知乎的软件架构

微服务架构，知乎从 11 年就开始了微服务的探索，尝试过 protocol buffers**、**Avro**、**Thrift，最终在 16 年确认使用 Thrift，同时使用 Consul 和 HAProxy 作为注册中心和负载均衡。是在 14 年确认的这套微服务架构。

微服务架构

好处：

1.易于开发和维护：一个服务只关注一个特定的业务功能，所以它业务清晰，代码量少。开发和维护单个微服务相当简单。而整个应用是若干个微服务构建而成的，所以整个应用在被维持在一个可控的状态；

2.单个服务启动快：单个服务代码量少，所以启动快；

3.局部修改易部署：单个应用只要有修改，就得重新部署整个应用，微服务解决了这个问题。一般来说，对某个微服务进行修改，只需要重新部署这个服务即可；

4.技术栈不受限：在微服务架构中，可以结合业务和团队的特点,合理选用技术栈。例如有些服务可以使用关系型数据库Mysql，有的服务可以使用非关系型数据库redis。甚至可根据需求，部分服务使用JAVA开发，部分微服务使用Node.js开发

5.按需收缩：可根据需求，实现细粒度的扩展。例如，系统中的某个微服务遇到了瓶颈，可以结合微服务的特点，增加内存，升级CPU或增加节点。

缺点：

运维要求高：更多的服务意味着更多的运维投入。在微服务中，需要保证几十甚至几百个服务器正常运行和协作，这给运维带来了巨大的挑战；

分户式固有的复杂性：使用微服务架构的是分布式系统。对于一个分布式系统，系统容错，网络延迟都会带来巨大挑战。

接口调整成本高：微服务之间通过接口进行通信