公司一般使用的分布式RPC框架及其原理面试

作者：有故事的人 来源：范文网 www.wtabcd.cn/fanwen/

本文原地址：https://www.wtabcd.cn/fanwen/zuowen/f7396136b8d8b8e7f08f3980a6b5f840.html

引言

以前在做一个规模不大的系统的时候，用的是单体架构，一台服务器部署上一个应用和数据库也就够了。

但是现代化互联网公司业务逐渐扩大，服务逐渐细分，很多服务之间需要通过远程分布式接口调年度工作总结开头用通讯，即不同的服务不是部署在同一个服务器上，比如订单服务在 a 服务上，付款服务在另一个服务上，有同步调用、也有异步调用，这个时候我们就需要远程调用不同的服务，使用的时候调用远程服务就像调用本地服务一样，引入一个 jar 包，就能通过this.xxx( ) 一样调用远程服务，这背后的机制就是通过 rpc 技术。

重点：rpc
技术一定是今后工作必备基础，熟练掌握其中一种，知道原理，阅读源码，甚至自己手写一个。

1、面试官：

公司使用什么 rpc 框架？，可以介绍一下 rpc 的工作原理吗？

问题分析：面试官想了解基础设施是否和我们项目用的一样，一样最好了，能直接上手，不一样了解其它一个别的应该也问题不大，毕竟原理技术都大同小异，说你最熟悉的一个。

答：rpc 是一个分布式计算的 cs 模式，总是由 client 向 server 发出一个执行若干过程请求，server 接受请求，使用者客户端提供的参数，计算完成之后将结果返回给客户端。

使用最广泛的 spring cloud，基于 spring boot
特性整合了开源行业中优秀的组件，整体对外提供了一套在微服务架构中服务治理的解决方案。

国内开源的框架中，使用比较广泛的有阿里的 dubbo，后来捐献给了 apache。还有腾讯的 tars 框架，还有 thrift 框架，也有基于 thrift 二次开发的 rpc 框架，比如美团的 mtthrift。

这些 rpc 大致原理基本都是一样的。（这个时候，跟面试官要纸和笔，画图解释 rpc 原理）

这个图既不显得太过复杂给自己挖坑，也不会显得简单潦草。

1-5 逐行解释：

服务集成 rpc 后，服务（这里的服务就是图中的 provider，服务提供者）启动后会通过 register（注册）模块，把服务的唯一 id 和 ip 地址，端口信息等注册到 rpc 框架注册中心（图中的 registry 部分）。当调用者（consumer）想要调用服务的时候，通过 provider 注册时的的服务唯一 id 去注册中心查找在线可供调用的服务，返回一个 ip 列表（3.notify 部分）。第三步 consumer 根据一定的策略，比如随机 or 轮训从 registry 返回的可草房子人物介绍用 ip 列表真正调用服务（4.invoke）。最后是统计功能，rpc 框架都提供监控功能，监控服务健康状况，控制服务线上扩展和上下线（5.count）

有清晰的流程图，有每一步的解释，面试官表示很满意，继续追加提问。

2、面试官：

服务启动的时候服务基本信息被注册到注册中心，如果服务提供者挂了，注册中心如何知道服务不可用了呢？

答：服务掉线分为主动下线和心跳检测

比如服务由于发版时，在重启之前先主动通知注册中心：我要重启了，有流量进来先不要分给我，让别的机器服务，等我重启成功后在放流量进来，或者是在管理后台手动直接摘掉机器，这个是主动下线。

心跳检测是处理服务非正常下线（如断电断网）的情况，这个时候如果注册中心不知道该服务已经掉线，一旦被其调用就会带来问题。为了避免出现这样的情况，注册中心增加一个心跳检测功能，它会对服务提供者（provider）进行心跳检测，比如每隔 30s 发送一个心跳，如果三次心跳结果都没有返回值，就认为该服务已下线，赶紧更新 consumer 的服务列表，告诉 圣诞礼物送什么consumer 调用别的机器。

问题分析：阐述了服务端挂了注册中心如何感知的问题，你以为此问题已经完事儿了？还没有，你成功给自己挖了个坑，面试官可能继续变更管理制度深挖，服务提供者（provider）挂了注册中心能解决，那注册中心自己就不挂了吗？三连问继续。

3、面试官：

如果注册中心挂了，比如你用的是 zookeeper，如果 zookeeper 挂了，那服务之间还能相互调用吗？

答：首先注册中心挂掉也要分两种情况，如果数据库挂了，zk 还是能用的，因为 zk 会缓存注册机列表在缓存里。

其次 zk 本身就是一个集群的，一台机器挂了，zk 会选举出集群中的其他机器作为 master 继续提供服务，如果整个集群都挂了也没问题，因为调用者本地会缓存注册中心获取的服务列表。省略和注册中心的交互，consumer 和 provider 采用直连方式，这些策略都是可配置的。

问题分析：面试是一个自由交流时间，任何一个点都可能被发散继续深入挖掘，刨根问题，总有你覆盖不到的知识盲区，目的不是为难你，是想了解你的技术沉淀深度。

4、面试官：

你对 rpc 了解的很透彻，那你能否自己写一个 rpc 框架？可以简答描述下思路也行。

答：这个问题，虽然没有自己动手写过，但是我阅读过源码，大致实现思路是这样的。（画图给面试官）

客户端 invoke 方法编写，使用 jdk 的动态代理技术，客户端调用远程服务方法时调用的是 invocationhandler 的 invoke 方法。客户端 filter 方法编写，完善的 rpc 框架少不了监控、路由、降级、鉴权等功能。创建 socket，在 filter 方法中实现 client.write 方法，其逻辑为从连接池（channelpool）中获取连接，然后将数据写进 channel。实现数据序列化、压缩，目的减少网络传输的数据量，向服务端发送 request 数据，这里可以使用 netty 异步通讯框架。服务端收到客户端发过的消息后，从 channel

中将消息读出来之前，也会先经反序列化解压。请求就到了服务端 filter 中。请求依次经过监控、鉴权方法。根据客户端传递来的服务信息和参数，通过反射调用相应的业务服务并拿到业务处理结果。然后在 responsefilter 中将返回结果写入
channel。服务端序列化、压缩等，发送给客户端。客户端收到消息后，经过客户端反序列化、解压缩，后交给 responsethreadpoolprocessor 线程池处理。responsethreadpoolprocessor
收到消息后，就将结果返回给之前的方法调用，整个调用请求就结束了。

面试官：可以可以，确实是看了，这个问题就到这。（面试官心理：虽然目前项目里不会让你真正去写一个 rpc 框架，知其然知其所以然，遇到这类 rpc 相关问题一定能搞定了，项目组正好缺一个这样的人）

深入分析

已经有 http 协议接口，或者说 restful 接口，为什么还要使用 rpc 技术？

在接口不多的情况下，使用 http 确实是一个明智的选择，比如在初创企业，我们不确定业务能顺利开展下去，可能面临随时倒闭，开发人员也不足，这个时候使用简洁高效的技术，先把东西做出来是最明智的选择，无需一步登天。

系统与系统交互较少的情况下，使用 http 协议优点显而易见：开发简单、测试也比较直接、部署方便，利用现成的 http 协议进行系统间通讯，如果业务真的慢慢做大，系统也慢慢扩大，rpc 框架的好处就显示出来 了，首先 rpc 支持长链接，通信不必每次都要像 http 一样去重复 3 次握手，减少了网络开销。

其次就是 rpc 框架一般都有注册中心模块，有完善的监控管理功能，服务注册发现、服务下线、服务动态扩展等都方便操作，服务化治理效率大大提高。

基于 tcp 协议实现的 rpc，能更灵活地对协议字段进行定制，相比 http 能减少网络传输字节数，降低网络开销（握手）提高性能。实现更大的吞吐量和并发数，但是需要更多的关注底层复杂的细节，
对开发人员的要求也高，增加开发成本。

总结

在面试官夺命三连问的攻击下，前三个题目一定要掌握，最后一个加分项徒手写 rpc
深入分析，可以点菜的英文大大拉升面试官对你的好感，只要有亮点，即使其他问题答得不好，那么问题也不大。

rpc 工作原理总结：

provider：服务提供方，cs 模型中的 server。

consumer： 调用远程服务服务消费方，cs 模型中的 client。

registry：服务注册与发现的服务管理中心。

monitor：统计服务的调用次数和调用时间的监控中心。

container：服务运行容器，如 jetty。

rpc 执行过程总结：

服务容器负责启动，加载，运行服务提供者。服务提供者在启动时，向注册中心注册自己提供的服务，暴露自己的 ip 和端口信息。服务消费者在启动时，向注册中心订阅自己所需的服务。注册中心返回服务提供者列表给消费者，如果有变更，注册中心将基于长连接推送给数据消费者。服务消费者，从提供这地址列表中，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，再选另外一台服务调用。服务消费者和提供者，在内存中累计调用次数和调用时间，定时发送一次统计数据到监控中心。

以上就是公司一般使用的分布式rpc框架及其原理的详细内容，更多关于分布式rpc框架原理的资料请关注www.887551.com其它相关文章！