翻译-服务注册与发现

原文地址http://microservices.io/patterns/service-registry.html，谷歌翻译（略微调整）如下。

背景

使用服务的客户端可以采取客户端发现（Client-side discovery）和服务端发现（Server-side discovery）两种方式进行服务的发现，那么我们如何做到这些呢？

考虑因素

一个服务的每个实例公开一个远程接口如HTTP/ REST、Thrift等。

服务实例数量和它们的位置是动态变化的。如：虚拟机和容器通常分配动态IP地址，一个EC2自动缩放集群可以调整基于负载实例的数量。

解决方案

实现服务注册，服务的实例和它们的位置保存在服务注册里。服务实例启动时进行注册，服务停止时进行注销。服务的使用客户或者路由器查询服务注册发现服务的可用实例。

示例

经常用作服务注册的技术包括：

l Eureka

l Apache Zookeeper

l Consul

l Etcd

一些系统，如Kubernetes，Marathon和AWS ELB有一个隐含的服务注册表（译注：很多传统应用使用LDAP存放服务注册表，当然还有数据库、缓存等，不过这些都只是存储服务注册表的方法，和监测服务状态的机制一起才能构建服务注册功能）。

解决方案背景

Service Registry模式的好处包括：

l 客户端或路由器可以发现服务实例的位置。

也有一些缺点：

除非服务注册表内置于基础设施，否则这又是一个基础设施组件，必须单独设置、配置和管理。此外，服务注册表是一个关键的系统组件。虽然客户端可以缓存注册表数据，但如果服务注册失败，缓存数据终将过时，因此，服务注册中心必须具有高可用性。

你需要决定如何服务实例注册到服务注册中心。有两种选择：

l 自注册模式 - 服务实例注册自己。

l 第三方注册模式 – 使用第三方进行注册。

服务注册的客户端需要知道服务注册表实例的位置（S）。服务注册表实例都必须部署在固定的和众所周知的IP地址，客户端被配置到这些地址。例如，Netflix的Eureka服务实例通常使用弹性IP地址发布。弹性IP地址的可用池使用属性文件或DNS来完成。当一个Eureka实例启动时，它会查询配置，以确定使用哪一个可用的弹性IP地址。Eureka的客户端也需要配置弹性IP地址池。

相关模式

- Client-side discovery 和 Server-side discovery 用户服务发现。
- Self registration 和3rd party registration 是两种服务注册方法。

自身注册服务

Self registration 解决方案部分。服务实例负责在服务注册中心注册自己。在启动服务实例将自己注册（主机和IP地址）到服务注册表，使自己可以发现。通常必须定期更新其客户端的注册，使注册表知道它仍然活着。在关闭时，服务实例从服务注册表中注销本身。Netflix的Eureka是一个服务注册中心的一个例子。它提供了一个注册API和一个客户端库，服务实例使用（UN）注册自己。当使用Apache Zookeeper作为服务注册，每个服务对应于特定Zookeeper znode。在启动时，每个服务实例创建服务znode的一个短暂的子znode。短暂znode包含实例的位置。服务的客户端只需检索服务znode的子节点就可以确定可用的实例。如果客户端终止，无需拆卸短暂的节点，Apache Zookeeper将超时客户端并删除znode（淘宝的dubbo使用该机制）。

第三方注册

3rd party registration解决方案部分。第三方注册管理器是负责注册和注销服务实例到服务注册表。当服务实例启动时，注册管理器注册服务到服务注册表，当服务实例关闭时，注册管理器从服务注册表中注销服务实例。使用第三方注册的例子如：

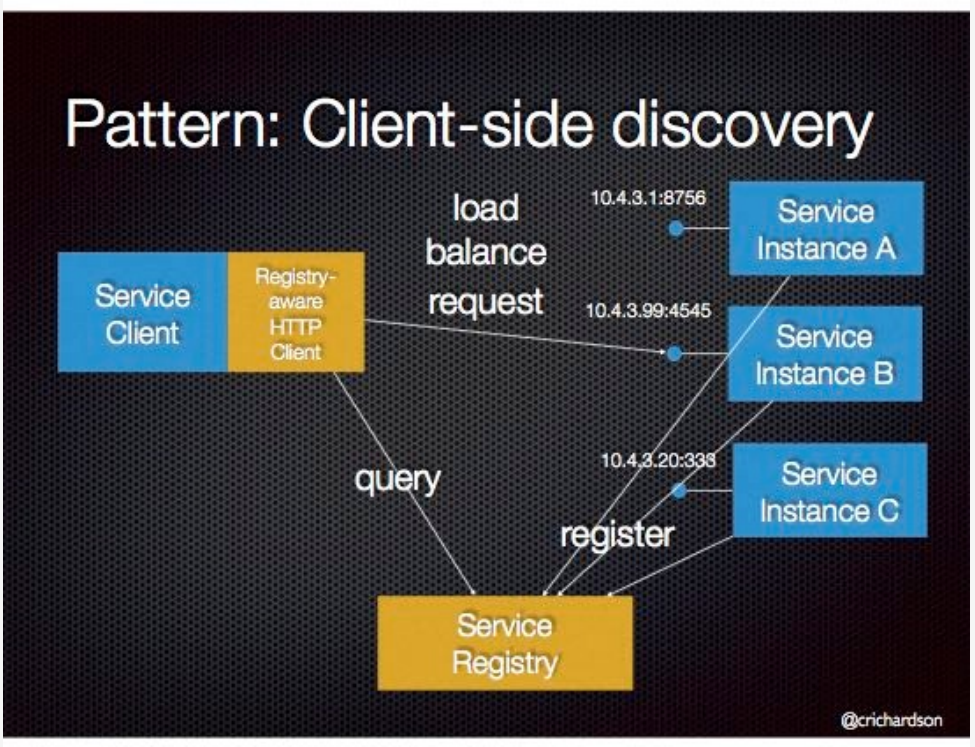
Netflix的Prana - 一个非JVM应用，以"side car"方式运行，注册服务到Eureka。

Registrator – 向服务中心Consul注册和注销Docker容器包含的各种服务。

集群框架（Docker集群），如Kubernetes和Marathon（UN）内置服务注册功能。

客户端服务发现

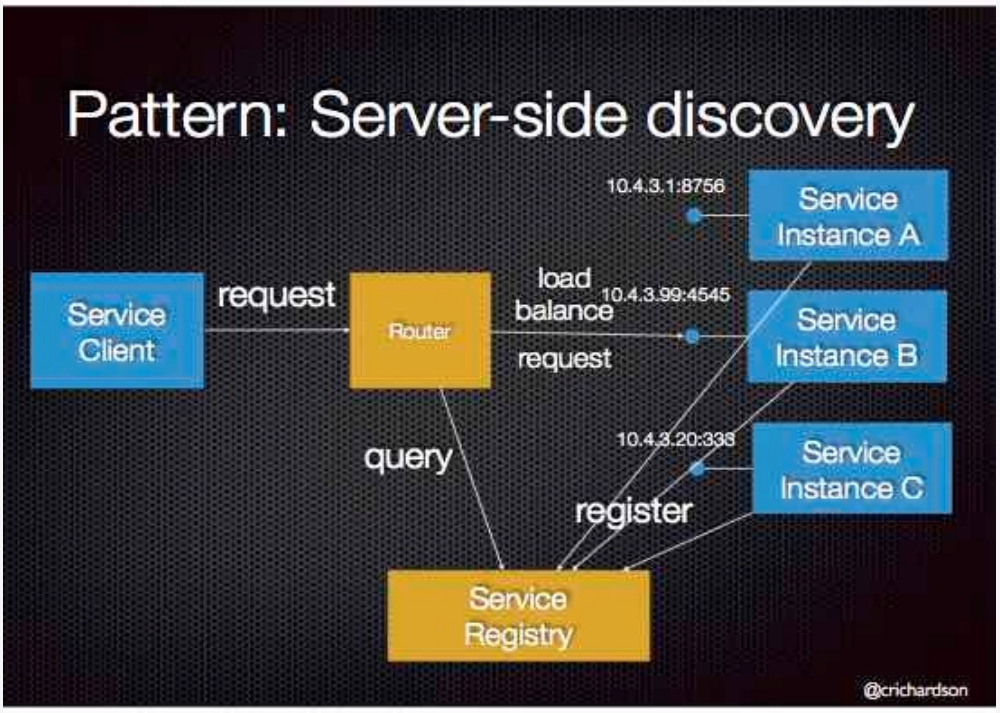
Client-side discovery方案解决方案部分。客户端请求服务时，客户端通过查询服务注册中心获得一个服务实例的位置。下图显示了这种模式的结构：



Netflix的OSS提供客户端发现一个很好的例子，Eureka提供服务注册，Ribbon Client是一个http client客户端通过查询Eureka获得可用的服务实例。客户端发现相比服务端发现减少了网络通讯并且更为简单。但是，相比服务端发现，你需要为每种编程语言/框架编写服务发现的逻辑实现，例如Java和Scala，JavaScript / Nodejs。例如，Netflix公司提供了一个HTTP代理为基础的非Java虚拟机客户服务发现方法。

服务端服务发现

Server-side discovery 解决方案部分。客户端请求服务时，客户端请求运行在一个公知位置的路由器（又名负载平衡器）。路由器再去查询服务注册，服务注册可能本身就被建立在路由器中，并且将可用服务实例转发给客户端。下图显示了这种模式的结构：



一个AWS弹性负载均衡器（ELB）是一个服务器端发现路由器的一个例子。客户端发出的HTTP（S）请求（或打开TCP连接）到ELB。一个ELB可以负载均衡的请求可以是来自互联网也可以是来自VPC，负载均衡的内部通信时。一个ELB可以用作服务注册。 EC2实例是通过调用一个API进行注册或自动成为集群的一部分。

一些集群解决方案，如Kubernetes和Marathon运行一个代理服务器作为作为服务器端发现的路由。为了访问服务，客户端连接到分配给该服务的本地代理，然后代理服务器将请求转发到集群中某处运行的服务实例。

分类: 微服务架构

标签: 微服务, 服务发现

« 上一篇：翻译-微服务API Gateway

» 下一篇：一句话概括下spring框架及spring cloud框架主要组件