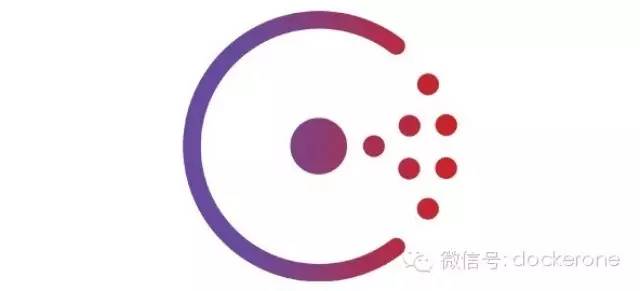
**Consul和ZooKeeper、Doozerd、Etcd的区别**

[Docker](http://www.wtoutiao.com/author/dockerone.html)(洪国安) · 2015-04-14 21:49



ZooKeeper、Doozerd、Etcd在架构上都非常相似，它们都有服务节点（server node），而这些服务节点的操作都要求达到节点的仲裁数（通常，节点的仲裁数遵循的是简单多数原则）。此外，它们都是强一致性的，并且提供各种原语。通过应用程序内部的客户端lib库，这些原语可以用来构建复杂的分布式系统。

Consul在一个单一的数据中心内部使用服务节点。在每个数据中心中，为了Consule能够运行，并且保持强一致性，Consul服务端需要仲裁。然而，Consul原生支持多数据中心，就像一个丰富gossip系统连接服务器节点和客户端一样。

当提供K/V存储的时候，这些系统具有大致相同的语义，读取是强一致性的，并且在面对网络分区的时候，为了保持一致性，读取的可用性是可以牺牲的。然而，当系统应用于复杂情况时，这种差异会变得更加明显。

这些系统提供的语义对开发人员构建服务发现系统很有吸引力，但更重要的是，强调开发人员要构建这些特性。ZooKeeper只提供一个原始的K/V值存储，并要求开发人员构建他们自己的系统来提供服务发现功能。相反的是，Consul提供了一个坚固的框架，这不仅仅是为了提供服务发现功能，也是为了减少推测工作和开发工作量。客户端只需简单地完成服务注册工作，然后使用一个DNS接口或者HTTP接口就可以执行工作了，而其他系统则需要你定制自己的解决方案。

一个令人信服的服务发现框架必须包含健康检测功能，并且考虑失败的可能性。要是节点失败或者服务故障了，即使开发人员知道节点A提供Foo服务也是没用的。Navie系统利用的是心跳、周期性更新和TTLs，这些系统不仅需要工作量与节点数量成线性关系，并且对服务器的固定数量提出了要求。此外，故障检测窗口的存活时间至少要和TTL一样长。

ZooKeeper提供了临时节点，这些临时节点就是K/V条目，当客户端断开连接时，这些条目会被删除。虽然这些临时节点比一个心跳系统更高级，但仍存在固有的扩展性问题，并且会增加客户端的复杂性。与ZooKeeper服务器端连接时，客户端必须保持活跃，并且去做持续性连接。此外，ZooKeeper还需要胖客户端，而胖客户端是很难编写，并且胖客户端会经常导致调试质询。

Consul使用一个完全不同的架构进行健康检测。Consul客户端可以运行在集群中的每一个节点上，而不是拥有服务器节点，这些Consul客户端属于一个gossip pool，gossip pool提供了一些功能，包括分布式健康检测。gossip协议提供了一个高效的故障检测工具，这个故障检测工具可以应用到任意规模的集群，而不仅仅是作用于特定的服务器组。同时，这个故障检测工具也支持在本地进行多种健康检测。与此相反，ZooKeeper的临时节点只是一个非常原始的活跃度检测。因为有了Consul，客户端可以检测web服务器是否正在返回200状态码，内存利用率是否达到临界点，是否有足够的数据存储盘等。此外，ZooKeeper会暴露系统的复杂性给客户端，为了避免ZooKeeper出现的这种情况，Consul只提供一个简单HTTP接口。

Consul为服务发现、健康检测、K/V存储和多数据中心提供了一流的支持。为了支持任意存储，而不仅仅是简单的K/V存储，其他系统都要求工具和lib库要率先建立。然而，通过使用客户端节点，Consul提供了一个简单的API，这个API的开发只需要瘦客户端就可以了， 而且，通过使用配置文件和DNS接口，开发人员可以建立完整的服务发现解决方案，最终，达到避免开发API的目的。