FUNCTIONAL SERVERS
Not only NodeJS

Jarek Ratajski

- Developer, wizard, anarchitect
- Lives in Luzern
- Works for Engenius GmbH
- C64, 6502, 68000, 8086, C++, Java, JEE, Scala, Akka, JS, ScalaJS....
- jratajski@gmail.com
- @jarek000000

Że co?

- Ancient times
- Dark ages
- Future<Either<Stupid, Awesome>>

Scala programmer confronts Java project that uses Maven, Spring, and Hibernate"
Salvador Dali Oil painting 1946 (Classic Programmers @progpaintings)

CGI

COMMON GATEWAY INTERFACE

- HTTP GET "localhost" "/path/name?query_string"
- SERVER parses http
- SERVER finds a script/program for that
- SERVER starts program giving PARAMS, INPUT and OUTPUT
- PROGRAM/SCRIPT runs - stdout is html output, stdin is request data

CGI in Java

- HTTP GET "localhost / Kto tam ?"
- SERVER parses http
- SERVER finds a Java program (jar) for that
- SERVER starts program (like JVM) giving PARAMS, INPUT and OUTPUT
- JVM starts
- JVM reads classes and starts main class
- Main class does the job:

```
System.out.println("Java ! ") ;
```

PUK PUK

- "Kto tam ?"
-
-
-
-
-
-

''Java ! ''

Acient Model

- 1 request - 1 proces
- Huge RAM overhead
- Slow startup time

Application servers

Application servers

- One big process to server http
- Programs as plugins (servlets)
- Lots of configurations (XMLs)

Dark ages model

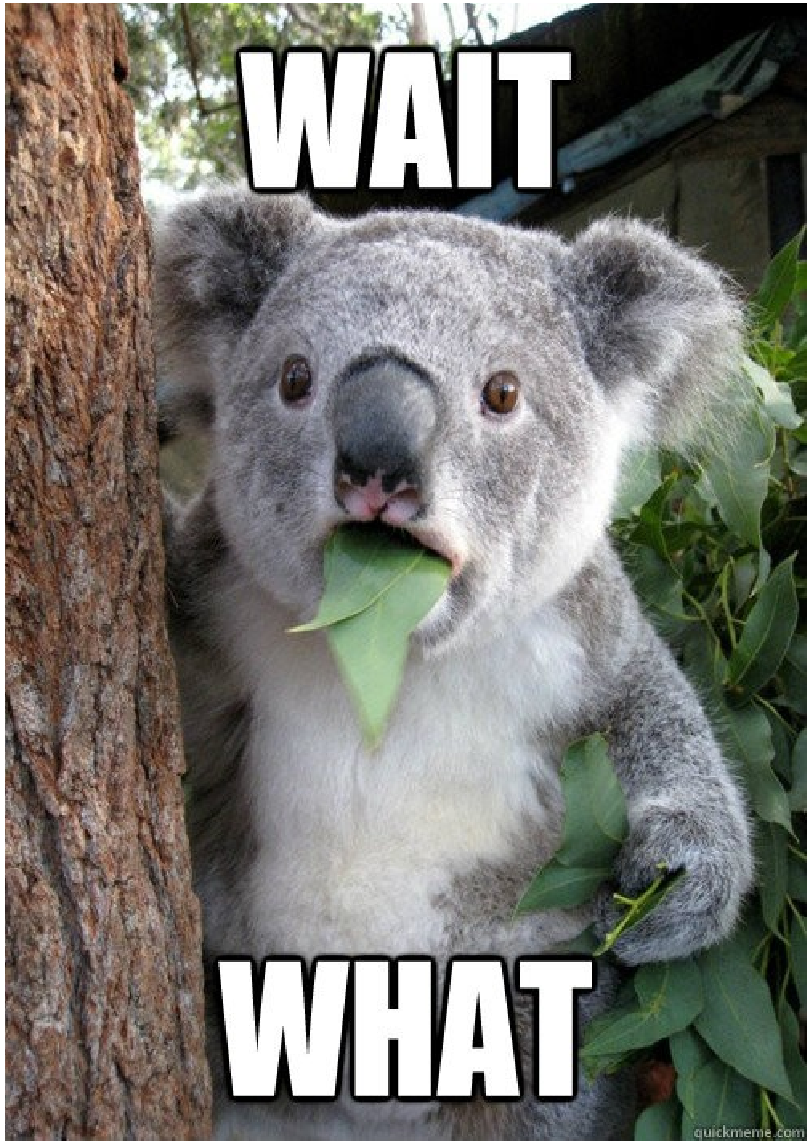
- 1 request – 1 thread
- Smaller memory overhead
- Very slow start of system
- Then quite fast

Problems of application servers

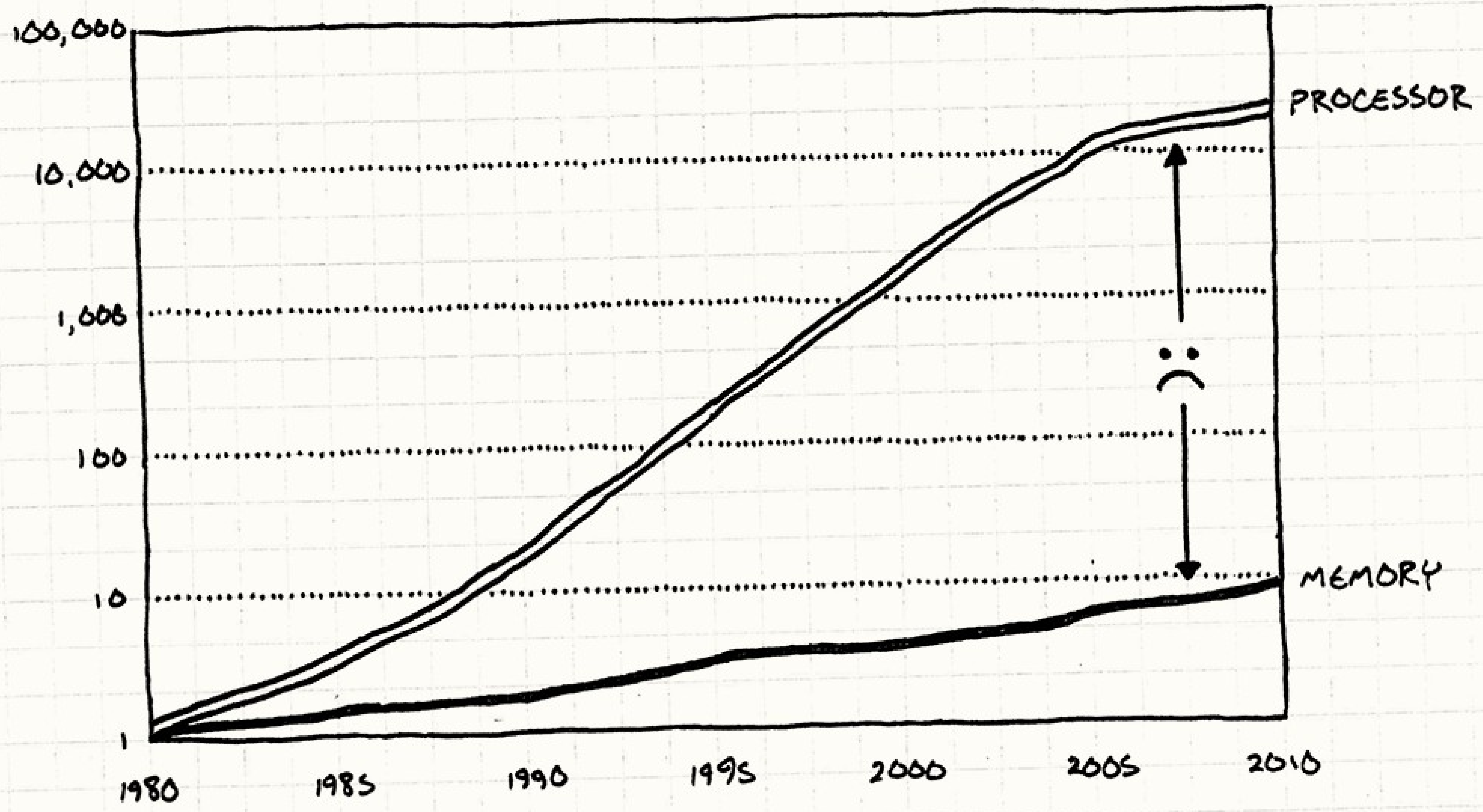
- Configuration
- Testing
- Magic in code
- Migrations
- Monoliths

Perfect when

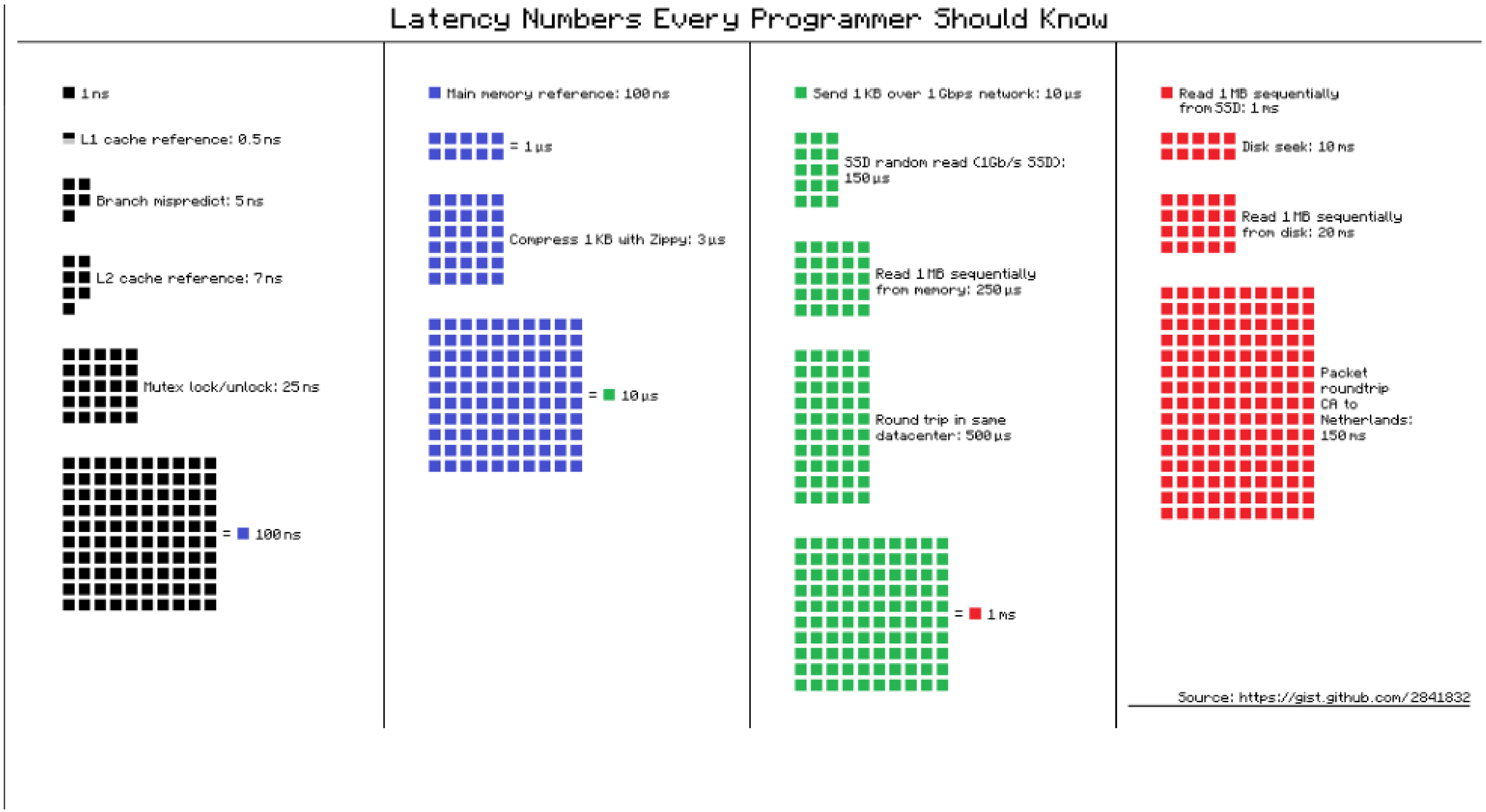
- RAM was still enough fast (vs CPU)
- One CPU - one CORE
- Small RAM (128-256mb)



Houston, we have a problem
DEMOTIVATORS.RU

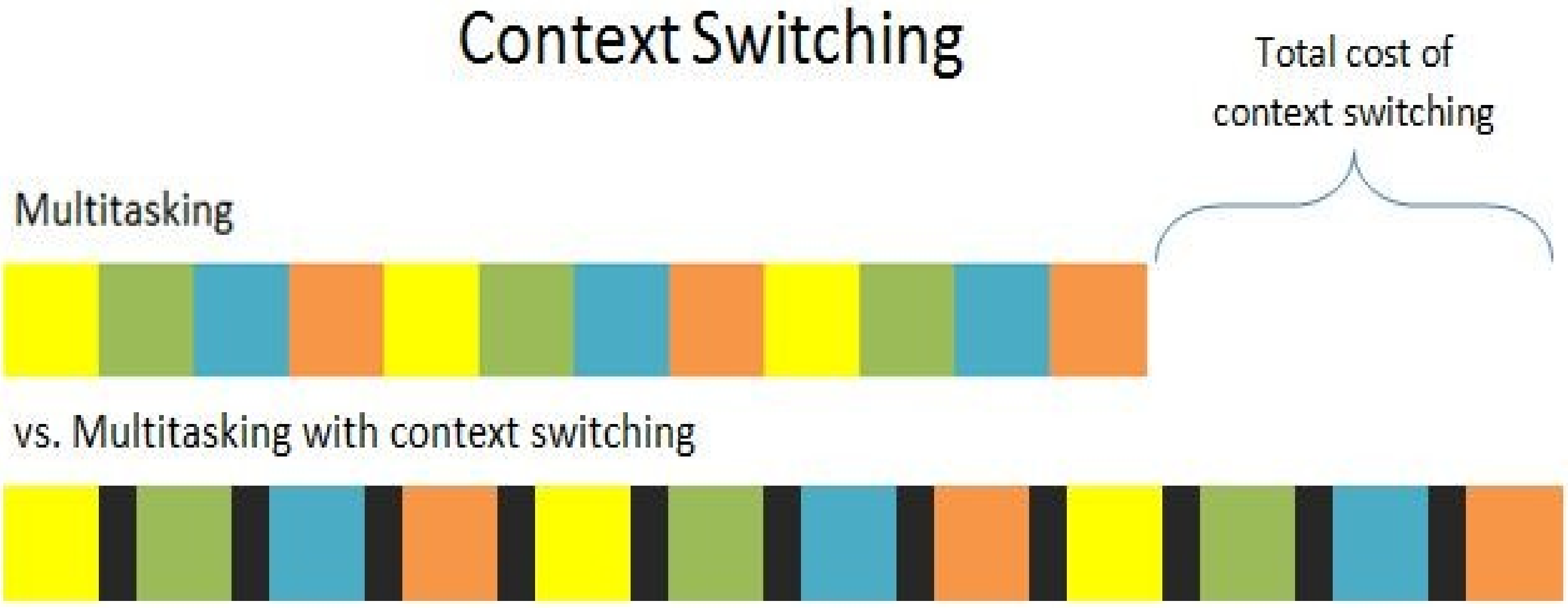

100,000
10,000
1,000
100
10
1
1980
1985
1990
1995
2000
2005
2010
PROCESSOR
MEMORY
:(

Latencies



Source: https://gist.github.com/2841832

Contex switching


Context Switching
Total cost of context switching
Multitasking
vs. Multitasking with context switching

From https://www.bryanbraun.com/2012/06/25/multitasking-and-context-switching/

WHAT IF I TOLD YOU
THAT WAS JUST THE BEGINNING
imgflip.com

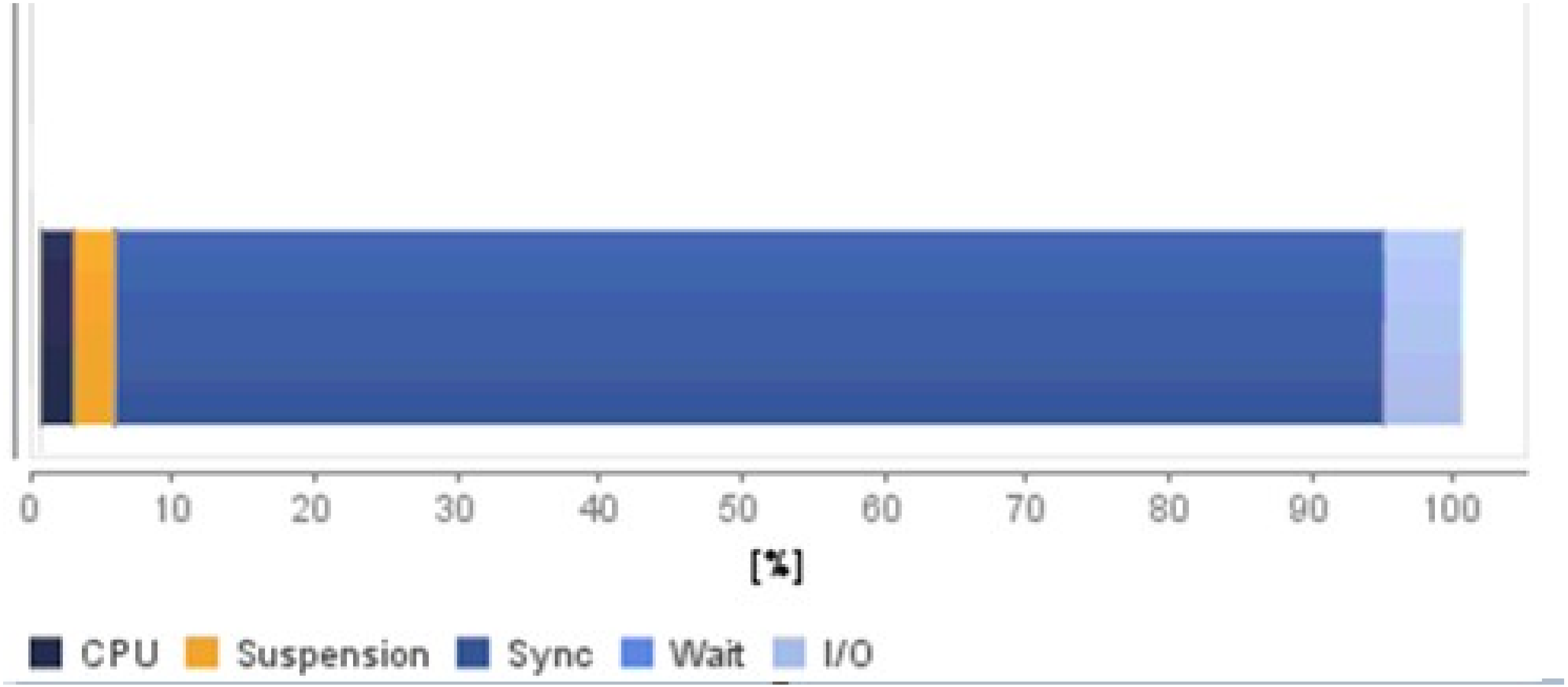
Synchronization

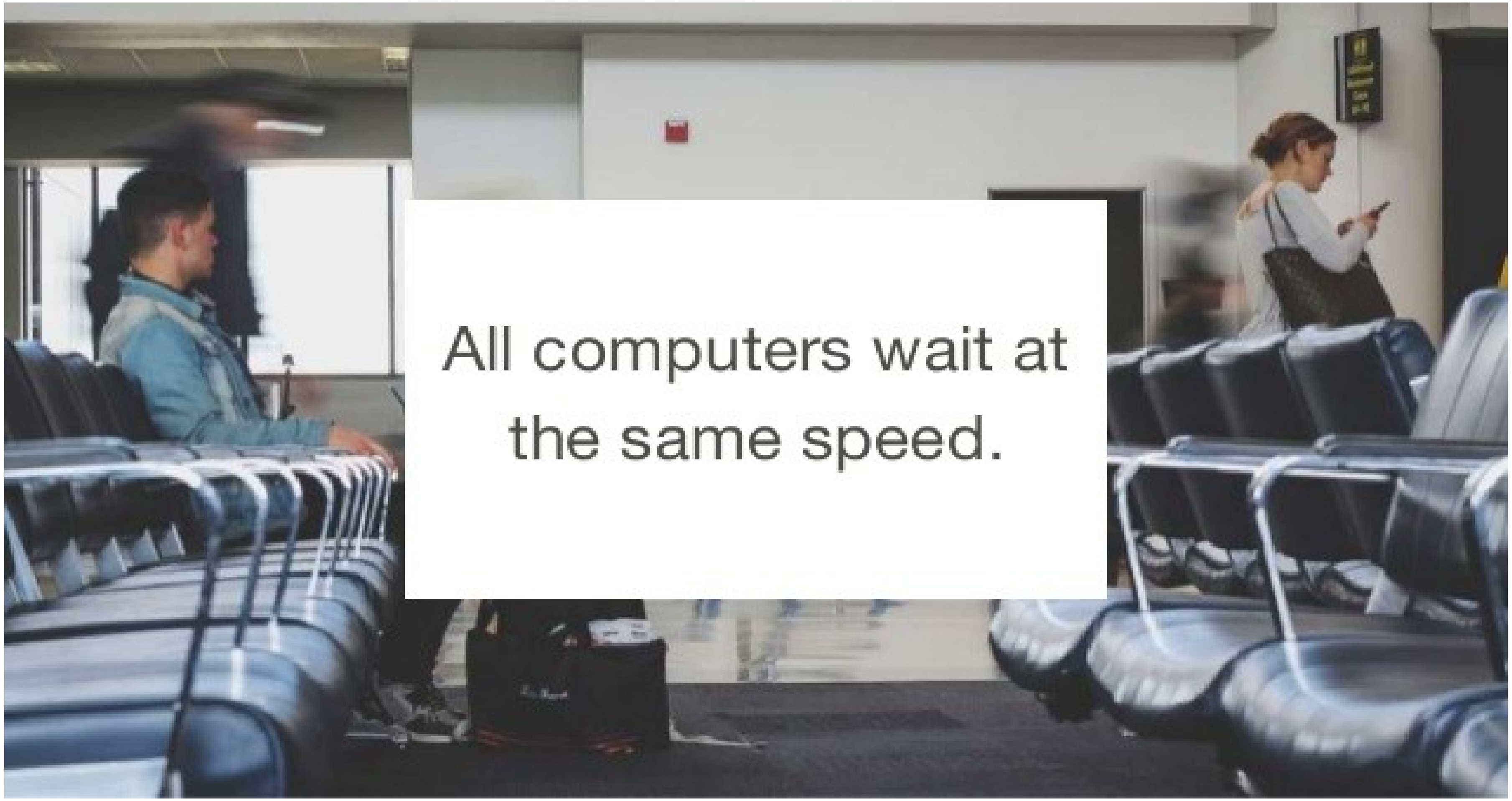
```
private int  counter;


synchronized (this) {
    counter = counter + 1 ;
}
```

Synchronization

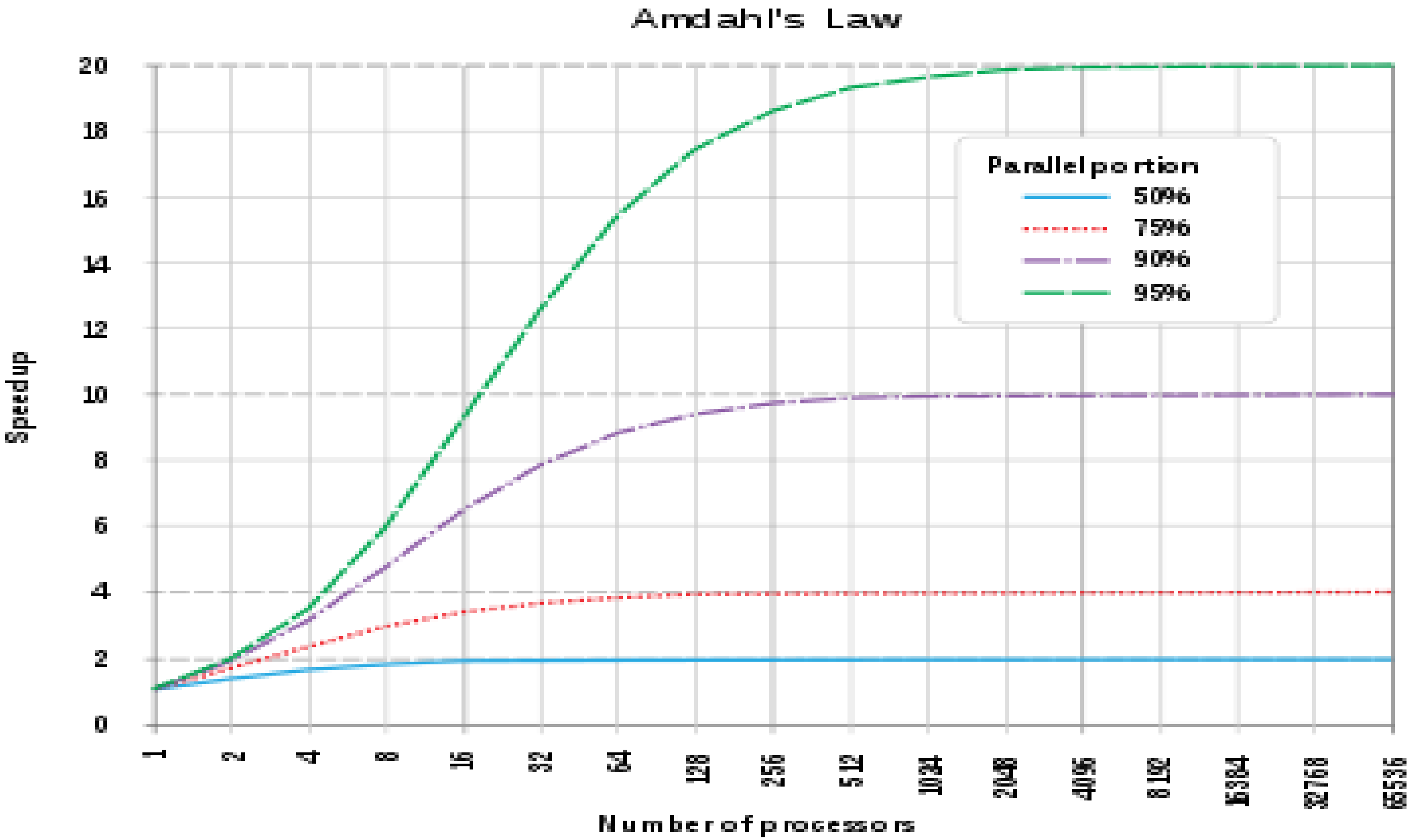
```
private int  counter;


synchronized (this) {
    counter = counter + 1 ;
    readFile() ;
    saveToDb() ;
}
```

How it ends...





All computers wait at the same speed.


Amdahl's Law
Parallel portion
5096
7596
9096
9596
Speedup
0
2
4
6
8
10
12
14
16
18
20
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of processors

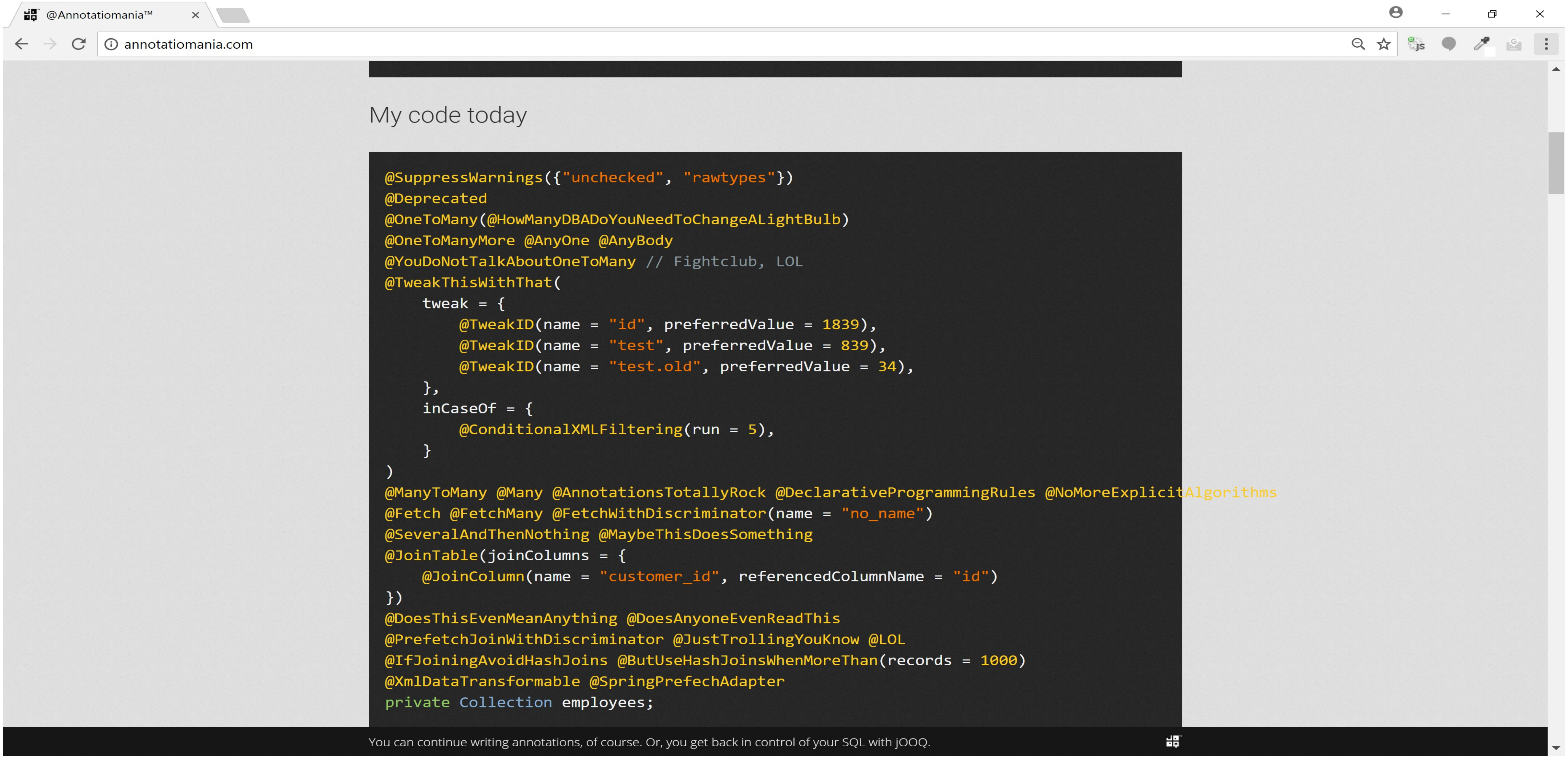
The biggest problem of mainstream java WEB frameworks

spring

Java EE™

MAGIC

Wait... where is XML?

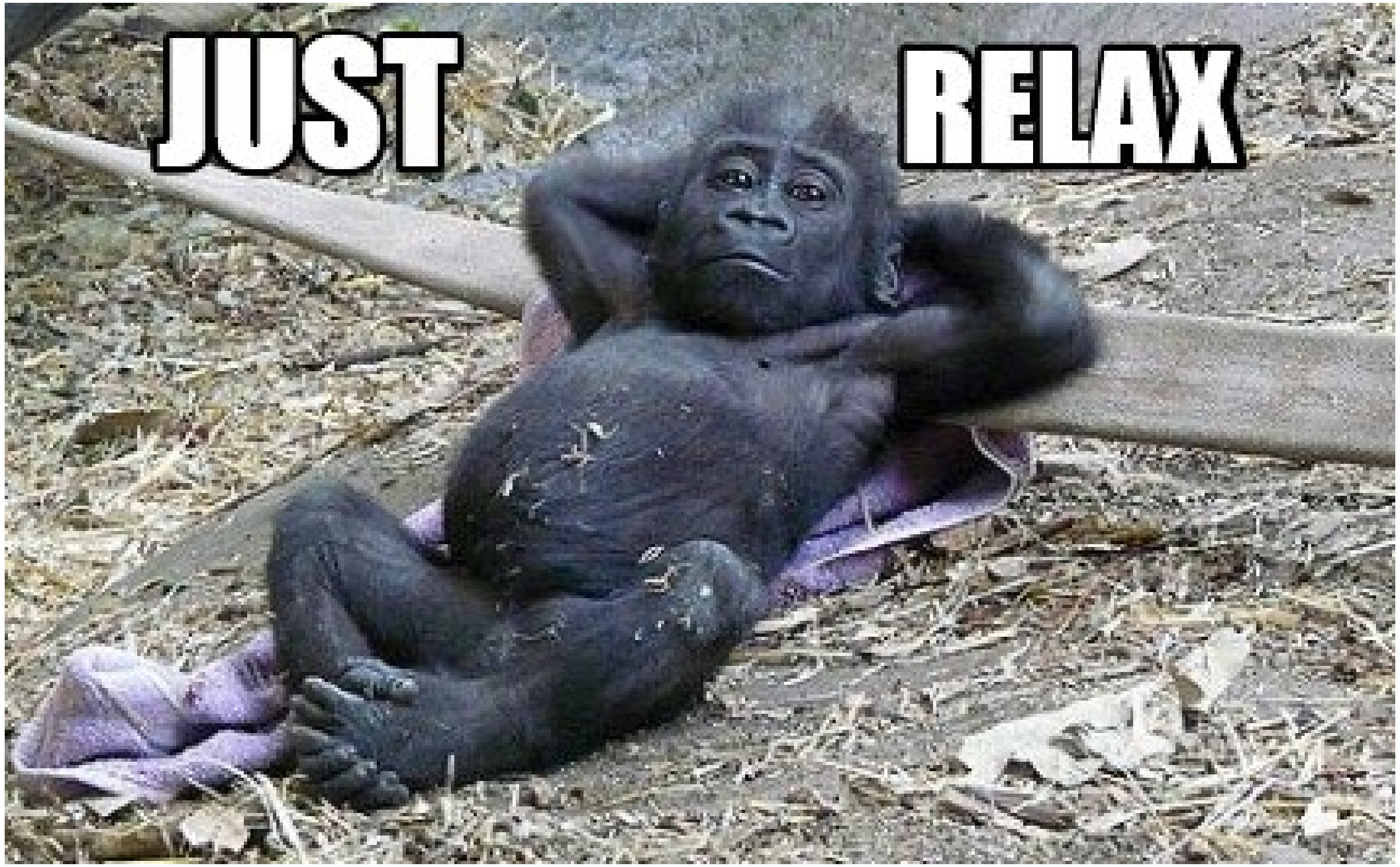


My code today

```
@SuppressWarnings({"unchecked", "rawtypes"})
@Deprecated
@OneToMany(@HowManyDBADoYouNeedToChangeALightBulb)
@OneToManyMore @AnyOne @AnyBody
@YouDoNotTalkAboutOneToMany // Fightclub, LOL
@TweakThisWithThat(
    tweak = {
        @TweakID(name = "id", preferredValue = 1839),
        @TweakID(name = "test", preferredValue = 839),
        @TweakID(name = "test.old", preferredValue = 34),
    },
    inCaseOf = {
        @ConditionalXMLFiltering(run = 5),
    }
)
@ManyToMany @Many @AnnotationsTotallyRock @DeclarativeProgrammingRules @NoMoreExplicitAlgorithms
@Fetch @FetchMany @FetchWithDiscriminator(name = "no_name")
@SeveralAndThenNothing @MaybeThisDoesSomething
@JoinTable(joinColumns = {
    @JoinColumn(name = "customer_id", referencedColumnName = "id")
})
@DoesThisEvenMeanAnything @DoesAnyoneEvenReadThis
@PrefetchJoinWithDiscriminator @JustTrollingYouKnow @LOL
@IfJoiningAvoidHashJoins @ButUseHashJoinsWhenMoreThan(records = 1000)
@XmlDataTransformable @SpringPrefechAdapter
private Collection employees;
```

You can continue writing annotations, of course. Or, you get back in control of your SQL with jOOQ.

STOP THIS MADNESS
makeameme.org


JUST
RELAX

HTTP Server 201x in Java

```
public static void main(String... args) throws Exception {
  RatpackServer.start(server -> server
    .handlers(chain -> chain
      .get("pukpuk", ctx -> ctx.render("Java"))
    )
  );
}
```

Spring 5 (WebFlux)

```
HandlerFunction<ServerResponse> helloWorld =
    request -> ServerResponse.ok().body(fromObject("Hello World I am Spring 5"));
```

NodeJS

```
const express = require('express')
const app = express()

app.get('/pukpuk', (req, res) => res.send('NodeJS'))
```

ONE THREAD
TO SERVE THEM ALL
imgflip.com

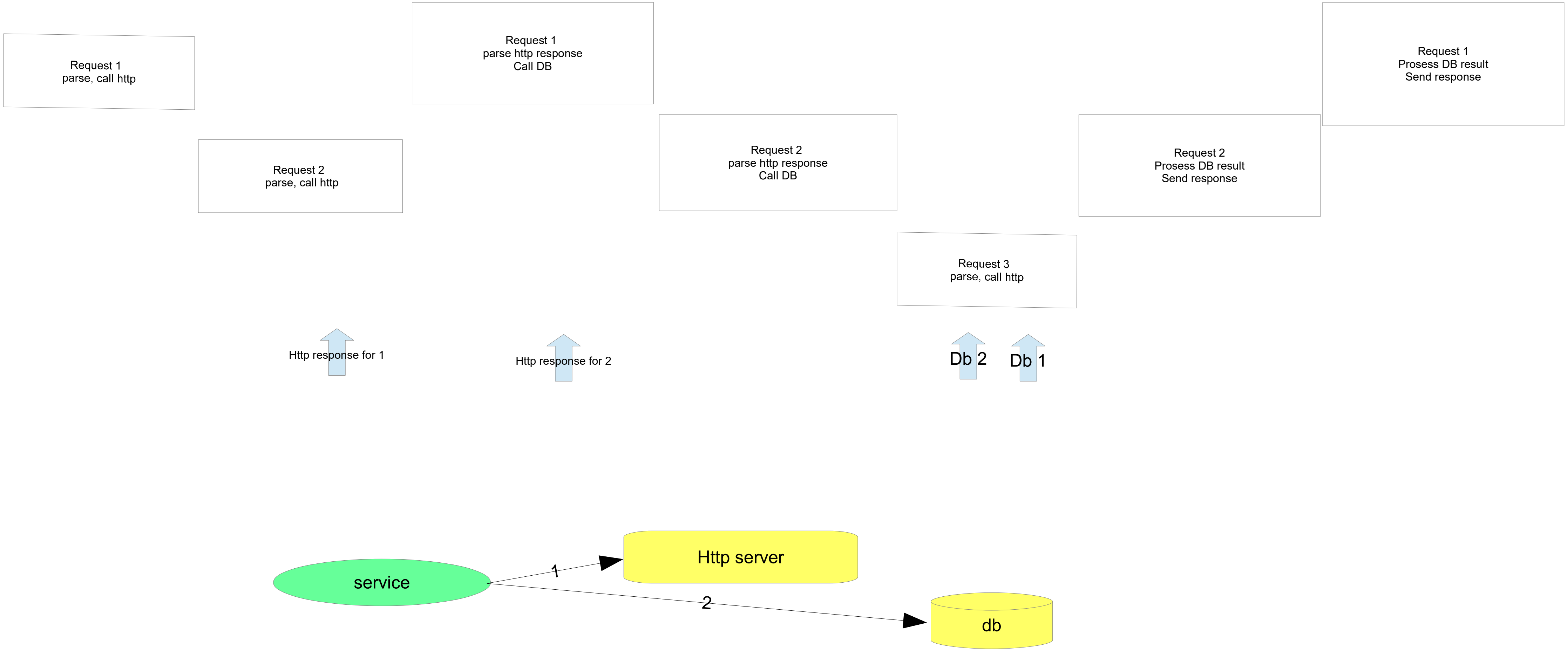
Akka HTTP

```
 val route =
      path("pupuk") {
        get {
          complete(HttpEntity(ContentTypes.`text/html(UTF-8)`,
"Scala!"))
        }
      }
```

Non blocking

```
RouterFunction<?> route = route(GET("/fib/{n}"),
        request -> {
          final int n = Integer.parseInt(request.pathVariable("n"));
          if (n < 2) {
            return ServerResponse.ok().contentType(MediaType.TEXT_HTML).body(fromObject(String.valueOf(n)));
          } else {
            Mono<Integer> n_1 = webClient.get("http://localhost:8080", "/fin{n}", n-1);
            Mono<Integer> n_2 = webClient.get("http://localhost:8080", "/fin{n}", n-2);


            Mono<String> result = n_1.flatMap( a -> n_2.map(b  -> a+b)).map(String::valueOf);
            return  ServerResponse.ok().contentType(MediaType.TEXT_HTML).body(fromPublisher(result,
String.class));
          }
        });
```

Non blocking architecture



Request 1
parse, call http

Request 2
parse, call http

Request 1
parse http response
Call DB

Request 2
parse http response
Call DB

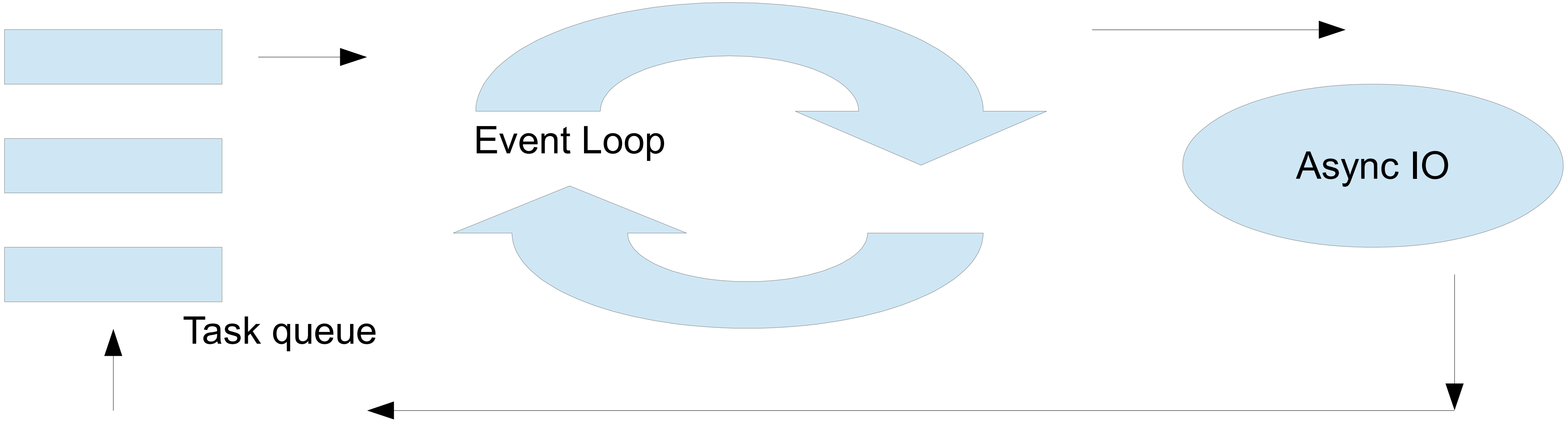
Request 3
parse, call http

Request 2
Prosess DB result
Send response

Request 1
Prosess DB result
Send response

Non blocking

- Tasks/events queue
- Executors
- Small number of threads
- Effective use of CPU caches



Why it works?

- Context switch happens when YOU want it
- Promotes no synchronization
- We eat more RAM for process... but less for threads

Future (is Now) model

- 1 thread – n requests
- Async io
- Do not block (please)

Problems

- Need async IO
- Functional code
- Monads

How to survive in a functional code?

Functional code

- High order functions
- Composition of functions

```
private HandlerFunction<ServerResponse> makeSecure( HandlerFunction<ServerResponse>  f) {
    return request -> {
      final String cookie = List.ofAll(request.headers().header("Authorization")).mkString();
      if ( cookie.equals("It is me")) {
        return f.handle(request);
      } else {
        return ServerResponse.status(HttpStatus.UNAUTHORIZED).body(fromObject("You evil"));
      }
    };
  }

      ...
RouterFunction route = nest( path("/socks"),
      route(
            POST("/new"), (postNewSock()))
      .andRoute(
            POST("/{name}/pair/{second}"), (pairSocks()))
      .andRoute(
           GET("/"), (listSocks()))
    );
```

```
private HandlerFunction<ServerResponse> makeSecure( HandlerFunction<ServerResponse>  f) {
        return request -> {
            final String cookie = List.ofAll(request.headers().header("Authorization")).mkString();
            if ( cookie.equals("It is me")) {
                return f.handle(request);
            } else {
                return ServerResponse.status(HttpStatus.UNAUTHORIZED).body(fromObject("You evil"));
            }
        };
    }

            ...
RouterFunction route = nest( path("/socks"),
            route(
                        POST("/new"), (makeSecure(postNewSock())))
            .andRoute(
                        POST("/{name}/pair/{second}"), (makeSecure(pairSocks())))
            .andRoute(
                       GET("/"), (listSocks()))
        );
```

Learn power of monads

```
userService.getUserFromDB(10001)
        .flatMap(user -> user.getGroup() )
        .flatMap(group -> group.getGroupOwner() )
        .ifPresent(user-> user.sendMessage("Hallo there!"));
```

Embrace immutability

- Scala case classes
- Kotlin data classes
- Java..... ... .. (OMG Lombok)
- Immutable collections :vavr.io

Exceptions

- **No more exceptions**
- Either Either or Try
- (Validation, \/ ...)

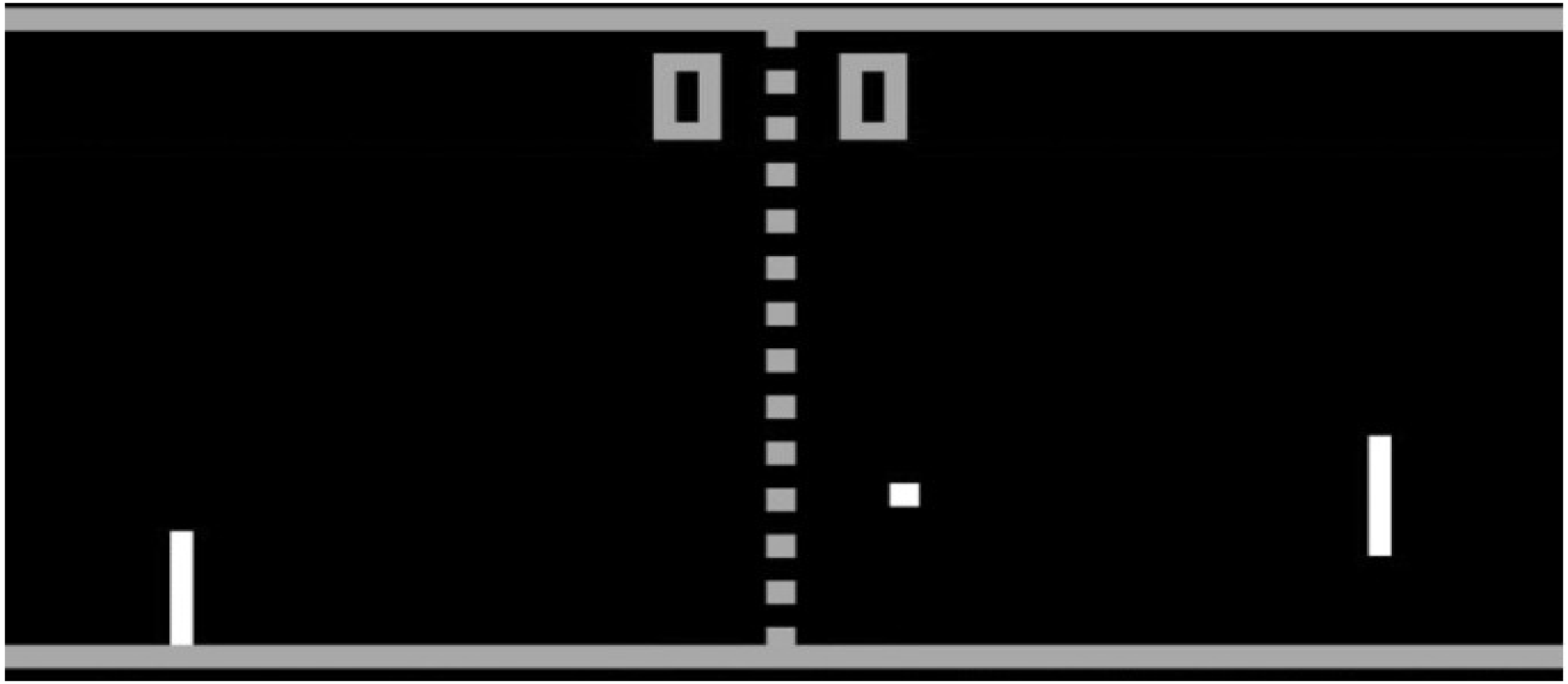
Testing !!!

```
@Test
 public void shouldRegisterUser() throws Exception {
   prepareServer().test(
         testHttpClient -> {
            final String response = testHttpClient.requestSpec(rs ->
                 rs.headers( mh -> mh.add("Content-type", "application/json"))
                 .body( body -> body.text("{\"password\": \"upa\"}")))
                 .post("/api/users/aa")
                 .getBody().getText();
                   assertEquals("{\"problem\":null,\"ok\":true}", response);

         }
    );
 }
```

Work with blocking core

```
public class ScoresRepositoryProcessor {

    private final Executor writesExecutor =     Executors.newSingleThreadExecutor();

    private final ScoresRepository repository;

    public void registerScore(List<ScoreRecord> rec){

            this.writesExecutor.execute(()->repository.registerScore(rec));

    }


}
```

Example - Pong check (1973)



https://github.com/javaFunAgain/ratpong

End

Q?

@jarek000000
jratajski@gmail.com