摘要：  
本人于2015年11月参与浙江省某在线教育平台“外教一对一在线教育”项目，该项目为客户提供了一对一欧美外教视频教学，社交圈，公众直播等功能提供全方位的软件支撑，在该项目组中我担任系统架构师岗位，主要负责整体架构设计与中间件选型。本文以该教育平台为例，主要讨论了该系统有关可靠性方面的设计与应用，以及遇到的问题与解决方案。一方面通过负载均衡进行容错技术中冗余设计的实现，另一方面通过层次架构风格来明确系统结构体系，从而降低系统设计复杂度，提高系统可靠性。整个系统开发工作历时18个月。目前，该系统已经稳定运行近一年半的时间。实践证明，通过容错设计，降低复杂度设计等，系统有效提高了可靠性，从而为公司业务提供持续稳定的服务支撑。

正文：  
随着国家对教育的越发重视，英语教育的市场份额逐步上升，其中用户口语提升的需求越来越大。为此，一些公司开始提供与外国人聊天的平台。我所在公司决定从国际通讯领域进军口语教育领域。为了这项战略转变，公司于2015年11月设计某在线教育平台系统（一下简称为“系统”）。该系统帮助人们与欧美外教进行面对面的口语交流和教学。其中随意聊提供了一种类似QQ视频通话，而正式课程还提供了H5互动课件与课后点评等，以提高教学质量。与此同时，还有公众直播用于拉新，AI测试用于评定学院能力，降低成本。我参与了该项目的开发工作，担任系统架构设计师职务，负责设计系统架构。本项目组全体成员共9人，我主要负责项目计划制定，需求分析，整体架构设计与技术选型，以及部分底层设计。该项目的架构工作与次年2月完成，选择了层次架构风格。整个项目耗时18个月，于2017年5月完成。  
目前主流的可靠性设计技术有容错设计，检错设计，降低复杂度设计等技术。容错设计技术分为恢复块设计，N版本程序设计和冗余设计。其中恢复块设计是选择一组软件操作作为容错设计单元，将普通的程序块编程恢复块。N版本程序设计的核心是通过设计出多个模块或不同版本，对于相同初始条件和相同输入的操作结果，实现多数表决，防止其中某一软件模块/版本的故障提供错误的服务，以实现软件容错。冗余设计是在一套完整的软件系统之外，设计一种不同路径，不同算法或不同实现方法的模块或系统作为备份，在出现故障时可以使用冗余的部分进行替换，从而维持软件系统的正常运行。缺点是费用和资源的消耗会有所增加。检错技术是在软件出现故障后能及时发现并报警。其缺点是不能自动解决故障。降低复杂度设计是因为软件复杂性与软件可靠性有着密切关系，是产生软件缺陷的重要根源。在设计时考虑降低软件的复杂性，是提高软件可靠性的有效方法。

在了解系统需求后，我们决定听从公司技术顾问的建议，容错设计主要应用在冗余设计方面，通过负载均衡，双机容错等机制完成冗余设计。检错设计则是通过对Java异常处理机制的设计与封装处理完成。至于降低复杂度方面，采用层次架构风格，使得系统的结构明确，立体，从而提高系统可靠性。接下来，我将从系统的冗余设计，复杂度降低设计介绍可靠性在系统中的设计与应用，以及应用过程中遇到的问题与解决方案。

1.冗余设计：

首先说冗余设计，冗余包含逻辑冗余，数据冗余，应用冗余等。这里以应用冗余为例。为了提高系统的性能，可靠性，可拓展性等，我们采用了负载均衡技术。常见的负载均衡技术有F5硬件，LVS软件，Nginx服务器配置等。出于便捷与成本的考虑，我们采用了Nginx服务器配置负载均衡技术。通过对Nginx服务器中upstream模块的配置，就可以实现在多台服务器的反向代理家在负载均衡。采用负载均衡后，应用服务器集群存在Session问题无法统一的问题。解决方法有Session Sticky，Session Replication，Session数据集中存储，Cookie Based四个方案。Session Sticky是通过确保同一个会话的请求都在同一个Web服务器上处理实现。Session Replication是增加Web服务器间会话数据的同步来保证不同Web服务器间的Session数据的一致。Cookie Based就是通过Cookie传递Session数据完成。经过考虑，我们采用了Session数据集中存储。Session数据集中存储通过令每台服务器从专门的session服务器获取session数据来解决问题。优点是可靠性，可移植性与可拓展性的大幅提高。缺点是一方面读写Session数据引入了网络操作，对数据读取存在时延和不稳定性，但对于使用内网通信的系统并没有太大影响。另一方面，如果Session服务器或集群出现问题，将会影响整个应用。我们通过双机容错机制解决该问题。除此之外，还有心跳线，看门狗等技术。限于篇幅，不再赘述。

2.降低复杂度设计：

再者就是降低复杂度设计，由于系统的复杂性和综合性，我们决定采用层次架构风格，将系统架构分为接入层，应用层，服务层，数据层四个层次。这里以应用层与服务层为例。应用层分为视图层与业务逻辑层，视图层负责App与网站的表现效果，业务逻辑层负责业务层的逻辑处理。为了解决系统日益复杂，应用日益臃肿问题，我们将系统按照应用横向划分，将系统划分为课件管理系统，课程管理系统等十余个子系统。如课件管理系统负责学员上课所用课件，有课件编辑，课件预览，课件交互等多个功能模块。功能模块需调用服务层的服务支撑，如课件交互模块需要调用stomp通信服务，实现学生与老师间课件的交互功能。另外，课件交互模块通过对账户服务的调用，确立课件双方的身份，从而明确双方在课件交互过程中对课件交互部分的交互权限。该划分使得系统体系变得清晰明了，极大降低系统复杂度，提高系统可靠性。应用层采用基于J2ee的MVC框架-Structs框架，主要通过Servlet和JSP技术实现。另外还有动静分离，动态资源静态化等，这里不再赘述。

服务层提供通用服务。系统在应用层中按照应用横向划分，有效降低系统复杂度。但系统代码仍然存在冗余，比如用户信息的调用在诸多应用子系统中都有相关模块。另外应用的大小依旧十分巨大，复杂，而过小的应用划分会增加数据库连接数负担，故我们提出服务化解决方案。服务化方案就是提取出各个应用的通用服务，如账户服务，Session服务等。出于技术成熟度与技术支持等考虑，我们最终采用了阿里的dubbo服务框架，建立服务层。开发过程中，产生了服务框架部署问题与实现服务框架的jar包和应用自身依赖的jar包冲突的问题。前者，我们通过Tomcat作为Web容器，而服务框架作为容器的一部分来解决。后者，我们通过Java的ClassLoader将服务框架自身用的类与应用的类进行隔离。除此之外，我们通过服务线程池隔离，分布请求合并，服务调用端的流程控制来降低系统复杂度，提高系统可靠性。详情限于篇幅，不再赘述。

最终项目成功上线，正常运行了近一年半，收到各方好评。尤其是H5课件的良好互动性，使得大量业界同行争相模仿，改用H5制作课件。还有我们的服务化方案架构被作为许多传统互联网企业系统重构的经典方案。在系统的架构设计中，我们引入了层次架构的设计思想，有效地降低了维护成本，提高了系统的开放性，可扩展性，可重用性以及可移植性。当然还是存在一些问题的。如H5课件采用http协议，易被非法劫持，嵌入广告，可以将协议修改为https来解决。还有我们采用的负载均衡算法是加权轮转算法，过于简单，常常出现资源分配不合理的现象，可以将算法改为加权最小连接数算法来解决。这些都是我在今后的系统设计和开发中需要注意与改进的地方，也是日后我应该努力的方向。

这篇论文的项目，依旧是之前那片论文的项目-在线教育系统。但是其中很多技术，其实在原有项目中是没有涉及的。

另外这篇论文与之前论文存在一个结构上的不同之处，那就是这次的核心论点只有两个分论点。不过，第二个论点-降低复杂度设计，是通过两个方面进行阐述的。这也算是论文中核心论点的一种回答方式。往往论文的核心论点，推荐使用三个分论点进行论述，而部分论文的核心论点就只能拆分为两个分论点（或者，三个论点的拆分维度，自己不熟悉）。这时候就需要灵活的转变自己的思想，将核心论点的两个分论点氛围主次论点回答，实际体现就是主论点两个段落，次论点一个段落。

既然说到这里，也说一下，如果核心论点可以拆分出多个分论点。如架构风格的层次架构完全可以拆分为接入层，应用层，服务层（基础服务层，通用服务层，业务服务层），数据接入层，数据源等。那么这种情况，我们完全可以从中挑选三点自己熟悉的部分，进行阐述。如果担心这样写，文章显得比较僵硬，就在相关位置写上“此处，我们以XXX,XXX,XXX为重点，进行论述"这样的话语即可。