11 | 反应式编程框架设计：如何使方法调用无阻塞等待？

2022-03-11 李智慧

讲述：李智慧
时长 12:07　大小 11.10M

你好，我是李智慧。

反应式编程本质上是一种异步编程方案，在多线程（协程）、异步方法调用、异步 I/O 访问等技术基础之上，提供了一整套与异步调用相匹配的编程模型，从而实现程序调用非阻塞、即时响应等特性，即开发出一个反应式的系统，以应对编程领域越来越高的并发处理需求。

反应式系统应该具备如下的 4 个特质。

- **即时响应**：应用的调用者可以即时得到响应，无需等到整个应用程序执行完毕。也就是说应用调用是非阻塞的。
- **回弹性**：当应用程序部分功能失效的时候，应用系统本身能够进行自我修复，保证正常运行，保证响应，不会出现系统崩溃和宕机的情况。

- **弹性**：系统能够对应用负载压力做出响应，能够自动伸缩以适应应用负载压力，根据压力自动调整自身的处理能力，或者根据自身的处理能力，调整进入系统中的访问请求数量。
- **消息驱动**：功能模块之间、服务之间通过消息进行驱动，以完成服务的流程。

目前主流的反应式编程框架有 RxJava、Reactor 等，它们的主要特点是基于**观察者设计模式**的异步编程方案，编程模型采用函数式编程。

观察者模式和函数式编程有自己的优势，但是反应式编程并不是必须用观察者模式和函数式编程。我们准备开发一个纯消息驱动，完全异步，支持命令式编程的反应式编程框架，框架名称为“Flower”。

需求分析

互联网及物联网场景下的应用系统开发，基本上都是高并发系统开发。也就是说，在同一个时刻，会有大量的用户或设备请求到达系统，进行计算处理。但是传统的编程模型都是阻塞式编程，阻塞式编程有什么特点，会产生什么问题呢？我们来看一段代码示例。

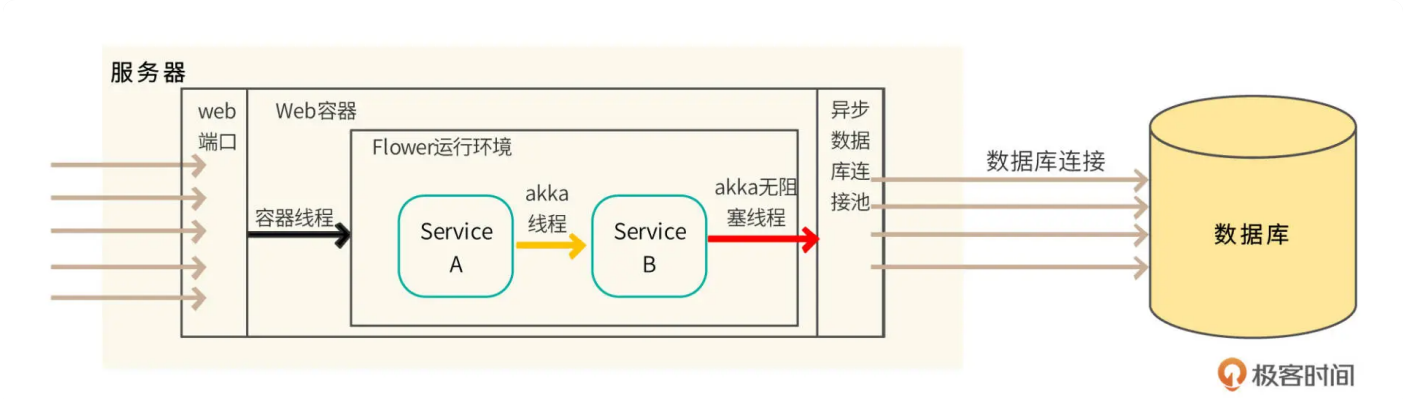
复制代码

```
void a(){
  ....
  int x = m();
  int y = n();
  return x + y;
}
```

在方法 a 中调用了方法 m，那么在方法 m 返回之前，就不会调用方法 n，即方法 a 被方法 m 阻塞了。这种编程模型下，方法 m 和方法 n 不能同时执行，系统的运行速度就不会快，并发处理能力就不会很高。

还有更严重的情况。服务器通常为每个用户请求创建一个线程，而创建的总线程数是有限的，每台服务器通常几百个。如果方法 m 是一个远程调用，处理比较慢，当方法 a 调用方法 m 时，执行方法 a 的线程就会被长期挂起，无法释放。如果所有线程都因为方法 m 而无法释放，导致服务器线程耗尽，就会使服务器陷入假死状态，外部表现就是服务器宕机，失去响应，系统严重故障。

Flower 框架应该满足如下典型 Web 应用的线程特性。



当并发用户请求到达应用服务器时，Web 容器线程不需要执行应用程序代码，它只是将用户的 HTTP 请求变为请求对象，将请求对象异步交给 Flower 框架的 Service 去处理，而 Web 容器线程自身立刻就返回。

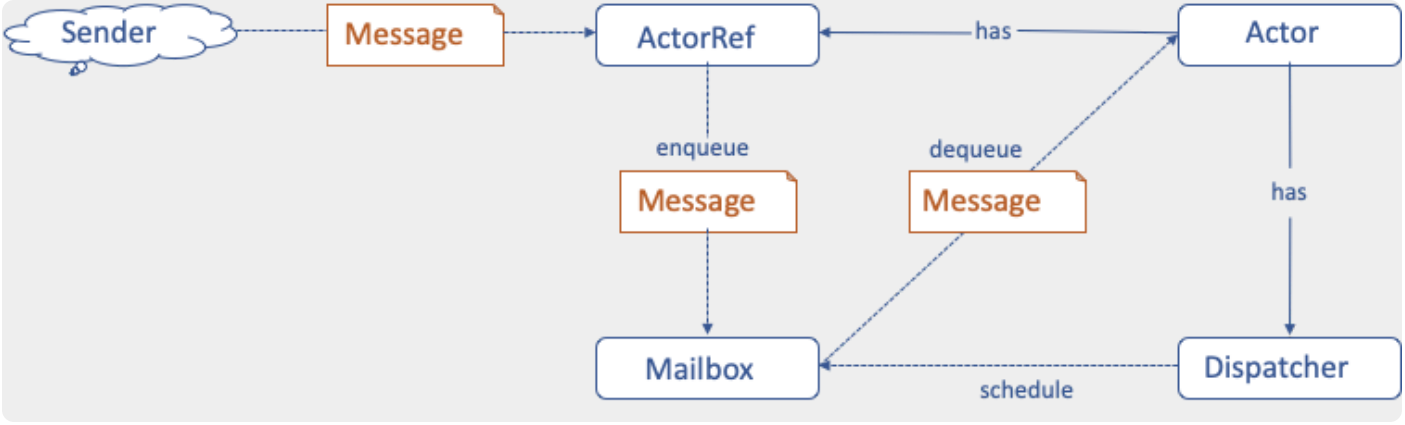
如果是传统的阻塞式编程，Web 容器线程要完成全部的请求处理操作，直到返回响应结果才能释放线程，所以需要很多 Web 容器线程。但使用 Flower 框架只需要极少的容器线程就可以处理较多的并发用户请求，而且容器线程不会阻塞。

同样，在 Flower 框架中，用户请求交给业务 Service 对象以后，Service 之间依然是使用异步消息通讯而非阻塞式的调用。一个 Service 完成业务逻辑处理计算以后，会返回一个处理结果，这个结果会以消息的方式异步发送给下一个 Service。

概要设计

Flower 框架实现异步无阻塞，一方面是利用了 Java Web 容器的异步特性，主要是 Servlet3.0 以后提供的 AsyncContext，快速释放容器线程；另一方面则利用了异步的数据库驱动和异步的网络通信，主要是 HttpAsyncClient 等异步通信组件。而 Flower 框架内，核心应用代码之间的异步无阻塞调用，则是利用了 Akka 的 Actor 模型。

Akka Actor 的异步消息驱动实现如下。



一个 Actor 向另一个 Actor 发起通讯时，当前 Actor 就是一个消息的发送者 Sender，它需要获得另一个 Actor 的 ActorRef，也就是一个引用，通过引用进行消息通信。而 ActorRef 收到消息以后，会将这个消息放到目标 Actor 的 Mailbox 里面，然后就立即返回了。

也就是说，一个 Actor 向另一个 Actor 发送消息时，不需要等待对方真正地处理这个消息，只需要将消息发送到目标 Actor 的 Mailbox 里面就可以了。Sender 不会被阻塞，可以继续执行自己的其他操作。而目标 Actor 检查自己的 Mailbox 中是否有消息，如果有，则从 Mailbox 里面获取消息，并进行异步的处理。而所有的 Actor 会共享线程，这些线程不会有任何的阻塞。

但是 Actor 编程模型无法满足人们日常的编程习惯以及 Flower 的命令式编程需求，所以我们需要将 Akka Actor 封装到一个 Flower 的编程框架中，并通过 Flower 提供一个新的编程模型。

Flower 基于 Akka 的 Actor 进行开发，将 Service 封装到 Actor 里面，并且将 Actor 收到的消息作为参数传入 Service 进行调用。

Flower 框架的主要元素包括：Flower Service（服务）、Flower 流程和 Flower 容器。其中，Service 实现一个细粒度的服务功能，Service 之间会通过 Message 关联，前一个 Service 的返回值（Message），必须是后一个 Service 的输入参数（Message）。而 Flower 容器就负责在 Service 间传递 Massage，从而使 Service 按照业务逻辑编辑成一个 Flow（流程）。

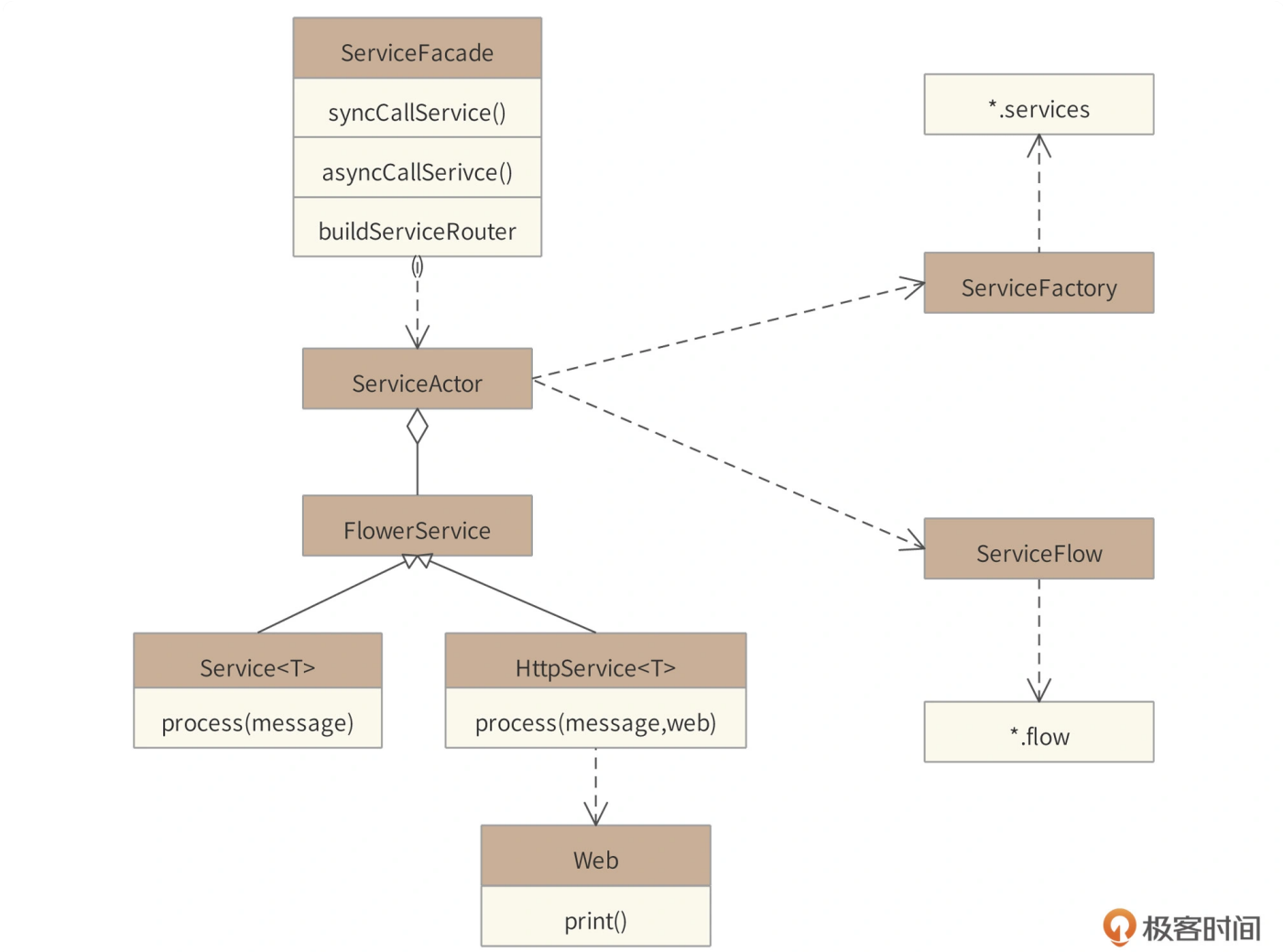
在 Flower 内部，消息是一等公民，基于 Flower 开发的应用系统是面向消息的应用系统。消息由 Service 产生，是 Service 的返回值；同时消息也是 Service 的输入。前一个 Service 的返回消息是下一个 Service 的输入消息，**没有耦合**的 Service 正是通过消息关联起来，组成一个 Service 流程，并最终构建出一个拥有完整处理能力的应用系统。流程举例：

```
// -> service1 -> service2 -> service5 -> service4
//      ^      |             ^              |
//      |       -> service3 -|              |
//      |_____________________________________|
```

复制代码

详细设计

Flower 核心类图如下。

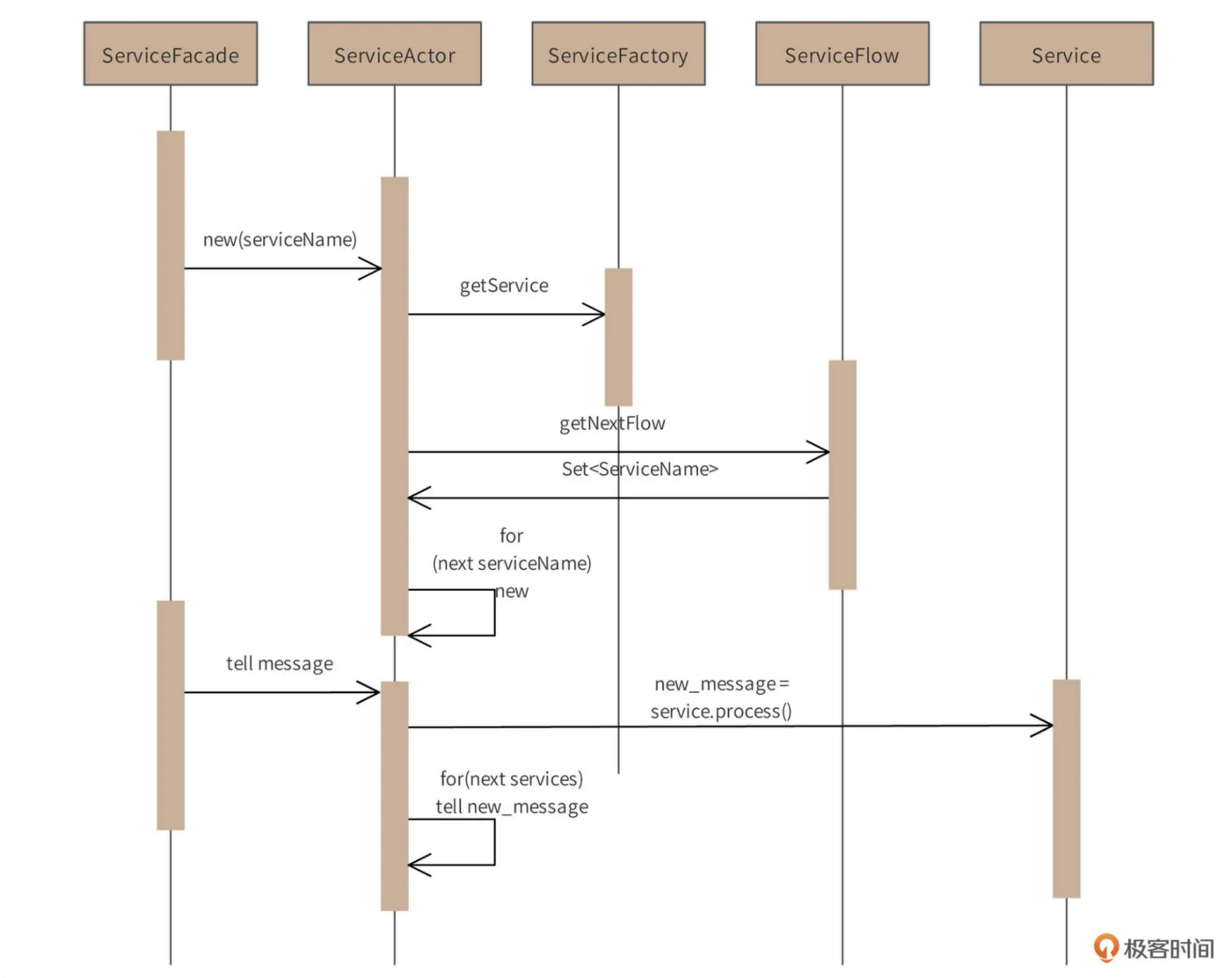


Flower 框架核心关键类及其职责如下：

1. Service 以及 HttpService 接口是框架的编程核心，开发者开发的 Service 需要实现 Service 或者 HttpService 接口。HttpService 与 Service 的不同在于 HttpService 在接口方法中传递 Web 参数，开发者利用 Web 接口可以将计算结果直接 print 到 HTTP 客户端；
2. ServiceFactory 负责用户以及框架内置的 service 实例管理（加载 *.services 文件）；
3. ServiceFlow 负责流程管理（加载 *.flow 文件）；

4. ServiceActor 将 Service 封装到 Actor。

Flower 初始化及调用时序图如下。



图中包含两个过程，第一个过程是**服务流程初始化**过程。首先，开发者通过 ServiceFacade 调用已经定义好的服务流程。然后，ServiceFacade 根据传入的 flow 名和 service 名，创建第一个 ServiceActor。这个 ServiceActor 将通过 ServiceFactory 来装载 Service 实例，并通过 ServiceFlow 获得当前 Service 在流程中所配置的后续 Service（可能有多个）。依此递归，创建后续 Service 的 ServiceActor，并记录其对应的 ActorRef。

时序图中的第二个过程是**消息流处理**过程。调用者发送给 ServiceFacade 的消息，会被 flow 流程中的第一个 ServiceActor 处理，这个 ServiceActor 会调用对应的 Service 实例，并将 Service 实例的返回值作为消息发送给流程定义的后续 ServiceActor。

使用 Flower 框架开发应用程序，就是开发各种 Service，开发服务 Service 类必须实现 Flower 框架的 Service 接口或者 HTTP 接口，在 process 方法内完成服务业务逻辑处理。Service 代码示例如下。

```
复制代码
1  public class UserServiceA implements Service<User, User> {
2    static final Logger logger = LoggerFactory.getLogger(UserServiceA.class);
3    @Override
4    public User process(User message, ServiceContext context) throws Throwable {
5      message.setDesc(message.getDesc() + " --> " + getClass().getSimpleName());
6      message.setAge(message.getAge() + 1);
7      logger.info("结束处理消息, message : {}", message);
8      return message;
9    }
10 }
```

服务注册

开发者开发的服务需要在 Flower 中注册才可以调用，Flower 提供两种服务注册方式：配置文件方式和编程方式。

编程方式示例如下。

```
复制代码
1  ServiceFactory serviceFactory = flowerFactory.getServiceFactory();
2  serviceFactory.registerService(UserServiceA.class.getSimpleName(), UserService
3  serviceFactory.registerService(UserServiceB.class.getSimpleName(), UserService
4  serviceFactory.registerService(UserServiceC1.class.getSimpleName(), UserServic
```

配置文件方式支持用配置文件进行注册，服务定义配置文件扩展名: .services，放在 classpath 下，Flower 框架自动加载注册，比如 flower_test.services。配置文件内容如下。

```
复制代码
1  UserServiceA = com.ly.train.flower.base.service.user.UserServiceA
2  UserServiceB = com.ly.train.flower.base.service.user.UserServiceB
3  UserServiceC1 = com.ly.train.flower.base.service.user.UserServiceC1
```

流程编排

在 Flower 中，服务之间的依赖关系不能通过传统的服务之间依赖调用实现，如开头的方法 a 调用方法 m 那样。而需要通过流程编排方式，实现服务间依赖。服务编排方式也有两种，配置文件方式和编程方式。

下面的例子演示的是以**编程方式**编排流程。

```
// UserServiceA -> UserServiceB -> UserServiceC1
final String flowName = "flower_test";
ServiceFlow serviceFlow = serviceFactory.getOrCreateServiceFlow(flowName);
serviceFlow.buildFlow(UserServiceA.class, UserServiceB.class);
serviceFlow.buildFlow(UserServiceB.class, UserServiceC1.class);
serviceFlow.build();
```

而流程**配置文件方式**则使用扩展名: .flow，放在 classpath 下，Flower 框架会自动加载编排流程。 比如 flower_test.flow，文件名 flower_test 就是流程的名字，流程执行时需要指定流程名。配置文件内容示例如下。

```
UserServiceA -> UserServiceB
UserServiceB -> UserServiceC1
```

我们将服务 Service 代码开发好，注册到了 Flower 框架中，并通过流程编排的方式编排了这几个 Service 的依赖关系，后面就可以用流程名称进行调用了。调用代码示例如下，其中 flowName 是流程的名字，user 是流程中的一个 Service 名，是流程开始的 Service。

```
final FlowRouter flowRouter = flowerFactory.buildFlowRouter(flowName, 16);
flowRouter.asyncCallService(user);
```

Flower 框架源代码及更多资料可参考 https://github.com/zhihuili/flower。

小结

架构师是一个技术权威，他应该是团队中最有技术影响力的那个人。所以，架构师需要具备卓越的代码能力，否则就会沦为 PPT 架构师。PPT 架构师可以一时成为团队的焦点，但是无法

长远让大家信服。

那么架构师应该写什么样的代码？架构师如果写的代码和其他开发工程师的代码一样，又何以保持自己的技术权威，实现技术领导？简单来说，代码可以分成两种，一种代码是给最终用户使用的，处理用户请求，产生用户需要的结果；另一种是给开发工程师使用的，各种编程语言、数据库、编译器、编程框架、技术工具等等。

编程语言、数据库这些是业界通用的，但是编程框架、技术工具，每个公司都可以依据自身的业务特点，开发自己的框架和工具。而架构师应该是开发框架的那个人，每个开发工程师都使用架构师的开发框架以及约定的编程规范开发代码。架构师通过这种方式落地自己的架构设计，保持自己的技术影响。

也许你的开发中不会用到反应式编程，你可能也不需要深入学习 Flower 框架如何设计、如何使用。但是希望你能通过本文学习到如何设计一个编程框架，结合你所在公司的业务场景，将来开发一个你自己的编程框架。

思考题

Flower 纯消息驱动、异步无阻塞的优良特点，适合许多对并发处理要求高，需要快速、及时响应的场景，你能想到的现实应用场景有哪些呢？

欢迎在评论区分享你的思考，我们共同进步。

上一篇　10 | 搜索引擎设计：信息搜索怎么避免大海捞针？

下一篇　12 | 高性能架构的三板斧：分析系统性能问题从哪里入手？

精选留言 (9)

写留言

2022-03-15

因为异步，无阻塞。我觉得其实最主要解决的是io访问和网络请求。其基本原理是，CPU的运算速度远远高于IO访问（比如磁盘访问，外设访问），更远高于网络请求（局域网内的数据库，或者微服务下的服务间通信）

传统串行化，就好像，我要组装一台电脑，我拿到图纸以后，图纸第一步是组装机箱，我订购了一个机箱3天到货，然后机箱里要放主板，我又定了一个主板2天到货，主板上要放cpu，我又定了一个CPU半个月到货。。。最终一台电脑组装好，两个月过去了。组装时间取决于所有产品订购到货的总时间。

我理解的响应式就是，我收到图纸，发现图纸要CPU，主板，内存，硬盘，机箱。。。我全部发起订购。然后都开始送货，等货全到齐了之后，开始组装。这个时候，组装时间取决于最长到货的那个配件。所以大幅提升了性能。

反过来说，如果你本身不需要io访问，网络调用之类的操作。响应式对于性能的提升其实是很有限的。

应用场景的话，我觉得可以有金融领域的风控。因为风控其实是一整串校验，而且这一串校验服务很可能是独立的。甚至部分服务是由不同的供应商提供的。如果这种业务场景，走串行化之行，执行效率肯定是不可接受的。使用响应式，可以很大程度缓解这个问题。只需要保证，我的每一个风控校验的服务，或者提供商，响应时间控制在一个范围内，就能保证整个请求的执行时间不会太离谱。

作者回复: 很赞，确实，Akka的很多应用案例都出自金融行业。

共 2 条评论 >

3

ABC

2022-03-30

老师有推荐的反应式框架吗？最近想学习一下。

作者回复: Akka

1

peter

2022-03-11

请教老师几个问题啊：

Q1：服务器创建几百个线程与线程公式的矛盾问题。

记得有一个公式，线程数等于CPU核数的2倍。假设CPU有二十个核，则线程数是40。服务器创建几百个线程，有什么用？与这个公式不矛盾吗？

Q2：akka是什么意思？

Q3：目前的主流JAVA开发中有响应式开发框架吗？

A JDK中有响应式开发的东西吗？

B 目前微服务一般用SpringBoot/SpringCloud，这两个部分有响应式开发的东西吗？

C 后端架构，常见的组件:Nginx、Redis、MQ(e.g,RocketMQ)、ES,这几个组件有响应式开发的东西吗？

Q4：Flower的消息部分是怎么实现的？队列吗？

Q5：RxJava也是封装Actor、消息驱动吗？

Q6：这个框架开发难度大吗？老师一个人多长时间能完成开发？

作者回复:

1 其一，最佳线程数不止和CPU数目有关，还有IO等待时间有关，这正是Flower关注的，线程公式如下：

最佳线程数=[任务执行时间/(任务执行时间 - IO 等待时间)] * CPU 内核数

其二，应用程序开启多少线程，和以上指标都没关系，想开多少开多少。

2 https://akka.io/

3 文中提到两种反应式编程框架，都是Java框架
A Java Streaming
B Flutter，另外，Flower也是一个反应式微服务框架，更彻底的反应式微服务，具体可以看github文档
C 没有，反应式编程和这些没关系

4 利用Akka Actor传递消息

5不是

6 第一个版本，一个人一个月，具体可以看git log

1

学而不思则惘
2022-03-31

看下来我理解是一个countdownLatch的作用？

作者回复: 并不是，flower可以做到用一个线程执行（看起来）并发执行数百个方法，并发处理数百个请求

javaadu
2022-03-26

我是做风控策略引擎的，在策略引擎中，同一个事件过来，需要并行跑很多模型、特征，然后统一决策。这些模型和特征都是外部的系统，对于io并发的要求很高，使用响应式编程有助于减少阻塞环节。

其他的场景，暂时没想到，看看其他同学的回答

作者回复: 赞，学习了~

201
2022-03-26

有点晕，只能多刷几次理解了。

HappyHasson

2022-03-21

首先，对java不熟悉，所以这里面的有些概念不清楚，导致理解困难。

作者能不能画一张示例图，说明各个service怎么并行处理，然后综合结果给到请求端的。
我是看着看着被绕晕了，文中的概念名词很像

作者回复: 可以参考github里的文档哈，https://github.com/zhihuili/flower。


Geek_7347cf

2022-03-11

非http异步调用 怎么获取执行service的返回值，感觉flow原理和netty框架比较像

作者回复: HttpService接口提供web参数，调用web.print()返回响应结果。

共 2 条评论 >

Geek_7347cf

2022-03-11

文中提到执行service方法有异步和同步 同步有返回值 是阻塞的，异步没有返回值 ，但是大多数情况下是需要返回值的 否则搞成生产者消费者不更简单吗

作者回复: 方法的返回值和请求的返回值不是一回事，Service可以在方法内部调用web.print(), 直接返回请求的响应结果。
计算请求的响应结果，需要多个Service异步计算得到的，这些Service之间异步无阻塞，通常在流程的最后一个Service调用web.print()，返回请求处理结果。