Juan Reséndiz

Email: jresendiz@nearsoft.com
Twitter: @jresendiz27
Github: jresendiz27

Índice

- What’s Vert.x?
- A brief history about it.
- Why async?
  - Reactor Pattern vs Multi Reactor Pattern.
- Vert.x Core
  - Polyglot
  - Verticles
  - Event bus
    - Clustered event bus
- Get your hands dirty!
- Q&A
- Resources

What's Vert.x?

Es una herramienta que te permite construir aplicaciones reactivas en la JVM. Es orientado a eventos (event driven) y no bloqueante (non-blocking), lo cual significa que puede manejar mucha concurrencia usando un número limitado de hilos.

Es políglota (polyglot) es decir, puedes desarrollar en Java, Javascript, Groovy, Ruby y Ceylon; esto gracias a su API idiomática.

What's Vert.x?

Vert.x es ligero.

Vert.x es rápido.

Vert.x no es un servidor de aplicaciones.

Vert.x es ideal para microservicios.

A brief story about it.

Su desarrollo comenzó en 2011 por Tim Fox mientras aún era empleado en VMware. El proyecto se iba a llamar Node.x, haciendo referencia a que es políglota y no solo usa Javascript, pero se evitaron problemas legales y se pasó a su nombre actual, Vert.x.

Toma varios conceptos de Node e implementa ideas de Akka (modelo de actores) y corre sobre Netty y Hazelcast.

Ganó el premio JAX en 2014 como la tecnología más innovadora de Java.

Why async?

- C10K problem.
- Altos niveles de concurrencia.
- Los hilos del sistema operativo son valiosos.
- Reducido número de hilos >> Muchas conexiones.
- IoT.

587

Synchronous (one thread):

```
1 thread ->   |----A-----||-----B-----------||-------C------|
```

Synchronous (multi-threaded):

```
thread A -> |----A-----|
                      \
thread B ------------>   ->|-----B-----------|
                                             \
thread C ----------------------------------->   ->|-------C------|
```

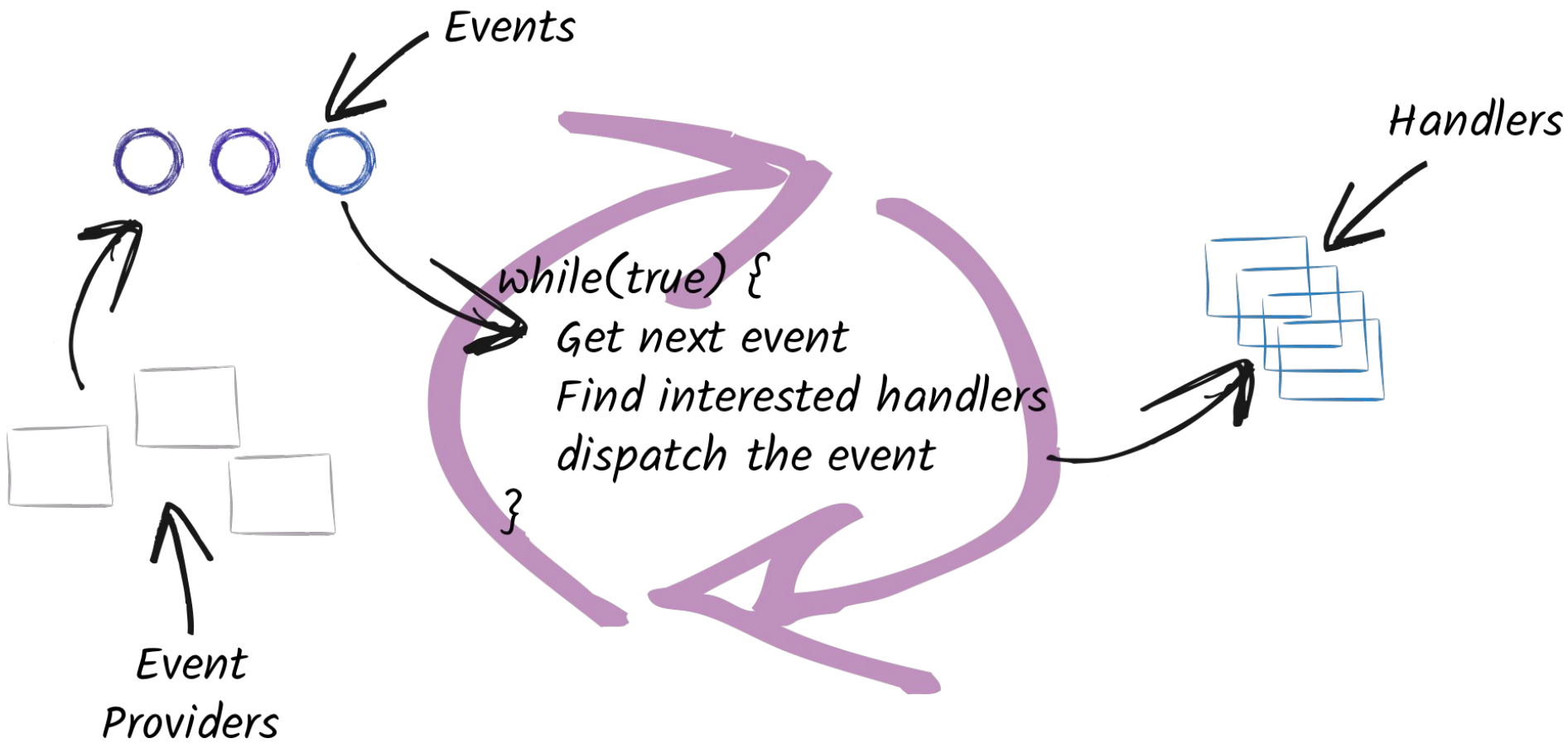
Asynchronous (one thread):

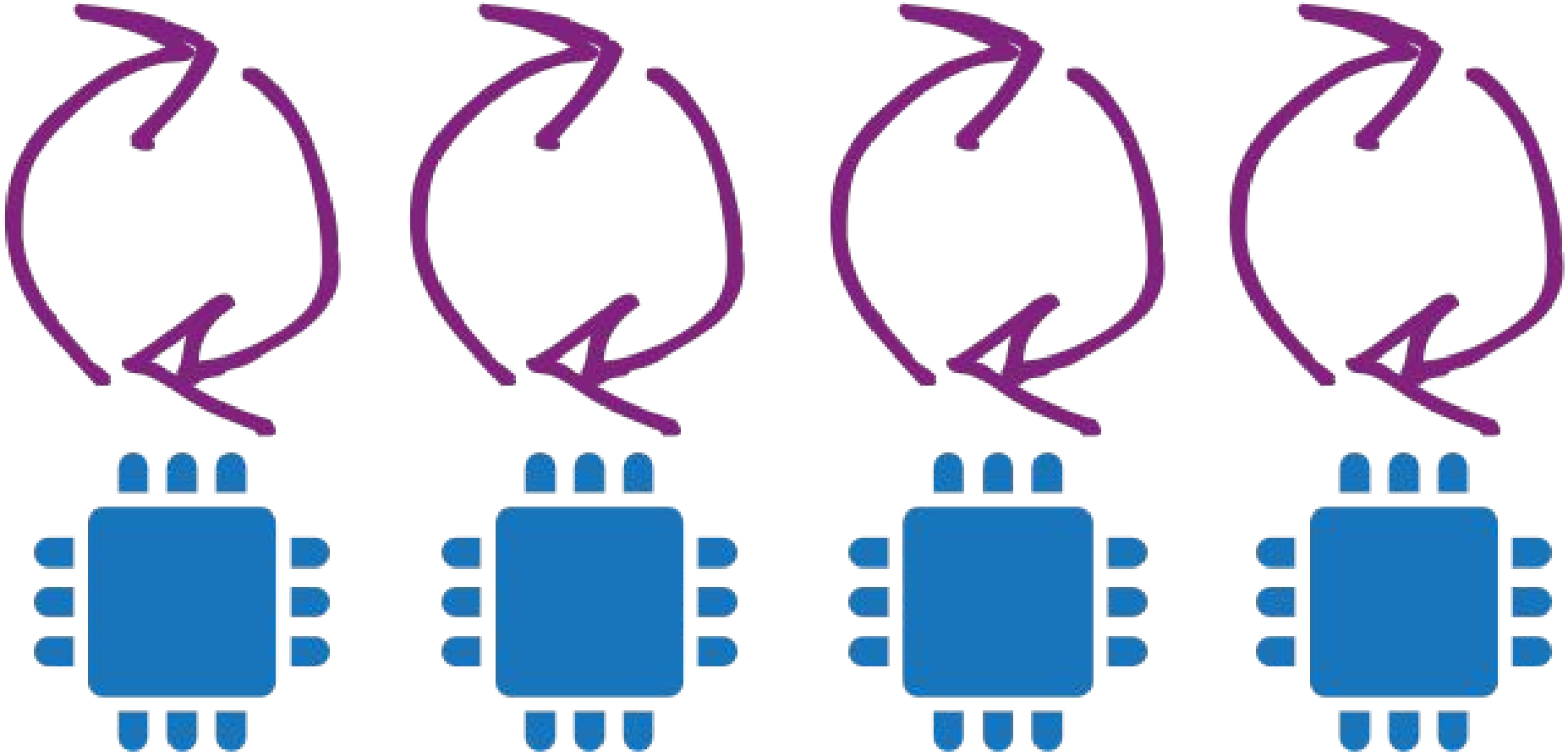
```
         A-Start ---------------------------------------- A-End
           | B-Start ------------------------------------------|--- B-End
           |   |     C-Start -------------------- C-End       |     |
           V   V       V                          V           V     V
1 thread-> |-A-|---B---|-C-|-A-|-C-|--A--|-B-|--C--|---A-----|--B--|
```

Asynchronous (multi-Threaded):

```
 thread A ->     |----A-----|
 thread B ----->     |-----B-----------|
 thread C --------->     |-------C----------|
```

Reactor Pattern vs Multi-Reactor Pattern


Events
Handlers
while(true) {
Get next event
Find interested handlers
dispatch the event
}
Event
Providers



Reactor Pattern vs Multi-Reactor

Never block the event loop!

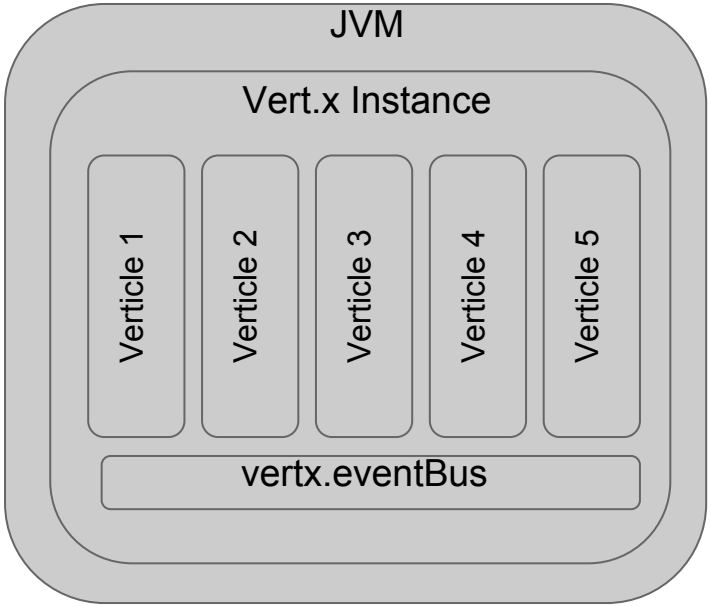
Never block the event loop!

Never block the event loop!

Vert.x Core: Polyglot

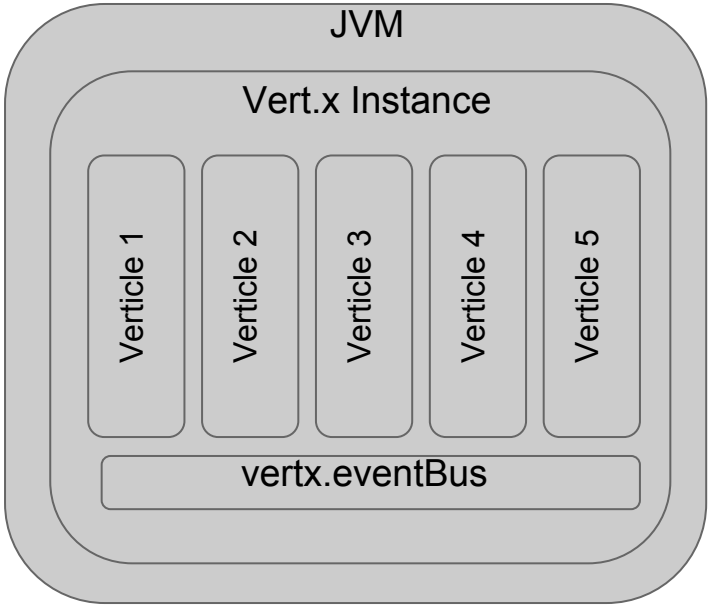
Vert.x Core: Verticle

- Unidad mínima de programación en Vert.x.
- Ejecutada en el mismo hilo.
- Un mismo hilo puede ejecutar varias Verticles. (event loop)
- Ejecuta un hilo/event loop por núcleo de procesador.



Vert.x Core: Verticle

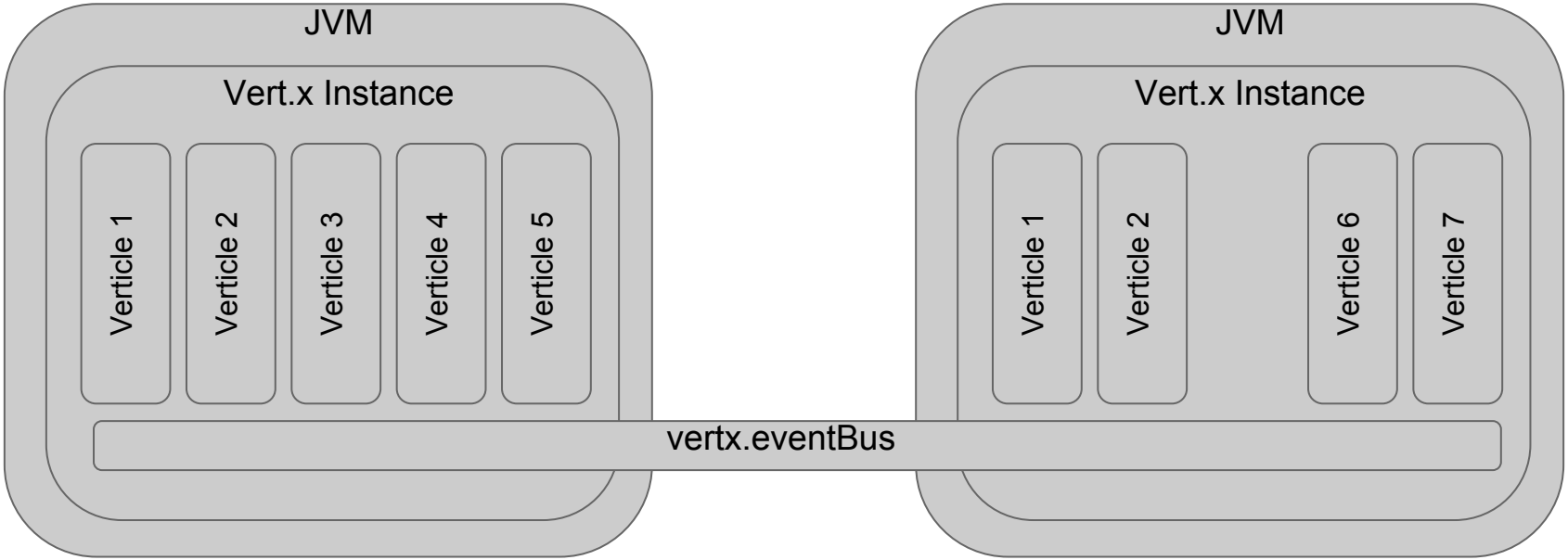
- Modelo de actores.
- Diferentes lenguajes.
- Scripts, classes, containers.
- Tipos de verticles
  - Standard
  - Worker
  - Multi-threaded



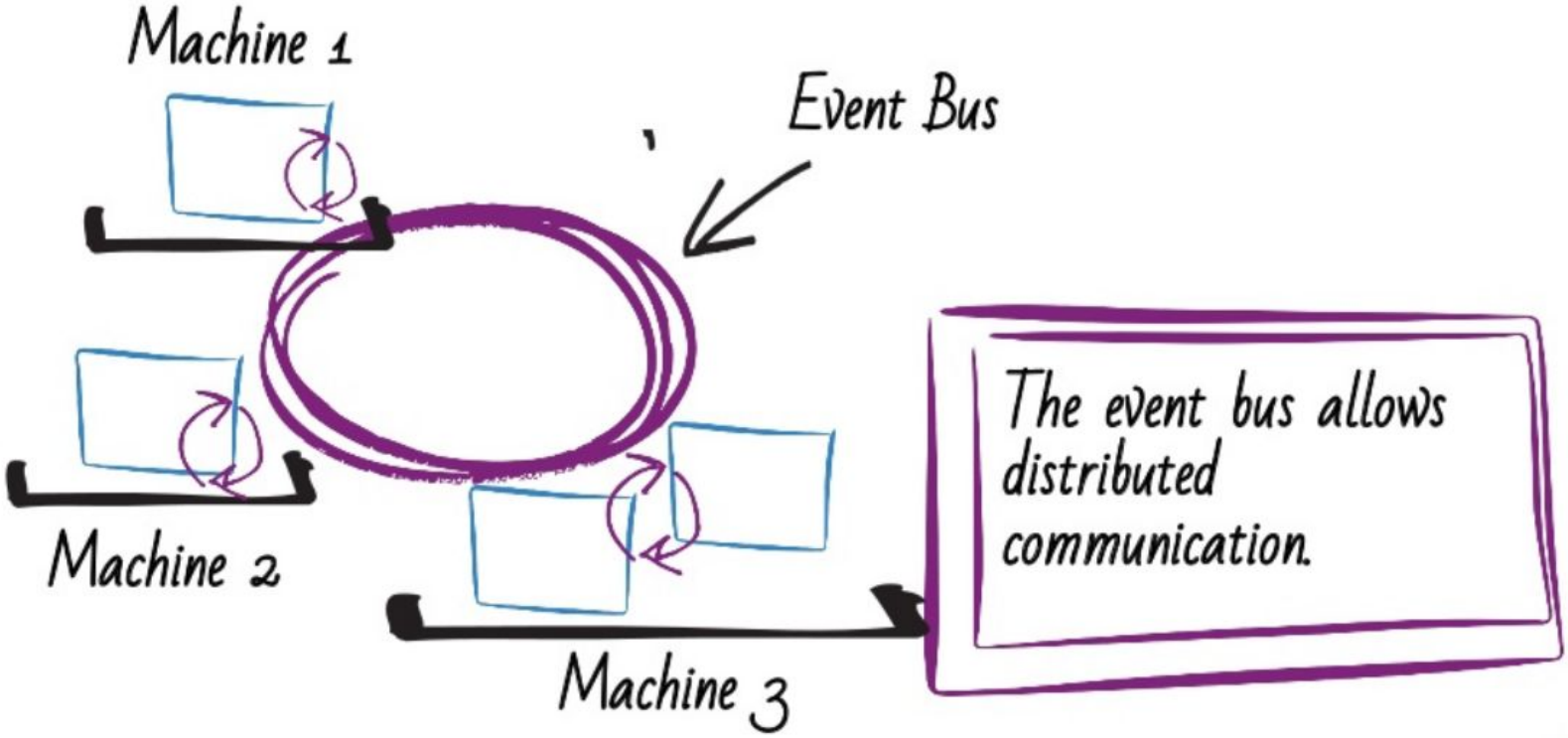
Vert.x Core: Event bus

- Sistema nervioso de Vert.x.
- Permite la comunicación entre Verticles.
- Puede crear un puente (bridge) entre un navegador e instancias en un servidor.
- Distribuido.
- Tipos de mensajes
  - Publish/Subscribe
  - Point to Point
  - Request/Response
- Handlers

Vert.x Core: Event bus



Vert.x Core: Event bus




We Can Do It!
LET'S GET TO WORK

Resources.

http://vertx.io/materials/

https://www.techempower.com/benchmarks/#section=data-r8&hw=i7&test=plaintext

http://stackoverflow.com/questions/748175/asynchronous-vs-synchronous-execution-what-does-it-really-mean

https://docs.google.com/presentation/d/18wPraTym-rJyyXWcSuq493-bzvxD3Qij9Hhl8r543eU/edit?usp=sharing

https://github.com/vert-x3/vertx-awesome

http://es.slideshare.net/clement.escoffier/vertx-31-be-reactive-on-the-jvm-but-not-only-in-java

http://vertx.io/docs/vertx-core/java/