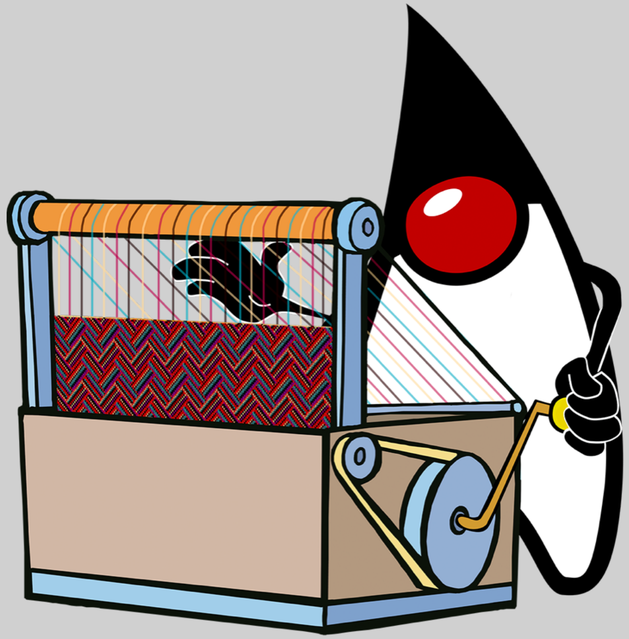
JDK 21
LTS

Virtual Threads



a Different Approach to Async/Await

JDK 21

LTS

Aims to Bring

easy-to-use, high-throughput,

lightweight concurrency

and **new programming models**

on the **Java** platform.

JDK 21: Virtual Threads

A different Approach to asynchronous programming

CHOOSE
CODE COMPATIBILITY

minimizing the impact on existing code
to benefit from this model

C#, Python, Javascript, C++, Rust

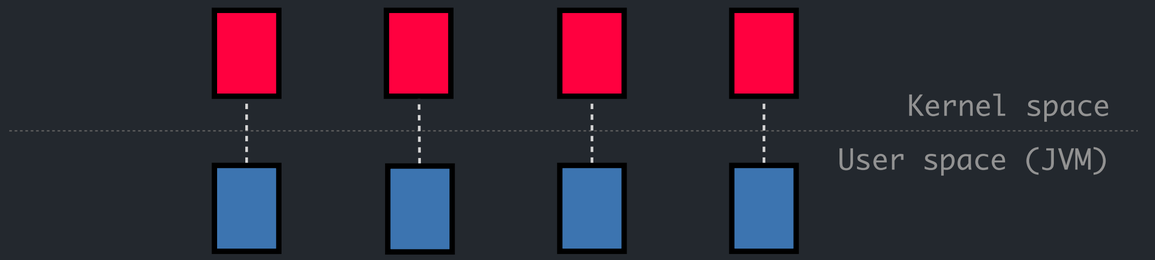
Introduced New Keywords

async/await

requiring a significant rewrite
to take advantage of this model

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



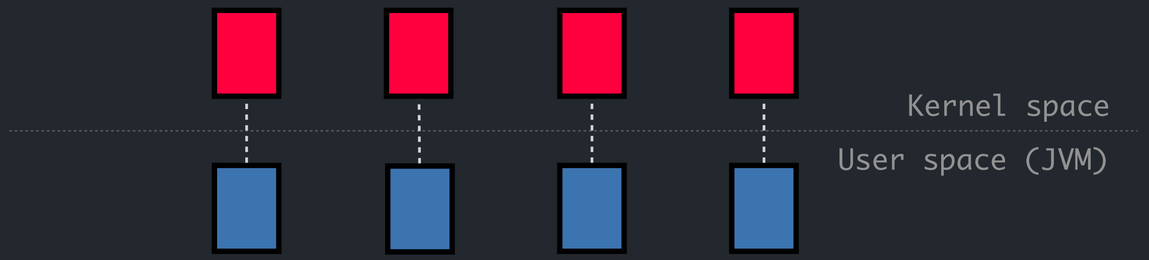

```java
// create a platform thread, and use .start() to start it.
Thread thread = new Thread(Runnable, "thread-name")
thread.start();

// create a platform thread, and use .start() to start it.
Thread thread = Thread.ofPlatform().name("thread-name").unstarted(Runnable);
thread.start();

// create a platform thread and start it. No need to call .start()
Thread thread = Thread.ofPlatform().name("thread-name").start(Runnable);
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

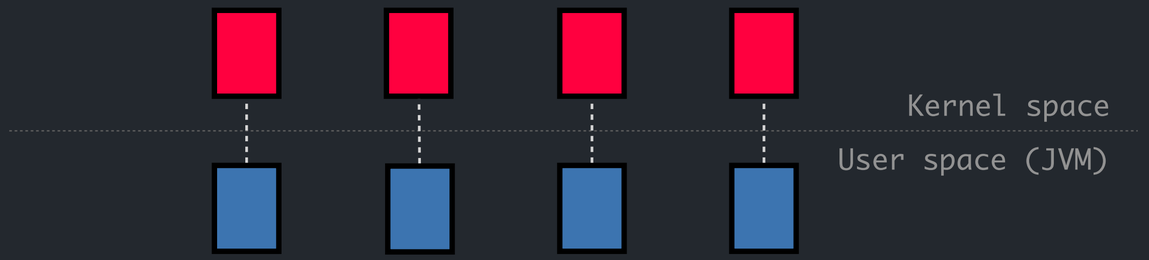


Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



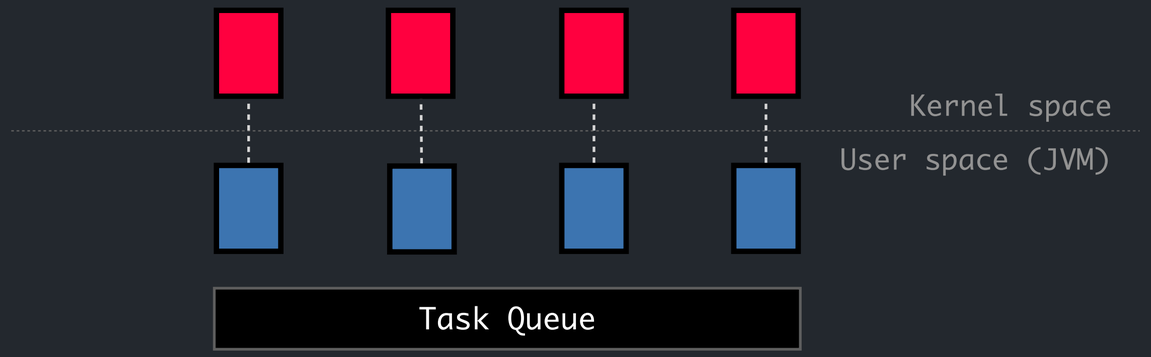
Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

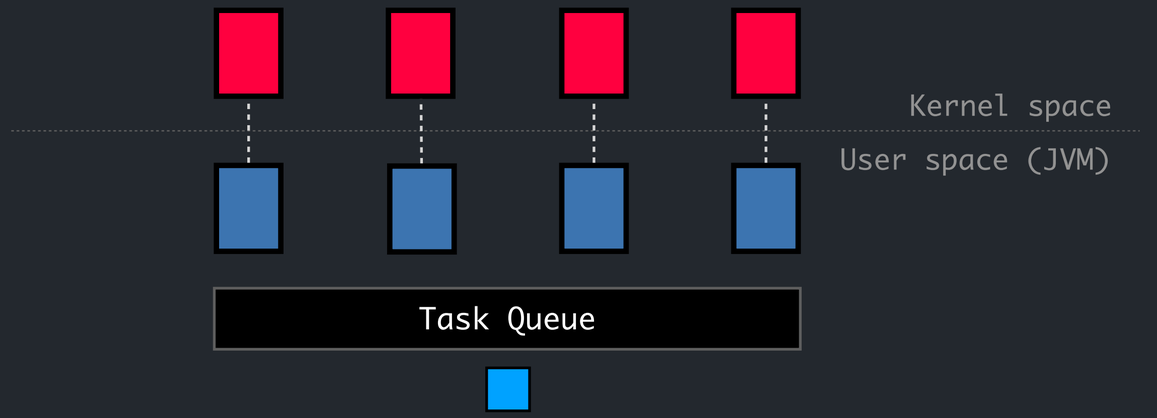
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

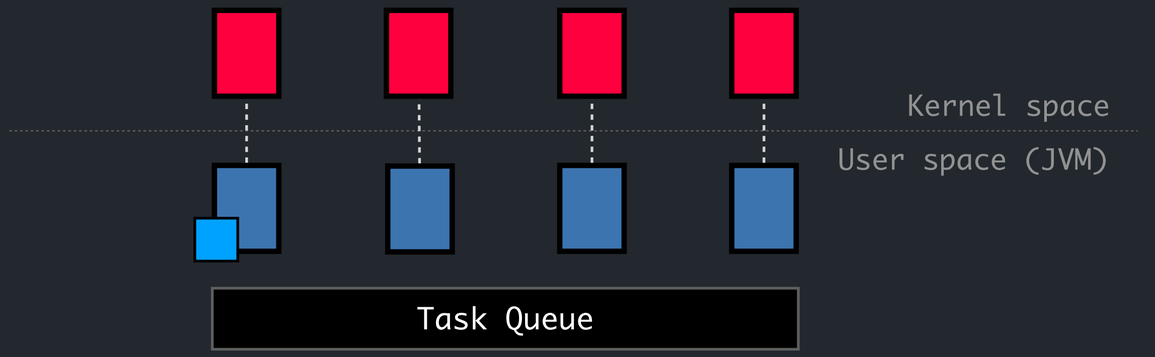
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

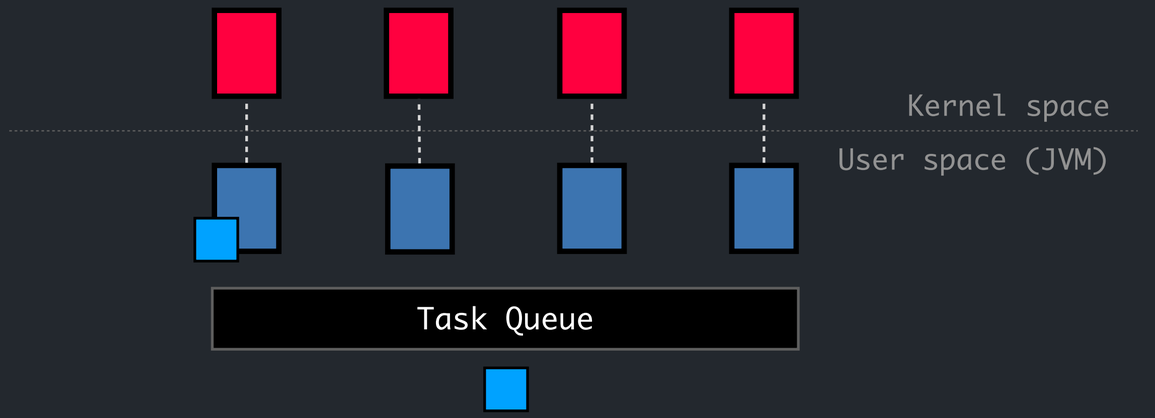
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

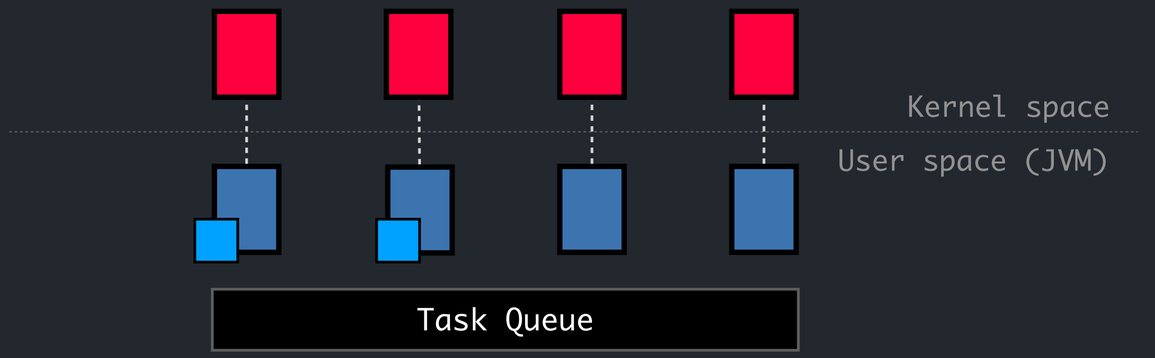
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

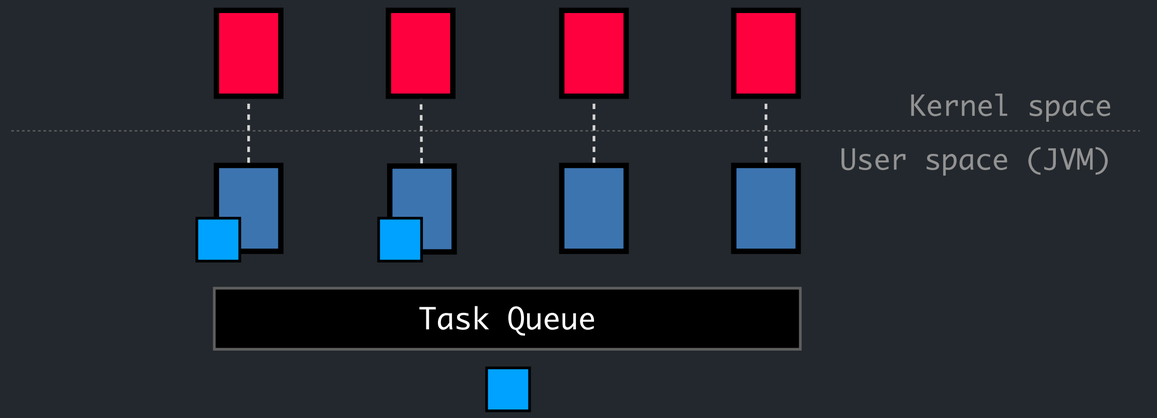
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

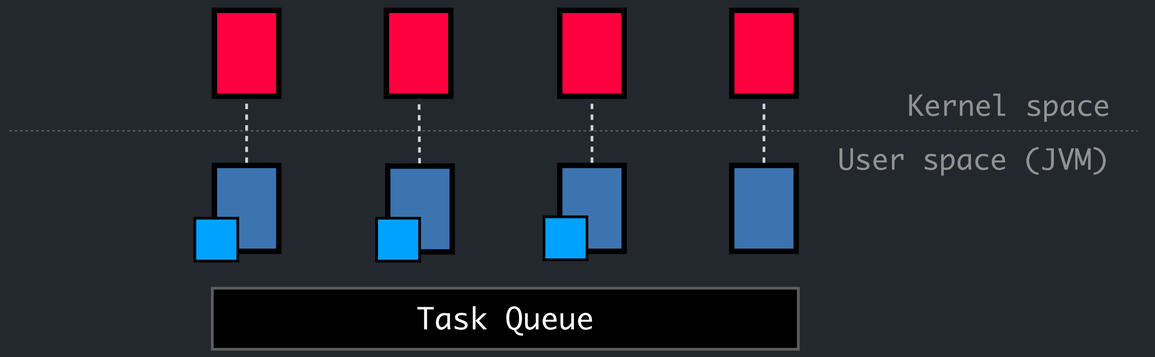
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

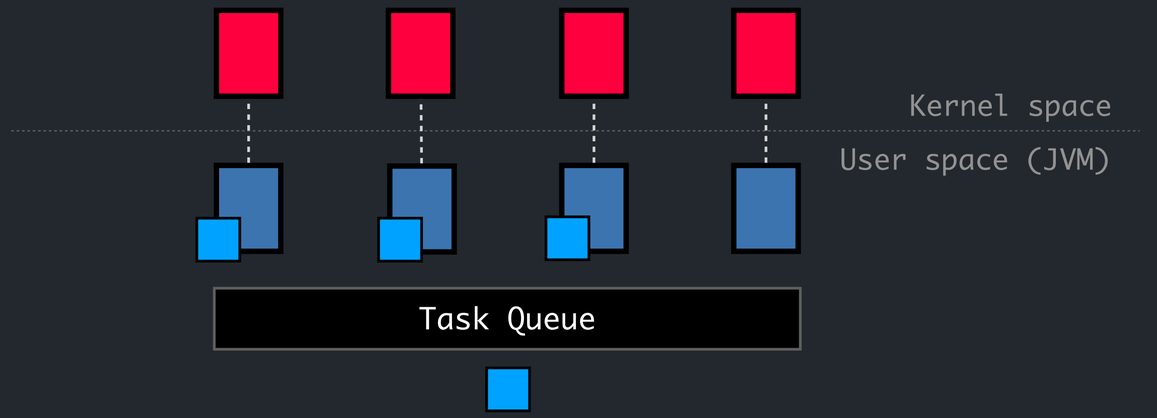
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

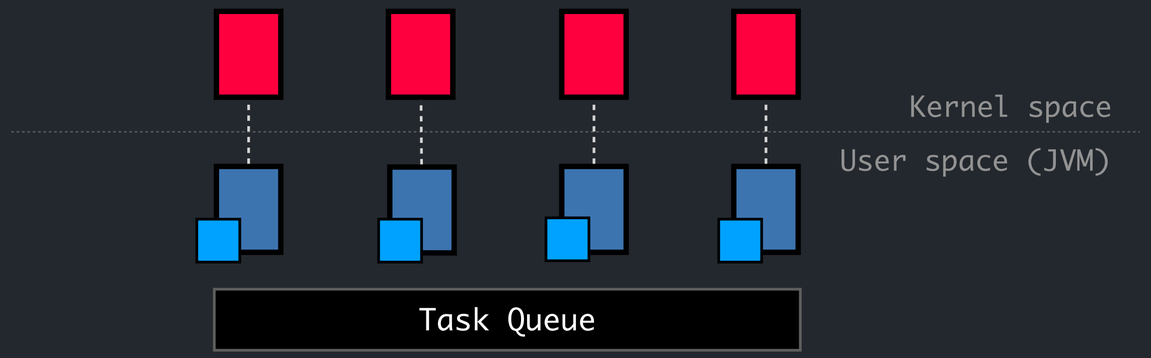
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

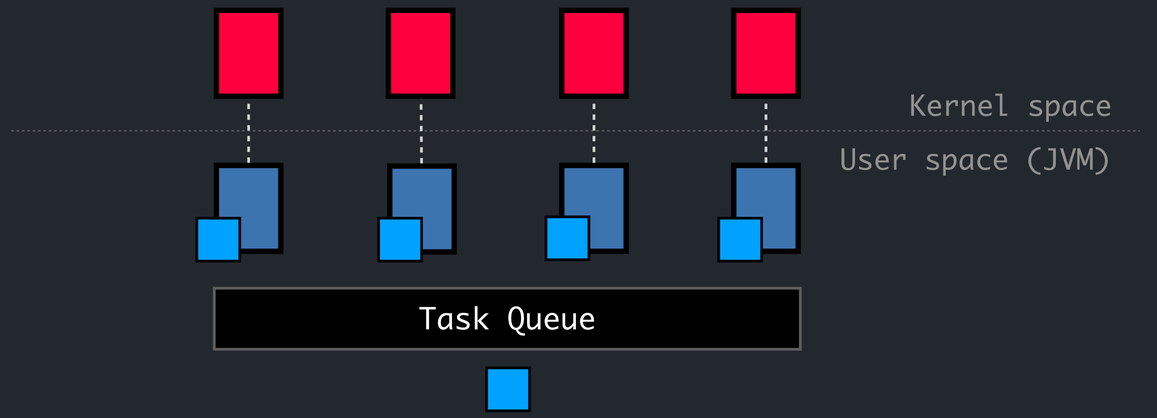
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

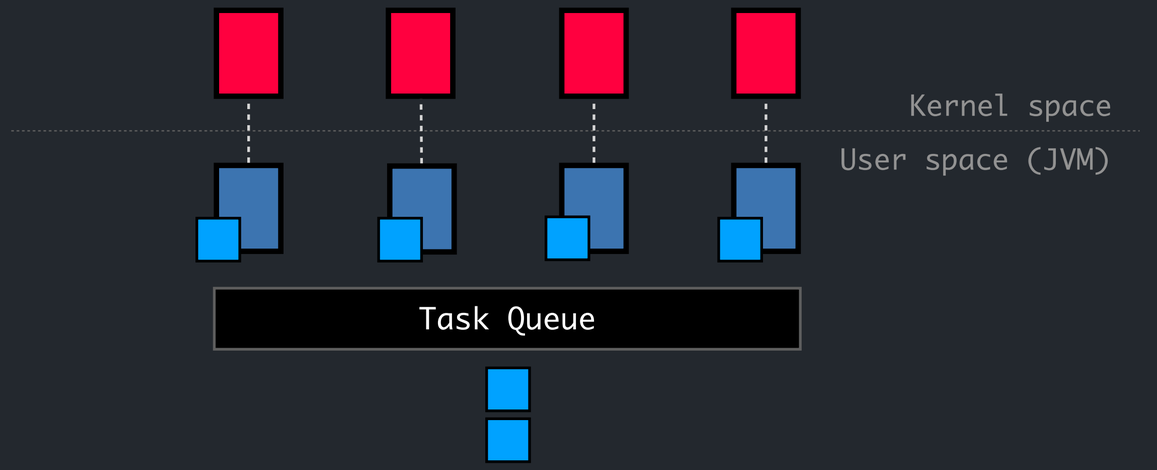
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

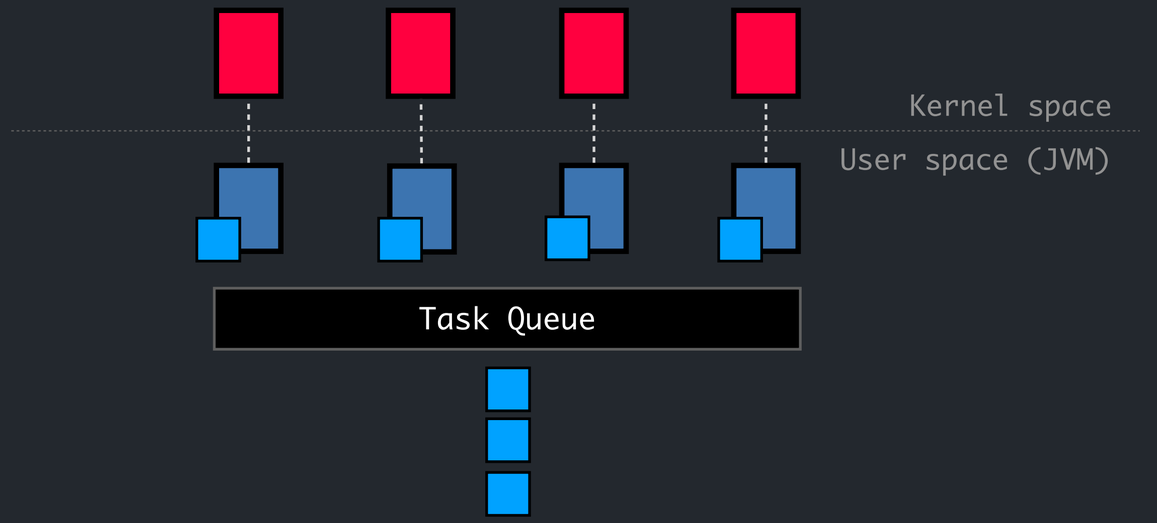
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

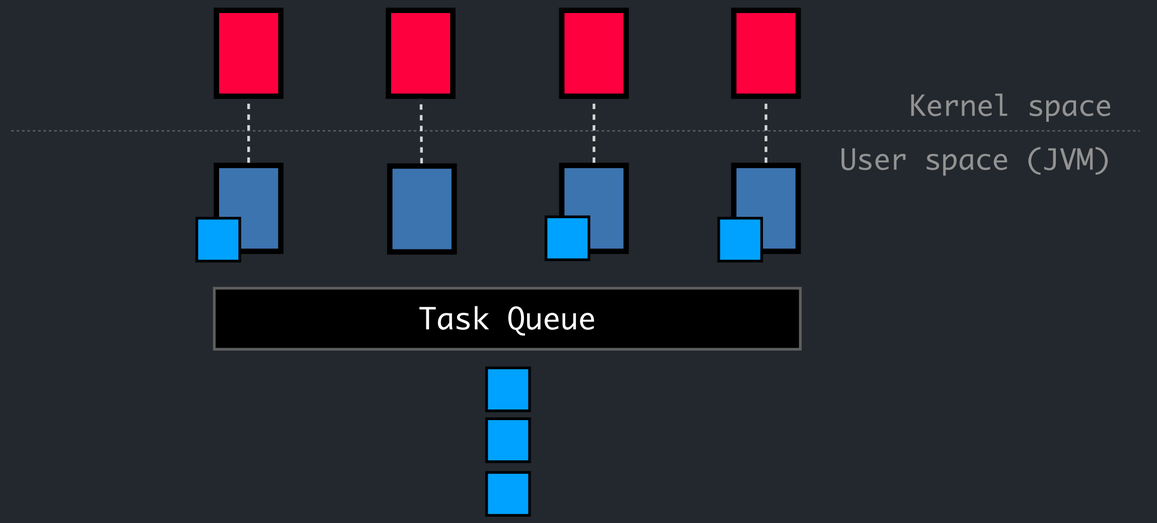
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

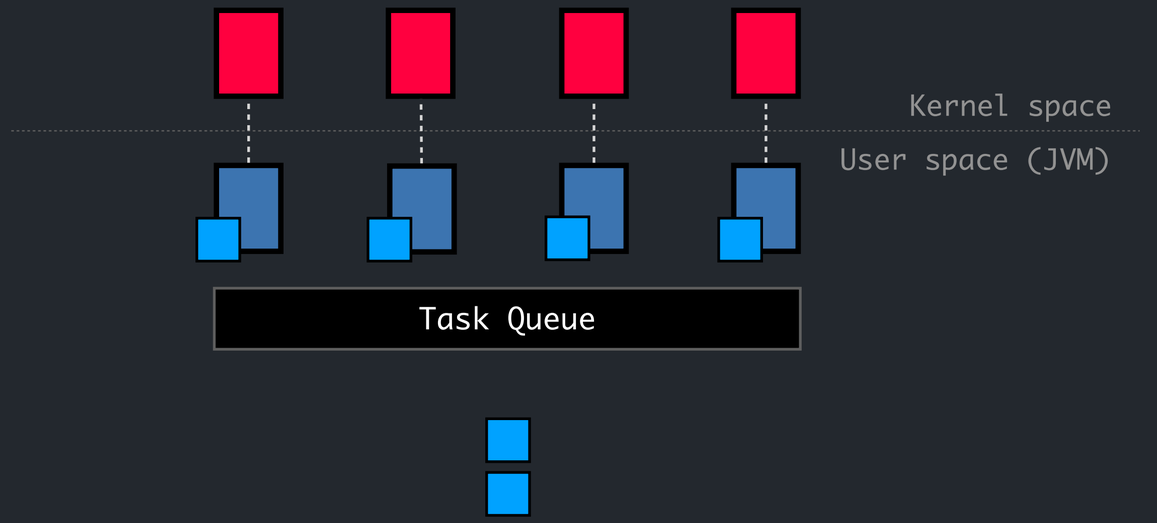
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

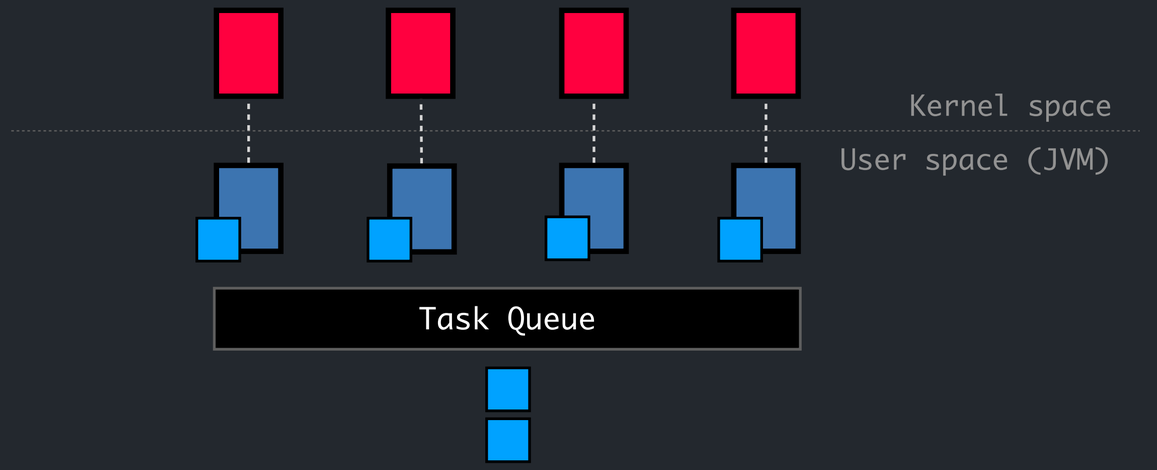
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

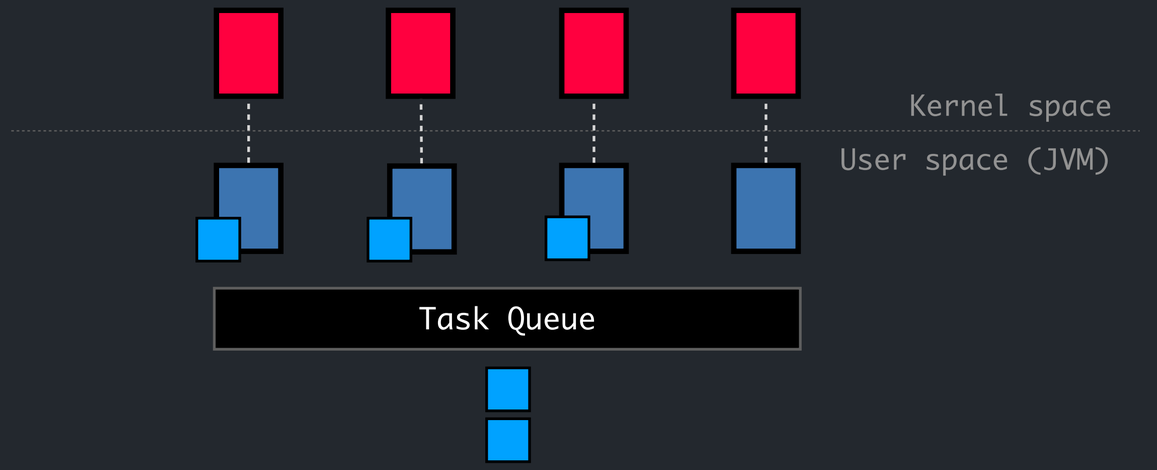
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



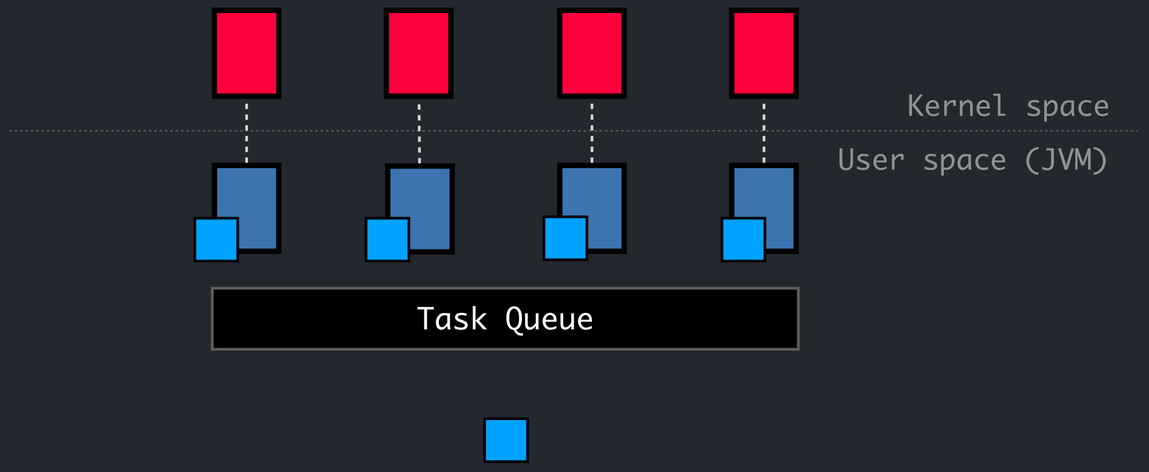
Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

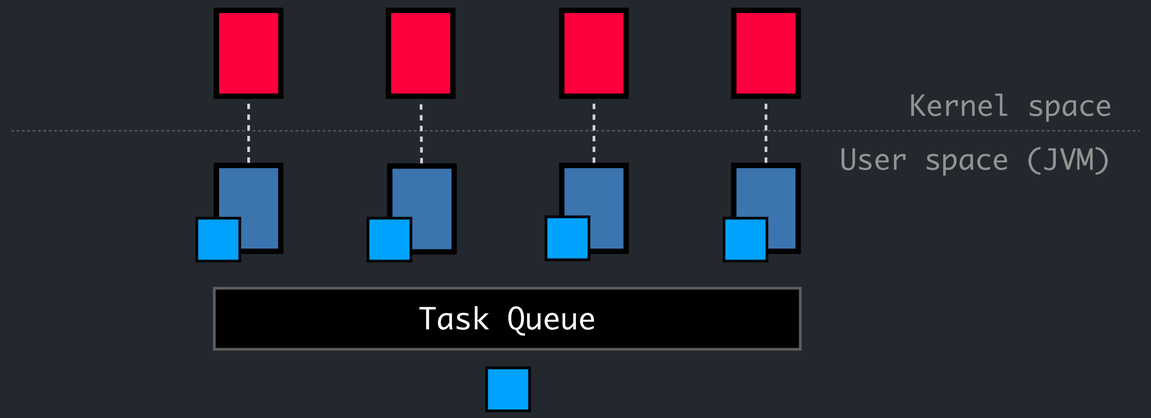
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

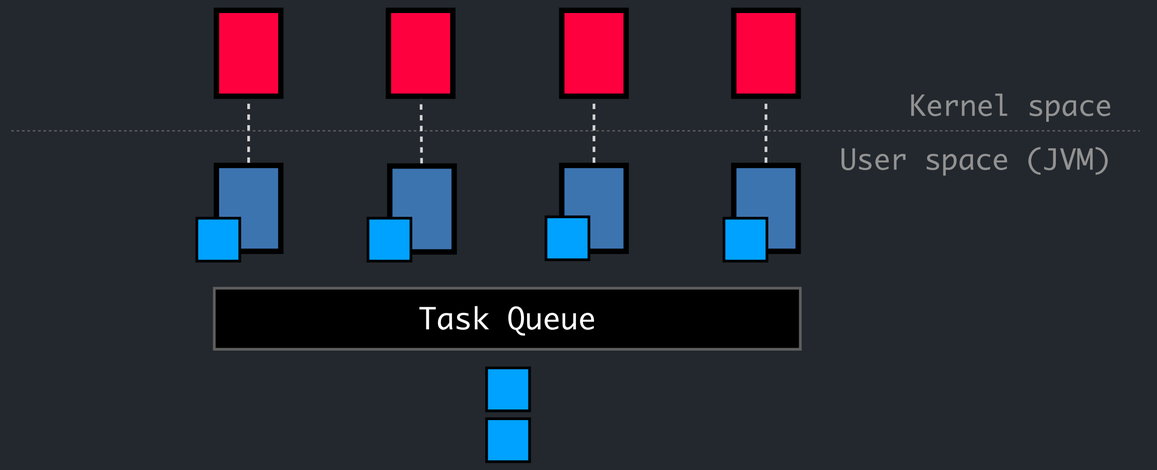
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Threads

Platform Threads & Executor Services

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

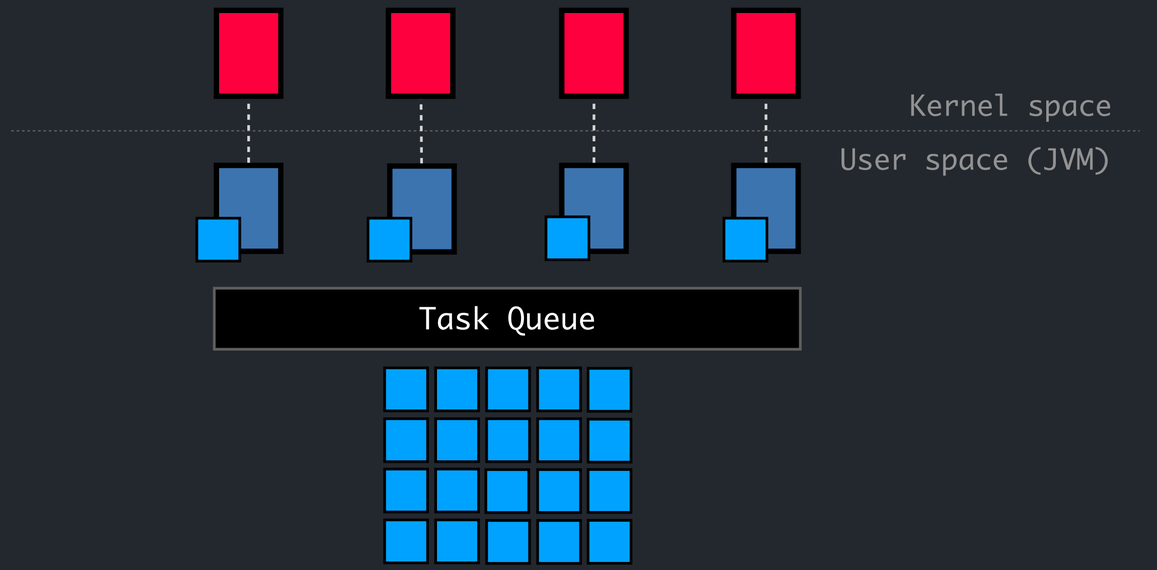
```
try (ExecutorService executor = Executors.newFixedThreadPool(4)) {
  for (int i = 0; i < 100; ++i) {
    executor.submit(() -> myTask());
  }
}
```

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

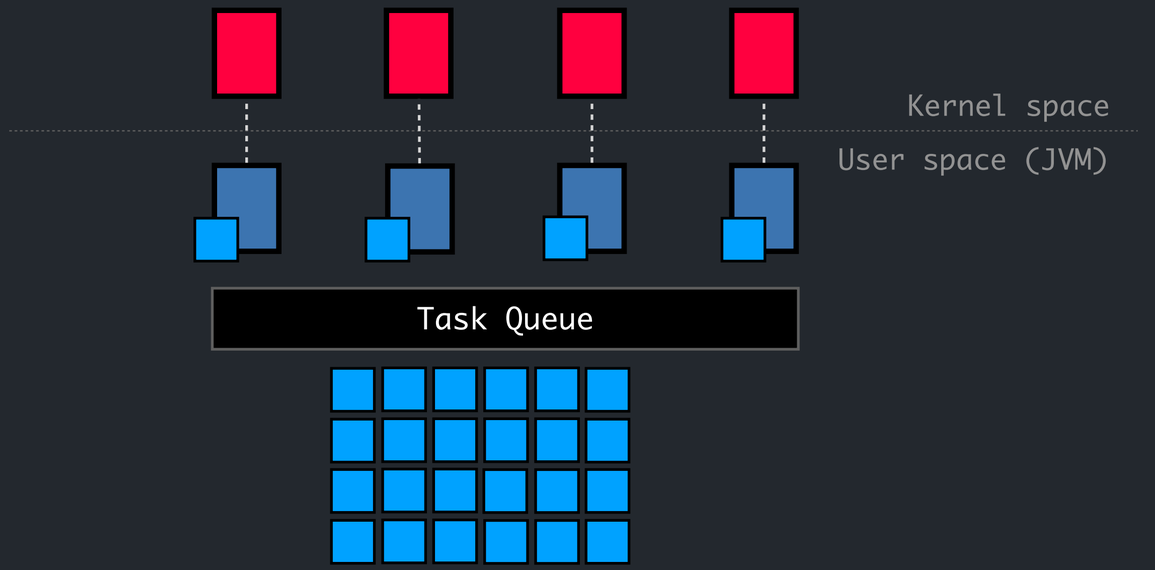
- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

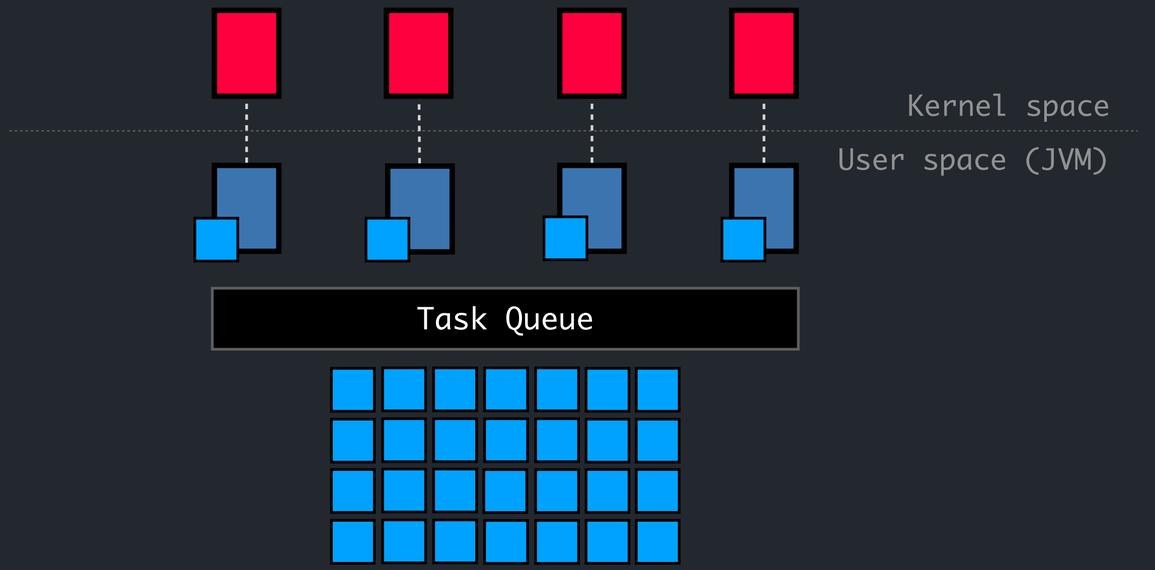
- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

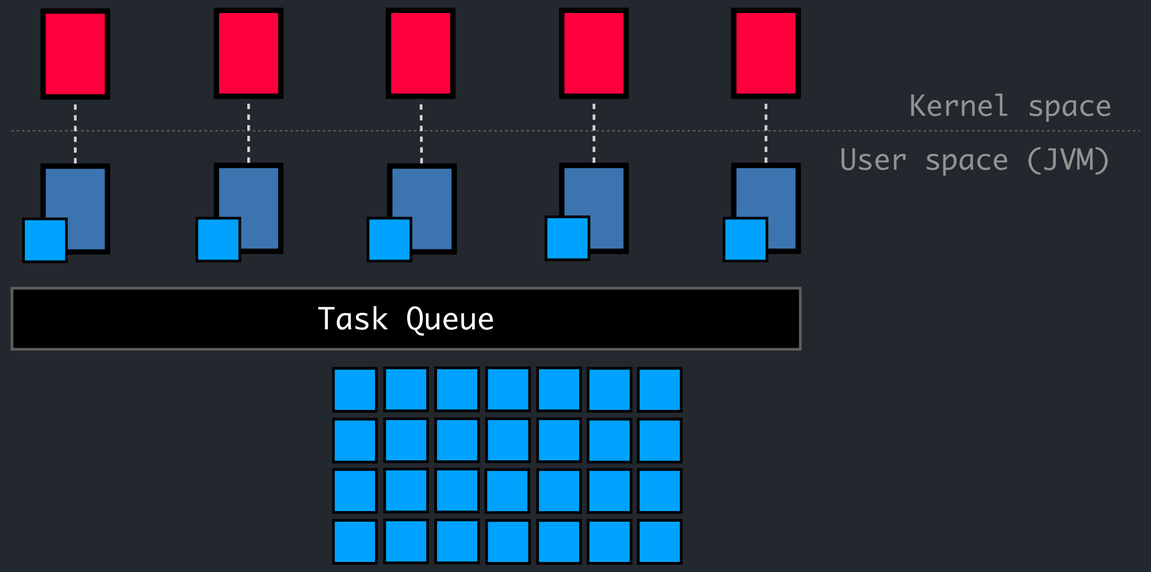
- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

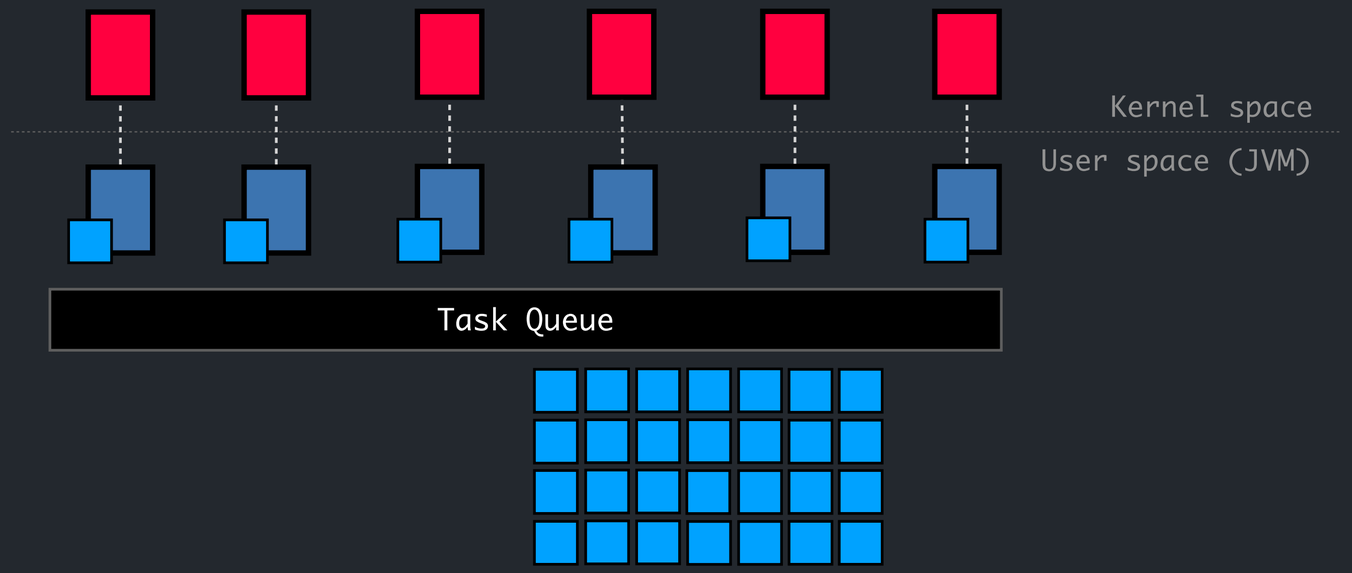
- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.



Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

In most backend services
Slow Tasks
are I/O Bound Tasks

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```

My Task
Not being blocked

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

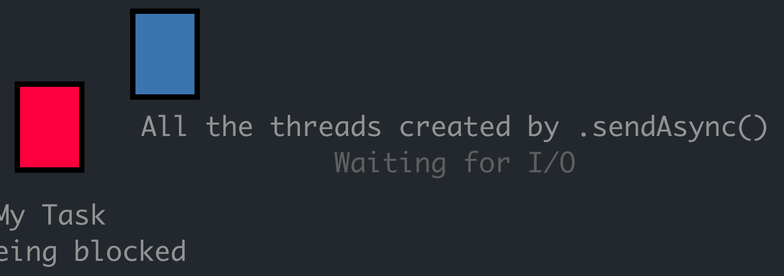
Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```



JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```

All the threads created by .sendAsync()
Waiting for I/O

My Task
Not being blocked

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```

JDK 21: Executor Services

Slow Tasks & Completable Future

Platform Threads

Typically mapped 1:1 to kernel threads scheduled by the operating system.

Platform Threads are not cheap

- startup time: ~1ms
- memory consumption: 2MB of stack
- context switch: ~100µs

CompletableFuture and the Async feeling

```
System.out.println("Active Threads " + Thread.activeCount());
for (int i = 0; i < 128; ++i) {
  httpClient.sendAsync(HttpRequest.newBuilder()
    .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
    .GET()
    .build(), BodyHandlers.discarding());
  System.out.println("Active Threads " + Thread.activeCount());
}
```

JDK 21: Threads

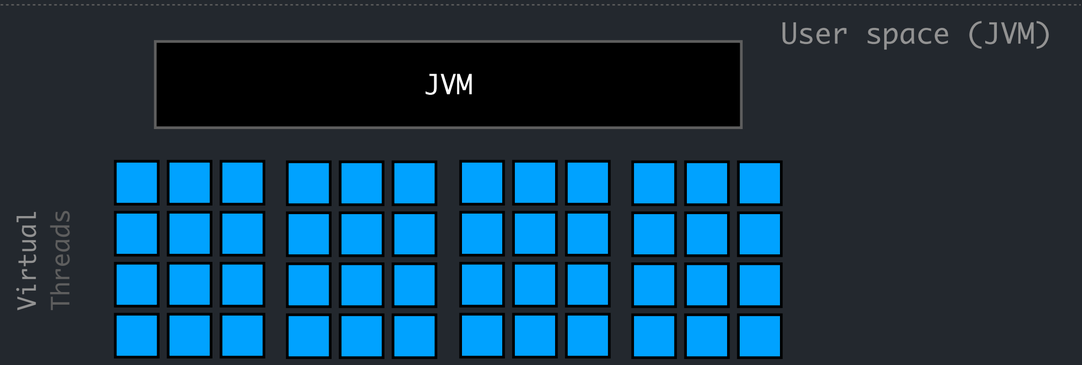
Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system




```java
// create a virtual thread and start it.
Thread thread = Thread.startVirtualThread(Runnable);

// create a virtual thread and start it.
Thread thread = Thread.ofVirtual().name(“thread-name").start(Runnable);

// create a virtual thread, and use .start() to start it.
Thread thread = Thread.ofVirtual().name("thread-name").unstarted(Runnable);
thread.start();
```

JDK 21: Threads

Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Eliminates the overhead
of
I/O Bound Tasks

JDK 21: Threads

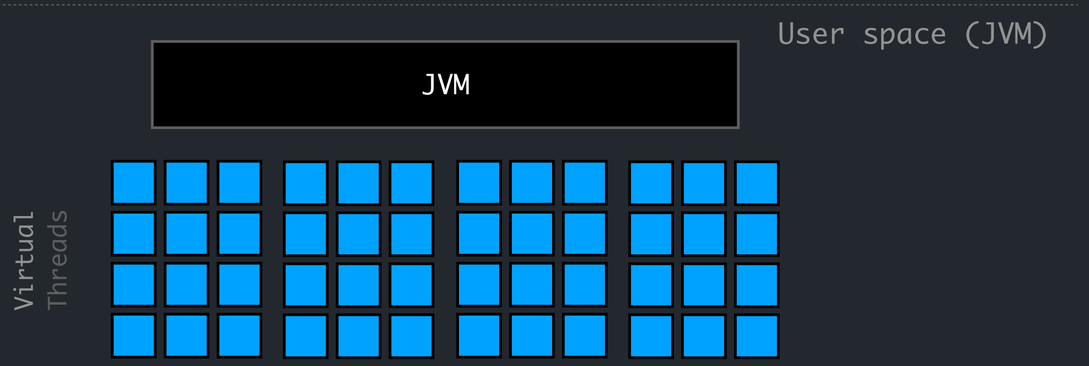
Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system



JDK 21: Threads

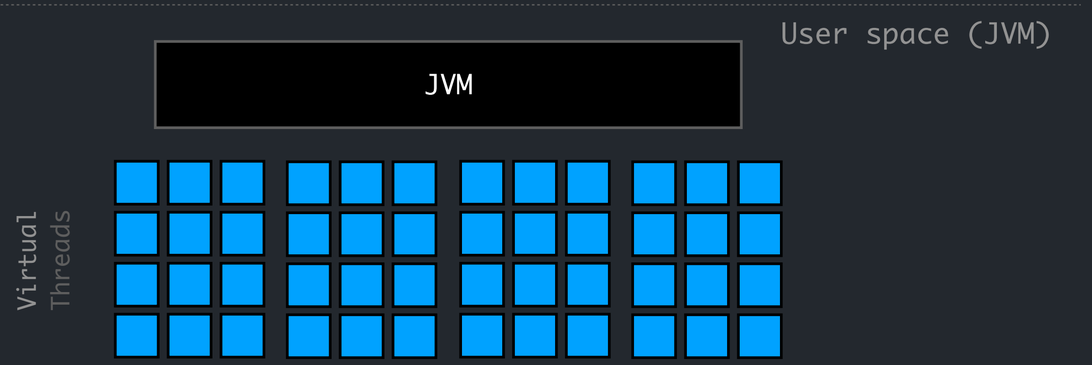
Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system



Use them for I/O Bound Tasks

JDK 21: Threads

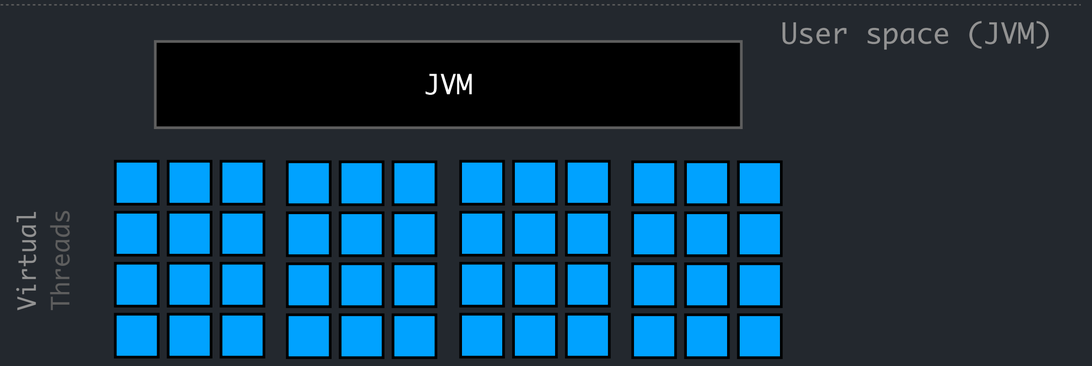
Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system



Use them for I/O Bound Tasks

CPU Bound	I/O Bound
Spends most of the time using CPU	Spends most of the time waiting for I/O
Longer CPU bursts	Shorter CPU bursts

JDK 21: Threads

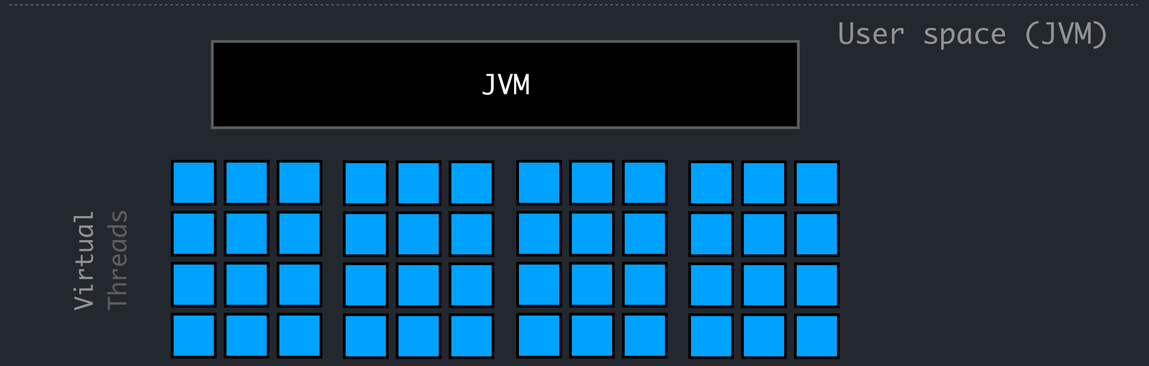
Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system



Almost no changes to the code required!

Java did all of the work underneath

Rewriting the I/O & Blocking Operations

To Cooperate with the Virtual Threads Scheduler

JDK 21: Threads

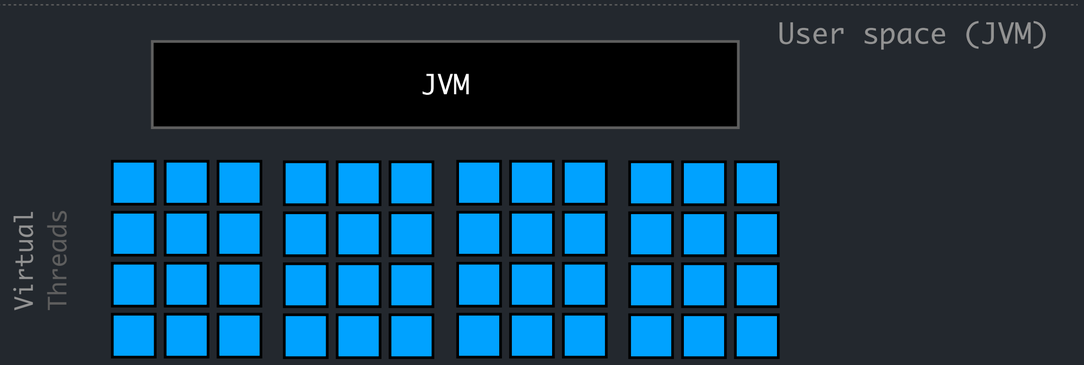
Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system




```
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  executor.submit(() -> myTask());
}
```

JDK 21: Threads

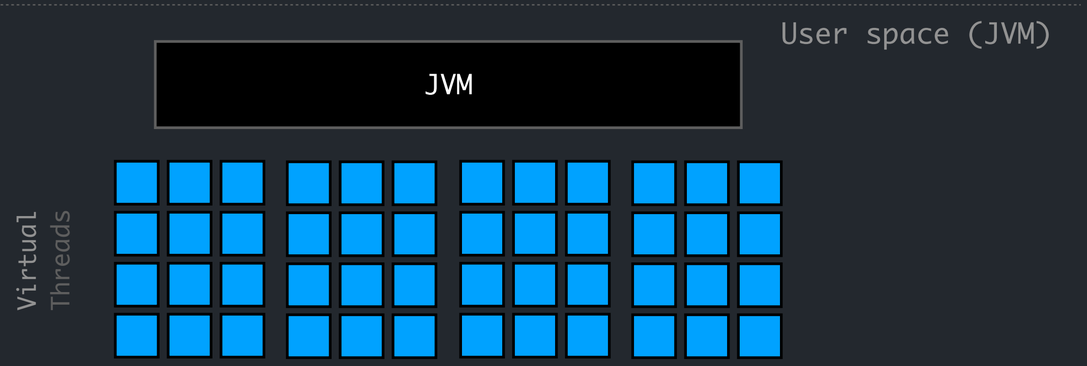
Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system




```
// create a virtual thread and start it.
Thread thread = Thread.startVirtualThread(Runnable);

// create a virtual thread and start it.
Thread thread = Thread.ofVirtual().name(“thread-name").start(Runnable);

// create a virtual thread, and use .start() to start it.
Thread thread = Thread.ofVirtual().name("thread-name").unstarted(Runnable);
thread.start();
```

JDK 21: Threads

Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

```
// create a platform thread and start it. No need to call .start()
Thread thread = Thread.ofPlatform().name("my-thread-name").start(Runnable);

// create a virtual thread and start it.
Thread thread = Thread.ofVirtual().name("my-thread-name").start(Runnable);

boolean isVirtual = thread.isVirtual();
```

JDK 21: Threads

Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

```
// create a platform thread and start it. No need to call .start()
Thread thread = Thread.ofPlatform().name("my-thread-name").start(Runnable);

// create a virtual thread and start it.
Thread thread = Thread.ofVirtual().name("my-thread-name").start(Runnable);

boolean isVirtual = thread.isVirtual();
```

jstack or Thread.currentThread() shows

```
Thread[#29,my-thread-name,5,main]
VirtualThread[#30,my-thread-name]/runnable@ForkJoinPool-1-worker-1
```

JDK 21: Threads

Virtual Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

```
// create a platform thread and start it. No need to call .start()
Thread thread = Thread.ofPlatform().name("my-thread-name").start(Runnable);

// create a virtual thread and start it.
Thread thread = Thread.ofVirtual().name("my-thread-name").start(Runnable);

boolean isVirtual = thread.isVirtual();
```

jstack or Thread.currentThread() shows

```
Thread[#29,my-thread-name,5,main]
VirtualThread[#30,my-thread-name]/runnable@ForkJoinPool-1-worker-1
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

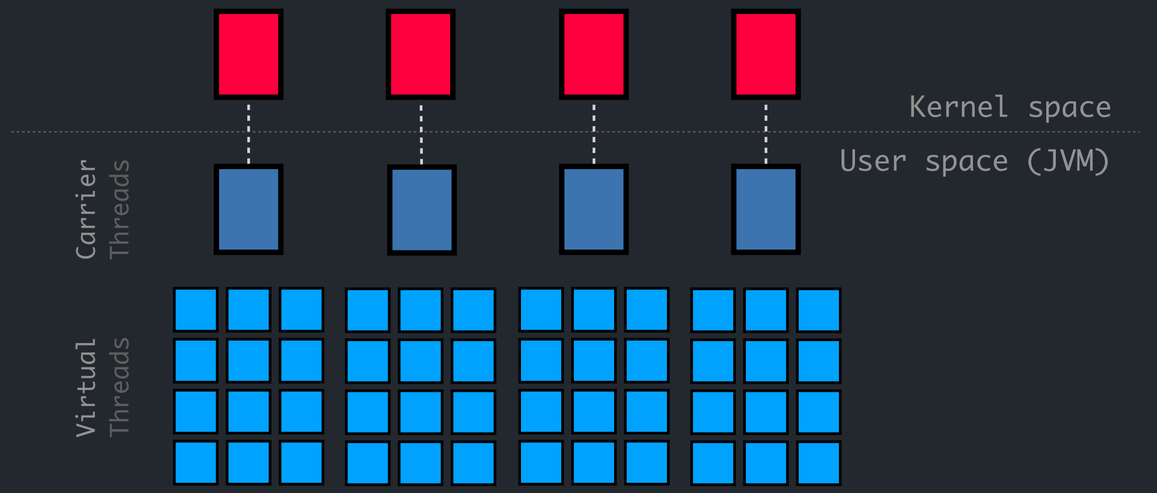
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.



JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

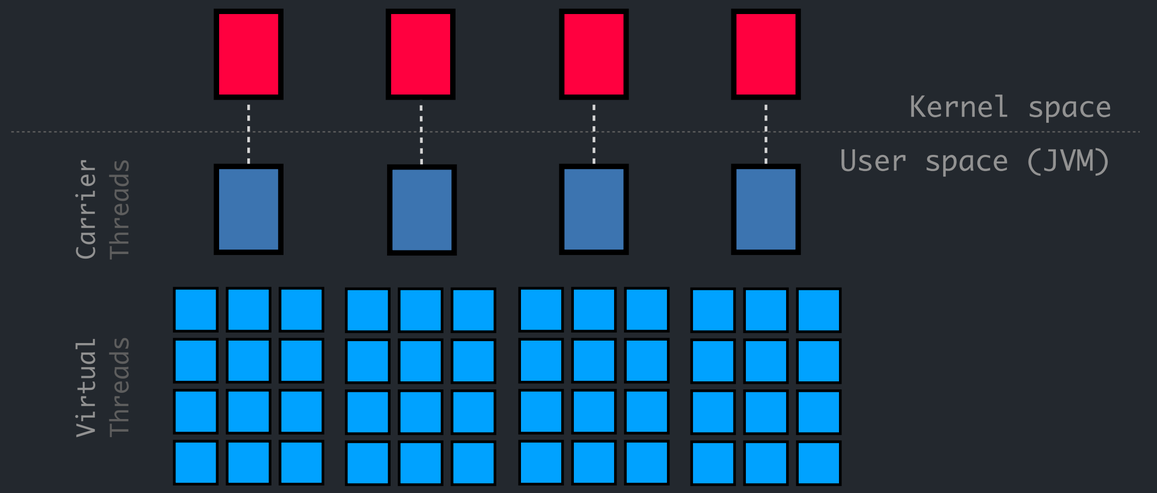
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.



ForkJoinPool with FIFO order

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

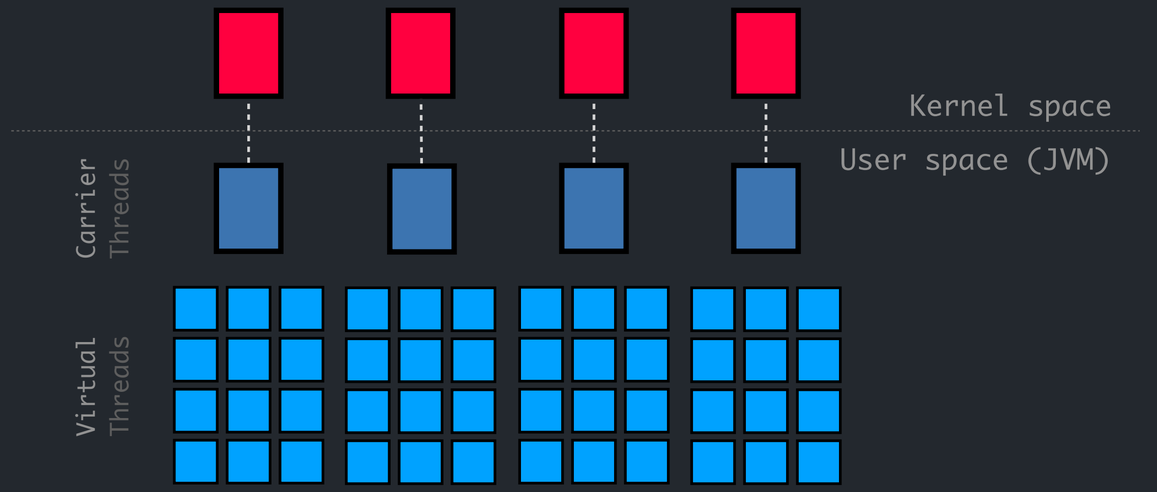
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.



ForkJoinPool with FIFO order

-Djdk.virtualThreadScheduler.parallelism=16

The number of platform threads available for scheduling virtual threads.
It defaults to the number of available processors.

-Djdk.virtualThreadScheduler.maxPoolSize=256

The maximum number of platform threads available to the scheduler.
It defaults to 256.

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

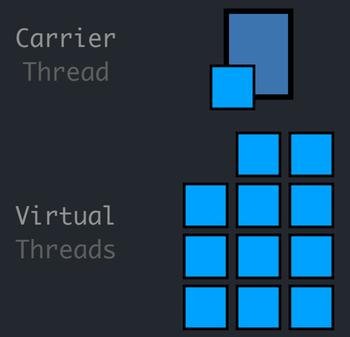
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.



JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

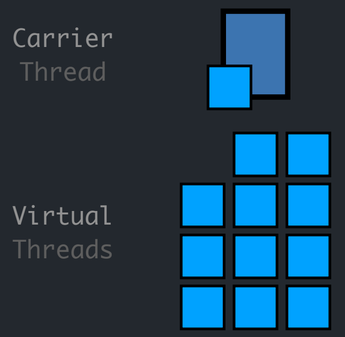
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

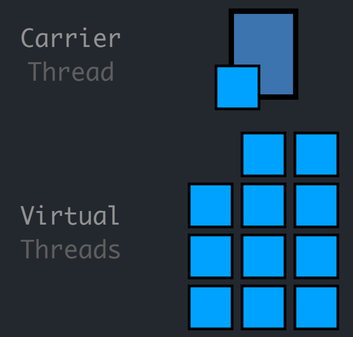
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

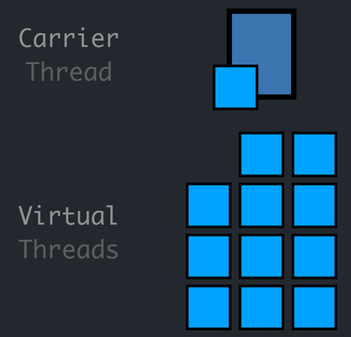
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

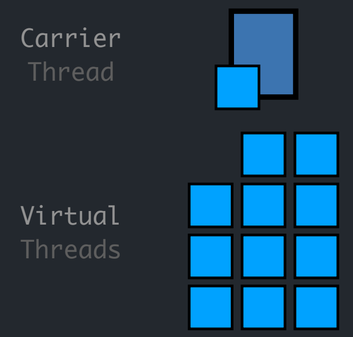
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

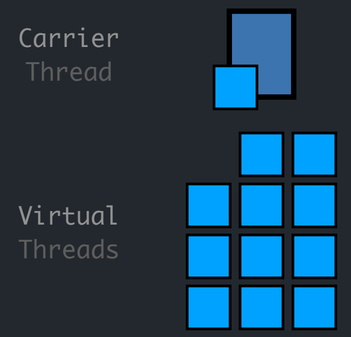
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.


```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

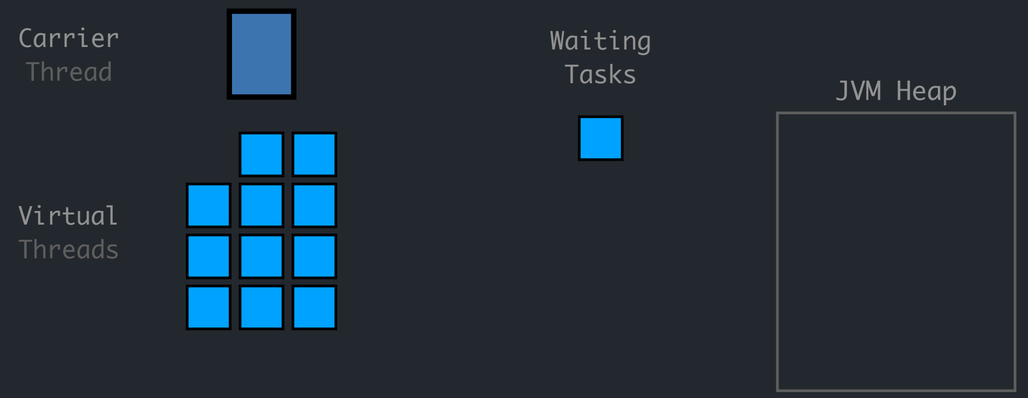
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

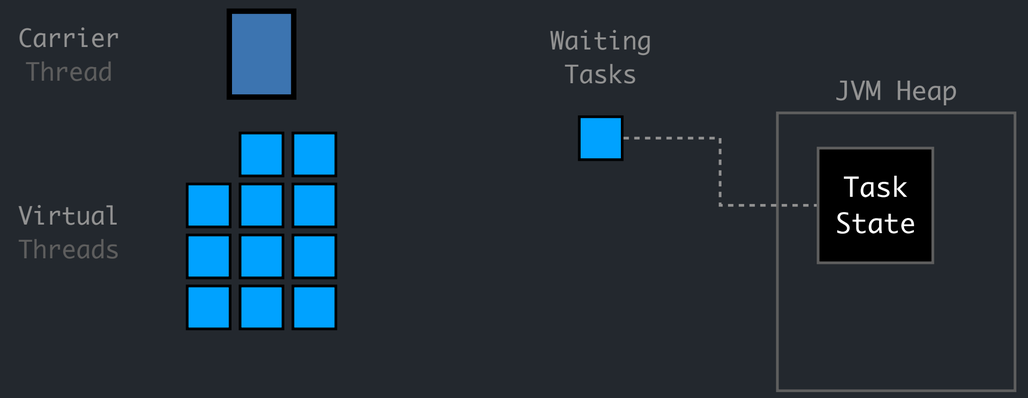
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

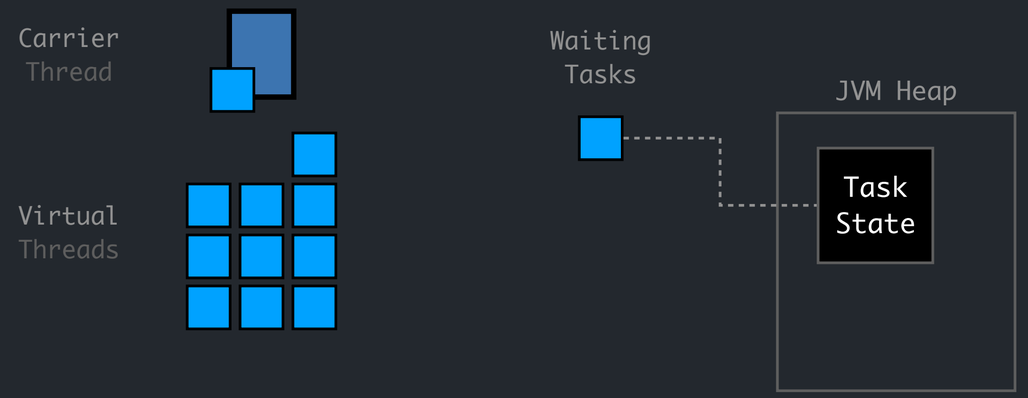
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
   ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

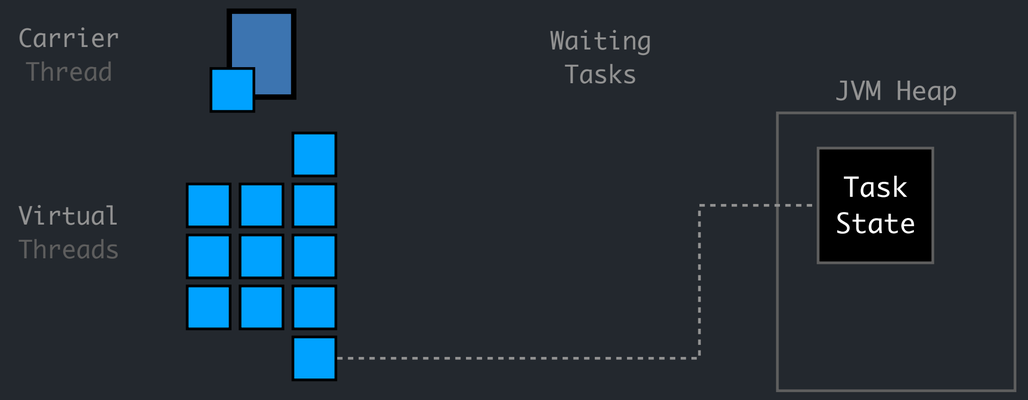
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

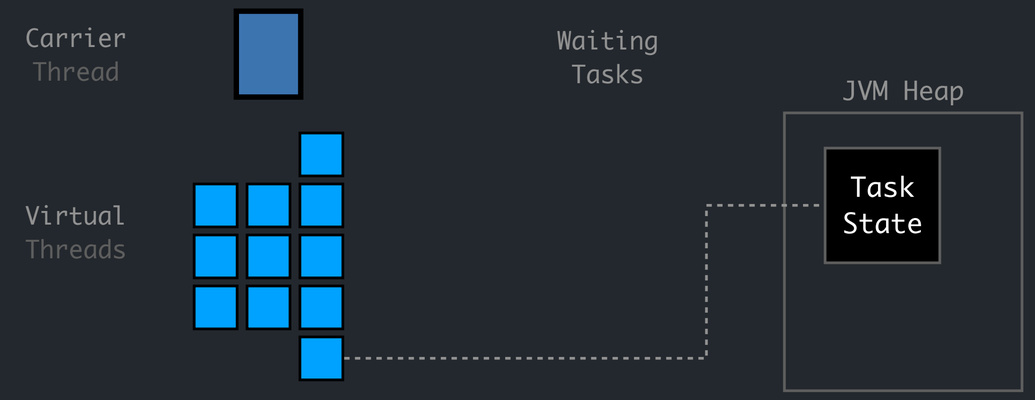
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

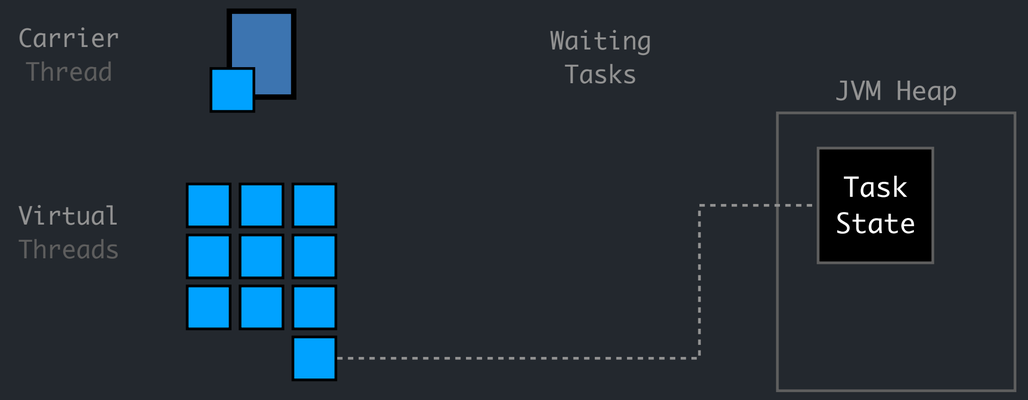
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

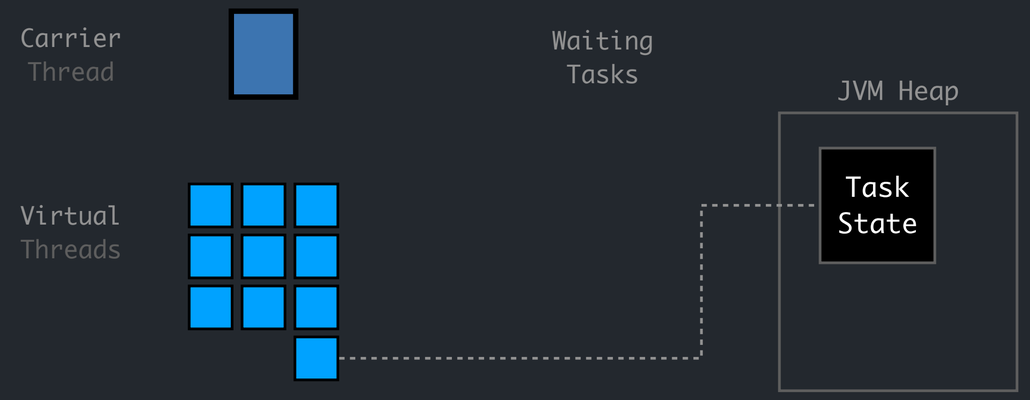
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
   ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

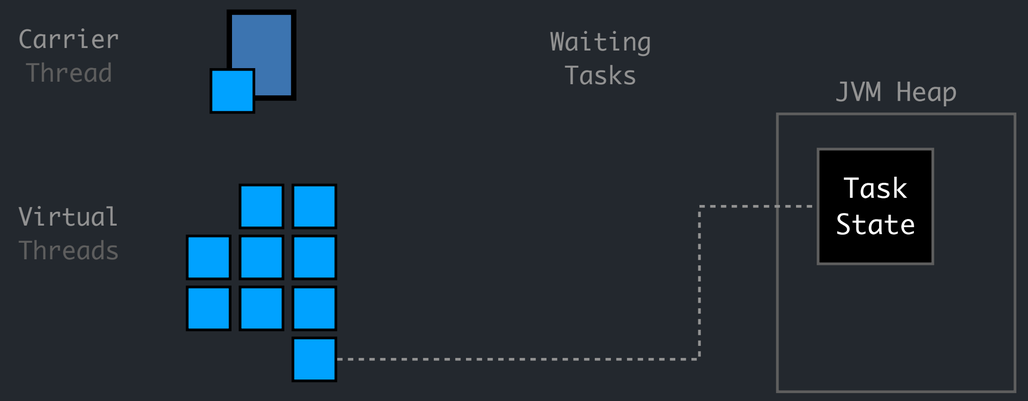
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

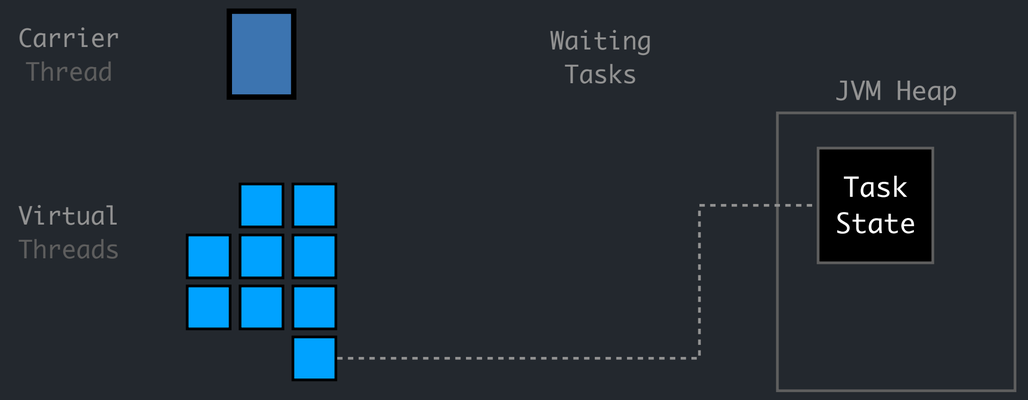
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

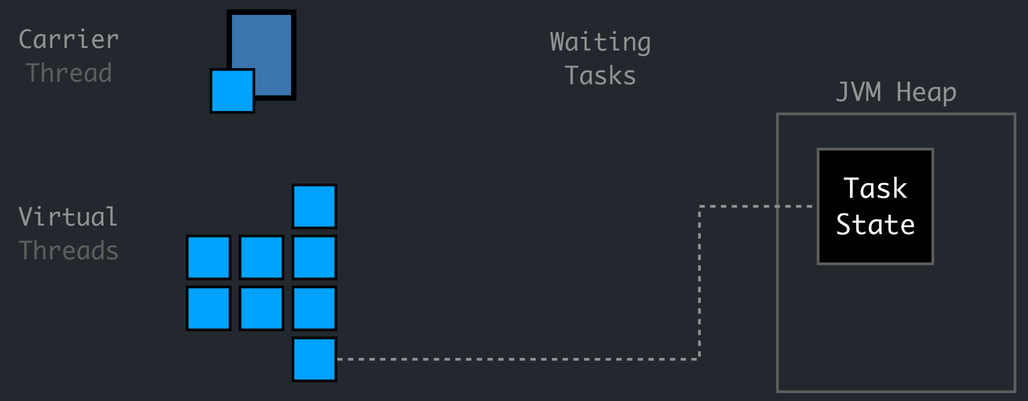
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
   ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

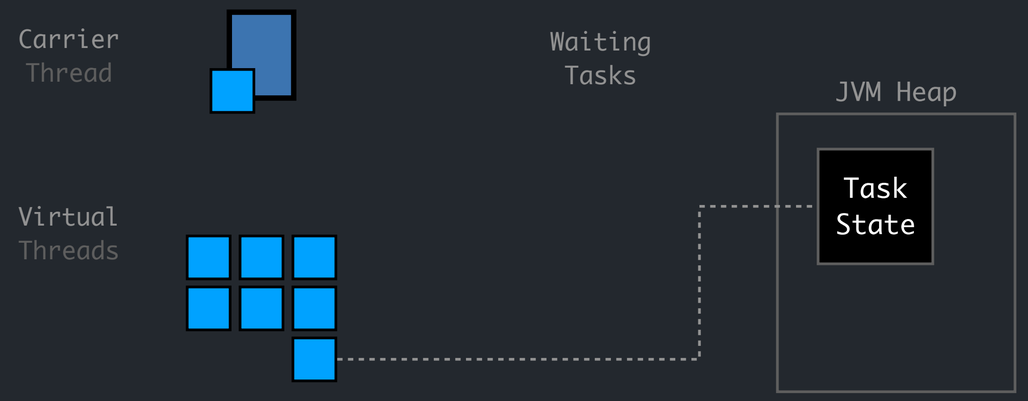
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
   ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

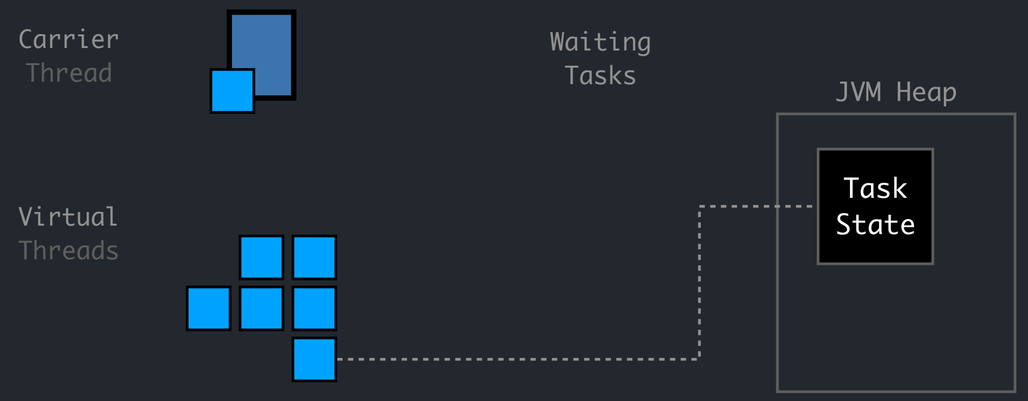
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
   ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

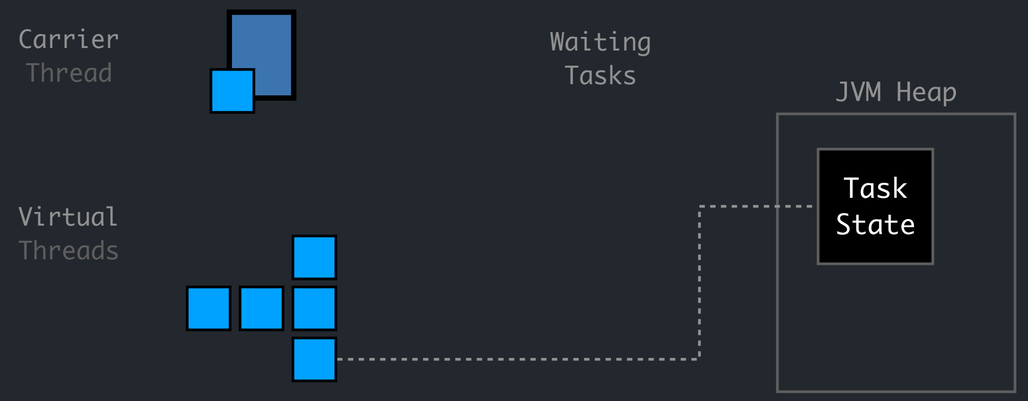
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
   ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

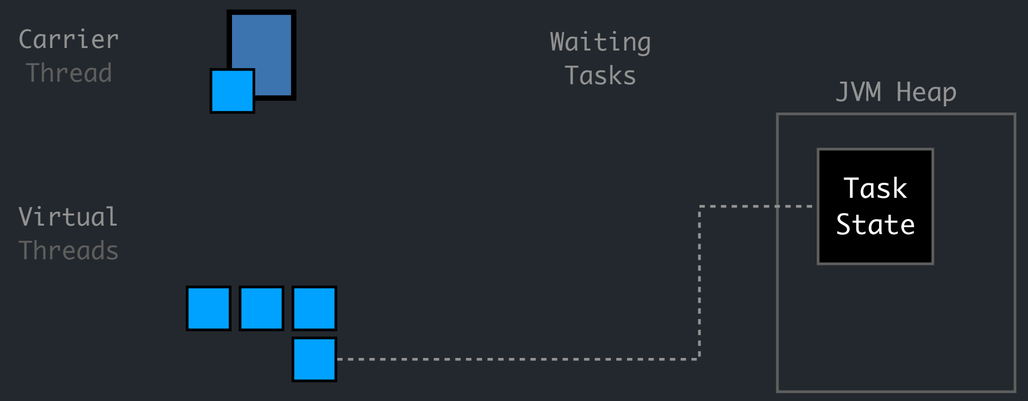
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

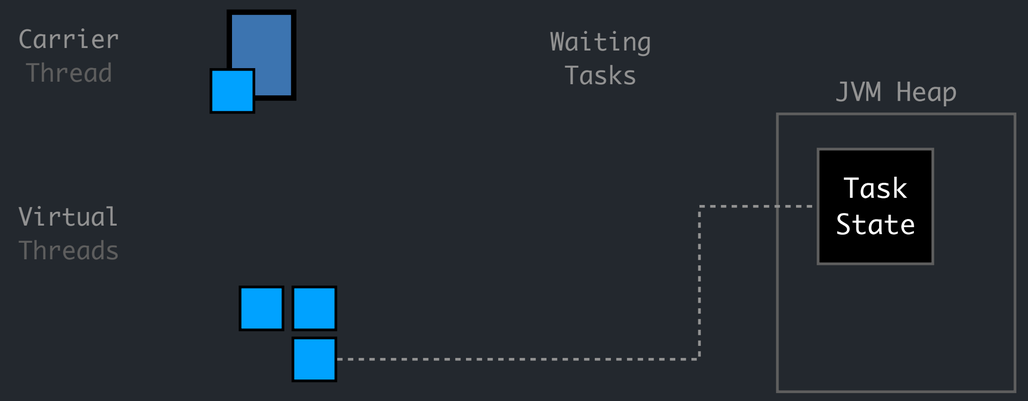
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

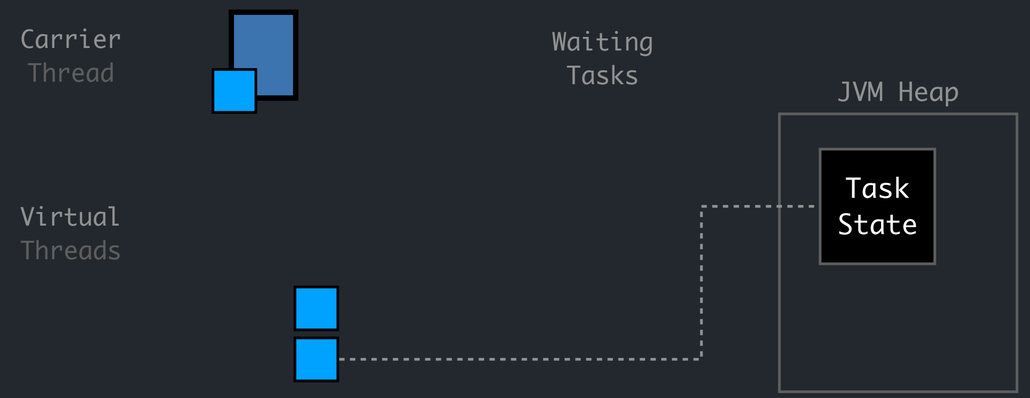
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

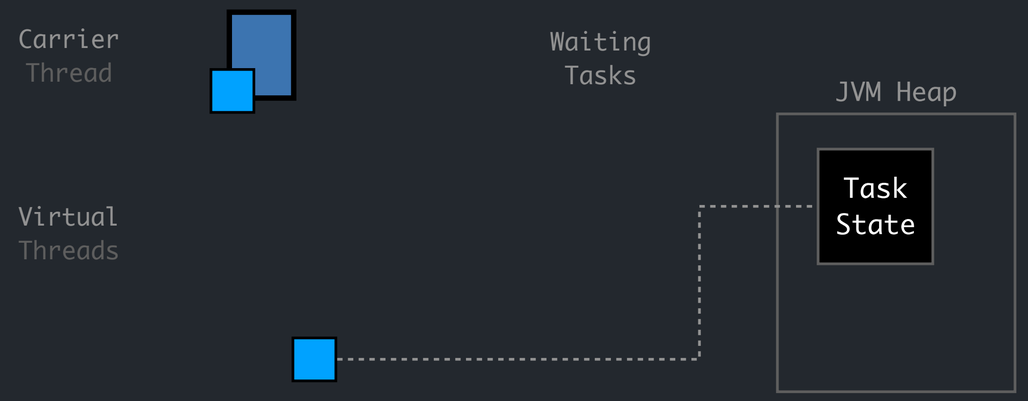
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

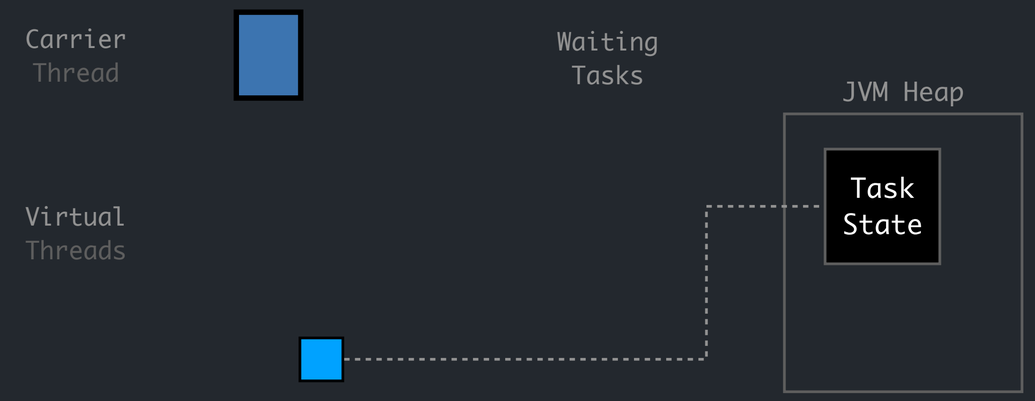
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

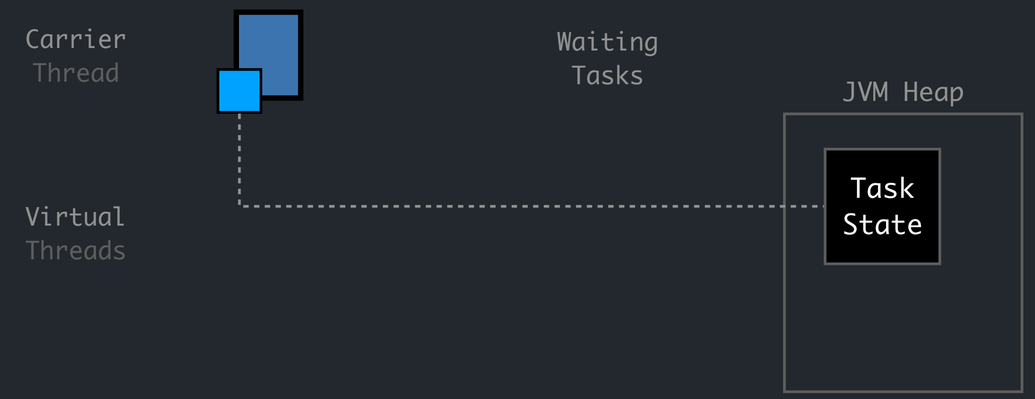
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

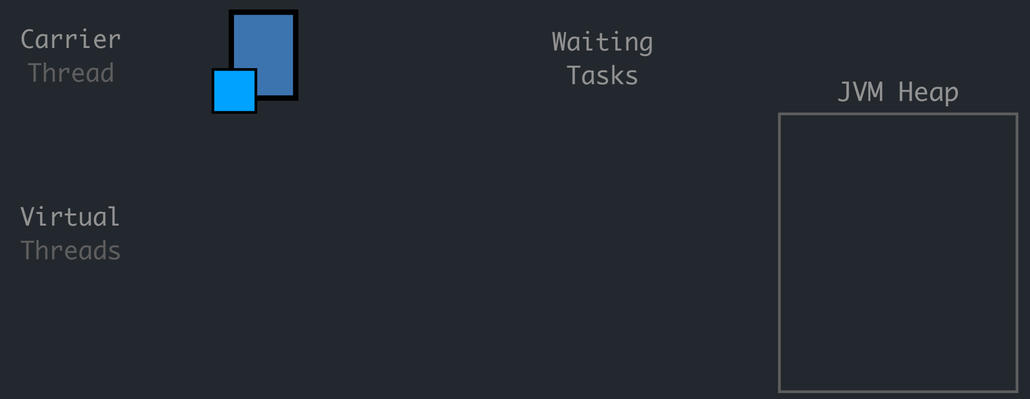
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
   ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

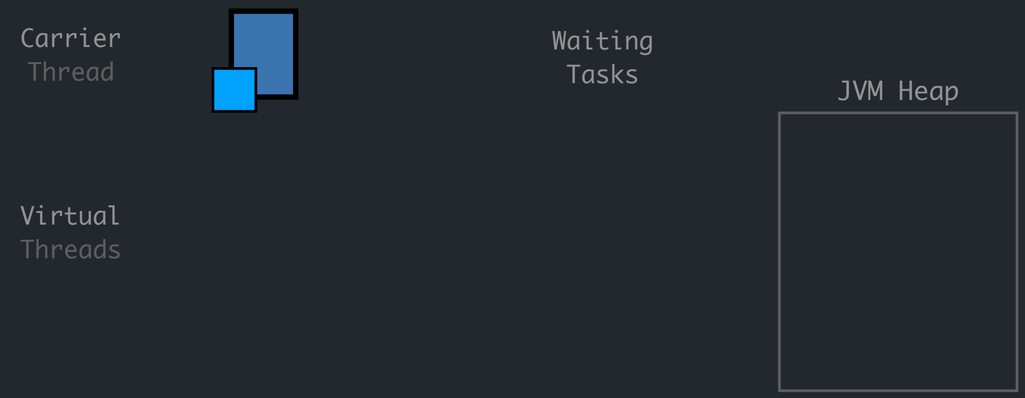
Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.




```
// some cpu bound code
long result = 0;
for (int i = 0; i < 100; ++i) {
  result += i * 3;
}

// some blocking (I/O bound) code
HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
        .uri(URI.create("https://myservice/test"))
        .GET()
        .build(), BodyHandlers.discarding());

// more cpu bound code
if (resp.statusCode() != 200) {
    ...
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.

```
public void myTask() {
  // each execution step may run on a different Carrier Thraed
  cpuBoundOperations(); // run on CarrierThread-5
  ioBoundOperations();  // goes to "waiting"
  cpuBoundOperations(); // run on CarrierThread-3
  ioBoundOperations();  // goes to "waiting"
  cpuBoundOperations(); // run on CarrierThread-3
  // ...
}
```

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.

```
public void myTask() {
  // each execution step may run on a different Carrier Thraed
  cpuBoundOperations(); // run on CarrierThread-5
  ioBoundOperations();  // goes to "waiting"
  cpuBoundOperations(); // run on CarrierThread-3
  ioBoundOperations();  // goes to "waiting"
  cpuBoundOperations(); // run on CarrierThread-3
  // ...
}
```

ThreadLocals are available for compatibility
but not really useful. Do not pool VirtualThreads!

JDK 21: Threads

Platform, Virtual & Carrier Threads

Platform Threads

Typically mapped 1:1 to kernel threads
scheduled by the operating system.

Virtual Threads

User-mode threads
scheduled by the Java Runtime
rather than the operating system

Carrier Threads

Set of Platform threads responsible for
running virtual threads.

```
public void myTask() {
  // each execution step may run on a different Carrier Thraed
  cpuBoundOperations(); // run on CarrierThread-5
  ioBoundOperations();  // goes to "waiting"
  cpuBoundOperations(); // run on CarrierThread-3
  ioBoundOperations();  // goes to "waiting"
  cpuBoundOperations(); // run on CarrierThread-3
  // ...
}
```

ThreadLocals are available for compatibility
but not really useful. Do not pool VirtualThreads!

ScopedValues, the better alternative to ThreadLocal

```
ScopedValue<MyObject> SHARED_OBJECT = ScopedValue.newInstance();
```

are only available as preview (in JDK 21)

Code Examples

Let's see how Virtual Threads behaves

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

We are using the synchronous .send()

No callbacks, futures & co

JDK 21: Virtual Threads

I/O Bound Tasks

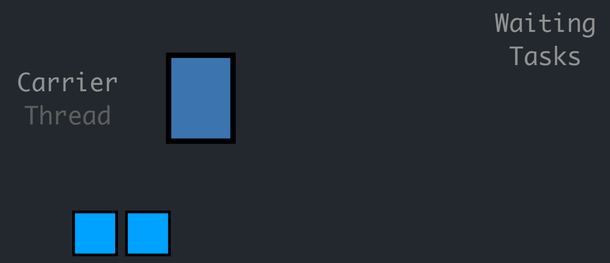
```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

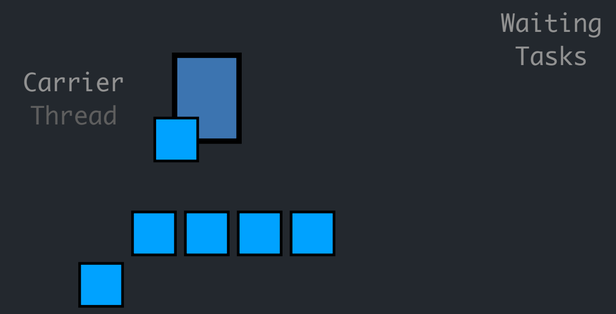
```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

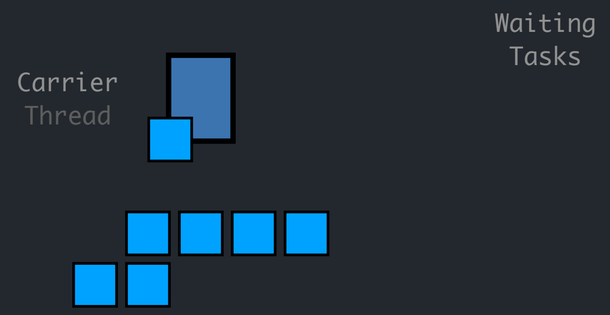



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

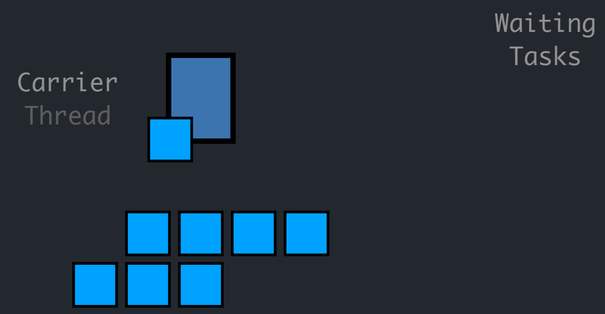



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

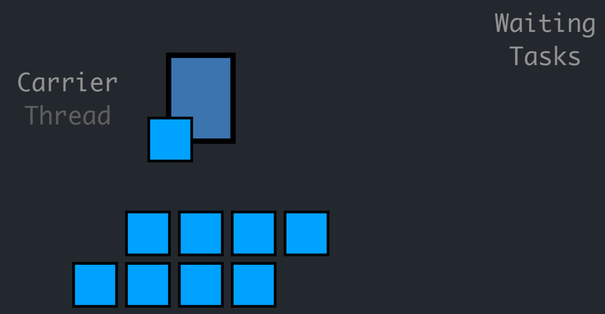



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

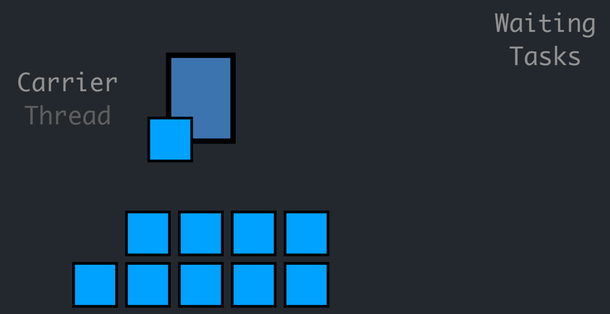



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

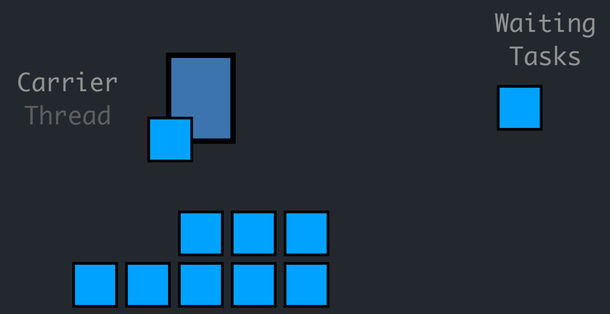
```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

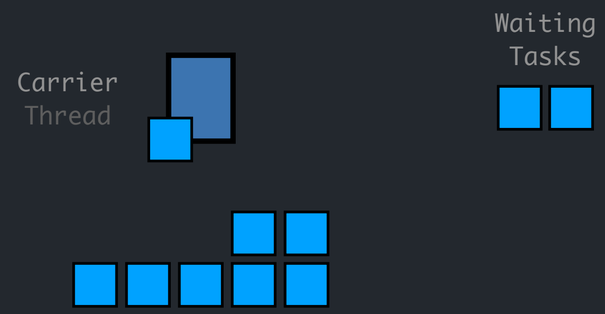
```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

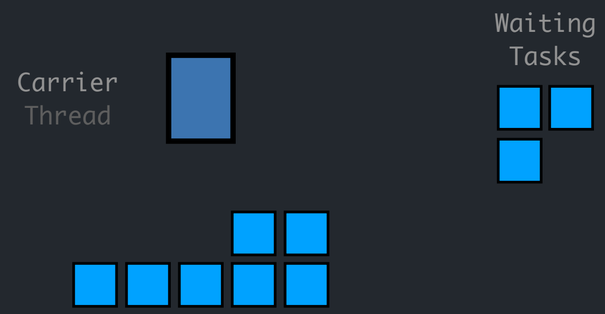



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

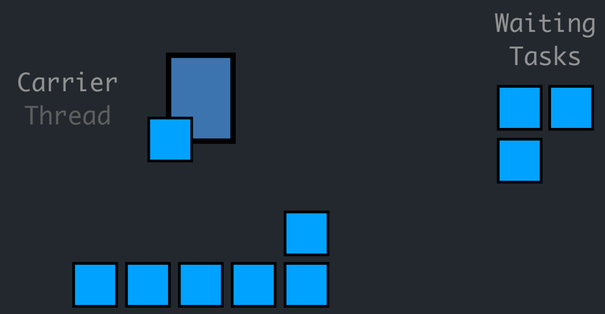



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

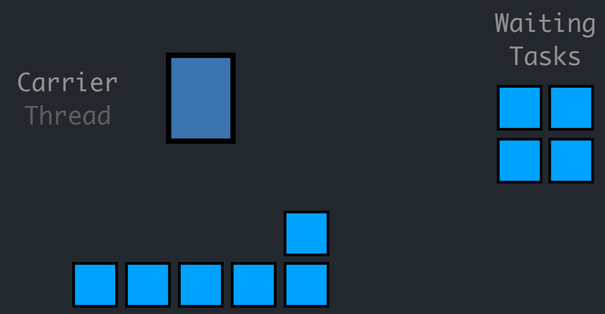



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

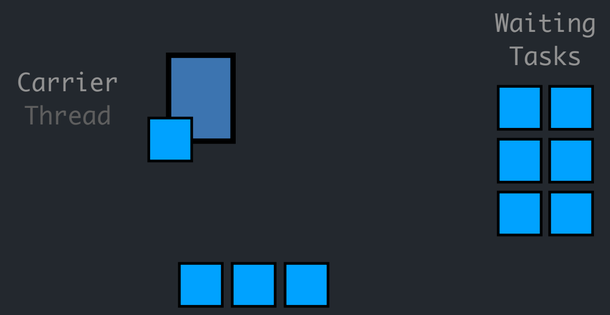
```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

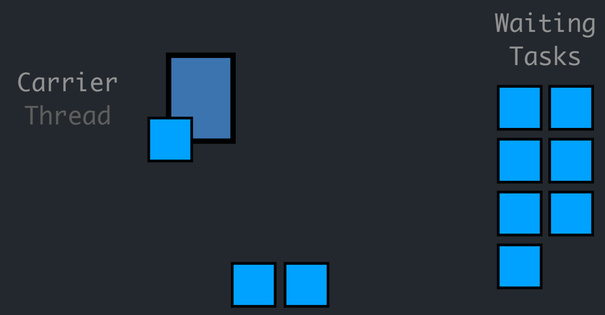
```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

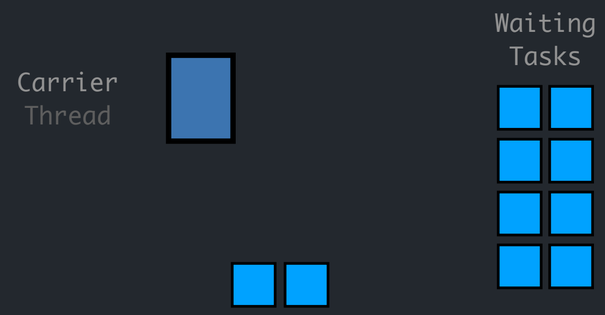



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

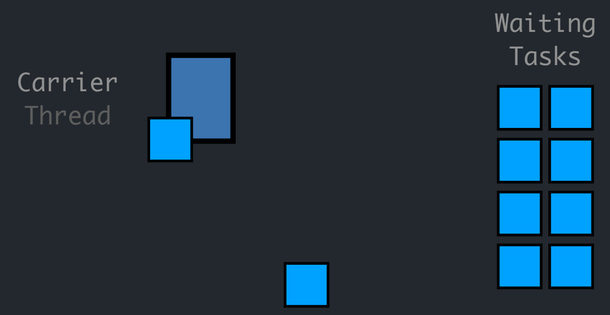



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

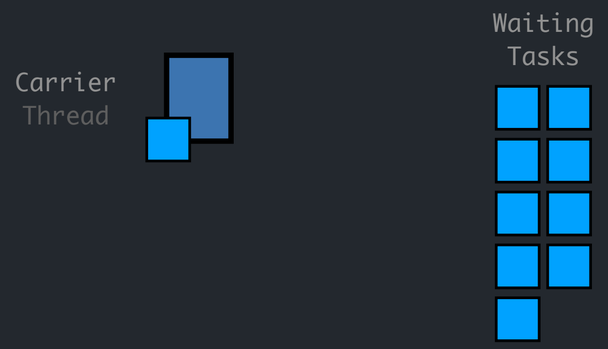
```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```java
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```




```java
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

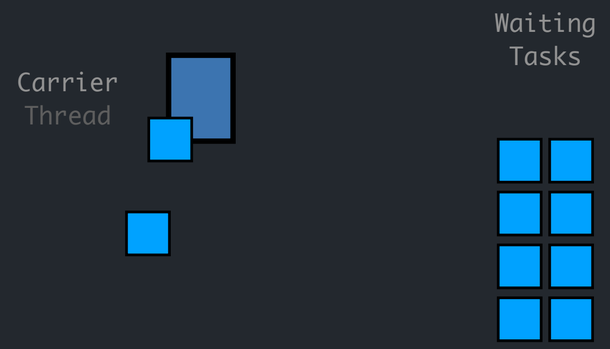
```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

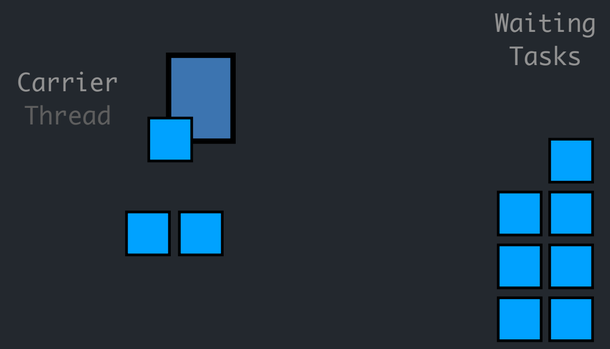



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

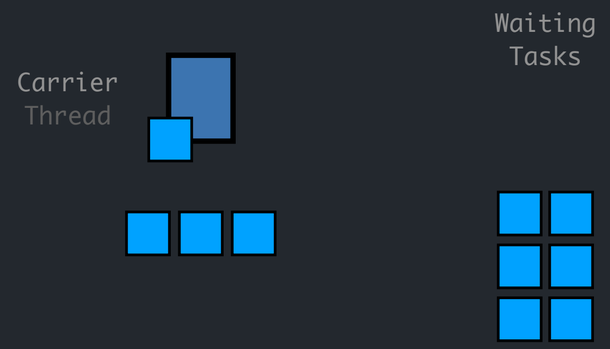



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

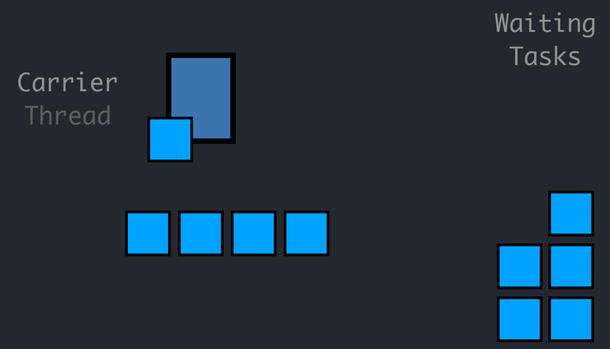



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

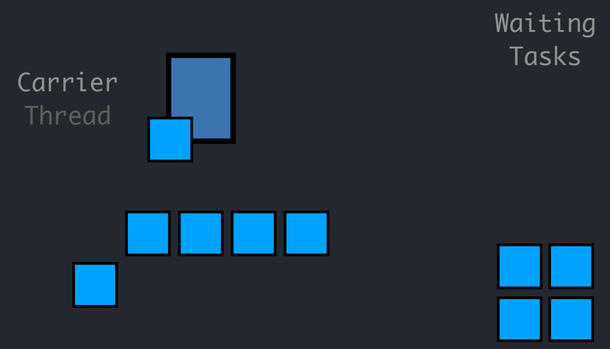



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

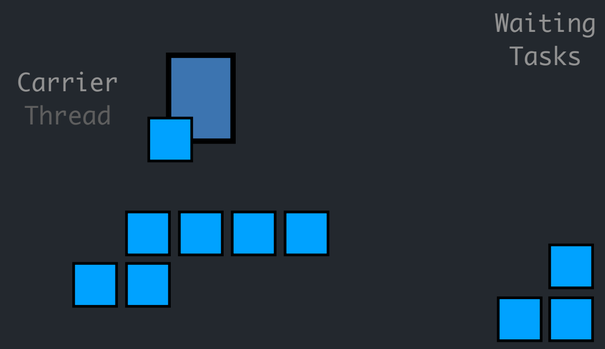



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

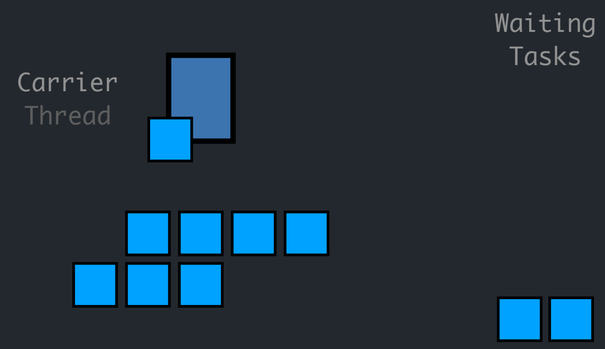



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

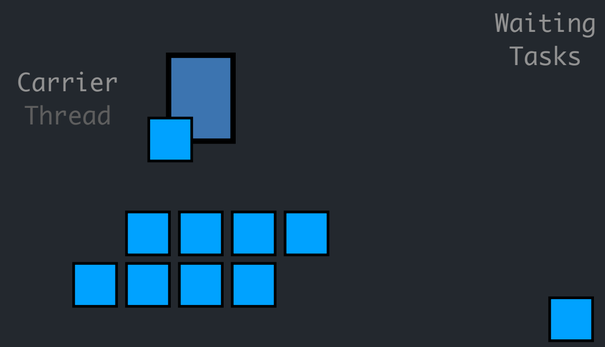



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

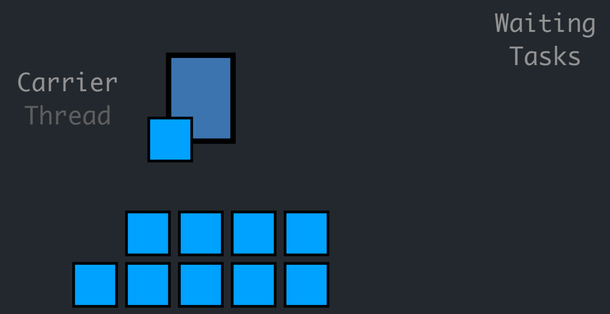



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

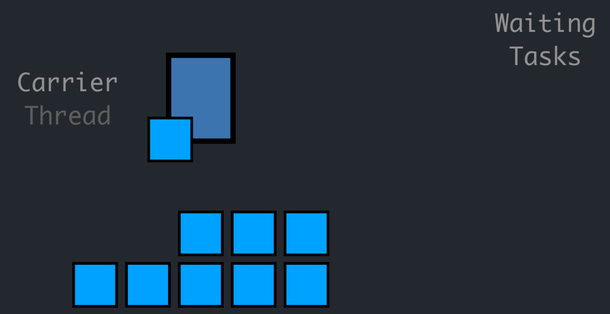
```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

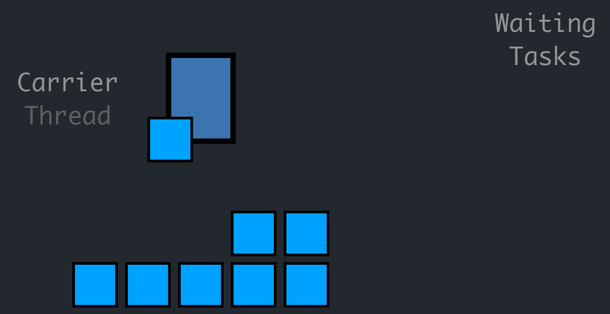
```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

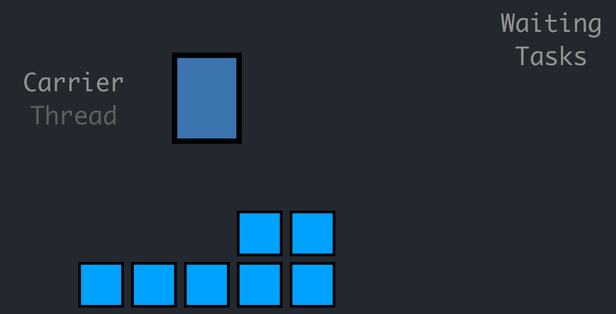



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```

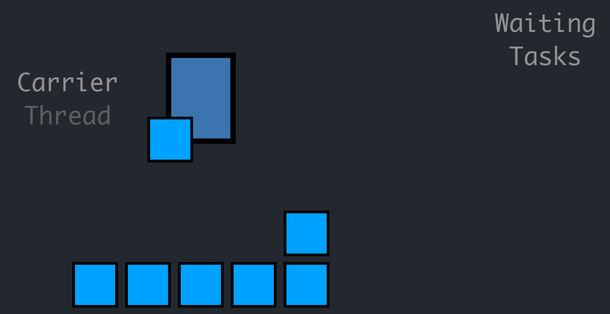



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static long ioBoundTask() {
  try {
    // calls a service that waits 5sec before responding…
    final HttpResponse<Void> resp = httpClient.send(HttpRequest.newBuilder()
            .uri(URI.create("https://hub.dummyapis.com/delay?seconds=5"))
            .GET()
            .build(), BodyHandlers.discarding());
    // ...
    return resp.statusCode();
  } catch (final Exception e) {
    // ignore...
    System.err.println("http req failed: " + e.getMessage());
    return -1;
  }
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

I/O Bound Tasks

```
private static int sleepBoundTask() {
  try {
    // sleep() acts as an I/O operation, allowing the task unmount
    Thread.sleep(5_000);
    return 0;
  } catch (final InterruptedException e) {
    // ignore...
    return -1;
  }
}
```

```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
final Thread[] threads = new Thread[10];
for (int i = 0; i < threads.length; ++i) {
  final int taskId = i;
  threads[i] = Thread.ofVirtual().start(() -> {
      sleepBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
}
for (final Thread thread : threads) {
  thread.join();
  final boolean isVirtual = thread.isVirtual();
}
```

JDK 21: Virtual Threads

CPU Bound Tasks

Virtual Threads are made for I/O Bound Tasks

Let's see how they will behave with CPU Bound Tasks

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```

There are no blocking points in the code
To Unmount the Virtual Thread

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```

1. Completed in 5s

```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```

1. Completed in 5s

```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```

1. Completed in 5s
2. Completed in 5s

```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```

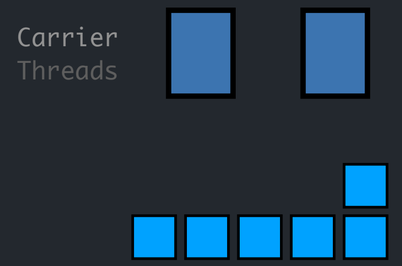
1. Completed in 5s
2. Completed in 5s

```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

1. Completed in 5s
2. Completed in 5s
3. Completed in 10s
4. Completed in 10s

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

1. Completed in 5s
2. Completed in 5s
3. Completed in 10s
4. Completed in 10s

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

1. Completed in 5s
2. Completed in 5s
3. Completed in 10s
4. Completed in 10s
5. Completed in 15s
6. Completed in 15s

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```

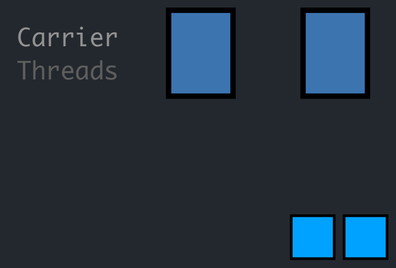

```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

1. Completed in 5s
2. Completed in 5s
3. Completed in 10s
4. Completed in 10s
5. Completed in 15s
6. Completed in 15s

JDK 21: Virtual Threads

CPU Bound Tasks

```java
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```




```java
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

1. Completed in 5s
2. Completed in 5s
3. Completed in 10s
4. Completed in 10s
5. Completed in 15s
6. Completed in 15s
7. Completed in 20s
8. Completed in 20s

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

1. Completed in 5s
2. Completed in 5s
3. Completed in 10s
4. Completed in 10s
5. Completed in 15s
6. Completed in 15s
7. Completed in 20s
8. Completed in 20s

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

1. Completed in 5s
2. Completed in 5s
3. Completed in 10s
4. Completed in 10s
5. Completed in 15s
6. Completed in 15s
7. Completed in 20s
8. Completed in 20s
9. Completed in 25s
10. Completed in 25s

JDK 21: Virtual Threads

CPU Bound Tasks

```
private static long cpuBoundTask() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    count++;
  }
  return count;
}
```

For CPU bound Tasks
Virtual Threads gives no advantage
over traditional
(Platform Thread based)
ExecutorServices

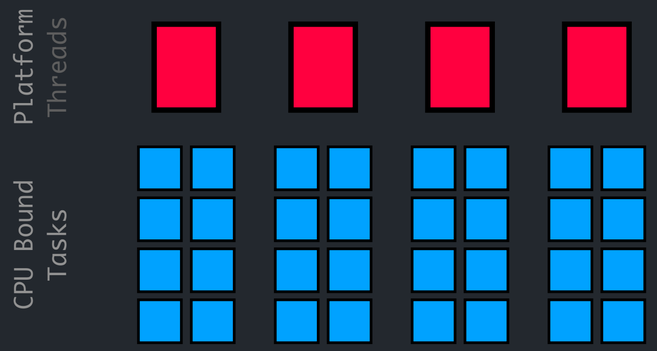
```
// Run with -Djdk.virtualThreadScheduler.parallelism=2
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

And I/O Bound Tasks
Have to wait
CPU Bound Tasks to Finish

JDK 21: Virtual Threads

Send CPU and I/O bound tasks to dedicated executors

ThreadPoolExecutor



VirtualThraedPerTaskExecutor

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

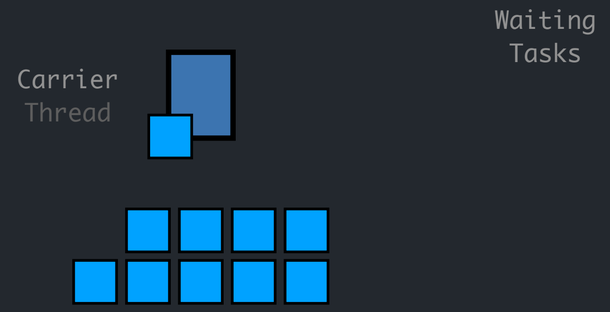
CPU Bound Tasks
can cooperate
Using the yield() method

```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

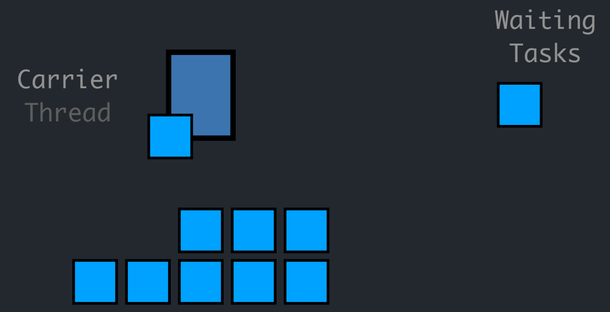
```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

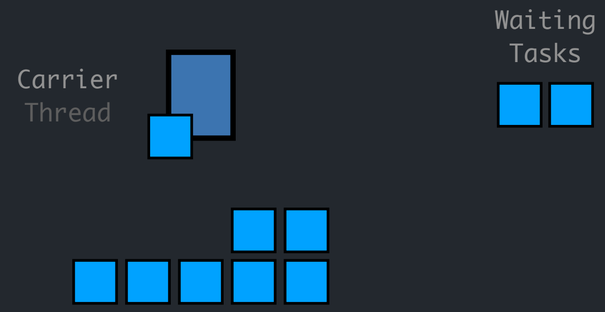
```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

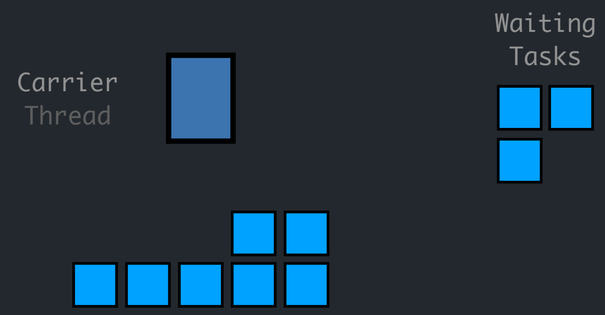



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

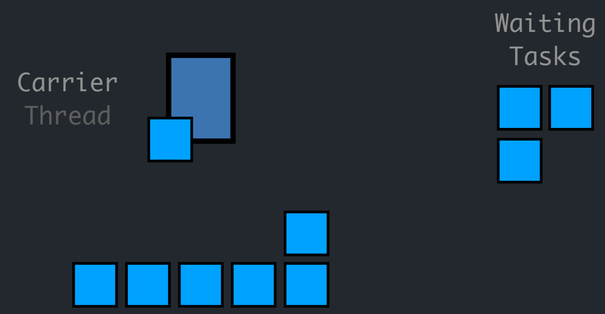



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

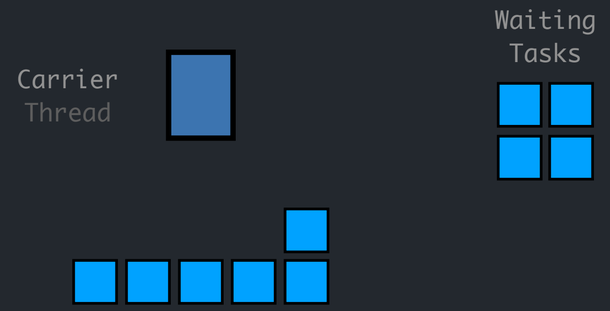



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

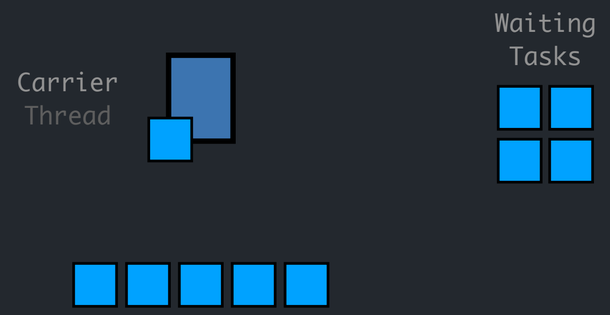



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

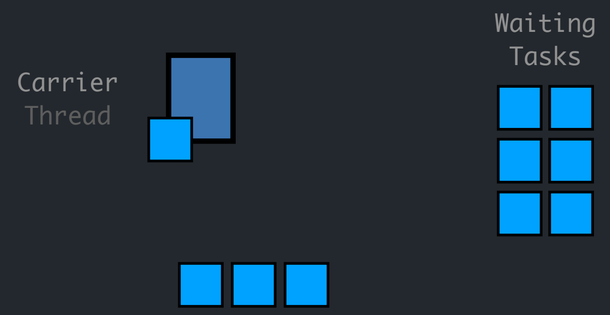
```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

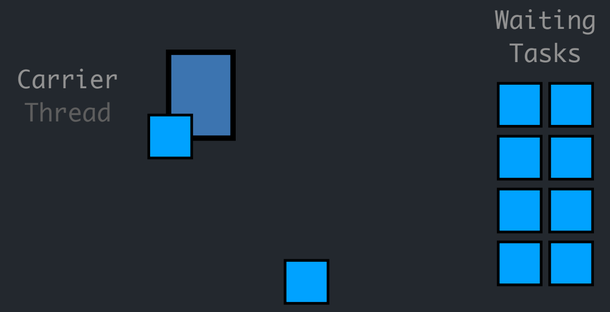
```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

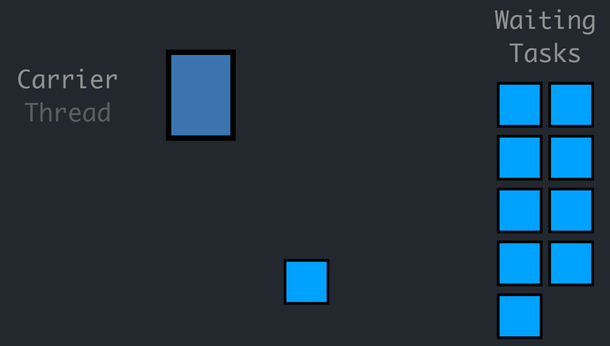



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

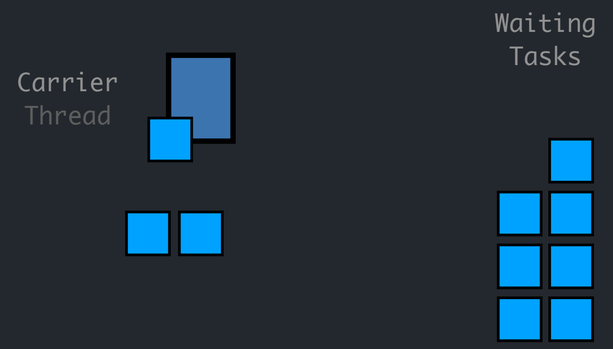
```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

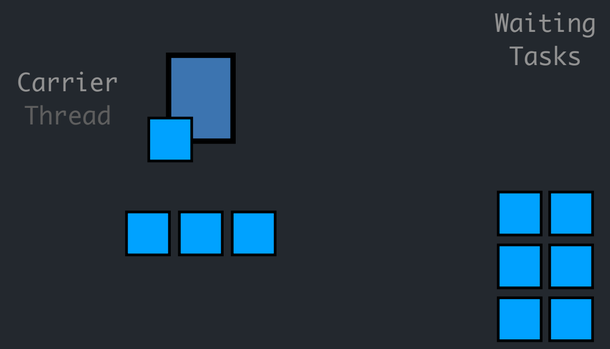



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

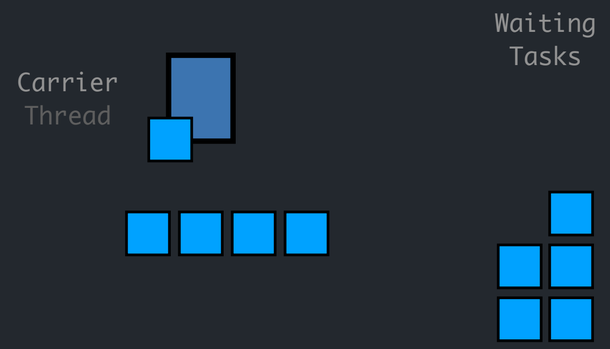



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

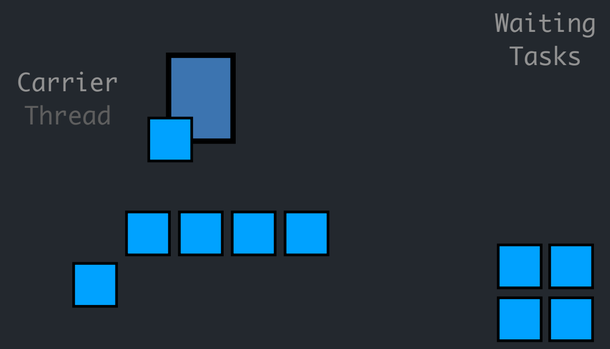



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

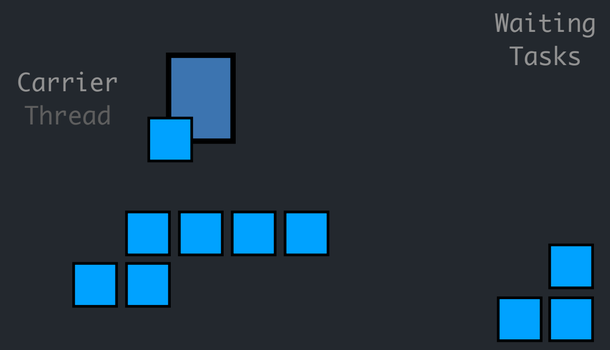



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

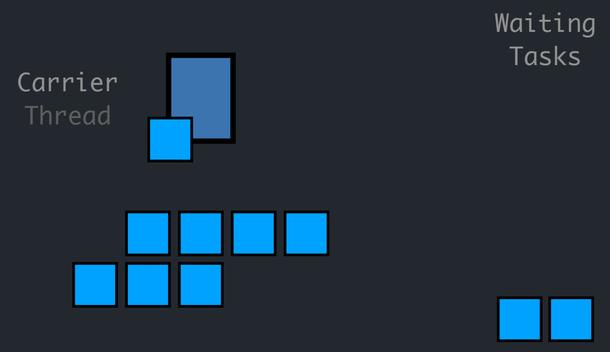



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

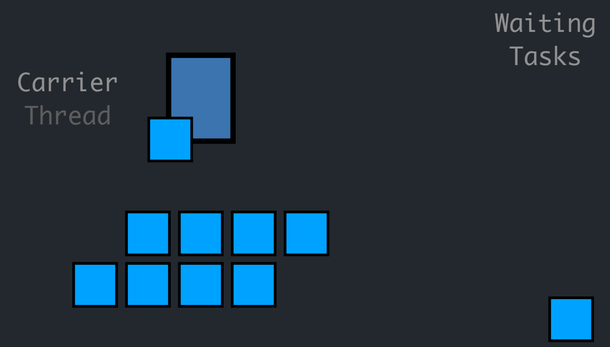



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

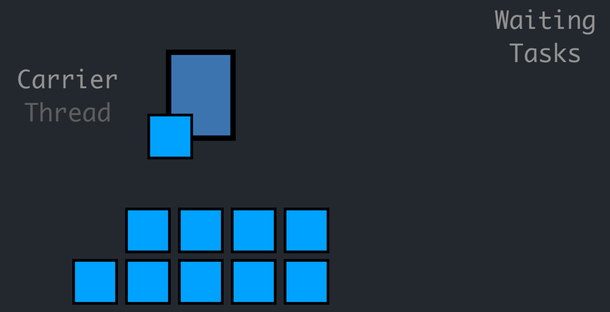



```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

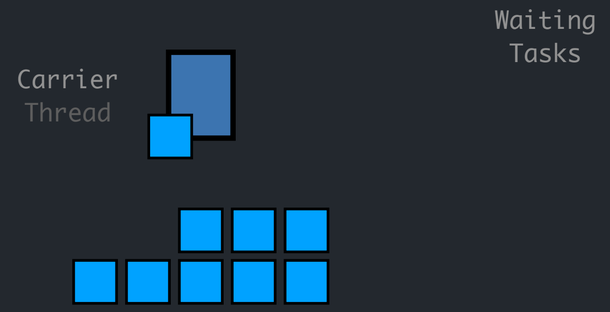
```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

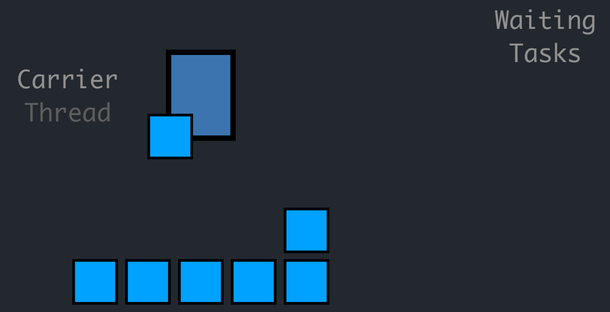
```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```


```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```




```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

Cooperative CPU Bound Tasks

```
private static long cpuBoundTaskWithYield() {
  final long startTime = System.nanoTime();
  long count = 0;
  while ((System.nanoTime() - startTime) < TimeUnit.SECONDS.toNanos(5)) {
    if (count++ % 100 == 0) {
      Thread.yield();
    }
  }
  return count;
}
```

Slower than running the task alone
(Since the CPU is shared)

But it’s the same behaviour
of running multiple platform Threads

```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      cpuBoundTaskWithYield();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21: Virtual Threads

The synchronized keyword

```
private final static Object[] lock = new Object[4];
static {
  for (int i = 0; i < lock.length; ++i) {
    lock[i] = new Object();
  }
}

private static long ioSynchronizedTask(final int taskId) {
  synchronized (lock[taskId % lock.length]) {
    return ioBoundTask();
  }
}
```

SYNCHRONIZED
WILL BLOCK
THE CARRIER THREAD

JDK 21: Virtual Threads

The synchronized keyword

```
private final static Object[] lock = new Object[4];
static {
  for (int i = 0; i < lock.length; ++i) {
    lock[i] = new Object();
  }
}

private static long ioSynchronizedTask(final int taskId) {
  synchronized (lock[taskId % lock.length]) {
    return ioBoundTask();
  }
}
```

Multiple synchronized Object
Will not help

If you are lucky
You'll get the
CPU Bound Task Behaviour
Even on I/O Operations

JDK 21: Virtual Threads

The synchronized keyword

```
private final static Object[] lock = new Object[4];
static {
  for (int i = 0; i < lock.length; ++i) {
    lock[i] = new Object();
  }
}

private static long ioSynchronizedTask(final int taskId) {
  synchronized (lock[taskId % lock.length]) {
    return ioBoundTask();
  }
}
```

Remove the use of Synchronized

JDK 21: Virtual Threads

Locks: Reentrant Lock, Semaphores, ...

```
private static final ReentrantLock[] rlock = new ReentrantLock[4];
static {
  for (int i = 0; i < rlock.length; ++i) {
    rlock[i] = new ReentrantLock();
  }
}

private static long ioLockedTask(final int taskId) {
  final ReentrantLock lock = rlock[taskId % rlock.length];
  lock.lock();
  try {
    return ioBoundTask(); // http.send(), sleep(), ...
  } finally {
    lock.unlock();
  }
}
```

4 Tasks will run concurrently

Locks are a blocking operation
Releasing the Virtual Thread from the Carrier

```
// Run with -Djdk.virtualThreadScheduler.parallelism=1
final long startTime = System.nanoTime();
try (ExecutorService executor = Executors.newVirtualThreadPerTaskExecutor()) {
  for (int i = 0; i < 10; ++i) {
    final int taskId = i;
    executor.submit(() -> {
      ioLockedTask();
      long elapsedNs = System.nanoTime() - startTime;
    });
  }
}
```

JDK 21

Virtual Threads

Do not go and replace all your ExecutorServices with Virtual Threads!

CPU Bound Tasks will not benefit from it

Virtual Threads are made for I/O bound Tasks

	CPU Bound	I/O Bound
	Spends most of the time using CPU	Spends most of the time waiting for I/O
	Longer CPU bursts	Shorter CPU bursts

JDK 21

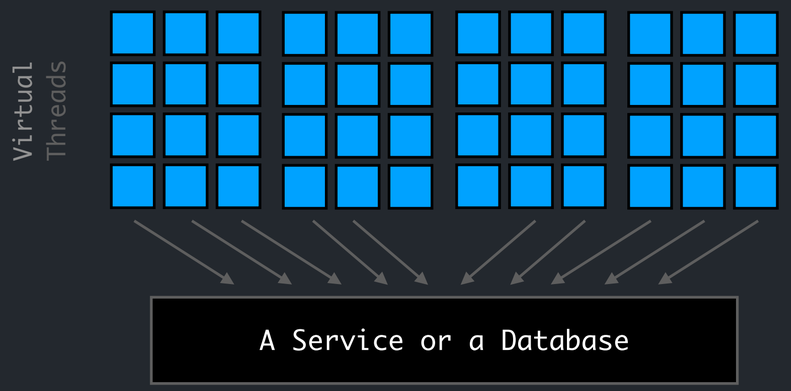
Virtual Threads

Do not go and replace all your ExecutorServices with Virtual Threads!

CPU Bound Tasks will not benefit from it

Virtual Threads are made for I/O bound Tasks

	CPU Bound	I/O Bound
	Spends most of the time using CPU	Spends most of the time waiting for I/O
	Longer CPU bursts	Shorter CPU bursts



JDK 21

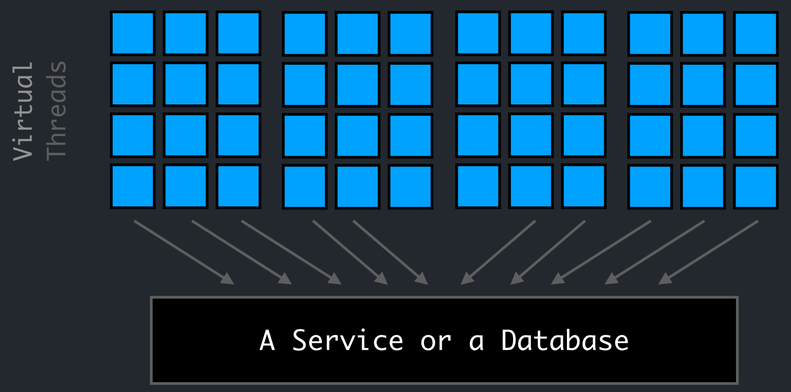
Virtual Threads

Do not go and replace all your ExecutorServices with Virtual Threads!

CPU Bound Tasks will not benefit from it

Virtual Threads are made for I/O bound Tasks

	CPU Bound	I/O Bound
	Spends most of the time using CPU	Spends most of the time waiting for I/O
	Longer CPU bursts	Shorter CPU bursts



Can your Database, Service, Disk, …
handle the extra concurrency?
(Probably not)

If you are using Virtual Threads
You should limit the access
To a Resource
Using a Semaphore (or similar)