

## 一文搞懂注册中心的设计与实现

=====

服务治理在微服务架构中扮演着至关重要的角色，它使得各个微服务能够自动完成注册和发现。本文将深入探讨实现服务治理的基本方法。

首先，设想我们正在构建一个分布式服务系统。在这种系统中，服务的数量可能非常庞大，并且服务之间需要相互沟通，形成错综复杂的调用路径。

面对众多服务，我们通常会遇到两个主要问题：

- \* 如何有效追踪服务实例的数量？
- \* 如何监控服务实例的当前状态？

当系统中服务数量激增，例如达到数十甚至数百个时，我们很难清楚地了解哪些服务正在运行。而且，由于自动扩展、服务重启等操作，服务实例的运行状态也会频繁变动。如下图所示：

![图片](https://p3-xtjj-sign.byteimg.com/tos-cn-i-73owjymdk6/68940242b1264d768a9d50a87d22de5a~tplv-73owjymdk6-watermark.image?rk3s=f64ab15b&x-expires=1722222299&x-signature=Hzx0%2FO%2BhxMsUWWZi0GMpr3%2B%2FDPw%3D)

为了更清晰地描述服务的运行状态，我们可以对每个服务实例进行抽象化处理，并采用统一且直观的方式来表达这些信息。如下所示：

![图片](https://p3-xtjj-sign.byteimg.com/tos-cn-i-73owjymdk6/b3a92d87eefd4f85b6c8f0f11c40d8f6~tplv-73owjymdk6-watermark.image?rk3s=f64ab15b&x-expires=1722222299&x-signature=rJmkGmCT9rFXcFSVO5LsLQXRR5U%3D)

但是，随着服务数量的增加和服务实例状态的不断变化，我们如何有效管理这些实例呢？这正是服务治理所要解决的问题。通常，为了实现高效的服务治理，我们会引入注册中心来管理服务实例。

## 什么是注册中心

---

注册中心是一个存储服务实例信息的仓库，同时也是服务提供者和消费者进行交流的枢纽。它主要提供两项核心功能：服务注册和服务发现。

![图片](https://p3-xtjj-sign.byteimg.com/tos-cn-i-73owjymdk6/650a46f987de4bc4aeba6511bb933695~tplv-73owjymdk6-watermark.image?rk3s=f64ab15b&x-expire=1722222299&x-signature=sOyprmE7CmwqkqnnX70Ne3sVsZ0%3D)

我们看这张服务注册流程图就知道，对于注册中心而言，服务的提供者和消费者都相当于是它的客户端，所以都内嵌了专门与注册中心实现交互的客户端组件。

服务提供者在启动时，会通过注册中心的客户端组件自动注册自己，这个过程也被称作服务发布。对于服务消费者而言，他们执行的是订阅操作，而非注册操作。通过订阅，消费者能够自动从注册中心获取已注册服务提供者的信息，这个过程就是服务发现。

我们还可以看到，服务消费者和提供者之间存在一个明显的区别：消费者拥有一个本地缓存，存储了他们获取到的服务提供者实例信息。

这个本地缓存有两个主要作用：一是提高服务发现效率，消费者可以通过查询本地缓存快速获取目标服务实例信息；二是在注册中心不可用或网络异常时，消费者依然可以基于本地缓存调用已注册的服务。

## 注册信息变更通知机制

---

讲到这里，我们实际上就已经了解了。通过获取注册中心中的服务实例信息，我们就可以掌握系统中服务的数量以及当前的运行时状态了。

但问题来了，一旦服务的运行状态发生变化，我们如何及时获取这些变更信息呢？这就需要在注册中心引入变更通知机制：

![图片](https://p3-xtjj-sign.byteimg.com/tos-cn-i-73owjymdk6/41c893079c3e46f8bd164ff97933a898~tplv-73owjymdk6-

watermark.image?rk3s=f64ab15b&x-expires=1722222299&x-signature=a%2Baj8yGie7gnt3egj1xcgCum0KI%3D)

变更通知机制是实现注册中心的一大难点，因为这个过程涉及服务提供者、消费者和注册中心三者之间的数据同步问题，想要在分布式环境下实现数据同步是有挑战的。接下来，我将介绍两种主流的实现方法：监听机制和轮询机制。

### ### 监听机制

从架构设计角度来看，状态变更管理可以利用注册中心的发布-订阅模式。因此也就诞生了服务监听机制。它确保服务消费者能够实时监控服务的更新状态，是一种被动接收变更通知的方案，通常采用监听器和回调机制。

![图片](https://p3-xtjj-sign.byteimg.com/tos-cn-i-73owjymdk6/4cc60c96e3dc4c0e8be99ac5a36e9e93~tplv-73owjymdk6-watermark.image?rk3s=f64ab15b&x-expires=1722222299&x-signature=LNOTmog1Un%2FoS7X61GyGO0%2F4sK0%3D)

服务消费者可以为具体的服务实例节点添加监听器。当这些节点发生变化时，例如服务 B 的第一个实例不可用、服务 C 的第一个实例地址变更，或服务 D 新增了一个实例 3，注册中心就会触发监听器中的回调函数，确保更新通知到每一个服务消费者。

### ### 轮询机制

另一种确保状态信息同步的方式是轮询机制，这是一种主动拉取策略。服务消费者会定期调用注册中心的服务获取接口，以获取最新的服务列表，并更新本地缓存。

![图片](https://p3-xtjj-sign.byteimg.com/tos-cn-i-73owjymdk6/fe168323e78649a3b1289de68777390f~tplv-73owjymdk6-watermark.image?rk3s=f64ab15b&x-expires=1722222299&x-signature=cTxftQch7oxDhD9G2a0k9SaSY5k%3D)

轮询机制实际上是一个定时器，我们需要考虑的主要问题是轮询频率。为了确保数据同步的及时性，轮询频率不能太短；但同时，考虑到轮询对注册中心性能的影响，也不能过于频繁。通常，将轮询频率控制在几十秒到几分钟之间是一个较好的选择。

## 注册中心实现

---

通过前面的分析，相信你对注册中心的实现原理有了全面的了解。注册中心本质上是一种架构模型。在开发过程中，为了避免重复劳动，我们通常不需要自己实现这一模型，而是可以采用业界的一些主流注册中心实现工具，如 Consul、Zookeeper、Eureka 和 Nacos。

Consul 由 HashiCorp 公司提供，主要用于分布式环境下的服务发现与配置；Zookeeper 是 Apache 的一个顶级项目，作为分布式协调领域的代表性框架，被广泛用于注册中心、配置中心和分布式锁等场景；Netflix 的 Eureka 采用了一种不同的实现方案，并集成到了微服务开发框架 Spring Cloud 中；Nacos 由阿里巴巴开发，是面向云原生应用的动态服务发现、配置和服务管理平台。

这些工具各有特点，都实现了注册中心的高可用性、服务实例存储和同步功能，并提供了方便集成的客户端组件。我们知道，注册中心主要应用于微服务系统，主流的微服务开发框架是 Dubbo 和 Spring Cloud，它们分别使用 Zookeeper 和 Eureka 作为默认的注册中心实现方案。

因此，接下来我们就重点探讨下这两款注册中心工具。

Zookeeper 是“服务监听机制”实现策略的典型代表，它本质上是一个树形结构，可以在树上创建临时节点，并对节点添加监听器。

临时节点的客户端与该节点建立长连接，并实时节点状态。客户端有一个回调函数，当节点状态变化时，通过监听器将变化传递到客户端并触发回调函数。如下图所示：

![图片](https://p3-xtjj-sign.byteimg.com/tos-cn-i-73owjymdk6/7d750b6528eb4fa6ba72a806d682ba6f~tplv-73owjymdk6-watermark.image?rk3s=f64ab15b&x-expire=1722222299&x-signature=ArRDGwVw%2BDmOGidK4j0LZ4IGloo%3D)

而对于 Netflix Eureka 而言，它采用的就是典型的“轮询机制”来实现服务实例状态的同步，如下所示：

![图片](https://p3-xtjj-sign.byteimg.com/tos-cn-i-73owjymdk6/98f48afa8ec94686a93250bf57f99ac9~tplv-73owjymdk6-

watermark.image?rk3s=f64ab15b&x-expire=1722222299&x-signature=HQJyRGaf2OdbDNbNkH10sl2YDxk%3D)

在 Eureka 中，客户端和服务端通过发送心跳来实现轮询机制。Eureka 有一个租约概念，服务提供者需要通过续约机制来确保注册中心中的服务实例状态得到更新。心跳的作用是完成续约操作。

一般来说，心跳频率是 30 秒，如果服务连续 90 秒没有发送心跳，Eureka 服务器会认为该服务失效，并更新其状态信息。这样，可以确保 Eureka 服务器中服务实例信息的正确性。

服务消费者也是通过轮询机制来获取服务提供者的实例信息，其默认轮询频率同样是 30 秒。

总结

--

你只需要记住，注册中心是一种服务治理工具，它可以管理所有服务实例的运行状态，并将这些状态的变化同步到各个服务中。在开发分布式系统时，通过引入注册中心，可以轻松实现对大规模服务的高效治理。

原文链接: <https://juejin.cn/post/7393893498434945058>