论云原生架构及其应用

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了云原生架构在项目中的具体应用。系统以 Spring Cloud 微服务框架开发，分为前端 Web 服务、平台保障服务、业务服务三部分。前端 Web 服务由负载均衡与服务器集群结合，实现高并发的前台界面；平台保障服务以 Eureka 为中心，由 API 网关、服务注册中心、监控平台等构成，实现基础服务框架；业务服务划分为多个微服务，基于 Docker 容器，协同工作实现具体业务功能。最终系统顺利上线，获得用户一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为“OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

云原生架构以微服务和容器技术为代表，有服务化、强韧性、可观测性和自动化四类设计原则。通过服务化的设计原则，应用被分解为多个服务，可分别选择不同的技术，单个服务模块很容易开发、理解和维护，无需协调其他服务对本服务的影响；通过强韧性的设计原则，微服务可以分布式云化部署，负载均衡管理请求的分发，避免单机失败对整体服务的影响，以及弹性调整资源容量；通过可观测性的设计原则，能够对系统进行健康检查、指标监控、日志管理和链路追踪，提高系统运维、管理和排错能力；通过自动化的设计原则，可实现系统的自动化部署、自动化扩展伸缩、自动化运维、持续交付和集成，有效减少人工操作的工作量。

OJ 系统基于 Spring Cloud 微服务框架开发，将平台服务划分为三类，分别为前端 Web 服务、平台保障服务、业务服务。下面针对这三类服务展开具体说明。

1. 前端 Web 服务

前端 Web 服务主要提供给用户使用的界面，分为前置 Nginx 负载均衡服务器、前端网站 Nginx 集群。当用户通过网络访问系统时，首先会访问到前置的 Nginx 负载均衡服务器，负载均衡服务器会将请求转发到前端网站的 Nginx 集群，前端网站通过发起 http 请求来和后端交互，具体是通过 Ajax 方式来调用后端 REST API 接口。用户访问网站通过前置的 Nginx 负载均衡服务器来转发到前端网站集群，以起到将用户请求进行分流的作用。当前端网站集群中的部分服务发生故障时，系统仍可正常地对外提供服务。前置 Nginx 负载均衡服务器使用软件反向代理的方式来实现负载均衡，部署为路由模式，系统内部网络与外部网络分属于不同的逻辑网络，以实现系统内部与外部网络的隔离。在负载均衡算法的选择上，使用最小连接法，每当用户的请求来临时，任务分发单元会将任务平滑分配给最小连接数的前端网站节点，这样的架构以廉价且透明的方式扩展了服务器和网络的带宽，可以大大提升系统的并发量，同时保证网站前端整体的稳定性和可靠性。

2. 平台保障服务

平台保障服务用以实现后端微服务的基础框架，包括 API 路由网关、服务注册中心、服务监控组件。API 网关收到前端的请求，不会直接调用后端的业务服务，而是首先会从服务注册中心根据当前请求来获取对应的服务配置，随后通过服务配置再调用已注册的服务。当后端微服务存在多个实例时，将采取负载均衡的方式调用。服务注册中心是整个云原生架构体系的核心部分，由 Spring Cloud 的 Eureka 组件来实现，专门提供微服务的服务注册和发现功能，涉及三种角色：服务提供者、服务消费者和服务注册中心。API 路由网关、所有业务服务，以及服务监控平台组件都注册到服务注册中心。通过服务注册中心两两互相注册、API 路由网关向服务注册中心注册多个实例等方式，来实现后端整体服务的高可靠性。服务监控平台通过注册到服务注册中心，获取所有注册到服务注册中心的后端业务服务，从而监控到所有后端业务服务的运行状态信息，最后收集并展示整个微服务系统的运行状态，更进一步保证整个后端的服务质量。

3. 业务服务

业务服务按功能模块，相应划分为题库服务、评测机服务、考试服务、比赛服务、抄袭判定服务等。各服务单独打包，基于 Docker 容器，连同运行环境一起封装，根据实际情况可在一台或多台物理机同时部署多个实例，服务启动后会将自身信息注册到服务注册中心。服务间协同工作，通过松耦合的服务发现机制，动态调用对方 REST API 接口。对于压力较大的服务，如评测机服务、抄袭判定服务等，将部署为多实例集群。以在线考试功能为例，用户进入考试时，考试服务核验考生信息通过，调用题库服务，题库服务返回试题，由考试服务组合成试卷，返回前端显示。用户交卷时，提交的程序代码到达考试服务，考试服务拆分后分发给题库服务，题库服务将程序代码和测试用例送入 MQ 队列排队。然后由负载均衡机制，依次将队列中待评测程序分发给评测机服务编译、执行、判分，完成评测后，题库服务统计试题通过率，考试服务统计成绩并向前端显示。在此期间服务请求者无需了解其他服务对数据如何具体处理和分析。

总结

系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，至今已有 3000 名以上的学生用户，评测了 70000 份以上的程序代码，获得了单位同事领导和学校教师们的一致好评。在开发和试运行过程中，主要遇到了两个问题。一是跨域问题。OJ 系统前后端分离，前端通过 Ajax 访问后端服务。由于浏览器同源策略的限制，导致前端 UI 无法正常访问不同端口和 IP 的后端服务。我们利用 Spring Boot 后端的 Cors 跨域机制解决了该问题。二是评测机宕机问题。评测机服务需要执行用户提交的代码，部分用户短时间内提交了大量不安全代码，导致评测机集群中所有实例全部宕机。我们引入心跳机制、快照回滚机制，以及基于机器学习技术的预判断机制，使评测机宕机时能够在 10 秒内自动重置恢复运行，最终解决了该问题。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了合适的架构设计密不可分。经过这次云原生架构的方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断更新知识，完善本系统在各方面的设计，使整个 OJ 系统能够更加好用，更有效地服务于高校师生。

论大规模分布式系统缓存设计策略

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了大规模分布式系统缓存设计策略在项目中的具体应用。系统缓存基于 Redis 内存数据库实现，工作模式的选择上根据不同数据类型，采用了主从模式与集群模式结合的设计；通过数据持久化、数据备份计划、冗余机制和监控平台等方法实现高可用性；通过数据访问层封装同步操作实现缓存一致性，通过哈希环实现分布式算法。最终系统顺利上线，获得用户一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

系统需要支撑全校学生编程课程的学习考试，以及大量校外用户的使用，为保证系统的性能和用户体验，以缓存技术进行优化尤为重要。常见的缓存分为三种工作模式：单点模式、主从模式和集群模式。单点模式各个应用服务共享同一个缓存实例，主要瓶颈在于缓存服务器的内存大小、处理能力和网络带宽，适合缓存要求简单、数据量和吞吐量小、性能要求低的场景；主从模式将缓存的数据复制到多台机器上，数据彼此同步，可分散压力、获得更高的吞吐能力，适合数据量不大、改动频率不大、性能要求高的场景；集群模式节点间共享数据，部分冗余保证一定程度的可用性，可存储超过单个服务内存容量数据，通过增加服务器缓解数据增长和压力增加，适合总体数据量大、可伸缩性需求高、客户端数量庞大的场景。

OJ 系统使用 Spring Cloud 微服务架构开发，基于 Redis 内存数据库实现缓存，运用了多种缓存设计策略，本文着重从工作模式的选择、高可用性的设计、缓存一致性与分布式算法三个方面的问题展开论述。

1. 工作模式的选择

OJ 系统需要支撑全校学生编程课程的学习考试，以及大量校外用户的使用，对可靠性和性能要求高，因此单点模式不予考虑。经过对 OJ 系统业务需求的分析，我们归纳了三种需要缓存的数据类型。第一种是系统配置项、角色权限分配等元数据信息。这部分信息使用频率很高但数据量小，不会经常改动，我们采用主从模式来缓存，主缓存节点供系统配置模块写入数据，从缓存节点分别供具体业务服务调用。在系统启动时，自动将相关数据装入缓存。信息修改时，先写入主缓存节点，再同步更新至从缓存节点。第二种是用户登录的会话状态，这部分信息使用频率高、更新频繁，且几乎所有的业务服务都需要依赖此类数据，为保证性能与可靠性，我们采用集群方式来缓存，集中管理用户会话，各个业务服务通过数据访问层来调用具体的缓存节点。第三种是业务逻辑中一定时间内不会变化，但数据量大的信息，如试题信息、实验作业信息等，以及统计机制自动识别的高频访问信息，也采用集群方式缓存。

2. 高可用性的设计

为实现 OJ 系统缓存的高可用性，一方面我们开启了 Redis 数据持久化功能，并配置了相应的备份策略，能够有效地解决数据误操作和数据异常丢失的问题，另一方面，我们设计了一些冗余机制。在主从模式中，采用多机主备架构实现冗余，在主节点故障时，自动进行主备切换，将从节点提升为主节点继续服务，保证服务的平稳运行。如果主节点和从节点之间连接断开，重新连接时从节点会进行数据的部分重同步，当无法进行部分重同步时，会进行全量重同步。在集群模式中，采用 Cluster 技术实现冗余，每个节点保存数据和整个集群状态，负责一部分哈希槽，每个节点都和其他所有节点连接并共享数据。为了使在部分节点失效或者大部分节点无法通信的情况下集群依然可用，在集群内部使用了主从复制模型，每个节点都会相应地有 1~N 个从节点，当某个节点不可用时，集群便会将它的从节点提升作为新的主节点继续服务。缓存服务发生异常时，可通过 OJ 系统的服务监控平台产生报警，提醒运维人员及时处理。

3. 缓存一致性与分布式算法

为保证 Redis 缓存与原数据库的数据一致性，当读取数据时，会先读取 Redis 缓存中的数据，如果 Redis 缓存中不存在所要的数据，则从原数据库中读取，并同步写入至 Redis 缓存中，当写回/删除数据时，写入到原数据库中，并同步淘汰 Redis 缓存中的数据。业务服务通过使用专门的数据访问层来调用增加缓存后的数据库，使数据缓存机制对应用透明。在缓存内部实现一致性，通过分布式哈希算法来实现，为考虑到日后缓存集群的扩展需要，因此不能使用简单的模 N 哈希法，我们在 OJ 系统中采用了哈希环的算法。该算法构造一个长度为 2^32 的整数环，将 Redis 节点放置于环上，当业务服务调用缓存时，首先以服务的应用 ID 作为键值计算哈希在环上定位，然后沿顺时针方向找到最近的 Redis 节点，完成映射。当缓存服务集群中有节点故障，以及添加新节点时，只会影响上一个节点的数据，

避免了发生缓存雪崩的情况，提高了容错性和扩展性。哈希环中缓存节点过少时易发生缓存倾斜，我们通过增加虚拟节点的方式解决了该问题。

总结

我们在这次系统设计中，还使用了很多其他的缓存设计策略，这里不再一一赘述。系统在经过性能测试、负载测试、压力测试、稳定性测试后，自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 名以上的学生用户、评测了 70000 份以上的程序代码，获得了单位同事领导和学校教师们的一致好评。日常使用过程中最高出现过 600 余用户同时在线进行实验作业提交、评测的情况，基本未出现页面加载缓慢、请求超时的问题，满足了高校编程课平台的基本性能需求。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了合理的缓存设计密不可分。经过这次大规模分布式系统缓存设计的方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统在各方面的设计，使整个系统能够更加好用，更有效地服务于高校师生。

论基于架构的软件设计方法及应用

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了基于架构的软件设计方法在项目中的具体应用，着重从架构需求、架构设计、架构实现三个阶段展开介绍。在架构需求阶段，通过用户访谈、问卷调查、现场观摩、构造原型的方式全面获取了需求；在架构设计阶段通过 UML 模型中的 4+1 视图来对系统的架构进行建模；在架构实现阶段，对系统构件进行了获取、开发和组装。最终系统顺利上线，获得了用户的一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为“OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

基于架构的软件设计方法（ABSD）包括架构需求、架构设计、架构文档化、架构复审、架构实现和架构演化六个阶段。架构需求阶段明确用户对系统在功能、行为、性能、设计约束等方面的期望，包括需求获取、标识构件和架构评审；架构设计阶段根据需求生成并调整架构决策，包括提出架构模型、映射构件、分析构件相互作用、产生架构和评审；架构文档化阶段对架构设计分析与整理，产生架构规格说明书和架构质量说明书；架构复审阶段评价架构能否满足需求与实现质量属性、层次构件划分是否合理，标识潜在的风险，及早发现设计中的缺陷错误；架构实现阶段对架构进行实现，包括架构分析与设计、构件实现、组装和系统测试；架构演化阶段主要解决开发中用户需求变更问题，包括架构演化计划、构件变动、更新构件相互作用、构件组装测试与技术评审。

系统使用基于 Spring Cloud 的微服务架构开发，本文着重从架构需求、架构设计、架构实现三个阶段，论述基于架构的软件设计方法在项目中的具体应用、遇到的问题和解决方案。

1. 架构需求

在此阶段遇到的主要问题是 OJ 平台需要支撑全校编程课程的教学活动，这就需要有效、快速地全面获取需求。在需求的前期阶段，我们采用了用户访谈和调查问卷结合的方式，把需求调研团队分成了几组，分别进行需求收集。通过与课程教学组长的详细沟通，我们对 OJ 系统的主要业务功能、用户角色等有了整体、全面的了解。之后又制作了调查问卷表格，下发给各位任课教师，经过统计整理后，我们获悉了编程课程的教学活动、过程细节。在需求的中期阶段，我们在课程教学组长的协助和安排下，跟随任课教师，前往学校的计算机实验室，对目前编程课程的实验教学现场进行了观摩，了解了在传统实验教学方式中，学生和教师们的具体操作流程。在需求的后期阶段，我们基本上已经完成了大部分业务需求的收集，通过快速原型法构造出了一个仿真的 OJ 系统，供用户试用与反馈，让用户也参与到设计中，提供了工作流程方面、业务领域方面不可或缺的经验，也为以后项目通过验收提供了有力支持。

2. 架构设计

在此阶段遇到的主要问题是如何合理设计与描述软件架构。我们以 UML 模型中的 4+1 视图来对架构建模。场景视图使用 UML 模型中的用例图来进行建模，结合用户需求，在系统中划定了在校学生、任课教师、系统管理员、校外人员四类用户角色，并分别为这些角色标识了相应用例。逻辑视图使用 UML 模型中的包图来进行建模，我们经过分析，决定采用微服务架构风格开发，将系统分为前端 Web 服务、平台保障服务、业务服务三部分。前端 Web 服务由负载均衡与服务器集群结合；平台保障服务分为 API 网关、服务注册中心、监控平台，用以实现基础服务框架；业务服务分为多个微服务，实现具体业务功能。物理视图使用 UML 模型中的部署图来进行建模，系统微服务采用分布式的部署方法，每个微服务根据实际情况可在同一台物理机器或多台物理机器上同时部署多个实例。集群通过负载均衡做统一访问，部署为路由模式，系统内部网络与外部网络分属于不同的逻辑网络。在负载均衡算法的选择上，使用了最小连接法。

3. 架构实现

在此阶段遇到的主要问题是系统构件如何实现和组装。构件实现分为两种方式：获取现有构件、开发新构件。OJ 系统需要与教务管理和 OA 系统对接，我们使用开发商提供的 SDK 实现；微服务架构的基础框架，如 Spring Cloud、Eureka，以及评测机调用的编译器等，直接集成第三方软件实现。对于常见信息系统共同具备的用户管理、角色权限管理、日志记录、内容维护、消息中心等基本功能，通过取用单位过往项目开发中所积累的相应构件来实现。对于 OJ 系统专属功能构件，需要专门开发，我们使用了多种设计模式，例如通过装饰器模式来实现同一试题在实验作业、比赛、考试等多种使用场景下的扩展功能，通过策略模式来实现评测机对 C 语言、Java、Python 等多种编程语言的不同编译方法。构件实现完成后，我们根据不同业务类型，采用了不同的构件组装方式。例如考试服务从题库

获取试题信息，采用同步消息方式；程序评测服务属于耗时操作，采用异步消息方式；涉及审批流程的业务，采用基于工作流的组装方式。

总结

系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 名以上的学生用户、评测了 70000 份以上的程序代码，获得了单位同事领导和学校教师们的一致好评。在使用中系统也出现了一些问题，比如评测机服务需要执行用户提交的代码，部分用户短时间内提交了大量不安全代码，导致评测机集群中所有实例全部宕机。这是由于早期在可靠性方面的需求分析不够周全，系统没有进行充分的容错设计。我们引入心跳机制、快照回滚机制，以及基于机器学习技术的预判断机制，使评测服务宕机时能够在 10 秒内自动重置恢复运行，最终解决了该问题。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了合适的设计方法密不可分。经过这次基于架构的软件设计方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统在各方面的设计，使整个系统能够更加好用，更有效地服务于高校师生。

论软件需求获取技术及应用

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了软件需求获取技术在项目中的具体应用。在需求的前期阶段，以用户访谈和调查问卷结合的方式来进行需求获取；在需求的中期阶段，以现场观摩的方式来进行需求获取；在需求的后期阶段，采取构造快速原型的方式，持续迭代，来进行需求细化和系统演进。利用多种技术实施需求获取，有效地降低了项目风险，最终系统顺利上线，获得了用户的一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

常见的需求获取技术有用户访谈、问卷调查、现场观摩、原型化方法等。用户访谈主要对三个之内代表性客户谈话沟通获取需求，优点是灵活性好，适用范围广，缺点是时间难以安排，信息量大记录困难，需要领域知识，对机密话题敏感等，适合简单小范围的需求获取；问卷调查主要通过设计调查表来收集用户需求，优点是可短时间内廉价从大量回答中收集数据，缺点是双方未见面无法澄清提问，反馈信息不全面，无法深入问题细节，适合大范围需求收集；现场观摩主要针对较复杂、难理解的流程、操作，优点是直观清晰，缺点是效率较低，适合复杂需求的获取；原型化方法通过构造一个简易原型系统，根据用户在试用过程中意见重复修改直到满意，优点是允许用户早期交互反馈，缺点是较为费时，会误导用户对未来系统有不切实际的希望，适合需求不明确的情况。

本文主要通过前期、中期、后期三个阶段，分别论述了 OJ 系统项目在需求获取过程中所采用的具体技术方法。

1. 前期阶段

在此阶段遇到的主要问题就是 OJ 平台需要支撑全校编程课程的教学活动，这就需要有效、快速地全面概括需求。我们采用了用户访谈和调查问卷结合的方式，来进行需求获取。由于涉及编程课程众多，为了能突出重点，我们先选择了在教学方式上有代表性的部分课程，如程序设计基础、算法与数据结构、操作系统等，优先进行，然后再逐渐铺开。我们把需求调研团队分成了几组，分别进行需求收集。课程教学组长负责课程组的教学领导工作，我们采用了访谈的方式进行需求获取，通过与课程教学组长的详细沟通，我们对 OJ 系统的主要业务功能、用户角色等有了整体、全面的了解。由于任课教师具体的教学活动过程较为复杂，如实验作业、考试、比赛培训等，在课程教学组长的配合下，我们制作了调查问卷表格，下发给各位任课教师，经过统计整理后，我们获悉了编程课程的教学活动、过程细节。这种安排主次分明、详略得当，在前期起到了不错的效果，给后续的需求获取活动搭起了良好的基础。

2. 中期阶段

在前面的阶段，虽然在总体需求的获取上已经有了一定的基础，但很多流程仅仅通过访谈和问卷无法直观了解。为了防止前期需求分析的缺陷带到后续阶段，我们决定采取现场观摩的方式来进行需求获取。我们征得单位领导的同意，在课程教学组长的协助和安排下，跟随任课教师，前往学校的计算机实验室，对目前编程课程的实验教学现场进行了观摩，了解了在传统实验教学方式中，学生和教师们的具体操作流程。比如，在传统的环境下，学生按题目要求编写完程序代码，需要手动进行编译、执行，然后输入测试数据，观察是否得到预期的结果。教师批改学生提交的实验作业时，也需要手动将代码逐个拷贝到编译环境中执行、测试。在这样的过程中，大部分时间和精力都在反复地复制粘贴，随意输入的测试用例并不能全面测试出程序的设计问题，且很难进行抄袭、雷同的判定。通过现场观摩的方式，我们更清楚地了解了业务流程，为后续 OJ 系统解决传统实验教学中存在的问题起到了良好的作用。

3. 后期阶段

在此阶段我们基本上已经完成了大部分业务需求的收集，通过快速原型法构造出了一个简易的 OJ 系统，供用户试用与反馈。这个原型只是一个系统框架，很多操作是空动作，目的是向用户说明系统的功能和操作方法，以后再随着开发进程以及需求明确逐步求精。例如程序代码评测和考试功能，暂时不会进行实际的程序评测和累分，而是直接显示一个固定的评测结果和成绩，展示给用户看。整个构建过程，让用户也参与到设计中，提供了工作流程方面、业务领域方面不可或缺的经验，也为以后项目通过验收提供了有力支持。在每一次迭代过程中，通过和课程组交流，在完善需求的基础上，完善对象模型。某次试用中，课程组向我们提出一个需求“用户提交的程序代码，评测应当在较短时间内完成出结果，不能让用户等待”，但无法明确“较短时间”是多少合适。为完成这一需求，采取模拟延迟的方式，让课程组教师现场试用，明确了这个时间应该在 10 秒内，并反复修改原型，完成了这一迭代需求。

总结

系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 名以上的学生用户、评测了 70000 份以上的程序代码，获得了单位同事领导和学校教师们的一致好评。在使用中系统也出现了一些问题，比如评测机服务需要执行用户提交的代码，部分用户短时间内提交了大量不安全代码，导致评测机集群中所有实例全部宕机。这是由于早期在可靠性方面的需求分析不够周全，系统没有进行充分的容错设计。我们引入心跳机制、快照回滚机制，以及基于机器学习技术的预判断机制，使评测服务宕机时能够在 10 秒内自动重置恢复运行，最终解决了该问题。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了合适的需求获取技术密不可分。经过这次需求获取技术的方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统在各方面的设计，使整个系统能够更加好用，更有效地服务于高校师生。

论软件系统建模方法及其应用

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了软件系统建模方法在项目中的具体应用。系统采用面向对象建模方法，基于 UML 中的 4+1 视图建模，着重从场景视图、逻辑视图与物理视图三个方面介绍。场景视图以用例图分析主要用户角色与用例；逻辑视图通过包图对系统的前端 Web 服务、平台保障服务、业务服务功能建模；物理视图使用部署图描述微服务在硬件环境的具体部署方法。最终系统顺利上线，获得用户一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

常用的软件系统建模方法有结构化建模方法、信息工程建模方法、面向对象建模方法三种。结构化建模方法以过程为中心，用于分析一个现有的系统以及定义新系统的业务需求，创建的模型为数据流图（DFD），适合流程较为稳定的系统；信息工程建模方法以数据为中心，但过程敏感，强调在分析和研究过程需求之前，首先研究和分析数据需求，创建的模型为实体联系图（ERD），主要用于数据建模；面向对象建模方法将数据和过程集成到对象中，创建的模型为对象模型，通过统一建模语言（UML）描述，定义了几种不同类型的模型图，以对象的形式共建一个信息系统或应用系统。

OJ 系统使用微服务架构开发，基于面向对象建模方法中的 4+1 视图建模，建模工具为 Rational Rose，描述语言为 UML，这里着重从场景视图、逻辑视图、物理视图三个方面展开介绍。

1. 场景视图

场景视图使用 UML 模型中的用例图来进行建模。OJ 系统功能主要面向高校学生程序设计语言的在线学习、考试、比赛，我们经过分析，结合用户的需求，在系统中划定了四类用户角色，这些角色分别为：在校学生、任课教师、系统管理员、校外人员。在校学生用户是 OJ 系统学生端的主要使用者，学生涉及的主要用例有：登录系统、提交代码、自由练习、参加考试、提交实验作业、参加比赛、信息维护、查看系统帮助、交流讨论等；任课教师用户属于 OJ 系统的管理者，教师涉及的主要用例有：登录系统、信息维护、班级管理、助教管理、学生管理、考试管理、题库管理、实验作业管理、比赛管理、代码查重、论坛管理等；系统管理员拥有 OJ 系统最高的系统权限，在具备学生与教师所有用例的基础上，还增加了教师管理、评测机管理、系统管理、服务器管理等用例。校外人员用户主要面向社会以及其他高校的编程学习爱好者，仅具备登录系统、提交代码、自由练习、参加比赛、查看系统帮助五种用例。

2. 逻辑视图

逻辑视图使用 UML 模型中的包图来进行建模。我们经过分析，决定采用微服务架构风格开发，将系统分为前端 Web 服务、平台保障服务、业务服务三部分。前端 Web 服务由负载均衡与服务器集群结合，解决前台界面并发问题；平台保障服务分为 API 网关、服务注册中心、监控平台，用以实现基础服务框架，所有业务服务都注册到服务注册中心；业务服务分为多个微服务，实现具体业务功能，解决协同问题。当用户通过网络访问系统时，首先会访问到前置的负载均衡服务器，负载均衡服务器会将请求转发到前端网站的集群，前端网站通过发起 http 请求来和后端交互。API 网关收到前端的请求，会从服务注册中心根据当前请求，来获取对应的服务配置，随后通过服务配置再调用已注册的服务。当后端微服务存在多个实例时，将采取负载均衡的方式调用。后端微服务通过同步消息、异步消息、工作流三种方式协同工作，完成具体业务功能。服务监控平台注册到服务注册中心，获取所有后端业务服务的运行状态信息。

3. 物理视图

物理视图使用UML模型中的部署图来进行建模。系统微服务采用分布式的部署方法，每个微服务单独编译、打包、部署，基于 Docker 容器连同运行环境一起封装，根据实际情况可在同一台物理机器或多台物理机器上同时部署多个实例来提高系统的性能。服务启动后会将自身信息注册到已部署好的分布式服务注册中心，所有客户端请求一律进入路由网关，路由网关根据请求通过服务注册中心来进行服务发现，然后通过反向代理的方式调用具体服务。在分布式部署多个服务的基础上，前端服务器集群通过 Nginx 负载均衡服务器做统一代理访问，部署为路由模式，系统内部网络与外部网络分属于不同的逻辑网络，以实现系统内部与外部网络的隔离。在负载均衡算法的选择上，使用了最小连接法，每当客户端的请求来临时，任务分发单元会将任务平滑分配给最小连接数的微服务节点，这样的部署方法以廉价且透明的方式扩展了服务器和网络的带宽，可以大大提升系统的并发量，同时保证系统整体的稳定性和可靠性。

总结

通过 4+1 视图的场景视图、逻辑视图和物理视图等建模方式，对系统进行了详细的分析，为系统的设计和接下来的项目开发提供了有力的支持。

系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 名以上的学生用户、评测了 70000 份以上的程序代码，获得了单位同事领导和学校教师们的一致好评。不可避免的，我们在设计过程中，也存在一些问题和不足，不少开发人员在实现过程中有时还是习惯于原有的结构化设计方法，对 4+1 视图模型的使用有些抵触。而且，这些视图在应用过程中，往往不是单独使用，需要多个视图综合运用。这方面，我们还缺少相关的经验。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了合适的软件系统建模方法密不可分。经过这次 4+1 视图的建模方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统在各方面的设计，使整个系统能够更加好用，更有效地服务于高校师生。

论软件架构建模技术与应用

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了软件架构建模技术在项目中的具体应用，重点从 4+1 视图模型的场景视图、逻辑视图与物理视图三个方面介绍。场景视图以 UML 用例图对系统主要用户角色与涉及用例进行分析；逻辑视图通过 UML 包图从前端 Web 服务、平台保障服务、业务服务三个层次对系统功能进行建模；物理视图使用 UML 部署图对微服务在硬件环境的具体部署方法进行描述。最终系统顺利上线，获得用户一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

4+1 视图模型从 5 个视角来描述软件架构，分别为：逻辑视图、物理视图、进程视图、开发视图和场景视图。逻辑视图主要描述系统的功能需求，将系统分解为一系列的功能抽象，可用对象模型来代表，用类图来描述；开发视图主要侧重于软件模块的组织和管理，考虑软件内部的需求；进程视图侧重于系统的运行特性，主要关注非功能需求，强调并发性、分布性、系统集成性和容错能力，以及逻辑视图中的功能抽象如何适应进程结构，定义了具体线程执行的操作；物理视图主要考虑软件向硬件的映射，解决系统的拓扑结构、安装和通信问题。场景视图可看做重要系统活动的抽象，使其他 4 个视图有机联系。

OJ 系统使用微服务架构开发，基于 4+1 视图建模，建模工具为 Rational Rose，描述语言为 UML，这里着重从场景视图、逻辑视图、物理视图三个方面展开介绍。

1. 场景视图

场景视图使用 UML 模型中的用例图来进行建模。OJ 系统功能主要面向高校学生程序设计语言的在线学习、考试、比赛，我们经过分析，结合用户的需求，在系统中划定了四类用户角色，这些角色分别为：在校学生、任课教师、系统管理员、校外人员。在校学生用户是 OJ 系统学生端的主要使用者，学生涉及的主要用例有：登录系统、提交代码、自由练习、参加考试、提交实验作业、参加比赛、信息维护、查看系统帮助、交流讨论等；任课教师用户属于 OJ 系统的管理者，教师涉及的主要用例有：登录系统、信息维护、班级管理、助教管理、学生管理、考试管理、题库管理、实验作业管理、比赛管理、代码查重、论坛管理等；系统管理员拥有 OJ 系统最高的系统权限，在具备学生与教师所有用例的基础上，还增加了教师管理、评测机管理、系统管理、服务器管理等用例。校外人员用户主要面向社会以及其他高校的编程学习爱好者，仅具备登录系统、提交代码、自由练习、参加比赛、查看系统帮助五种用例。

2. 逻辑视图

逻辑视图使用 UML 模型中的包图来进行建模。我们经过分析，决定采用微服务架构风格开发，将系统分为前端 Web 服务、平台保障服务、业务服务三部分。前端 Web 服务由负载均衡与服务器集群结合，解决前台界面并发问题；平台保障服务分为 API 网关、服务注册中心、监控平台，用以实现基础服务框架，所有业务服务都注册到服务注册中心；业务服务分为多个微服务，实现具体业务功能，解决协同问题。当用户通过网络访问系统时，首先会访问到前置的负载均衡服务器，负载均衡服务器会将请求转发到前端网站的集群，前端网站通过发起 http 请求来和后端交互。API 网关收到前端的请求，会从服务注册中心根据当前请求，来获取对应的服务配置，随后通过服务配置再调用已注册的服务。当后端微服务存在多个实例时，将采取负载均衡的方式调用。后端微服务通过同步消息、异步消息、工作流三种方式协同工作，完成具体业务功能。服务监控平台注册到服务注册中心，获取所有后端业务服务的运行状态信息。

3. 物理视图

物理视图使用UML模型中的部署图来进行建模。系统微服务采用分布式的部署方法，每个微服务单独编译、打包、部署，基于 Docker 容器连同运行环境一起封装，根据实际情况可在同一台物理机器或多台物理机器上同时部署多个实例来提高系统的性能。服务启动后会将自身信息注册到已部署好的分布式服务注册中心，所有客户端请求一律进入路由网关，路由网关根据请求通过服务注册中心来进行服务发现，然后通过反向代理的方式调用具体服务。在分布式部署多个服务的基础上，前端服务器集群通过 Nginx 负载均衡服务器做统一代理访问，部署为路由模式，系统内部网络与外部网络分属于不同的逻辑网络，以实现系统内部与外部网络的隔离。在负载均衡算法的选择上，使用了最小连接法，每当客户端的请求来临时，任务分发单元会将任务平滑分配给最小连接数的微服务节点，这样的部署方法以廉价且透明的方式扩展了服务器和网络的带宽，可以大大提升系统的并发量，同时保证系统整体的稳定性和可靠性。

总结

通过 4+1 视图的场景视图、逻辑视图和物理视图等建模方式，对系统进行了详细的分析，为系统的设计和接下来的项目开发提供了有力的支持。

系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 名以上的学生用户、评测了 70000 份以上的程序代码，获得了单位同事领导和学校教师们的一致好评。不可避免的，我们在设计过程中，也存在一些问题和不足，不少开发人员在实现过程中有时还是习惯于原有的结构化设计方法，对 4+1 视图模型的使用有些抵触。而且，这些视图在应用过程中，往往不是单独使用，需要多个视图综合运用。这方面，我们还缺少相关的经验。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了合适的软件架构建模技术密不可分。经过这次软件架构建模的方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统在各方面的设计，使整个系统能够更加好用，更有效地服务于高校师生。

论 Web 应用系统性能优化技术与应用

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了 Web 系统性能优化技术在项目中的具体应用，着重介绍了负载均衡技术、缓存技术、数据库主从部署三种技术。通过负载均衡技术结合服务器集群，提高网站的并发能力；缓存技术基于 Redis 内存数据库，降低系统数据库压力，提高页面加载速度；数据库主从部署实现读写分离，消除了数据库的负载瓶颈。最终系统顺利上线，并且稳定运行，获得用户一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

系统需要支撑全校学生编程课程的学习考试，以及大量校外用户的使用，为保证系统的可靠性和用户体验，对系统进行性能优化显得尤为重要。传统的基于单块架构的编程考试系统早已无法支撑日益增长的用户需求，存在严重的性能问题，且很难进行扩展，因此 OJ 系统在架构设计中必须考虑到 Web 应用系统性能优化技术的应用。OJ 平台以微服务架构作为基本架构风格，基于 Spring Cloud 框架，使用 Java 语言开发，分为前端 Web 服务、平台保障服务、业务服务三个部分。前端 Web 服务由 Nginx 负载均衡与服务器集群结合，解决前台界面的并发问题；平台保障服务以 Eureka 为中心，分为 API 网关、服务注册中心、监控平台，用以实现基础服务框架；业务服务分为多个微服务，实现具体业务功能，解决协同问题。

在开发过程中，我们运用了多种性能优化技术。这里重点以负载均衡技术、缓存技术、数据库主从部署为例，介绍本系统 Web 性能优化的方案和达到的效果。

1. 负载均衡技术

前端 Web 服务主要提供给用户使用的界面，分为前置 Nginx 负载均衡服务器、前端网站 Nginx 集群。当用户通过网络访问系统时，首先会访问到前置的 Nginx 负载均衡服务器，负载均衡服务器会将请求转发到前端网站的 Nginx 集群，前端网站通过发起 http 请求来和后端交互，具体是通过 Ajax 方式来调用后端 REST API 接口。用户访问网站通过前置的 Nginx 负载均衡服务器来转发到前端网站集群，以起到将用户请求进行分流的作用。当前端网站集群中的部分服务发生故障时，系统仍可正常地对外提供服务。前置 Nginx 负载均衡服务器使用软件反向代理的方式来实现负载均衡，部署为路由模式，系统内部网络与外部网络分属于不同的逻辑网络，以实现系统内部与外部网络的隔离。在负载均衡算法的选择上，使用最小连接法，每当用户的请求来临时，任务分发单元会将任务平滑分配给最小连接数的前端网站节点，这样的架构以廉价且透明的方式扩展了服务器和网络的带宽，可以大大提升系统的并发量，同时保证网站前端整体的稳定性和可靠性。

2. 缓存技术

通过对系统业务的分析，我们发现后端业务服务在进行相关的业务逻辑操作时，会有一部分数据使用频率很高但不会经常变化，如系统基本设置和登录用户信息等。这一部分数据如果每次使用时都直接访问数据库，将会给数据库服务增加非常大的开销。为解决这一问题，我们引入了 Redis 内存数据库作为缓存。在系统启动时，将系统设置从数据库读出并缓存在 Redis 中，用户登录时，也会将用户信息缓存在 Redis 中，这样当服务再次使用这部分数据时，便可直接从效率更高的 Redis 缓存中读取。对于部分在一定时间内不会发生变化的数据库表内容，如试题信息、实验作业信息、课程信息等，我们也使用 Redis 进行了缓存。为保证 Redis 缓存与原数据库的数据同步，当读取数据时，会先读取 Redis 缓存中的数据，如果 Redis 缓存中不存在所要的数据，则从原数据库中读取，并同步写入至 Redis 缓存中，当写回数据时，写入到原数据库中，并同步淘汰 Redis 缓存中的数据。这样的架构极大地提升了系统的性能，并且缓解了页面加载缓慢等问题。

3. 数据库主从部署

网站在使用缓存后，大部分数据读操作都可不通过数据库就能完成，但仍有一部分读操作（缓存不命中、过期）和全部写操作需要访问数据库，在网站用户达到一定规模后，数据库因负载压力过高而成为整个网站的瓶颈。对于 OJ 平台来说，后端数据库存储的性能是极其重要的，所有的程序提交记录、考试答题记录、成绩统计记录都将存储到数据库中。这里就要求数据库存储有非常强的性能，本系统主要采用主从式方法部署数据库结构，实现读写分离的架构。业务服务在写数据时，访问主数据库，主数据库通过主从复制机制将数据更新同步到从数据库，这样当业务服务读数据的时候，就可通过从数据库获得数据。业务服务通过使用专门的数据持久层访问模块来访问读写分离后的数据库，使数据库读写分离对应用透明。为了缓解由磁盘硬件带来的性能瓶颈，数据库所依赖的硬件存储采用了基于 RAID6 的固态硬盘阵列，能够在硬件方面提升数据库的随机存取性能。这样的设计提高了系统数据库的运行效率。

总结

我们在这次系统性能优化设计中，还使用了很多其他的性能优化策略，这里不再一一赘述。系统在经过性能测试、负载测试、压力测试、稳定性测试后，自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 名以上的学生用户、评测了 70000 份以上的程序代码，获得了单位同事领导和学校教师们的一致好评。日常使用过程中最高出现过 600 余用户同时在线进行实验作业提交、评测的情况，基本未出现页面加载缓慢、请求超时的问题，满足了高校编程课平台的基本性能需求。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了充分的性能优化设计密不可分。经过这次 Web 应用系统性能优化技术的方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统的设计，使整个系统能够更加好用，更有效地服务于高校师生。

论基于构件的软件开发

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了基于构件的软件开发技术在项目中的具体应用。系统基于 Spring Cloud 微服务框架来进行构件化开发，在构件获取阶段通过对接现有系统、使用构件库、集成第三方软件来实现需求，在构件开发阶段使用了多种设计模式来保证构件的可重用性，在构件组装阶段使用了同步消息、异步消息、工作流方式来实现不同业务类型下构件的组合。最终项目顺利上线并稳定运行，获得用户一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

构件技术是指通过组装一系列可复用的软件构件来构造软件系统的软件技术。通过运用构件技术，开发人员可以有效地进行软件复用，减少重复开发，缩短开发时间，降低软件的开发成本。主流的构件技术有三种：CORBA、EJB 和 COM。CORBA 分为对象请示代理、公共对象服务和公共设施三个层次，优点是大而全，互操作性和开放性好，缺点是庞大且复杂，技术标准更新缓慢；EJB 基于 Java 语言，支持跨平台，提供了远程访问、安全、持久化和生命周期等机制，支持分布式计算，缺点是服务治理能力差，逐渐被 Spring Cloud 等轻量级框架取代；COM 基于 Windows 平台，功能强大、效率高，有一系列开发工具支持，缺点是跨平台性差。基于构件的软件开发过程主要分为模块划分、构件标识、构件获取、构件组装与测试、构件管理等步骤。

OJ 系统采用了基于 Spring Cloud 的微服务架构开发。这里重点从构件获取、构件开发、构件组装三个阶段说明本系统采用构件化开发的过程。

1. 构件获取

在 OJ 系统中，可复用的功能需求和非功能需求从实现方式上分为三类。第一类需要对接现有系统实现，比如需要将学校在用的 OA、教务管理两套系统中的学生信息与课程信息同步到 OJ 系统中，我们通过访问 OA、教务管理的开发商网站，联系开发商，取得了相关接口调用的 SDK。第二类是常见信息系统中共同具备的用户管理、角色权限管理、日志记录、内容维护、消息中心等基本功能，我们通过取用单位过往项目开发中所积累的构件库中的相应构件（例如 RBAC 权限管理框架）来实现。第三类需要集成第三方的软件来实现，例如微服务注册中心、API 消息路由网关、负载均衡机制、程序评测时调用的编译器、代码抄袭判定的文本比对功能等，无需另行开发，我们使用了 Spring Cloud 中的 Eureka 框架来作为微服务注册中心，Nginx 反向代理来作为负载均衡机制，GCC、JDK 等作为编译器，针对文本比对功能，我们使用了 GitHub 上的开源程序。此外我们还根据 OJ 系统的使用场景，重新开发了一些功能构件，以支撑本系统特定的用户需求。

2. 构件开发

构件的优势体现在其粗粒度的重用性，因此为最大限度保证构件的可重用性和重用力度，保持其高内聚、低耦合的特性，我们在开发中用到了一些经典的设计模式，例如装饰器模式、策略模式、工厂模式等。在题库构件中，存放了全部的试题数据，包括题目介绍、测试用例等公共内容，同一试题在自由练习、实验作业、考试、比赛等不同使用场景下，需要增加不同的扩展功能。我们使用了装饰器模式，来给试题对象动态添加不同职责。在评测机构件中，需要同时支持对 C 语言、C++、Java、PHP、Python 等多种常用编程语言程序的评测，不同语言对应的编译与执行方法存在较大差异。我们使用了策略模式，将不同语言的编译和执行方法封装起来，并使它们可以相互替换。在数据库连接构件中，因不同的业务需求，需要同时支持不同种类型的数据库，包括 MySQL、Oracle、SQL Server 三种。我们使用了工厂方法模式，有效解决了不同数据库类型对软件程序的影响，具有很好的可扩展性。这样的设计有效地体现了构件的优势。

3. 构件组装

OJ 系统中不同的业务类型，需要采用不同的构件组装方式。在本系统的开发过程中，我们用到了以下三种方式。首先是同步消息方式，以考试功能为例，用户进入考试时，考试构件需要核验考生信息，然后调取题库中的试题信息组成试卷，在此过程中，考试构件使用同步方式，依次向考生构件、题库构件请求数据，等待返回结果后再加工、组合，提供给调用者。其次是异步消息方式，以程序代码的在线提交评测功能为例，评测机构件对代码的编译、执行、判分过程，相比于其他构件的数据处理过程，属于耗时操作，这时如果采取同步方式，将引起调用者阻塞，严重影响了用户体验，甚至出现雪崩效应。因此采用了异步消息队列，代码提交后写入消息队列立即返回原程序执行，待评测机评测完成后再异步显示评测结果。最后是基于工作流的方式，能够通过图形化的界面，动态编排系统构件之间的交互和依赖关系，灵活改变多个构件之间协同工作的顺序，通过简单构件的组合，以实现复杂功能的定制。

总结

系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 以上的学生用户、评测了 70000 条以上的程序代码，获得了单位同事领导和学校教师们的一致好评。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了基于构件的开发方式密不可分。经过这次构件化开发的方法和实施的效果后，我体会到了软件元素重用对开发过程的重要价值。从软件开发的方式看，由机器语言、汇编语言，到面向过程开发、面向对象开发，再到现在基于构件、面向服务的软件开发，软件元素在两个维度上呈现进化趋势：内部功能越来越强大、全面，对外的接口越来越简单、标准。最终各领域软件可在一个统一标准下无缝组装，届时面向协作的软件开发、基于职能的软件开发等新技术都将出现，上层应用功能的实现也将变得异常简单，计算机软件将会无所不在，数字化生活、智能地球等现在还处于概念阶段的事物，将会走进现实。这个目标值得我们每一位软件从业人员为之努力奋斗。

论软件系统架构风格

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了软件系统架构风格在项目中的具体应用。系统采用了微服务架构风格，基于 Spring Cloud 框架。通过微服务架构，将系统分解为多个服务，前端 Web 服务解决前台并发问题，平台保障服务实现基础服务框架，业务服务实现具体业务功能，解决协同问题。各服务采用不同技术开发，独立部署，系统的性能、可靠性与扩展性得到了大的提升。最终系统顺利上线，获得用户一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

常用的软件架构风格分为五类：数据流风格、调用/返回风格、独立构件风格、虚拟机风格、数据仓库风格。数据流风格包括批处理、管道-过滤器；调用/返回风格包括主程序/子程序、面向对象、层次结构；独立构件风格包括进程通信、隐式调用；虚拟机风格包括解释器、规则系统；数据仓库风格包括数据库系统、黑板系统、超文本系统。此外，架构风格还有 MVC、SOA、微服务等。其中，层次结构风格将构件按层次组织，每层为上层提供服务，使用下层服务，只能接触邻层；进程通信风格将构件作为独立过程，通过消息传递调用；微服务架构将单一应用划分为多个松耦合的小服务，相互协调配合，采用轻量级通信机制，各服务可选择不同技术，支持独立部署。

因 OJ 系统需要按功能模块进行拆分，且对性能、可靠性和扩展性要求较高，我们采用了微服务架构。系统将平台服务划分为三类，分别为前端 Web 服务、平台保障服务、业务服务。下面针对这三类服务展开具体说明。

1. 前端 Web 服务

前端 Web 服务主要提供给用户使用的界面，分为前置 Nginx 负载均衡服务器、前端网站 Nginx 集群。当用户通过网络访问系统时，首先会访问到前置的 Nginx 负载均衡服务器，负载均衡服务器会将请求转发到前端网站的 Nginx 集群，前端网站通过发起 http 请求来和后端交互，具体是通过 Ajax 方式来调用后端 REST API 接口。用户访问网站通过前置的 Nginx 负载均衡服务器来转发到前端网站集群，以起到将用户请求进行分流的作用。当前端网站集群中的部分服务发生故障时，系统仍可正常地对外提供服务。前置 Nginx 负载均衡服务器使用软件反向代理的方式来实现负载均衡，部署为路由模式，系统内部网络与外部网络分属于不同的逻辑网络，以实现系统内部与外部网络的隔离。在负载均衡算法的选择上，使用最小连接法，每当用户的请求来临时，任务分发单元会将任务平滑分配给最小连接数的前端网站节点，这样的架构以廉价且透明的方式扩展了服务器和网络的带宽，可以大大提升系统的并发量，同时保证网站前端整体的稳定性和可靠性。

2. 平台保障服务

平台保障服务用以实现后端业务服务的基础框架，包括 API 路由网关、服务注册中心、服务监控组件。API 网关收到前端的请求，不会直接调用后端的业务服务，而是首先会从服务注册中心根据当前请求来获取对应的服务配置，随后通过服务配置再调用已注册的服务。当后端服务存在多个实例时，将采取负载均衡的方式调用。服务注册中心是整个后端服务架构体系的核心部分，由 Spring Cloud 的 Eureka 组件来实现，专门提供服务的注册和发现功能，涉及三种角色：服务提供者、服务消费者和服务注册中心。API 路由网关、所有业务服务，以及服务监控平台组件都注册到服务注册中心。通过服务注册中心两两互相注册、API 路由网关向服务注册中心注册多个实例等方式，来实现后端整体服务的高可靠性。服务监控平台通过注册到服务注册中心，获取所有注册到服务注册中心的后端业务服务，从而监控到所有后端业务服务的运行状态信息，最后收集并展示整个后端服务系统的运行状态，更进一步保证整个后端的服务质量。

3. 业务服务

业务服务按系统业务模块，相应划分为题库服务、评测机服务、实验作业服务、考试服务、比赛服务、抄袭判定服务、MQ 队列服务等。服务间协同工作，通过松耦合的服务发现机制，来动态调用对方的 REST API 接口。其中，对于压力较大的服务，如评测机服务、抄袭判定服务等，将部署为多实例集群。以在线考试功能为例，用户进入考试时，考试服务核验考生信息通过后，根据预先设置的参数，调用题库服务，题库服务返回试题信息，由考试服务组合为试卷，返回前端显示。用户交卷时，提交的程序代码到达考试服务，考试服务拆分后分发给题库服务，题库服务将程序代码和测试用例送入 MQ 队列排队。然后由负载均衡机制，依次将队列中排队的待评测程序分发给空闲的评测机服务编译、执行、判分，评测机服务完成评测后，将结果返回给题库服务和考试服务，题库服务进行试题通过率统计，考试服务进行成绩统计，以及向前端显示成绩。在这期间服务请求者无需了解其他服务对数据如何具体处理和分析。

总结

系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 以上的学生用户、评测了 70000 条以上的程序代码，获得了单位同事领导和学校教师们的一致好评。在开发和试运行过程中，主要遇到了两个问题。一是跨域问题。OJ 系统前后端分离，前端通过 Ajax 访问后端服务。由于浏览器同源策略的限制，导致前端 UI 无法正常访问不同端口和 IP 的后端服务。我们利用 Spring Boot 后端的 Cors 跨域机制解决了该问题。二是评测机宕机问题。评测机服务需要执行用户提交的代码，但由于部分用户短时间内提交了大量不安全代码，导致所有评测机服务全部宕机。我们引入心跳机制、快照回滚机制，以及基于机器学习技术的预判断机制，使评测服务宕机时能够在 10 秒内自动重置恢复运行，最终解决了该问题。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了合适的架构风格密不可分。经过这次微服务架构应用的方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统的架构设计，使整个系统能够更加好用，更有效地服务于高校师生。

论面向服务的架构及其应用

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了面向服务的架构在项目中的具体应用。系统划分为前端 Web 服务、平台保障服务、业务服务。前端 Web 服务由 Nginx 负载均衡与服务器集群结合，解决前台界面的并发问题；平台保障服务以 Eureka 为中心，分为 API 网关、服务注册中心、监控平台，用以实现基础服务框架；业务服务基于 Spring Cloud 开发，分为多个服务，实现具体业务功能，解决协同问题。最终系统顺利上线，获得用户一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

我们使用面向服务的架构（SOA）来设计 OJ 系统。SOA 技术参考架构包含 6 类服务：业务逻辑服务、控制服务、连接服务、业务创新和优化服务、开发服务、IT 服务管理。业务逻辑服务用于实现和执行业务逻辑，包括业务应用服务、业务伙伴服务、应用和信息资产；控制服务实现人、流程和信息的集成以及执行集成逻辑；连接服务通过企业服务总线，实现在各服务间的连接件；业务创新和优化服务用于监控业务系统运行时的业务性能，并采取措施适应变化的市场；开发服务贯彻整个软件开发生命周期的开发平台，提供全面的工具支持；IT 服务管理支持业务系统运行的各种基础设施管理能力或服务，如安全服务、目录服务、系统管理和资源虚拟化。

系统以基于 Spring Cloud 框架的微服务架构作为 SOA 的实现方式，使用 Java 语言开发，将平台服务划分为三类，分别为前端 Web 服务、平台保障服务、业务服务。下面针对这三类服务展开具体说明。

1. 前端 Web 服务

前端 Web 服务主要提供给用户使用的界面，分为前置 Nginx 负载均衡服务器、前端网站 Nginx 集群。当用户通过网络访问系统时，首先会访问到前置的 Nginx 负载均衡服务器，负载均衡服务器会将请求转发到前端网站的 Nginx 集群，前端网站通过发起 Http 请求来和后端交互，具体是通过 Ajax 方式来调用后端 REST API 接口。用户访问网站通过前置的 Nginx 负载均衡服务器来转发到前端网站集群，以起到将用户请求进行分流的作用。当前端网站集群中的部分服务发生故障时，系统仍可正常地对外提供服务。前置 Nginx 负载均衡服务器使用软件反向代理的方式来实现负载均衡，部署为路由模式，系统内部网络与外部网络分属于不同的逻辑网络，以实现系统内部与外部网络的隔离。在负载均衡算法的选择上，使用最小连接法，每当用户的请求来临时，任务分发单元会将任务平滑分配给最小连接数的前端网站节点，这样的架构以廉价且透明的方式扩展了服务器和网络的带宽，可以大大提升系统的并发量，同时保证网站前端整体的稳定性和可靠性。

2. 平台保障服务

平台保障服务用以实现后端业务服务的基础框架，包括 API 路由网关、服务注册中心、服务监控组件。API 网关收到前端的请求，不会直接调用后端的业务服务，而是首先会从服务注册中心根据当前请求来获取对应的服务配置，随后通过服务配置再调用已注册的服务。当后端服务存在多个实例时，将采取负载均衡的方式调用。服务注册中心是整个后端服务架构体系的核心部分，由 Spring Cloud 的 Eureka 组件来实现，专门提供服务的注册和发现功能，涉及三种角色：服务提供者、服务消费者和服务注册中心。API 路由网关、所有业务服务，以及服务监控平台组件都注册到服务注册中心。通过服务注册中心两两互相注册、API 路由网关向服务注册中心注册多个实例等方式，来实现后端整体服务的高可靠性。服务监控平台通过注册到服务注册中心，获取所有注册到服务注册中心的后端业务服务，从而监控到所有后端业务服务的运行状态信息，最后收集并展示整个后端服务系统的运行状态，更进一步保证整个后端的服务质量。

3. 业务服务

业务服务按系统业务模块，相应划分为题库服务、评测机服务、实验作业服务、考试服务、比赛服务、抄袭判定服务、MQ 队列服务等。服务间协同工作，通过松耦合的服务发现机制，来动态调用对方的 REST API 接口。其中，对于压力较大的服务，如评测机服务、抄袭判定服务等，将部署为多实例集群。以在线考试功能为例，用户进入考试时，考试服务核验考生信息通过后，根据预先设置的参数，调用题库服务，题库服务返回试题信息，由考试服务组合为试卷，返回前端显示。用户交卷时，提交的程序代码到达考试服务，考试服务拆分后分发给题库服务，题库服务将程序代码和测试用例送入 MQ 队列排队。然后由负载均衡机制，依次将队列中排队的待评测程序分发给空闲的评测机服务编译、执行、判分，评测机服务完成评测后，将结果返回给题库服务和考试服务，题库服务进行试题通过率统计，考试服务进行成绩统计，以及向前端显示成绩。在这期间服务请求者无需了解其他服务对数据如何具体处理和分析。

总结

系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 以上的学生用户、评测了 70000 条以上的程序代码，获得了单位同事领导和学校教师们的一致好评。在开发和试运行过程中，主要遇到了两个问题。一是跨域问题。OJ 系统前后端分离，前端通过 Ajax 访问后端服务。由于浏览器同源策略的限制，导致前端 UI 无法正常访问不同端口和 IP 的后端服务。我们利用 Spring Boot 后端的 Cors 跨域机制解决了该问题。二是评测机宕机问题。评测机服务需要执行用户提交的代码，但由于部分用户短时间内提交了大量不安全代码，导致所有评测机服务全部宕机。我们引入心跳机制、快照回滚机制，以及基于机器学习技术的预判断机制，使评测服务宕机时能够在 10 秒内自动重置恢复运行，最终解决了该问题。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了合适的架构设计密不可分。经过这次面向服务的架构应用的方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统的架构设计，使整个系统能够更加好用，更有效地服务于高校师生。

论软件设计模式及其应用

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了软件设计模式在项目中的具体应用，着重介绍了装饰器、策略、工厂方法三种模式。装饰器模式实现试题在不同场景下的扩展功能，提高功能定制灵活性；策略模式实现评测机不同语言的编译执行方法，降低模块耦合度；工厂方法模式实现数据库不同类型的统一访问，提高系统的可扩展性。这些模式改善了设计质量和开发效率，最终系统顺利上线，获得了用户的一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

一般来说，设计模式可分为创建型模式、结构型模式、行为型模式三种。创建型模式抽象了实例化过程，它们帮助一个系统独立于创建、组合和表示它的那些对象，包括工厂方法、抽象工厂、生成器、原型、单例模式等。结构型模式涉及到如何组合类和对象以获得更大的结构，包括适配器、桥接、组成、装饰、外观、享元、代理等。行为模式涉及到算法和对象间职责的分配，不仅描述对象或类的模式，还描述了它们之间的通信模式，包括观察者、策略等。

在 OJ 平台的开发中，我们综合使用了多种设计模式。本文着重对装饰器模式、策略模式、工厂方法模式 3 种设计模式在该项目中的具体应用进行介绍。

1. 装饰器模式

OJ 系统中，全部的试题数据都存放在题库服务中，包括题目介绍、测试用例等公共内容。同一试题在自由练习、实验作业、考试、比赛等不同使用场景下，需要增加不同的扩展功能。例如：实验作业功能需要限定编程语言，并且只允许用户在给定的代码框架中编辑指定的部分。为了解决此问题，我们使用了装饰器模式，给试题对象动态添加职责。在设计装饰器模式中，我们定义了四种角色：被装饰对象的基类、具体被装饰对象、装饰者抽象类、具体装饰者。被装饰对象的基类定义为 Problem 类，定义一个可动态添加职责的对象接口；具体被装饰对象定义为 ConcreteProblem 类，继承自 Problem 类，用以实现具体试题内容；装饰者抽象类定义为 Decorator 类，维持一个指向 Problem 实例的引用，并定义一个一致的接口；具体装饰者类根据不同的场景，分别定义为 TaskDecorator、TestDecorator、ContestDecorator 等类，用以实现不同场景的扩展功能。通过使用装饰器模式，我们可以在任何时候实现新的装饰器，给试题功能添加新的职责，而无需修改原有试题功能，比继承方式拥有更好的灵活性。

2. 策略模式

根据业务需求，OJ 系统的评测机需要同时支持对 C 语言、C++、Java、PHP、Python 等多种常用编程语言程序的评测。不同语言对应的编译与执行方法存在较大差异，例如：C 语言需要先调用 cl 再调用 link，生成 exe 后执行；Java 需要先调用 jdk 的 javac 编译器，生成 class 后，再调用 jre 执行；Python 等语言无需编译可直接解释执行。为了解决此问题，我们采用了策略模式将不同语言的编译和执行方法封装起来，并使它们可以相互替换。在设计策略模式中，我们定义了三种角色：环境角色、抽象策略角色、具体策略角色。环境角色持有一个抽象策略角色 StrategyCompile 接口的引用，并通过 StrategyCompile 接口，来实现一个具体的编译和执行方法；抽象策略角色定义所有具体策略类所需的统一访问接口；具体策略角色包装了不同语言编译程序的具体调用方法，给每种语言分别定义了 CStrategyCompile、JavaStrategyCompile、PythonStrategyCompile 等具体策略类。通过使用策略模式，评测机可根据不同的语言类型，选择不同的具体方法来编译执行程序代码，降低了模块的耦合度，提高了软件的可扩展性和可维护性。

3. 工厂方法模式

因 OJ 系统采取分布式架构部署，不同的模块根据不同的业务需求，选用了不同种类型的数据库，包括 MySQL、Oracle、SQL Server 三种。在程序设计过程中如果直接使用每种数据库的原生访问方法，将会使业务逻辑复杂化，并且在开发完成后很难再变更为其他数据库类型。为了解决这一问题，我们使用了工厂方法模式。在设计工厂方法模式中，我们定义了抽象产品、具体产品、抽象工厂、具体工厂四种角色。抽象产品定义了数据库操作 DataAccess 类，包括数据库连接开关、查询执行、事务等基本公用方法；具体产品继承自抽象产品类，根据数据库类型，建立 MySQLDataAccess、OracleDataAccess、SQLServerDataAccess 类，分别进行了不同的具体实现；抽象工厂建立一个 DataAccessFactory 类，定义一个通用的返回数据库操作类的方法；具体工厂继承自抽象工厂，根据不同的数据库类型，分别实现返回不同种数据库具体的操作类。通过使用工厂方法模式，可以有效解决不同数据库类型对软件程序的影响，当需要更换和添加新的数据库类型时，也不需要修改具体的业务逻辑，具有很好的可扩展性。

总结

通过采用了以上的设计模式，基本达到了预期的效果。系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 以上的学生用户、评测了 70000 条以上的程序代码，获得了单位同事领导和学校教师们的一致好评。不可避免的，我们在设计过程中，也存在一些问题和不足，不少开发人员在实现过程中有时还是习惯于原有的过程化设计方法，对模式的使用有些抵触。而且，这些设计模式在应用过程中，往往不是单独使用，需要多个模式综合运用。这方面，我们还缺少相关的经验。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统选用了合适的设计模式密不可分。经过这次 OJ 平台的开发后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统的设计，使整个系统能够更加好用，更有效地服务于高校师生。

论微服务架构及其应用

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该系统为例，主要论述了微服务架构在项目中的具体应用。系统划分为前端 Web 服务、平台保障服务、业务服务。前端 Web 服务由 Nginx 负载均衡与服务器集群结合，解决前台界面的并发问题；平台保障服务以 Eureka 为中心，分为 API 网关、服务注册中心、监控平台，用以实现基础服务框架；业务服务基于 Spring Cloud 开发，分为多个微服务，实现具体业务功能，解决协同问题。最终系统顺利上线，获得用户一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

我们经过分析，决定使用微服务架构来开发 OJ 系统项目。在单块架构设计中，所有功能模块都属于一个庞大的系统，各模块之间耦合性太强，无法进行分布式部署，单个功能模块也很难在水平方向上灵活的按需扩展。与单块架构相比，微服务架构的优势有：1. 解决了复杂性问题，应用被分解为多个服务，每个服务都有消息驱动 API 定义清楚的边界；2. 提供了模块化的解决方案，单个服务模块很容易开发、理解和维护，无需协调其他服务对本服务的影响；3. 技术自由，每个服务都可由专门的开发团队来开发，可分别选择合适的技术，只需提供标准的 API 服务；4. 性能与可扩展性强，不同类型的服务可针对性的部署在适合资源需求的硬件上，支持弹性配置。

OJ 系统的微服务架构基于 Spring Cloud 框架，使用 Java 语言开发，将平台服务划分为三类，分别为前端 Web 服务、平台保障服务、业务服务。下面针对这三类服务展开具体说明。

1. 前端 Web 服务

前端 Web 服务主要提供给用户使用的界面，分为前置 Nginx 负载均衡服务器、前端网站 Nginx 集群。当用户通过网络访问系统时，首先会访问到前置的 Nginx 负载均衡服务器，负载均衡服务器会将请求转发到前端网站的 Nginx 集群，前端网站通过发起 http 请求来和后端交互，具体是通过 Ajax 方式来调用后端 REST API 接口。用户访问网站通过前置的 Nginx 负载均衡服务器来转发到前端网站集群，以起到将用户请求进行分流的作用。当前端网站集群中的部分服务发生故障时，系统仍可正常地对外提供服务。前置 Nginx 负载均衡服务器使用软件反向代理的方式来实现负载均衡，部署为路由模式，系统内部网络与外部网络分属于不同的逻辑网络，以实现系统内部与外部网络的隔离。在负载均衡算法的选择上，使用最小连接法，每当用户的请求来临时，任务分发单元会将任务平滑分配给最小连接数的前端网站节点，这样的架构以廉价且透明的方式扩展了服务器和网络的带宽，可以大大提升系统的并发量，同时保证网站前端整体的稳定性和可靠性。

2. 平台保障服务

平台保障服务用以实现后端业务服务的基础框架，包括 API 路由网关、服务注册中心、服务监控组件。API 网关收到前端的请求，不会直接调用后端的业务服务，而是首先会从服务注册中心根据当前请求来获取对应的服务配置，随后通过服务配置再调用已注册的服务。当后端服务存在多个实例时，将采取负载均衡的方式调用。服务注册中心是整个后端服务架构体系的核心部分，由 Spring Cloud 的 Eureka 组件来实现，专门提供服务的注册和发现功能，涉及三种角色：服务提供者、服务消费者和服务注册中心。API 路由网关、所有业务服务，以及服务监控平台组件都注册到服务注册中心。通过服务注册中心两两互相注册、API 路由网关向服务注册中心注册多个实例等方式，来实现后端整体服务的高可靠性。服务监控平台通过注册到服务注册中心，获取所有注册到服务注册中心的后端业务服务，从而监控到所有后端业务服务的运行状态信息，最后收集并展示整个后端服务系统的运行状态，更进一步保证整个后端的服务质量。

3. 业务服务

业务服务按系统业务模块，相应划分为题库服务、评测机服务、实验作业服务、考试服务、比赛服务、抄袭判定服务、MQ 队列服务等。服务间协同工作，通过松耦合的服务发现机制，来动态调用对方的 REST API 接口。其中，对于压力较大的服务，如评测机服务、抄袭判定服务等，将部署为多实例集群。以在线考试功能为例，用户进入考试时，考试服务核验考生信息通过后，根据预先设置的参数，调用题库服务，题库服务返回试题信息，由考试服务组合为试卷，返回前端显示。用户交卷时，提交的程序代码到达考试服务，考试服务拆分后分发给题库服务，题库服务将程序代码和测试用例送入 MQ 队列排队。然后由负载均衡机制，依次将队列中排队的待评测程序分发给空闲的评测机服务编译、执行、判分，评测机服务完成评测后，将结果返回给题库服务和考试服务，题库服务进行试题通过率统计，考试服务进行成绩统计，以及向前端显示成绩。在这期间服务请求者无需了解其他服务对数据如何具体处理和分析。

总结

系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 以上的学生用户、评测了 70000 条以上的程序代码，获得了单位同事领导和学校教师们的一致好评。在开发和试运行过程中，主要遇到了两个问题。一是跨域问题。OJ 系统前后端分离，前端通过 Ajax 访问后端服务。由于浏览器同源策略的限制，导致前端 UI 无法正常访问不同端口和 IP 的后端服务。我们利用 Spring Boot 后端的 Cors 跨域机制解决了该问题。二是评测机宕机问题。评测机服务需要执行用户提交的代码，但由于部分用户短时间内提交了大量不安全代码，导致所有评测机服务全部宕机。我们引入心跳机制、快照回滚机制，以及基于机器学习技术的预判断机制，使评测服务宕机时能够在 10 秒内自动重置恢复运行，最终解决了该问题。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统采用了合适的架构设计密不可分。经过这次微服务架构应用的方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统的架构设计，使整个系统能够更加好用，更有效地服务于高校师生。

论高可靠性系统中软件容错技术的应用

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文以该平台的评测功能、数据库功能、自动出卷功能为例，主要论述了软件容错技术和方法在项目中的具体应用。系统通过采用集群化的形式进行应用部署，通过主备形式的数据库部署进行软件容错，通过程序设计方面进行软件的容错与避错。以上措施对提高系统的可用性、安全性和可扩展性方面起到了很好的效果，满足了系统的性能需求，保证了系统的稳定运行，获得了用户的一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

OJ 平台以微服务架构作为基本架构风格，基于 Spring Cloud 框架，采用 Java 语言开发，将平台服务划分为三类，分别为核心服务、平台 Web 服务、平台保障服务。其中核心服务主要分为程序评测服务、试卷判分服务、比赛判分服务、代码相似度比对服务等。平台 Web 服务主要提供给用户使用的界面。平台保障服务主要包括 WebAPI 框架、平台报警服务、MQ 消息中间件服务等。对于核心服务、平台 Web 相关的服务必须要能保证在考试、竞赛期间的稳定运行，对软件的可靠性要求非常高，所以在 OJ 平台中的这些核心服务就必须具备一定的容错能力，在某个服务运行时出错的情况下不能影响到整个系统的正常运行，这就要求在软件架构设计中必须考虑到软件容错技术的应用。

提高软件系统可靠性的技术主要分为：容错技术、避错技术。容错技术主要通过冗余实现，分为结构冗余、时间冗余、信息冗余、冗余附加。结构冗余又分为静态冗余、动态冗余和混合冗余。软件避错技术主要有 N 版本程序设计、恢复块方法和防卫式程序设计。

结合 Web 软件的性质，我主要采用了集群技术、数据库主从方式和程序设计方面来进行软件的容错与避错处理。下面就从以上三方面详细讨论我所采用的容错技术和方法。

1. 集群技术

平台中各服务如果在运行时部署在一台服务器上，那么当服务器故障，整个平台将不能再提供任何服务，所以一般规模的应用需要采取集群的部署方式。以 OJ 系统中的评测机模块为例，程序评测属于耗时操作，用户提交的程序需要统一存入 MQ 队列中排队，然后由评测机依次进行读取和判分。如果评测机只部署了单个应用，那么当评测机因执行用户提交的不安全代码导致宕机时，后续所有待评测程序都将被阻塞。通过多机同时部署相同功能的评测机集群，再结合服务管理中心的软负载均衡功能，在系统正常运转时，多个评测机并行工作，可提高评测效率，在部分评测机宕机时，评测工作交由其他正常评测机分担，以留出供故障评测机重置恢复的时间，防止了因单台评测机宕机致整个平台不可用的问题出现，保证了业务的连续性，提高了系统的可靠性。

2. 数据库主从部署

对于 OJ 平台来说，后端数据库存储的稳定性是极其重要的，所有的程序提交记录、考试答题记录、成绩统计记录都将存储到数据库中。如果数据库在运行过程中频繁宕机，那么带来的问题是不可容忍的。这里就要求数据库存储有非常高的可靠性，同时有很强的容错性，在这里主要采用主从式方法部署数据库结构，实现读写分离的架构。在不出问题的情况下，对于一些时效性要求不高的场合，从库可以分担一部分读流量，当主库发生读写问题时，可快速由其他的从库升级为主库，继续服务，达到容错的效果。此外，OJ 系统还增加了数据库宕机报警的功能，防止宕机的数据库实例过多以至于并发高的情况下没有可用的从库可升级为主库，提高了平台的可靠性。数据库所依赖的硬件存储采用了基于 RAID6 的磁盘阵列，最高可容许两块硬盘同时发生故障，降低数据文件丢失的可能性。

3. 程序设计方面

根据以往的设计经验，系统的不可靠大部分是由于程序内部的设计、网络请求参数的配置或者连接池参数的配置不当所导致的。所以通过程序设计方面进行软件的容错是非常重要的。最普遍的就是防卫式程序设计，例如平台中的考试服务，需要根据预设的试卷规格，调用题库服务中的题目信息来自动生成试卷。当大量考生同时进入考试功能时，题目服务在被调用的一刻如果出现网络拥塞或者丢包，这时候考试服务必然会收到抛出的错误信息，如果没有通过恰当的容错处理，那么一定会给考生显示进入考试失败的错误。这里我采用了 try-catch 机制加 10 次重试的容错处理机制，就解决了进入考试时因网络原因导致进入考试失败的问题。

总结

通过采用了以上容错技术的方法和措施后，评测机模块达到了六余度，数据库达到了四余度，考试出卷功能等待时间控制在了 5 秒内，使整个系统达到了可靠性和实时性的要求。

系统自 2019 年 10 月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 以上的学生用户、评测了 70000 条以上的程序代码，获得了单位同事领导和学校教师们的一致好评。

然而在使用过程中，系统还是出现了可靠性方面的故障。比如评测机虽然使用了集群部署，但由于提交作业的学生初次学习编程、对试题理解有误，短时间内提交了大量不安全代码，导致所有评测机实例全部宕机。我们对问题分析和试验后，将原部署在物理机上的评测机改为部署在虚拟化平台，结合心跳机制，自动回滚快照重置失效实例，省去了物理机系统重启的时间，将单个评测实例的恢复时间缩短到了 10 秒内。另外，我们使用了机器学习技术，对代码进行预判断，有宕机风险的代码将会被分流和延后处理。最终这个问题得到了解决。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，与系统的可靠性设计密不可分。经过这次软件容错技术的方法和实施的效果后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统在可靠性方面的设计，使整个系统能够更加好用，更有效地服务于高校师生。

论信息系统的安全性与保密性设计

摘要

2019 年 3 月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

本文从网络硬件层、数据层、应用层三个方面，论述了针对该系统的安全性与保密性问题，以及所采用的技术手段和解决方案。网络硬件层设置硬件防火墙，解决病毒木马与外部攻击的隐患；数据层设置数据加密与容灾备份机制，解决数据泄露丢失的隐患；应用层统一采用 RBAC 授权机制等方案，解决越权操作的隐患，提高了整个系统的抗风险和安全保密能力。最终系统顺利上线，获得了用户的一致好评。

正文

笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019 年 3 月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为 “OJ 系统”），以取代原有传统的编程上机考试平台。

系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于 ACM 竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

OJ 系统中存储着学生成绩、学生选课数据、试题数据等大量重要信息，确保系统的安全性与保密性显得尤为重要。系统在安全性与保密性方面主要面临以下问题：网络安全隐患，包括病毒木马、外部攻击等，缺乏主动的网络安全防御机制。数据库安全隐患，例如数据库访问控制不严、敏感数据以明文存储、未制定容灾备份计划等，容易造成系统数据泄露、丢失、篡改。应用安全隐患，包括权限管理混乱、权限设置不够合理等，容易出现人为操作导致数据错误或丢失，威胁到系统的信息安全。

如果敏感数据不经加密传输，入侵者可通过网络嗅探工具获得用户的账号和口令。在业务逻辑中对用户提交数据的合法性没有判断或过滤不严，攻击者可在事先定义好的查询语句的结尾上注入额外的 SQL 语句，在管理员不知情的情况下实现非法操作，欺骗服务器执行非授权的任意查询。

因此，我们在网络硬件层、数据层、应用层三个方面，对提高系统的安全性与保密性，做了充分的设计。通过在网络硬件层设置硬件防火墙，数据层设置数据加密与容灾备份机制，应用层统一采用 RBAC 授权机制等方案，提高了整个系统的抗风险和安全保密能力。

1. 网络硬件层安全方案

网络和硬件是整个系统运行的基础，也是很多外部攻击的主要途径，所以 OJ 系统在这一方面需要进行严密合理的规划。为解决病毒木马与外部攻击的隐患，我们对网络拓扑结构划分为外部网络、内部网络与 DMZ 三个部分。在外部网络和内部网络之间设置了硬件防火墙，主要是防止外部的恶意攻击。在防火墙之后，为加强对病毒、木马入侵的防范效果，又设置了硬件的防毒墙，实时保证能够拦截与查杀符合最新特征库的病毒、木马。为防止校外访问者通过 DNS 服务器直接访问到应用服务器的目标地址，在校内网络中又增加了反向代理服务器，对外只暴露代理服务器的虚拟 IP，更好地减小了应用服务器被攻击的可能性。应用服务器与数据库服务器做了物理隔离，内部人员也无法通过 IP 直接访问，只能通过堡垒机由服务器管理员进行操作，阻断了外界通过内部客户端代理的访问对服务器造成攻击。对于一些核心的 FTP 服务器、DNS 服务器、Web 服务器都规划到 DMZ 中。

2. 数据层安全方案

数据是整个平台业务运转的核心，其中涉及大量学生成绩、试题数据等重要信息，安全不容忽视。为解决数据泄露、数据丢失的隐患，系统主要从数据存储、数据访问和数据容灾几方面进行了规划。由于系统主要存放的考试题目与学生成绩数据对保密性要求高，为了防止泄露，我们在存储时进行了加密处理。虽然在处理过程中会有性能损失，但也因此提高了数据的安全性与保密性。系统的数据库软件采用 Oracle 作为后台存储，其公司的商业口碑极佳，对数据存储的安全提供强大保证。为了防止数据信息的泄露，加强了访问权限的限制，在数据库的访问权限上，都根据不同的角色进行了详细划分，包括对数据库、表、索引、记录的增、删、查、改进行了限制，严格禁止非法用户对数据库恶意破坏。同时，在数据库本身，也制定了基于全量、增量、差量的按周备份计划，保证每时每刻的数据都可以及时恢复。在物理存储上也做了本地多机房的容灾备份，充分保证了数据抗突发风险的安全。

3. 应用层安全方案

由于 OJ 系统采用 MVVM 前后端分离架构，通过 RESTful 风格的 WebAPI 通讯，所以在用户认证、接口传输等方面做了充分设计。用户认证方面，因系统用户涉及多种类型，如学生、授课教师、课程组、参赛选手、举办方、系统管理员等，统一采用 RBAC 授权机制，既可增强系统安全性，满足业务需求，也可减轻权限信息维护的负担。在登录界面上，不仅需要提供用户名+口令，同时设置了验证码和动态口令，通过手机或邮箱接收，杜绝非法破解口令登录，保证整个系统认证环节安全。密码提交和存储过程中，一律通过 MD5+SHA256+随机 SALT 哈希加密，即使被拖库也不会泄露密码。接口设计是整个系统通信的安全保障，为防止外部人员恶意调用与窃取信息，使用了 https 传输协议和令牌机制，保障每次通讯传输安全。即在登录后，通过加密传输的用户信息和时间戳获取到 token 令

牌，在后续每次通讯过程中，服务器端都会校验 token，充分保障接口调用安全。在对数据库进行增删查改调用时，统一使用带预处理功能的 ORM 中间件实现操作，可过滤 SQL 注入攻击，防止对数据库执行非授权的任意查询。

总结

系统自 2019 年 10 月正式上线后，已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有 3000 以上的学生用户、评测了 70000 条以上的程序代码，获得了单位同事领导和学校教师们的一致好评。在任何情况下使用系统，基本没有信息泄露的情况发生。但在使用过程中一些用户反映，手机端操作编辑题目与保存成绩有时会需要等待，我们分析发现该问题是因为编辑题目信息与保存成绩涉及数据量较大，且手机硬件运行效率比 PC 低，导致前端加密传输更加耗时，通过优化数据结构、减少数据传输量、选择效率更高的加密算法等方法改善了这个问题。

实践证明，OJ 系统项目能够顺利上线，并且稳定运行，保障了信息安全，与系统在安全性和保密性方面的设计密不可分。系统安全是一个永久的话题，我们对系统安全性的完善是一个持续、迭代的过程，在未来还会不断地完善本系统安全方面的设计，改善缺陷与不足，使整个 OJ 系统能够更加好用，更有效地服务于高校师生。

2020 软考系统架构设计师总结

0. 背景

一开始要参加软考我是拒绝的，你不能说我一个游戏 UP 主当得好好的，毕业工作也有几年了，突然就叫我重新回到那种学习考试的生活吧。后来是 2019 年夏天时候单位发了一个专业技术职务的通知，有软考职称可以聘任相应的技术岗位，我看待遇还不错，再加上领导催得紧，那就去先考个中级看看吧。单位是做运维工作的，大部分同事中级考的都是信息系统管理工程师。

本人大学读的是计算机科学与技术专业，大三时候分软件和网络两个方向，自己一直想学网络，但被系里安排统一选了软件。本想报网络工程师，补补网络知识，一看题目要默写配置命令啥的，完全记不住，顿时觉得还是早点拿证要紧，就报了基础相对好一点的软件设计师。

初看试卷，都是学过的计算机专业课，以前混过 NOIP 竞赛，算法编程应该也没问题，结果做完一对答案，上午下午都扣了 30 多分，全挂。主要问题是几年没看专业课，很多东西记岔了，自己擅长的编程部分又想得太多（比如下午题 Java 面向对象，写了一堆强制类型转换，以及和 C#的关键字弄混）。所以紧急买了本《32 小时通关》，刷了一遍知识点，又做了配套的真题，之后就渐渐掌握了套路。考试时候很快就把卷子写完了，剩下来的时间都在算 01 背包问题的答案，最后直接用二进制枚举法暴力破解出来了，成绩 68/71。

1. 准备工作

中级成绩出来后，我想难得复习了一遍计算机专业课，趁着没忘，干脆一鼓作气把高级也考掉吧。同事里有其他考高级的，他们报的资格都是高项，都说高项简单（但也没一个过的）。我对项目管理实在不感兴趣，就把每个高级资格的卷子都找到看了一下，最后决定考系统架构设计师。

首先是买了一大堆书，然后在网上也找了很多资料，个人推荐 github 上面的 xxlllq/system_architect 资料库，很全，也有交流群可以加。

大概是从 12 月底到 4 月底，仔细读了遍《系统架构设计师教程（第 4 版）》（希赛版，清华官方版的 09 年就没更新了太老），看完了整本书。之后又看了遍《系统架构设计师考试全程指导（第 2 版）》，从 4 月底到 7 月中看完，正好到了报名时间。这段时间细看书，个人只是以兴趣为目的去了解，对考试没多大帮助，这么厚的书也记不住，题目还是不会做，没耐心的可以跳过这一步。

2. 选择题

这个部分是以看视频和刷题为主。视频主要看资料库里面的 179 讲（可以二倍速播放），基础知识至少看两轮，软件工程、架构设计两章至少看四轮。再去了解了一下 AI 和区块链的知识，B 站上有科普视频。

刷题方面，安装了三个 APP，分别是希赛网、软考通、51cto 题库微信小程序。希赛功能比较全，但用的时候他们的老师会隔三差五打电话拉人报班，没有报班打算的话慎用；软考通支持打乱选项顺序，以及按照正确率、已做/未做筛选题目；51cto 小程序每天题目都是固定知识点。三个 APP 加起来，每天一共刷 30 道题，以概念题和推理题为主，遇到数学英语的直接跳过。刷到最后，差不多每次都能全对，就可以开始做真题了。

真题里面的题目，基本上在刷题 APP 里面都做过了，主要目的是摸清楚试卷，哪些知识点占多少篇幅。个人认为高级考试中的计算机基础知识、操作系统、组原、网络等，反而比中级的容易。几套真题做下来，选择题基本上都在 68 分左右，错的几道题都是数学英语，这部分自己实在搞不明白就放弃了，其他题目练好了一样能补回来。

今年考试题目比较偏，考了好多行业知识，什么 linux 运维、软件测试、概率法求体积、比特币双花攻击等等，好在自己积累够，提前半小时交卷，最后考了 63 分。所以平时多关心行业动态，知道什么技术流行也是很重要的。

3. 案例题

这个部分从 09 年的真题开始做的时候，差点被劝退了。上来就是那种“请用 300 字解释 XXX 概念、描述 XXX 原理”。从小就是背书弱鸡，背诵默写是要我命的事。好在看了后面几年的题，16 年~18 年左右的，就以图形填空为主了，而且很多能在文案里找到答案。

第一题必做，架构风格评估几乎每年都有，做几道题了解下质量属性、风险点、非风险点、权衡点之类的特征。嵌入式基本放弃，剩下的有 UML 数据流图、状态图、数据库设计等等，和软件设计师套路比较像。最近几年的考试趋势偏向于互联网应用，考 CDN、主从数据库、集群、负载均衡、Redis 缓存、Spring 框架等等，这部分教程里没怎么讲，个人推荐看《大型网站架构》之类社会上的架构师书来了解，越新越好。真题推荐蓝色封面的《13-18 年解析》资料，案例题做完回顾重点看图，要搞明白为什么这样填写答案。大段背概念的那种反正我记不住，也就看看了。

今年的案例题考得很深很细。第一题除了效用树选择填空之外，还问了管道和仓库风格的对比和选择。根据刷选择题的经验，现代编译器主要用仓库风格，就写了仓库风格更合适。具体对比细节，完全没想到会考这么细，都没怎么背，就干脆用反义词写了一通。后面的选做题，嵌入式不看，Redis 考了数据类型、雪崩和淘汰策略，不是做这块开发的人还真答不好，所以选做了第二题和第五题。第二题是快递单数据库设计，问超类的概念，以前学数据库都没印象用过，但看题目里收件人和寄件人都有姓名、地址和手机号，超类估计是指共同属性，就这么写了。第五题是 SSM 架构，平时没怎么做过 Java 开发（我主攻 PHP 和.NET），没见过这玩意，猜想应该和三层架构差不多。题目里说了是用的 SpringMVC + Spring + MyBatis 技术，所以架构图里 SpringMVC 下面填 Spring，再下面填 MyBatis。这三个确定了，对应右边从上到下分别填表示层（View Layer）、控制层（Controller Layer）、持久层。Persistent 这个单词我考试时候都还不认识，但 XX Layer 只剩下这个了，应该就是持久层，所以填了上去。持久层要连数据库，存在连接池，就填 Connection Pool。剩下一个和 html 并列的就只能是 JSP 了。

最终案例题考了 61 分，真的觉得有运气和发挥的成分，考的东西不会，但题目的提示恰到好处，按自己的猜想去写，就答对了。

4. 论文题

软考高级要 2 个小时手写完将近 3000 字的论文，基本上不会给你现场组织内容的时间，只能按照考点提前准备，然后考试时套用素材。架构师考试论文有六个方向：系统建模、系统设计、架构设计、分布式、可靠性、安全保密性。论文可以早点开始准备，查资料编故事也有助于理解知识点。

论文纸一共 6 页 A4 纸。个人使用的论文格式：摘要（130 字介绍项目+170~180 字概括论点和总结） + 正文（450 字介绍项目背景+350 字回答概念问题和过渡段落+1200 字 3 个论点+450 字总结项目效果不足和展望）。对应到论文纸上，第 1 页写摘要，第 2 页项目背景，第 6 页写总结，然后从第 5 页最下面倒着往上数，每 20 行一个论点，一共 3 个论点，最后第 3 页剩下来的部分回答概念和过渡段落。考试拿到答题纸把行数数好，每个地方用笔在格子里轻轻地捣一个点。这样每段写多少篇幅都会有个大概。

论文涉及项目的选择，自己做过不少系统，像《健康档案》《库房管理》就基本的 CRUD 太简单了，《组原仿真实验》单机软件不好写架构，最后就选择了我大学时候做的大创项目《OJ 系统》（Online Judge，现在学校还在用）。因为论文里评分标准说写大学生实习项目要扣分，所以就给自己虚构了一个在高校计算机专业教学平台研发单位的工作。自己做的真实 OJ 系统是基于 ThinkPHP 的单块架构，写出来太减分，就吹成了 Spring Cloud 微服务架构。所有论文都围绕这个项目来写，也参考了一些类似题目学术论文的素材。

考试前一共准备了 13 篇论文（链接附后）。虽说是 13 篇，但写到后面就发现，很多内容可以直接复用前面写好的论文，重组一下就又是新的一篇。又从资料库里下载了论文答题纸模板打印出来，每篇论文都练习了一遍手抄。练习时间选下午 3:20-5:20，跟考试时间一样。一开始手抄差不多正好用完两个小时，再往后面速度就提升上来了，也没那么累了，基本上 90 分钟左右能写完一整篇论文，平均下来 15 分钟写完一面，什么时间写到哪里都会有个印象。

今年考试的论文题一度怀疑是不是和系分搞反了，拿到卷子一看四道题，企业集成、缺陷管理、云原生、数据分片，顿时一万头羊驼奔腾而过。我考前写完 13 篇论文后犹豫了一下要不要写企业集成，觉得我准备的够多了应该够用了，一念之差没写，结果还就考到了……后面三个题目见都没见过。只能先不选题了，把摘要第一段，还有正文里面介绍项目背景、收尾总结先给写了。选题无关的内容写完后，重新仔细读题，在云原生的题目里看到了一句 “云原生架构以微服务和容器技术为代表” （没有这句话提示我论文准挂），就选了云原生，在过渡段落里，根据自己的理解回答了题目问的四个设计原则，然后直接套用考前准备的微服务论文，加了基于 Docker 容器分布式部署的内容。

考完查了下云原生架构，发现我论文里破绽不少，担心了许久。最终论文考了 52 分，还好过了。所以准备论文不要有侥幸心理，一定要能写尽写（数据湖那种一般人写不来的除外），哪个题目犹豫了没写，没准考试就考到了。

5. 总结

历时 15 个月，相继通过了软考中级和高级，算是大学毕业后在计算机方面取得的最高成就了吧。中级基本上是吃老本，不多说。高级原本计划三次考过，这次能一次通过，除了运气成分也有临场发挥的成分吧。

架构师的题最近几年的越来越天马行空、接近互联网，早就超脱了教程的范畴，这部分就需要平时多关心行业动态，常见的技术不要求实际做过，但看到名字要能知道这个技术是干啥的，以及怎么和其他技术配合。

按照现在这样的题目设置，死记硬背基本上是行不通的，相关知识还是要以理解运用为重，刷题就很有必要。比如我费了很大劲，直到考试前架构评估 ATAM 和 SAAM 的步骤都还背不下来，设计模式也列举不全，质量属性的定义解释也记不住，ABSD 光记得六个步骤的名字，每个步骤干啥事也记不住，案例分析考的 SSM 架构就没见过。不过刷题多了，选择填空我还是能凭感觉选对，问答题如果直接叫我写，那就答不上来了。