微服务模式—以Java举例

# 微服务架构中的跨进程通信

本章内容

* 应用通信模式：远端进程调用，熔断器，客户端侧发现，自注册，服务端侧发现，第三方注册，异步消息，事务性发件箱，事务性日志跟踪，轮询发布者；
* 跨进程通信在微服务框架中的重要性；
* 定义及演进API；
* 各种各样的跨进程通信选项及其权衡；
* 通信使用异步消息的好处；
* 可靠地发送作为数据库事务一部分的消息；

像许多开发者一样，玛丽和她的团队有跨进程通信机制的经验。FTGO应用有一个REST API供移动端应用和浏览器端JavaScript使用。它也使用各种云服务，例如Twilio消息服务和Stripe支付服务。但在像FTGO这样的单体应用中，模块间调用是通过语言级别的方法或者函数调用实现的。除非需要与REST API或与云服务集成的模块打交道，FTGO开发者通常不需要考虑跨进程通信（IPC）。

与之对比的是，如你在第二章看到的那样，微服务架构将应用拆成一组服务。为处理请求这些服务必须经常协作。因为服务实例通常是运行在不同机器上的进程，他们必须使用IPC进行交互。IPC在微服务架构中有着比在单体应用中重要得多的作用。所以，在迁移应用为微服务，玛丽和FTGO开发者需要花很多的时间来考虑IPC。

IPC机制选择颇多。现今，流行的做法是REST（使用JSON）。然而需要记住所有的选项没有银弹。你必须小心考虑这些选项。本章将探讨各种各样的IPC选项，包括REST和消息，并讨论他们的权衡。

对IPC机制的选择是重要的架构决策。他会影响应用的可获得性。并且，正如我将在本章和下一章解释的那样，IPC甚至会影响到事务管理。我偏爱包含使用异步消息通信的松耦合服务的架构。像REST这样的同步协议绝大部分用于与其他应用通信。

本章开始时我将回顾微服务架构中的跨进程通信。接下来，我将描述基于IPC的远程进程调用，这其中REST是最流行的例子。我会讨论包括服务发现和怎样处理部分失效这样的重要话题。然后，我会描述基于IPC的异步消息。我也会讨论在保证消息顺序、正常处理重复消息并事务性消息通信的同时扩容消费者。最后，我会过下自含服务的概念，其处理同步请求而无需与其他服务通信以提高可用性。

## 微服务架构中的跨进程通信回顾

有许多的IPC技术可供选择。服务可以使用同步请求/基于响应的通信机制，例如基于HTTP的REST或者gRPC。也可以使用异步的、基于消息的通信机制，例如AMQP或者STOMP。有众多的消息格式。服务可使用人可读的、基于文本的格式，例如JSON或者XML。也可以选择更高效的二进制格式，例如Avro或者Protocol Buffers。

在深入到具体的技术细节前，我想提出几个你应该考虑的设计问题。本节我首先讨论交互方式，一种技术无关方式来描述客户端和服务间交互。接着，我讨论在微服务架构中精确定义API的重要性，包括API优先的设计概念。随后，我讨论API演进这一重要话题。最后，我讨论消息格式的不同选项，以及它们怎样决定API演进的难度。让我们来首先看看交互方式。

### 交互方式

在为服务API选择IPC机制前，先想想服务和客户端间的交互方式是有用的。首先考虑交互方式会帮助你聚焦于需求而不必陷入具体的IPC技术的细节中。同样，如3.4节描述的那样，选择交互方式会影响应用的可用性。你将会在第9章和第10章看到，这也会帮助你选择合适的集成测试策略。

有多种客户端与服务端交互方式。如表3.1所示，这些方式可归为两个维度。第一个维度是交互是一对一还是一对多

* 一对一：每个客户端请求只被一个服务处理；
* 一对多：每个请求被多个服务端处理；

第二个维度是交互是同步还是异步的

* 同步：客户端期待服务端即时响应，在等待时甚至会阻塞；
* 异步：客户端不阻塞，响应（如果存在）不必立即发送。

|  |  |  |
| --- | --- | --- |
|  | 一对一 | 一对多 |
| 同步 | 请求/响应 | - |
| 异步 | 异步请求/响应  单向通知 | 发布/订阅  发布/异步响应 |

以下是不同的一对一交互：

请求/响应：客户端向服务端发出请求并且等待响应。客户端期待响应能即时到达，在等待中甚至会阻塞。这种交互方式通常会导致服务间紧耦合；

异步请求/响应：客户端向服务端发出请求，并异步回应。客户端在等待过程中不会阻塞，因为服务端可能会长时间不发送响应。