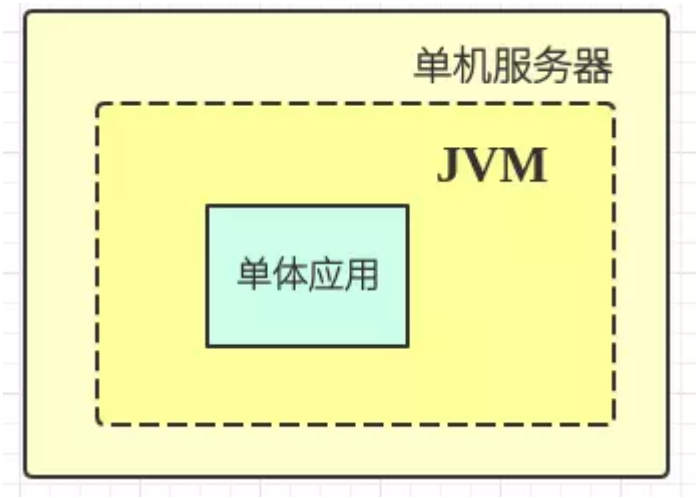
Eureka服务注册与发现

> 更多学习资料，一点课堂：www.yidiankt.com
>
> 一点课堂QQ群：984370849， QQ：2868289889
>
> 微信号：chengweixin9
>
> 免费公开课：https://ke.qq.com/course/394307

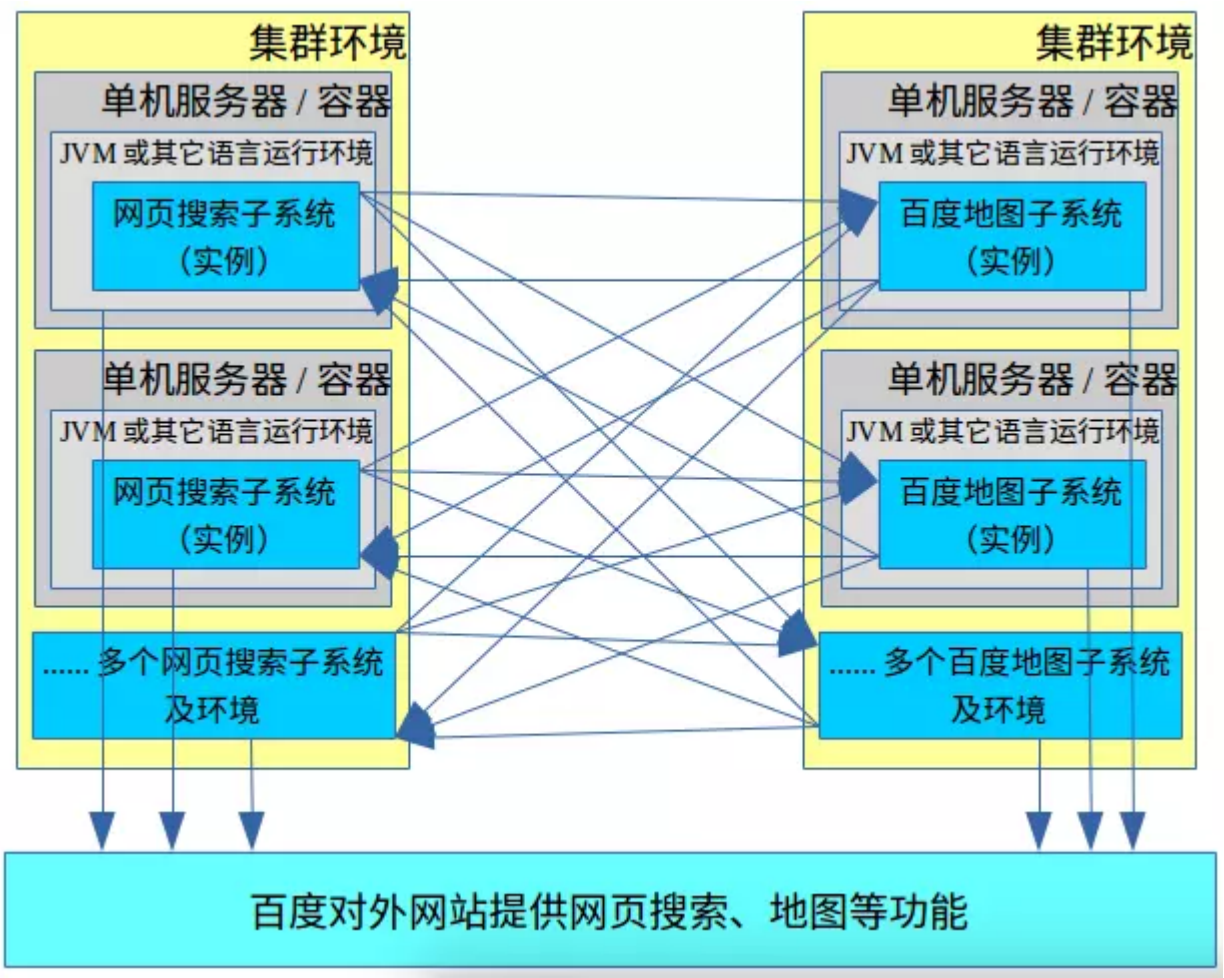
为什么需要服务中心

过去，每个应用都是一个CPU，一个主机上的单一系统。然而今天，随着大数据和云计算时代的到来，任何独立的程序都可以运行在多个计算机上。并且随着业务的发展，访问用户量的增加，开发人员或小组的增加，系统会被拆分成多个功能模块。拆分后每个功能模块可以作为一个独立的子系统提供其职责范围内的功能。而多个子系统中，由于职责不同并且会存在相互调用，同时可能每个子系统还需要多个实例部署在多台服务器或者镜像中，导致了子系统间的相互调用形成了一个错综复杂的网状结构。用几幅图说明一下：

单体应用：



随着业务的发展，经过了多个系统架构的演变，变成了这样（拿百度的功能举个栗子）：



图中，每个网页搜索子系统和百度地图子系统的实例都可以视同为一个微服务。网页搜索子系统为百度地图子系统提供了“用户查询内容、用户IP地址”等信息提供的服务接口，为百度地图子系统定位用户地理信息情况提供数据依据。

百度地图子系统提供了“根据内容查询出地图信息”的接口提供给其他子系统调用，而这里网页搜索子系统调用了这个接口，获取地图相关信息。

网页搜索子系统和百度地图子系统又提供了各自对外用户调用的网页搜索、地图搜索等各自的对外服务。这个过程就形成了以上错综复杂的网状结构。而实际上这样还远远不够，因为每个子系统往往会提供多个对内的其他子系统调用的服务接口，同时也会调用多个不同子系统提供的多个服务接口，还会对外提供多个各自的服务接口。所以实际中上图的网状调用结构将会成几何倍的扩张。而且随着用户量的增加，每个子系统还需要继续增加更多的实例来提供服务，从而导致了凌乱的加剧。

对于微服务之间错综复杂的调用关系，通过eureka来管理，可以让每个服务之间不用关心如何调用的问题，专注于自己的业务功能实现。

Eureka的管理：

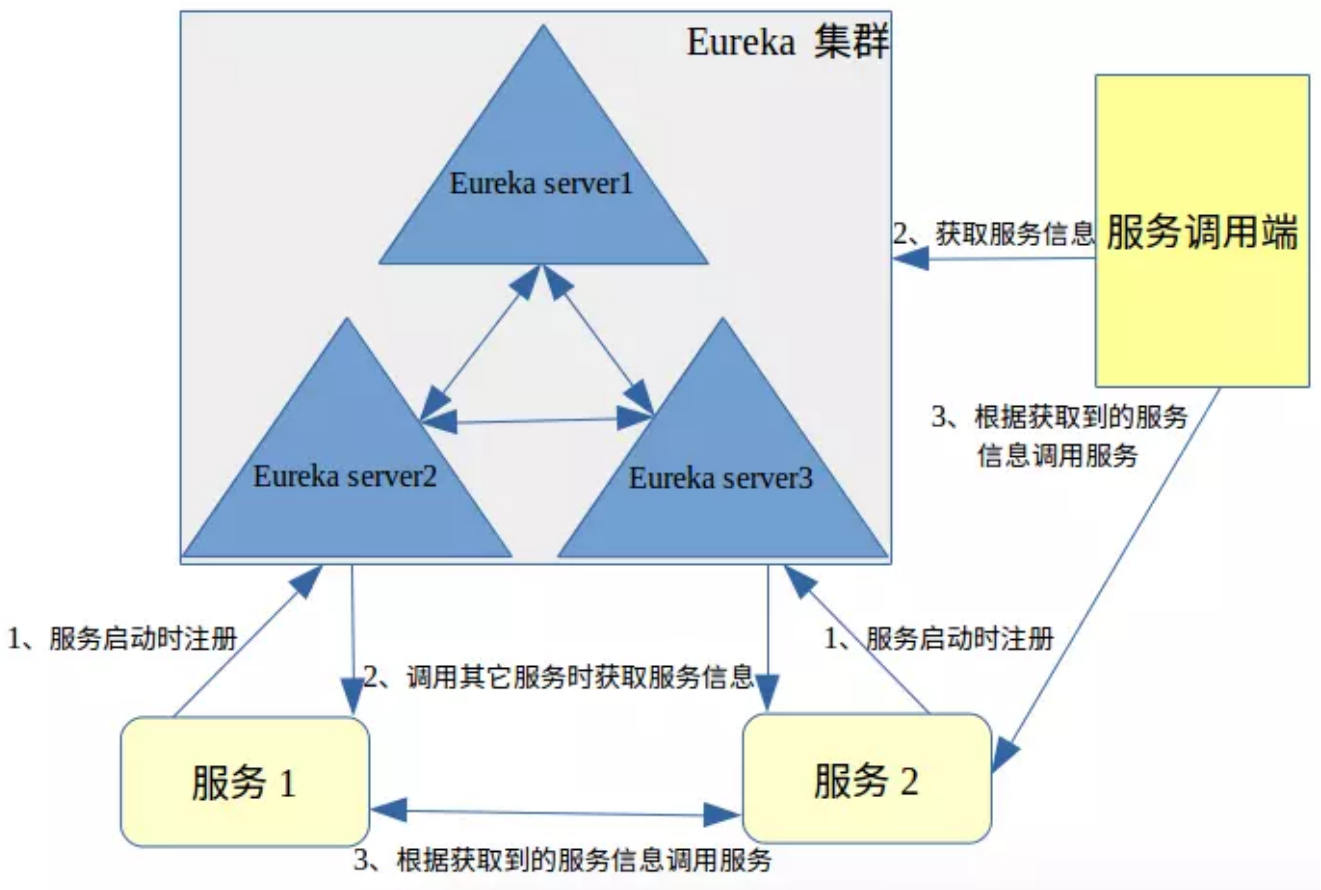
基于以上概念，使用Eureka管理时会具备几个特性：

→服务需要有一个统一的名称（或服务ID）并且是唯一标识，以便于接口调用时各个接口的区分。并且需要将其注册到Eureka Server中，其他服务调用该接口时，也是根据这个唯一标识来获取。

→服务下有多个实例，每个实例也有一个自己的唯一实例ID。因为它们各自有自己的基础信息如：不同的IP。所以它们的信息也需要注册到Eureka Server中，其他服务调用它们的服务接口时，可以查看到多个该服务的实例信息，根据负载策略提供某个实例的调用信息后，调用者根据信息直接调用该实例。

eureka如何管理服务调用

eureka如何管理服务调用的？我们先来看个图：



→在Eureka Client启动的时候，将自身的服务的信息发送到Eureka Server。然后进行2调用当前服务器节点中的其他服务信息，保存到Eureka Client中。当服务间相互调用其它服务时，在Eureka Client中获取服务信息（如服务地址，端口等）后，进行第3步，根据信息直接调用服务。（注：服务的调用通过http(s)调用）

→当某个服务仅需要调用其他服务，自身不提供服务调用时。在Eureka Client启动后会拉取Eureka Server的其他服务信息，需要调用时，在Eureka Client的本地缓存中获取信息，调用服务。

→Eureka Client通过向Eureka Serve发送心跳（默认每30秒）来续约服务的。 如果客户端持续不能续约，那么，它将在大约90秒内从服务器注册表中删除。 注册信息和续订被复制到集群中的Eureka Serve所有节点。 以此来确保当前服务还“活着”，可以被调用。

→来自任何区域的Eureka Client都可以查找注册表信息（每30秒发生一次），以此来确保调用到的服务是“活的”。并且当某个服务被更新或者新加进来，也可以调用到新的服务。

1. Eureka Server和Eureka Client。
    - Eureka Server：

提供服务注册：各个微服务启动时，会通过Eureka Client向Eureka Server进行注册自己的信息（例如服务信息和网络信息），Eureka Server会存储该服务的信息。

提供服务信息提供：服务消费者在调用服务时，本地Eureka Client没有的情况下，会到Eureka Server拉取信息。

提供服务管理：通过Eureka Client的Cancel、心跳监控、renew等方式来维护该服务提供的信息以确保该服务可用以及服务的更新。

信息同步：每个Eureka Server同时也是Eureka Client，多个Eureka Server之间通过P2P复制的方式完成服务注册表的同步。

- Eureka Client

Eureka Client是一个Java客户端，用于简化与Eureka Server的交互。并且管理当前微服务，同时为当前的微服务提供服务提供者信息。

Eureka Client会拉取、更新和缓存Eureka Server中的信息。即使所有的Eureka Server节点都宕掉，服务消费者依然可以使用缓存中的信息找到服务提供者。

Eureka Client在微服务启动后，会周期性地向Eureka Server发送心跳（默认周期为30秒）以续约自己的信息。如果Eureka Server在一定时间内没有接收到某个微服务节点的心跳，Eureka Server将会注销该微服务节点（默认90秒）。

服务续约、下线、剔除

服务续约

Application Service内的Eureka Client后台启动一个定时任务，跟Eureka Server保持一个心跳续约任务，每隔一段时间(默认30S)向Eureka Server发送一次renew请求，进行续约，告诉Eureka Server我还活着，防止被Eureka Server的Evict任务剔除。

服务下线

Application Service应用停止后，向Eureka Server发送一个cancel请求，告诉注册中心我已经退出了，Eureka Server接收到之后会将其移出注册列表，后面再有获取注册服务列表的时候就获取不到了，防止消费端消费不可用的服务。

服务剔除

Eureka Server启动后在后台启动一个Evict任务，对一定时间内没有续约的服务进行剔除。

服务通讯方式

服务间使用标准的REST方式通讯，所以Eureka服务注册中心并不仅适用于Java平台，其他平台也可以纳入到服务治理平台里面。

自我保护

本地调试Eureka的程序时，会出现：

```
EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.
```

该警告是触发了Eureka Server的自我保护机制。

Eureka Server在运行期间，会统计心跳失败的比例在15分钟之内是否低于85%，如果低于，就会将当前实例注册信息保护起来，让实例不会过期，尽可能保护这些注册信息。

但是如果在保护期间，实例出现问题，那么客户端很容易拿到实际已经不存在的服务实例，会出现调用失败。这个时候客户端的容错机制就很重要了。（重新请求，断路器）

保护机制，可能会导致服务实例不能够被正确剔除。

在本地开发时，可使用：eureka.server.enable-self-preservation=false关闭保护机制，使不可用实例能够正常下线。

Eureka和Zookeeper区别

Eureka：可以在发生因网络问题导致的各节点失去联系也不会暂停服务，但是最新的数据可能不统一。

Zookeeper：如果发生网络问题导致的Master和其他节点失去联系，就会使得其他的节点推选出新的Master，但是推选的时间内无法提供服务，但是可以保证任何时候的数据都是统一的。