1. PRC的介绍

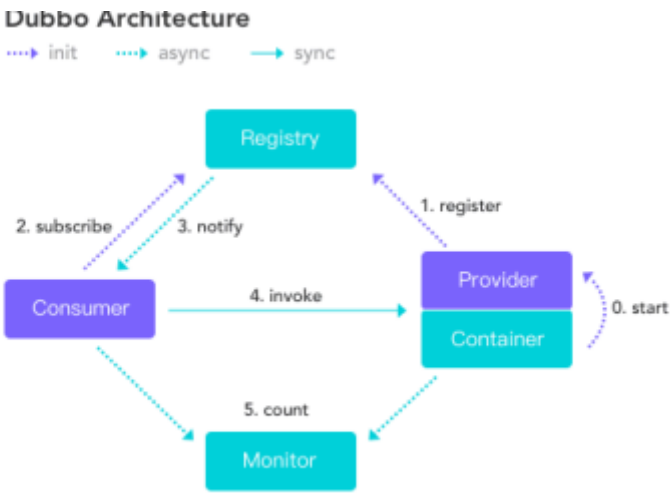
了解RPC之前先介绍一下IPC，即Inter-Process Communication进程间通信，指进程或者是线程之间传送数据和信号的一些技术和方法。本质是为了解决资源隔离的不同进程间互访资源问题。RPC是指远程过程调用，也就是说两台服务器A，B，一个应用部署在A服务器上，想要调用B服务器上应用提供的函数/方法，由于不在一个内存空间，不能直接调用，需要通过网络来表达调用的语义和传达调用的数据。比如在一台计算机中的A进程写了一个吃饭的方法，那在以前如果在B进程中也要有一个吃饭的方法，必须要在B进程中进行创建，但有了RPC后B只需要调用A进程的程序即可完成，这些进程可以是在同一台计算机上，也可能是在网络联通的不同计算机上。根据进程所处位置不同，进程间通信的方法包括两类：

本地过程调用(LPC，Local Procedure Call)：LPC用在多任务操作系统中，使得同时运行的任务能互相会话。这些任务共享内存空间使任务同步和互相发送信息。

远程过程调用(RPC，Remote Procedure Call)：RPC是一种进程间通信方式。它允许程序调用另一个地址空间的过程或函数，另一个地址空间通常是共享网络的另一台机器上，这样就不用程序员显式编码这个远程调用的细节。

2. 什么是RPC

RPC是一种远程调用过程，是一种通过网络远程调用其他服务的协议。通俗的说就是，A通过打电话的方式让B帮忙办一件事，B办完事后将结果告知A。 在一个完整的RPC框架中存在的角色以及整个远程调用的过程大致是以下。



节点角色说明：

Provider: 暴露服务的服务提供方。
Consumer: 调用远程服务的服务消费方。
Registry: 服务注册与发现的注册中心。
Monitor: 统计服务的调用次调和调用时间的监控中心。
Container: 服务运行容器

0. 服务容器负责启动，加载，运行服务提供者。
1. 服务提供者在启动时，向注册中心注册自己提供的服务。
2. 服务消费者在启动时，向注册中心订阅自己所需的服务。

3. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。
4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。
5. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

3. RPC 功能目标

RPC 的主要功能目标是让构建分布式应用更容易，在提供强大的远程调用能力时不损失本地调用的语义简洁性。为实现该目标，RPC 框架需提供一种透明调用机制让使用者不必显式的区分本地调用和远程调用，

4. RPC 调用分类

RPC 调用分以下两种：

1. 同步调用： 客户方等待调用执行完成并返回结果。
2. 异步调用： 客户方调用后不用等待执行结果返回，但依然可以通过回调通知等方式获取返回结果。若客户方不关心调用返回结果，则变成单向异步调用，单向调用不用返回结果。

5. 为什么需要RPC

而一个技术的出现不是为了使用新技术而去使用，而是旧技术存在某些瓶颈，存在难以支撑或者扩展性差等问题诸多问题之后，需要用新技术来进行解决。所以说RPC有哪些优点呢?

1.在一个大型的网站，内部子系统较多、接口非常多的情况下，RPC框架的好处就显示出来了，首先就是长链接，不必每次通信都要像http 一样去3次握手什么的，减少了网络开销；

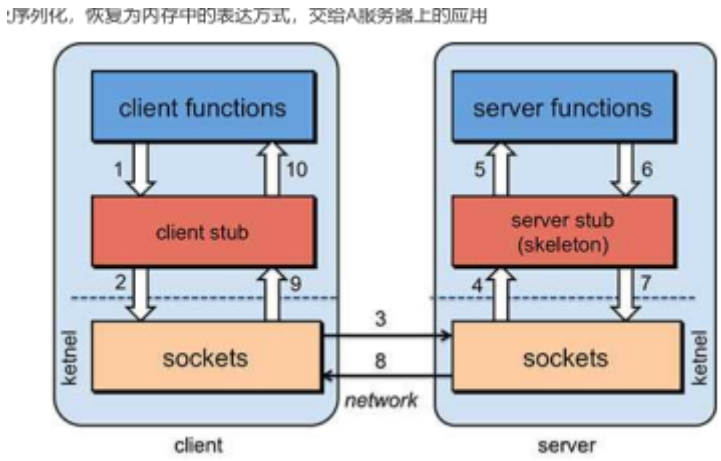
2.其次就是RPC框架一般都有注册中心，有丰富的监控管理；发布、下线接口、动态扩展等，对调用方来说是无感知、统一化的操作。

3.安全性，安全性高，虽然RPC的复杂度高于HTTP接口,但是HTTP接口由于受限于HTTP协议，需要带HTTP请求头，导致传输起来效率或者说安全性不如RPC。

4.最后就是流行的服务化架构、服务化治理，RPC框架是一个强力的支撑。RPC是一种概念，http也是RPC实现的一种方式，用http交互其实就已经属于RPC了。

5.他还有灵活部署，解耦的优点

6. RPC流程



首先，要解决通讯的问题，主要是通过在客户端和服务器之间建立TCP连接，远程过程调用的所有交换的数据都在这个连接里传输。连接可以是按需连接，调用结束后就断掉，也可以是长连接，多个远程过程调用共享同一个连接。

第二，要解决寻址的问题，也就是说，A服务器上的应用怎么告诉底层的RPC框架，如何连接到B服务器（如主机或IP地址）以及特定的端口，方法的名称是什么，这样才能完成调用。

第三，当A服务器上的应用发起远程过程调用时，方法的参数需要通过底层的网络协议如TCP传递到B服务器，由于网络协议是基于二进制的，内存中的参数的值要序列化成二进制的形式，也就是序列化（Serialize）或编组（marshal），通过寻址和传输将序列化的二进制发送给B服务器。

第四，B服务器收到请求后，需要对参数进行反序列化（序列化的逆操作），恢复为内存中的表达方式，然后找到对应的方法（寻址的一部分）进行本地调用，然后得到返回值。

第五，返回值还要发送回服务器A上的应用，也要经过序列化的方式发送，服务器A接到后，再反序列化，恢复为内存中的表达方式，交给A服务器上的应用

7. RPC协议

RPC开始是出现在UNIX操作系统的计算机中。他是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。 所以RPC的实现可以通过不同的协议去实现比如可以使http、RMI等。RPC的核心并不在于使用什么协议。RPC的目的是让你在本地调用远程的方法，这个调用是透明的，你并不知道这个调用的方法是部署哪里。通过RPC能解耦服务，是服务耦合性降低，这才是使用RPC的真正目的。RPC的原理主要用到了动态代理模式，RPC是一个软件结构概念，是构建分布式应用的理论基础。就好比为啥你家可以用到发电厂发出来的电？是因为电是可以传输的。至于用铜线还是用铁丝还是其他种类的导线，也就是用http还是用其他协议的问题了。这个要看什么场景，对性能要求怎么样。在OSI网络通信模型中，RPC跨越了传输层和应用层。RPC使得开发包括网络分布式多程序在内的应用程序更加容易。

8. RPC采用的模式

RPC采用客户机/服务器模式。请求程序就是一个客户机，而服务提供程序就是一个服务器。第一步，客户机调用进程发送一个调用信息到服务进程，然后等待应答信息。在服务器端，进程保持睡眠状态直到调用信息到达为止。当一个调用信息到达，服务器获得进程参数，计算结果，并发送答复信息，然后等待下一个调用信息。最后，客户端调用进程接收到答复信息，获得进程结果后，继续执行并进行下一步。RPC框架有很多：比如 JAVA RMI 、 Dubbo 、 grpc 等。

9. 影响RPC框架性能的因素

1.使用的网络IO模型：RPC服务器可以只支持传统的阻塞式同步IO，也可以做一些改进让RPC服务器支持非阻塞式同步IO，或者在服务器上实现对多路IO模型的支持。这样的RPC服务器的性能在高并发状态下，会有很大的差别。特别是单位处理性能下对内存、CPU资源的使用率。

2.基于的网络协议：一般来说您可以选择让您的RPC使用应用层协议，例如HTTP或者HTTP/2协议，或者使用TCP协议，让您的RPC框架工作在传输层。工作在哪一层网络上会对RPC框架的工作性能产生一定的影响，但是对RPC最终的性能影响并不大。但是至少从各种主流的RPC实现来看，没有采用UDP协议做为主要的传输协议的。

3.消息封装格式：选择或者定义一种消息格式的封装，要考虑的问题包括：消息的易读性、描述单位内容时的消息体大小、编码难度、解码难度、解决半包/粘包问题的难易度。当然如果您只是想定义一种RPC专用的消息格式，那么消息的易读性可能不是最需要考虑的。消息封装格式的设计是目前各种RPC框架性能差异的最重要原因，这就是为什么几乎所有主流的RPC框架都会设计私有的消息封装格式的原因。dubbo中消息体数据包含dubbo版本号、接口名称、接口版本、方法名称、参数类型列表、参数、附加信息

4.Schema 和序列化（Schema & Data Serialization）：序列化和反序列化，是对象到二进制数据的转换，程序是可以理解对象的，对象一般含有 schema 或者结构，基于这些语义来做特定的业务逻辑处理。

5.实现的服务处理管理方式：在高并发请求下，如何管理注册的服务也是一个性能影响点。您可以让RPC的Selector/Processor使用单个线程运行服务的具体实现（这意味着上一个客户端的请求没有处理完，下一个客户端的请求就需要等待）、您也可以为每一个RPC具体服务的实现开启一个独立的线程运行（可以一次处理多个请求，但是操作系统对于“可运行的最大线程数”是有限制的）、您也可以线程池来运行RPC具体的服务实现（目前看来，在单个服务节点的情况下，这种方式是比较好的）、您还可以通过注册代理的方式让多个服务节点来运行具体的RPC服务实现。