vert.x

Effortless asynchronous application development for the modern web and enterprise

Stuart Williams

'Pid'

Consulting Architect
SpringSource Division of VMware*

vert.x committer

@pidster

** Correct at time of writing*

What is vert.x?

- Polyglot
- Simple
- Scalable
- Asynchronous
- Concurrent

Polyglot

Write application components in:

- JavaScript + CoffeeScript + AngularJS
- Ruby
- Python
- Groovy
- Java

Mix and match several programming languages in a single app.

Simple

- ...without being simplistic
- Create real applications in just a few lines of code
- No sprawling XML config

Scalable

- Linear horizontal scale
- Uses message passing
- Automatic load-balancing
- Efficiently utilise your server cores

Asynchronous

- NIO
- Handlers
- EventBus
- FileSystem
- WebSockets
- SockJS

SockJS

- Handles the communication between the browser and the server.
- Provides a websocket-like API in client-side JS
- Works when websockets not available
- JSON-Polling, XHR-Polling/Streaming, etc
- Similar in some ways to socket.io
- More info at http://sockjs.org

Concurrency Model

- A verticle instance is single-threaded.
- Move away from 'Java-style' multi-threaded concurrency
- No more *synchronized*, *volatile* or *locking*
- Wave goodbye to many race conditions

Threading Model – Event Loops

- vert.x implements the *Multi-Reactor Pattern*
- An event loop is an OS thread
- Handles events for many handlers
- vert.x has *multiple* event loops – typically one per core.
- Don't block the event loop!

Event Loop != Silver Bullet

- Event loops not the best fit for all types of stuff
- Long running calculations (Fibonacci anyone?)
- Using blocking APIs – e.g. JDBC
- Most traditional Java libs have blocking APIs
- … how do we leverage them?

Hybrid Threading Model

- Don't force everything to run on an event loop
- Worker verticles can block
- Communicate with other verticles by message passing.
- Allows us to leverage the huge ecosystem of blocking Java libs

Real-time Web Applications

- Modern definition of *real-time*
- Event bus key technology for *real-time* web applications
- Focus on sending messages to server components or other browser nodes
- Mobile applications, online games
- Use vert.x with any JS toolkit.

Examples

JavaScript

```
load('vertx.js')

vertx.createHttpServer().requestHandler(function(req) {
    var file = req.path === '/' ? 'index.html' : req.path;
    req.response.sendFile('webroot/' + file);
}).listen(8080)
```

Ruby

```
require "vertx"

Vertx::HttpServer.new.request_handler do |req|
    file = req.uri == "/" ? "index.html" : req.uri
    req.response.send_file "webroot/#{file}"
end.listen(8080)
```

Python

```
import vertx

server = vertx.create_http_server()

@server.request_handler
def request_handler(req):
    file = "index.html" if req.uri == "/" else req.uri
    req.response.send_file("webroot/%s"%file)
server.listen(8080)
```

Groovy

```
vertx.createHttpServer().requestHandler { req ->
    def file = req.uri == "/" ? "index.html" : req.uri
    req.response.sendFile "webroot/$file"
}.listen(8080)
```

Java

```
import org.vertx.java.core.Handler;
import org.vertx.java.core.http.HttpServerRequest;
import org.vertx.java.deploy.Verticle;
public class Server extends Verticle {
    public void start() {
        vertx.createHttpServer().requestHandler(new Handler<HttpServerRequest>(){
            public void handle(HttpServerRequest req) {
                String file = req.path.equals("/") ? "index.html" : req.path;
                req.response.sendFile("webroot/" + file);
            }
        }).listen(8080);
    }
}
```

Architecture

Key Dependencies

- Java 7
- Netty
- Hazelcast
- Jackson
- JRuby
- Jython
- Rhino

Key API Methods

vertx.createXXXServer

vertx.createXXXClient

vertx.fileSystem

vertx.sharedData

vertx.eventBus

vertx.setPeriodic

vertx.setTimer

Servers

- NetServer
- HttpServer
  - RouteMatcher
  - ServerWebSocket
- SockJSServer

Clients

- NetClient
- HttpClient
  - WebSocket
- SockJSSocket

EventBus

```
eventBus.send('address', message, replyHandler)
eventBus.publish('address', message)
```

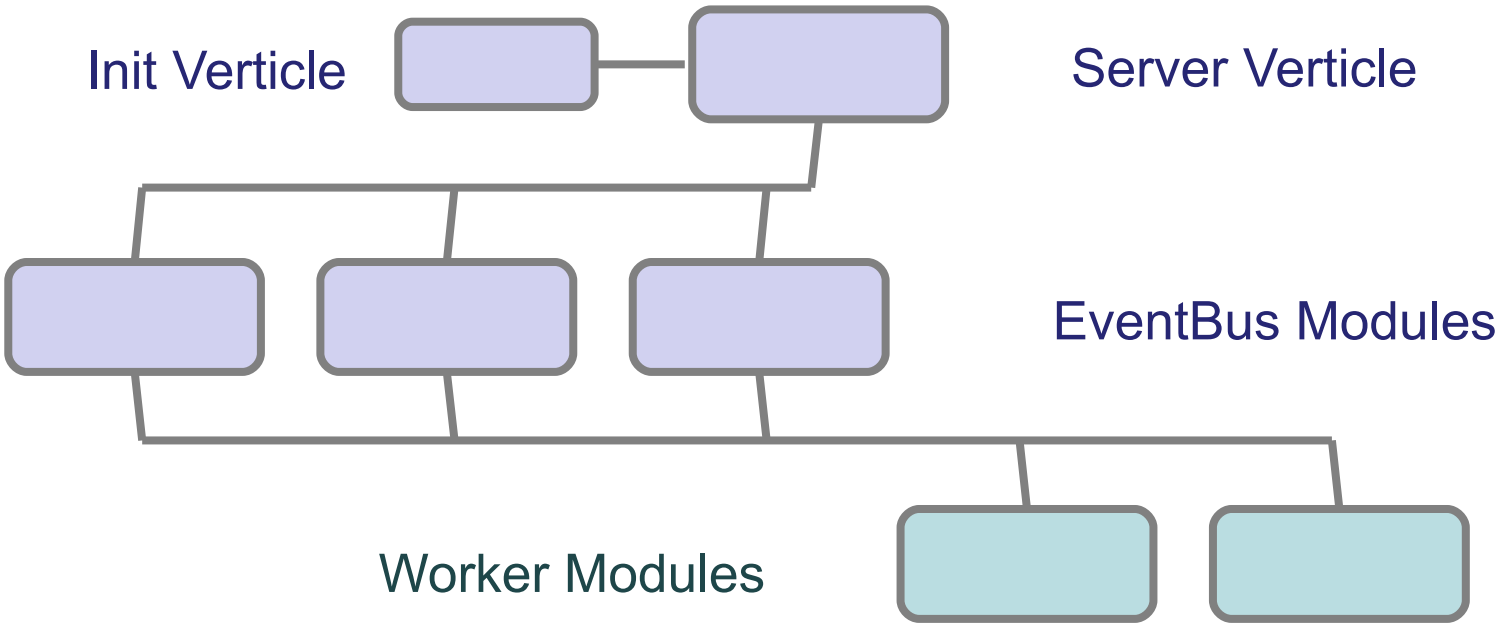
- EventBus Bridge – send messages direct to JavaScript in web browsers
- Messages are strongly typed in Java
- JSON in JavaScript
- Hash in Python, Ruby

Cluster

- Hazelcast manages event bus address map and cluster membership
- Explain cache, listeners
- vert.x NetServer & NetClient used for comms
- Fully configurable network ports
- Use SSL for security

vert.x Applications

Application Components



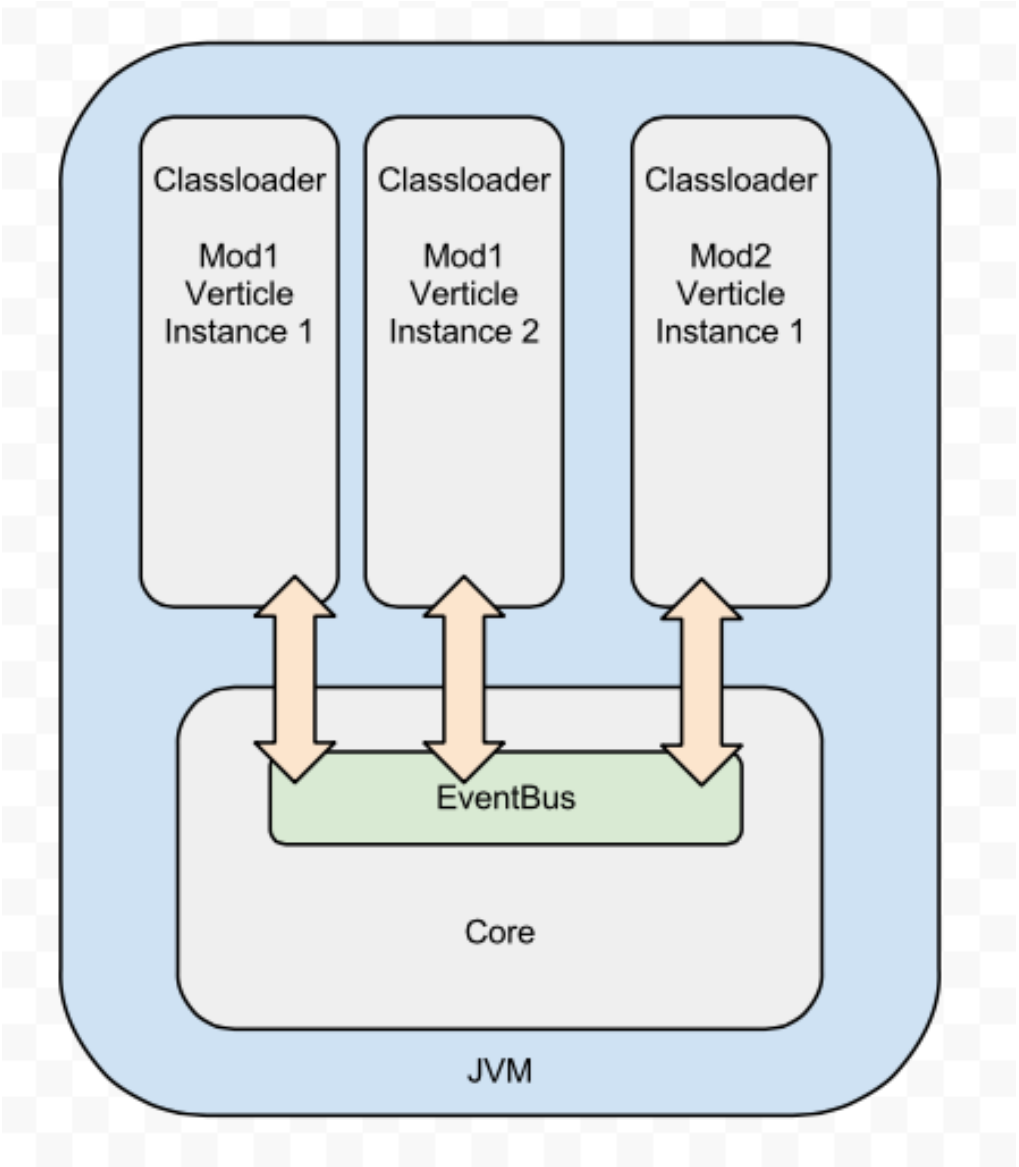
8

Event Loop

20

Worker Pool

Classloaders & Messages



launcher.js

```
load('vertx.js')
var config = {
  "address": "org.pidster.foobar.control",
  ”port": 8081
}
vertx.deployModule('org.pidster.foobar-v1.0', config, 1, function(id) {
  // called when deployed
});
function vertxStop() {
  // stop & clean up
}
```

Modules

- Authentication Manager
- Form Upload
- JDBC Persistor
- Mailer
- Mongo Persistor
- Session Manager
- Web Server
- Work Queue
- AMQP

mod-web-server

```
load('vertx.js')

var config = {
    "web_root": <web_root>,
    "port", <port>,
    "ssl": <ssl>,
    "key_store_password": <key_store_password>,
    "key_store_path": <key_store_path>,
}

vertx.deployModule('vertx.web-server-v1.0', config, 1, function() {
  // deployed
});
```

Module

- Simple structure
- Optional lib dir
- Lazy installation
- Modules can include other modules
- mod.json

```
{
 “main”:”org.pidster.jdbc.Main”,
 “worker”: “true”,
 “includes: “org.pidster-foobar-v1.0”
}
```

Developing & Testing

- Gradle or Maven (or Ant!)
- Use a verticle script to launch tests
- vertx-junit-annotations

src/main/groovy
src/test/groovy
src/vertxInteg/groovy

Best Practice

- Verticle script to manage lifecycle
- Define configuration in JSON
- Compose from modules
- Never block the event loop!

What’s next?

Languages

- Languages as modules
- Lazy installation
- Scala
- Clojure

Scala

- Basic implementation complete
- TODO: examples and docs
- github.com/swilliams-vmw/vertx-lang-scala

Scala Web Server

```
vertx.createHttpServer
  .requestHandler({ req: HttpServerRequest =>
    val file : String = if (req.path == “/”) “/index.html” else req.uri
    req.response.sendFile("webroot/" + file)
}).listen(8080)
```

Management

- JMX API
- Publish metrics on EventBus in JSON
- Configurable sample time & filters
- GUI
- github.com/swilliams-vmw/mod-management

Developers & API

- IDE support
- Build config and plugins
- Better, easier testing
- More examples
- Promises API?
- Direct-style API using continuations?

Project

Community

- vert.x is Open Source
- Active community
- Contributions welcome
- Module repository is growing
- 8th most popular Java project on Github

Project

- http://vertx.io
- @timfox – project lead
- @pidster – project lurker
- Uses Gradle for build
- https://github.com/vert-x
- Google Group: vertx

Summary

- Polyglot – use the languages you want
- Simple concurrency – wave goodbye to most race conditions
- Leverage existing Java library ecosystem
- Powerful distributed event bus which spans client and server
- Flexible module system

Questions?