Vert.x là một framework để hỗ trợ rất nhiều ngôn ngữ như java, RxJava, Kotlin, JavaScript, Groovy, Ruby, Ceylon, Scala. Ở dây có rất nhiều cách để component trên Vert.x để dử dung. Và tùy biến dễ dàng.

**Vert.X**– Ta có thể dễ dàng dựng và dùng bất kì component hoặc library nào mà ta muốn.

**Vert.X**chạy trên JVM – Java Virtual Machine – cho phép bạn test code và scale tức thì. Nên cần cài đặt JDK trước.

**Vert.X**hỗ trợ rất nhiều chủng loại lập trình khác nhau, trong đó hỗ trợ lập trình Back End viết các API cực tốt. Còn hỗ trợ mở rộng xây dựng các ứng dụng MicroService.

**Vert.X**– được rất nhiều hãng công nghệ lớn trên thế giới đang sử dụng:

Để nhúng Vert.X ta thêm dòng lệnh sau vào nhóm depencies trong tập tin build.gradle:

dependencies {  
compile ‘io.vertx:vertx-web:3.9.4’  
}

3.9.4 là phiên bản mới nhất tính tới ngày 30/11/2020

Ta có thể xem chi tiết tại đây: <https://vertx.io/docs/vertx-web/java/>

Lõi của Vert.X

* Các API cốt lõi cua Vert.X chứa xương sống để viết các ứng dụng Vert.X và hỗ trợ cấp thấp cho HTTP, TCP, UDP, hệ thống tệp, luồng đồng bộ và nhiều khối xây dựng khác. Nó cũng ddwocjswar dung bởi nhiều thành phần khác của Vert.X.

Lõi của Vert.X cung cấp các chức năng sau:

* Viết máy khách và máy chủ TCP.
* Viết máy khách và máy chủ HTTP bao gồm hỗ trợ websocket.
* Luồng sự kiện.
* Dữ liệu được cia sẻ- bản đồ cục bộ và bản đồ phân tán.
* Hành động định kì và trì hoãn.
* Triển khai và hủy triển khai Verticls.
* Ổ cắm dữ liệu.
* Máy khách DNS.
* Truy caapjheej thống tập tin.
* Tính khả dung cao.
* Phương tiện giao thông bản địa.
* Phân cụm.

Chức năng trong lõi ở mức khá thấp bạn sẽ không tìm thấy những thứ như truy cập cơ sở dữ liệu, ủy quyền hoặc chức năng web cấp cao. Bạn chỉ có thể tìm thấy chúng ở Vert.Xext.

Lõi của Vert.X nhỏ và nhẹ, Bnạ có thể sử dụng những phần bạn muốn, nó cũng hoàn toàn có thể nhúng vào các ứng dung hiện có của bạn.

Bạn có thể dung bất kỳ ngôn ngữ nào mà Vert.X hỗ trợ, nếu bạn đang sử dụng Maven hay Gradle thì bạn hay thêm phần phụ thuộc sau vào để truy cập API của Vert.X.

Maven(trong porm.xml)

<dependency>

<groupId> io.vertx</groupId>

<artrifactId>vertx-core</artrifactId>

<version>4.3.5</version>

</dependency>

Gradle(build.gradle)

Deprndencies{

Compile ‘io.vertx:vertx-core:4.3.5’

}

Bạn không thể giao tiếp nhiều với Vert.X-land trừ khi bạn có thể giao tiếp với một đối tương Vert.X, đây là trung tâm điều khiển một Vert.X và là cachs để ta thực hiện hầu hết mội thứ, bao gồm tạo máy khách và máy chủ, nhận tham chiếu đến từ luồng sự kiện., đặt bộ hen giờ cũng như nhiều thứ khác.

Nếu bạn đang nhúng Vert.X thì chỉ cần tạo ra một phiên bản như sau:

Vertx vertx = Vertx.vertx();

Khi tạo một đối tượng Vert.X ta có thể đặt giá trị tùy chọn nếu giá trị mặc định không phù hợp với yêu cầu.

Vertx vertx = Vertx.vertx(new VertxOption().setWorkerPoolSize(40));

Đối tượng VertxOption có nhiều cài đặt và cho phép định cấu hình những thứ như phân cụm, tính khả dụng cao, kích thước nhóm và nhiều cài đặt khác.

1. Tạo một đối tượng Vert.x được nhóm

Để tao một nhóm Vert.X thì thường ta sẽ sử dụng biến thể không đồng bộ để tạo ra ra các đối tượng Vert.x.

Điều này là do thường mất một hoảng thời gian để các phiên bản Vert.X khác nhau trong một cụm nhóm lai với nhau, trong thời gian đó nó không chặn các cuộc gọi do vậy kết quả cung cấp không đồng bộ.

Khi mà bạn đã thông thạo Vert.X thì bạn có thể kết hợp nhiều lện gọi phương thwccs lại với nhau:

Request.response().putHeader(“Content-Type”, “text/pain”).end(“some text”);

Sâu chuỗi các cuộc gọi cho phép bạn viết mã ít dòng hơn.

HttpServerResponse response = resquest.response();

Response.putHeaaader(“Content-Type”, “text/plain”);

Response.write(“some text”);

Responsse.end();

Các API Vert.x phần lớn được điều khiển theo sự kiện . Điều này có nghĩa là khi có sự kiện xảy ra trong Vert.x mà bạn quan tâm, Vert.x sẽ gọi cho bạn bằng cách gửi cho bạn các sự kiện.

Gửi thông điệp

Vertx.setPeriodic(1000, id -> {

System.out.println(“timer fired”);

});

Nhận thông điệp

Server.resquestHandler(request -> {

Request.response().end(“Hello word”);

});

1. Đừng chặn tôi

Với rất ít trường hợp ngoại lệ, không có API nào chặn chuỗi cuộc gọi, Nếu mọt kết quả có thể được cung cấp ngay lập tức thì sẽ được trả về ngay lập tức. Vì không có API nào Vert.X chặn cuộc gọi nên ta có thể dùng Vert.X sử lý nhiều hoạt động cùng một lúc bằng một số lượng nhỏ luồng.

Với API chặn cuộc gọi thông thường có thể chặn cuộc gọi khi:

* Đọc dữ liệu từ socket.
* Ghi dữ liệu vào đĩa.
* Gửi tin nhắn đến người nhận và chờ trả về.
* Nhiều tình huống khác.

Trong tất cả các tình huống trên khi luồng của bạn đang chờ kết quả nó không thể làm gì khác- nó thực sự vô dụng.

Điều này có nghia là nếu bạn muốn đồng thời sử lý nhiều hoạt động thì dùngAPI chặn thì bạn cần nhiều luồng để ngăn ứng dụng không bị tắt.

Các luồng có chi phí chung về bộ nhớ mà chúng yêu cầu và trong chuyển đổi ngữ cảnh.

Đối với các ứng dụng hiện đại API chặn không được dùng nhiều.

1. Lò phản ứng và đa lò phản ứng

* Vì không có gì trong Vert.X hoặc các khối ứng dụng vòng lặp sự kiện có thể thaoir mái chạy quanh, việc cung cấp sự kiện liên tục cho các trình sử lý khác nhau khi chúng đến.
* Vì không có gì chặn nên vòng lặp sự kiện có thể cung cấp một lượng lớn các sự kiện trong một khoảng thời gian ngắn, đây đượ gọi làn lò phản ứng.
* Trong quá trình triển khai lò phản ứng tiêu chuẩn có một chuỗi vòng lặp các vòng ặp sự kiện duy nhất chạy quanh vòng lặp phân phối tất cả các

Sự kện tới tất cả các quá trình.

Vấn đè với một luồng đơn là nó chỉ có thể chạy trên một lõi tại một thời điểm, vì vậy nếu muốn lò phản ứng đơn chạy quy mô trên máy chủ đa lõi, bản phải khởi động quản lý mọi quá trình khác nhau.

* Vert.X hoạt động khác ở đây, thay vì một vòng lặp sự kiện duy nhất có mỗi phiên bản có thể có một vòng lặpsự kiện. Theo mặc định thì số lõi sẽ chọn theo số lượng lõi của máy, nhưng điều này cos thể bị ghi đè.
* Một quy trình Vert.X có thể mở rộng quy mô trên máy chủ, đây là đa lò phản ứng.

1. Không chặn vòng lặp sự kiện

* Chúng ta đã biết rằng các API của Vert.X không chặn vòng lặp của sự kiện nhưng điều đó không giúp ích gì nhiều, Nếu bạn thực hiện chặn vòng lặp sự kiện thì vòng lặp sẽ không hoạt động trong quá trình bị chặn, nếu chặn tất cả các vòng lặp sự kiện thì ứng dụng sẽ dừng hoàn toàn.
* Ví dụ về chặn:
* Thread.sleeep();
* Chờ đợi trên một ổ khóa.
* Đang đợi trên một mutex hoặc màn hình.
* Thực hiện thao tác cơ sở dữ liệu lâu dài và chờ kết quả.
* Thực hiện một phép tính phức tạp cần một khoảng thời gian.
* Quay trong một vòng lặp.
* Nếu bất cứ trương hợp nào ở trên xảy ra thì ta hải chuuyeenr nó về chế độ tự do và chờ hướng dẫn thêm.
* Nếu bạn muốn sử lý 1000 yêu cầu khác nhau mỗi giây thì rõ ràng mỗi yêu cầu không thể mất quá 0.1 milies.
* Nếu ứng dụng của bạn không chạy có nghoa là bạn đang chặn sự kiện ở đâu ddos, để giúp bạn chẩn đoán các sự kiện như vậy Vert.X sẽ tự động chẩn đoán các cảnh báo nếu các vòng lặp sự kiện không quay lại.
* Vert.X cũng cung cấp ngăn xếp để giúp xác định chính xác vị trí xảy ra chặn.

1. Kết quả trong tương lai

* Vert.X dùng hợp đồng tương lai để biểu thị kết quả đồng bộ.
* Bất kì phương thức không đồng bộ nào cũng trả về một Future đối tượng cho kết quả: thanh công hay thất bại.
* Bạn không thể tướng tác trực tiếp vói kết quả của tương lai thay vào đó bạn cần đặt một trình sử lý được gọi khi tương lai hoàn thành.

FileSystem = fs = vertx.fileSystem();

Future<FilePróp> future = fs.props(“/my\_file.txt”);

Future.onComplete((AsyncResult<FileProps> ar ->{

If(ar.succeeded()){

FileProps props = ar.result();

System.out.println(“File size = “ + props.size());

}else{

System.out.println(“Fallure” + ar.cause().getMessage());

}

});

1. Sáng tác tương lai

Compose có thể dùng để xâu chuỗi tương lai:

* Khi tương lai hiện tại hoàn thành hãy áp dung hàm sau để trả về tương lai.

FileSystem fs = vertc.fileSystem();

Future<Void> future = fs

.createFile(“/foo”)

.compose(v -> {

Return fs.writeFile(“/foo”, Bufer.buffer());

})

.compose(v -> {

Return fs move(“/foo”, “/bar”);

});

* Một tập tin được tạo ra
* Dữ liệu sẽ lưu trữ ở tập tin này.tập tin được di chuyển