1、面试题

为什么要进行系统拆分？如何进行系统拆分？拆分后不用dubbo可以吗？

2、面试官心里分析

从这个问题开始就进行分布式系统环节了，好多同学给我反馈说，现在出去分布式成标配了，没有哪个公司不问问你分布式的事儿。你要是不会分布式的东西，简直这简历没法看，没人会让你去面试。

其实为啥会这样呢？这就是因为整个大行业技术发展的原因

早些年，我印象中在2010年初的时候，整个IT行业，很少有人谈分布式，更不用说微服务，虽然很多BAT等大型公司，因为系统的复杂性，很早就是分布式架构，大量的服务，只不过微服务大多基于自己搞的一套框架来实现而已。

但是确实，那个年代，大家很重视ssh2，很多中小型公司几乎大部分都是玩儿struts2、spring、hibernate，稍晚一些，才进入了spring mvc、spring、mybatis的组合。那个时候整个行业的技术水平就是那样，当年oracle很火，oracle管理员很吃香，oracle性能优化啥的都是IT男的大杀招啊。连大数据都没人提，当年OCP、OCM等认证培训机构，火的不行。

但是确实随着时代的发展，慢慢的，很多公司开始接受分布式系统架构了，这里面尤为对行业有至关重要影响的，是阿里的dubbo，某种程度上而言，阿里在这里推动了行业技术的前进。

正是因为有阿里的dubbo，很多中小型公司才可以基于dubbo，来把系统拆分成很多的服务，每个人负责一个服务，大家的代码都没有冲突，服务可以自治，自己选用什么技术都可以，每次发布如果就改动一个服务那就上线一个服务好了，不用所有人一起联调，每次发布都是几十万行代码，甚至几百万行代码了。

直到今日，我很高兴的看到分布式系统都成行业面试标配了，任何一个普通的程序员都该掌握这个东西，其实这是行业的进步，也是所有IT码农的技术进步。所以既然分布式都成标配了，那么面试官当然会问了，因为很多公司现在都是分布式、微服务的架构，那面试官当然得考察考察你了。

3、友情提示

如果有个同学看到这里说，我天，我不知道啥是分布式系统？我也不知道啥是dubbo？那你赶紧百度啊，搜个dubbo入门，去里面体验一下。

分布式系统，我用一句话给你解释一下，实在没时间多唠了，就是原来20万行代码的系统，现在拆分成20个小系统，每个小系统1万行代码。原本代码之间直接就是基于spring调用，现在拆分开来了，20个小系统部署在不同的机器上，得基于dubbo搞一个rpc调用，接口与接口之间通过网络通信来请求和响应。就这个意思。

4、面试题剖析

（1）为什么要将系统进行拆分？

网上查查，答案极度零散和复杂，很琐碎，原因一大坨。但是我这里给大家直观的感受：

1）要是不拆分，一个大系统几十万行代码，20个人维护一份代码，简直是悲剧啊。代码经常改着改着就冲突了，各种代码冲突和合并要处理，非常耗费时间；经常我改动了我的代码，你调用了我，导致你的代码也得重新测试，麻烦的要死；然后每次发布都是几十万行代码的系统一起发布，大家得一起提心吊胆准备上线，几十万行代码的上线，可能每次上线都要做很多的检查，很多异常问题的处理，简直是又麻烦又痛苦；而且如果我现在打算把技术升级到最新的spring版本，还不行，因为这可能导致你的代码报错，我不敢随意乱改技术。

假设一个系统是20万行代码，其中小A在里面改了1000行代码，但是此时发布的时候是这个20万行代码的大系统一块儿发布。就意味着20万上代码在线上就可能出现各种变化，20个人，每个人都要紧张地等在电脑面前，上线之后，检查日志，看自己负责的那一块儿有没有什么问题。

小A就检查了自己负责的1万行代码对应的功能，确保ok就闪人了；结果不巧的是，小A上线的时候不小心修改了线上机器的某个配置，导致另外小B和小C负责的2万行代码对应的一些功能，出错了

几十个人负责维护一个几十万行代码的单块应用，每次上线，准备几个礼拜，上线 -> 部署 -> 检查自己负责的功能

最近从2013年到现在，5年的时间里，2013年以前，基本上都是BAT的天下；2013年开始，有几个小巨头开始快速的发展，上市，几百亿美金，估值都几百亿美金；2015年，出现了除了BAT以外，又有几个互联网行业的小巨头出现。

BAT工作，在市值几百亿美金的小巨头工作

有某一个小巨头，现在估值几百亿美金的小巨头，5年前刚开始搞的时候，核心的业务，几十个人，维护一个单块的应用

维护单块的应用，在从0到1的环节里，是很合适的，因为那个时候，是系统都没上线，没什么技术挑战，大家有条不紊的开发。ssh + mysql + tomcat，可能会部署几台机器吧。

结果不行了，后来系统上线了，业务快速发展，10万用户 -> 100万用户 -> 1000万用户 -> 上亿用户了。

2）拆分了以后，整个世界清爽了，几十万行代码的系统，拆分成20个服务，平均每个服务就1~2万行代码，每个服务部署到单独的机器上。20个工程，20个git代码仓库里，20个码农，每个人维护自己的那个服务就可以了，是自己独立的代码，跟别人没关系。再也没有代码冲突了，爽。每次就测试我自己的代码就可以了，爽。每次就发布我自己的一个小服务就可以了，爽。技术上想怎么升级就怎么升级，保持接口不变就可以了，爽。

所以简单来说，一句话总结，如果是那种代码量多达几十万行的中大型项目，团队里有几十个人，那么如果不拆分系统，开发效率极其低下，问题很多。但是拆分系统之后，每个人就负责自己的一小部分就好了，可以随便玩儿随便弄。分布式系统拆分之后，可以大幅度提升复杂系统大型团队的开发效率。

但是同时，也要提醒的一点是，系统拆分成分布式系统之后，大量的分布式系统面临的问题也是接踵而来，所以后面的问题都是在围绕分布式系统带来的复杂技术挑战在说。

（2）如何进行系统拆分？

这个问题说大可以很大，可以扯到领域驱动模型设计上去，说小了也很小，我不太想给大家太过于学术的说法，因为你也不可能背这个答案，过去了直接说吧。还是说的简单一点，大家自己到时候知道怎么回答就行了。

系统拆分分布式系统，拆成多个服务，拆成微服务的架构，拆很多轮的。上来一个架构师第一轮就给拆好了，第一轮；团队继续扩大，拆好的某个服务，刚开始是1个人维护1万行代码，后来业务系统越来越复杂，这个服务是10万行代码，5个人；第二轮，1个服务 -> 5个服务，每个服务2万行代码，每人负责一个服务

如果是多人维护一个服务，<=3个人维护这个服务；最理想的情况下，几十个人，1个人负责1个或2~3个服务；某个服务工作量变大了，代码量越来越多，某个同学，负责一个服务，代码量变成了10万行了，他自己不堪重负，他现在一个人拆开，5个服务，1个人顶着，负责5个人，接着招人，2个人，给那个同学带着，3个人负责5个服务，其中2个人每个人负责2个服务，1个人负责1个服务

我个人建议，一个服务的代码不要太多，1万行左右，两三万撑死了吧

大部分的系统，是要进行多轮拆分的，第一次拆分，可能就是将以前的多个模块该拆分开来了，比如说将电商系统拆分成订单系统、商品系统、采购系统、仓储系统、用户系统，等等吧。

但是后面可能每个系统又变得越来越复杂了，比如说采购系统里面又分成了供应商管理系统、采购单管理系统，订单系统又拆分成了购物车系统、价格系统、订单管理系统。

扯深了实在很深，所以这里先给大家举个例子，你自己感受一下，核心意思就是根据情况，先拆分一轮，后面如果系统更复杂了，可以继续分拆。你根据自己负责系统的例子，来考虑一下就好了。

（3）拆分后不用dubbo可以吗？

当然可以了，大不了最次，就是各个系统之间，直接基于spring mvc，就纯http接口互相通信呗，还能咋样。但是这个肯定是有问题的，因为http接口通信维护起来成本很高，你要考虑超时重试、负载均衡等等各种乱七八糟的问题，比如说你的订单系统调用商品系统，商品系统部署了5台机器，你怎么把请求均匀地甩给那5台机器？这不就是负载均衡？你要是都自己搞那是可以的，但是确实很痛苦。

所以dubbo说白了，是一种rpc框架，就是本地就是进行接口调用，但是dubbo会代理这个调用请求，跟远程机器网络通信，给你处理掉负载均衡了、服务实例上下线自动感知了、超时重试了，等等乱七八糟的问题。那你就不用自己做了，用dubbo就可以了。