讲堂 □ 从0开始学微服务 □ 文章详情

17 | 如何识别服务节点是否存活?

2018-09-29 胡忠想

17 | 如何识别服务节点是否存活?
朗读人：胡忠想　08′12″ | 3.76M

今天我要与你分享如何识别服务节点是否存活，这在服务治理中是十分重要的。在进入正题之前，你可以先复习一下专栏第 5 期，我在讲解注册中心原理的时候，以开源注册中心 ZooKeeper 为例，描述了它是如何管理注册到注册中心的节点的存活的。

其实 ZooKeeper 判断注册中心节点存活的机制其实就是注册中心摘除机制，服务消费者以注册中心中的数据为准，当服务端节点有变更时，注册中心就会把变更通知给服务消费者，服务消费者就会调用注册中心来拉取最新的节点信息。

这种机制在大部分情况下都可以工作得很好，但是在网络频繁抖动时，服务提供者向注册中心汇报心跳信息可能会失败，如果在规定的时间内，注册中心都没有收到服务提供者的心跳信息，就会把这个节点从可用节点列表中移除。更糟糕的是，在服务池拥有上百个节点的的时候，每个节点都可能会被移除，导致注册中心可用节点的状态一直在变化，这个时候应该如何处理呢?

下面就结合我在实践中的经验，给你讲解几种解决方案。

心跳开关保护机制

在网络频繁抖动的情况下，注册中心中可用的节点会不断变化，这时候服务消费者会频繁收到服务提供者节点变更的信息，于是就不断地请求注册中心来拉取最新的可用服务节点信息。当有成百上千个服务消费者，同时请求注册中心获取最新的服务提供者的节点信息时，可能会把注册中心的带宽给占满，尤其是注册中心是百兆网卡的情况下。

所以针对这种情况，需要一种保护机制，即使在网络频繁抖动的时候，服务消费者也不至于同时去请求注册中心获取最新的服务节点信息。

我曾经就遇到过这种情况，一个可行的解决方案就是给注册中心设置一个开关，当开关打开时，即使网络频繁抖动，注册中心也不会通知所有的服务消费者有服务节点信息变更，比如只给 10% 的服务消费者返回变更，这样的话就能将注册中心的请求量减少到原来的 1/10。

当然打开这个开关也是有一定代价的，它会导致服务消费者感知最新的服务节点信息延迟，原先可能在 10s 内就能感知到服务提供者节点信息的变更，现在可能会延迟到几分钟，所以在网络正常的情况下，开关并不适合打开；可以作为一个紧急措施，在网络频繁抖动的时候，才打开这个开关。

服务节点摘除保护机制

服务提供者在进程启动时，会注册服务到注册中心，并每隔一段时间，汇报心跳给注册中心，以标识自己的存活状态。如果隔了一段固定时间后，服务提供者仍然没有汇报心跳给注册中心，注册中心就会认为该节点已经处于“dead”状态，于是从服务的可用节点信息中移除出去。

如果遇到网络问题，大批服务提供者节点汇报给注册中心的心跳信息都可能会传达失败，注册中心就会把它们都从可用节点列表中移除出去，造成剩下的可用节点难以承受所有的调用，引起“雪崩”。但是这种情况下，可能大部分服务提供者节点是可用的，仅仅因为网络原因无法汇报心跳给注册中心就被“无情”的摘除了。

这个时候就需要根据实际业务的情况，设定一个阈值比例，即使遇到刚才说的这种情况，注册中心也不能摘除超过这个阈值比例的节点。

这个阈值比例可以根据实际业务的冗余度来确定，我通常会把这个比例设定在 20%，就是说注册中心不能摘除超过 20% 的节点。因为大部分情况下，节点的变化不会这么频繁，只有在网络抖动或者业务明确要下线大批量节点的情况下才有可能发生。而业务明确要下线大批量节点的情况是可以预知的，这种情况下可以关闭阈值保护；而正常情况下，应该打开阈值保护，以防止网络抖动时，大批量可用的服务节点被摘除。

讲到这里，我们先小结一下。

心跳开关保护机制，是为了防止服务提供者节点频繁变更导致的服务消费者同时去注册中心获取最新服务节点信息；服务节点摘除保护机制，是为了防止服务提供者节点被大量摘除引起服务消费者可以调用的节点不足。

可见，无论是心跳开关保护机制还是服务节点摘除保护机制，都是因为注册中心里的节点信息是随时可能发生变化的，所以也可以把注册中心叫作动态注册中心。

那么是不是可以换个思路，服务消费者并不严格以注册中心中的服务节点信息为准，而是更多的以服务消费者实际调用信息来判断服务提供者节点是否可用。这就是下面我要讲的静态注册中心。

静态注册中心

前面讲过心跳机制能保证在服务提供者出现异常时，注册中心可以及时把不可用的服务提供者从可用节点列表中移除出去，正常情况下这是个很好的机制。

但是仔细思考一下，为什么不把这种心跳机制直接用在服务消费者端呢？

因为服务提供者是向服务消费者提供服务的，是否可用服务消费者应该比注册中心更清楚，因此可以直接在服务消费者端根据调用服务提供者是否成功来判定服务提供者是否可用。如果服务消费者调用某一个服务提供者节点连续失败超过一定次数，可以在本地内存中将这个节点标记为不可用。并且每隔一段固定时间，服务消费者都要向标记为不可用的节点发起保活探测，如果探测成功了，就将标记为不可用的节点再恢复为可用状态，重新发起调用。

这样的话，服务提供者节点就不需要向注册中心汇报心跳信息，注册中心中的服务节点信息也不会动态变化，也可以称之为静态注册中心。

从我的实践经历来看，一开始采用了动态注册中心，后来考虑到网络的复杂性，心跳机制不一定是可靠的，而后开始改为采用服务消费者端的保活机制，事实证明这种机制足以应对网络频繁抖动等复杂的场景。

当然静态注册中心中的服务节点信息并不是一直不变，当在业务上线或者运维人工增加或者删除服务节点这种预先感知的情况下，还是有必要去修改注册中心中的服务节点信息。

比如在业务上线过程中，需要把正在部署的服务节点从注册中心中移除，等到服务部署完毕，完全可用的时候，再加入到注册中心。还有就是在业务新增或者下线服务节点的时候，需要调用注册中心提供的接口，添加节点信息或者删除节点。这个时候静态注册中心有点退化到配置中心的意思，只不过这个时候配置中心里存储的不是某一项配置，而是某个服务的可用节点信息。

总结

今天我给你讲解了动态注册中心在实际线上业务运行时，如果遇到网络不可靠等因素，可能会带来的两个问题，一个是服务消费者同时并发访问注册中心获取最新服务信息导致注册中心带宽被打满；另一个是服务提供者节点被大量摘除导致服务消费者没有足够的节点可以调用。

这两个问题都是我在业务实践过程中遇到过的，我给出的两个解决方案：心跳开关保护机制和服务节点摘除保护机制都是在实践中应用过的，并且被证明是行之有效的。

而静态注册中心的思路，是在斟酌注册中心的本质之后，引入的另外一个解决方案，相比于动态注册中心更加简单，并且基于服务消费者本身调用来判断服务节点是否可用，更加直接也更加准确，尤其在注册中心或者网络出现问题的时候，这种方案基本不受影响。

思考题

在实际的微服务架构中，注册中心主动心跳机制和客户端摘除机制可能会同时使用，比如 Spring Cloud 就把这两种机制结合起来识别服务节点是否存活。如果注册中心没有收到某一个服务节点的心跳汇报，而服务消费者又调用这个服务节点成功了，你认为应该以哪个为准？为什么？

欢迎你在留言区写下自己的思考，与我一起讨论。

写留言

精选留言

拉欧 0

当然要以服务消费端为准，服务消费者才是真实的调用方

2018-09-29

oddrock 0

应该以客服端为准，谁使用谁有发言权。

我觉得注册中心的主动心跳探测不应作为客户端摘除不可用服务端节点的依据，而是要作为参考值，比如将这种探测信息发给客户端，客户端在负载均衡时，将这种疑似问题节点降权重或作为备用节点，这样是不是比较好

2018-09-29

feimeng0532 0

开关或者保护，怎么检测到网络抖动?
2018-09-29