https://mp.weixin.qq.com/s/e3GbPu81XdiIjW4MjolHmQ

# 对于注册中心，ZooKeeper、Eureka 哪个更合适？

* 简介
* 对比
* Zookeeper
* Eureka
* 总结

# **简介**

Eureka本身是Netflix开源的一款提供服务注册和发现的产品，并且提供了相应的Java封装。在它的实现中，节点之间相互平等，部分注册中心的节点挂掉也不会对集群造成影响，即使集群只剩一个节点存活，也可以正常提供发现服务。哪怕是所有的服务注册节点都挂了，Eureka Clients（客户端）上也会缓存服务调用的信息。这就保证了我们微服务之间的互相调用足够健壮。

Zookeeper主要为大型分布式计算提供开源的分布式配置服务、同步服务和命名注册。曾经是Hadoop项目中的一个子项目，用来控制集群中的数据，目前已升级为独立的顶级项目。很多场景下也用它作为Service发现服务解决方案。

# **对比**

在分布式系统中有个著名的CAP定理（C-数据一致性；A-服务可用性；P-服务对网络分区故障的容错性，这三个特性在任何分布式系统中不能同时满足，最多同时满足两个）；

# **Zookeeper**

Zookeeper是基于CP来设计的，即任何时刻对Zookeeper的访问请求能得到一致的数据结果，同时系统对网络分割具备容错性，但是它不能保证每次服务请求的可用性。从实际情况来分析，在使用Zookeeper获取服务列表时，如果zookeeper正在选主，或者Zookeeper集群中半数以上机器不可用，那么将无法获得数据。所以说，Zookeeper不能保证服务可用性。

诚然，在大多数分布式环境中，尤其是涉及到数据存储的场景，数据一致性应该是首先被保证的，这也是zookeeper设计成CP的原因。但是对于服务发现场景来说，情况就不太一样了：针对同一个服务，即使注册中心的不同节点保存的服务提供者信息不尽相同，也并不会造成灾难性的后果。**因为对于服务消费者来说，能消费才是最重要的——拿到可能不正确的服务实例信息后尝试消费一下，也好过因为无法获取实例信息而不去消费。** （尝试一下可以快速失败，之后可以更新配置并重试）所以，对于服务发现而言，可用性比数据一致性更加重要——AP胜过CP。

# **Eureka**

而Spring Cloud Netflix在设计Eureka时遵守的就是AP原则。Eureka Server也可以运行多个实例来构建集群，解决单点问题，但不同于ZooKeeper的选举leader的过程，Eureka Server采用的是Peer to Peer对等通信。这是一种去中心化的架构，无master/slave区分，每一个Peer都是对等的。在这种架构中，节点通过彼此互相注册来提高可用性，每个节点需要添加一个或多个有效的serviceUrl指向其他节点。每个节点都可被视为其他节点的副本。

如果某台Eureka Server宕机，Eureka Client的请求会自动切换到新的Eureka Server节点，当宕机的服务器重新恢复后，Eureka会再次将其纳入到服务器集群管理之中。当节点开始接受客户端请求时，所有的操作都会进行replicateToPeer（节点间复制）操作，将请求复制到其他Eureka Server当前所知的所有节点中。

一个新的Eureka Server节点启动后，会首先尝试从邻近节点获取所有实例注册表信息，完成初始化。Eureka Server通过getEurekaServiceUrls()方法获取所有的节点，并且会通过心跳续约的方式定期更新。默认配置下，如果Eureka Server在一定时间内没有接收到某个服务实例的心跳，Eureka Server将会注销该实例（默认为90秒，通过eureka.instance.lease-expiration-duration-in-seconds配置）。当Eureka Server节点在短时间内丢失过多的心跳时（比如发生了网络分区故障），那么这个节点就会进入自我保护模式。

什么是自我保护模式？默认配置下，如果Eureka Server每分钟收到心跳续约的数量低于一个阈值（instance的数量(60/每个instance的心跳间隔秒数)自我保护系数），并且持续15分钟，就会触发自我保护。在自我保护模式中，Eureka Server会保护服务注册表中的信息，不再注销任何服务实例。当它收到的心跳数重新恢复到阈值以上时，该Eureka Server节点就会自动退出自我保护模式。它的设计哲学前面提到过，那就是宁可保留错误的服务注册信息，也不盲目注销任何可能健康的服务实例。该模式可以通过eureka.server.enable-self-preservation = false来禁用，同时eureka.instance.lease-renewal-interval-in-seconds可以用来更改心跳间隔，eureka.server.renewal-percent-threshold可以用来修改自我保护系数（默认0.85）。

# **总结**

ZooKeeper基于CP，不保证高可用，如果zookeeper正在选主，或者Zookeeper集群中半数以上机器不可用，那么将无法获得数据。Eureka基于AP，能保证高可用，即使所有机器都挂了，也能拿到本地缓存的数据。作为注册中心，其实配置是不经常变动的，只有发版和机器出故障时会变。对于不经常变动的配置来说，CP是不合适的，而AP在遇到问题时可以用牺牲一致性来保证可用性，既返回旧数据，缓存数据。

所以理论上Eureka是更适合做注册中心。而现实环境中大部分项目可能会使用ZooKeeper，那是因为集群不够大，并且基本不会遇到用做注册中心的机器一半以上都挂了的情况。所以实际上也没什么大问题。