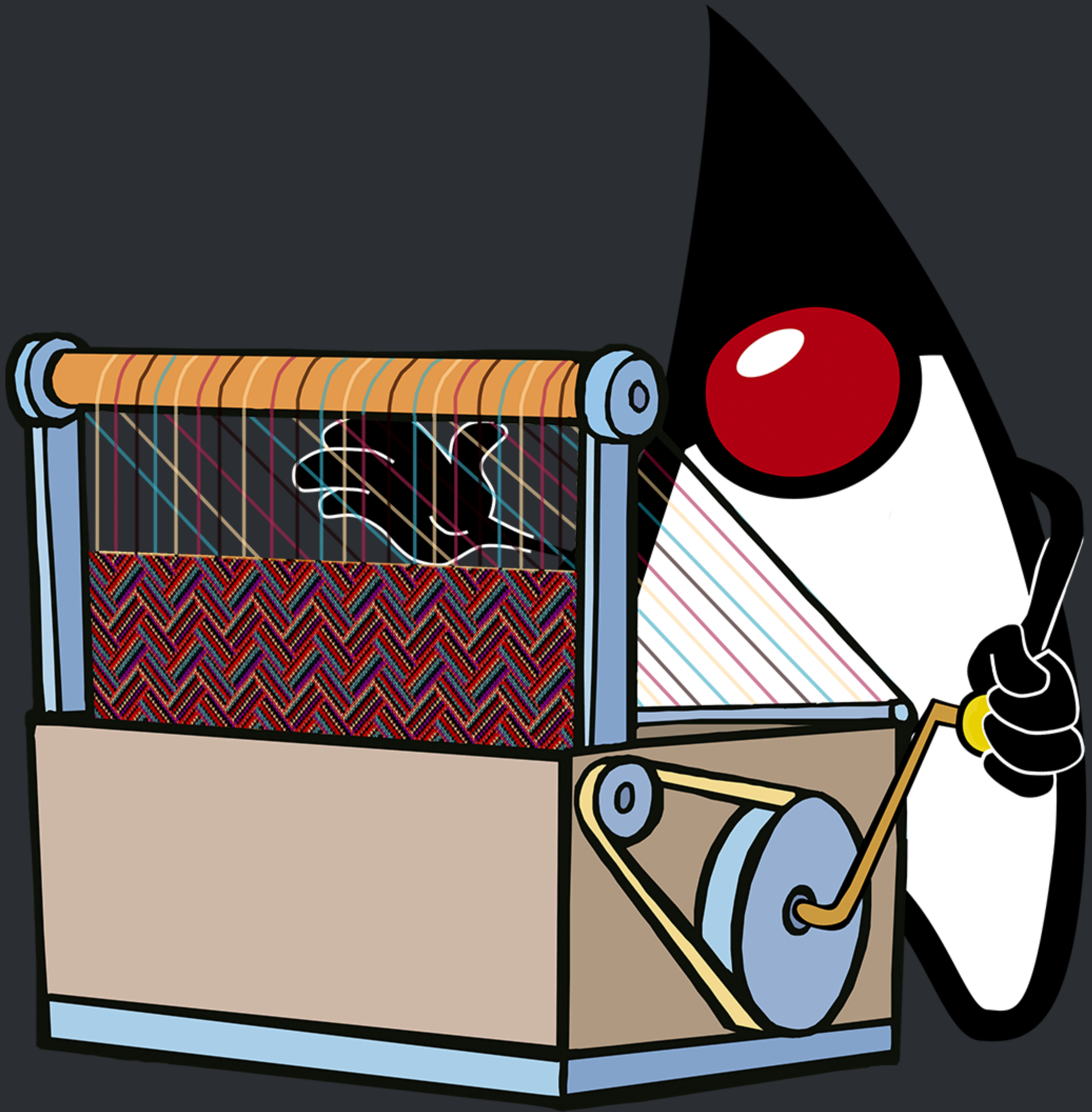
ORACLE

OpenJDK™

Project Loom: Modern scalable concurrency for the Java platform

Alan Bateman
Java Platform Group
November 27, 2020



Java is made of Threads

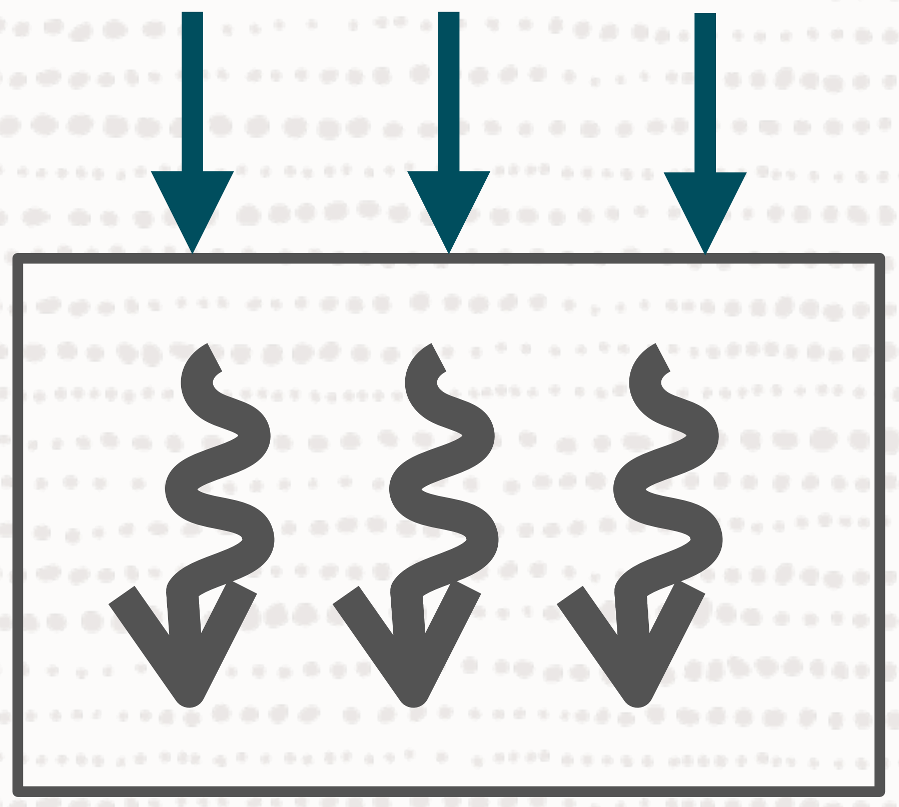
- Exceptions
- Thread Locals
- Debugger
- Profiler

Threads in Java

- java.lang.Thread
- One implementation based on OS thread
- OS threads support all languages
- Large fixed stacks
- Task-switching requires switch to kernel
- Scheduling is a compromise for all usages

Synchronous

- Easy to read
- Fits well with the Java Language
  - control flow, exceptions, …
- Fits well with tooling (debuggers, profilers)
- But a costly resource



Programmer 😀

OS / Hardware ☹️

Reuse with thread pools

Reuse with thread pools

- Return at end
  - May leak thread locals
  - Problematic cancellation

Reuse with thread pools

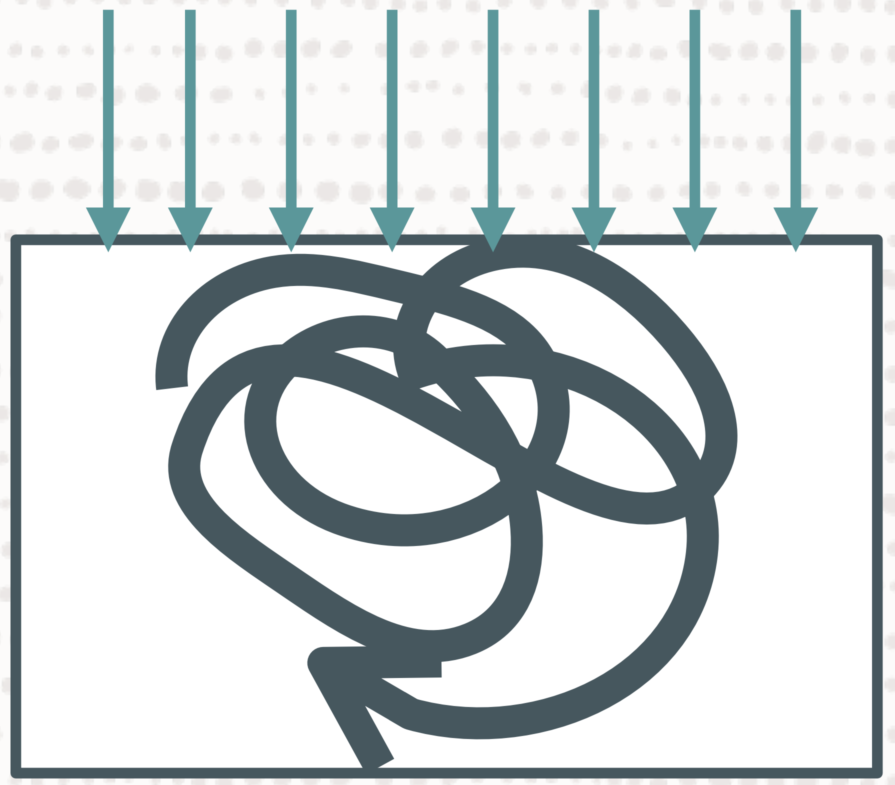
- Return at end
  - May leak thread locals
  - Complex cancellation
- Return at waiting/blocking points
  - Incomplete APIs
  - Lost context
  - Intrusive, nearly impossible to migrate

Asynchronous

- Scalable

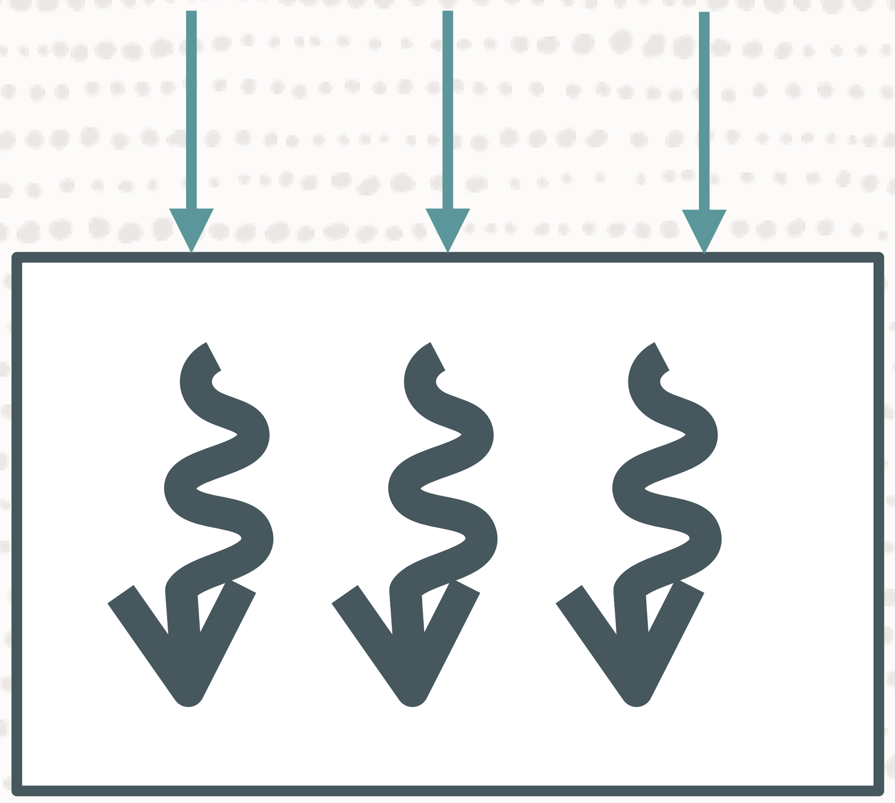
But

- Hard to read
- Lost context so hard to debug and profile



Programmer ☹️

OS / Hardware 😀



simple

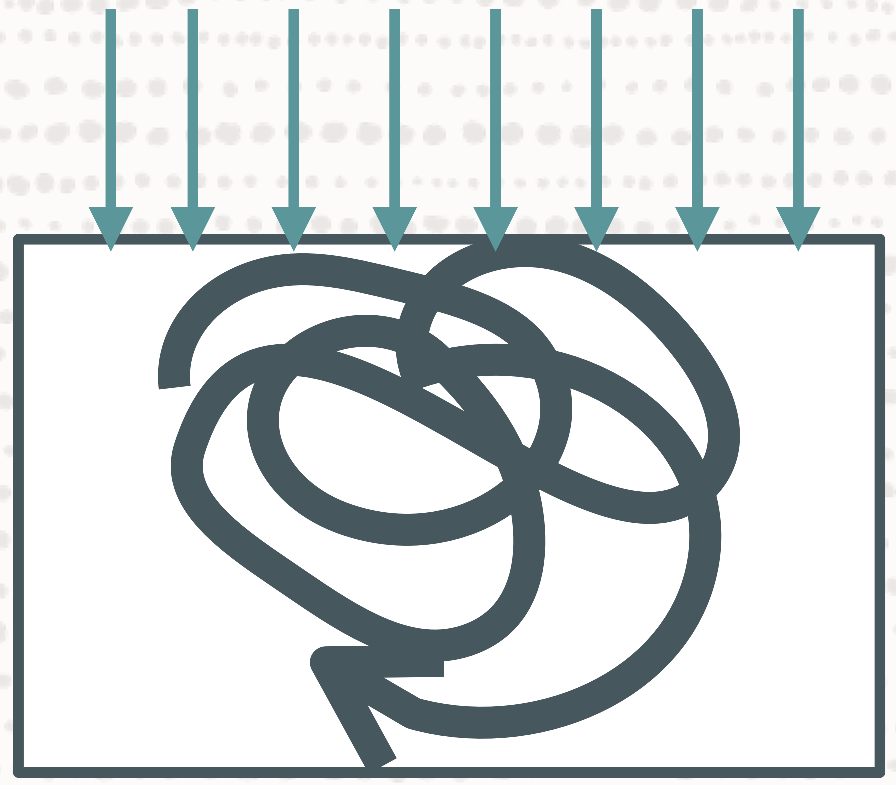
less scalable

SYNC

Programmer 😀

OS / Hardware ☹️

OR



scalable,

complex,

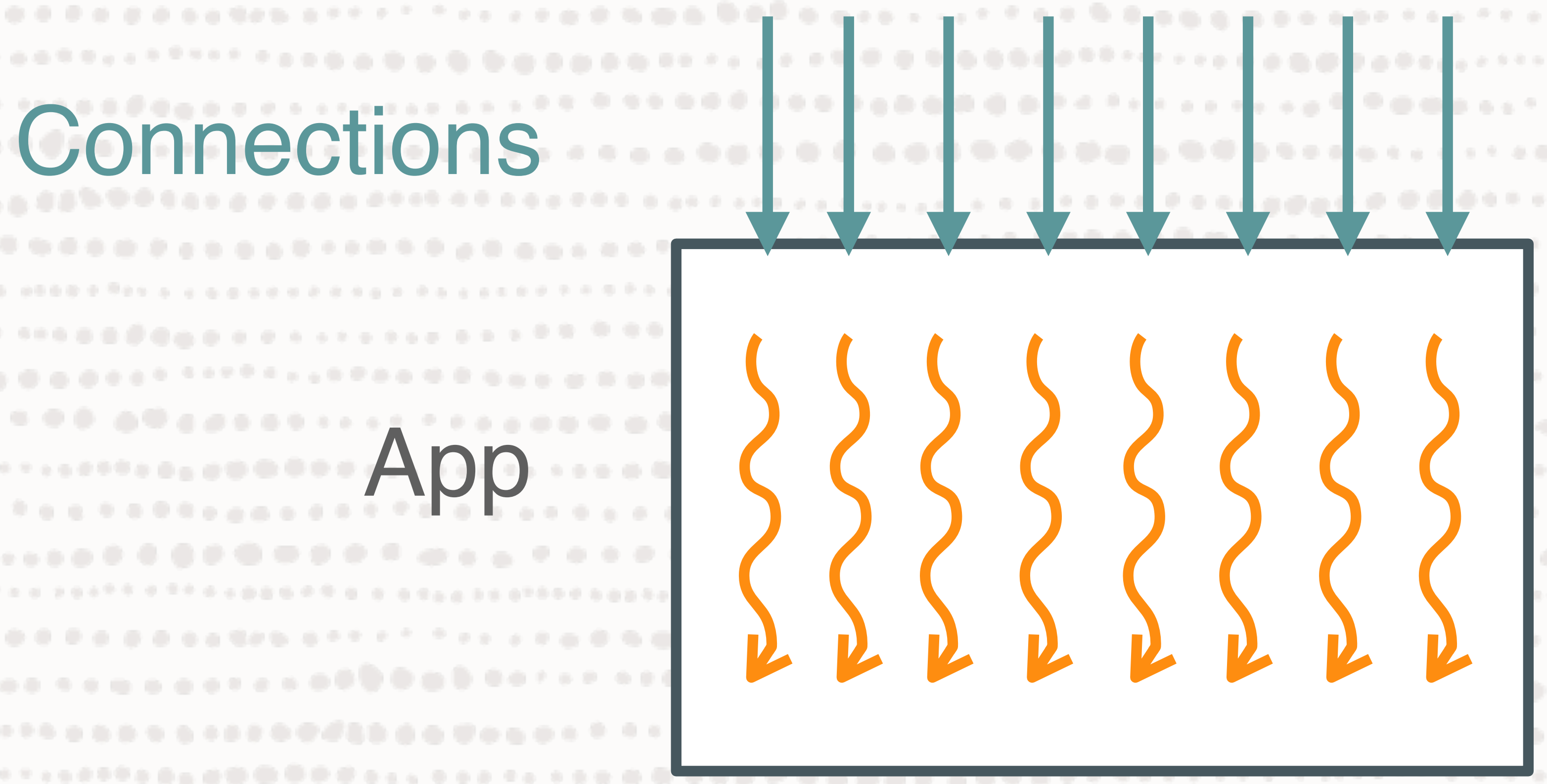
non-interoperable,

hard to debug/profile

ASYNC

Programmer ☹️

OS / Hardware 😀



Codes Like Sync, Scales Like Async

Programmer 😀

OS / Hardware 😀

API

API

- Use java.lang.Thread ?
- Introduce new API (maybe Fiber) ?

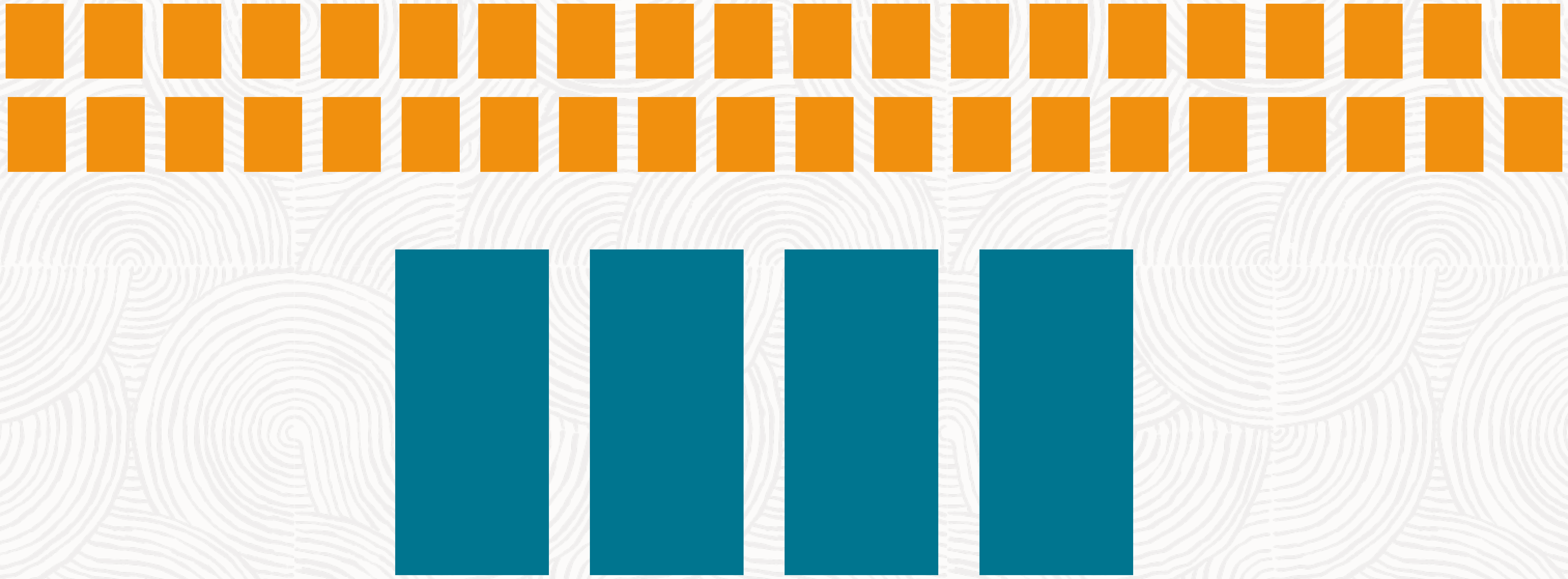
API

- Use java.lang.Thread ?
- Introduce new API (maybe Fiber) ?
- Use of `Thread.currentThread()` and `ThreadLocal` is pervasive
- Other aspects of Thread are rarely used

API

- Use java.lang.Thread ?
- Introduce new API (maybe Fiber) ?
- Use of `Thread.currentThread()` and `ThreadLocal` is pervasive
- Other aspects of Thread are rarely used
- Gravitational pull of 25 years of existing code is impossible to escape
- java.lang.Thread represent all threads
- The new low cost threads will be called “Virtual Threads”

Virtual threads



“carrier” OS threads managed by a scheduler

>2KB metadata 1MB stack	200-300B metadata Pay-as-you-go stack
1-10µs	~200ns

<<<Switch to IDE>>>

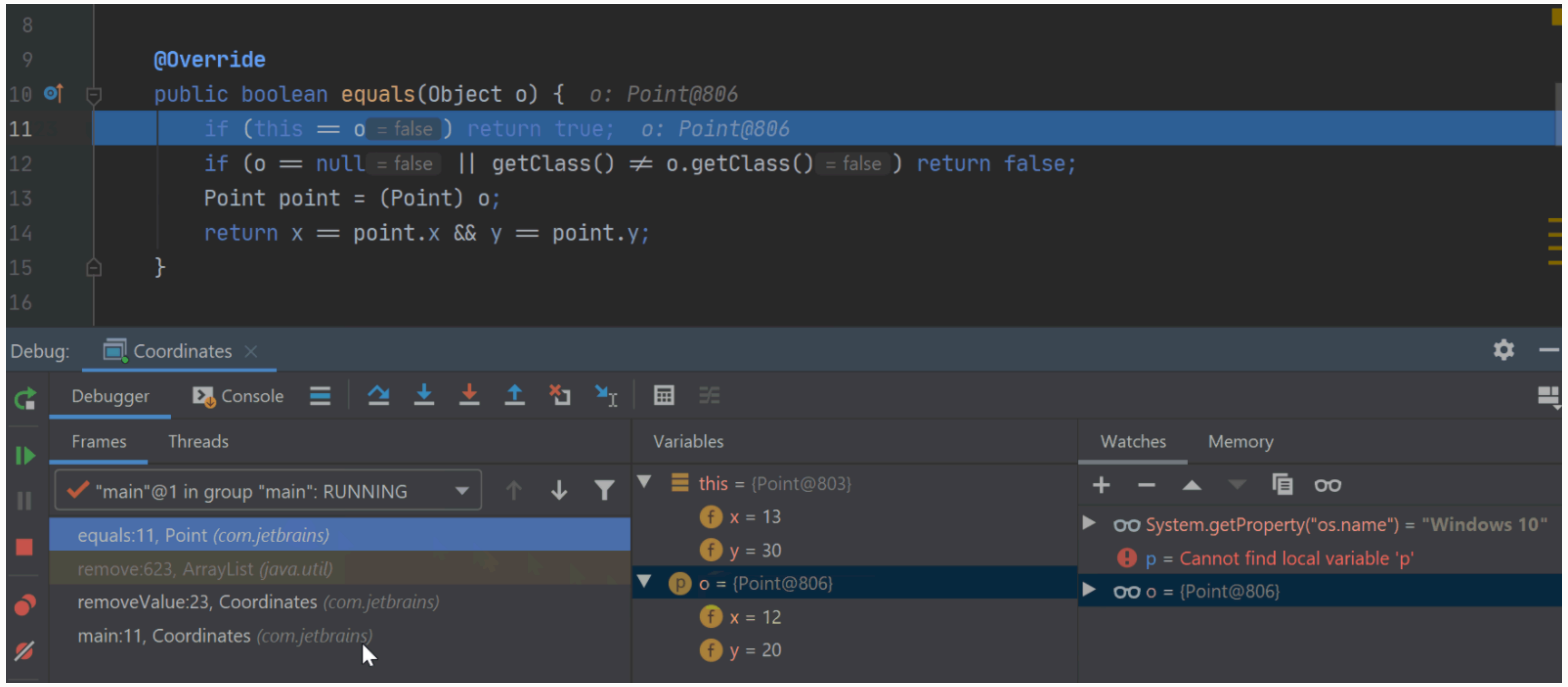
Limitations

- Blocking with native frames on stack
- Blocking while holding monitors
- In both cases, the OS thread is pinned

Preparing for Loom

- What can you do
  - Reduce use of thread locals
  - Reduce footprint of per thread/request data
  - Identify places where code is doing blocking I/O while holding a monitor, replace these with java.util.concurrent locks

A virtual thread is a Thread in the debugger



```
    @Override
    public boolean equals(Object o) {  o: Point@806
        if (this == o = false ) return true;  o: Point@806
        if (o == null = false  || getClass() ≠ o.getClass() = false ) return false;
        Point point = (Point) o;
        return x == point.x && y == point.y;
    }
```

```
"main"@1 in group "main": RUNNING
equals:11, Point (com.jetbrains)
remove:623, ArrayList (java.util)
removeValue:23, Coordinates (com.jetbrains)
main:11, Coordinates (com.jetbrains)
```

```
▼ this = {Point@803}
    x = 13
    y = 30
▼ o = {Point@806}
    x = 12
    y = 20
```

```
▶ System.getProperty("os.name") = "Windows 10"
  p = Cannot find local variable 'p'
▶ o = {Point@806}
```

A virtual thread is a Thread in the profiler

- Java Flight Recorder

```
$ jfr print --events jdk.Socket* --stack-depth 100 server.jfr
```

```
jdk.SocketRead {
  startTime = 08:27:08.077
  duration = 1.00 s
  host = "localhost"
  address = "127.0.0.1"
  port = 8081
  timeout = 0 s
  bytesRead = 161 bytes
  endOfStream = false
  eventThread = "<unnamed>" (javaThreadId = 84, virtual = true)
  stackTrace = [
    java.net.Socket$SocketInputStream.read(byte[], int, int) line: 67
    java.io.BufferedInputStream.fill() line: 255
    java.io.BufferedInputStream.read1(byte[], int, int) line: 310
    java.io.BufferedInputStream.lockedRead(byte[], int, int) line: 382
    java.io.BufferedInputStream.read(byte[], int, int) line: 361
    sun.net.www.http.HttpClient.parseHTTPHeader(MessageHeader, ProgressSource, HttpURLConnection) line: 791
    sun.net.www.http.HttpClient.parseHTTP(MessageHeader, ProgressSource, HttpURLConnection) line: 723
    sun.net.www.protocol.http.HttpURLConnection.getInputStream0() line: 1676
    sun.net.www.protocol.http.HttpURLConnection.getInputStream() line: 1577
    java.net.HttpURLConnection.getResponseCode() line: 527
    org.glassfish.jersey.client.HttpUrlConnector._apply(ClientRequest) line: 321
    org.glassfish.jersey.client.HttpUrlConnector.apply(ClientRequest) line: 227
    org.glassfish.jersey.client.ClientRuntime.invoke(ClientRequest) line: 225
    org.glassfish.jersey.client.JerseyInvocation$2.call() line: 671
    org.glassfish.jersey.internal.Errors.process(Callable, boolean) line: 315
    org.glassfish.jersey.internal.Errors.process(Producer, boolean) line: 297
    org.glassfish.jersey.internal.Errors.process(Producer) line: 228
    org.glassfish.jersey.process.internal.RequestScope.runInScope(Producer) line: 424
    org.glassfish.jersey.client.JerseyInvocation.invoke(Class) line: 667
    org.glassfish.jersey.client.JerseyInvocation$Builder.method(String, Class) line: 396
    org.glassfish.jersey.client.JerseyInvocation$Builder.get(Class) line: 296
    demo.AggregatorServices.query(String) line: 94
    demo.AggregatorServices.lambda$allOf$3(String) line: 74
    java.util.concurrent.ThreadExecutor$ThreadBoundCompletableFuture.run() line: 314
    java.lang.VirtualThread.lambda$new$0(Runnable) line: 128
    java.lang.Continuation.enter0() line: 396
    java.lang.Continuation.enter(Continuation, boolean) line: 389
    java.lang.Continuation.enterSpecial(Continuation, boolean)
  ]
}
```

A virtual thread is a Thread in the profiler

- Java Flight Recorder
- JVM TI based tools
- Challenges

Serviceability

- Troubleshooting and diagnosability
  - Identify pinned threads
  - Identify compute bound virtual threads
  - Thread dumps
  - …

Current status

- Current focus
  - Stability
  - Performance
  - API
  - Debugger support
- Important for a first preview
  - Aarch64 port
  - Thread dump

Further topics for exploration

- Channels
- Structured concurrency
- Scope variables
- Cancellation

Key Takeaways

- A virtual thread is a Thread in code, at run-time, in the debugger and in the profiler
- A virtual thread is not a wrapper around an OS thread, instead it is just a Java object
- Creating a virtual thread is cheap - you can have millions of them, don’t pool them!
- Blocking a virtual thread is cheap - be synchronous!

More information

- Early access builds: https://jdk.java.net/loom
- Mailing list: loom-dev@openjdk.java.net
- Wiki: https://wiki.openjdk.java.net/display/loom/Main

Safe harbor statement