随着业务的发展，应用系统中的配置会越来越多，配置之间也有不同的业务特点，比如业务依赖的数据库配置、缓存信息配置、索引存储配置等。这类配置一般比较稳定，不会频繁更改，通常会放在工程中作为配置文件随应用一起发布。

除了这些配置，还有一部分配置会经常发生修改，比如限流降级开关配置、业务中的白名单配置等。这些配置项除了变更频繁，还要求实时性，如果采取和应用一起发布的方式，那么每次变更都要重新发布服务，非常不方便。

为了解决这类配置问题，出现了分布式配置管理平台，这一课时我们就来了解一下分布式配置管理相关的内容。

配置管理的应用场景

在项目开发中，数据库信息等配置管理，一般是随着工程一起上线的，比如 Java 的 Web 系统，习惯把数据库的配置信息放到 jdbc.properties 这个配置文件中。

在分布式场景下，配置管理的应用范围更加广泛。比如上面说的限流和降级配置，电商网站在举行大型促销活动时，由于访问人数暴增，为了保证核心交易链路的稳定性，会把一些不太重要的业务做降级处理，那么如何关闭非核心服务呢？就需要分布式配置管理系统，能够实时管理被降级的业务，保证系统安全。

在一些异步业务场景中，配置管理也广泛应用，比如工作中经常会有数据同步，需要控制同步的速度；在一些定时任务中，需要控制定时任务触发的时机，以及执行的时长等，这些都可以通过配置管理来实现。

配置管理如何实现

分布式配置管理的本质就是一种推送-订阅模式的运用。配置的应用方是订阅者，配置管理服务则是推送方，客户端发布数据到配置中心，配置中心把配置数据推送到订阅者。

配置管理服务往往会封装一个客户端，应用方则是基于该客户端与配置管理服务进行交互。在实际实现时，客户端可以主动拉取数据，也可以基于事件通知实现。

实现配置管理中心，一般需要下面几个步骤:

- 提取配置信息，放到一个公共的地方存储，比如文件系统、数据库、Redis；
- 使用发布/订阅模式，让子系统订阅这些配置信息；
- 对外开放可视化的配置管理中心，对配置信息进行操作维护。

分布式配置管理的特性要求

一个合格的分布式配置管理系统，除了配置发布和推送，还需要满足以下的特性:

- **高可用性**，服务器集群应该无单点故障，只要集群中还有存活的节点，就能提供服务；
- **容错性**，保证在配置平台不可用时，也不影响客户端的正常运行；
- **高性能**，对于配置平台，应该是尽可能低的性能开销，不能因为获取配置给应用带来不可接受的性能损耗；
- **可靠存储**，包括数据的备份容灾，一致性等，尽可能保证不丢失配置数据；
- **实时生效**，对于配置的变更，客户端应用能够及时感知。

可以看到，一个好的配置管理系统，不只是提供配置发布和推送就可以，还有许多高级特性的要求。

分布式配置中心选型

分布式配置管理系统可以选择自研，也可以选择开源组件，比如携程开源的 Apollo、淘宝的 Diamond、百度的 Disconf 等。

Diamond

淘宝的 Diamond 是国内比较早的配置管理组件，设计简单，配置信息会持久化到 MySQL 数据库和本地磁盘中，通过数据库加本地文件的方式来进行容灾。

客户端和服务端通过 Http 请求来交互，通过比较数据的 MD5 值感知数据变化。在运行中，客户端会定时检查配置是否发生变化，每次检查时，客户端将 MD5 传给服务端，服务端会比较传来的 MD5 和自身内存中的 MD5 是否相同。如果相同，则说明数据没变，返回一个标示数据不变的字符串给客户端；如果不同，则说明数据发生变更，返回变化数据的相关信息给客户端，客户端会重新请求更新后的配置文件。

Diamond 开源版本已经很久没有更新了，比较适合小型的业务系统的配置管理，源码规模也比较小，可以下载对应的源码来查看，下载地址为：github-diamond。

Disconf

Disconf 是百度的一款分布式配置管理平台，代码仓库地址为：knightliao-disconf。

Disconf 的实现是基于 ZooKeeper 的，应用安装需要依赖 ZooKeeper 环境，配置动态更新借助 ZooKeeper 的 watch 机制实现。在初始化流程会中会对配置文件注册 watch，这样当配置文件更新时，ZooKeeper 会通知到客户端，然后客户端再从 Disconf 服务端中获取最新的配置并更新到本地，这样就完成了配置动态更新。

关于 Disconf 的细节，可以查看作者提供的设计文档：
https://disconf.readthedocs.io/zh_CN/latest/design/index.html。

Apollo

Apollo 是携程开源的分布式配置中心，官方的描述是：Apollo 能够集中化管理应用不同环境、不同集群的配置。配置修改后能够实时推送到应用端，并且具备规范的权限、流程治理等特性，适用于微服务配置管理场景。

Apollo 服务端基于 Spring Boot 和 Spring Cloud 开发，打包后可以直接运行，不需要额外安装 Tomcat 等应用容器。Apollo 支持多种语言的客户端，包括 Java 和 .Net 客户端，客户端运行不需要依赖其他框架，对系统侵入较小。

相比 Diamond 和 Disconf，Apollo 一直保持着稳定的版本更新，开源社区也比较活跃，管理界面友好，适合大型的业务系统，比较推荐使用。可以在 Apollo的代码仓库 ctripcorp-apollo 中了解更多的信息。

除了以上几款组件，大家熟悉的 ZooKeeper 也经常被用作分布式配置管理，和 Disconf 的实现类似，是依赖 ZooKeeper 的发布订阅功能，基于 watch 机制实现。

总结

这一课时分享了分布式配置管理的应用，实现分布式配置管理应该考虑的一些问题，以及常用分布式配置管理的选型。

内容中介绍的配置管理选型都是单独提供配置管理功能的，其实在大部分业务系统中，配置管理都不是一个单独的功能，一般是和服务治理，或者网关集成在一起。比如 Spring Cloud Nacos，除了支持服务发现，还提供了配置管理的功能，Dubbo 的控制台 Dubbo Admin 也内置了服务配置推送的功能。

精选评论