在分布式系统的高可用设计中，负载均衡非常关键，我们知道，分布式系统的特性之一就是支持快速扩展，那么集群扩展之后，服务请求如何从服务器列表中选择合适的一台呢？这就需要依赖负载均衡策略。

负载均衡在处理高并发，缓解网络压力，以及支持扩容等方面非常关键，在不同的应用场景下，可以选择不同的负载均衡，下面一起来看一下负载均衡相关的知识。

负载均衡的应用

负载均衡是指如何将网络请求派发到集群中的一个或多个节点上处理，一般来说，传统的负载均衡可以分为硬件负载均衡和软件负载均衡。

- 硬件负载均衡，就是通过专门的硬件来实现负载均衡，比如常见的 F5 设备。

- 软件负载均衡则是通过负载均衡软件实现，常见的就是 Nginx。

无论是硬件负载均衡还是软件负载均衡，实现原理都是一样的，在负载均衡中会记录一个可用的服务列表，负载均衡服务器会通过心跳机制来确认服务可用性，在网络请求到达后，F5 或者 Nginx 等负载均衡设备，会按照不同的策略，进行服务器的路由，这就是负载均衡的流程。

负载均衡的应用非常广，这一课时我们主要关注在分布式系统的请求调用，服务分发中的负载均衡。

常见的复杂均衡策略

一般而言，有以下几种常见的负载均衡策略。

轮询策略

轮询策略是最容易想到也是应用最广泛的负载均衡策略。轮询策略会顺序地从服务器列表中选择一个节点，请求会均匀地落在各个服务器上。轮询适合各个节点性能接近，并且没有状态的情况，但是在实际开发中，不同节点之间性能往往很难相同，这时候就可以应用另一种**加权轮询策略**。

加权轮询

加权轮询是对轮询策略的优化，给每个节点添加不同的权重。举个简单的例子，在实际开发中通常使用数组的数据结构来实现轮询，比如现在我有 A、B、C 三个节点，就可以在数组中添加 1、2、3 的数据，分别对应三个节点。现在我进行一个加权调整，让 1、2、3 对应 A，4、5 对应 B、C，这时候继续进行轮询，不同节点的权重就有变化了。

随机策略

随机策略和轮询相似，从列表中随机的取一个。我们都学过概率论的课程，真正的随机是很难实现的，所以如果访问量不是很大，最好不要应用随机策略，可能会导致请求不均匀。

最小响应时间

这个主要是在一些对请求延时敏感的场景中，在进行路由时，会优先发送给响应时间最小的节点。

最小并发数策略

你可以对比最小响应时间，最小并发策略会记录当前时刻每个节点正在处理的事务数，在路由时选择并发最小的节点。最小并发策略可以比较好地反应服务器运行情况，适用于对系统负载较为敏感的场景。

除了这些，还有哈希策略等，另外，在第 35 课时中我们提到过一致性哈希，其实一致性哈希也是一种负载均衡策略，一致性哈希经常应用在数据服务的路由中。

负载均衡如何实现

在分布式服务调用中，根据负载均衡实现的位置不同，可以分为服务端负载均衡和客户端负载均衡。

- 在服务器端负载均衡中，请求先发送到负载均衡服务器，然后通过负载均衡算法，在众多可用的服务器之中选择一个来处理请求。

- 在客户端负载均衡中，不需要额外的负载均衡软件，客户端自己维护服务器地址列表，自己选择请求的地址，通过负载均衡算法将请求发送至该服务器。

相信你已经看到了，这两种负载均衡，最大的区别就是服务器列表维护的位置。

下面我们来看一下，服务端负载均衡和客户端负载均衡如何实现呢？

在分布式服务调用中，服务端负载均衡常用的组件是 Spring Cloud Eureka，如果你选择了 Dubbo 作为中间件，那么可以应用 Dubbo 内置的路由策略。

在 Spring Cloud 中开启负载均衡的方法很简单，有一个专门的注解 @LoadBalanced 注解，配置这个注解之后，客户端在发起请求的时候会选择一个服务端，向该服务端发起请求，实现负载均衡。另外一种客户端负载均衡，也有对应的实现，典型的是 Spring Cloud Ribbon。

Ribbon 实际上是一个实现了 HTTP 的网络客户端，内置负载均衡工具、支持多种容错等。

我们上面提到的几种策略，在 Ribbon 中都有提供，包括 RoundRobinRule 轮询策略、RandomRule 随机策略、BestAvailableRule 最大可用策略、WeightedResponseTimeRule 带有加权的轮询策略等。

如果你的应用需要比较复杂的负载均衡场景，推荐应用 Ribbon，本课时的目的是讲解负载均衡被实现的原理，你可以到 Ribbon 的官方仓库，去了解相关的应用。

总结

以上就是本课时的内容了，我和大家一起讨论了负载均衡的应用场景、常见负载均衡策略，以及服务端和客户端负载均衡实现组件。

现在我们来思考一个问题，为什么说分布式高可用设计中，负载均衡很关键呢？我们都知道，在分布式场景下，特别是微服务拆分后，不同业务系统之间是解耦的，负载均衡策略，也就是描述了各个应用之间如何联系。

我们用订单场景来举例子，下单时依赖商品服务，假设我们选择的是轮询策略，当某台商品服务器出现网络故障、服务超时，此时下单就会受影响，如果改为最小可用时间策略，订单服务就会自动进行故障转移，不去请求超时的节点，实现高可用。

在你的工作中，应用过哪些负载均衡策略呢，又是如何进行配置的，欢迎留言进行分享~

精选评论