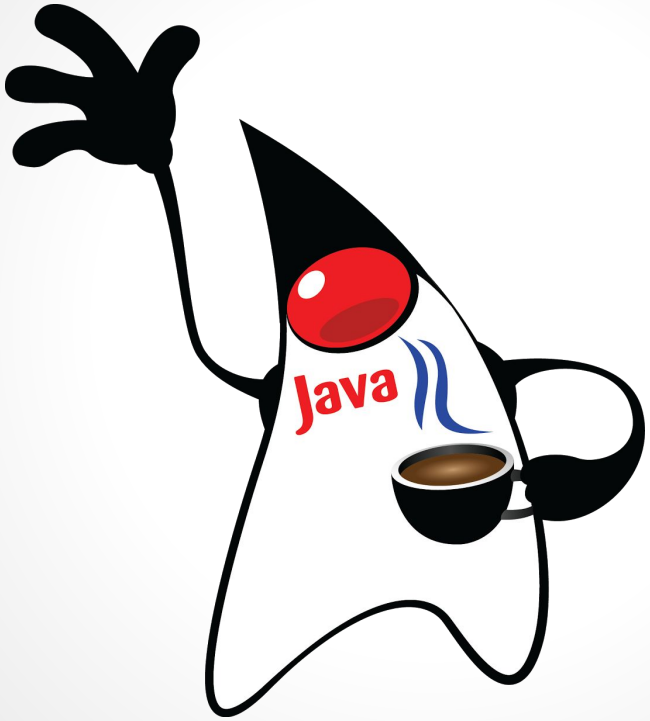
C10K and beyond

Uri Shamay

$>whoami

- BIO :: http://cmpxchg16.me
- Twitter :: https://twitter.com/cmpxchg16
- Github :: https://github.com/cmpxchg16
- Email :: shamayuri@gmail.com

$>Java.IL community co-founder



http://www.meetup.com/JavaIL/

C10K and beyond, how much beyond…….

WhatsApp 2011:

$>netstat -an | grep -c EST
$>1016313

WhatsApp 2012:

$>sysctl kern.ipc.numopensockets
$>kern.ipc.numopensockets: 2277845

What I will not cover

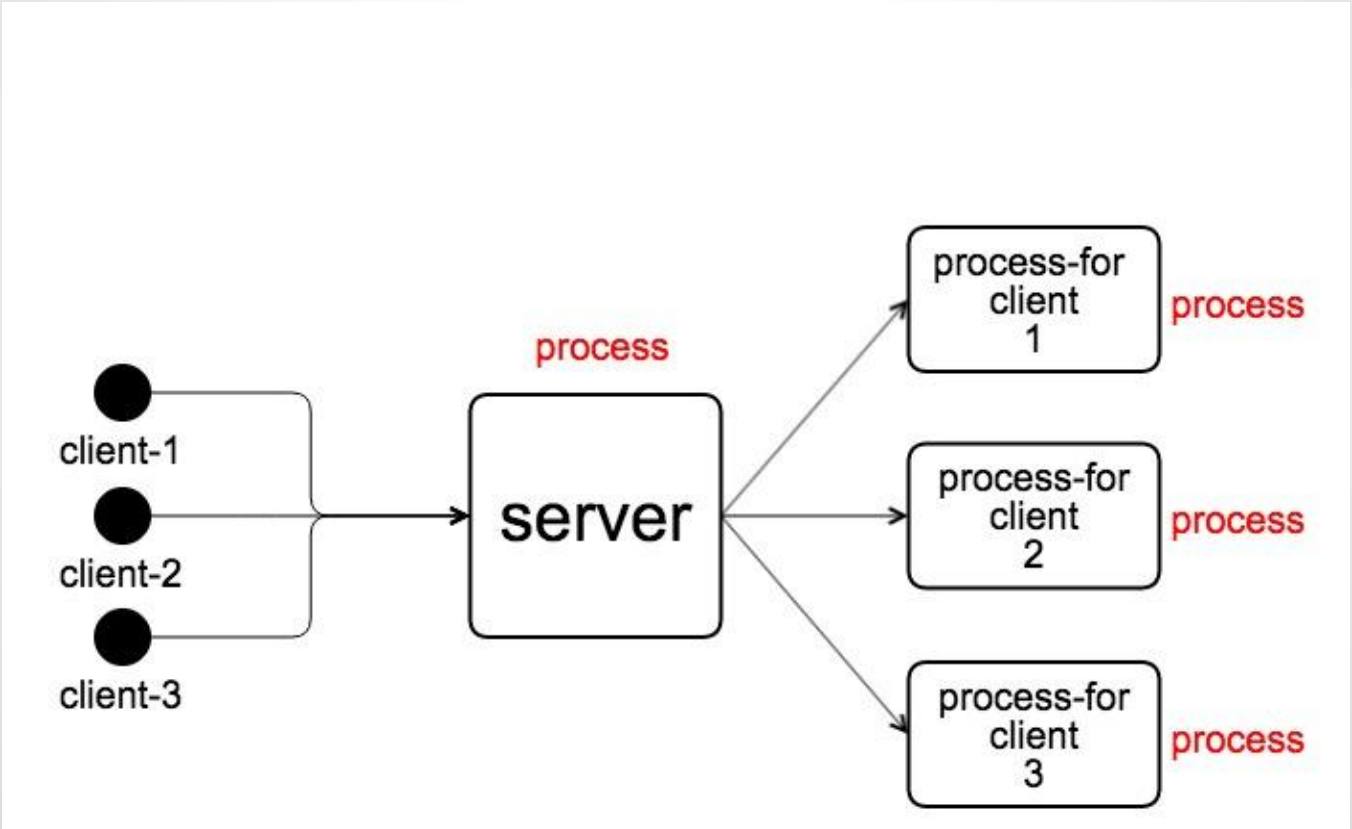
- Specific OS implementation
- OS system configuration
- Networking hard-core
- Hardware

What I will cover

- The history of C10K problem
- concurrency models abstract
  - Processes/Threads
  - Event-driven/Actors
  - lightweight threads :: Coroutines/Fibers
- Java related solution

C10K problem - Processes

The old beast - @process per connection



C10K problem - Processes

Pseudo:

```
fork () -> {
    try {
        io_func_1 ()
        io_func_2 ()
        io_func_3 ()
    } catch (timeout) {
        io_handle_timeout ()
        return
    } catch (error) {
        io_handle_error ()
        return
    }
    handle_success ()
}
```

C10K problem - Processes

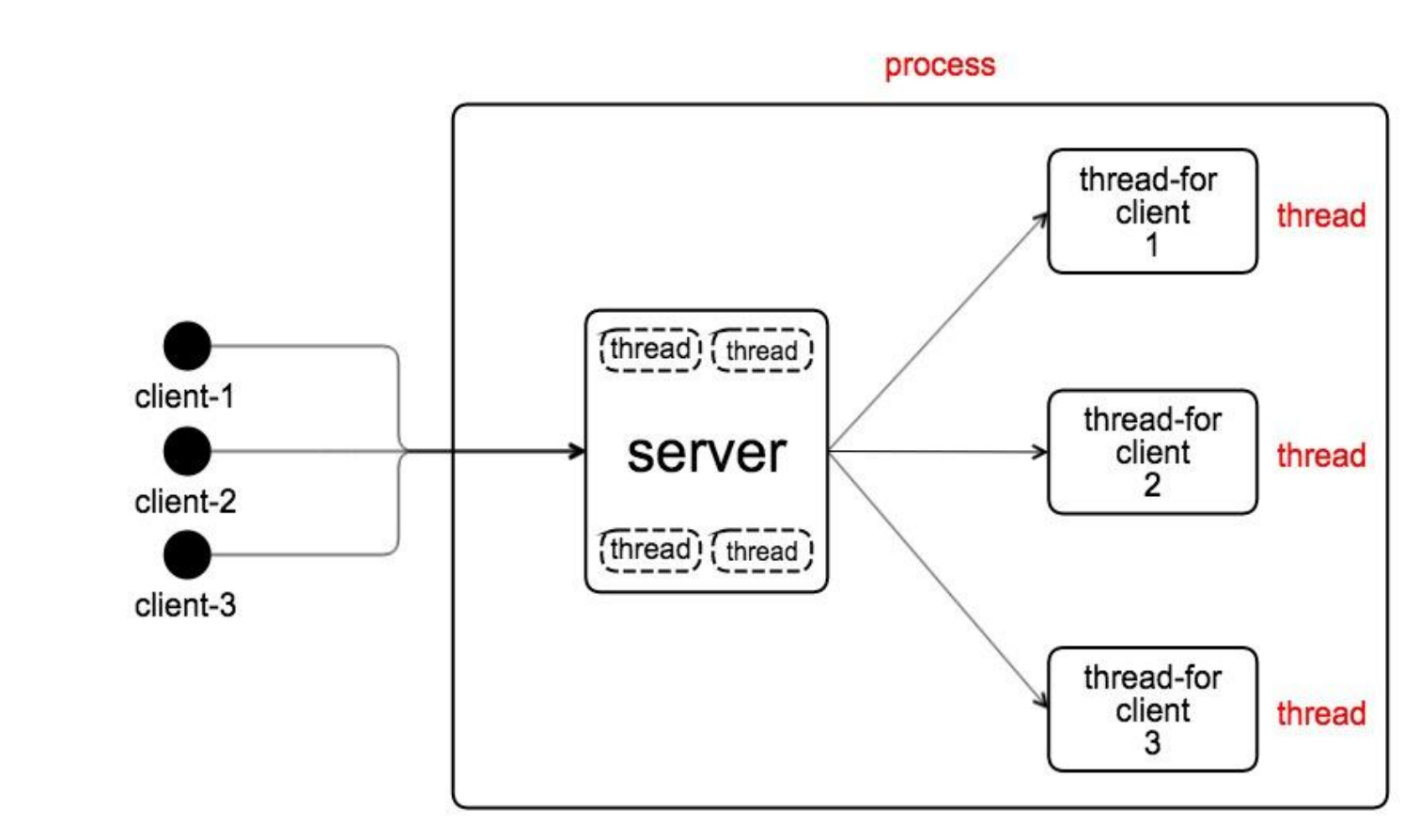
Usage:

- CGI with scripts and ENV passing
- Apache HTTP Server old model
- PostgreSQL model - even today!

But processes cannot scale as connections...

C10K problem - Threads

@native OS thread per connection



C10K problem - Threads

Pseudo:

```
thread () -> {
    try {
        io_func_1 ()
        io_func_2 ()
        io_func_3 ()
    } catch (timeout) {
        io_handle_timeout ()
        return
    } catch (error) {
        io_handle_error ()
        return
    }
    handle_success ()
}
```

C10K problem - Threads

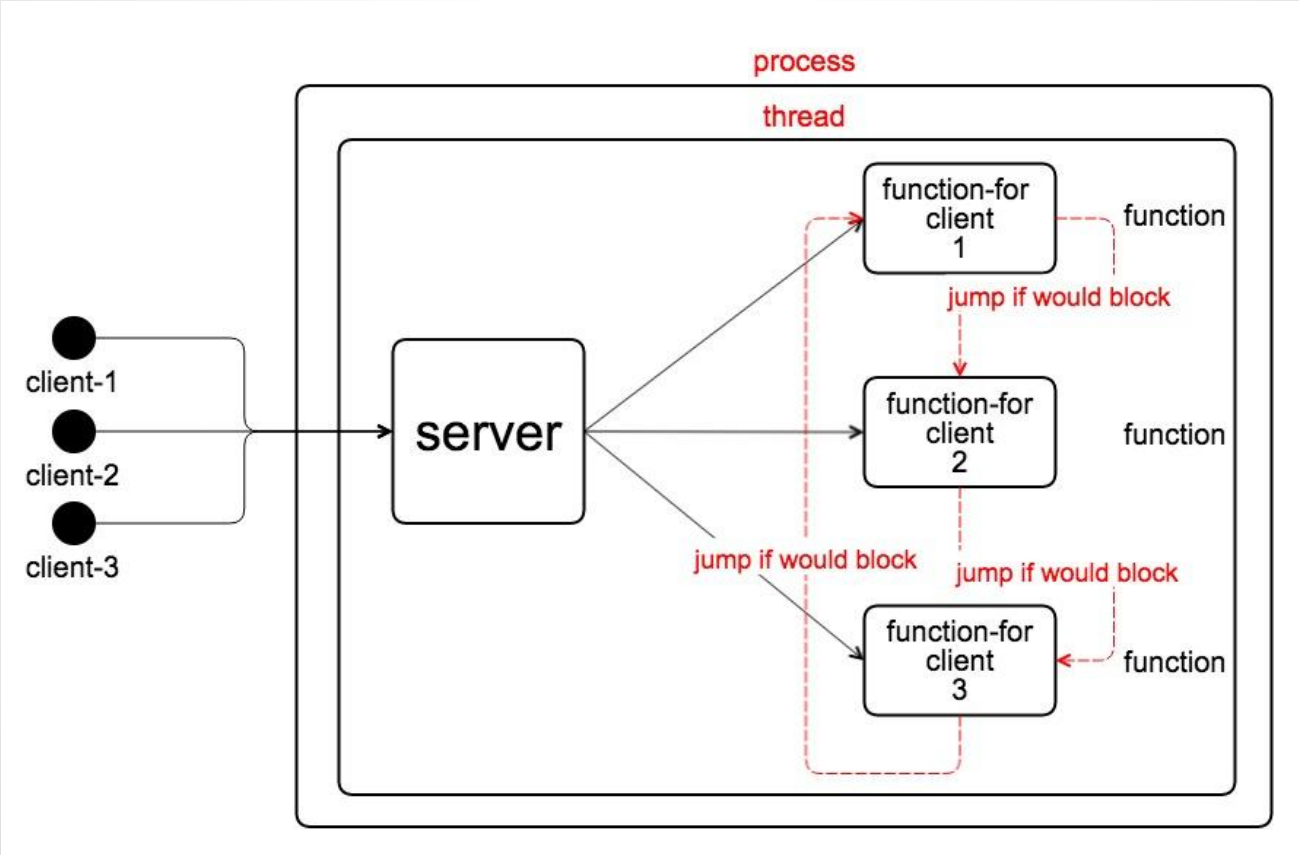
Usage:

- The old model of Apache
- Any servlet container like :: Tomcat / Jetty / Resin / Jboss / Glassfish
- Any HttpClient

But threads cannot scale as connections...

C10K problem - Event-driven

@one native OS thread rule them all!



C10K problem - Event-driven

"Hi Mate, we heard you like callbacks, so we put callbacks in your callback so you can callback when you callback."

C10K problem - Event-driven
The CALLBACK HELL!

Pseudo:

```
main_loop () -> {
    async -> (io_func_1 (state)) -> {
        onSuccess (state) {
            async -> (io_func_2 (state)) -> {
                onSuccess (state) {
                    async -> (io_func_3 (state)) -> {
                       onSuccess (state) {
                           handle_success (state)
                       }

                       onError (state) { io_handle_error () }
                       onTimeout (state) { io_handle_timeout () }
                       }
                }

                onError (state) { io_handle_error () }
                onTimeout (state) { io_handle_timeout () }
            }
        }

        onError (state) { io_handle_error () }
        onTimeout (state) { io_handle_timeout () }
    }
}
```

```
io_handle_error (state) {
    async -> (...) -> {
        onSuccess (state) {}
        onError (state) {}
        onTimeout (state) {}
    }
}

io_handle_timeout (state) {
    async -> (...) -> {
        onSuccess (state) {}
        onError (state) {}
        onTimeout (state) {}
    }
}
```

C10K problem - Event-driven

support multi-core system - open NUM_OF_PROCESSORS evented native thread's

Usage:

- servers: Nginx/Httpd/Squid
- Node.js
- Java world frameworks for client/server that scale
  :: Netty / Grizzly / Mina

C10K problem - Event-driven with promises

Promises is just a syntactic sugar for event-driven.

- Node.js
- Java 8: CompletableFuture

```
main_loop () -> {
    async -> (io_func_1 (state))
    -> then (io_func_2 (state))
    -> then (io_func_3 (state))
    -> success (handle_success())
    -> error (io_handle_error ())
    -> timeout (io_handle_timeout ())
}
```

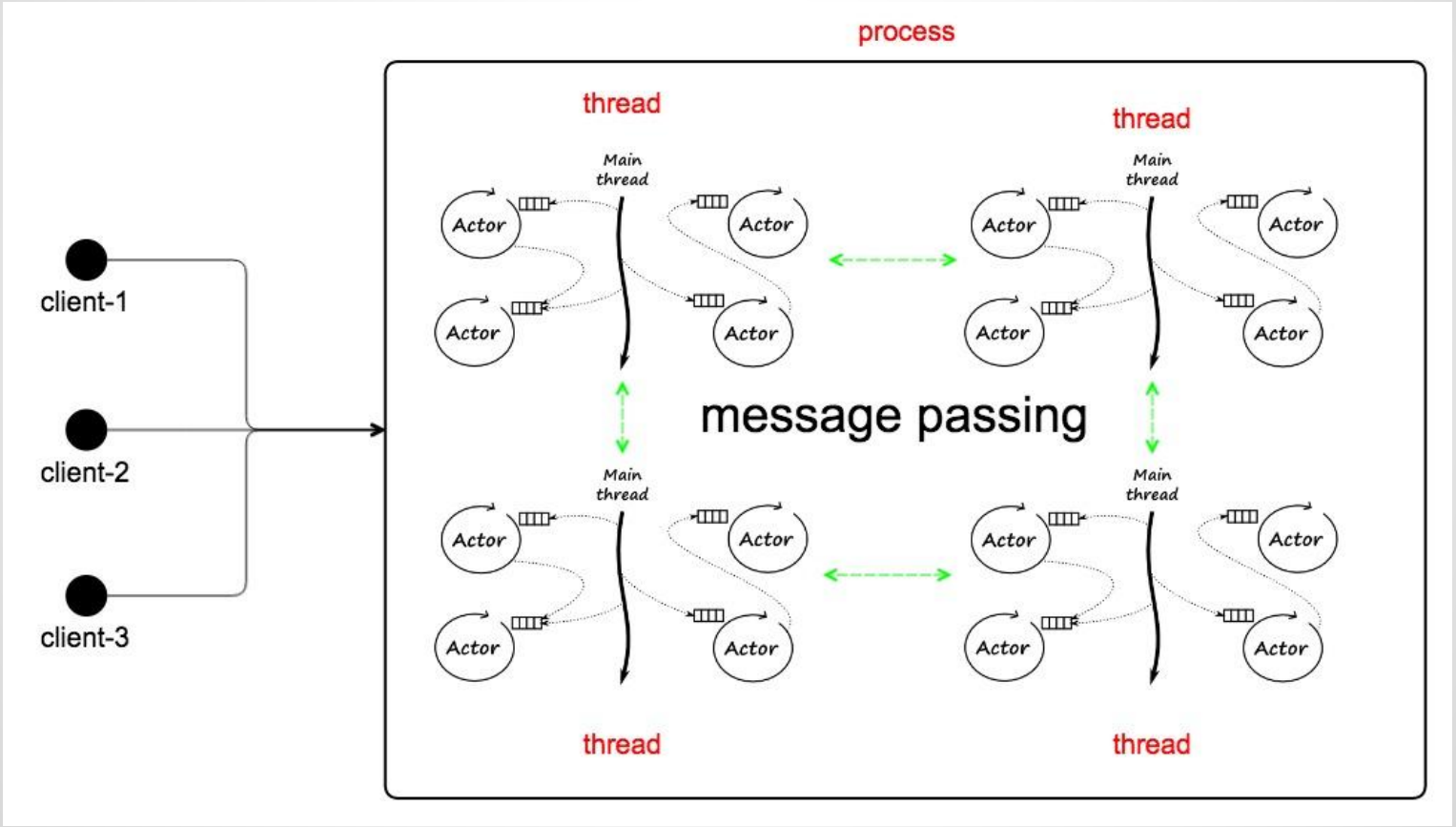
But what if our I/O is endless like WebSocket?!

C10K problem - Actor

Don't communicate by sharing memory, share memory by communicating

C10K problem - Actor

@one Actor per connection



C10K problem - Actor

- There are a lot of frameworks out there
- Java :: Akka (concrete implementation for Actor model in Scala)

But Actor model is asynchronous by design and our flow is synchronous…

C10K problem - Coroutines/Fibers

Let's face it - I/O is **asynchronous**, our state is **synchronous**, so let's be cooperative by hybrid mode

C10K problem - Coroutines example

Pseudo:

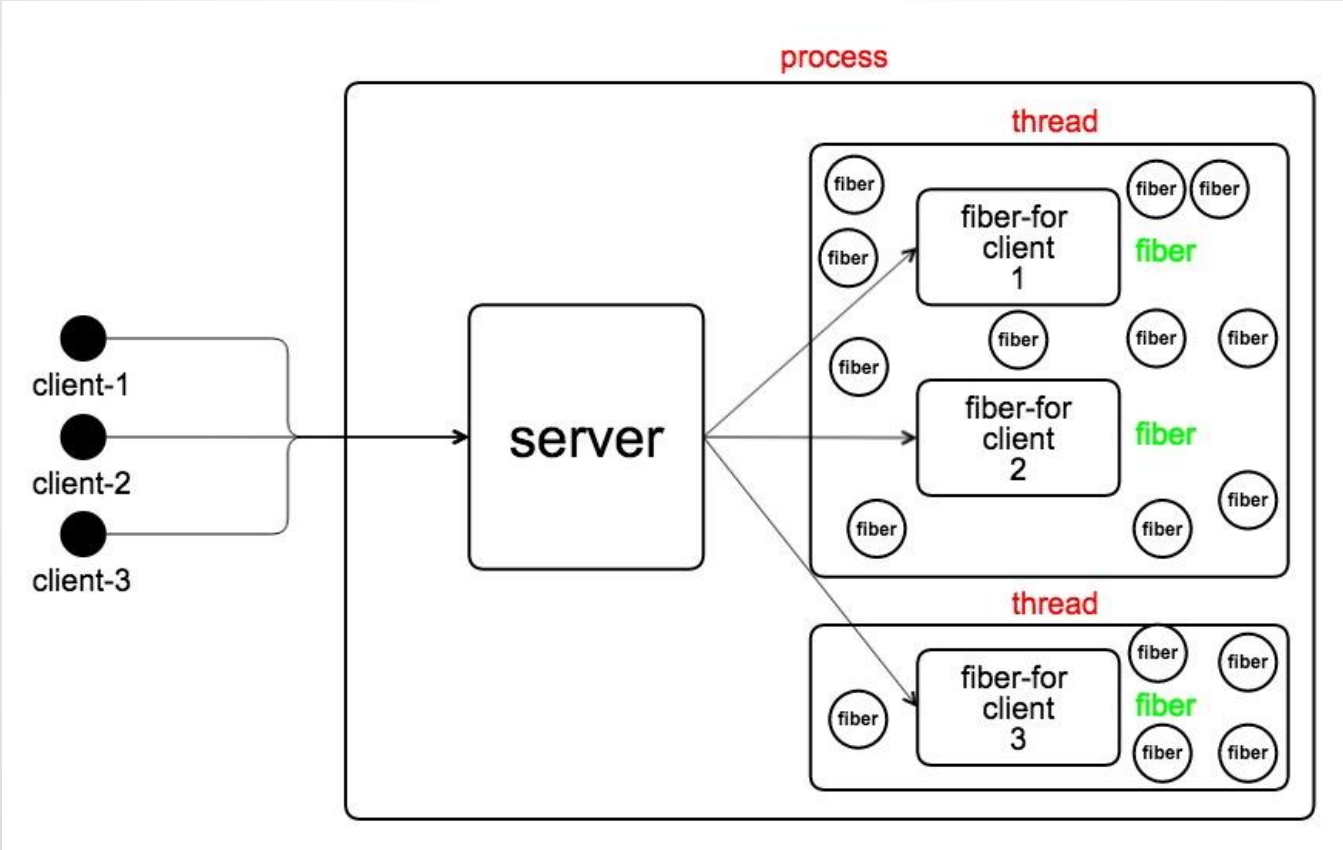
```
coroutine_1 () -> {
    i := -1
    while (true) {
        i++
        if i % 2 == 0 {
            println “even is the dominant ($i)!”
            if i < 10 {
                yield ()
            }
        }
    }
}

coroutine_2 () -> {
    i := 0
    while (true) {
        i++
        if i % 2 == 1 {
            println “odd is the dominant ($i)!”
            yield ()
        }
    }
}
// only one native thread
main {
    coroutine_1.start ()
    coroutine_2.start ()
}
```

Output:

```
even is the dominant (0)!
odd is the dominant (1)!
even is the dominant (2)!
odd is the dominant (3)!
even is the dominant (4)!
odd is the dominant (5)!
even is the dominant (6)!
odd is the dominant (7)!
even is the dominant (8)!
odd is the dominant (9)!
even is the dominant (10)!
even is the dominant (12)!
even is the dominant (14)!
even is the dominant (16)!
even is the dominant (18)!
even is the dominant (20)!
.
.
.
```

C10K problem - Coroutines/Fibers

@one coroutine/fiber per connection



C10K problem - Coroutines/Fibers

Pseudo:

```
fiber () -> {
    try {
        io_func_1 () // implicit yield() when blocking
        io_func_2 () // implicit yield() when blocking
        io_func_3 () // implicit yield() when blocking
    } catch (timeout) {
        io_handle_timeout () // implicit yield() when blocking
        return
    } catch (error) {
        io_handle_error () // implicit yield() when blocking
        return
    }
    handle_success() // implicit yield() when blocking
}
```

C10K problem - Coroutines/Fibers

- New programming languages includes that model:
  GO (golang) / D (dlang) / Rust (via library)
- Java
  - Kilim (Actor semantics)
    - compile time bytecode weaving
  - Quasar (Fibers as a lightweight threads)
    - runtime bytecode weaving with javaagent

C10K problem - Coroutines/Fibers

Caveats

- As any transparent concurrency model, also that model suffers from blocking calls - be careful when using third libraries
- OS Scheduling is very optimized to CPU & I/O
  - If we have a CPU intensive task our I/O corporative model is “unfair” (e.g SSL/TLS, compression/decompression)

C10K and beyond

Uri Shamay

Questions?