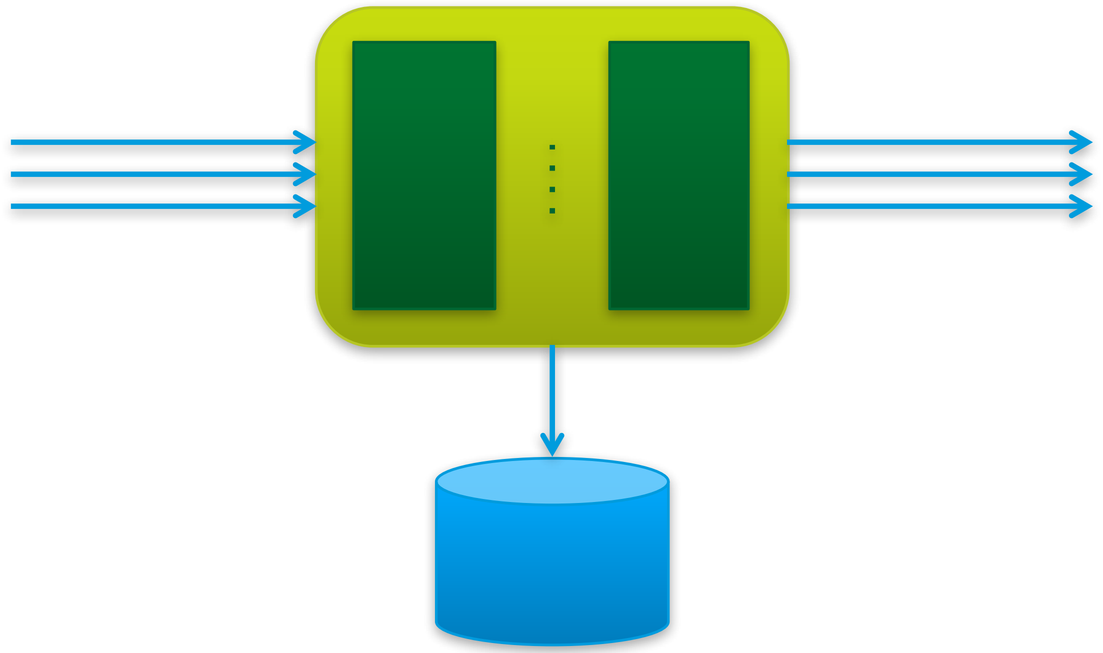
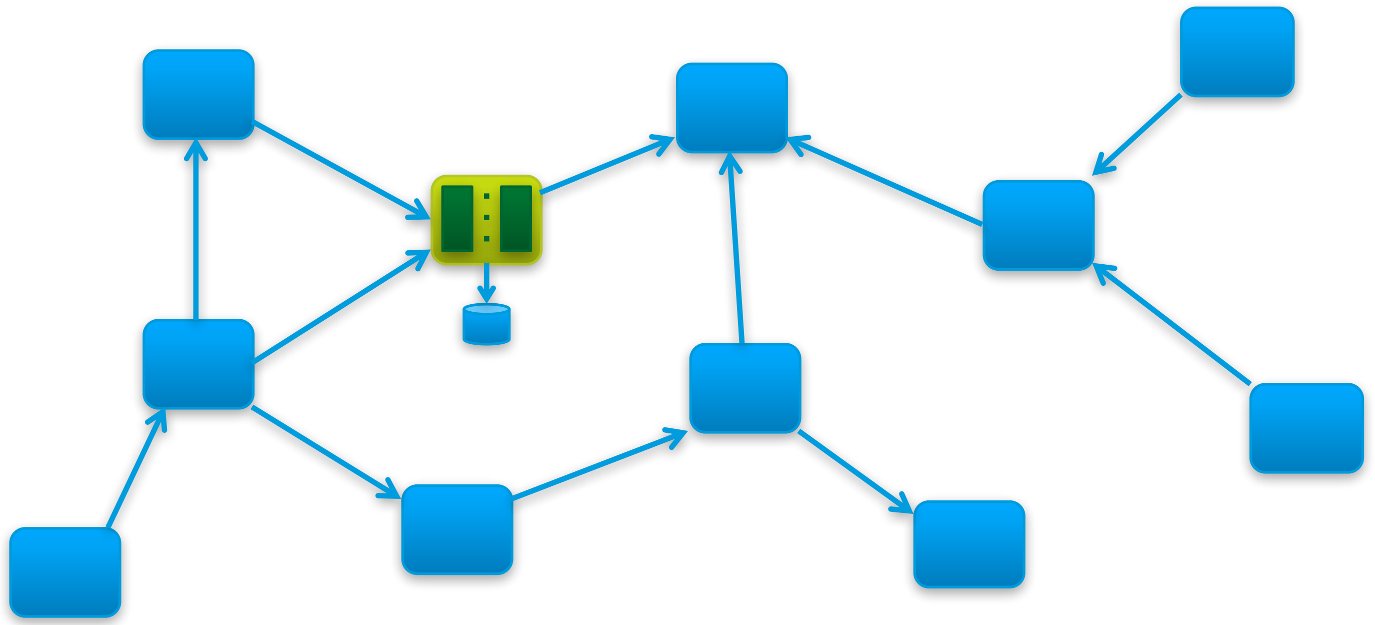
# Web Application Threads

A short story of threads trapped in a job they never wanted

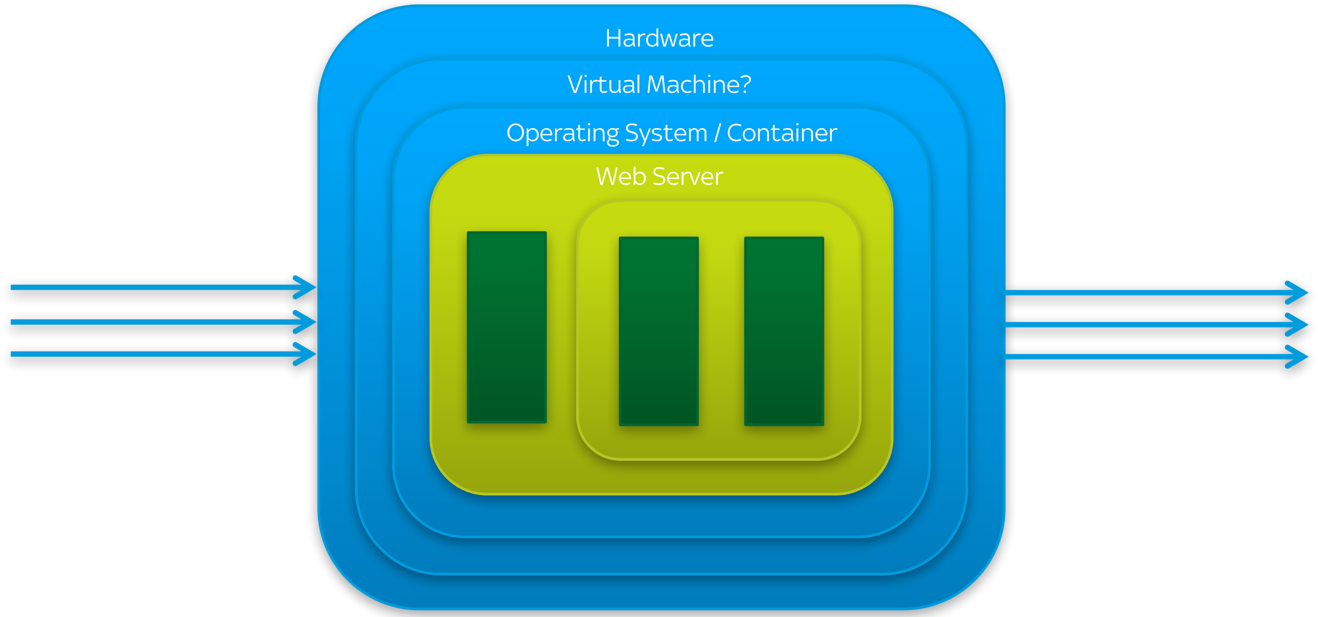
## Introduction - Your WebApp?



## Introduction - Your WebApp!



## Introduction - Your WebApp!



## Concepts

- **Synchronous / Asynchronous**

  A method call is considered synchronous if the caller cannot make progress until the method returns a value or throws an exception. On the other hand, an asynchronous call allows the caller to progress after a finite number of steps, and the completion of the method may be signalled via some additional mechanism (it might be a registered callback, a Future, or a message).

- **Blocking / Non-blocking**

  We talk about blocking if the delay of one thread can indefinitely delay some of the other threads. [...] In contrast, non-blocking means that no thread is able to indefinitely delay others.

Source: http://doc.akka.io/docs/akka/2.4.2/general/terminology.html

## Concepts

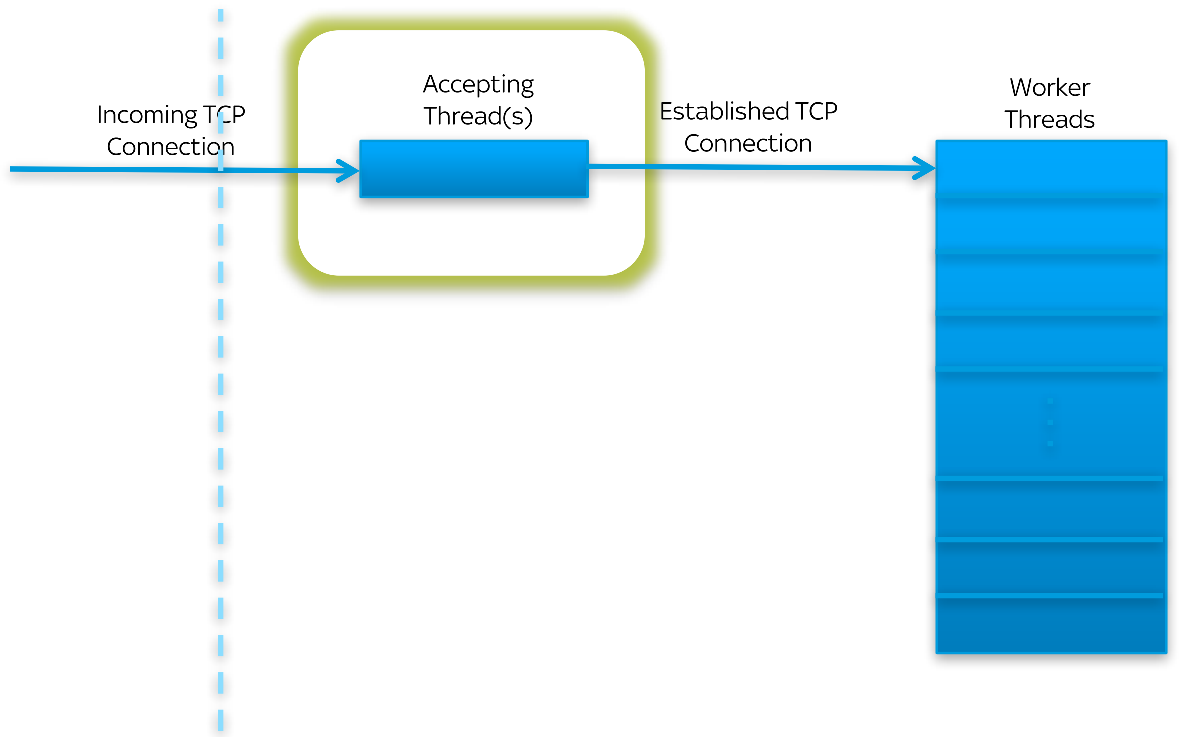
- **Interruptible task**

  An *interrupt* is an indication to a thread that it should stop what it is doing and do something else. It's up to the programmer to decide exactly how a thread responds to an interrupt, but it is very common for the thread to terminate. […]

  A thread sends an interrupt by invoking interrupt on the Thread object for the thread to be interrupted. For the interrupt mechanism to work correctly, the interrupted thread must support its own interruption.

Source: https://docs.oracle.com/javase/tutorial/essential/concurrency/interrupt.html

## Web Server

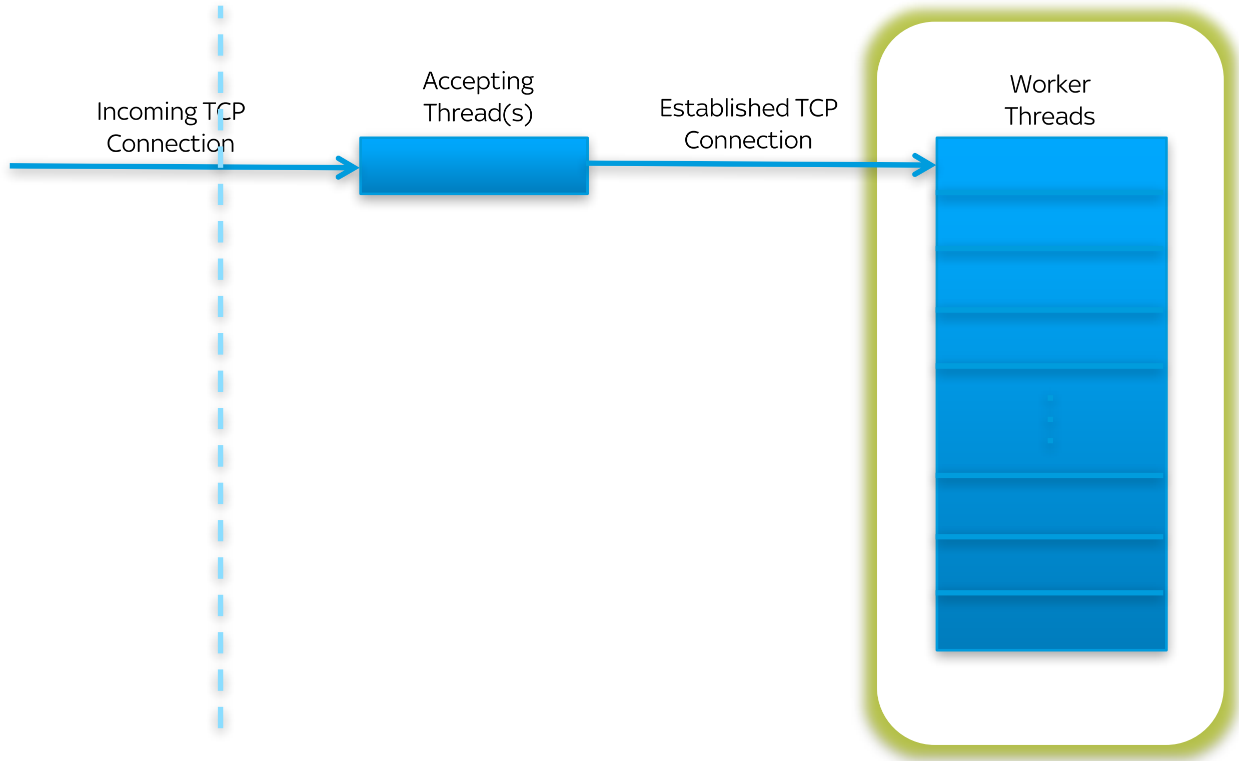


Accept incoming **TCP** connections

Pass them to worker threads

**Always blocking** on the *accept* system call

# Web Server - BIO

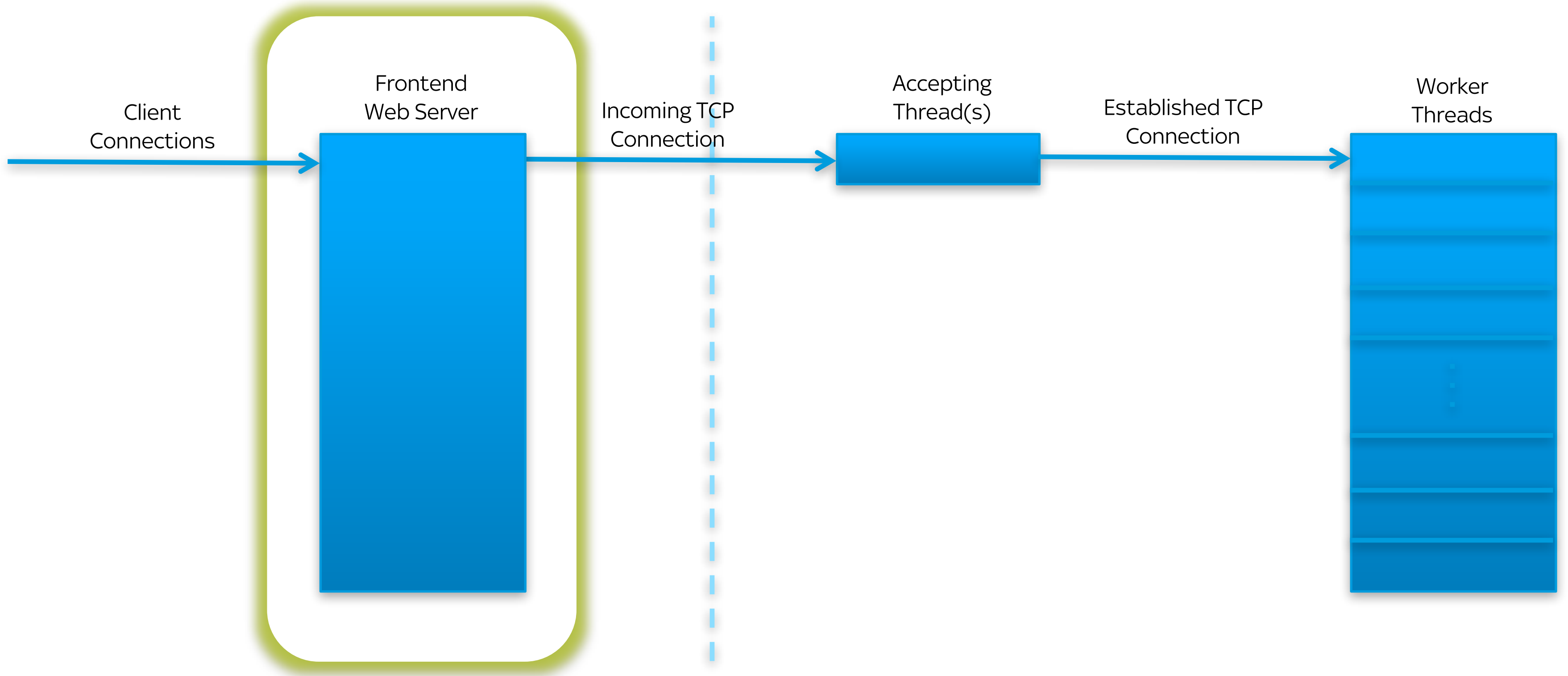


Process **all HTTP** requests for a TCP connection

**Blocking one thread per connection**

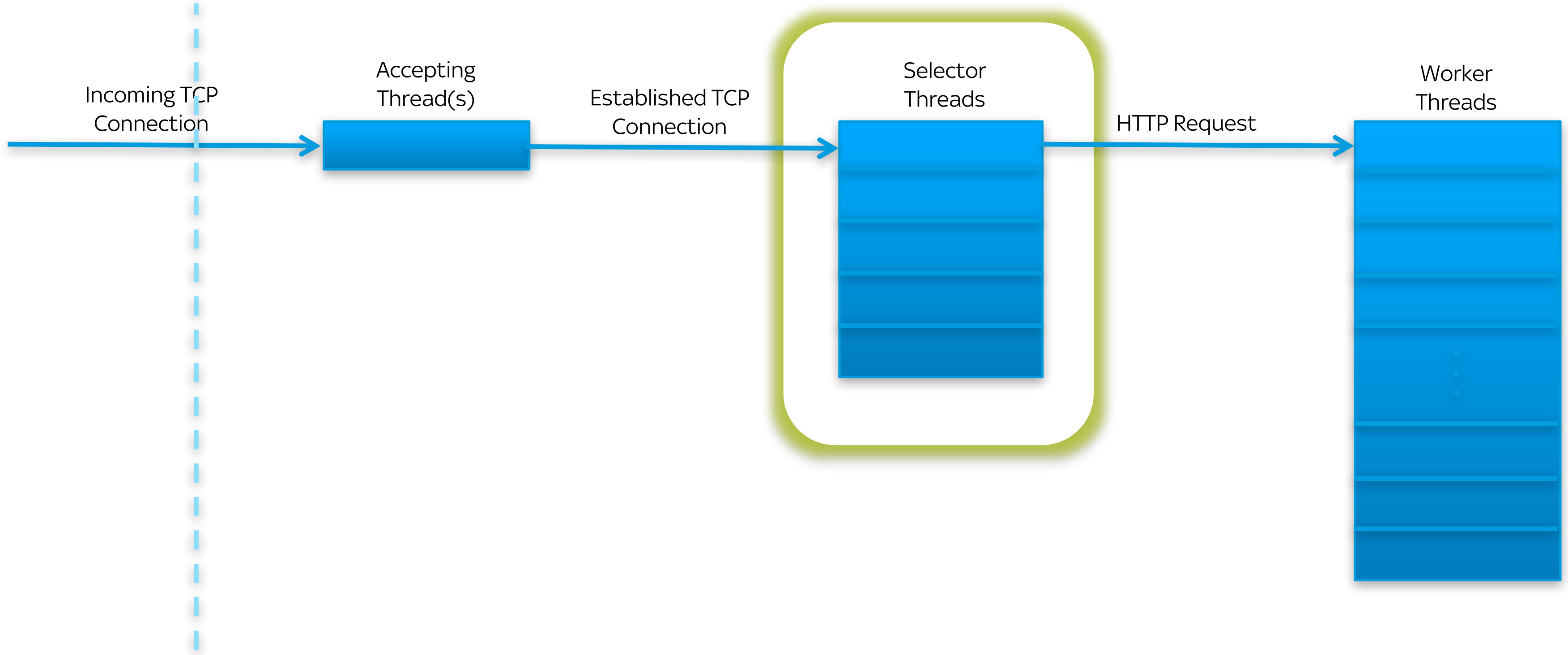
Usually issues downstream calls to other services or DB

## Web Server - BIO



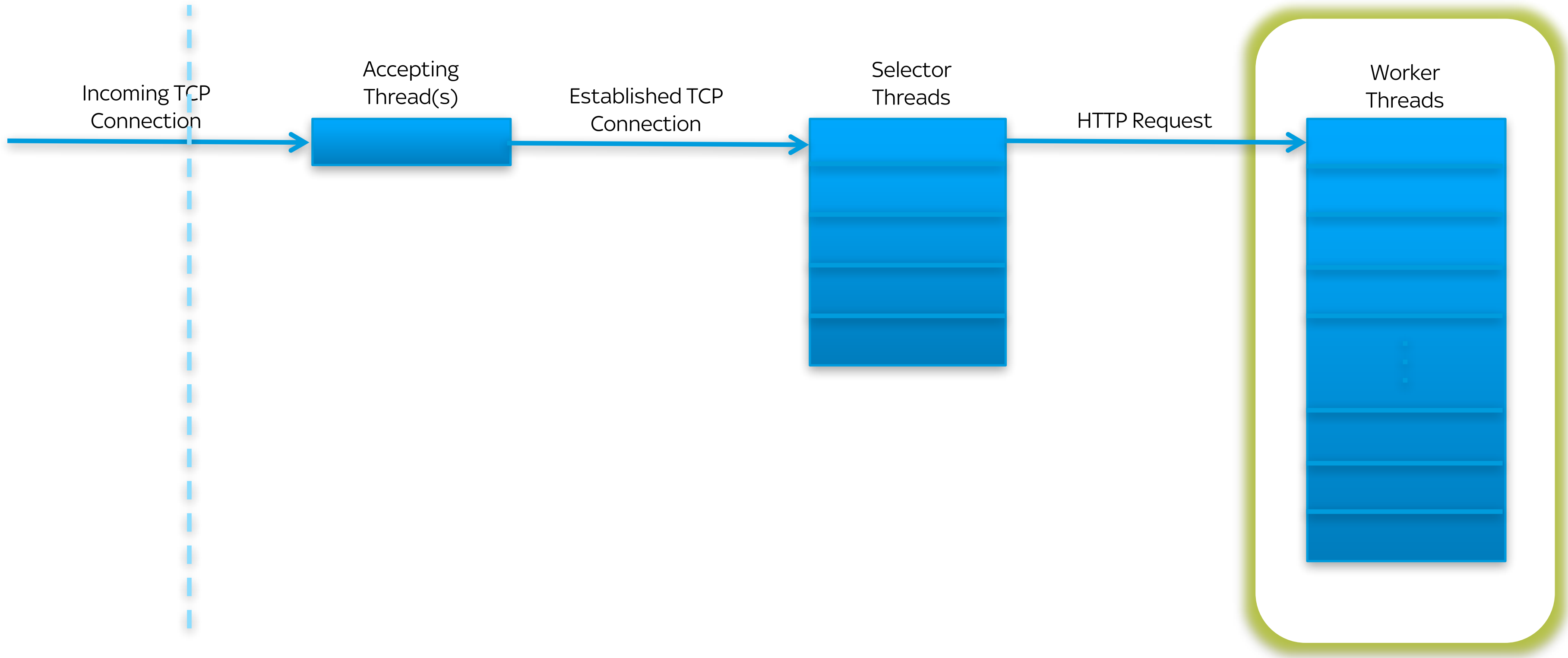
Accepts a number of incoming TCP connections
Limits the number of TCP connections open to the Java Web Server

## Web Server - NIO



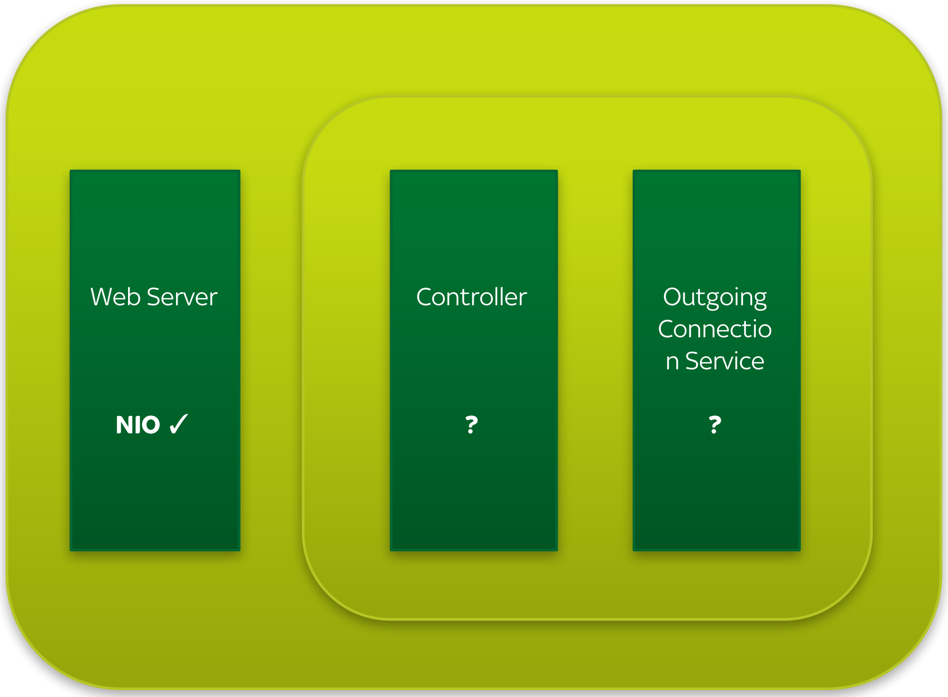
Listen for I/O in **TCP** connections
Pass **HTTP** requests to processing threads

# Web Server - NIO



Process a **single HTTP request**
Usually issues downstream calls to other services or DB

## Web Server



**NIO Web Server** can handle virtually an unlimited number of connections
It's a **configuration change**

# Web Application – BIO HTTP Client

When can outgoing connection block?

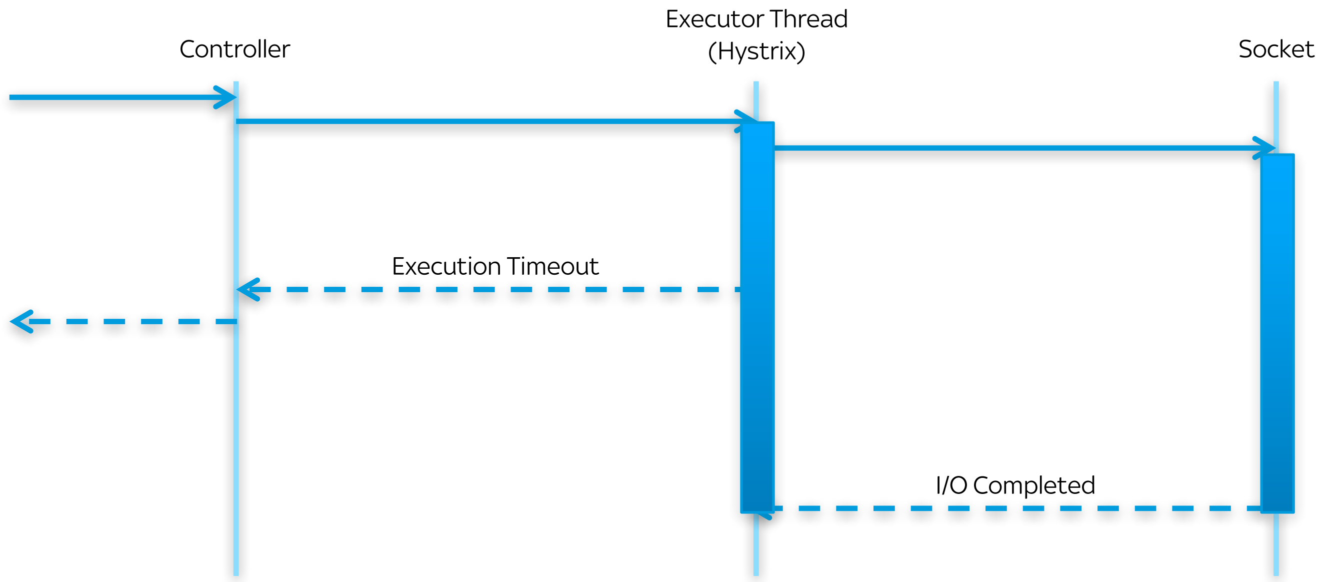
| Operation | Common name | Interruptible |
|---|---|---|
| Get connection (from pool) | Connection Request Timeout | Yes |
| Establish TCP connection | Connection Timeout | **No** |
| Request / Response | Socket Timeout<br>on (each) Read **O.S. timeout** (~15m) on Write | **No** |

## Web Application – BIO HTTP Client + Thread Pool Executor

Can a Thread Pool Executor help? (e.g. Netflix's Hystrix)

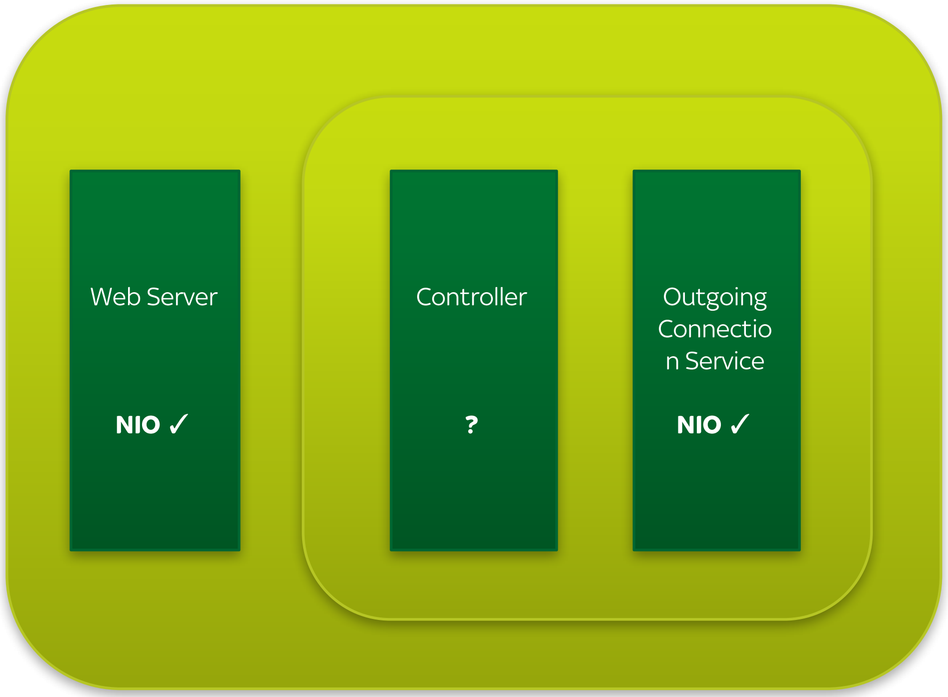
| Operation | Interruptible | Executor thread | Application thread |
|---|---|---|---|
| Get connection (from pool) | Yes | Unblocked | Unblocked |
| Establish TCP connection | **No** | **Blocked** | Unblocked |
| Request / Response | **No** | **Blocked** | Unblocked |

## Web Application – BIO HTTP Client + Thread Pool Executor



When I/O blocks so does the executor thread
The controller thread is unblocked instead

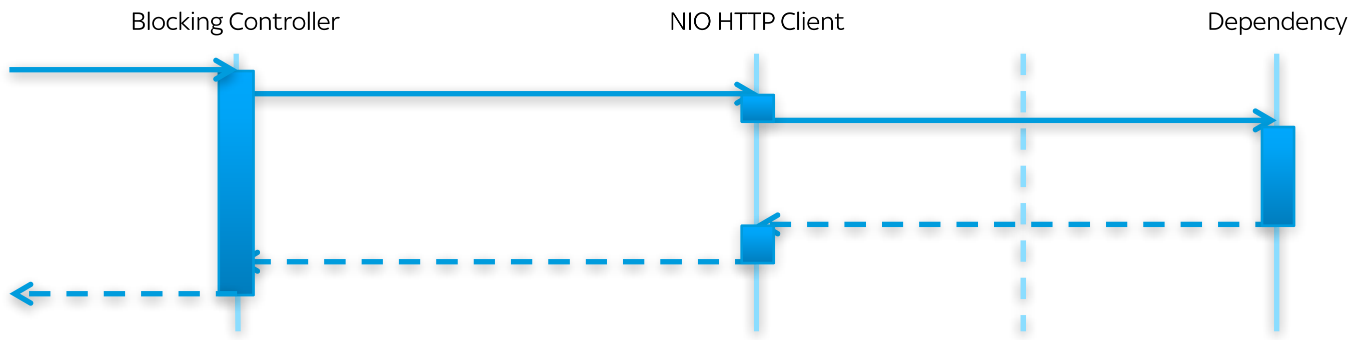
# Web Application – HTTP Client

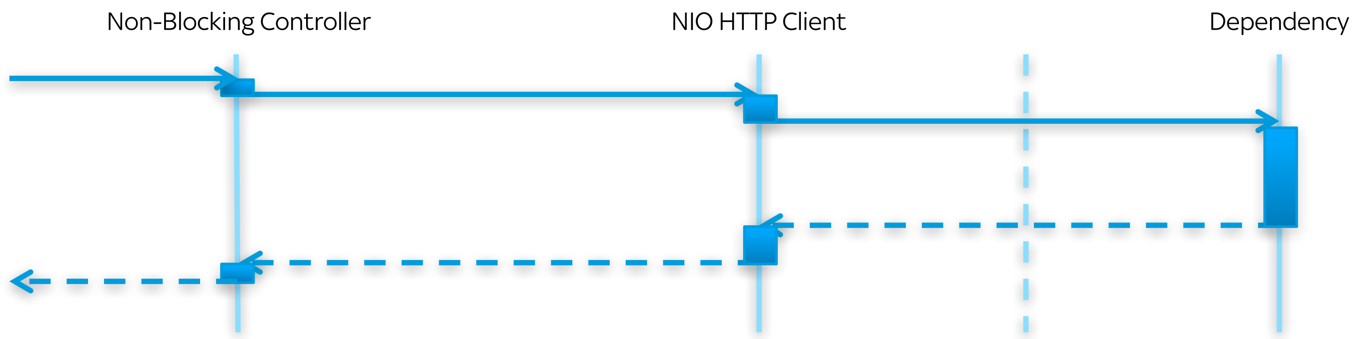


**NIO for outgoing connections** guarantees timeouts
**Easy** to implement by changing HTTP client library

Introduction
Web Server
Web Application – Outgoing connections
**Web Application – Controller**
Demo

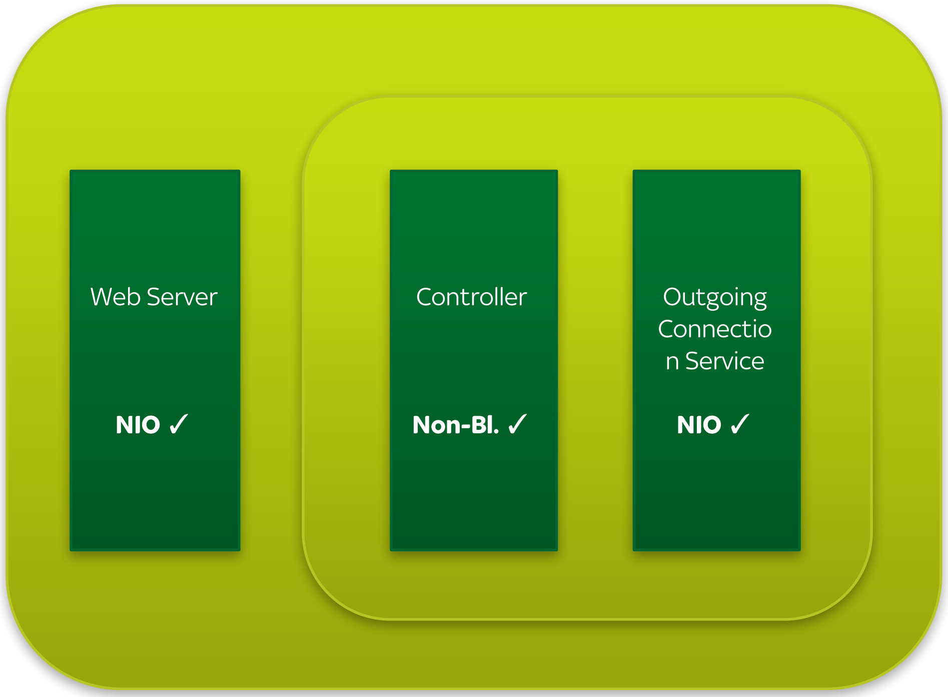
## Web Application – Blocking vs Non-Blocking Controller





Non-blocking controllers let threads free to process other requests

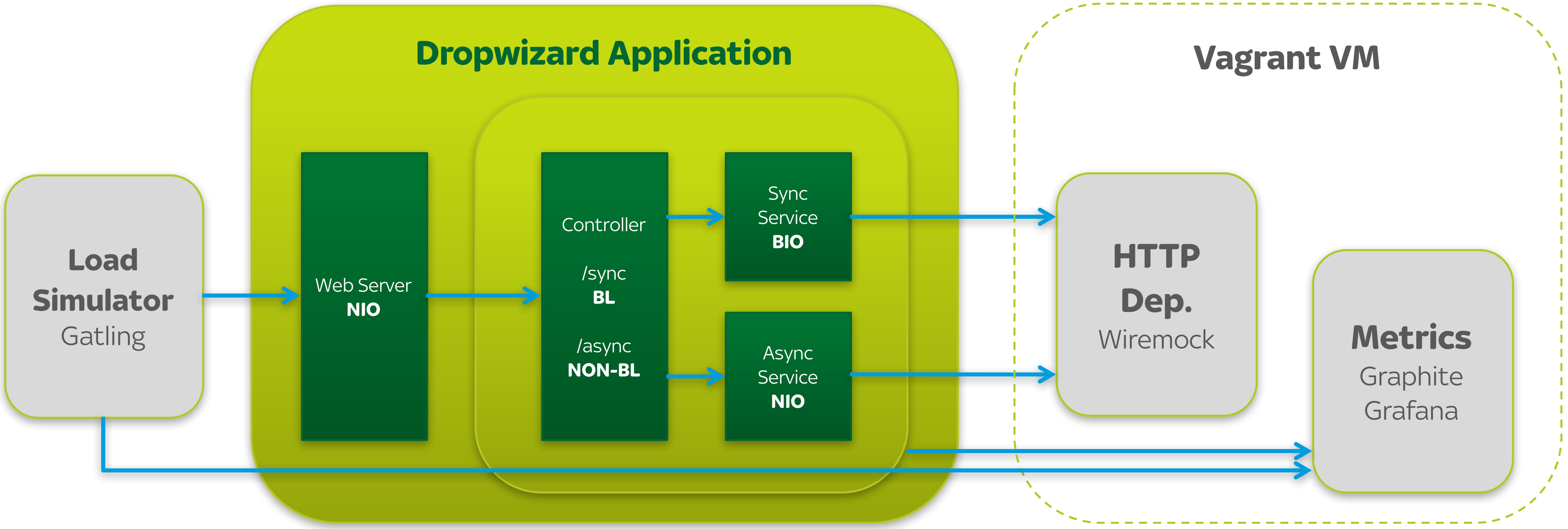
## Web Application



**NIO in the controller** decouples concurrent requests from response times
**Non-trivial** to implement in existing applications

Introduction
Web Server
Web Application – Outgoing connections
Web Application – Controller
**Demo**

## Demo



**Virtual Users: 8**

Connect Timeout: 0.1s
**Request Timeout: 3s**
**Socket Timeout: 3s**

Acceptor Threads: 1
Selector Threads: 1
**Worker Threads: 4**

Connect Request Timeout: 0.1s
Connections: 20

Connect Timeout: 0.1s
**Request Timeout (NIO): 2s**
**Socket Timeout: 2s**

Acceptor Threads: 1
Selector Threads: 1
Worker Threads: 26

https://github.com/paoloambrosio-skyuk/talk-webappthreads

## Q & A

Paolo Ambrosio
Principal Engineer – Identity

@paolo.ambrosio2
#community-java
#reactive