vert.x的特性

1.多语言：除了java语言外，还支持javascript、ruby、python、scala、groovy、clojure、php
等动态语言，这个很大程度归功于JVM对动态语言的支持。

2.非阻塞/事件驱动:非阻塞、异步、多路复用是高性能的关键，特别是在IO密集型的场景中，这些内容在C10k/C1XK问题中较深入的讨论。

3.灵活通用：用途广泛，既可用于一般的简单网络类应用，也可用于复杂的web应用、微服务、大容量的事件系统和消息系统等。它底层的网络库采用netty，支持http客户&服服务器、webscoket、tcp/udp、dns客户端应用的开发。

4.轻量：Vert.x核心包(core)只有650kB，基于它开发的web应用和服务不需要application server，如tomcat、jetty。伴随着Vert.x的成熟，也预兆着java application server将逐渐死亡或衰落。

5.负载均衡和高可用集群：这个是开箱即用的，只要配置下就行。

vert.x的架构

1.vert.x实例是vert.x api的入口点，我们调用vert.x中的核心服务时，均要先获取vert.x实例，通过该实例来调用相应的服务，例如部署verticle、创建http server。一个JVM中通常只有一个vert.x实例，也可有多个实例。每个vert.x实例有自己的事件循环（一个事件循环对应一个事件循环线程），其事件循环数量缺省是cpu核数的2倍（文档中提到缺省数量等于cpu核数，但经过测试发现vert.x 3.1.0版本中是2倍）。

2.verticle是vert.x中的组件，可理解成java中的servlet。

verticle可分为两种类型：标准verticle和worker verticle。

标准verticle运行在vert.x实例的事件循环线程中，也就是当有事件发生时，在事件循环线程中回调verticle实例中的event handler。标准verticle（更确切的说是

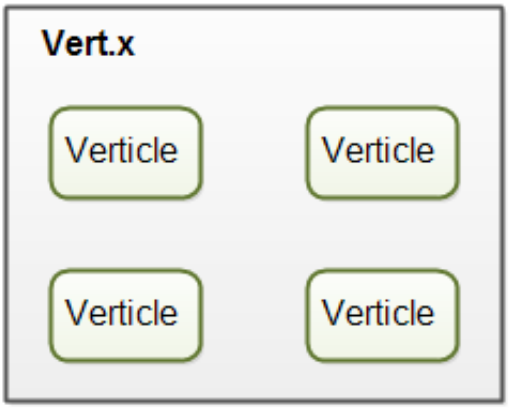
verticle event handler中的回调方法）不能阻塞事件循环

worker verticle在background workers线程池中执行，该线程池的大小缺省为40。
worker verticle又可分成两种：

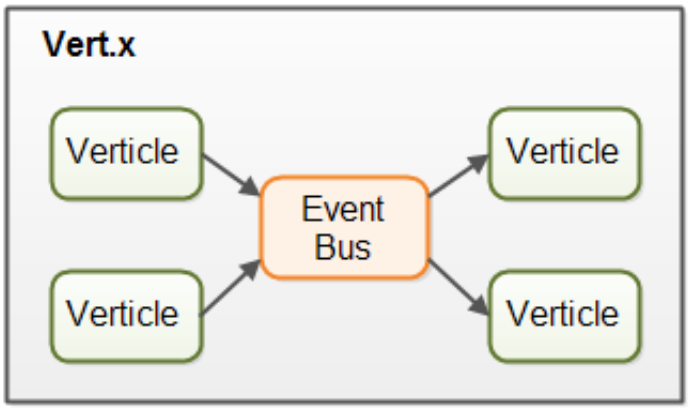
一种是多线程worker verticle，一个多线程worker verticle实例可在多个worker线程中并发执行

另一种是单线程worker verticle，在同一时间只能有一个线程执行（串行执行）

一句话：
Event Loop Vertical：事件的业务处理线程，存在于Event Loop中，用于处理非阻塞短任务。
Worker Vertical：事件的业务处理线程，用于处理长任务阻塞任务。

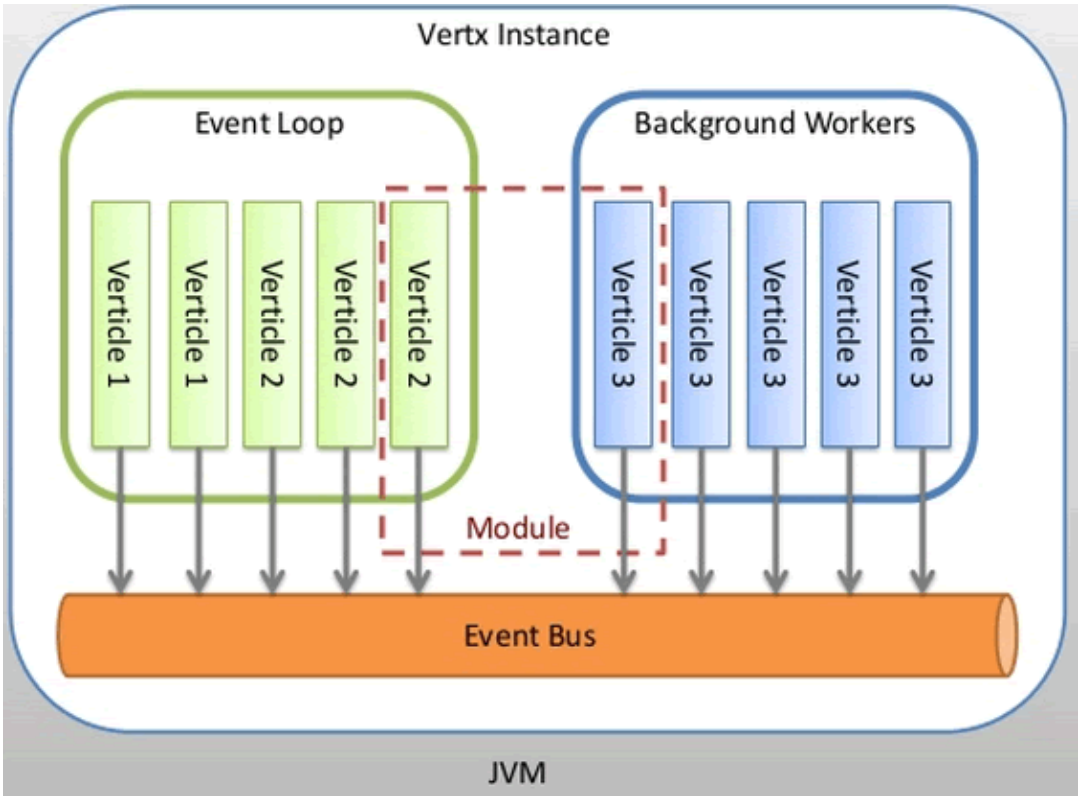


3.event bus可理解为一个分布式的消息队列系统，支持点对点（队列）、发布-订阅（topic）、请求-响应模式。verticle实例之间的事件均通过event bus进行传递，这些verticle实例可分布在不同的JVM或不同机器上，也可在用户的web浏览器上

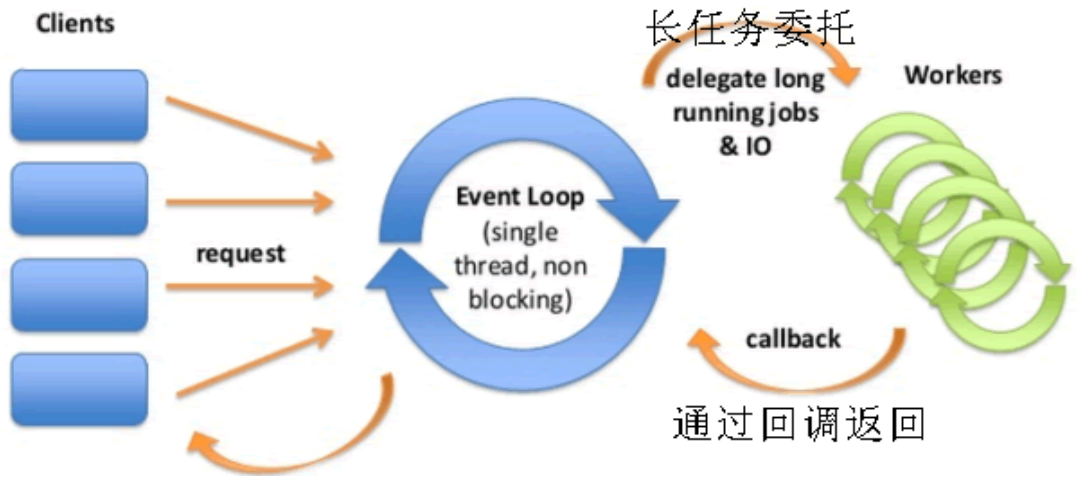


4.Event Loop：即事件循环 https://www.cnblogs.com/Nelsen8/p/7966661.html
Vert.x 不是单事件循环，每个 Vertx 实例都维护若干个事件循环。默认情况下，我们
选择数量基于在机器上可用的内核数，但可以自己设置

vert.x的架构图



如图所示：*一个jvm中通常有一个Vertx实例（也可以多个），每个实例可以包含多个verticle，不同类型的verticle分别在Event Loop 和 Background Workers中执行*



安装vert.x工具箱

vert.x可在https://vertx.io/下载，解压后将bin路径加到path中。运行vertx version命令，输出3.1.0即表示正确安装。

vertx命令可用来启动、停止vertx开发的应用，但在开发阶段vertx工具箱的安装不是必须的，结合maven可以自动下载vertx的jar包，使用java命令即可运行vertx开发的应用。