePrefix="catalina-exec-"
/>
```

Only **namePrefix** setting!

No **maxThreads** setting!

No **maxQueueSize** setting!

Application in Tomcat

SpringBoot

```
spring.threads.virtual.enabled=true

// The following configuration is no longer setting
server.tomcat.threads.max=200
server.tomcat.threads.min-spare=10
server.tomcat.threads.max-queue-capacity=111
```

Default Thread Prefix: tomcat-handler-

```
 (process running for 5.757)
2024-07-20T17:56:16.455+08:00  INFO 1888 --- [vt-demo] [omcat-handler-0] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet
'dispatcherServlet'
2024-07-20T17:56:16.455+08:00  INFO 1888 --- [vt-demo] [omcat-handler-0] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2024-07-20T17:56:16.456+08:00  INFO 1888 --- [vt-demo] [omcat-handler-0] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
```

COMMUNITY
THE ASF CONFERENCE
CODE

Performance Comparison

Environment

Debian 11

4C4G

Oracle-jdk 21.0.4

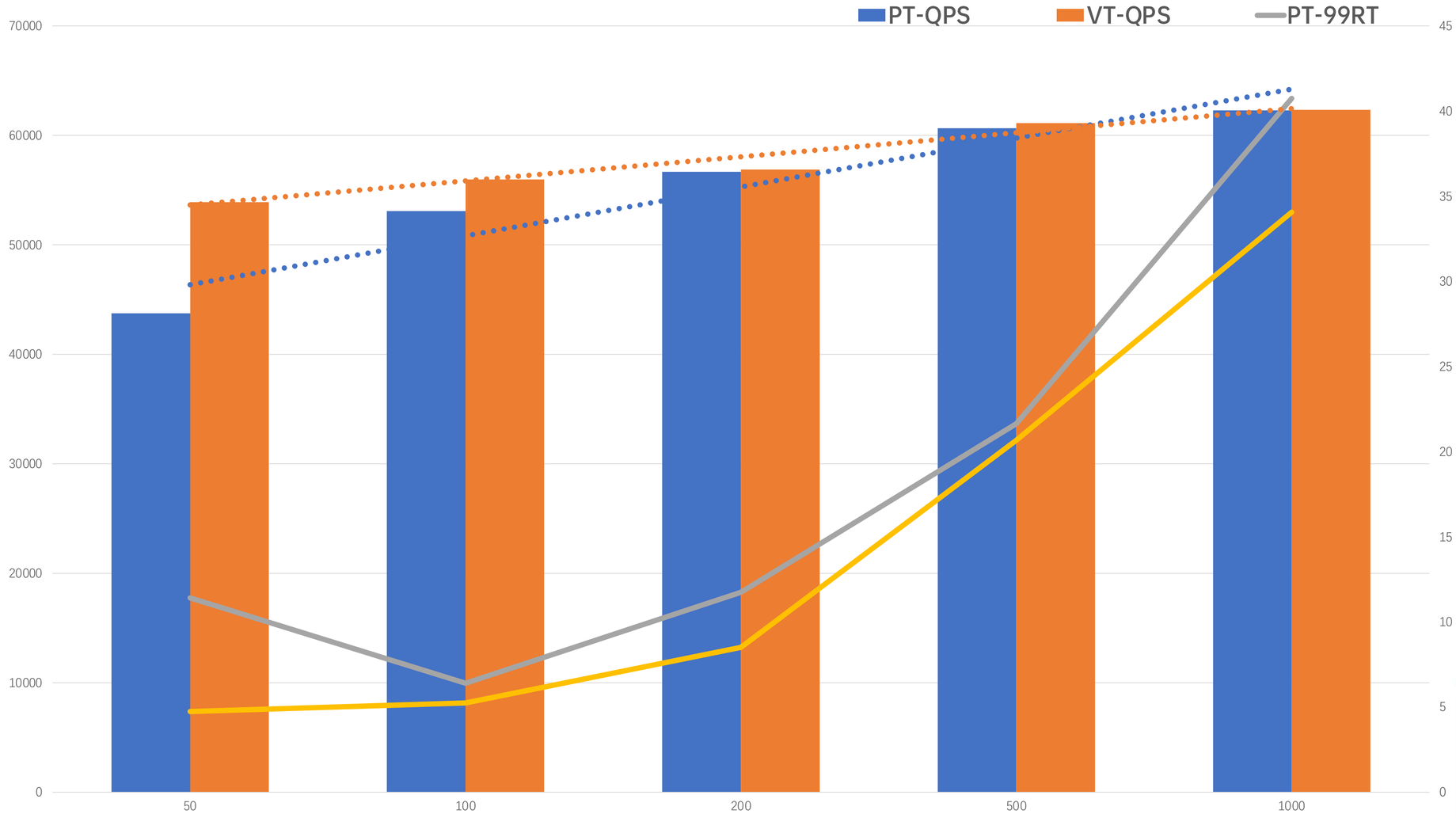
SpringBoot 3.3.1

wrk

```java
@RestController
public class TestController {

    @GetMapping("fib")
    public Object fib() {
        fibonacci();
        return "ok";
    }

    @GetMapping("block")
    public Object block() throws InterruptedException {
        Thread.sleep(200);
        return "ok";
    }

    private long fibonacci() {
        int number = 10_000;
        long sum = 2;
        int f1 = 1, f2 = 2, fb = 1;
        for(int i = 3; i <= number; i++) {
            fb = f1 + f2;
            f1 = f2;
            f2 = fb;
            sum += fb;
        }
        return sum;
    }
}
```

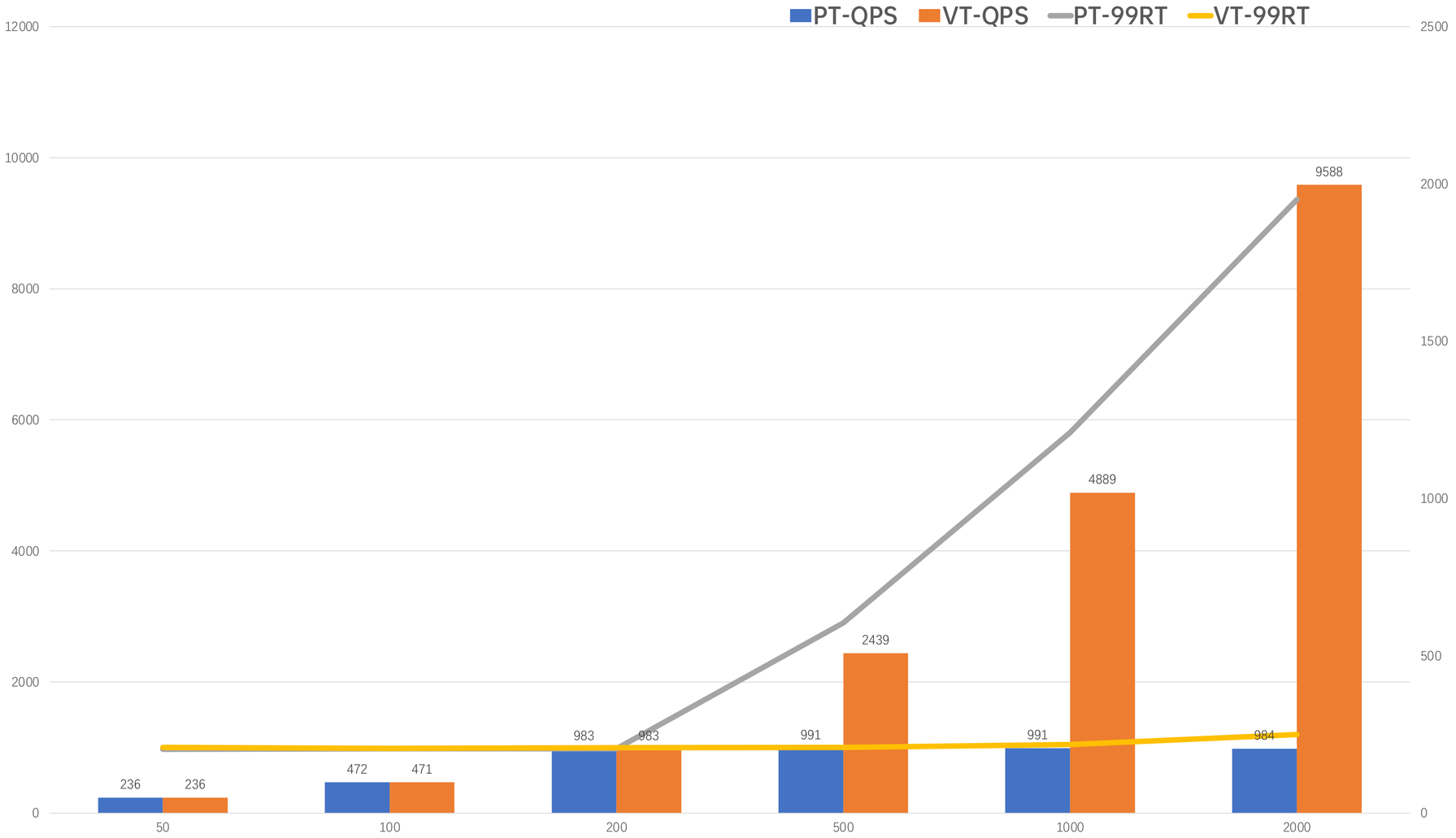
Throughput

Fibonacci calculation



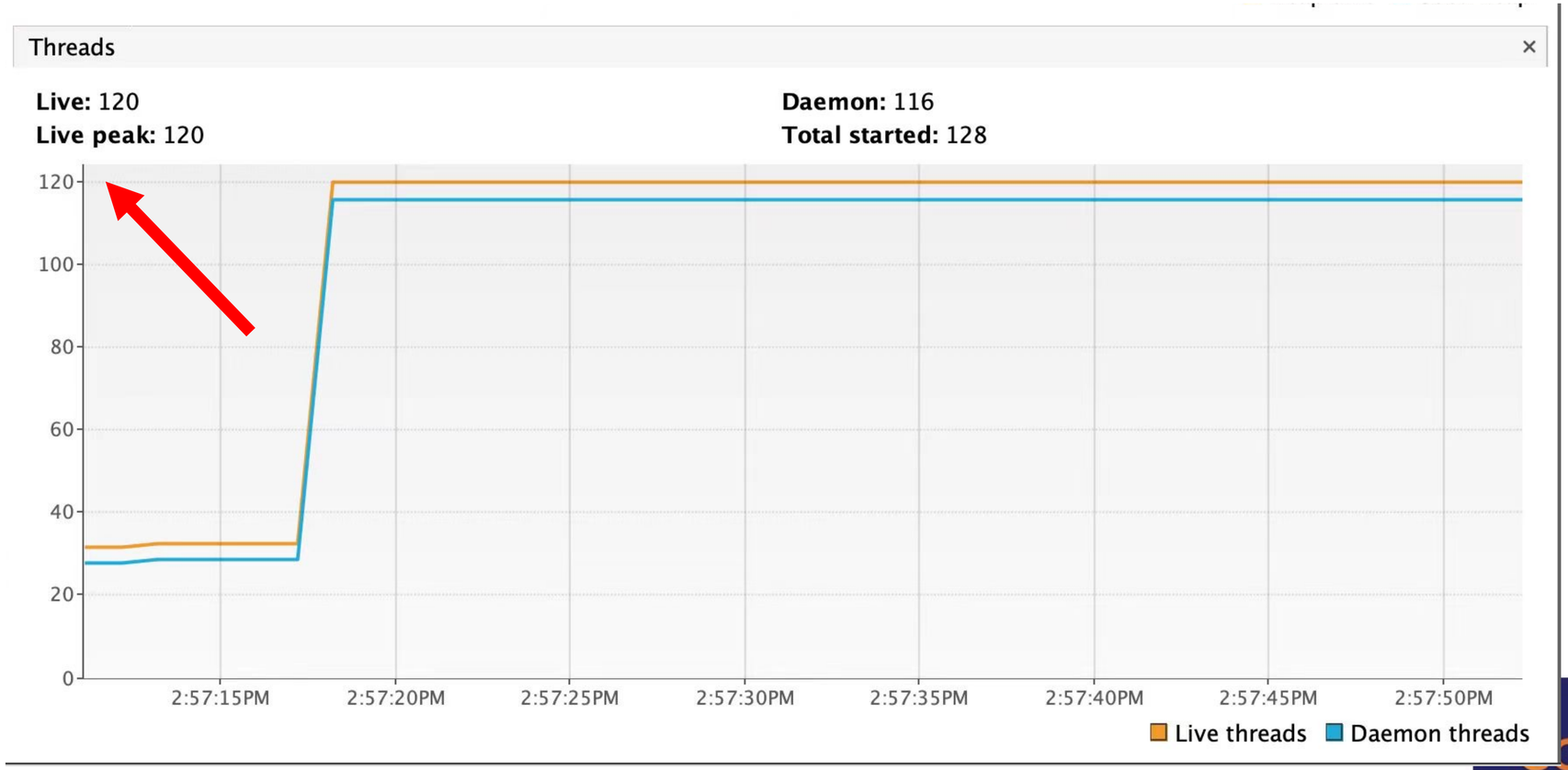
Throughput

Blocking Operation



Threads

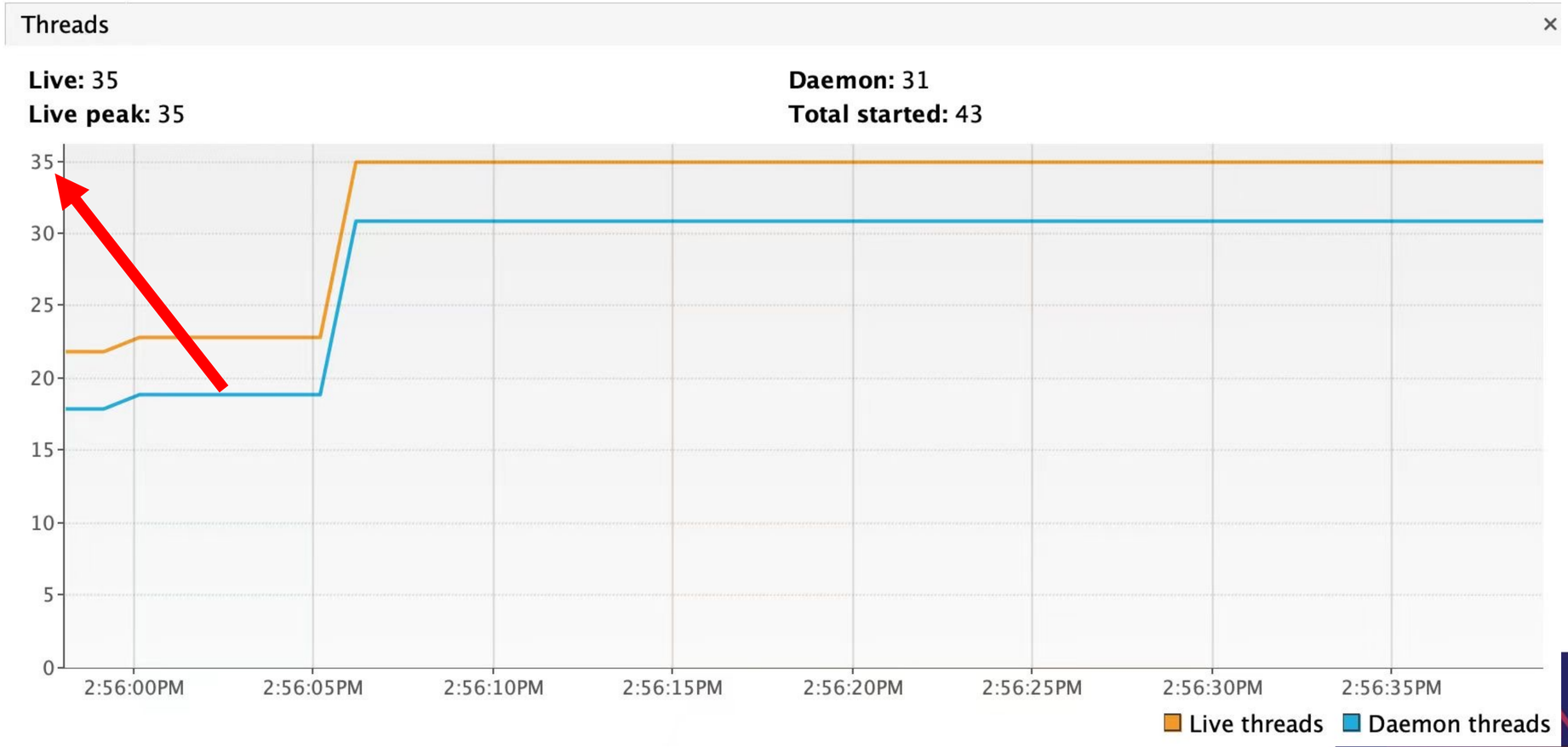
Platform Thread



100 Connections – 30s

Threads

Virtual Thread



Only a small number of platform threads are needed to handle a large number of tasks

Notes

Notes

Virtual Thread

- Do not use with native methods（JNI）or Foreign Function & Memory API
- Instead of synchronized, use ReentrantLock
- Be careful when using ThreadLocal
- Don‘t pool it

COMMUNITY
THE ASF CONFERENCE
CODE

Thanks

https://github.com/aooohan