19 | 分布式环境下如何快速定位问题?

2020-04-03 何小锋

讲述：张浩
时长 11:51 大小 10.87M

你好，我是何小锋。上一讲我们学习了如何建立可靠的安全体系，关键点就是“鉴权”，我们可以通过统一的鉴权服务动态生成秘钥，提高 RPC 调用的安全性。

回顾完上一讲的重点，我们就切入今天的主题，一起看看 RPC 在分布式环境下如何快速定位问题。重要性看字面也是不言而喻了，只有准确地定位问题，我们才能更好地解决问题。

分布式环境下定位问题有哪些困难?

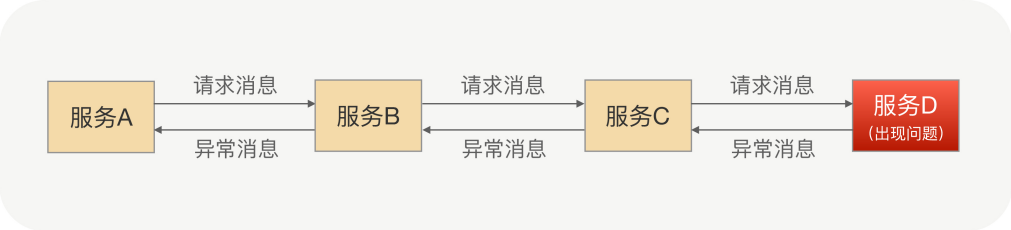
在此之前，我想先请你想想，在开发以及生产环境运行的过程中，如果遇见问题，我们是如何定位的?

在开发过程中遇见问题其实很好排查，我们可以用 IDE 在自己本地的开发环境中运行一遍代码，进行 debug，在这个过程中是很容易找到问题的。

那换到生产环境，代码在线上运行业务，我们是不能进行 debug 的，这时我们就可以通过打印日志来查看当前的异常日志，这也是最简单有效的一种方式了。事实上，大部分问题的定位我们也是这样做的。

那么如果是在分布式的生产环境中呢？比如下面这个场景：

我们搭建了一个分布式的应用系统，在这个应用系统中，我启动了 4 个子服务，分别是服务 A、服务 B、服务 C 与服务 D，而这 4 个服务的依赖关系是 A->B->C->D，而这些服务又都部署在不同的机器上。在 RPC 调用中，如果服务端的业务逻辑出现了异常，就会把异常抛回给调用端，那么如果现在这个调用链中有一个服务出现了异常，我们该如何定位问题呢？



服务异常

可能你的第一反应仍然是打印日志，好，那就打印日志吧。

假如这时我们发现服务 A 出现了异常，那这个异常有没有可能是因为 B 或 C 或 D 出现了异常抛回来的呢？当然很有可能。那我们怎么确定在整个应用系统中，是哪一个调用步骤出现的问题，以及是在这个步骤中的哪台机器出现的问题呢？我们该在哪台机器上打印日志？而且为了排查问题，如果要打印日志，我们就必须要修改代码，这样的话我们就得重新对服务进行上线。如果这几个服务又恰好是跨团队跨部门的呢？想想我们要面临的沟通成本吧。

所以你看，分布式环境下定位问题的难点就在于，各子应用、子服务间有着复杂的依赖关系，我们有时很难确定是哪个服务的哪个环节出现的问题。简单地通过日志排查问题，就要对每个子应用、子服务逐一进行排查，很难一步到位；若恰好再赶上跨团队跨部门，那不死也得去半条命了。

如何做到快速定位问题？

明白了难点，我们其实就可以有针对性地去攻克它了。有关 RPC 在分布式环境下如何快速定位问题，我给出两个方法，很实用。

方法 1：借助合理封装的异常信息

我们前面说是因为各子应用、子服务间复杂的依赖关系，所以通过日志难定位问题。那我们就想办法通过日志定位到是哪个子应用的子服务出现问题就行了。

其实，在 RPC 框架打印的异常信息中，是包括定位异常所需要的异常信息的，比如是哪类异常引起的问题（如序列化问题或网络超时问题），是调用端还是服务端出现的异常，调用端与服务端的 IP 是什么，以及服务接口与服务分组都是什么等等。具体如下图所示：

链路调用异常

这样的话，在 A->B->C->D 这个过程中，我们就可以很快地定位到是 C 服务出现了问题，服务接口是 com.demo.CSerivce，调用端 IP 是 192.168.1.2，服务端 IP 是 192.168.1.3，而出现问题的原因就是业务线程池满了。

由此可见，一款优秀的 RPC 框架要对异常进行详细地封装，还要对各类异常进行分类，每类异常都要有明确的异常标识码，并整理成一份简明的文档。使用方可以快速地通过异常标识码在文档中查阅，从而快速定位问题，找到原因；并且异常信息中要包含排查问题时所需要的重要信息，比如服务接口名、服务分组、调用端与服务端的 IP，以及产生异常的原因。总之就是，要让使用方在复杂的分布式应用系统中，根据异常信息快速地定位到问题。

以上是对于 RPC 框架本身的异常来说的，比如序列化异常、响应超时异常、连接异常等等。那服务端业务逻辑的异常呢？服务提供方提供的服务的业务逻辑也要封装自己的业务异常信息，从而让服务调用方也可以通过异常信息快速地定位到问题。

方法 2：借助分布式链路跟踪

无论是 RPC 框架本身，还是服务提供方提供的服务，只要对异常信息进行合理地封装，就可以让我们在分布式环境下定位问题变得更加容易。那这样是不是就满足我们定位问题的需求了呢？

我们还是回到前面提过的那个分布式场景：我们搭建了一个分布式的应用系统，它由 4 个子服务组成，4 个服务的依赖关系为 A->B->C->D。

假设这 4 个服务分别由来自不同部门的 4 个同事维护，在 A 调用 B 的时候，维护服务 A 的同事可能是不知道存在服务 C 和服务 D 的，对于服务 A 来说，它的下游服务只有 B 服务，那这时如果服务 C 或服务 D 出现异常，最终在整个链路中将异常抛给 A 了呢？

在这种情况下维护服务 A 的同事该如何定位问题呢？

因为对于 A 来说，它可能是不知道下游存在服务 C 和服务 D 的，所以维护服务 A 的同事会直接联系维护服务 B 的同事，之后维护服务 B 的同事会继续联系下游服务的服务提供方，直到找到问题。可这样做成本很高啊！

现在我们换个思路，其实我们只要知道整个请求的调用链路就可以了。服务 A 调用下游服务 B，服务 B 又调用了 B 依赖的下游服务，如果维护服务 A 的同事能清楚地知道整个调用链路，并且能准确地发现在整个调用链路中是哪个环节出现了问题，那就好了。

这就好比我们收发快递，我们可以在平台上看到快递配送的轨迹，实时获知快递在何时到达了哪个站点，这样当我们没有准时地收到快递时，我们马上就能知道快递是在哪里延误了。

在分布式环境下，要想知道服务调用的整个链路，我们可以用“分布式链路跟踪”。

先介绍下分布式链路跟踪系统。从字面上理解，分布式链路跟踪就是将一次分布式请求还原为一个完整的调用链路，我们可以在整个调用链路中跟踪到这一次分布式请求的每一个环节的调用情况，比如调用是否成功，返回什么异常，调用的哪个服务节点以及请求耗时等等。

这样如果我们发现服务调用出现问题，通过这个方法，我们就能快速定位问题，哪怕是多个部门合作，也可以一步到位。

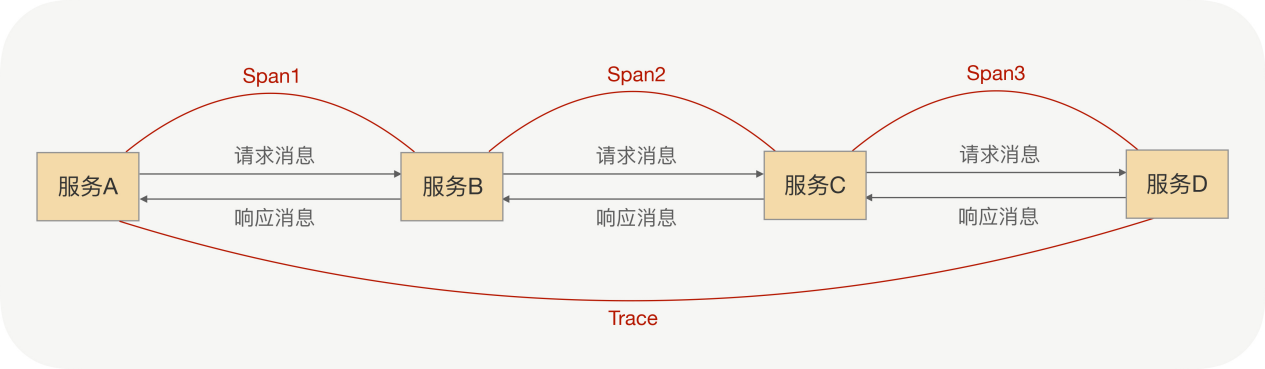
紧接着，我们再看看在 RPC 框架中是如何整合分布式链路跟踪的?

分布式链路跟踪有 Trace 与 Span 的概念，什么意思呢，我逐一解释。

Trace 就是代表整个链路，每次分布式都会产生一个 Trace，每个 Trace 都有它的唯一标识即 TraceId，在分布式链路跟踪系统中，就是通过 TraceId 来区分每个 Trace 的。

Span 就是代表了整个链路中的一段链路，也就是说 Trace 是由多个 Span 组成的。在一个 Trace 下，每个 Span 也都有它的唯一标识 SpanId，而 Span 是存在父子关系的。还是以讲过的例子为例子，在 A->B->C->D 的情况下，在整个调用链中，正常情况下会产生 3 个 Span，分别是 Span1（A->B）、Span2（B->C）、Span3（C->D），这时 Span3 的父 Span 就是 Span2，而 Span2 的父 Span 就是 Span1。

Trace 与 Span 的关系如下图所示：



示意图

分布式链路跟踪系统的实现方式有很多，但它们都脱离不开我刚才说的 Trace 和 Span，这两点可以说非常重要，掌握了这两个概念，其实你就掌握了大部分实现方式的原理。接着我们看看在 RPC 框架中如何利用这两个概念去整合分布式链路跟踪。

RPC 在整合分布式链路跟踪需要做的最核心的两件事就是“埋点”和“传递”。

所谓“埋点”就是说，分布式链路跟踪系统要想获得一次分布式调用的完整的链路信息，就必须对这次分布式调用进行数据采集，而采集这些数据的方法就是通过 RPC 框架对分布式链路跟踪进行埋点。

RPC 调用端在访问服务端时，在发送请求消息前会触发分布式跟踪埋点，在接收到服务端响应时，也会触发分布式跟踪埋点，并且在服务端也会有类似的埋点。这些埋点最终可以记录一个完整的 Span，而这个链路的源头会记录一个完整的 Trace，最终 Trace 信息会被上报给分布式链路跟踪系统。

那所谓“传递”就是指，上游调用端将 Trace 信息与父 Span 信息传递给下游服务的服务端，由下游触发埋点，对这些信息进行处理，在分布式链路跟踪系统中，每个子 Span 都存有父 Span 的相关信息以及 Trace 的相关信息。

总结

今天我们讲解了在分布式环境下如何快速定位问题。这里面的难点就是分布式系统有着较为复杂的依赖关系，我们很难判断出是哪个环节出现的问题，而且在大型的分布式系统中，往往会有跨部门、跨团队合作的情况，在排查问题的时候会面临非常高的沟通成本。

为了在分布式环境下能够快速地定位问题，RPC 框架应该对框架自身的异常进行详细地封装，每类异常都要有明确的异常标识码，并将其整理成一份简明的文档，异常信息中要尽量包含服务接口名、服务分组、调用端与服务端的 IP，以及产生异常的原因等信息，这样对于使用方来说就非常便捷了。

另外，服务提供方在提供服务时也要对异常进行封装，以方便上游排查问题。

在分布式环境下，我们可以通过分布式链路跟踪来快速定位问题，尤其是在多个部门的合作中，这样做可以一步到位，减少排查问题的时间，降低沟通成本，以最高的效率解决实际问题。

课后思考

在分布式环境下，你还知道哪些快速定位问题的方法？

期待你能在留言区中和我分享，也欢迎你把文章分享给你的朋友，邀请他加入学习，共同探讨。我们下节课再见！

上一篇　18 | 安全体系：如何建立可靠的安全体系?

下一篇　20 | 详解时钟轮在RPC中的应用

精选留言 (4)

写留言

Darren
2020-04-03

老师应该介绍下skywalking、Zipkin等开源分布式链路跟踪的优缺点和使用问题等，毕竟能自研分布式链路跟踪的公司不多

展开

3

陈国林
2020-04-05

老师好，说下自己定位问题的一些经验和方法
1. 确认清楚问题的现象或本质
2. 如果时间允许可以复现下问题，对问题理解更直观

3. 查看日志，确认报错的异常信息（日志这步很关键，物理机时代通常做法是上机器grep，如果有多台机器这是比较麻烦的点，所以一般会有日志中心。容器时代的话屏蔽了la...

展开

1

Jxin

2020-04-03

1.tracid在碰到线程池异步时如何传递id?
2.mq消息是否也该带上tracid?

2 1

嬰

2020-04-05

分布式链路系统中，最常用的调试问题方法还是文中提到的方法，主要依赖分布式调用监控系统，比如CAT，从里面可以看到是哪个链路出问题了。

展开