# 论微服务与层次架构的应用

## 概要：

本人在2019年10月参加了昆明某互联网公司的二手商品交易系统的开发，在该系统开发中本人担任架构师的工作。该系统包含买卖二手商品交易过程中所涉及的各个信息，包括用户个人信息，商品详情，订单信息，库存信息，物流信息，物流画像信息，即时通信信息。由于本公司采用了C2C的经营模式，使得此业务发展空间比较大，后期用户增加迅猛，所以系统需要具备一定的扩展性，并且要保证一定的高可用。根据这些情况，开发了此系统。

在一些短时间的大访问量的电商场景，系统的灵活扩展尤为重要。因此我们采用松耦合、可插拔的微服务架构。同时针对众多服务调用复杂的问题，我们采用层次架构缓解系统的复杂性。本文讲述了微服务和层次结构架构在项目中应用过程以及使用效果，在最后，讨论了项目完成后的效果和项目的不足。

## 正文：

本人于2019年10月参加了昆明某互联网公司的二手商品交易系统开发，并在该项目中担任架构师的工作。该系统包含二手交易过程中所需要的各个信息系统，包括用户身份，购物车，订单，库存，物流，即时通讯，用户画像等。考虑到公司采用了C2C的经营模式，使得此业务发展空间较大，后期用户增多较快，订单成交量较多，所以此系统需要具备易扩展性、易集成性、高性能。

该二手交易系统（以下简称STS系统），需要在日后电商活动的高并发量、高访问量的场景里最大程度避免宕机，服务雪崩，缓存击穿等一系列由于负载过大而造成的种种故障，并保证较低响应时间，较高的可用性。我们需要使用较为轻量级的通信机制、服务降级、服务熔断、负载均衡、缓存等技术。所以我们需要松耦合，可插拔，支持独立部署的系统设计。基于以上考虑，我们采用了微服务的架构。

我们采用微服务架构，是因为微服务在系统开发好电商场景下具有种种优势。微服务使用较为轻量级的通讯机制，使得服务之间的通讯较为简单，其异步通信机制能缓解通信阻塞。微服务颗粒度较小，能提高组件的复用。微服务支持独立的部署，本系统较为复杂，构件的独立部署能方便整个项目的部署，并且每个构件能独立扩展。而且在一些场景中，能够限制一些构件的计算量，让这些构件仅做到柔性可用，让一些重要的构件获得大量的资源，来提高整个系统的吞吐率和响应速度，让用户有良好的购物体验。此项目比较大，使用微服务能够细化团队，从而获得理想的团队及生产力。同时为了缓解微服务带来的系统沉重，职责混乱问题，我们还采用了层次结构的架构架构风格。我们把整个系统划分为4个层次，分别为接入层，逻辑层，路由层，数据层，数据存储层。然后将每个子系统放入对应的层次中。

接入层负责将Apps或Web浏览器端的请求接收过来，因为请求并发量较大，还要保证整个系统的高可用，所以在接入层使用了nginx反向代理来达到负载均衡的目的，将请求按照权值分发到逻辑层的多台服务器中。负载均衡中有服务器可用状态实时监测的机制，使用心跳监测机制可以实时监测服务器的状态，当某台服务器不可用时，可以将任务自动转移到其他服务器。当不可用的机器再次可用时，可以自动重连恢复。同时在nginx服务器中设置了动静分离，将动态资源和静态资源进行了分离，我们对静态文件缓存到服务器中或者压缩来提高整个网站的性能。除此之外，我们还设计了供第三方调用的接口，攻防检查，用户校验等。

逻辑层负责响应接入层的调用，处理业务逻辑。逻辑层负责整个平台的逻辑处理。根据业务流程，我们将逻辑层垂直划分为用户服务，购物车服务，订单服务，库存服务，物流服务，即时通讯服务，数据挖掘服务，商品详情服务。我们让微服务数量和团队数量结合在一起，使每个团队只负责一个微服务，提高每个团队对自己所负责的服务的理解。同时我们划分出核心服务和非核心服务，在一些电商活动的场景里，如商品秒杀，抢优惠券等，这些会使整个系统的计算量爆发式地增多。为了使得整个系统的平稳运行，保持一定的响应时间，我们会让一些非核心服务进行服务降级或服务限流。在这些场景里，数据挖掘服务的重要性远远低于其他服务的重要性，我们直接将其降级，让其减少运算直接返回数据。及时通讯服务虽然也有一定的重要性，但是我们公司的STS不主推即时通信功能，不要求即时通讯功能在市场上占有一定比例，仅仅是买卖双方沟通的工具。所以我们将即时通信服务进行了服务限流。得以非核心服务性能的代价来换取其他核心服务的计算性能。

路由层是IM特有的一层，负责路由转发的功能。用户的一次登录session相关的数据（比如用户在线状态，登录IP等），这部分设计数据变化非常快，没必要固化到数据库中，直接储存到内存中，并记录用户登录接入层的路由信息。用户在发送消息通讯时，路由层会将用户的消息直接返回给接入层，然后接入层再把传递到另外一个用户的客户端中。

数据层能够对外暴露统一的接口，屏蔽了底层的数据存储引擎。上层不用担心实际的存储是RDBM或者是NoSQL，对上层提供友好地、统一的访问接口，封装原子逻辑。我们使用MyBatis通过CURD接口的抽象封装，增加新的数据操作只需要增加配置，不用变动数据层代码，较容易完成关键数据的存储，简化了系统的开发难度。

数据存储层是系统固化数据的系统。为了在高并发场景里，减少关系型数据库的访问次数，提高数据库系统的稳定性，我们引入了Redis分布式缓存。在数据库系统设计的过程中，发现业务数据读的概率大于写的概率，且数据库系统读的效率大于写的效率，所以我们在数据库中做了读写分离机制。然后为了数据的一致性，做了主从同步机制。在设置数据库放服务器规模时，我们考虑到当某些缓存时效，流量打入数据库时，数据库系统多大才能承受住并保证一定的性能。

通过综合利用以上设计，系统于2020年2月投入了运行，并取得了良好的效果，公司业务能平稳的运行，系统也能适应短时间高并发的应用场景。系统的得到了领导的好评，。

尽管系统的建设取得了成功，但是在平时建设我也看到了一些不足之处，最主要的是不同服务的对硬件的需求不一样，但各服务器的硬件差别并不大。有些服务需要大量的计算资源，例如逻辑层的数据挖掘服务；有的服务需要大量的内存资源，例如缓存服务。但是硬件并没有根据软件的需求进行定制化，导致有些服务器的部分资源利用率不高，形成了浪费。

如果未来对这套系统进行修改，我希望能与团队做好充分的服务器硬件需求分析，个性定制硬件，使硬件得到充分的利用率。