



# PROJECT LOOM / VTHREADS

EXPLORING CONCURRENCY IN JAVA 21

# CHALLENGES

- Blocking operations e.g. http connections
- Web Servers / thread-per-request model / throughput not speed
- Synchronous I/O anti-pattern
  - Blocking the calling thread while I/O completes
  - Can reduce performance and affect vertical scalability



# CONCURRENCY IS HARD

- Expensive
- Thread pools
- Memory errors
- Sharing resources (Atomic)
- Blocking operations
- Semaphores / Mutex / Locks
- Hard to maintain
- Error handling

# WHAT ARE THE ALTERNATIVES?

- Async – The JS Model.. Completable Futures
- Reactive Programming?? WebClient / Flux / Mono
- Give up



DEMO



# PROJECT LOOM

- Incubating since Java 19
- Released in Java 21 (JEP: 444)
- Same API / existing code



## Example

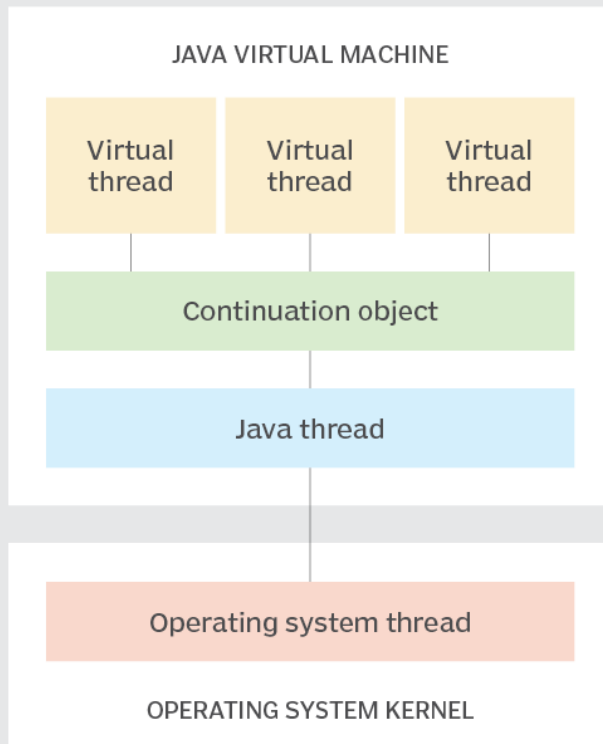
```
Runnable printThread = () ->
    System.out.println(Thread.currentThread());

ThreadFactory virtualThreadFactory =
    Thread.builder().virtual().factory();
ThreadFactory kernelThreadFactory =
    Thread.builder().factory();

Thread virtualThread =
    virtualThreadFactory.newThread(printThread);
Thread kernelThread =
    kernelThreadFactory.newThread(printThread);

virtualThread.start();
kernelThread.start();
```

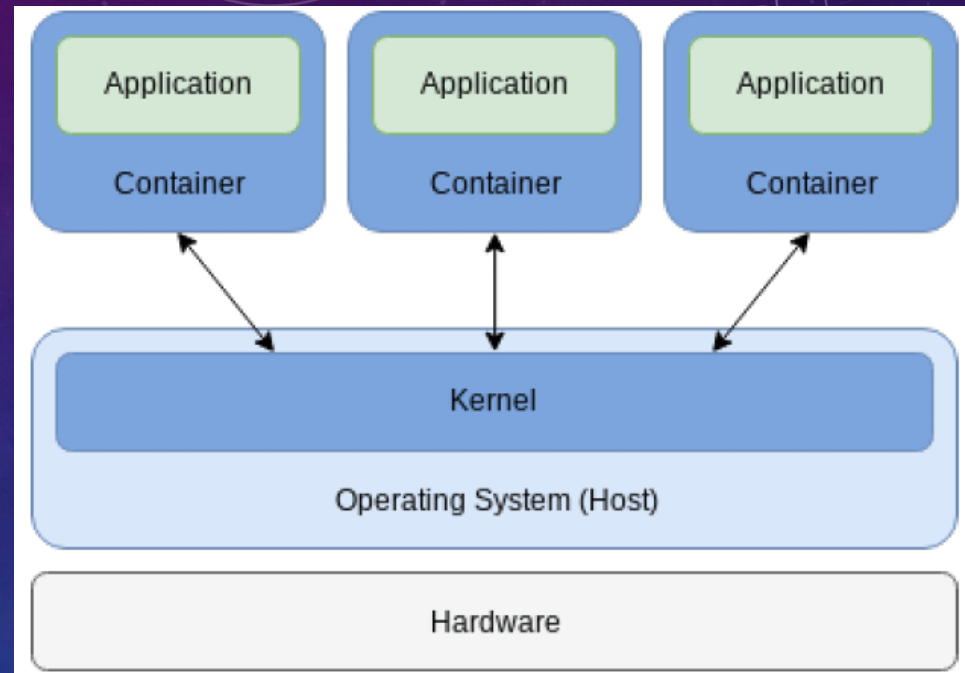
## Understanding virtual threads under Java



©2023 TECHTARGET. ALL RIGHTS RESERVED

<https://www.theserverside.com/tip/A-primer-on-Java-21-virtual-threads-with-examples>

## Docker



[https://www.researchgate.net/figure/Docker-container-architecture\\_fig1\\_333235708](https://www.researchgate.net/figure/Docker-container-architecture_fig1_333235708)



# EXISTING SUPPORT

- VirtualThreadExecutor is now available with Tomcat 10.1.10.
- Spring Boot 3.2 support - significant scalability improvements
  - I recommend checking out Dan Vega's video or blog post
  - <https://youtu.be/THavIYnlwck?si=z2Sp-OAW-yRCc2id>
  - <https://www.danvega.dev/blog/virtual-threads-spring-boot>

Servers like Tomcat already allow for virtual threads. If you are curious about servers and virtual threads, consider [this blog post by Cay Horstmann](#), where he shows the process of configuring Tomcat for virtual threads. He enables the virtual threads preview features and replaces the `Executor` with a custom implementation that differs by only a single line (you guessed it, `Executors.newThreadPerTaskExecutor`). The scalability benefit is significant, as he says: "With that change, 200 requests took 3 seconds, and Tomcat can easily take 10,000 requests."

InfoWorld





# WHAT NEXT?

## JEP 428: Structured Concurrency (Incubating)

- Simplify multithreaded programming by introducing an API for structured concurrency
- Treats multiple tasks running in different threads as a single unit of work
- Will streamline error handling and cancellation
- Improved reliability and enhanced observability

<https://openjdk.org/jeps/428>

BONUS DEMO





# REFERENCES

InfoWorld – Intro to Virtual Threads

<https://www.infoworld.com/article/3678148/intro-to-virtual-threads-a-new-approach-to-java-concurrency.html>

Microsoft - Synchronous I/O antipattern

<https://learn.microsoft.com/en-us/azure/architecture/antipatterns/synchronous-io/>

TheServerSide – Virtual Threads Primer

<https://www.theserverside.com/tip/A-primer-on-Java-21-virtual-threads-with-examples>

Baeldung – Virtual Threads vs Platform Threads

<https://www.baeldung.com/java-virtual-thread-vs-thread>

Aseem Savio – 5 million virtual threads

<https://blog.aseemsavio.com/how-i-spun-up-5-million-virtual-threads-without-stalling-the-jvm/>