王璞：基于容器的服务发现与负载均衡

曾在Google广告部门任职，负责广告的架构任务，14年回国同年9月创立数人云，主要基于Docker容器技术为企业级客户打造私有的PaaS平台，帮助企业客户解决互联网新业务挑战下的IT问题。

今天主要分享三个议题，首先是Google数据中心的简单介绍：Google的数据中心约有200万台服务器且都是X86PC服务器，Google的数据中心没有买任何大、小型机，完全使用廉价的PC服务器搭建，因规模庞大，对网络的要求非常高，包括交换机都是自己设计后定制的。服务发现、负载均衡的问题，对于Google的量级来说非常复杂，今天跟大家分享下Google内部如何实现服务发现和负载均衡。

静态的服务发现方式其实很好理解——基于IP地址和端口做服务发现，应用程序绑定了服务器的IP地址和端口之后，有请求发到这个IP地址和端口上，应用程序就可以接收到相

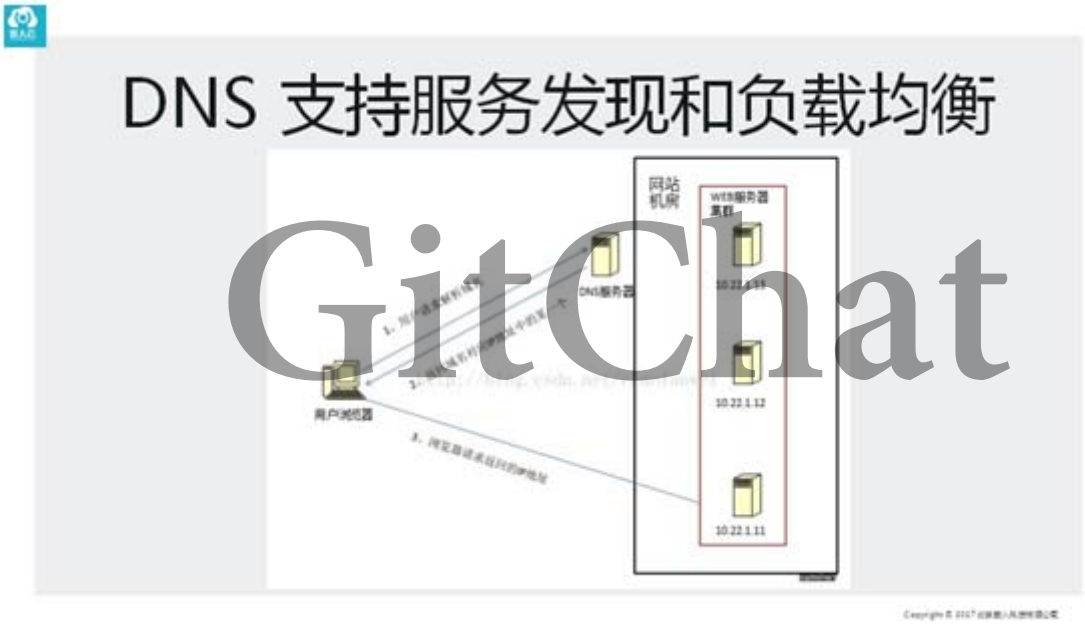
应的请求。

经典的负载均衡器也是绑定某个特定的IP地址和端口，同时负载均衡器将需要做负载均衡的应用实例预先配置好，当负载均衡器收到请求后即可分发给后台的应用实例。

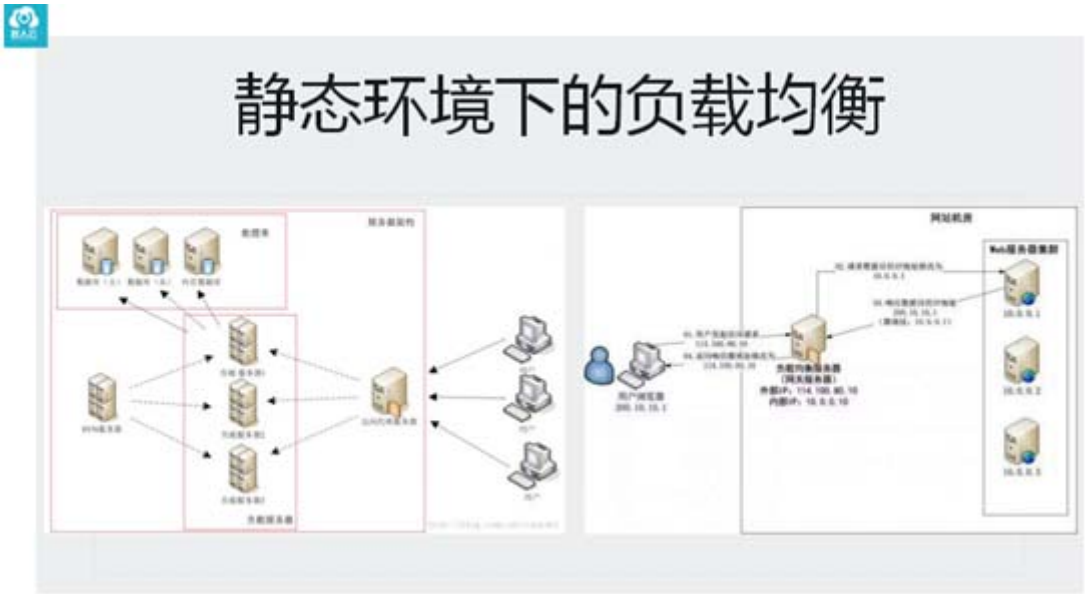
用IP地址+端口的方式做服务发现对人不友好，因为IP地址不好记忆，所以人们又发明了DNS作为非常经典的服务发现方式。

DNS实现的是域名解析，比较常用的DNS解析方式是A记录：向DNS查询某个域名的A记录会返回该域名对应的一个或多个IP地址，上图展示了向DNS查询某个域名的A记录返回IP地址的例子，给定一个域名，通过查询DNS服务器返回来这个域名所对应的一个IP地址。

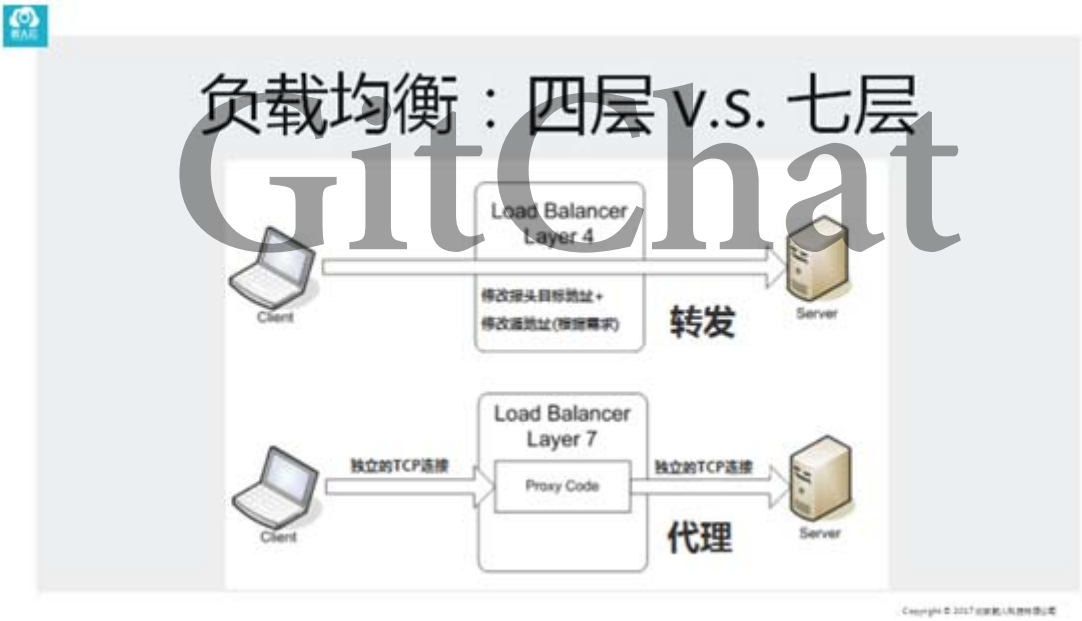
另外一种DNS解析方式SRV记录，这是DNS里面实现更高级的服务发现的一种方式，向DNS查询某个域名的SRV记录要返回该域名对应的一对或多对地址和端口，如上图所示，向DNS查询一个域名地址，DNS返回了该域名对应的一系列地址和端口。



DNS除了具有服务发现功能，也可以实现负载均衡的功能，如上图所示，DNS可以根据用户的请求动态的返回某域名的A记录，比如DNS返回的A记录是目前最不繁忙的实例的IP地址，这样DNS就可以实现负载均衡功能。



静态环境下的负载均衡是最常见的负载均衡器使用场景。如上图所示，用户的请求发给负载均衡器，负载均衡器根据一定的策略，如轮转策略或者按照一定的权重把收到的请求分发给后面具体的应用实例，应用实例在处理完请求后把响应返回给负载均衡器，然后负载均衡器再把请求响应返回给最终用户。



常见的负载均衡器支持四层和七层协议，具体讲就是TCP协议和HTTP协议。四层负载均衡器，按照TCP协议来说是实现了一种路由转发：一个TCP请求数据包经过四层负载均衡器时，负载均衡器只修改这个TCP请求数据包的目的地址然后转给后面的应用实例；当负载均衡器收到应用实例返回的TCP响应数据包时，会修改这个TCP响应数据包的目的地址然后返回给用户。

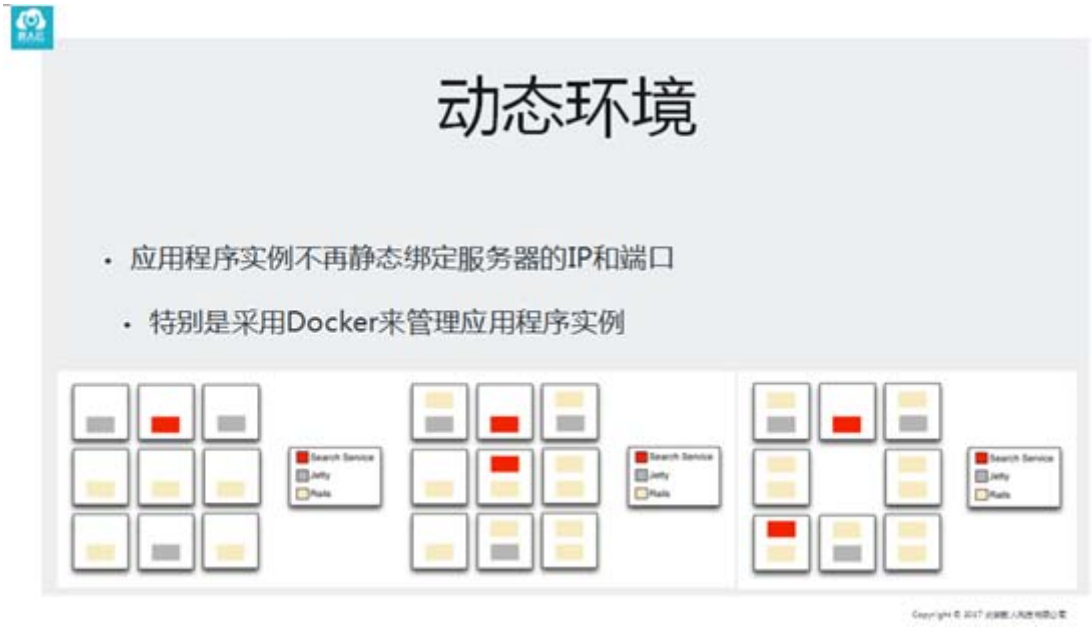
七层负载均衡器和四层负载均衡器的工作原理不一样；当七层负载均衡器收到一个用户的HTTP请求数据包会把该请求包拆掉，然后封装成一个新的HTTP请求数据包传给后面的应用实例；当负载均衡器收到应用实例返回的HTTP响应数据包时，会把HTTP响应数据包拆掉然后重新封装一个新的HTTP响应数据包返回给用户。所以四层和七层负载均衡器的工作原理不同，四层类似于路由转发，七层则是完全重新封装的包。

常见的服务发现方式有三种，分别适用于同的TCP协议或HTTP协议。第一种是用IP地址+端口或者域名+端口的方式做服务发现，比如，“website.com:8080”代表一个应用，“website.com:8081”代表另一个应用，虽然这两个应用的域名相同。这种方式适用于四层和七层协议，即TCP及HTTP协议都可以用。

第二种是子域名的方式，仅适用于七层协议。子域名的方式是指不同的应用可能有共同的根源，但是有不同的子域名，比如，http://service1.zone1.website.com 和 http://service2.zone1.website.com，这两个不同的域名（访问端口都是80）），有共同的根域名website.com，但是子域名不同，因此七层协议，比如HTTP协议，会通过不同的子域名解析到不同的应用。

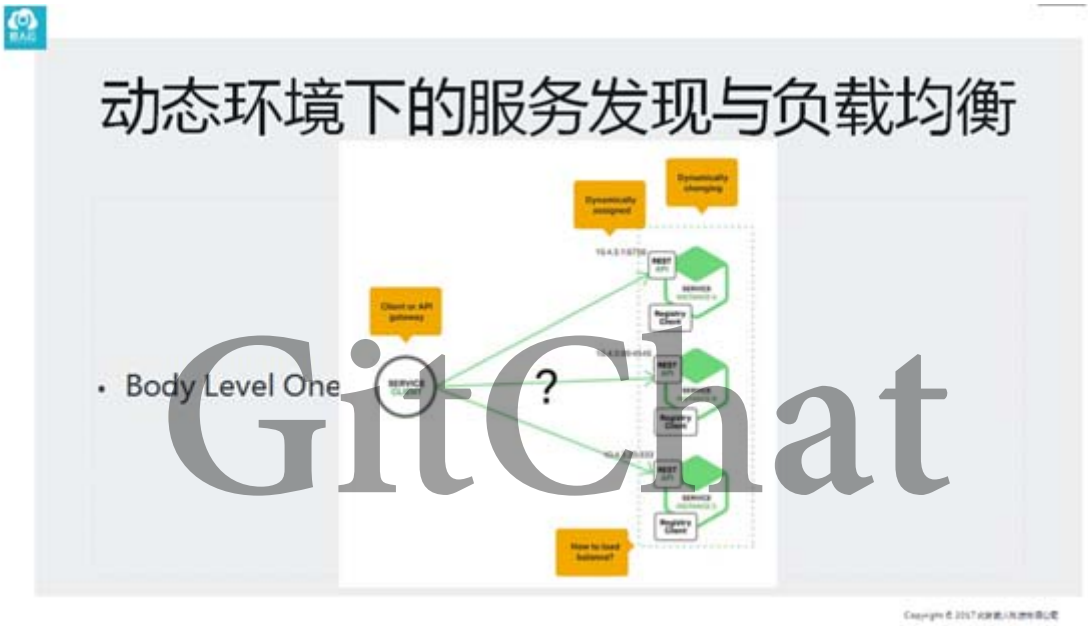
第三种是子路径的方式，也仅适用于七层协议。比如，http://zone1.website.com/service1 和http://zone1.website.com/service2，这两个路径的域名完全一样，但是子路径不一样，可以用于区分不同的应用服务。

这三种服务发现方式其实总结下来只有IP地址或者域名+端口是同时适用于四层、七层，其他如子域名、子路径的方式只适用于七层服务发现。

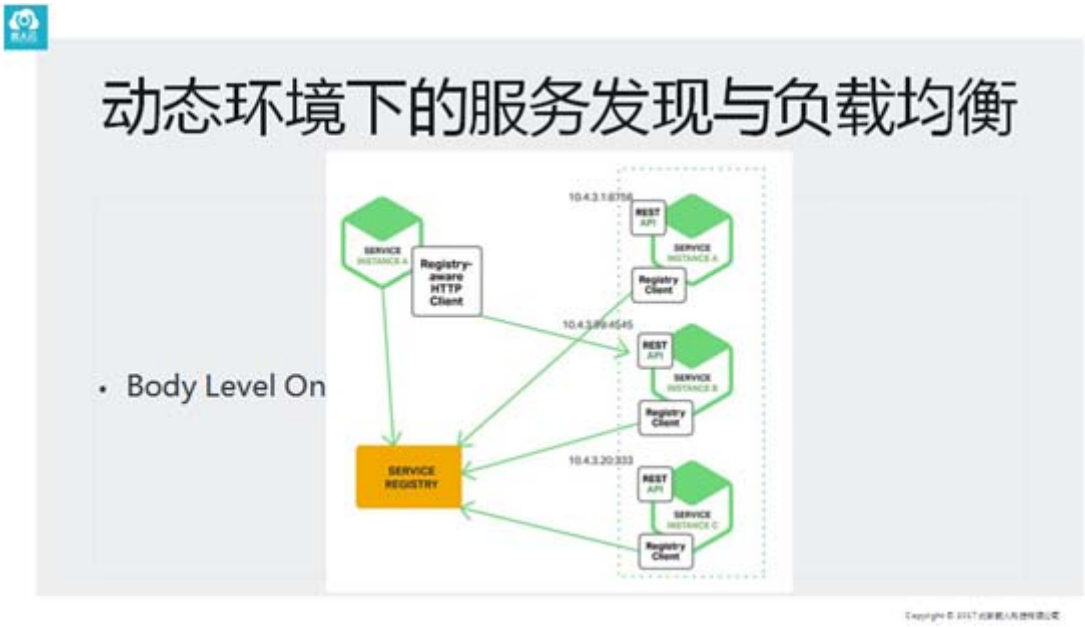


上述都是非常经典的负载均衡、服务发现的基本概念和做法，但当数据中心规模达到一定程度时，应用和服务器之间或更确切地说应用和具体IP地址+端口之间就不再是静态的绑定关系。

如Google的数据中心里面大约有200万台服务器，如果Google的应用和服务器之间是一一对应的静态绑定关系：即某个应用程序必须要绑在某一个服务器上，或是绑定某个服务器的IP地址+端口上，那么对Google来说，每时每刻大概会有几百万到上千万个应用程序运行在200多万台服务器，这样静态管理应用和服务器会非常复杂。所以对于Google如此大规模的数据中心来讲必须用动态的管理，即要求应用不能静态绑定在服务器的IP地址+端口上，应用可以在不同的服务器之间动态迁移来实现故障自愈：程序运行在某一个服务器上，这个服务器宕机或程序有问题，该程序会被自动迁移到别的服务器上恢复运行。动态的应用调度管理方式可以使得应用的管理及服务器的管理进行解耦，即应用和服务器之间不再是静态的绑定关系。



动态环境下如何做负载均衡和服务发现？首先把问题明确下，动态环境最根本的一点是要把服务发现实现，即客户端要找到服务的后台，它从哪里找？这就是服务发现。在动态环境下如何做到？其实并不复杂，每个服务的后台实例绑定的IP地址和端口注册到一个服务注册中心，注册的方式可以是被动注册也可以是主动注册，被动注册是指负责应用调度的调度器来完成应用实例的IP地址+端口注册；主动注册是指每一个服务的实例要主动的上报自己目前所绑定的IP地址+端口。



有了动态服务注册的机制后，动态环境下的负载均衡也就好实现了。在动态环境下，当负载均衡器收到一个请求后，会去服务注册中心进行查询相应的应用的实例地址，然后把请求路由到该应用的后台实例上。

Google内部的服务发现和负载均衡外面看不到，数人云借鉴Google的理念实现了Swan（Github地址：https://github.com/Dataman-Cloud/swan），Swan基于Mesos来做容器化应用的动态调度，同时Swan实现了DNS和Proxy支持服务发现和负载均衡，跟Google的方式几乎一模一样，所以后面用Swan作例子给大家分享下Google怎么做服务发现和负载均衡。

先讲一下如何给应用命名，这在动态的应用调度和运行的环境下非常总要，因为经典的应用发现方式都是按照IP和端口，没有对应用有统一的命名，但是Google对于每个应用、每个实例都会有相应的命名。首先明确几个概念：

一个实例，是应用的某个Task，运行在一个容器里，应用会包含多个Task，都是运行同样的二进制程序；

一个应用，是一组运行同样二进制程序的实例集合，每个实例是这个应用的某个Task；

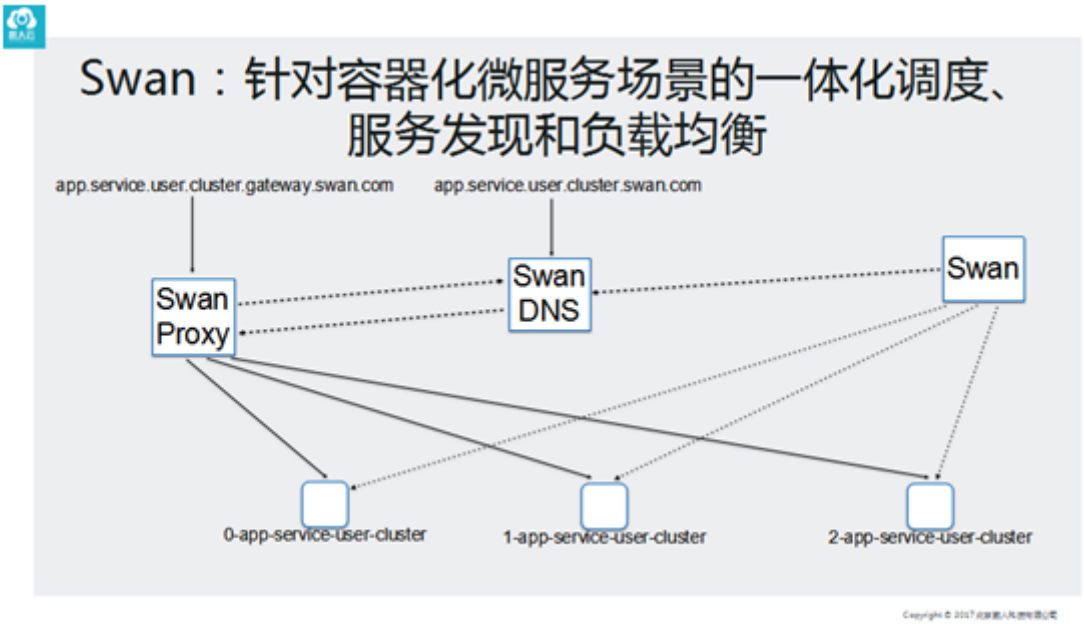
一个服务可以是一组应用程序；

一个服务会由一个用户在某个集群上发起运行。

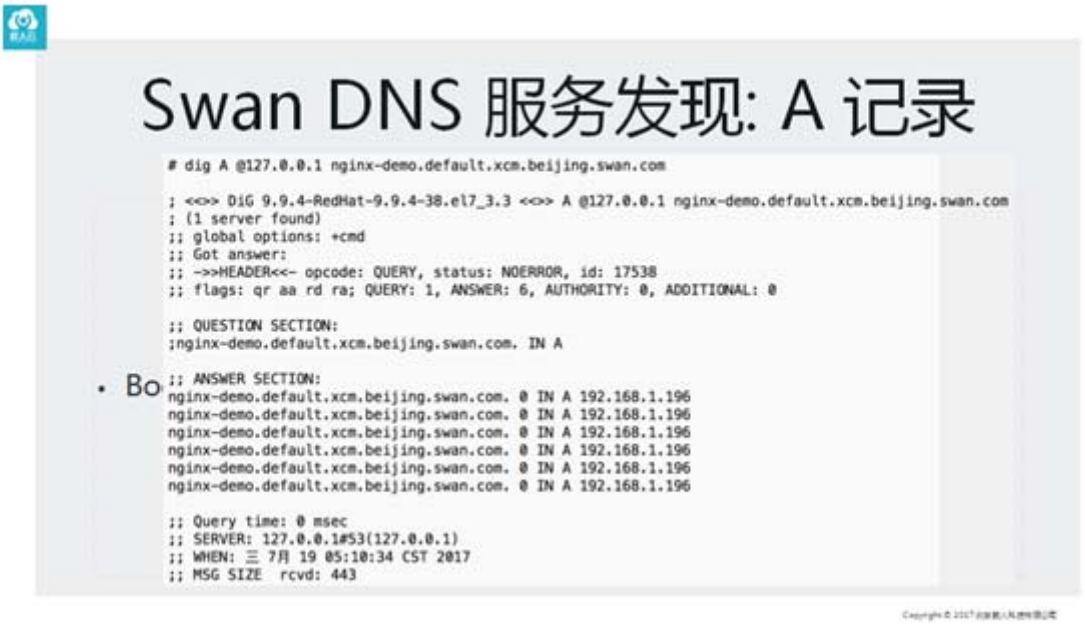
Swan给每个实例用五个标签来命名，task-app-service-user-cluster，task是从0开始的连续整数，用于标识不同实例；相应地Swan给每个应用四个标签来命名，app-service-user-cluster；进而，Swan给每个服务用是三个标签来命名，service-user-cluster。

Swan实现了DNS用于服务发现，就是Swan DNS把Swan调度的每一个实例所绑定的IP地址+端口的信息都记录下来，或是A记录或是SRV记录。

对于每个应用，Swan的DNS也生成一个相应的域名用于四层服务发现，即app.service.user.cluster.swan.com。另外，七层的应用Swan Prox会解析应用的另外一个域名 http://app.service.user.cluster.gateway.swan.com用于七层应用的服务发现和负载均衡。



上图是Swan架构的一个示意图，简单解释了Swan、 DNS、 Proxy之间的关系：如何通过Swan对应用动态调度后实现服务发现和负载均衡。举个例子，首先Swan发布一个应用app-app-service-user-cluster，包含三个实例分别是：0-app-service-user-cluster，1-app-service-user-cluster ，2-app-service-user-cluster；当Swan把三个实例都运行起来后，Swan会把三个实例目前运行时所绑定的IP+端口信息提交给Swan DNS。比如我们可以访问Swan DNS去解析app.service.user.cluster.swan.com这个域名，会解析出来三个容器的实例；当用户的请求访问app.service.user.cluster.gateway.swan.com，该请求会送达到Swan Proxy上，因为Swan Proxy地址是gateway.swan.com，Swan Proxy采用子域名的方式解析app.service.user.cluster。Swan Proxy解析这个地址时会查Swan DNS，查这个应用所对应的实例，每一个实例分别在哪个IP+端口上。Swan Proxy查询了Swan DNS之后，发现后面它有三个实例，这三个实例分别在不同的IP和端口上。当Swan Proxy收到对这个应用请求时会分别往后面三个实例上进行分发。



上图详细地解析了Swan DNS如何做服务发现，展示的是Swan DNS里面的A记录，图里对应的A纪录是nginx-demo.default.xcm.beijing.swan.com应用，应用名称叫nginx-demo，属于default这个服务，用户名是xcm，default这个服务目前运行在beijing这个数据中心里，

swan.com作为一个后缀去表示，完整的表示出这是Swan的内网域名。A纪录展现出来这个应用有6个实例，每个实例都在192.168.1.196的IP地址上。

上图是Swan DNS的SRV记录，SRV记录和A记录不一样的地方是A记录只返回域名的IP地址，SRV记录要返回域名的IP地址+端口。上图所示的SRV查询结果包含了6个不同的应用实例，分别在不同的端口上，6个不同的实例又在同一个IP地址上：192.168.1.196，但它们绑定的端口不一样，从31000、31001、301002、301003、301004、301005、31006。

Swan实现了Proxy用于负载均衡。先讲一下Swan Proxy如何支持七层负载均衡。Swan Proxy支持子域名方式实现七层负载均衡。前面提到过，用户的HTTP请求发往app.service.user.cluster.gateway.swan.com这个域名地址时，先是.gateway.swan.com解析到Swan Proxy的IP地址上，然后因为Swan Proxy针对HTTP协议做解析的时候它会解析HTTP协议里面的域名，这个域名的子域名就是app.service.user.cluster，也就是这个域名里面的前缀。按照这个前缀，Swan Proxy可以区分出该HTTP请求是要访问哪个具体的应用。 Swan Proxy在做HTTP这个服务发现负载均衡的时候会支持会话保持，也会支持HTTPS。但是Swan Proxy不支持HTTP子路径方式，因为子路径的方式本质上讲不是一种

负载均衡的方式，子路径其实和应用所提供的不同服务相关的，所以具体的子路径服务的注册方式需要用额外的，比如微服务自身的服务发现支持，比如SpringCloud里面的Eurake或者阿里的Dubbo这些服务注册中心来做子路径方式的服务注册。

再讲一下Swan Proxy对于四层的负载均衡。因为四层协议，比如TCP协议，的特殊性，Swan Proxy支持的TCP协议只能是端口方式，根据一个Swan Proxy的IP或者Swan Proxy的域名，加上不同的端口来区分不同的应用。Swan Proxy在对TCP进行负载均衡的时候也会支持会话保持。

最后汇总下Swan的服务发现、负载均衡方式。结合容器目前的几种网络模式：Bridge方式、Host方式还有固定IP的方式，上图给出Swan在不同容器的网络模式下如何做服务发现、负载均衡。