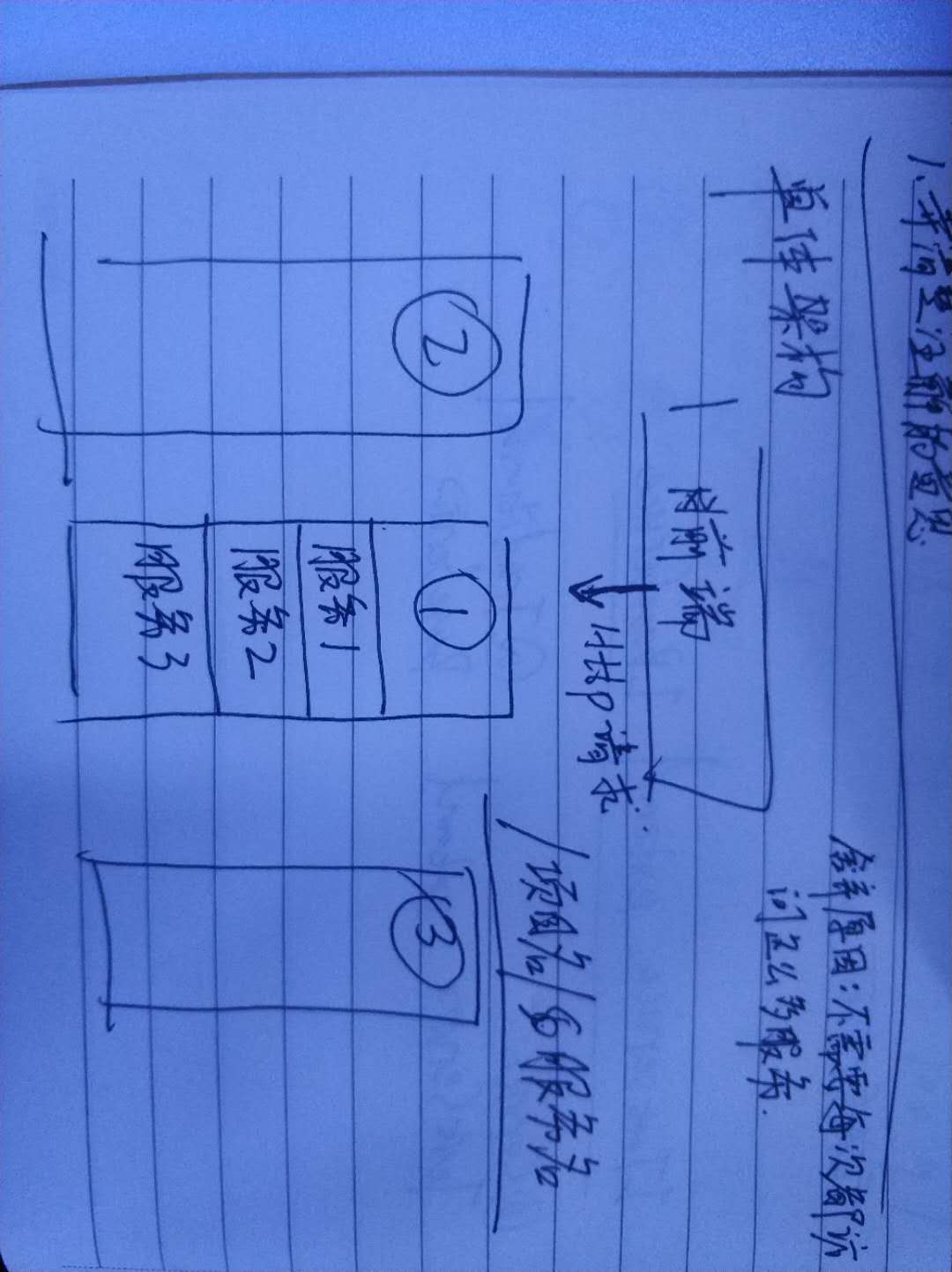
# 单体应用架构MVC



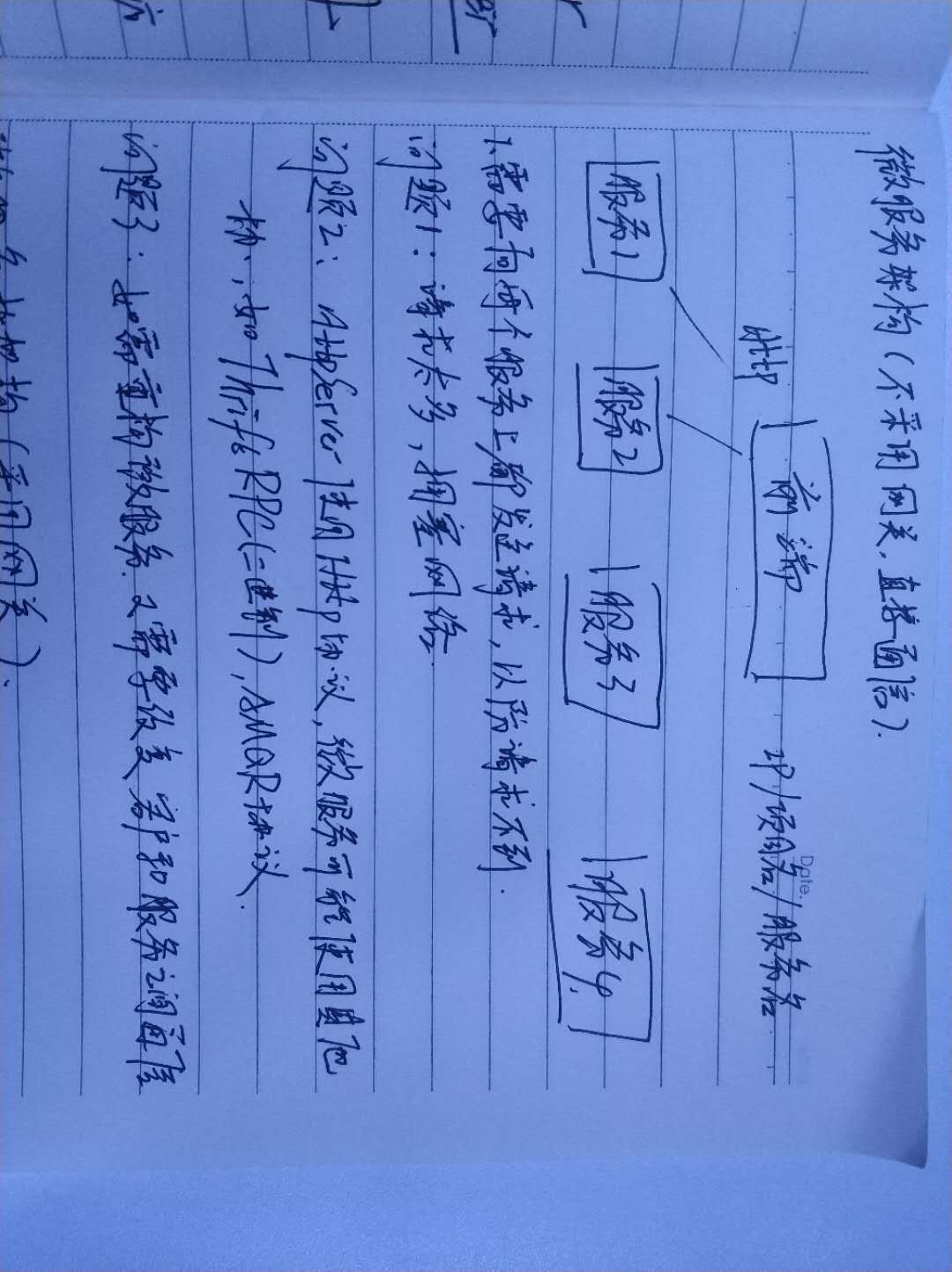
1. 访问流程

使用单体应用架构时，手机客户端可以简单地通过一个 REST 接口检索到该数据。**一个负载均衡器将请求路由到 N 个相同的应用实例中的一个。**然后这个应用实例查询多张数据库表，最后将数据响应给客户端。

1. 问题

当只需要服务1的时候，却要访问全部的应用服务，造成了效率的低下。对于服务器部署应用来说，也会造成服务器资源的浪费。当出现大量请求的时候，会造成网络拥塞。

# 微服务架构（不采用网关，客户端和微服务直接交互）

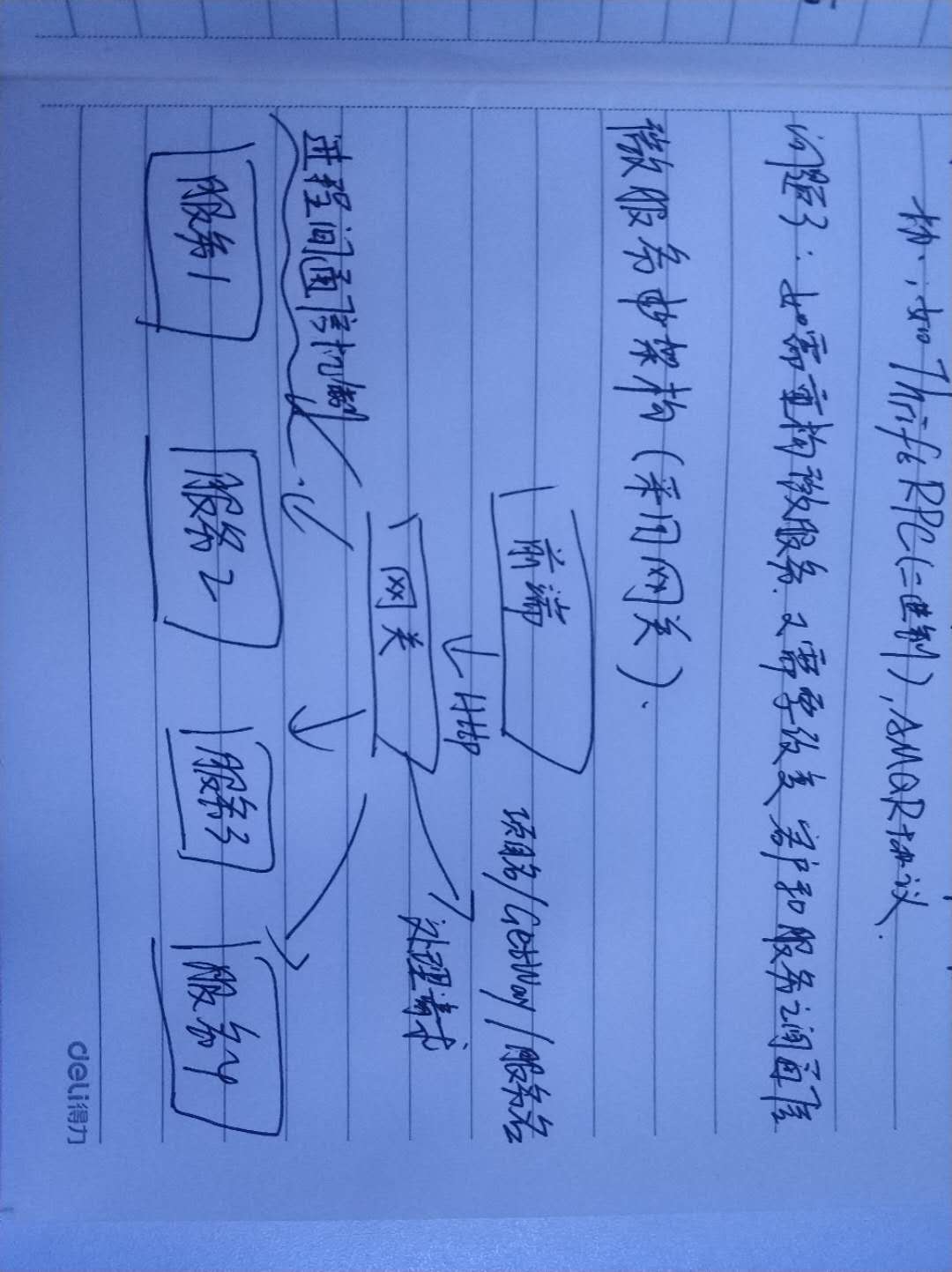


1. 访问流程

客户端需要向每个服务架构都发送请求，因为客户端不知道服务架构的具体位置。

1. 问题：
2. 客户端需求和每个微服务暴露的细粒度的 API 是不匹配的。在这个例子中，客户端需要分别发送七次请求，在更复杂的应用中可能需要请求更多次。
3. 一些微服务使用的协议不是 web 友好的。一个服务使用的是 Thrift RPC 的二进制协议，而另一个服务可能使用的是 AMQP 消息协议，这两个协议对浏览器和防火墙都是不友好的，最好是在内部使用。一个应用应该使用例如 HTTP 和 WebSocket 这样能穿透防火墙的协议。
4. 会使微服务的重构比较苦难。随着时间的推移，我们可能需要将某个系统拆分成服务。例如，我们可能将两个服务合并成一个，或者将一个服务拆分成两个甚至更多的服务。不管怎样，如果客户端和很多服务之间都是直接通信，这样的重构可能是极端困难。

# 微服务架构（采用网关）



1. 访问流程

**API 网关负责请求的路由、组合和协议转换，所有来自客户端的请求首先都要经过 API 网关，然后它再路由请求到合适的微服务。**API 网关处理请求的方式通常是，调用多个微服务，然后合并响应结果。API 网关可以在 web 协议（如 HTTP、WebSocket）和内部使用的 web 不友好的协议之间做转换。

1. 网关调用多个微服务的意思是，例如登录的时候要先校验用户名账号，之后再调用具体的业务，所以调用了多个微服务。而不是在网关中会再进行代码层面的整合。

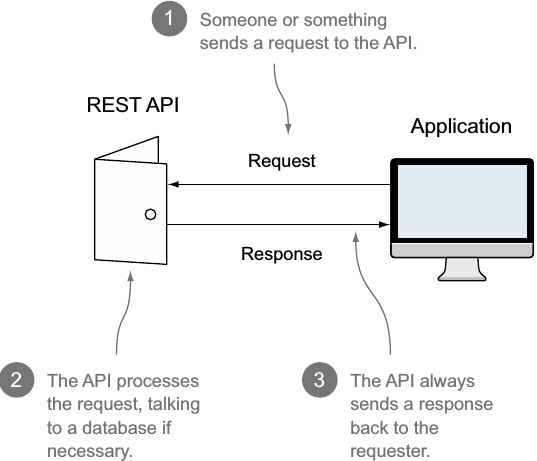
# 网关

1. 性能和伸缩性
2. 构建 API 网关的时候使用异步调用和非阻塞 I/O，来达到伸缩性网关的目的。
3. 有很多种不同的技术都可以用来实现可伸缩的 API 网关，在 JVM 平台中你可以使用 Netty、Vertx、Spring Reactor 或者 JBoss Undertow 等这些基于 NIO 的框架
4. 在非 JVM 的平台中流行的技术是 Node.js，它是运行在 chrome 的 JavaScript 引擎中的，另一个选择是 Nginx Plus，Nginx Plus 提供了一个成熟的、可扩展的、高性能的 Web 服务器，并且还提供了易于部署、配置和编程的反向代理，Nginx Plus 可以管理授权、访问控制、请求的负载均衡、响应的缓存，并提供了应用本身的健康检查和监控。
5. 响应式编程
6. 响应式编程是一种面向数据流和变化传播的编程范式。这意味着可以在编程语言中很方便地表达静态或动态的数据流，而相关的计算模型会自动将变化的值通过数据流进行传播。例如，在命令式编程环境中，a:=b+c 表示将表达式的结果赋给 a，而之后改变 b 或 c 的值不会影响 a。但在响应式编程中，a 的值会随着 b 或 c 的更新而更新。
7. 响应式概念的例子有 Scala 中的 Future，Java 8 中的 CompletableFuture，JavaScript 中的 Promise。也还有 Reactive Extensions，也叫 Rx 或者 ReactiveX，最初是微软为 .Net 平台开发的；Netflix 创建了 JVM 平台的 RxJava，并将它使用在他们的 API 网关上；也有 JavaScript 上的 RxJS，它运行在浏览器或者 Node.js 上。使用响应式方法编写 API 网关的代码简单又高效。
8. 服务调用
9. 基于微服务的应用是一个分布式系统，必须使用进程间通信机制。
10. 一种是异步的、基于消息的机制，有些是用消息中间件（JMS or AMQP）实现的，其它的是直接与服务通信的无中间件模式，如 Zeromq。
11. 进程间通信的另外一种方式是采用同步机制，如 HTTP 或者 Thrift。
12. 通常一个系统会同时使用同步和异步方式，甚至会使用每种方式的不同实现形式，因此 API 网关必须支持多种通信机制。
13. 服务发现
14. API 网关要像其他的服务客户端一样，需要使用系统的服务发现机制：ServerSide Discovery 或者 ClientSide Discovery。
15. 如果系统使用的是客户端侧的发现，API 网关必须能够查询到服务注册中心，它是所有微服务实例和对应位置的数据库存储。

# REST接口

## 简介

REST 指一组架构约束条件和原则，满足约束条件和原则的应用程序设计。满足该规则设计的接口都可以叫做REST接口。



## 特点

1. 每个资源都应该有唯一的一个标识（一切都是资源）
2. 使用标准的方法更改资源的状态
3. request 和 response 的自描述
4. 资源多重表述
5. 无状态服务

## 具体规则

1. 用 HTTP 协议里的动词来实现资源的添加，修改，删除等操作。即通过 HTTP 动词来实现资源的状态扭转：

GET 用来获取资源

POST 用来新建资源（也可以用于更新资源）

PUT 用来更新资源，

DELETE 用来删除资源。

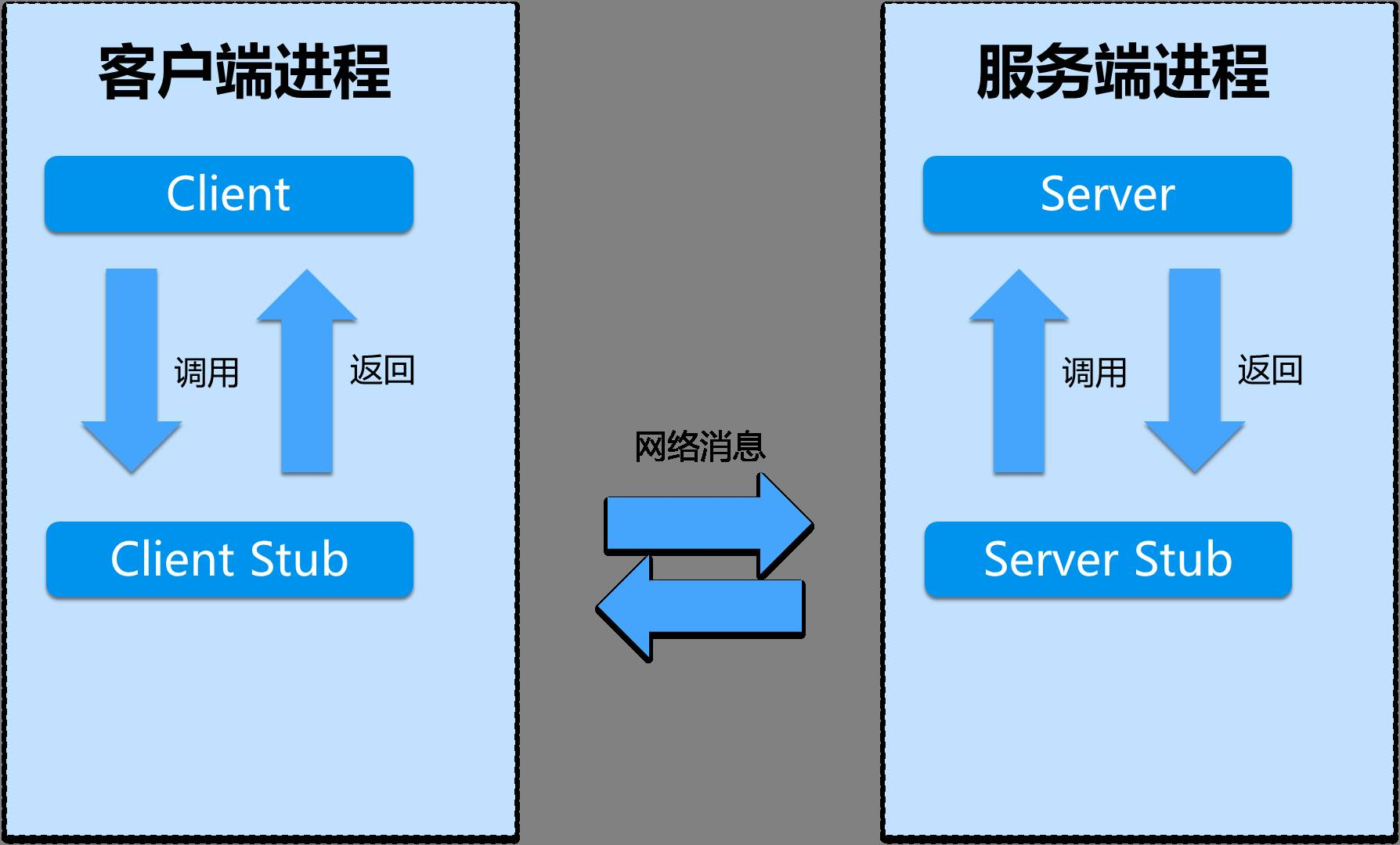
1. 通俗的来说就是前端可以通过url直接操作数据库资源。

# RPC

## 简介

RPC（即 Remote Procedure Call，远程过程调用），调用远程服务就跟调用本地服务一样方便，一般用于将程序部署在不同的机器上，供客户端进行调用。

主要流程是 Client -> Client Stub -> Network -> Server Stub -> Server 执行完成之后再进行返回。



## HTTP协议和RPC的关系

1. RPC是一种架构，在客户机上像调用本地方法一样，去调用服务机上的方法，这叫做远程过程调用。而HTTP协议是实现RPC架构的一种方式，也就是说HTTP和RPC不是并行的概念，RPC也可以用其他协议去实现。
2. HTTP的优点是可以跨系统跨语言跨平台去调用，主要用于公网上的客户端去访问服务器，不过要经过三次握手四次挥手，所以效率会比较低。其他比如JAVA上的RMI也是RPC的实现方式之一，不过RMI只能适用于客户机和服务机都是JAVA的场景，主要用于公司内部分布式架构的调用。
3. 在分布式架构中，当客户端发送请求到服务网关，网关解析后发到调用中心，此时就是应用RPC请求的场景。调用中心需要访问注册中心，进而发现服务，在公司内网中都是通过RPC请求去调用服务，因为RPC的效率比较高，适用于分布式架构。