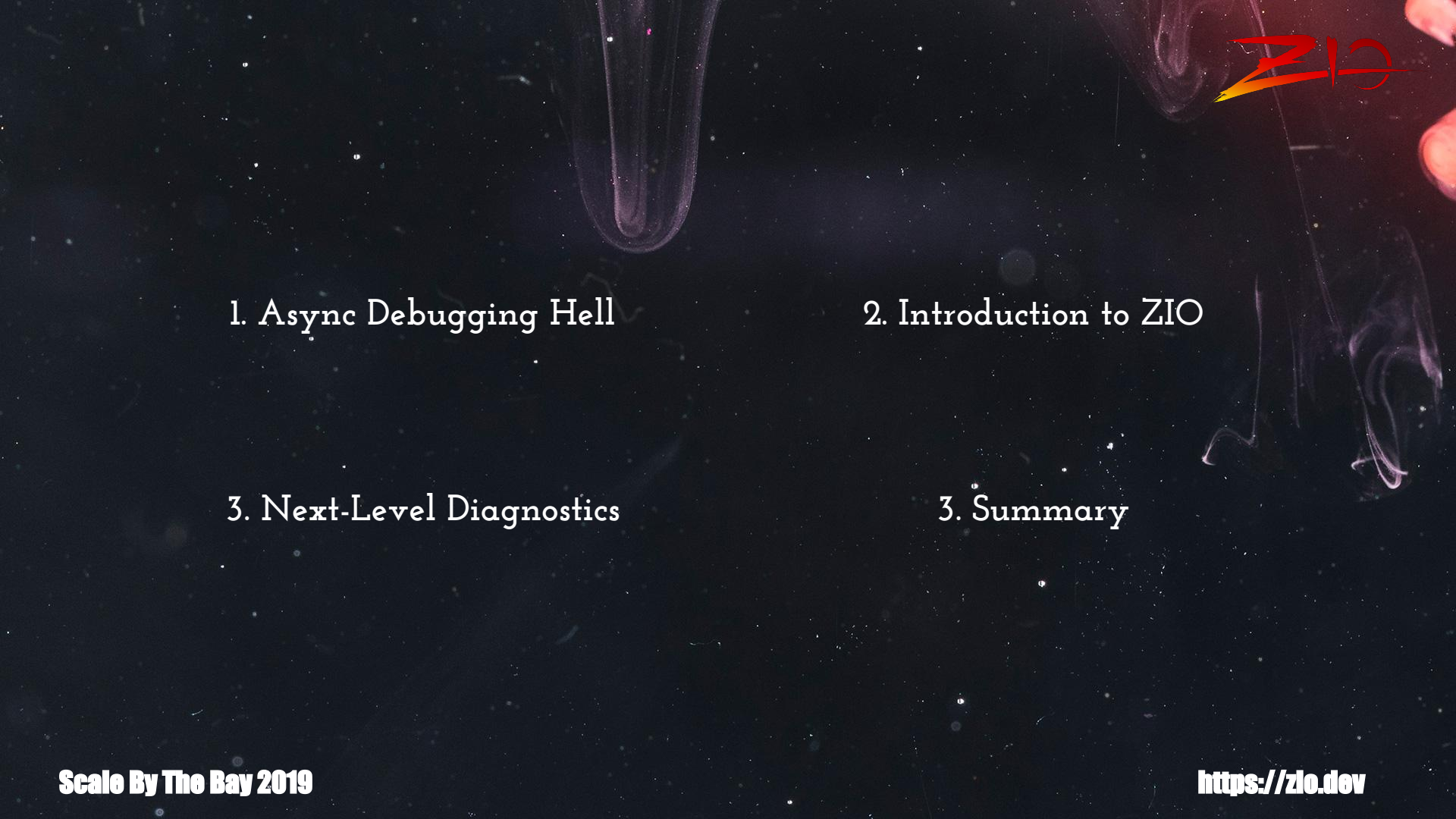
Next-Level Diagnostics for Async & Concurrent Errors with ZIO

John A De Goes @jdegoes
Salar Rahmanian @SalarRahmanian

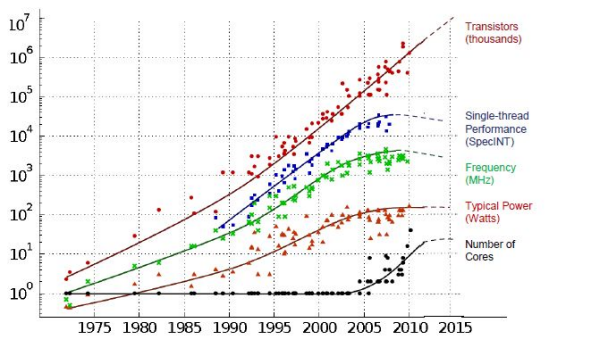


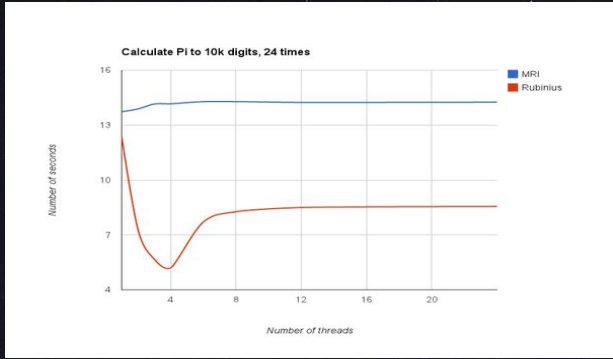
1. Async Debugging Hell

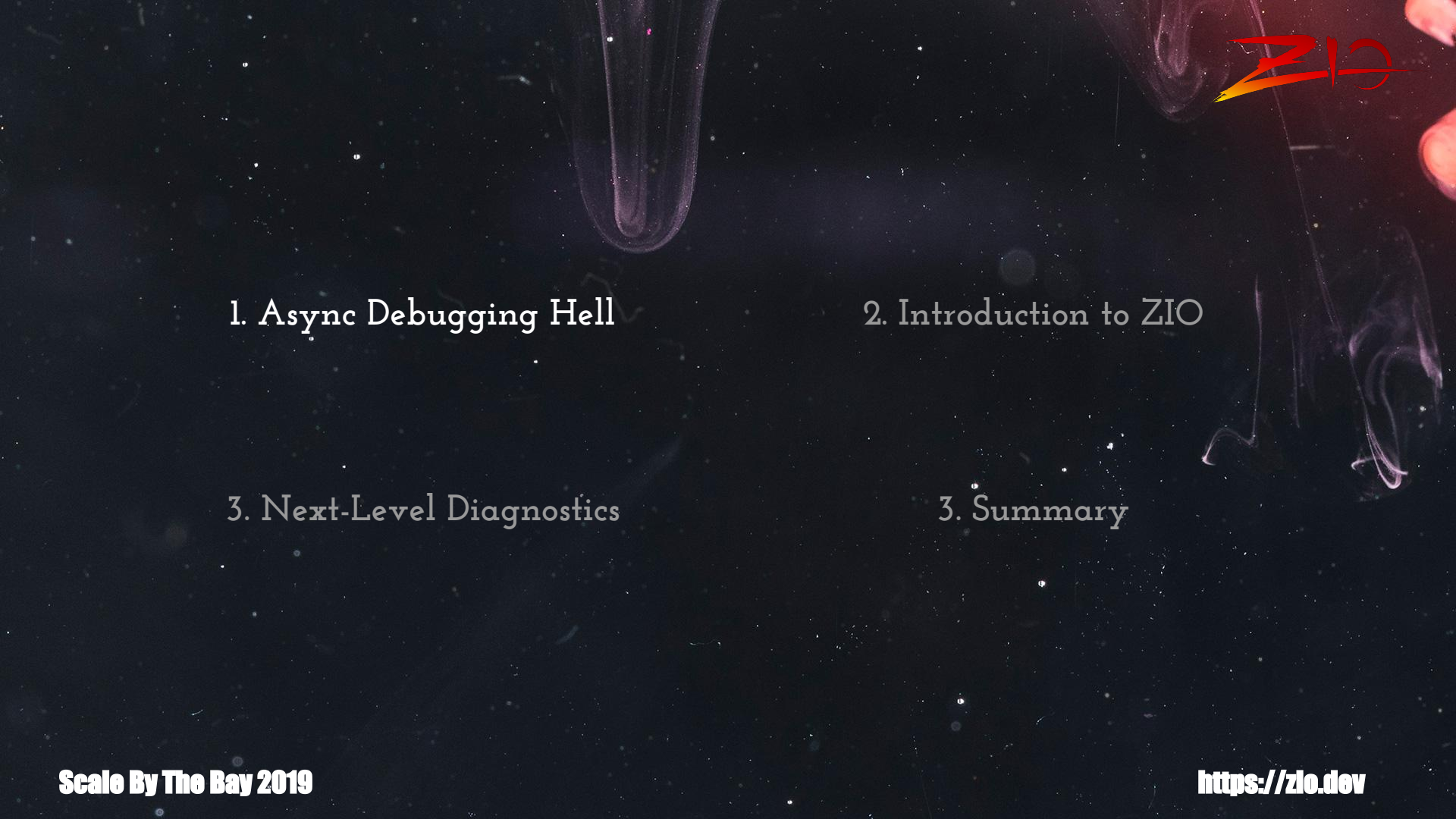
2. Introduction to ZIO

3. Next-Level Diagnostics

3. Summary


10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
1975 1980 1985 1990 1995 2000 2005 2010 2015
Transistors (thousands)
Single-thread Performance (SpecINT)
Frequency (MHz)
Typical Power (Watts)
Number of Cores


Calculate Pi to 10k digits, 24 times
MRI
Rubinius
Number of seconds
16
13
10
7
4
4
8
12
16
20
Number of threads


1. Async Debugging Hell
2. Introduction to ZIO
3. Next-Level Diagnostics
3. Summary

Async Debugging Hell

```
PostgresException: Syntax error at or near 42
    at example$.getConnection(example.scala:43)
    at example$.$anonfun$asyncDbCall$1(example.scala:23)
    at scala.concurrent.Future$.$anonfun$apply$1(Future.scala:658)
    at scala.util.Success.$anonfun$map$1(Try.scala:255)
    at scala.util.Success.map(Try.scala:213)
    at scala.concurrent.Future.$anonfun$map$1(Future.scala:292)
    at scala.concurrent.impl.Promise.liftedTree1$1(Promise.scala:33)
    at scala.concurrent.impl.Promise.$anonfun$transform$1(Promise.scala:33)
    at scala.concurrent.impl.CallbackRunnable.run(Promise.scala:64)
    at
java.util.concurrent.ForkJoinTask$RunnableExecuteAction.exec(ForkJoinTask.java:1402)
    at java.util.concurrent.ForkJoinTask.doExec(ForkJoinTask.java:289)
    at java.util.concurrent.ForkJoinPool$WorkQueue.runTask(ForkJoinPool.java:1056)
    at java.util.concurrent.ForkJoinPool.runWorker(ForkJoinPool.java:1692)
    at java.util.concurrent.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:157)
```

— Stack Traces

Async Debugging Hell

```
Full thread dump Java HotSpot(TM) 64-Bit Server VM (10.0.1+10 mixed mode):
Threads class SMR info:
_java_thread_list=0x00000250e5488a00, length=13, elements={
0x00000250e4979000, 0x00000250e4982800, 0x00000250e52f2800, 0x00000250e4992800,
0x00000250e4995800, 0x00000250e49a5800, 0x00000250e49ae800, 0x00000250e5324000,
0x00000250e54cd800, 0x00000250e54cf000, 0x00000250e54d1800, 0x00000250e54d2000,
0x00000250e54d0800
}
"Reference Handler" #2 daemon prio=10 os_prio=2 tid=0x00000250e4979000 nid=0x3c28 waiting on
condition  [0x000000b82a9ff000]
  java.lang.Thread.State: RUNNABLE
   at java.lang.ref.Reference.waitForReferencePendingList(java.base@10.0.1/Native Method)
   at java.lang.ref.Reference.processPendingReferences(java.base@10.0.1/Reference.java:174)
   at java.lang.ref.Reference.access$000(java.base@10.0.1/Reference.java:44)
   at java.lang.ref.Reference$ReferenceHandler.run(java.base@10.0.1/Reference.java:138)
  Locked ownable synchronizers:
   - None
```

— Thread Dumps

Async Debugging Hell

```
"Thread-0" #12 prio=5 os_prio=0 tid=0x00000250e54d1800 nid=0xdec waiting for monitor entry  [0x000000
   java.lang.Thread.State: BLOCKED (on object monitor)
    at DeadlockProgram$DeadlockRunnable.run(DeadlockProgram.java:34)
    - waiting to lock <0x00000000894465b0> (a java.lang.Object)
    - locked <0x00000000894465a0> (a java.lang.Object)
    at java.lang.Thread.run(java.base@10.0.1/Thread.java:844)

   Locked ownable synchronizers:
    - None

"Thread-1" #13 prio=5 os_prio=0 tid=0x00000250e54d2000 nid=0x415c waiting for monitor entry  [0x00000
   java.lang.Thread.State: BLOCKED (on object monitor)
    at DeadlockProgram$DeadlockRunnable.run(DeadlockProgram.java:34)
    - waiting to lock <0x00000000894465a0> (a java.lang.Object)
    - locked <0x00000000894465b0> (a java.lang.Object)
    at java.lang.Thread.run(java.base@10.0.1/Thread.java:844)

   Locked ownable synchronizers:
    - None
```

Hanging Causes

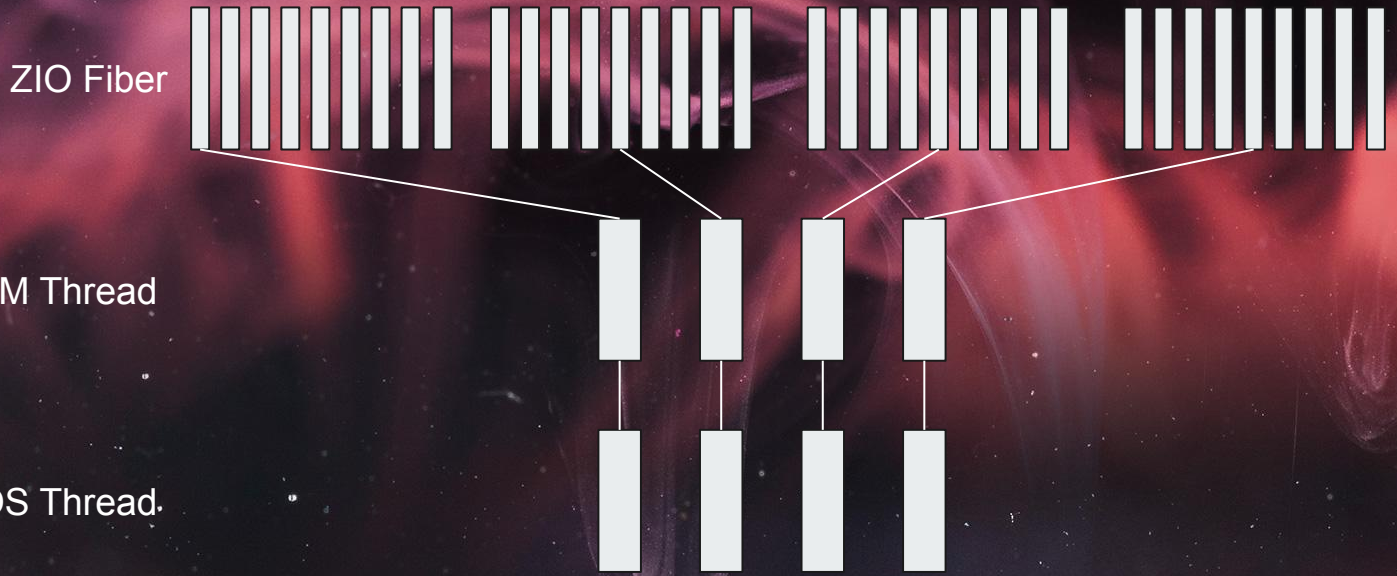
1. Async Debugging Hell

2. Introduction to ZIO

3. Next-Level Diagnostics

3. Summary

Introduction to ZIO



Introduction to ZIO

	Future	ZIO
No Implicits	✗	✓
Precise Parallelsim	✗	✓
Composable Efficiency	✗	✓
Execution Traces	✗	✓
Refactoring	✗	✓
Astractable	✗	✓
Statically Typed Errors	✗	✓
Resource Safety	✗	✓
Batteries Included	✗	✓
Performant	✗	✓
Testable	✗	✓

Introduction to ZIO

```scala
import zio._
import zio.console._

object MyApp extends App {
  def run(args: List[String]) =
    putStrLn("Hello World!")
      .fold(_ => 1, _ => 0)
}
```

Introduction to ZIO

```scala
import zio._
import zio.console._

object MyApp extends App {
  def run(args: List[String]) =
    (for {
      _    <- putStrLn("What is your name?")
      name <- getStrLn
      _    <- putStrLn(s"Hello, ${name}!")
    } yield 0) orElse ZIO.succeed(1)
}
```

Introduction to ZIO

```
import zio._

def par[A, B](l: Task[A], r: Task[B]): Task[(A, B)] =
  (for {
    fiberL <- l.fork
    fiberR <- r.fork
    a      <- fiberL.join
    b      <- fiberR.join
  } yield (a, b)).interruptChildren
```

Introduction to ZIO

```
import zio._
import zio.duration._
import ZSchedule._

def httpGet(url: URL): Task[Response] = …

val retried: Task[Response] =
  httpGet(myUrl).retry {
    (exponential(10.millis) ||
     fixed(60.seconds)) && recurs(100)
  }
```

Introduction to ZIO

```
import zio._

def httpGet(url: URL): Task[Response] = …

def loadTest(url: URL): Task[Unit] =
  ZIO.foreachPar(1 to 1000) { _ =>
    httpGet(url)
  }.unit
```

1. Async Debugging Hell

2. Introduction to ZIO

3. Next-Level Diagnostics

3. Summary

Next-Level Diagnostics

```
def run(args: List[String]) =
  (for {
    _       <- putStrLn("What is your name?")
    name    <- getStrLn
    seconds <- (putStrLn(s"How many seconds do you want to wait, ${name}?") *>
                getStrLn.flatMap(input => Task(input.toDouble))).eventually
    _       <- ZIO.sleep((seconds * 1000).toLong.millis)
    _       <- putStrLn(s"Time to wake up, ${name}!")
  } yield 0) orElse ZIO.succeed(1)
```

Testable

Next-Level Diagnostics

```
testM("Prompt") {
  for {
    _      <- TestConsole.feedLines("John", "10")
    fiber <- PromptName.run(Nil).fork
    _      <- TestClock.adjust(10.seconds)
    _      <- fiber.join
    out    <- TestConsole.output.map(_.last)
  } yield assert(out, equalTo("Time to wake up, John!\n"))
},
```

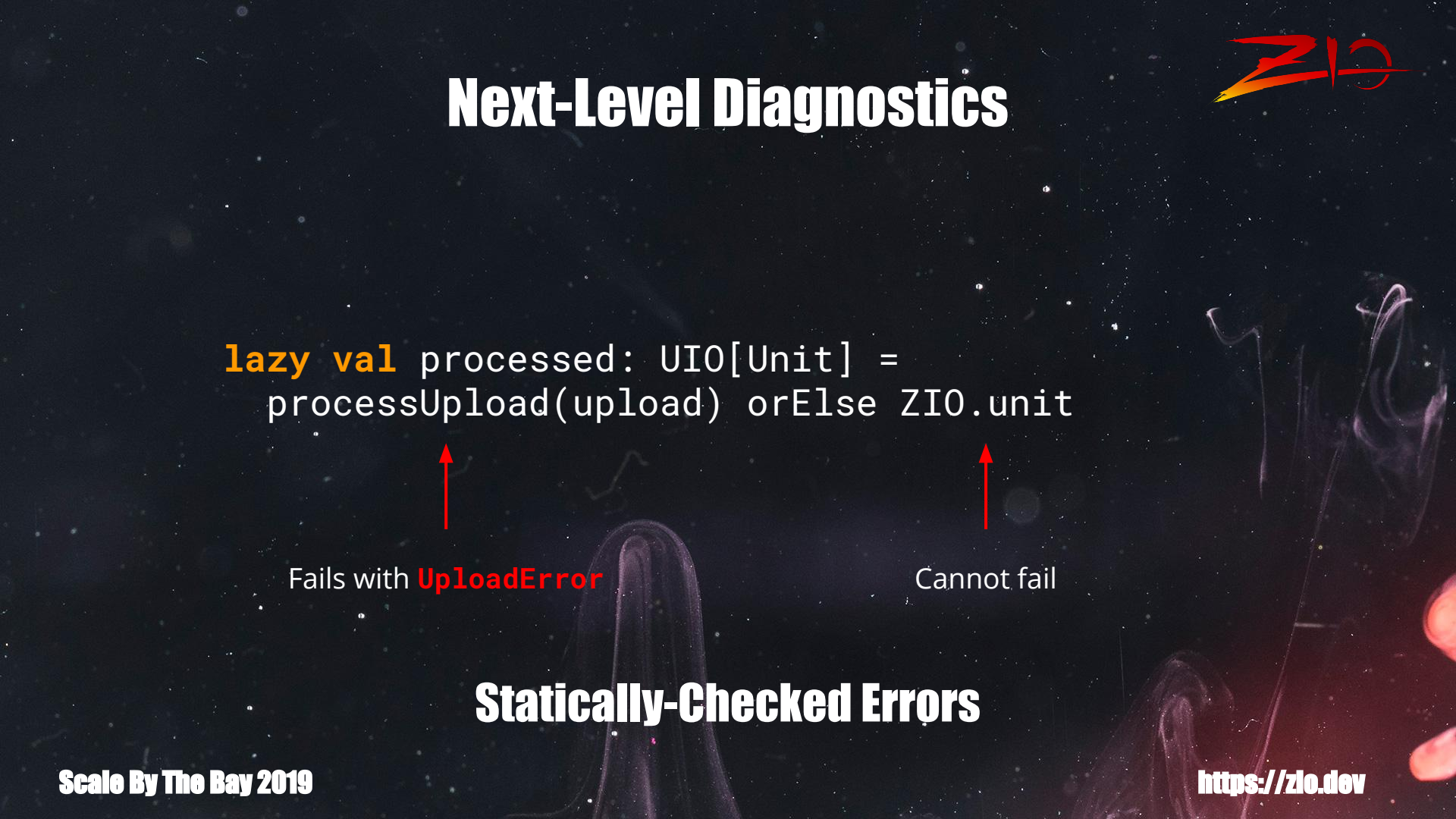
Testable

Next-Level Diagnostics

```
type UIO[+A] = ZIO[Any, Nothing, A]
```

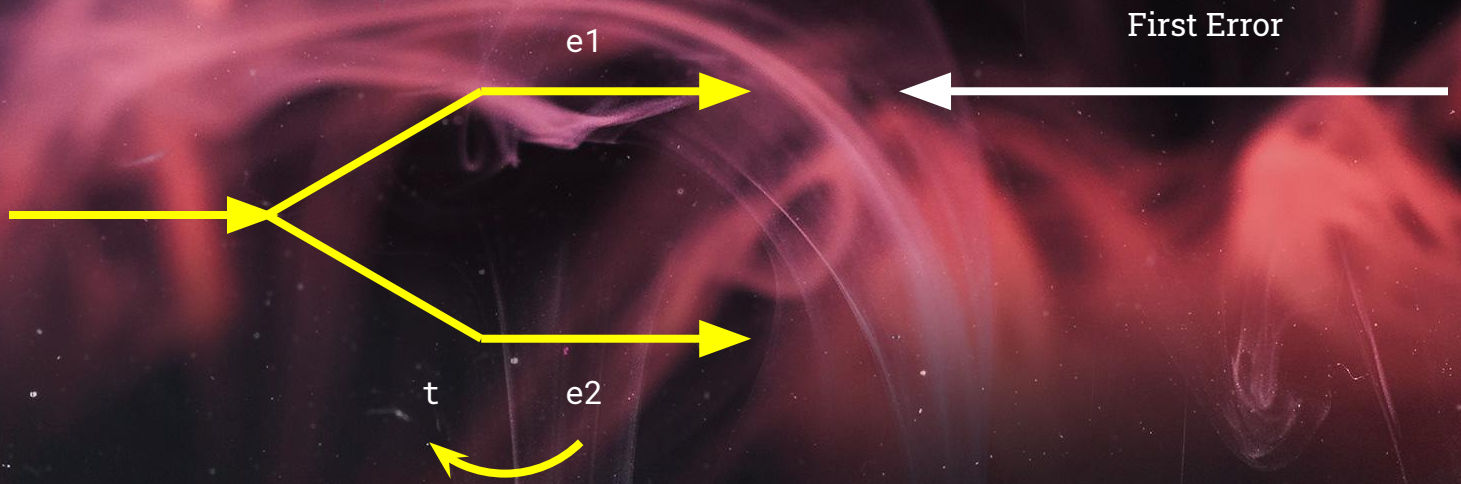
Statically-Checked Errors

Next-Level Diagnostics



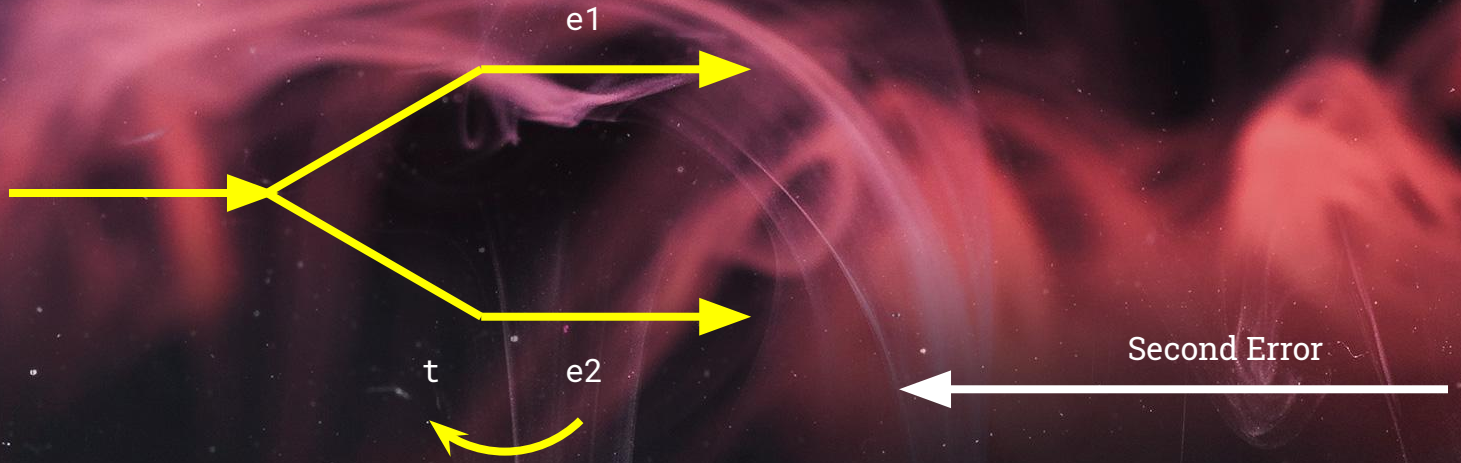
Statically-Checked Errors

Next-Level Diagnostics

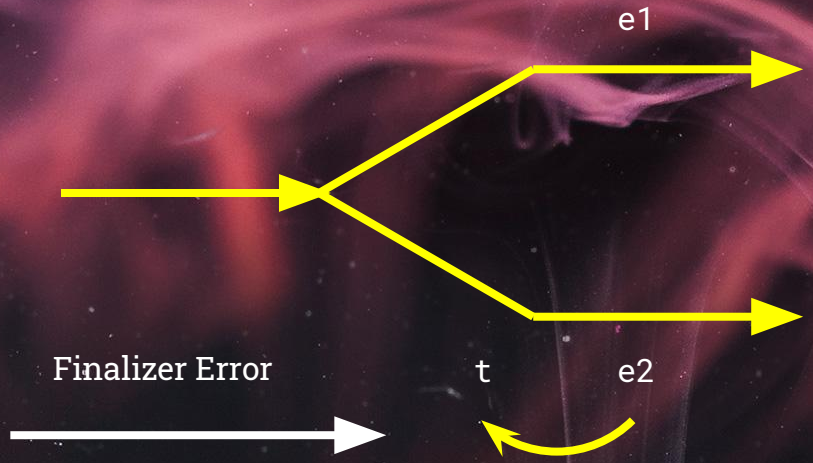


Lossless Errors

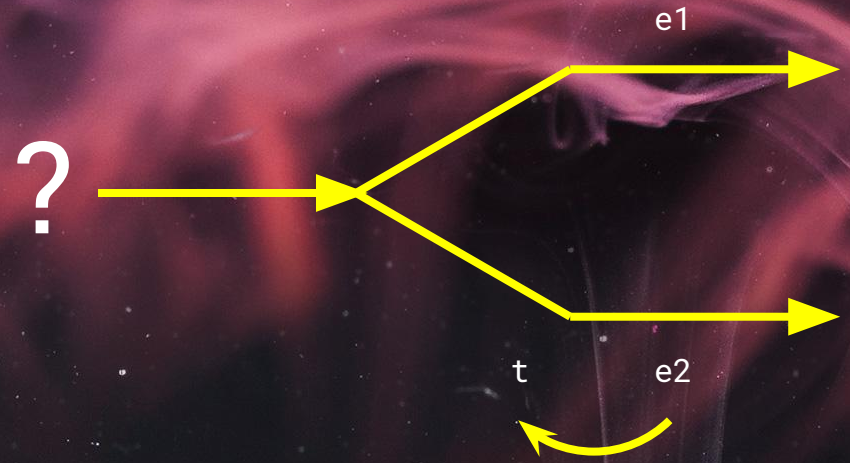
Next-Level Diagnostics



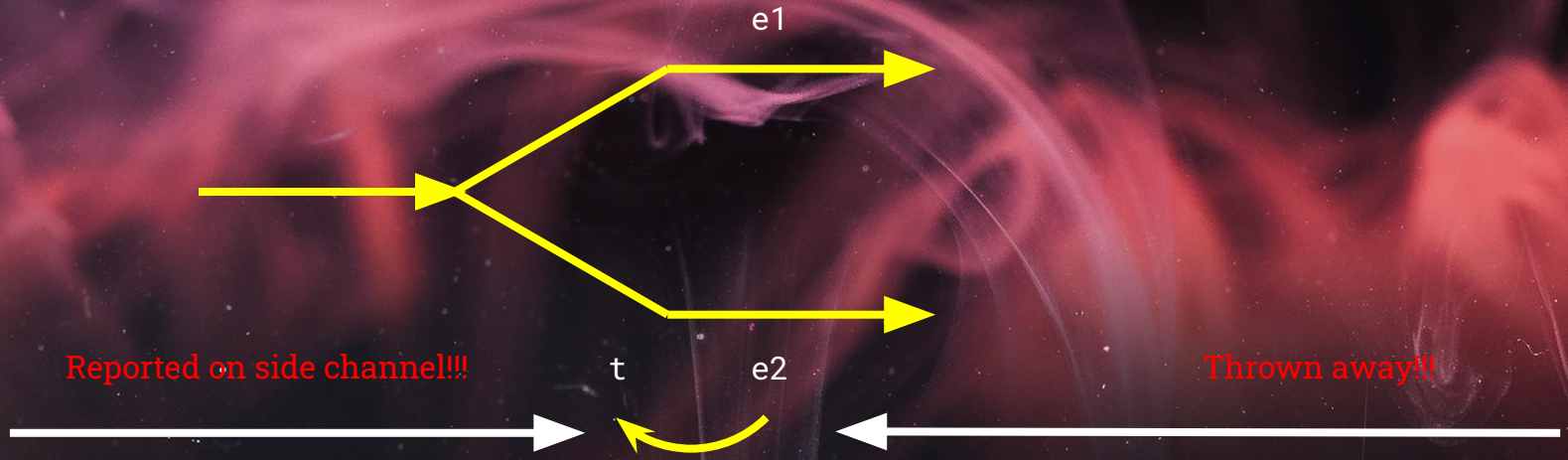
Next-Level Diagnostics



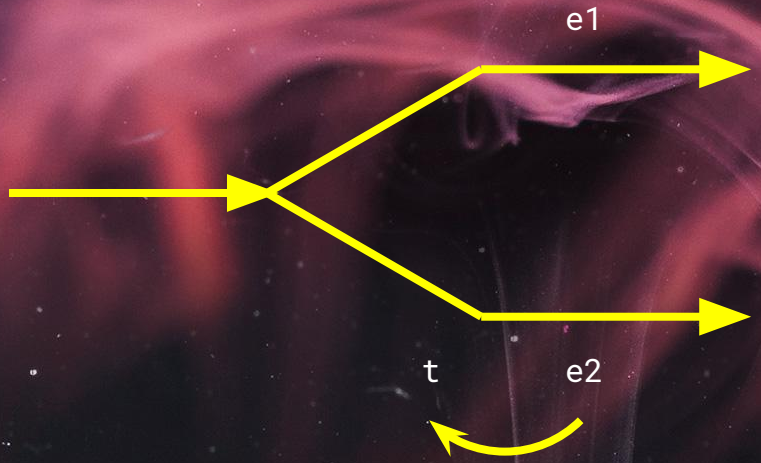
Next-Level Diagnostics



Next-Level Diagnostics



Next-Level Diagnostics



Cause[E]

```
Cause.Both(
Cause.Fail(e1),
  Cause.Then(
Cause.Fail(e2),
 Cause.Die(t))
```

Next-Level Diagnostics

```
zio.FiberFailure: Fiber failed.
╥
╠═╗
║ ╠═ A checked error was not handled:
║ ╚═ Failed(DatabaseUnreachableError)
╠═╗
║ ╠═ A finalizer threw an error:
║ ▼  Die(IndexOutOfBoundsException())
╠═╗
║ ╠═ A checked error was not handled:
▼ ▼  Failed(UserIdNotFoundError)
```

e2

t

e1

Next-Level Diagnostics

```
def myQuery =
  UIO(println("Querying!"))
    .flatMap(_ =>
      queryDatabase
        .map(res => res))
```

The Past

The Future

```
Fiber:0 ZIO Execution trace:
  at myQuery(example.scala:4)
  at myQuery(example.scala:3)

Fiber:0 was supposed to continue to:
  a future continuation at myQuery(example.scala:5)
```

ZIO Execution Traces

Next-Level Diagnostics

```
#17 (41ms) waiting on #14
  Status: Suspended(interruptible, 0 asyncs, zio.Promise.await(Promise.scala:49))
  Fiber:Id(1573702379054,17) was supposed to continue to:
    a future continuation at zio.ZIO$._IdentityFn(ZIO.scala:2630)
    a future continuation at zio.ZIO.ensuring(ZIO.scala:350)

  Fiber:Id(1573702379054,17) execution trace:
    at zio.Promise.await(Promise.scala:49)
    at zio.ZIOFunctions.effectAsyncInterrupt(ZIO.scala:1952)
    at zio.ZIOFunctions.effectAsyncInterrupt(ZIO.scala:1952)
```

Fiber Dumps

Next-Level Diagnostics

```
#17 (41ms) waiting on #14
   Status: Suspended(interruptible, 0 asyncs, zio.Promise.await(Promise.scala:49))
   Fiber:Id(1573702379054,17) was supposed to continue to:
     a future continuation at zio.ZIO$._IdentityFn(ZIO.scala:2630)
     a future continuation at zio.ZIO.ensuring(ZIO.scala:350)

   Fiber:Id(1573702379054,17) execution trace:
     at zio.Promise.await(Promise.scala:49)
     at zio.ZIOFunctions.effectAsyncInterrupt(ZIO.scala:1952)
     at zio.ZIOFunctions.effectAsyncInterrupt(ZIO.scala:1952)
```

Fiber Dumps

Next-Level Diagnostics

```
#17 (41ms) waiting on #14
  Status: Suspended(interruptible, 0 asyncs, zio.Promise.await(Promise.scala:49))
  Fiber:Id(157370237905 4,17) was supposed to continue to:
    a future continuation at zio.ZIO$._IdentityFn(ZIO.scala:2630)
    a future continuation at zio.ZIO.ensuring(ZIO.scala:350)

  Fiber:Id(157370237905 4,17) execution trace:
    at zio.Promise.await(Promise.scala:49)
    at zio.ZIOFunctions.effectAsyncInterrupt(ZIO.scala:1952)
    at zio.ZIOFunctions.effectAsyncInterrupt(ZIO.scala:1952)
```

Fiber Dumps

Next-Level Diagnostics

```
#17 (41ms) waiting on #14
  Status: Suspended(interruptible, 0 asyncs, zio.Promise.await(Promise.scala:49))
  Fiber:Id(1573702379054,17) was supposed to continue to:
    a future continuation at zio.ZIO$._IdentityFn(ZIO.scala:2630)
    a future continuation at zio.ZIO.ensuring(ZIO.scala:350)

  Fiber:Id(1573702379054,17) execution trace:
    at zio.Promise.await(Promise.scala:49)
    at zio.ZIOFunctions.effectAsyncInterrupt(ZIO.scala:1952)
    at zio.ZIOFunctions.effectAsyncInterrupt(ZIO.scala:1952)
```

Fiber Dumps

Next-Level Diagnostics

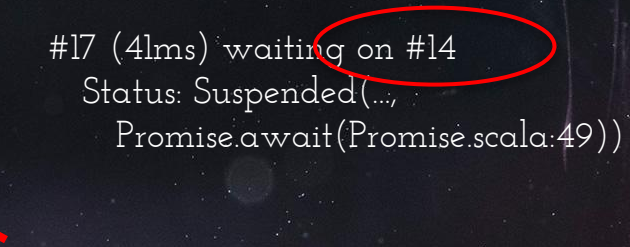
```
#17 (41ms) waiting on #14
   Status: Suspended(interruptible, 0 asyncs, zio.Promise.await(Promise.scala:49))
   Fiber:Id(1573702379054,17) was supposed to continue to:
     a future continuation at zio.ZIO$._IdentityFn(ZIO.scala:2630)
     a future continuation at zio.ZIO.ensuring(ZIO.scala:350)

   Fiber:Id(1573702379054,17) execution trace:
     at zio.Promise.await(Promise.scala:49)
     at zio.ZIOFunctions.effectAsyncInterrupt(ZIO.scala:1952)
     at zio.ZIOFunctions.effectAsyncInterrupt(ZIO.scala:1952)
```

Fiber Dumps

Next-Level Diagnostics

```
for {
  promise <- Promise.make[Nothing, Int]
  _       <- handOff(promise)
  fiber   <- promise.await.fork
  ...
} yield ()
```



Hanging Causes

Next-Level Diagnostics

```
for {
  queue    <- Queue.bounded[Int](100)
  producer <- queue.offer(42).forever.fork
  rez      <- producer.interrupt
  ...
} yield ()
```

Fiber failed.
An interrupt was
produced by #29.

Cancellation Causes

1. Async Debugging Hell

2. Introduction to ZIO

3. Next-Level Diagnostics

3. Summary

Thank You !

Special Thanks: Alexy Khrabrov
ZIO Hackathon San Francisco - Early 2020
Learn More: https://zio.dev
Chat with us on Discord: https://discord.gg/2ccFBr4

Follow John A De Goes:

- Twitter: @jdegoes
- Support John's work: https://www.patreon.com/jdegoes
- Blog: http://degoes.net

Follow Salar Rahmanian:

- Twitter: @SalarRahmanian
- Blog: https://www.softinio.com