Creating a Million Virtual Threads Using **Project Loom** to Improve Throughput

A N M Bazlur Rahman (@bazlur_rahman)

Who I am

Java™ Champions
Contrast SECURITY
Java User Group Bangladesh
foojay.io
InfoQ
DZone MVB Most·Valuable·Blogger
JAKARTA™ EE AMBASSADORS
দ্বিমিক প্রকাশনীর অন্যান্য বই
অ্যাডভান্সড জাভা প্রোগ্রামিং আ ন ম বজলুর রহমান
জাভা থ্রেড প্রোগ্রামিং আ ন ম বজলুর রহমান
জাভা ওয়েব প্রোগ্রামিং আ ন ম বজলুর রহমান
দ্বিতীয় সংস্করণ
জাভা প্রোগ্রামিং
আ ন ম বজলুর রহমান

Agenda

- Context
  - Java concurrency
- What is a virtual thread!
- Does it make sense?
- Why do we care about it!
- Limitations

This will be an extremely high level.

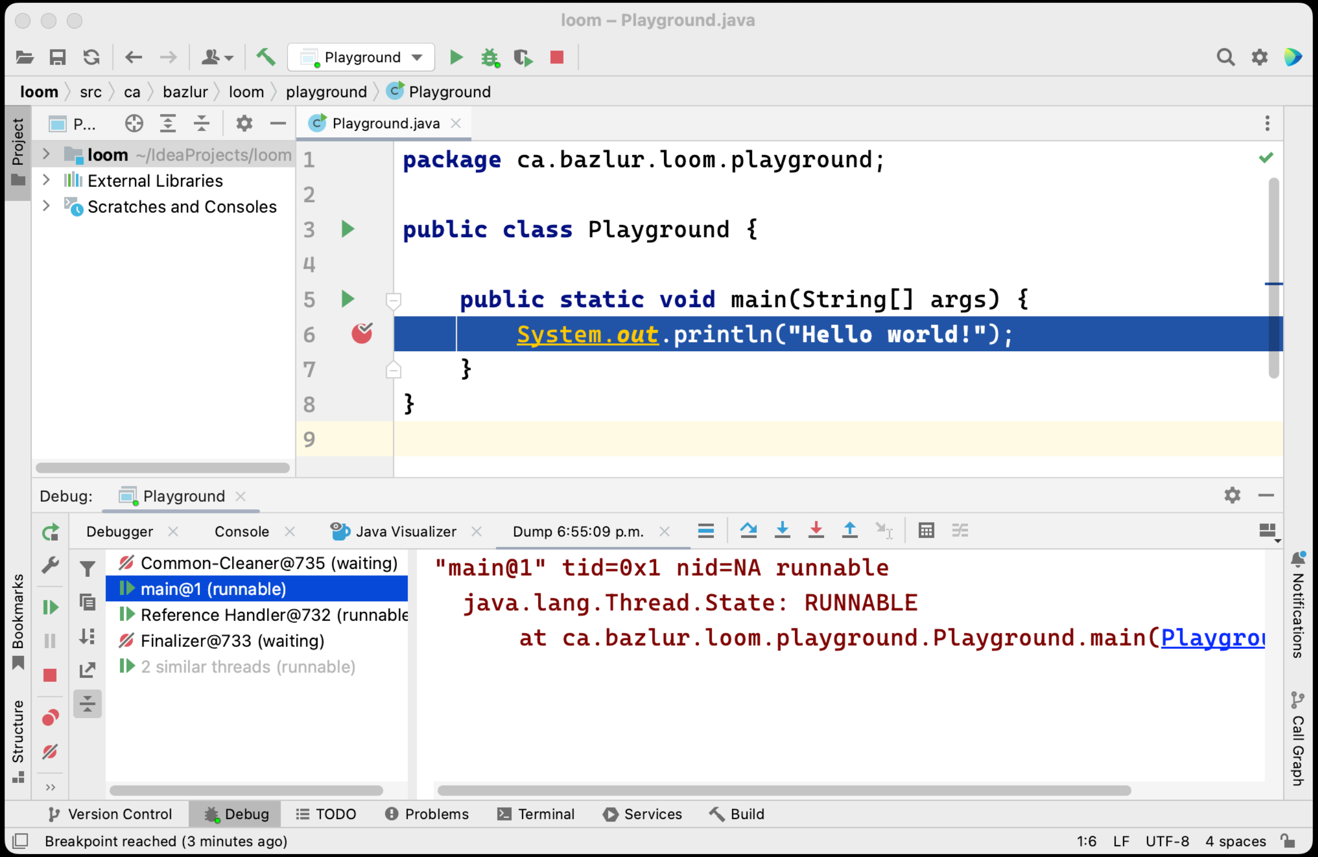
Java is made of Threads

From day 0, Java introduced Threads

Threads are units of execution in Java.

More threads == more Execution Environment

That means, we can do more things simultaneously



```java
package ca.bazlur.loom.playground;

public class Playground {

    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

```
Common-Cleaner@735 (waiting)
main@1 (runnable)
Reference Handler@732 (runnable
Finalizer@733 (waiting)
2 similar threads (runnable)
```

```
"main@1" tid=0x1 nid=NA runnable
  java.lang.Thread.State: RUNNABLE
      at ca.bazlur.loom.playground.Playground.main(Playgrou
```

Threads are in all layer of the Java platform

- Exceptions
- Thread Locals
- Debugger
- Profiler (JFR)

How do we create threads?

```
var thread = new Thread(() -> {
    System.out.println("Hello world!");
});

thread.start();
thread.join();
```

Java Threads == OS threads

```
"MyThread" #22 [27915] prio=5 os_prio=31 cpu=0.45ms elapsed=25.76s
tid=0x00007f7a83808200 nid=27915 waiting on
condition  [0x000000030af56000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
at java.lang.Thread.sleep0(java.base@19-loom/Native Method)
at java.lang.Thread.sleep(java.base@19-loom/Thread.java:462)
at Main.lambda$main$0(Main.java:13)
at Main$$Lambda$14/0x00000008010031f0.run(Unknown Source)
at java.lang.Thread.run(java.base@19-loom/Thread.java:1584)
```

Threads are Expensive

- 2 MiB of memory (outside of the heap)
- A wrapper of OS threads
- Context switching: ~100 µs
- Thread.start() consider inefficient

- So 1 million Threads require 2 Terabytes of memory!

Threads are Limited

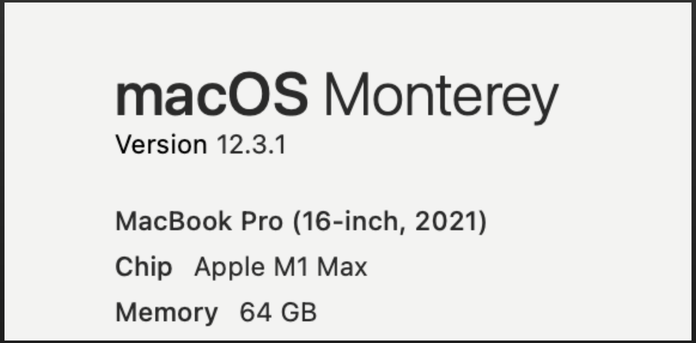
- In modern Java web applications, throughput is achieved by Concurrent Connections
- Modern OS can handle millions of concurrent connections
- More Concurrent Connection == More Throughput
- But the OS threads are limited in number

Let’s counts Threads

```
import java.util.Locale;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.concurrent.locks.LockSupport;

public class CountThreads {
   public static void main(String[] args) {
      var counter = new AtomicInteger();


      while (true) {
         var thread = new Thread(() -> {
            int threadsCount = counter.incrementAndGet();
            System.out.printf(
                  Locale.US, "started %,d\tthreads %,d%n",
                  threadsCount, Thread.currentThread().threadId()
            );
            LockSupport.park();
         });

         thread.start();
      }
   }
}
```

Native Thread COUNT on My Laptop

```
started 16,351  threads 16,381
started 16,352  threads 16,382
[53.027s][warning][os,thread] Failed to start thread
"Unknown thread" - pthread_create failed (EAGAIN) for
attributes: stacksize: 1024k, guardsize: 4k, detached.
[53.027s][warning][os,thread] Failed to start the native
thread for java.lang.Thread "Thread-16352"
Exception in thread "main" java.lang.OutOfMemoryError:
unable to create native thread: possibly out of memory or
process/resource limits reached
        at java.base/java.lang.Thread.start0(Native
Method)
        at
java.base/java.lang.Thread.start(Thread.java:1531)
        at
ca.bazlur.loom.lab1.CountThreads.main(CountThreads.java:20
)
```


macOS Monterey
Version 12.3.1

MacBook Pro (16-inch, 2021)
Chip Apple M1 Max
Memory 64 GB

DEMO (0)

```
public static void main(String[] args) throws IOException {
    var serverSocket = new ServerSocket(8080);
    while (true) {
        var socket = serverSocket.accept();
        handle(socket);
    }
}

private static void handle(Socket socket) throws IOException {
      //rest of the codes
      // this is a blocking operations
}
```

Demo (0) continues (2)

```
public static void main(String[] args) throws IOException {
    var serverSocket = new ServerSocket(8080);

    try (var executors = Executors.newCachedThreadPool()) {
        while (true) {
            var socket = serverSocket.accept();
            executors.submit(() -> {
                handle(socket);
            });

        }
    }
}

private static void handle(Socket socket) {
      //blocking ops
}
```

DEMO (0) Continues (3)

```
public class DDosAttack {
    public static void main(String[] args)
            throws IOException, InterruptedException {

        var sockets = new Socket[20_000];
        for (int i = 0; i < sockets.length; i++) {
            sockets[i] = new Socket("localhost", 8080);
            System.out.println("Connected: " + sockets[i]);
        }
    }
}
```

DEMO (0) Continues (4)

```
public static void main(String[] args) throws IOException {
    var serverSocket = new ServerSocket(8080);

    try (var executors = Executors. newVirtualThreadPerTaskExecutor()) {
        while (true) {
            var socket = serverSocket.accept();
            executors.submit(() -> {
                handle(socket);
            });

        }
    }
}

private static void handle(Socket socket) {

}
```

Project Loom

Allows us to create millions of cheap, lightweight virtual threads

Threads are divided into two groups

- Platform/Career thread – the real native threads
- Virtual threads, aka user threads

Creating Virtual Threads Easy

```
public class Playground {
    public static void main(String[] args) throws InterruptedException {

        Thread thread = Thread.startVirtualThread(() -> {
            System.out.println("Current Thread: " + Thread.currentThread());
        });
        thread.join();
    }
}



// Current Thread: VirtualThread[#22]/runnable@ForkJoinPool-1-worker-1
```

Creating Virtual Threads Easy (2)

```
Thread thread = Thread.ofVirtual().unstarted(() -> {
    System.out.println("Current Thread: " +
Thread.currentThread());
});

thread.start();


// Current Thread:
VirtualThread[#24]/runnable@ForkJoinPool-1-worker-2
```

Creating Platform Threads

```
Thread platformThread =Thread.ofPlatform().unstarted(() -> {
    System.out.println("Current Thread: " + Thread.currentThread());
});

platformThread.start();



// Current Thread: Thread[#27,Thread-0,5,main]
```

Virtual Thread can JUMP from one thread to another

(DEMO)

TARGET JDK 19 (preview)

```
javac --release 19 --enable-preview Main.java
```

```
java --enable-preview Main
```

```
java --source 19 --enable-preview Main.java
```

```
jshell --enable-preview
```

How it works

Virtual Threads = (Continuation + Scheduler)

$\Rightarrow$

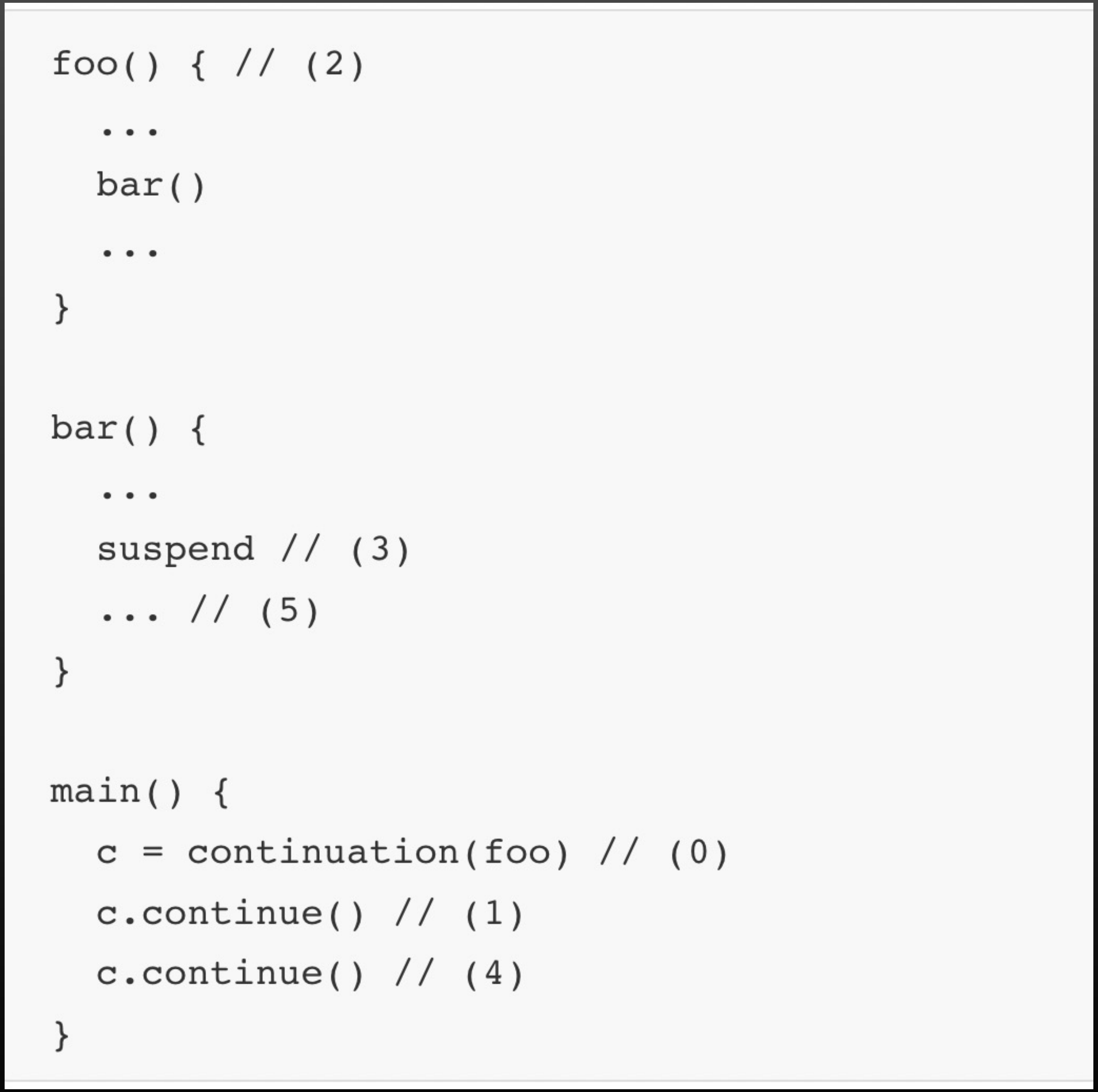
Schedulers = A bunch of career threads in Fork/Join Pool

Continuations

Basic definition: a sequential code that may suspend (itself) and can be resumed later

May suspend or yield execution at some point

When it suspends, control is passed outside of the continuation, and when it is resumed, control returns to the last yield point,



```
foo() { // (2)
  ...
  bar()
  ...
}

bar() {
  ...
  suspend // (3)
  ... // (5)
}

main() {
  c = continuation(foo) // (0)
  c.continue() // (1)
  c.continue() // (4)
}
```

- A continuation is created (0), whose entry point is foo;
- it is then invoked (1), which passes control to the entry point of the continuation (2),
- which then executes until the next suspension point (3) inside the bar subroutine,
- at which point the invocation (1) returns.
- When the continuation is invoked again (4),
- control returns to the line following the yield point (5).

Thread.SLEEP() in JDK 19

```
if (currentThread() instanceof VirtualThread vthread) {
    long nanos = MILLISECONDS.toNanos(millis);
    vthread.sleepNanos(nanos);
    return;
}
```

```
boolean yielded;
Future<?> unparker = scheduleUnpark(nanos);
setState(PARKING);

try {
    yielded = yieldContinuation();
} finally {
    assert (Thread.currentThread() == this)
            && (state() == RUNNING || state() == PARKING);
    cancel(unparker);
}
```

Other languages

WHO Else Uses the SAME concept?

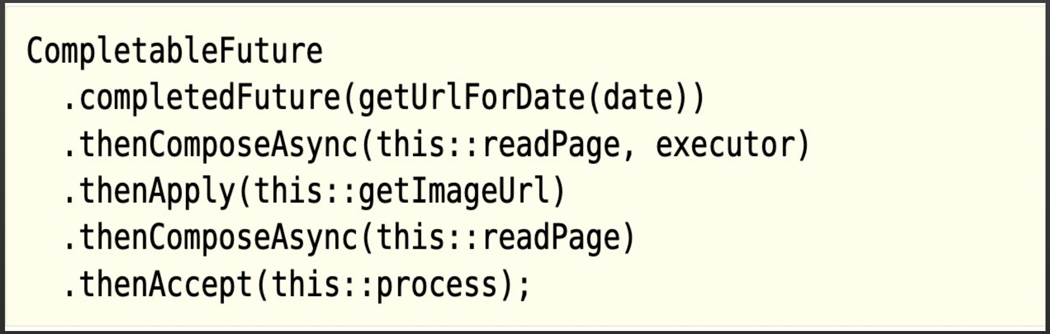
- Coroutine in Kotlin
- Goroutine go
- Async/await in JavaScript
- Python Generators
- Ruby Fibers
- Finally in Java as well

So Virtual Threads don’t consume CPU while they are sleeping.

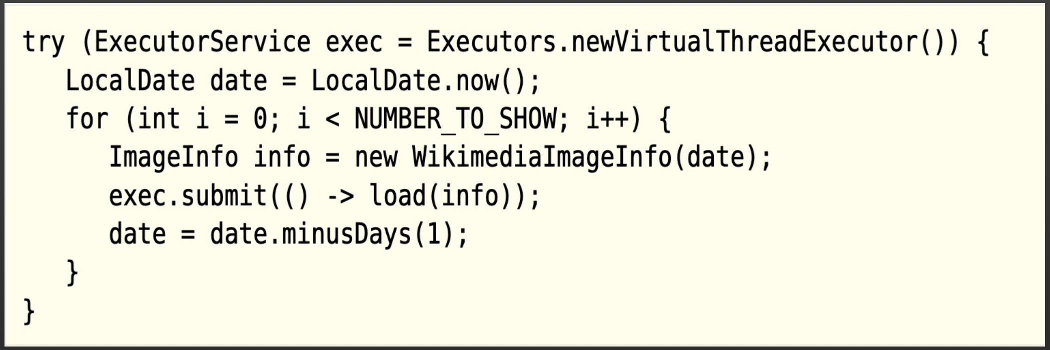
That means, technically (in reality 🤔 probably not) we can have million sleeping virtual threads.

Benefits

- Threads are cheap
- Request-par-thread style of Server-side programming becomes easy again as before.
- No longer required to do reactive-mind-bending programming
- You can create millions of them == more throughput
- On blocking I/O, Virtual threads get automatically suspended
- Supports thread-local variables, synchronization block, thread interruptions etc.
- Java code written in classic threads can easily run virtual threads without changes
- We can use old I/O (InputStream, OutputStream, Reader) as opposed to Channel & Buffer, which are great but difficult to write
- We no longer need to maintain a ThreadPool (no fine tuning)



```
CompletableFuture
  .completedFuture(getUrlForDate(date))
  .thenComposeAsync(this::readPage, executor)
  .thenApply(this::getImageUrl)
  .thenComposeAsync(this::readPage)
  .thenAccept(this::process);
```



```
try (ExecutorService exec = Executors.newVirtualThreadExecutor()) {
   LocalDate date = LocalDate.now();
   for (int i = 0; i < NUMBER_TO_SHOW; i++) {
      ImageInfo info = new WikimediaImageInfo(date);
      exec.submit(() -> load(info));
      date = date.minusDays(1);
   }
}
```

DEMO (2)

- Reference: https://horstmann.com/presentations/2021/jchampions/#(17)

- So downloading 10K images is just easy using Virtual Thread
- I/O call is yielding, which means the virtual thread will voluntarily suspend themselves, so career threads are not blocked. E.g. BlockingQueue.take()
- When data arrives, the JVM will wake up the Virtual Threads.
- Alternatively, with the native thread, we might have to use ThreadPool.
- 100 Threads in ThreadPool is the maximum 100 connections at the moment,

Analogy: Think about a bank with fixed number of tellers

- In Java terms, a teller is like a Platform Thread.
- Generally, it would take time to process each customer, say an average of 5 minutes.
- Sometimes, a customer would block the process, such as the teller needing to make a phone call to get some information.
- No work is performed while the teller is blocked waiting, and consequently, the entire line is blocked.

Before Loom

In Java terms, a teller is still like a Platform Thread but can park a customer.

Generally, it still takes time to process each customer, say an average of 5 minutes.

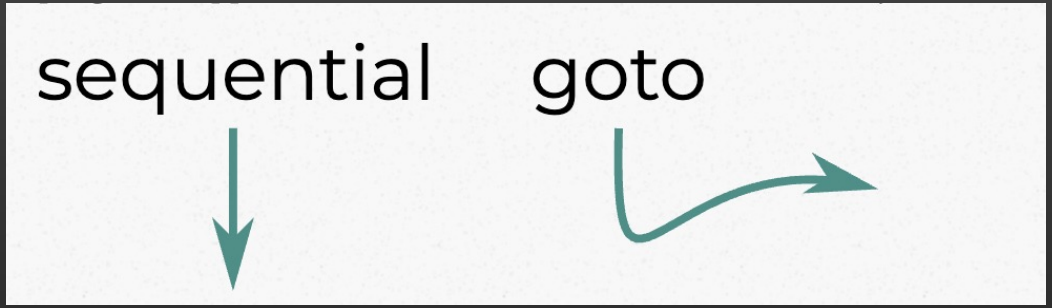
Sometimes, a customer would block the process, such as the teller needing some information before proceeding...

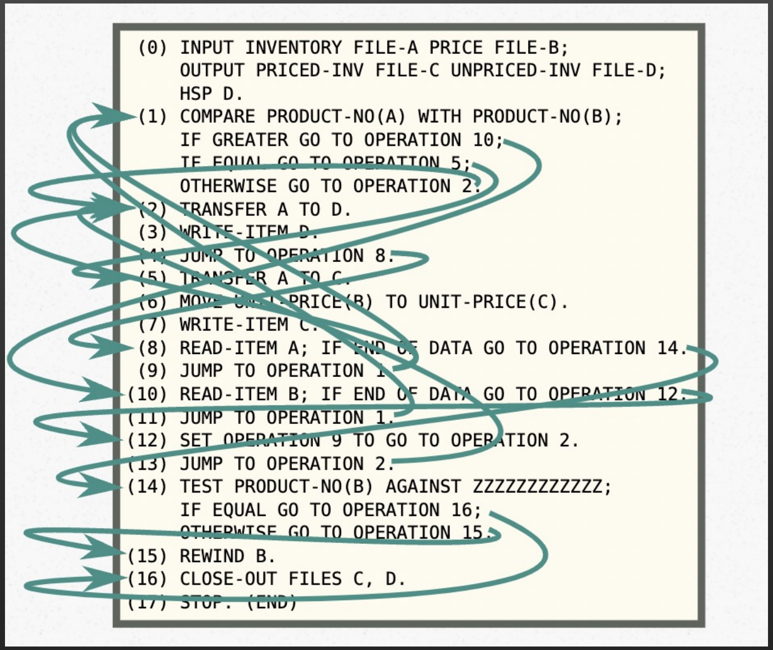
- The teller sends a text message or email to get the necessary information
- The teller asks the customer to be seated, and as soon as the information is available, they will be the next customer processed by the first available teller
- The teller starts processing the next customer in line
- This is analogous to a parked Virtual Thread, where the teller is like a Platform Thread, and the customer is like a Virtual Thread

Concurrency is increased by better policies and procedures in dealing with blocking operations

After Loom

Opens door for structured concurrency

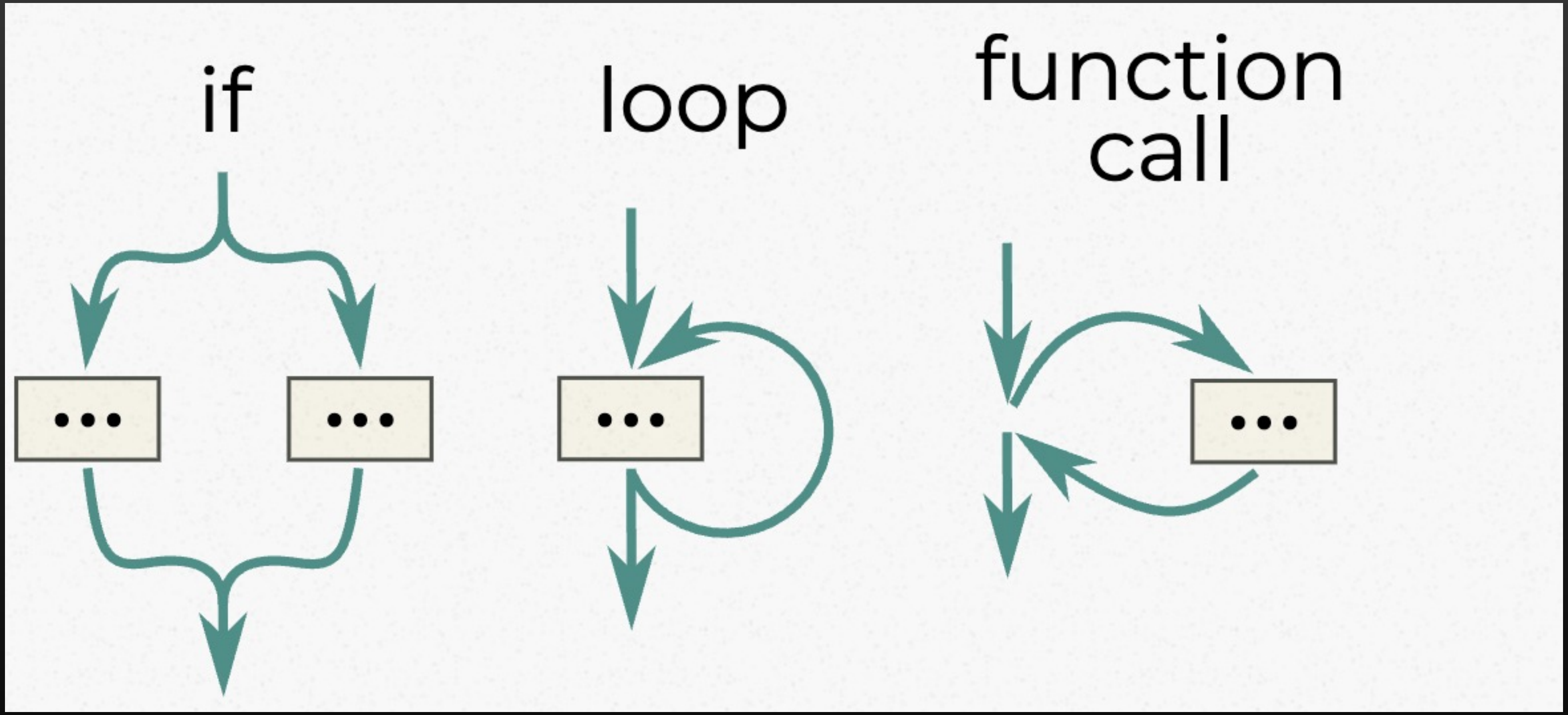




```
 (0) INPUT INVENTORY FILE-A PRICE FILE-B;
     OUTPUT PRICED-INV FILE-C UNPRICED-INV FILE-D;
     HSP D.
 (1) COMPARE PRODUCT-NO(A) WITH PRODUCT-NO(B);
     IF GREATER GO TO OPERATION 10;
     IF EQUAL GO TO OPERATION 5;
     OTHERWISE GO TO OPERATION 2.
 (2) TRANSFER A TO D.
 (3) WRITE-ITEM D.
 (4) JUMP TO OPERATION 8.
 (5) TRANSFER A TO C.
 (6) MOVE UNIT-PRICE(B) TO UNIT-PRICE(C).
 (7) WRITE-ITEM C.
 (8) READ-ITEM A; IF END OF DATA GO TO OPERATION 14.
 (9) JUMP TO OPERATION 1.
(10) READ-ITEM B; IF END OF DATA GO TO OPERATION 12.
(11) JUMP TO OPERATION 1.
(12) SET OPERATION 9 TO GO TO OPERATION 2.
(13) JUMP TO OPERATION 2.
(14) TEST PRODUCT-NO(B) AGAINST ZZZZZZZZZZZZ;
     IF EQUAL GO TO OPERATION 16;
     OTHERWISE GO TO OPERATION 15.
(15) REWIND B.
(16) CLOSE-OUT FILES C, D.
(17) STOP. (END)
```

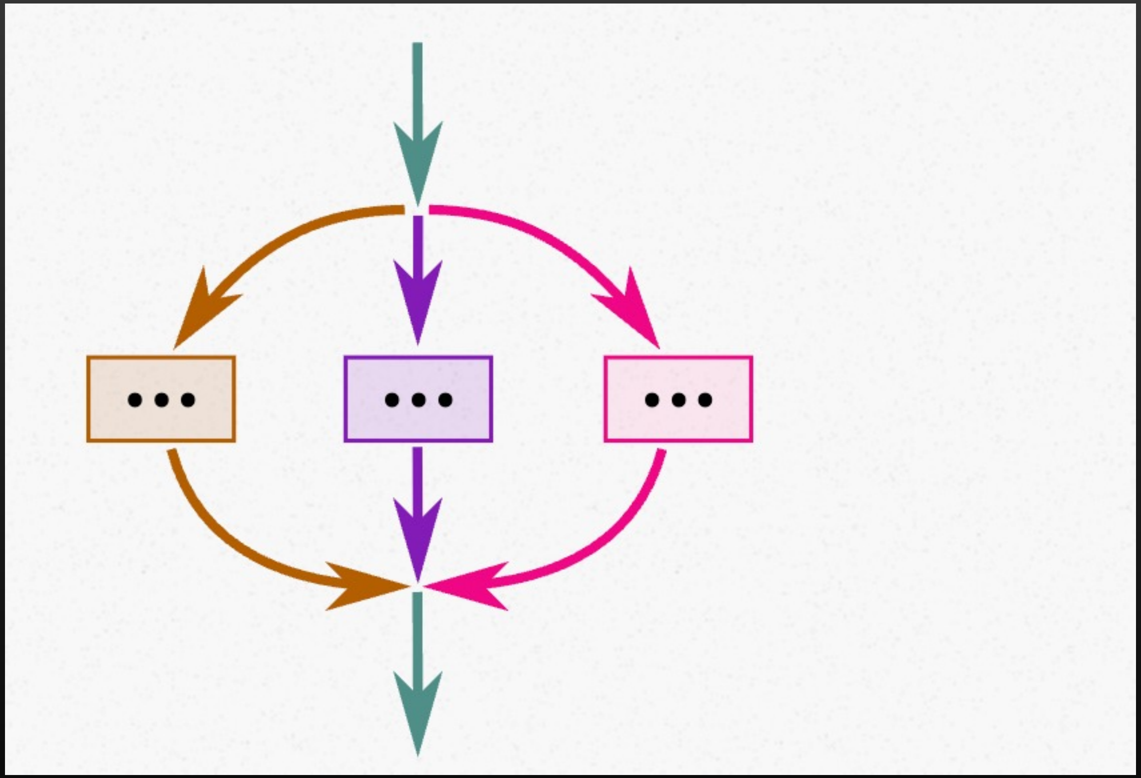
The EVIL of GOTO

1960: Structure programming replace GOTO with branches, Loops, functions



Reference: https://vorpus.org/blog/notes-on-structured-concurrency-or-go-statement-considered-harmful/

Structure concurrency is the SAME



Synchronous call

```
Response handle() throws IOException {
    String theUser = findUser();
    int theOrder = fetchOrder();
    return new Response(theUser, theOrder);
}
```

Asynchronous call

```
Response handle() throws ExecutionException, InterruptedException {
    Future<String>  user  = es.submit(() -> findUser());
    Future<Integer> order = es.submit(() -> fetchOrder());

    String theUser  = user.get();   // Join findUser
    int    theOrder = order.get();  // Join fetchOrder

    return new Response(theUser, theOrder);
}
```

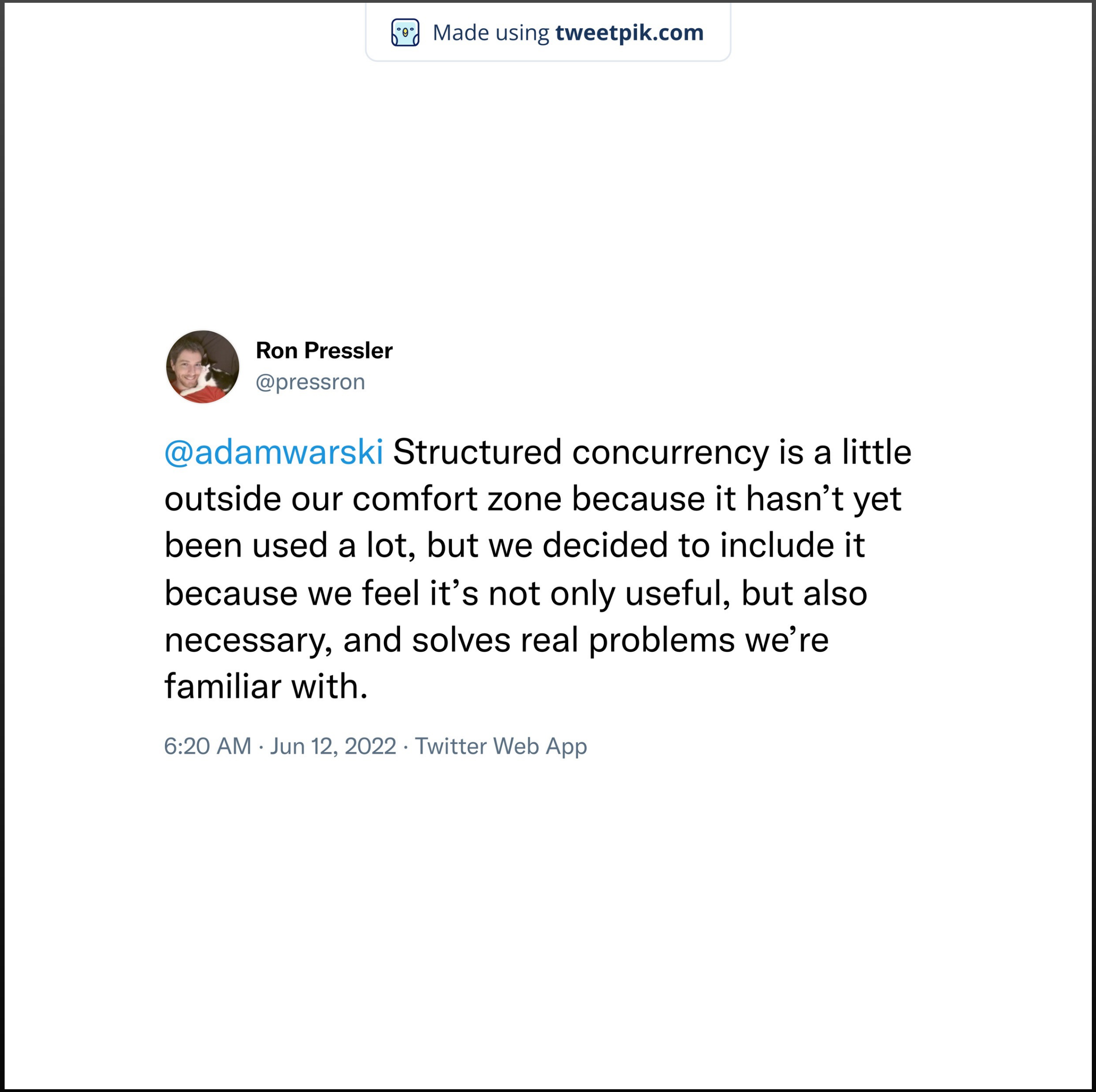
Both subtasks execute concurrently, and each can succeed or fail independently.

JEP 428: Structured Concurrency (Incubator)

```java
Response handle() throws ExecutionException, InterruptedException {
    try (var scope = new StructuredTaskScope.ShutdownOnFailure()) {
        Future<String>  user  = scope.fork(() -> findUser());
        Future<Integer> order = scope.fork(() -> fetchOrder());

        scope.join();           // Join both forks
        scope.throwIfFailed(); // ... and propagate errors

        // Here, both forks have succeeded, so compose their results
        return new Response(user.resultNow(), order.resultNow());
    }
}
```

Benefits

- Error handling with short-circuiting: If either findUser() or fetchOrder() fails, the other will be cancelled if it hasn't yet been completed
- Cancellation propagation: If the thread running handle is interrupted before or during the call to join, both forks will automatically cancel when the scope is exited.
- Clarity: clear code structure
- Observability: In thread, dump will demonstrate the task hierarchy

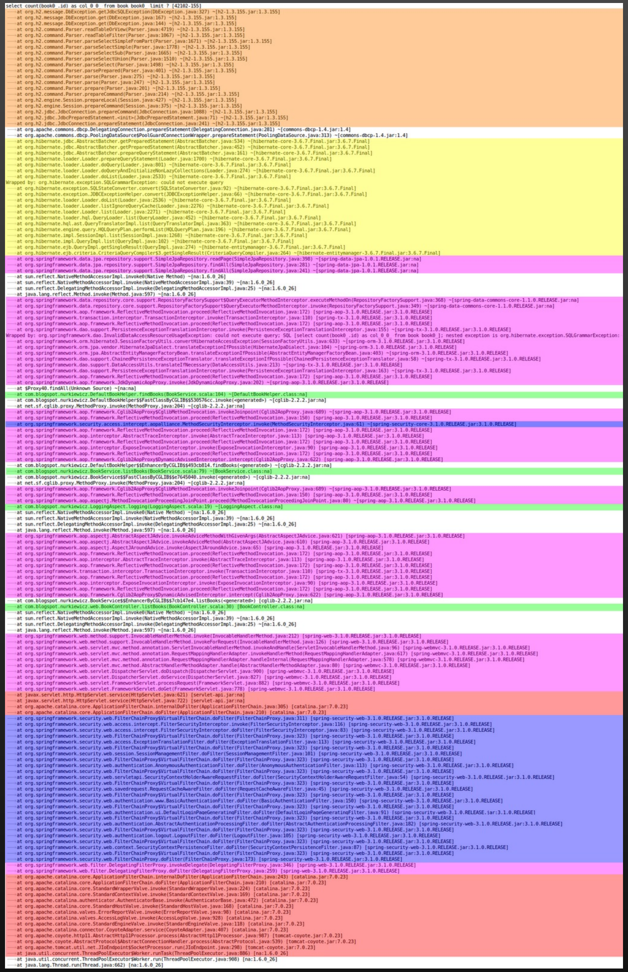


Made using **tweetpik.com**

Ron Pressler
@pressron

@adamwarski Structured concurrency is a little outside our comfort zone because it hasn't yet been used a lot, but we decided to include it because we feel it's not only useful, but also necessary, and solves real problems we're familiar with.

6:20 AM · Jun 12, 2022 · Twitter Web App



```
select count(book0_.id) as col_0_0_ from book book0_ limit ? [42102-155]
	at org.h2.message.DbException.getJdbcSQLException(DbException.java:327) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.message.DbException.get(DbException.java:167) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.message.DbException.get(DbException.java:144) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.readTableOrView(Parser.java:4719) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.readTableFilter(Parser.java:1067) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.parseSelectSimpleFromPart(Parser.java:1671) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.parseSelectSimple(Parser.java:1778) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.parseSelectSub(Parser.java:1665) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.parseSelectUnion(Parser.java:1510) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.parseSelect(Parser.java:1498) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.parsePrepared(Parser.java:401) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.parse(Parser.java:275) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.parse(Parser.java:247) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.prepare(Parser.java:201) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.command.Parser.prepareCommand(Parser.java:214) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.engine.Session.prepareLocal(Session.java:427) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.engine.Session.prepareCommand(Session.java:375) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.jdbc.JdbcConnection.prepareCommand(JdbcConnection.java:1088) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.jdbc.JdbcPreparedStatement.<init>(JdbcPreparedStatement.java:71) ~[h2-1.3.155.jar:1.3.155]
	at org.h2.jdbc.JdbcConnection.prepareStatement(JdbcConnection.java:241) ~[h2-1.3.155.jar:1.3.155]
	at org.apache.commons.dbcp.DelegatingConnection.prepareStatement(DelegatingConnection.java:281) ~[commons-dbcp-1.4.jar:1.4]
	at org.apache.commons.dbcp.PoolingDataSource$PoolGuardConnectionWrapper.prepareStatement(PoolingDataSource.java:313) ~[commons-dbcp-1.4.jar:1.4]
	at org.hibernate.jdbc.AbstractBatcher.getPreparedStatement(AbstractBatcher.java:534) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.jdbc.AbstractBatcher.getPreparedStatement(AbstractBatcher.java:452) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.jdbc.AbstractBatcher.prepareQueryStatement(AbstractBatcher.java:161) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.loader.Loader.prepareQueryStatement(Loader.java:1700) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.loader.Loader.doQuery(Loader.java:801) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.loader.Loader.doQueryAndInitializeNonLazyCollections(Loader.java:274) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.loader.Loader.doList(Loader.java:2533) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
Wrapped by: org.hibernate.exception.SQLGrammarException: could not execute query
	at org.hibernate.exception.SQLStateConverter.convert(SQLStateConverter.java:92) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.exception.JDBCExceptionHelper.convert(JDBCExceptionHelper.java:66) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.loader.Loader.doList(Loader.java:2536) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.loader.Loader.listIgnoreQueryCache(Loader.java:2276) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.loader.Loader.list(Loader.java:2271) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.loader.hql.QueryLoader.list(QueryLoader.java:452) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.hql.ast.QueryTranslatorImpl.list(QueryTranslatorImpl.java:363) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.engine.query.HQLQueryPlan.performList(HQLQueryPlan.java:196) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.impl.SessionImpl.list(SessionImpl.java:1268) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.impl.QueryImpl.list(QueryImpl.java:102) ~[hibernate-core-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.ejb.QueryImpl.getSingleResult(QueryImpl.java:274) ~[hibernate-entitymanager-3.6.7.Final.jar:3.6.7.Final]
	at org.hibernate.ejb.criteria.CriteriaQueryCompiler$3.getSingleResult(CriteriaQueryCompiler.java:264) ~[hibernate-entitymanager-3.6.7.Final.jar:3.6.7.Final]
	at org.springframework.data.jpa.repository.support.SimpleJpaRepository.readPage(SimpleJpaRepository.java:398) ~[spring-data-jpa-1.0.1.RELEASE.jar:na]
	at org.springframework.data.jpa.repository.support.SimpleJpaRepository.findAll(SimpleJpaRepository.java:281) ~[spring-data-jpa-1.0.1.RELEASE.jar:na]
	at org.springframework.data.jpa.repository.support.SimpleJpaRepository.findAll(SimpleJpaRepository.java:241) ~[spring-data-jpa-1.0.1.RELEASE.jar:na]
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:1.6.0_26]
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39) ~[na:1.6.0_26]
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25) ~[na:1.6.0_26]
	at java.lang.reflect.Method.invoke(Method.java:597) ~[na:1.6.0_26]
	at org.springframework.data.repository.core.support.RepositoryFactorySupport$QueryExecutorMethodInterceptor.executeMethodOn(RepositoryFactorySupport.java:368) ~[spring-data-commons-core-1.1.0.RELEASE.jar:na]
	at org.springframework.data.repository.core.support.RepositoryFactorySupport$QueryExecutorMethodInterceptor.invoke(RepositoryFactorySupport.java:349) ~[spring-data-commons-core-1.1.0.RELEASE.jar:na]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.java:110) [spring-tx-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.dao.support.PersistenceExceptionTranslationInterceptor.invoke(PersistenceExceptionTranslationInterceptor.java:155) ~[spring-tx-3.1.0.RELEASE.jar:3.1.0.RELEASE]
Wrapped by: org.springframework.dao.InvalidDataAccessResourceUsageException: could not execute query; SQL [select count(book0_.id) as col_0_0_ from book book0_]; nested exception is org.hibernate.exception.SQLGrammarException:
	at org.springframework.orm.hibernate3.SessionFactoryUtils.convertHibernateAccessException(SessionFactoryUtils.java:633) ~[spring-orm-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.orm.jpa.vendor.HibernateJpaDialect.translateExceptionIfPossible(HibernateJpaDialect.java:104) ~[spring-orm-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.translateExceptionIfPossible(AbstractEntityManagerFactoryBean.java:403) ~[spring-orm-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.dao.support.ChainedPersistenceExceptionTranslator.translateExceptionIfPossible(ChainedPersistenceExceptionTranslator.java:58) ~[spring-tx-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.dao.support.DataAccessUtils.translateIfNecessary(DataAccessUtils.java:213) ~[spring-tx-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.dao.support.PersistenceExceptionTranslationInterceptor.invoke(PersistenceExceptionTranslationInterceptor.java:163) ~[spring-tx-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:202) ~[spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at $Proxy40.findAll(Unknown Source) ~[na:na]
	at com.blogspot.nurkiewicz.DefaultBookHelper.findBooks(BookService.scala:104) ~[DefaultBookHelper.class:na]
	at com.blogspot.nurkiewicz.DefaultBookHelper$$FastClassByCGLIB$$530576cc.invoke(<generated>) ~[cglib-2.2.2.jar:na]
	at net.sf.cglib.proxy.MethodProxy.invoke(MethodProxy.java:204) ~[cglib-2.2.2.jar:na]
	at org.springframework.aop.framework.Cglib2AopProxy$CglibMethodInvocation.invokeJoinpoint(Cglib2AopProxy.java:689) ~[spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:150) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.access.intercept.aopalliance.MethodSecurityInterceptor.invoke(MethodSecurityInterceptor.java:61) ~[spring-security-core-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.interceptor.AbstractTraceInterceptor.invoke(AbstractTraceInterceptor.java:113) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:90) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.Cglib2AopProxy$DynamicAdvisedInterceptor.intercept(Cglib2AopProxy.java:622) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at com.blogspot.nurkiewicz.DefaultBookHelper$$EnhancerByCGLIB$$493cb814.findBooks(<generated>) ~[cglib-2.2.2.jar:na]
	at com.blogspot.nurkiewicz.BookService.listBooks(BookService.scala:79) ~[BookService.class:na]
	at com.blogspot.nurkiewicz.BookService$$FastClassByCGLIB$$e7645040.invoke(<generated>) ~[cglib-2.2.2.jar:na]
	at net.sf.cglib.proxy.MethodProxy.invoke(MethodProxy.java:204) ~[cglib-2.2.2.jar:na]
	at org.springframework.aop.framework.Cglib2AopProxy$CglibMethodInvocation.invokeJoinpoint(Cglib2AopProxy.java:689) ~[spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:150) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.aspectj.MethodInvocationProceedingJoinPoint.proceed(MethodInvocationProceedingJoinPoint.java:80) ~[spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at com.blogspot.nurkiewicz.LoggingAspect.logging(LoggingAspect.scala:19) ~[LoggingAspect.class:na]
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:1.6.0_26]
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39) ~[na:1.6.0_26]
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25) ~[na:1.6.0_26]
	at java.lang.reflect.Method.invoke(Method.java:597) ~[na:1.6.0_26]
	at org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethodWithGivenArgs(AbstractAspectJAdvice.java:621) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethod(AbstractAspectJAdvice.java:610) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.aspectj.AspectJAroundAdvice.invoke(AspectJAroundAdvice.java:65) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.interceptor.AbstractTraceInterceptor.invoke(AbstractTraceInterceptor.java:113) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.java:110) [spring-tx-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:90) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.aop.framework.Cglib2AopProxy$DynamicAdvisedInterceptor.intercept(Cglib2AopProxy.java:622) [spring-aop-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at com.blogspot.nurkiewicz.BookService$$EnhancerByCGLIB$$7cb147e4.listBooks(<generated>) [cglib-2.2.2.jar:na]
	at com.blogspot.nurkiewicz.web.BookController.listBooks(BookController.scala:30) [BookController.class:na]
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:1.6.0_26]
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39) ~[na:1.6.0_26]
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25) ~[na:1.6.0_26]
	at java.lang.reflect.Method.invoke(Method.java:597) ~[na:1.6.0_26]
	at org.springframework.web.method.support.InvocableHandlerMethod.invoke(InvocableHandlerMethod.java:212) [spring-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:126) [spring-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:96) [spring-webmvc-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:617) [spring-webmvc-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:578) [spring-webmvc-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:80) [spring-webmvc-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:900) [spring-webmvc-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:827) [spring-webmvc-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:882) [spring-webmvc-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:778) [spring-webmvc-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at javax.servlet.http.HttpServlet.service(HttpServlet.java:621) [servlet-api.jar:na]
	at javax.servlet.http.HttpServlet.service(HttpServlet.java:722) [servlet-api.jar:na]
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:305) [catalina.jar:7.0.23]
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210) [catalina.jar:7.0.23]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:311) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.access.intercept.FilterSecurityInterceptor.invoke(FilterSecurityInterceptor.java:116) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.access.intercept.FilterSecurityInterceptor.doFilter(FilterSecurityInterceptor.java:83) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.access.ExceptionTranslationFilter.doFilter(ExceptionTranslationFilter.java:113) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.session.SessionManagementFilter.doFilter(SessionManagementFilter.java:101) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.authentication.AnonymousAuthenticationFilter.doFilter(AnonymousAuthenticationFilter.java:113) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter.doFilter(SecurityContextHolderAwareRequestFilter.java:54) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.savedrequest.RequestCacheAwareFilter.doFilter(RequestCacheAwareFilter.java:45) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.authentication.www.BasicAuthenticationFilter.doFilter(BasicAuthenticationFilter.java:150) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.authentication.ui.DefaultLoginPageGeneratingFilter.doFilter(DefaultLoginPageGeneratingFilter.java:91) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.authentication.AbstractAuthenticationProcessingFilter.doFilter(AbstractAuthenticationProcessingFilter.java:182) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.authentication.logout.LogoutFilter.doFilter(LogoutFilter.java:105) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.context.SecurityContextPersistenceFilter.doFilter(SecurityContextPersistenceFilter.java:87) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:323) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.security.web.FilterChainProxy.doFilter(FilterChainProxy.java:173) [spring-security-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.filter.DelegatingFilterProxy.invokeDelegate(DelegatingFilterProxy.java:346) [spring-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.springframework.web.filter.DelegatingFilterProxy.doFilter(DelegatingFilterProxy.java:259) [spring-web-3.1.0.RELEASE.jar:3.1.0.RELEASE]
	at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:243) [catalina.jar:7.0.23]
	at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210) [catalina.jar:7.0.23]
	at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:224) [catalina.jar:7.0.23]
	at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:169) [catalina.jar:7.0.23]
	at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:472) [catalina.jar:7.0.23]
	at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:168) [catalina.jar:7.0.23]
	at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:98) [catalina.jar:7.0.23]
	at org.apache.catalina.valves.AccessLogValve.invoke(AccessLogValve.java:928) [catalina.jar:7.0.23]
	at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:118) [catalina.jar:7.0.23]
	at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:407) [catalina.jar:7.0.23]
	at org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java:987) [tomcat-coyote.jar:7.0.23]
	at org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol.java:539) [tomcat-coyote.jar:7.0.23]
	at org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEndpoint.java:298) [tomcat-coyote.jar:7.0.23]
	at java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.java:886) [na:1.6.0_26]
	at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:908) [na:1.6.0_26]
	at java.lang.Thread.run(Thread.java:662) [na:1.6.0_26]
```

Limitations (1)

- The size of a stack of platform threads is fixed. Usually 1MB
- The virtual threads stack can shrink and grow;
- That's why they are inexpensive, especially if we have simple tasks.
- So if the stack trace is long, when they suspend, they have to store it somewhere, which indicates that the cost of the virtual thread can approach platform threads.
- So millions of complex threads may not practical
- Virtual Threads gets pinned using classical synchronization or native code call

Ref: https://nurkiewicz.com/2012/03/filtering-irrelevant-stack-trace-lines.html

Limitation (2)

- CPU intensive works are not suitable for virtual threads
- They will keep working as the platform thread since there would be no blocking or sleeping.
- So if a handful of virtual threads do CPU-intensive work, that means other virtual threads may not get a chance.
- On the other hand, this issue never happened on the platform thread because of preemption. They are stopped at an arbitrary moment.
- In future, preemption may be done in the virtual thread as well.

Limitation (3)

- A virtual thread cannot be created using a public constructor
- They are daemon thread. Can not be turned into a non-daemon thread
- Priority cannot be change
- Virtual threads are part of Career threads, so they are not part of any ThreadGroup.
- Virtual threads should never be pooled since each is intended to run only a single task over its lifetime.

Limitations (4)

- Stack vs. heap memory
- Virtual threads reside on the heap, which means GC has to do a ton of extra work

Questions: Do we need reactive programming?

- So reactive programming was introduced so that we don’t block platform threads.
- Since we don’t have a problem with blocking anymore, maybe a reactive stack look for sunset!!
- Maybe not yet.
- It doesn’t address yet:
    - Backpressure
    - Change propagation
    - Composability etc.

Resources

- **Project Loom: https://wiki.openjdk.java.net/display/loom/Main**
- **JEP 425: Virtual Threads (Preview)**
- Release JDK 19
- Ron Pressler: https://www.infoq.com/podcasts/java-project-loom/
- Project Loom (fiber and continuation) : https://cr.openjdk.java.net/~rpressler/loom/Loom-Proposal.html
- Adam Warski: https://www.youtube.com/watch?v=_fFzyY_7UmA
- Heinz Kabutz: https://www.youtube.com/watch?v=MuxiaGyidNA
- Cay Hortsman: https://www.youtube.com/watch?v=NFYGZKpPwSM
-

Thank you

https://twitter.com/bazlur_rahman

https://www.getrevue.co/profile/bazlur_rahman