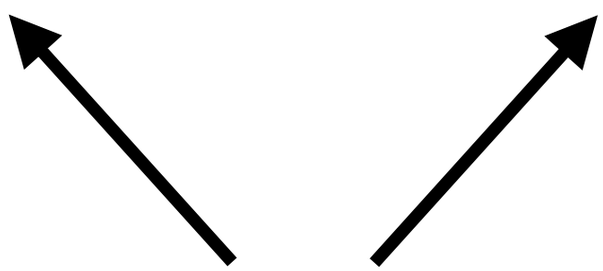
Asynchronous IO with Vert.x

Peter Ledbrook - SpringSource

springsource
A division of vmware

New kid on the block

Javascript Non-blocking Single-threaded



Reactor pattern

What's it about?

- C10K Problem
  - How do you handle 10,000+ concurrent connections?
  - Without grinding to a halt?
- Why?
  - All those mobile devices!
  - Rise of long-lived connections

Real-time chat

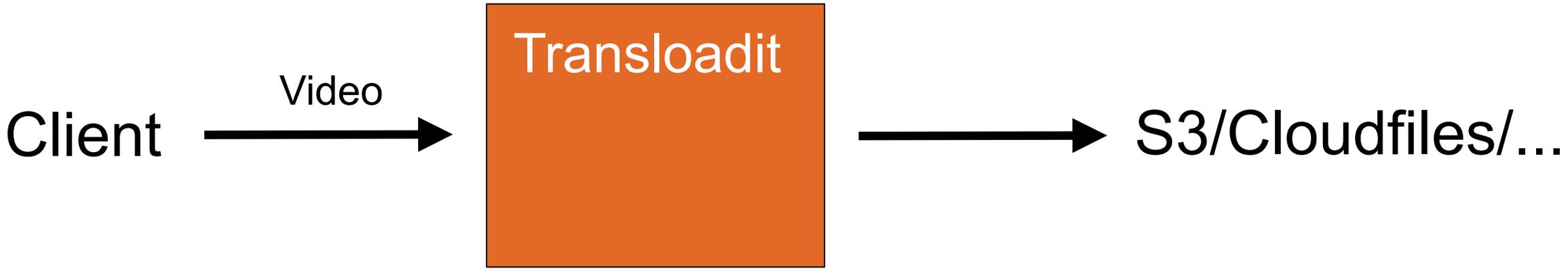
Instant messaging

Twitter

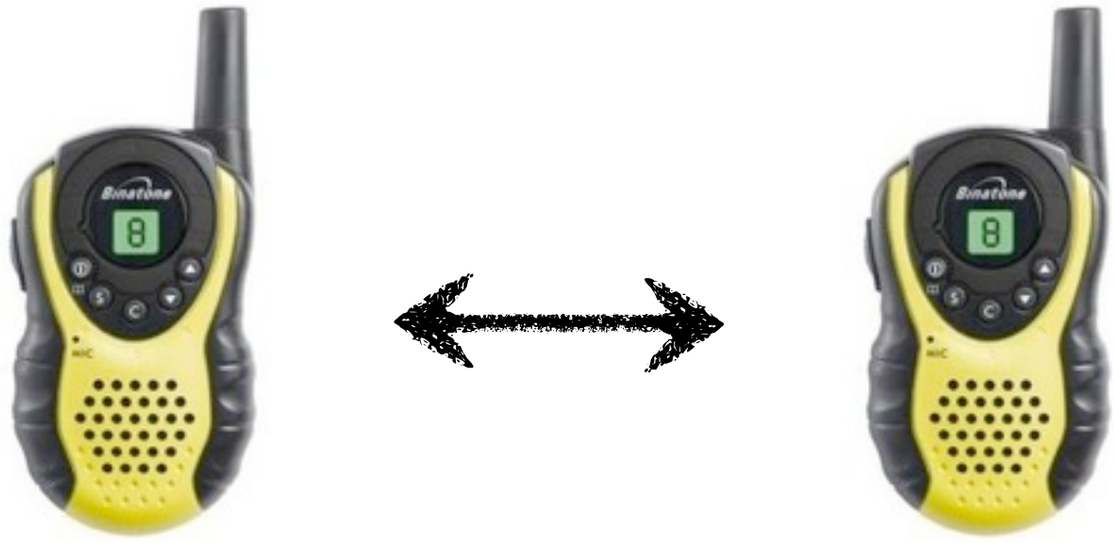
Voice and video

Transloadit

- *“Flexible, fast and scalable file uploading and encoding”*
- Video/audio encoding on the fly
  - 500mb/s



Voxer



- Walkie-talkie for iOS and Android
- Live audio
- Large number of connections
- Push notifications

Javascript and nothing but Javascript

What if?

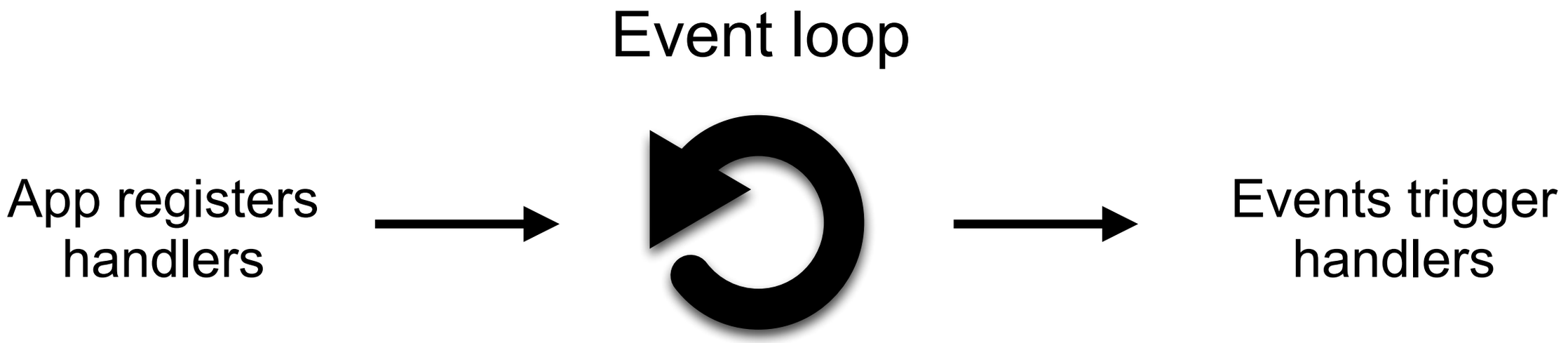
- Language neutral framework
- On the JVM
- Polyglot APIs
- Easy horizontal scalability
- Do non-eventy stuff in a non-eventy way

Vert.x!

Vert.x

- Written in Java
- Built on Netty and NIO 2
  - Java 7 only!
- Multiple language bindings
  - JRuby
  - Groovy
  - Javascript
  - Python/Clojure/Scala planned
- Based on the Reactor pattern

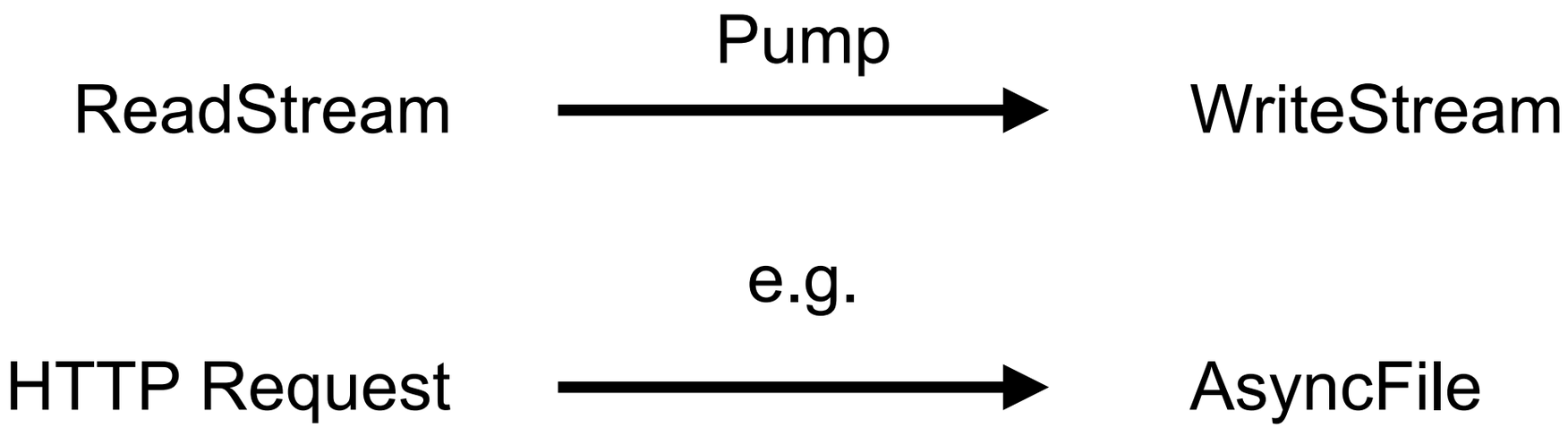
How it works



- Handlers executed synchronously
  - on a single thread
- Use handlers to pass messages
- Inter/intra application comms
  - EventBus with messages
  - Safe shared data structures

Features

- Non-blocking stdio
- TCP/IP and HTTP servers and clients
- Web Socket support
- (Non-blocking) Redis & Mongo modules (busmods)
- Asynchronous file system access and stream handling



Example - Echo socket

```
import static org.vertx.groovy.core.streams.Pump.createPump

vertx.createNetServer().connectHandler { socket ->
  createPump(socket, socket).start()
}.listen(1234)

println "Running echo server on port 1234"
```

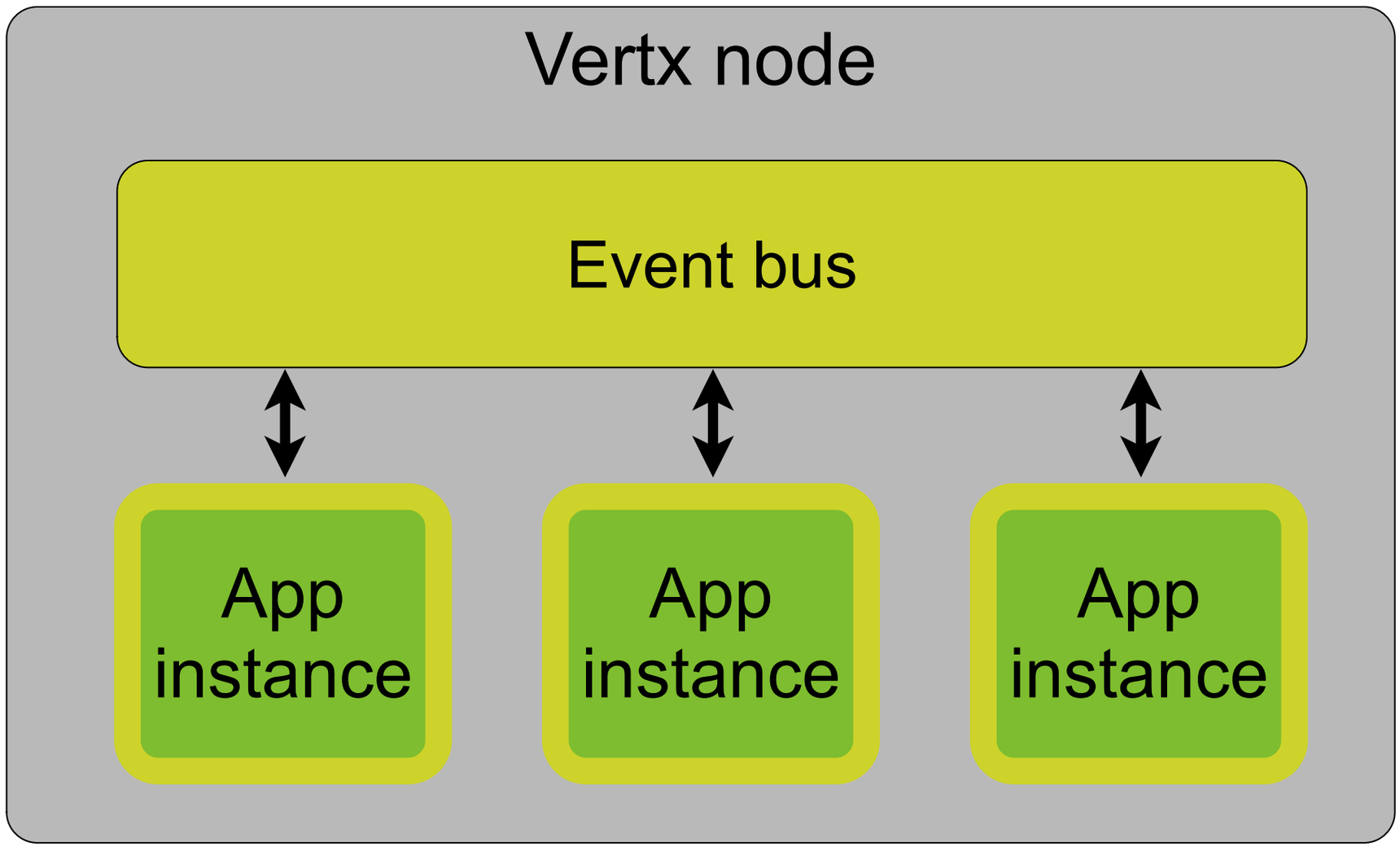
Example - Echo client

```
vertx.createNetClient().connect(1234, "localhost") { socket ->

  def msgCount = 10
  socket.dataHandler { buffer ->
    println "Net client receiving: ${buffer}"
  }

  // Now send some data
  msgCount.times {
    String str = "hello $it\n"
    print "Net client sending: $str"
    socket << str
  }

}
```

Scaling

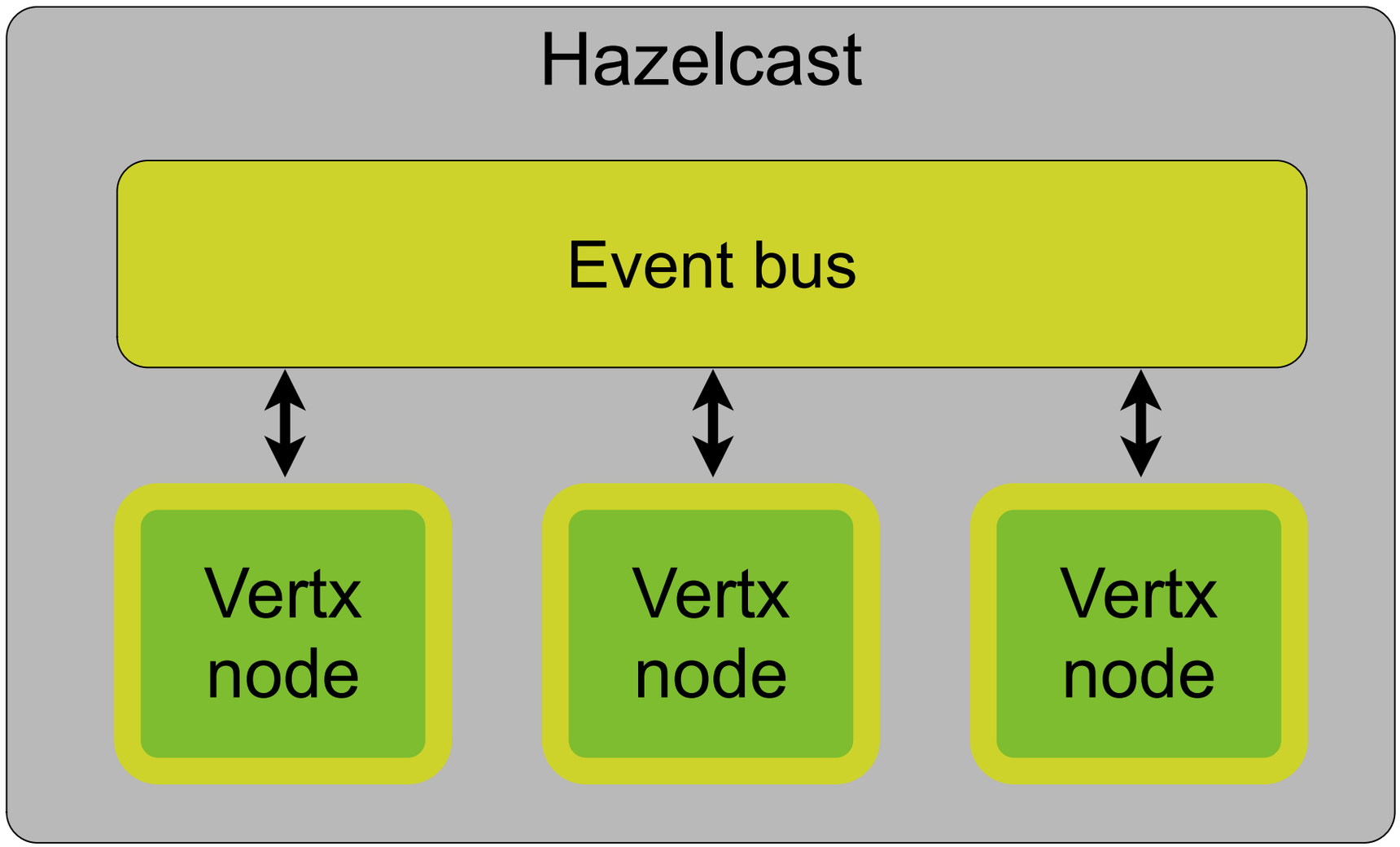
App instance

Scaling

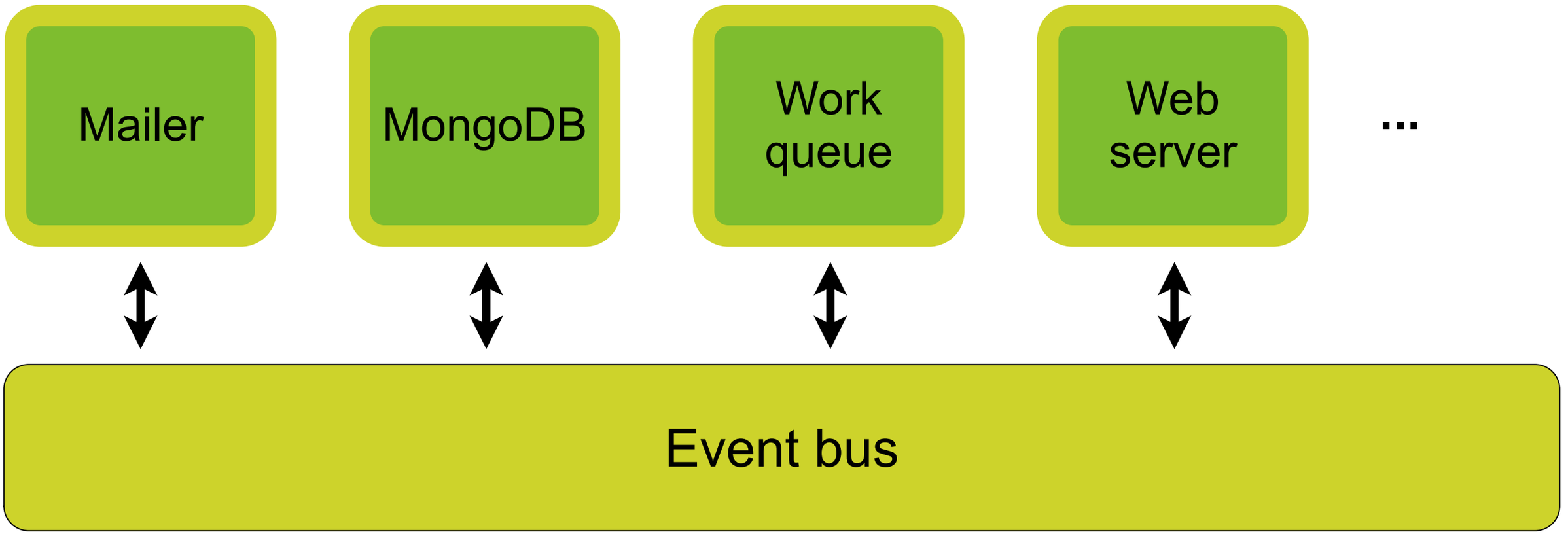


vertx run -instances 3

Scaling

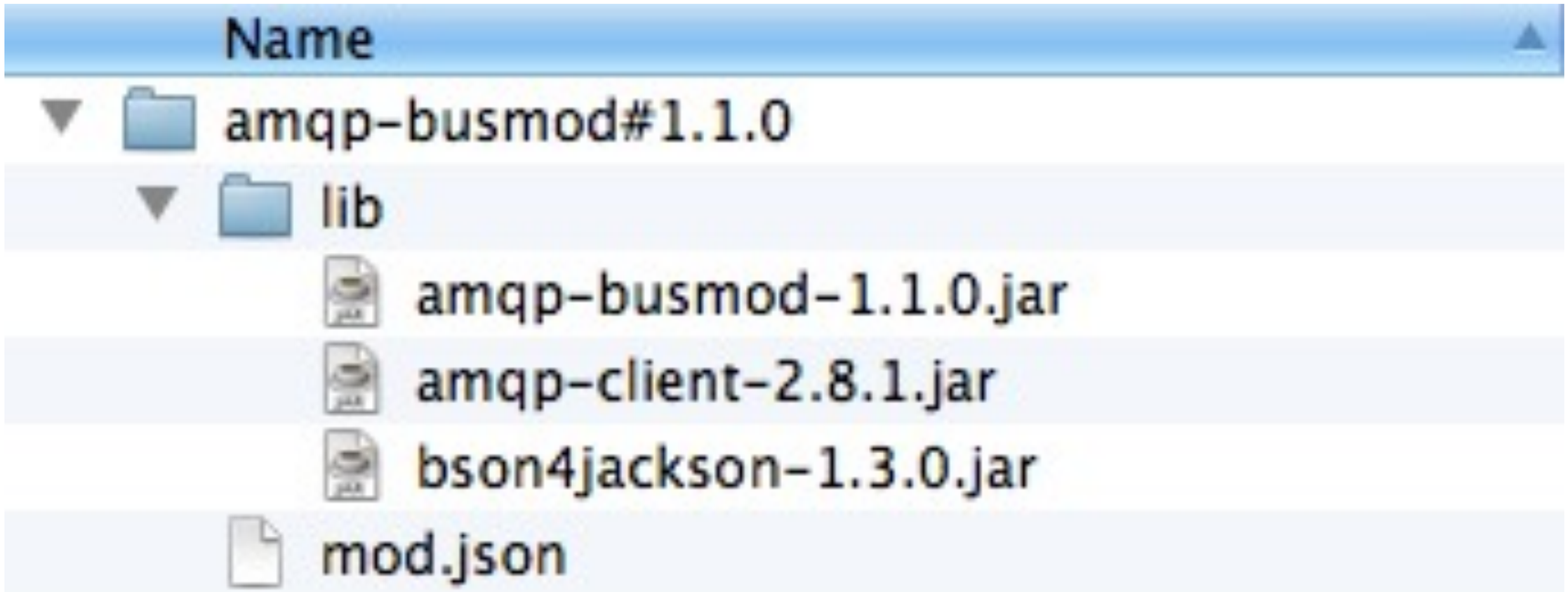


Extensibility via (Bus) Mods



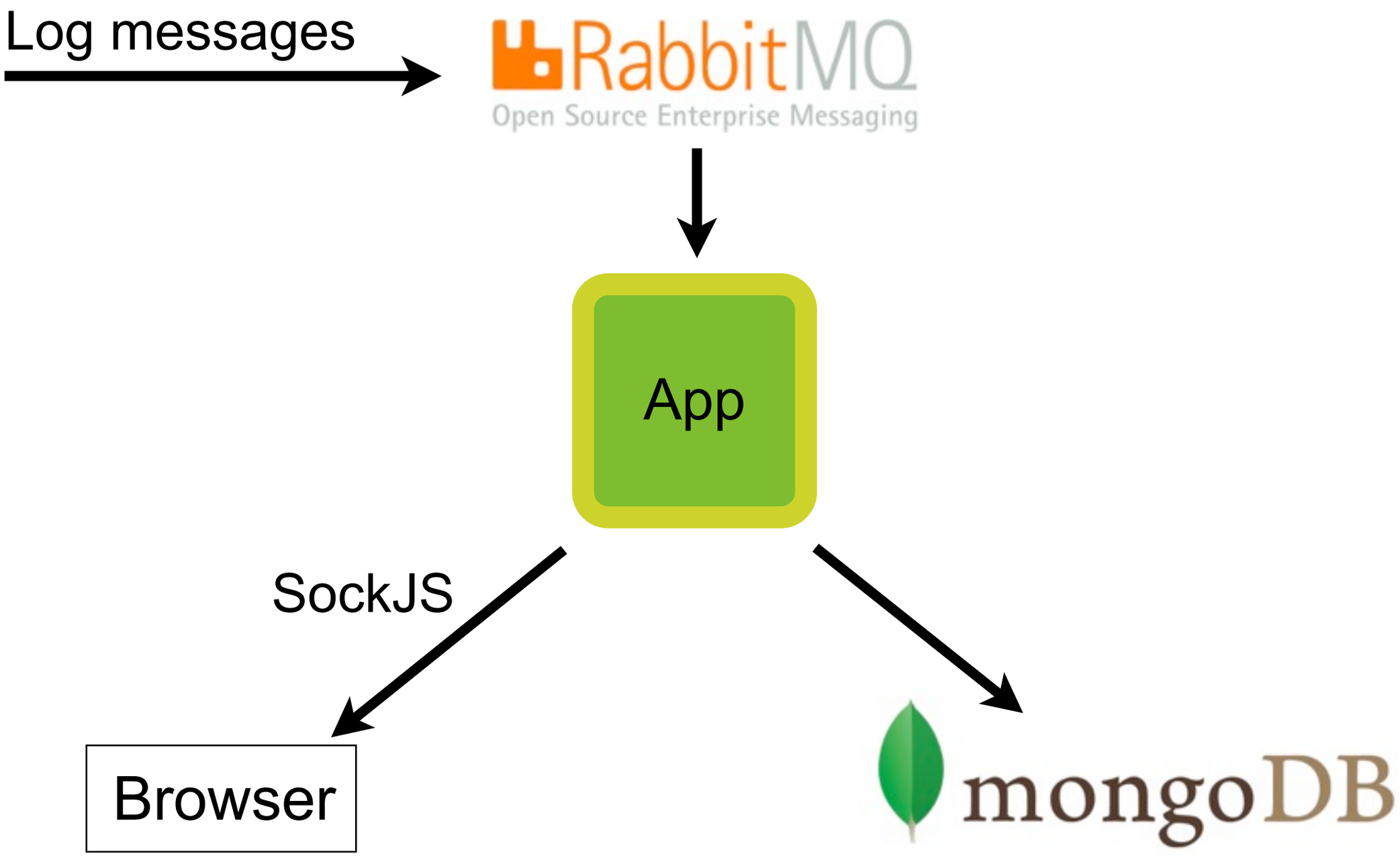
Mod structure

- Core mods in $VERTX_HOME/mods
- User mods in $VERTX_MODS



A more complex example

Realtime logging app



Realtime logging app

```
def eb = vertx.eventBus
def amqpAddress = "amqp_bridge"
def address = "amqp.logs"

// Start up AMQP bus mod.
container.deployWorkerVerticle(
        "amqp-busmod#1.1.0",
        [uri: "amqp://localhost", address: amqpAddress, defaultContentType: "application/json" ],
        1) {

    def createMsg = [exchange: "amq.topic", routingKey: "logs.#", forward: address ]
    eb.send("${amqpAddress}.create-consumer", createMsg) { reply ->
        startHttpServer()
    }

}
```

Realtime logging app

```
// MongoDB bus mod for storing the messages.
def mongoAddress = "vertx.mongopersistor"
container.deployWorkerVerticle("mongo-persistor", [address: mongoAddress, db_name: 'log-app'], 1) {

    // Clear existing logs first.
    eb.send mongoAddress, [action: "delete", collection: "logs", matcher: [:]]

    // Save incoming log messages to 'logs' collection.
    eb.registerHandler("logs") { msg ->
        eb.send mongoAddress, [action: "save", collection: "logs", document: [msg: msg.body.text] ]
    }
}

eb.registerHandler(address) { msg ->
    print "[log] ${msg.body.body}"
    eb.send "logs", [text: msg.body.body]
}
```

Realtime logging app

```
def startHttpServer() {
    def server = vertx.createHttpServer()

    // Serve the static resources
    server.requestHandler { req ->
        if (req.uri == '/') req.response.sendFile('index.html')
        if (req.uri == '/vertxbus.js') req.response.sendFile('vertxbus.js')
    }

    vertx.createSockJSServer(server).bridge(prefix: '/eventbus', [[:]])

    server.listen 8181

    println "Server started at http://localhost:8181/"
}
```

Demo

Summary

- Event-driven framework for the JVM
  - Low overhead
  - Handle large number of connections
  - Great for working with streams
- Pick your own language!
  - Which will of course be Groovy ;)
- Published release: 1.0.1
  - Still early days
  - Get involved now!

More info

- w: https://github.com/purplefox/vert.x
  http://purplefox.github.com/vert.x/
- f: http://groups.google.com/group/vertx

- t: pledbrook
- b: http://blog.springsource.com/author/peter-ledbrook/

Thank you!

Questions?