电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

学 士 学 位 论 文

BACHELOR DISSERTATION

论文题目作业帮高并发场景下的稳定服务端功能实现

学生姓名 尹俊民

学　　号 20132204020008

学　　院 信息与软件工程学院

专　　业 软件工程（大型主机）

指导教师 刘 玓

指导单位 作业帮教育科技（北京）有限公司

2017 年 5 月 20 日

摘 要

随着作业帮业务线的不断壮大与发展，随之而来的是越来越多的产品需求。实习期间我作为公司平台后端研发，主要参与了作业帮 app 内的黑板报、第三方登录、微课等功能模块的实现。项目开发过程中，我在百度的 ODP 框架体系内，使用 MySQL、memcached、redis、消息队列等技术开发了黑板报的 feed 流、短视频、定时上线、弹幕、点赞、腾讯新闻抓取与审核等一系列功能。在日活千万的作业帮 app 平台每天经受着超高并发的用户请求。同时还负责维护现用功能的稳定性，不断优化现有的黑板报数据结构、缓存体系来优化接口的访问效率提升用户体验。

关键词：服务端，高并发，稳定性，请求，黑板报

ABSTRACT

With the continuous development of Zuoyebang’s business, followed by more and more products demand .During the internship ,I worked as a back-end developer of the company platform apartment, mainly involved in the develop of app blackboard newspaper, third party login, micro class and other functional modules. Project development process, In Baidu's ODP framework system, I used MySQL, memcached, redis, message queque technology to developed a blackboard newspaper feed stream, short video, regular on-line, barrage, point praise, Tencent news crawl and Audit and a series of functions. Live in the app with tens of millions user, they are experiencing every day with high concurrent user requests. And I am also responsible for maintaining the stability of the existing functions, and constantly optimize the existing blackboard data structure, cache system to optimize the interface access efficiency to enhance the user experience.

Keywords:server, High-Concurrency, stability,request,blackboard

目 录

第一章 毕业设计（顶岗实习）概况

1.1 实习单位与实习岗位概况，对企业所属行业的认识

作业帮教育科技（北京）有限公司是百度旗下的教育科技公司,原本隶属于百度知道的架构体系下。2013 年 12 月作业帮官方内测 qq 群在百度内部成立，经过两年的发展,作业帮迅速占领市场。作为教育市场最有竞争力的 app,2015 年 5 月，教育部召开信息化教学调研会上，百度“作业帮”成为示范案例。同年作业帮用户数突破 6000 万。为了作业帮更好的发展，百度知识搜索体系负责人侯建斌向李彦宏申请将作业帮独立运营。经过一段时间的准备，的 2015 年 9 月 2 日，作业帮作为百度“航母计划”的一部分宣布分拆， A 轮由红杉资本、君联资本联合投资. 2015 年 10 月 31 日，作业帮教育科技（北京）有限公司正式独立运营。公司 16 年又获得 6000 万美元 b 轮融资。作业帮主体产品作业帮 APP 是面向全国中小学生的移动学习平台，也是习题搜索、高效练习和学习沟通的综合学习工具。[1]如今已有拍照搜题、同步练习、老师答疑、语音搜题、作文搜索、求助学霸、在线直播课等多项业务功能。

公司所在的互联网教育行业是当前缓解或解决教育资源分配不均的最佳途径，国家对互联网教育继续给予大力政策扶持。2014 年 11 月，教育部等五部门联合下发《构建利用信息化手段扩大优质教育资源覆盖面有效机制的实施方案》。2015 年 4 月，教育部出台《关于加强高等学校在线开放课程建设应用与管理的意见》，为慕课发展营造良好的政策环境。2015 年 7 月，《国务院关于积极推进“互联网+”行动的指导意见》提出加快发展基于互联网的教育新兴服务，探索新型教育服务供给方式。这些政策的出台，为互联网教育营造了良好发展环境。

我实习所在作业帮平台服务端研发岗是为作业帮整个教育平台提供基础架构服务的重要部门。部门负责了公司后端服务的的基础框架搭建，完成了公司的用户系统、消息系统、广告系统、检索系统、支付系统等核心模块的开发。同时负责公司核心产品作业帮 app 版本迭代过程中的服务端功能开发，负责分析产品方提出的需求，然后对相关功能进行技术调研、架构设计并整理出相关方案文档。考虑功能的应用场景、并发量等因素进行服务端代码编写，接口设计、数据库设计、缓存设计、集群服务调度。并在完成功能代码后，进行联调，功能测试、回归、上线。

并负责上线之后的数据回归等工作。

1.2 实习项目课题背景、价值、意义以及国内外研究现状

在 2016 年，公司经历了飞速的成长， 整体团队人员扩张了数倍， 公司整体业务线从单一的拍照搜提扩张到支持多个多样化在线教育模块。如今作业帮作为互联教育领域的领军者，日活已达千万级别，瞬时压力会达数万 qps。检索数据表每日新增数千万行，在这种业务场景下，同时又是资源有限的环境下必须思考如何靠小规模集群承载大量请求。如今国内外大规模互联网公司都面临着高瞬时并发的挑战。国内顶级服务端设计如淘宝在高峰期间都有着单个上百万 qps 的压力。国外如谷歌、推特等公司靠着几十万上百万的服务器集群承载着世界各地发送来的请求。虽然面对高并发场景构建稳定的服务端的业务场景已经有很多的公司已经完成的很好，大公司的特定业务线都有着较为成熟的模型去应对所谓的高并发场景，但由于每个公司的业务场景的差异性，再优秀的架构都不具备通用性，需要根据每种应用特点针对性来设计，且不同公司对于高并发的核心技术都并非透明，故在集群资源的有限性等因素下，针对所在的公司特定化设计出特定的架构维持业务的可靠性、保证足够低的成本是非常具有挑战性的。

1.3 实习项目整体执行完成情况概述

在黑板报的多次改版过程中，由最初的简单的文章列表功能进行多次功能的改进和优化。最初的第一个改版实现了文章详情页的点赞，弹幕功能，第二次改版实现了获取腾讯天天快报的第三方内容作为独立的栏目展示，丰富了黑板报列表并实现列表的多种样式——多图、组图、图文的混排列表形式。第三版实现了作业帮原创内容与腾讯等第三方内容的混排功能，极大的增加了用户的停留时长。并在第四次改版增加发布短视频内容，并新增了百度秒懂百科的数据源再一次丰富了黑板报模块功能，加强了用户体验。并在几次改版过程中，还在后台添加了文章定时上线下线等功能。技术层面上在热点资源处都加上了一层缓存，极大的减少了数据库的压力，并随着几次的改版，将内容的存储数据结构重构，加强了数据格式的可拓展性。在不断迭代开发中，黑板报的点击、停留时长等数据都较之前有了巨大的提升。

关于作业帮的账号系统重建则是实习期间经手的最大规模项目，直接影响了

作业帮整个用户体系的过亿用户。作业帮最初所有用户都被整合成为手机注册，故作业帮核心用户表为两个表，表 a 是以手机为主键的存储用户 session，uid,password,表 b 是以 uid 作为主键存储用户信息的表。在增加了第三方登录后，用户的 session，password 就并非和手机有强关联关系。所以需要将 session、password 等字段拆为单独的表进行存储。需要进行的进行表数据迁移工作涉及全用户系统的上亿用户，涉及的接口包含登陆、注册、修改密码、单点登录、忘记密码等功能。所有的接口不能影响线上业务线，故在接口不变更的情况下重构底层服务端逻辑。重构完成后进行线下测试、内网测试、小流量测试、一致性测试后才能开启全量。项目前后经历较长时间，并于三月底上线，线上 0 用户反馈前后出现因账号系统迁移导致了账号不一致、数据错误等问题，线上回归正常，成功实现了账号迁移。

作为一个教育类 app，为用户提供优质的体验同时谋求良好的变现之道是必要的。作为互联网公司通用的变现途径之一就是通过广告。接入第三方广告平台的广告，请求第三方所提供的接口就会返回相应的广告资源通过站展现点击，就可从中获得收益。但由于广告平台投放的广告质量并非可控，作为教育平台不适宜出现的广告种类非常多。故在接入广告平台的基础上拓展作业帮现有的广告平台来源并在与广告平台对接中新增一层审核接口，已保证投放广告的优质性。审核功能包括广告审核、广告主审核。所有未被审核过的内容被请求过后不会被直接展现而是进入后台页面进行人工审核。该功能重构了广告请求接口，并新增了相应的后台服务。上线过后，广告的质量得到了把控，并通过将审核结果缓存，在添加了审核之后也并未给服务端增加过多压力。

第二章 开发环境及关键技术介绍

2.1 ODP&RAL&NMQ

2.1.1 核心框架 ODP

ODP（见图 2-1）是百度发布的在线业务开发平台，面向全百度的在线业务支撑平台，专注于总结大社区类业务模式，其提供了标准的 webserver 环境、标准 php 环境、AP 框架、SAF 社区业务框架、基础库、RAL 资源访问层、KSARCH 通用服务等组件，统一业务的逻辑和部署结构，为测试、运维等提供一致的视图。[2]ODP 作为百度内部 PHP 开发的标准框架，覆盖了公司大部分把 PHP 作为业务开发语言的团队，影响超过千人以上 PHP 工程师的开发，对 RD/QA 的学习、开发和测试效率提升 100%以上。

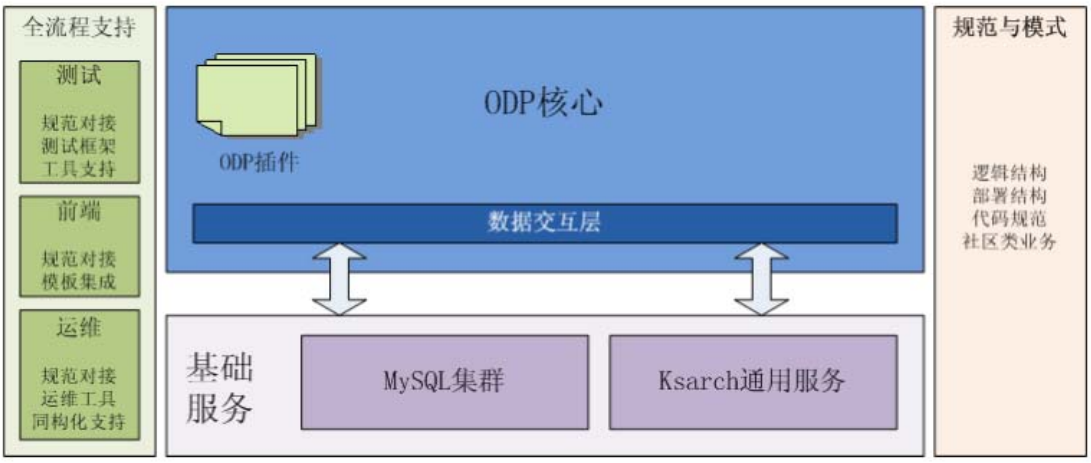

图 2-1 ODP 缩略图

ODP 核心包含了 ODP 的核心功能组件，包括运行环境、核心基础库、数据交互层、框架等。横向看，ODP 核心通过库、框架、工具等集成支持了各类规范和模式，也为全流程支持提供接口。向上看，ODP 核心直接为产品线业务提供运行环境和研发支持。向下看，ODP 核心通过数据交互层将底层的通用服务提供给业

务。ODP 核心支持插件机制，各技术组和产品线可通过插件向 ODP 增加新功能，比如基础库、测试框架、工具等。

ODP 核心通过数据交互层将底层的通用服务提供给业务，目前，ODP 为大社区类业务集成了较为完备的通用服务支持。

1.MySQL 集群为 DBA 统一运维和提供的 MySQL 存储服务，具有高可用、高性能、自动读写分离等工业级特性。

2.Ksarch 通用服务 Ksarch 为大社区类业务提供了大量成熟稳定的通用服务，如反作弊、检索、提交、高性能、kv 存储、memcached 等。

2.1.2 数据传输拓展 RAL

RAL 是一个支持多种交互协议和打包格式的 php 扩展。

RAL 规定了一套高度抽象的交互过程规范，将整个后端交互过程分成了交互协议和数据打包/解包两大块，可以支持一些常用的后端交互协议，标准化协议扩充的开发过程，促进代码复用。RAL 集成了负载均衡、健康检查等功能，让上游端不需要再关注这些繁琐的通用逻辑，同时实现版本可以在性能方面有更优的表现。

ral 采用客户机/服务器模式。请求程序就是一个客户机，而服务提供程序就是一个服务器。首先，客户机调用进程发送一个有进程参数的调用信息到服务进程，然后等待应答信息。在服务器端，进程保持睡眠状态直到调用信息到达为止。当一个调用信息到达，服务器获得进程参数，计算结果，发送答复信息，然后等待下一个调用信息，最后，客户端调用进程接收答复信息，获得进程结果，然后调用执行继续进行。

2.1.3 消息队列 NMQ

长期以来，百度社区几大产品线（贴吧、空间、知道、ks 等）都独立维护着自己的提交系统。虽然产品逻辑和规模不同，但从实现上来讲，面对的问题相近，解决的思路相似，维护重复、运维分散、人力浪费。

Nmq 是 New Message Queue 的简称，是一个通用的消息队列系统，为在线服务设计，用于替换并统一 UGC（User Generated Contents）产品线的提交服务，由 ksarch 开发，维护.简单地说，nmq 接收上游提交过来的消息，将它保存起来，另一方面，又将储存起来的消息发送出去。

消息队列中间件是分布式系统中重要的组件，主要解决应用耦合，异步消息，流量削锋等问题。实现高性能，高可用，可伸缩和最终一致性架构。是大型分布式系统不可缺少的中间件。

proxy 是 nmq 整个集群的入口，无状态、无单点，它的作用是将收到的数据发给 topic.Proxy 将上游收的消息发送给 topic，这里 topic 是单点；这个接收所有 proxy 发来的消息，并且保存起来的 topic 称为主 topic，它保存有全量数据。在某些应用场景，我们可能需要多份全量数据，所有数据必须一致。所以我们选择和主 topic 进行消息同步，从主 topic 同步数据的称为备 topic。

物理上不同 Topic 的消息分开存储，逻辑上一个 Topic 的消息虽然保存于一个或多个 broker 上但用户只需指定消息的 Topic 即可生产或消费数据而不必关心数据存于何处。

消息被发送到队列中。“消息队列”是在消息的传输过程中保存消息的容器。消息队列管理器在将消息从它的源中继到它的目标时充当中间人。队列的主要目的是提供路由并保证消息的传递；如果发送消息时接收者不可用，消息队列会保留消息，直到可以成功地传递它。

消息队列实现了调用者于被调用者的解耦，解决了许多应用场景下的并发问题。

2.2 缓存系统

工作中主要使用缓存系统主要为 memcached 和 redis。Redis 相对 Memcached 功能和特性更多； 而对于性能，Redis 作者的说法是平均到单个核上的性能，在单条数据不大的情况下 Redis 更好。为什么这么说呢，理由就是 Redis 是单线程运行的。 因为是单线程运行，所以和 Memcached 的多线程相比，整体性能肯定会偏低。因为是单线程运行，所以 IO 是串行化的，网络 IO 和内存 IO，因此当单条数据太大时，由于需要等待一个命令的所有 IO 完成才能进行后续的命令，所以性能会受影响。而就内存使用上来说，目前 Redis 结合了 tcmalloc 和 jemalloc 两个内存分配器，基本上和 Memcached 不相伯仲。如果是简单且有规律的 key value 存储，那么用 Redis 的 hash 结构来做，内存使用上会惊人的变小，优势是很明显的。

在缓存使用上，当服务器内存有限时，如果大量地使用缓存键，并且过期时间设置得过长，就会导致缓存占满内存； 反之，如果为了防止内存过大，而把

所有的 key 的过期时间设置过短，可能会导致缓存命中率太低，降低效率；同时，可能会存在大量空闲内存； 采取中庸之道，在时间和空间上找到平衡点，给每个 key 都选择合适的过期时间，这样做难度较大。所以我学习到的结论是：限制缓存能够使用的最大内存，防止因为缓存导致服务器崩溃； 添加淘汰策略，当缓存达到内存使用上限时，通过淘汰策略舍弃/持久化被淘汰的元素。

使用缓存对提高系统性能有很多好处，但是不合理使用缓存非但不能提高系统的性能，反而成为系统的累赘，甚至影响系统运作，产生很大的风险。对于频繁修改的数据、没有热点的访问数据、数据一致性要求非常高的数据，不建议使用缓存。

同时在缓存设计上也有着相当多的技巧和风险点。最主要的三个点在于缓存穿透、缓存并发和缓存失效。

缓存穿透是指查询一个”一定不存在“的数据，由于缓存是“不命中时”被动写的，并且出于“容错”考虑，如果从存储层查不到数据则“不写入缓存”，这将导致这个存在的数据每次请求都要到存储层去查询，失去了缓存的意义。

缓存并发问题主要是指有时候如果网站并发访问高，一个缓存如果失效，可能出现多个进程同时查询 DB，同时设置缓存的情况，如果并发确实很大，这也可能造成 DB 压力过大，还有缓存频繁更新的问题，缓存频繁更新容易造成数据的不一致，在分布式系统中经常需要使用锁来规避这个问题

引起缓存失效的主要原因还是高并发的时候，平时我们设定一个缓存的过期时间时，可能有一些会设置 1 分钟啊，5 分钟这些，并发很高时可能会出在”某一个时间同时“生成了很多的缓存，并且”过期时间都一样“，这个时候就可能引发一当过期时间到后，这些缓存同时失效，请求全部转发到 DB，DB 可能会压力过重。那如何解决这些问题呢？其中的一个简单方案就时讲缓存失效时间分散开，比如我们可以在原有的失效时间基础上增加一个”随机值“，比如 1-5 分钟随机，这样每一个缓存的过期时间的重复率就会降低，就很难引发集体失效的事件。

2.3 数据库的使用

作为后端开发最重要的技术之一，在学习接触新知识时，进一步加深对 mysql 的理解应用是非常重要的。如何能发挥 mysql 的高效，建立最优化的索引执行高效的查询都是非常重要的。MySQL 查询性能的优化涉及多个方面，其中包括库表结构、

建立合理的索引、设计合理的查询。库表结构包括如何设计表之间的关联、表字段的数据类型等。我们通过响应时间来对查询进行分析，找出消耗时间最长的查询或者给服务器带来压力最大的查询，然后检查查询的 schema、SQL 和索引结构，判断是否有查询扫描了太多的行，是否做了很多额外的排序或者使用了临时表，是否使用了随机 I/O 访问数据，或者太多回表查询哪些不在索引中的列的操作。在发现查询效率不高时，首先就需要考虑查询语句的设计是否合理。

同时在工作也在很多设计细节上进行了学习，如:选用更小的数据类型，因为越小的数据类型通常更好：越小的数据类型通常在磁盘、内存和 CPU 缓存中都需要更少的空间，处理起来更快；简单的数据类型更好：整型数据比起字符，处理开销更小，因为字符串的比较更复杂。在 MySQL 中，应该用内置的日期和时间数据类型,而不是用字符串来存储时间;以及用整型数据类型存储 IP 地址。尽量避免 NULL：应该指定列为 NOT NULL，除非你想存储 NULL。在 MySQL 中，含有空值的列很难进行查询优化，因为它们使得索引、索引的统计信息以及比较运算更加复杂。你应该用 0、一个特殊的值或者一个空串代替空值。[3]这些都是非常细节的东西，但如果没有注意到又会对性能产生极大的浪费。

关于索引，对于聚簇索引、覆盖索引等都有了更深刻的认识，在设计时尽量让大流量的查询使用覆盖索引，所谓的覆盖索引是指索引包含满足查询的所有数据。[4]覆盖索引是一种非常强大的工具，能大大提高查询性能。

2.4 作业帮体系架构

作业帮主系统体系后端架构分为四大个大的层次——接入层、业务层（读）、业务层写和数据层。独立子系统包括 PHP-APP 子系统、OCR 子系统、反作弊子系统 DB 子系统、检索子系统、长连接子系统、nmq 子系统（包括 nmqpusher、topic、proxy、zookeeper 几大功能模块）和答疑子系统。整个作业帮平台包含数十个独立的代码模块分别分布在 web 集群、session 集群、napi 集群上。存储层则包括 redis、memcached 和 mysql，其中 mysql 包括数十个数据库，数千张数据表，部分功能维护由百度的 DBA 进行维护管理。redis 则包括 redis-session、redis-napi、redis-common、redis-push 等几大缓存集群组成，供作业帮平台内的不同的业务线使用。

作业帮的接入层业务层的架构如图 2-2 所示：

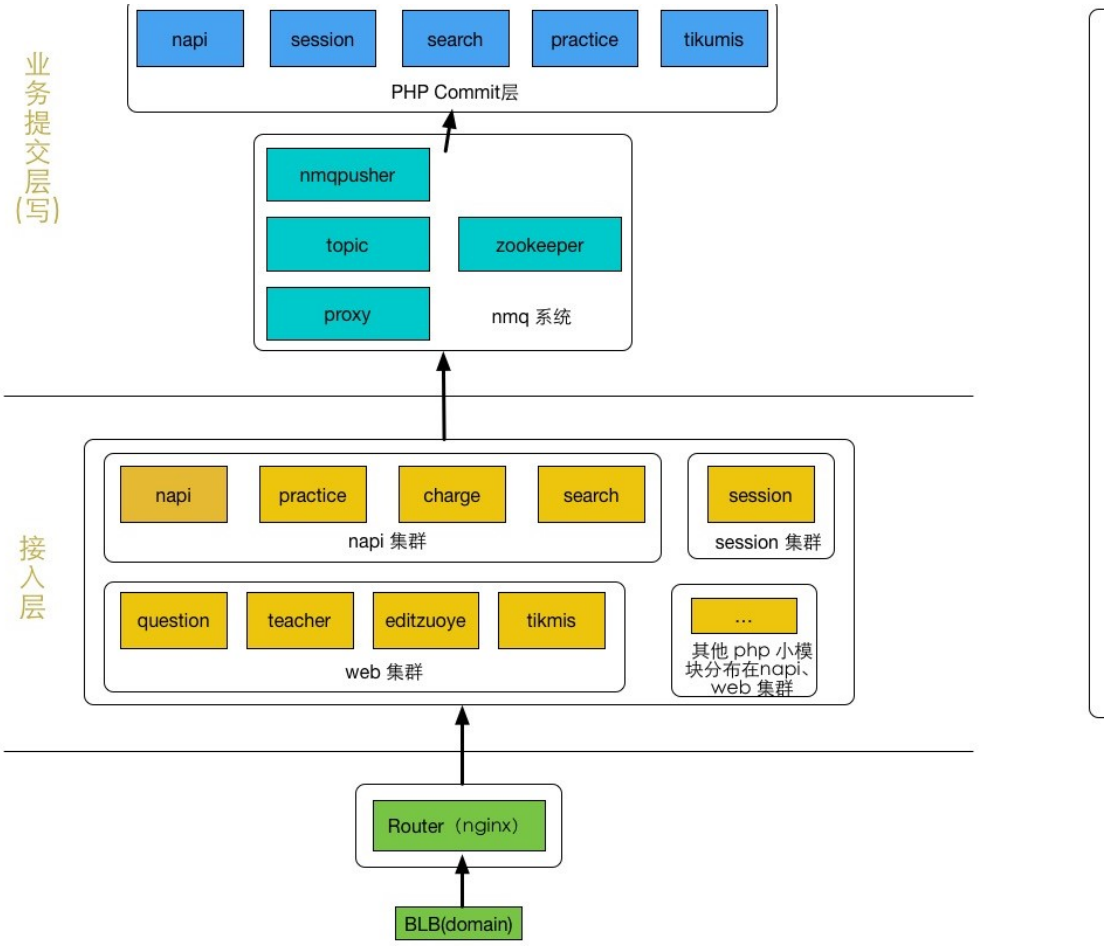

图 2-2 作业帮业务层接入层架构图

第三章 功能需求和难点分析

3.1 黑板报数版本开发问题

3.1.1 黑板报初版开发

在作业帮发展过程中，为了增加用户在使用 app 过程中的娱乐性和加长用户的使用时长。产品方决定在 app 内开发一个内容的运营平台，通过发布自己编辑的内容来加强用户的使用体验，同时也能够将其作为一个发布自己运营管理消息的平台。在几次评审后黑板报以“用户学习空闲时娱乐用的功能模块”的定位，出于上述的这些目的应运而生。

简单来说功能需求分为前后台两块，后台包括对文章的修改、增加、删除、查看等操作。设计的前台页面主要是由是一个单列表和文章详情页构成的。列表是一个非常简单的基于表存储的上线时间进行排序、取出表里存储的文章 id、文章封面图、作者、浏览量等字段吐出给前端进行展示的页面（黑板报列表页如图 3-1）。同时下拉时会请求更新的数据，上拉时则会请求时间更老的文章数据进行展示。

在做简单的内容平台时，主要的问题是内容的数据结构，为了增强文章的阅读体验，核心的内容绝对不能是简单的文字，而应该是包含图片、音频、表情等元素的富文本内容。

为了在黑板报更新时告知用户有新内容引导用户进行阅读，产品方提出在黑板报文章更新时在黑板报入口需要有一个小红点来提示用户加强对用户的吸引，从而提升用户的的点击量的目的。

图 3-1 黑板报列表

3.1.2 增加热点数据缓存与增加点赞弹幕功能

作为依托于大流量 app 作业帮的黑板报，它天然就有着较大的潜在流量。在黑板报上线了一段时间后，功能模块的日均请求量已达到一定规模。产品方根据数据统计得出接口的反馈时长在高峰时期有波动会降低用户体验需对接口做一次优化。经过断点分析发现波动的主要原因是因为在高峰期 db 读取的返回时长有波动，所以我决定在热点接口增加缓存来进行优化。

需求方同时提出单纯的文章列表展示太过普通，缺乏用户之间的交流且粘性较低，应该从用户角度思考，所以他们提出了在黑板报详情页增加 k12 用户喜欢的弹幕功能和点赞功能（弹幕功能和点赞功能见图 3-2）。在文章阅读的同时，用户可以选择对文章进行点赞、发送弹幕等操作以增加文章阅读的乐趣性。

图 3-2 点赞与弹幕功能

3.1.3 增加腾讯天天快报数据源与优化数据库存储

由于自身原创的内容，无论从数量和内容的丰富度上来说都远远不能满足用户的需求，使得功能模块的留存并不让产品经理感到满意，所以为了丰富内容源，因此决定引入第三方数据源——腾讯天天快报的内容。而引入第三数据源最大的问题就是数据混排问题，作为一个以上线时间排序的列表，若将从腾讯获取的数据放入一个单独的数据表进行存储，则无法使用数据库现有的方法进行排序，若在业务层将数据排序，则会导致接口请求缓慢，无法维护等问题。所以需要为了使两个数据源的数据进行统一，故从新改变了表的结构，使表的字段可以兼容两个数据源。

3.1.4 黑板报增加短视频内容

随着黑板报的用户逐渐增长，对于功能的需求有越来越多。为了加强用户体验，新的一次改版决定增加短视频内容。短视频内容的分为两类，一是作业帮内部来源的视频，二是百度秒懂百科提供的来源。

对于两个视频的处理方式都比较简单，对于百度秒懂百科的视频只需根据他们提供的链接，在前台页面展示即可。对于自己的来源的视频只需在我们的后台文件管理系统上传即可得到链接。

而这次的改版中有两个难点，其一最大难点是短视频的数据结构与之前的黑板报数据结构太不一致，不能像之前引入腾讯天天快报时一样用同一个表存储它的数据内容，因为如果要使用一个表存储三个数据源的不同数据格式的数据，会造成 mysql 的表的字段浪费，也会导致在后续的维护拓展中遇到很多问题，故这次采取了较大的数据结构改变。同时为了加强用户体验和增加用户停留时长，在用户点击进入一个短视频页面后，在视频结束后会选择相同类别的十个短视频以列表形式展示给用户推荐他们观看。

3.2 黑板报活动优惠券问题

为了提高用户对黑板报模块的认知度和促进运营方和用户的交流开展了许多相关的运营活动。在活动中为了使用户积极参加活动，都在活动页的任务中都有着发放优惠券等相关功能。活动中的优惠券可能属于不同业务线的数据，因此在请求相关的优惠券列表、优惠券领取接口时都是使用 rpc 去调用其他模块的代码来实现用户券的相关功能。在有优惠券的场景，往往容易产生较大的流量，故我经手开发后端的许多活动都有着较大的访问量，因此也就必然会有相关的并发问题。而我所遇到最典型的问题之一就是高并发场景的缓存并发问题，而面对并发保证数据可靠性最有效的方法就是使用锁。

3.3 作业帮第三方账号拓展

为了降低作业帮用户注册成本，提高新增用户注册率和登陆率，作业帮决定引入第三方账号登录，根据现在市场主流导向，作业帮决定采用的第三方登录平台包括 qq、微信和微博。功能本身是希望通过第三方提供的 oauth 接口来获得第三方的唯一标识 openid，并同时将相关的的用户数据——用户头像、昵称、性别等数据通过第三方提供的接口获取将之视为新注册用户的数据进行存储，与此同时再分配作业帮用户标识 uid 给第三方注册用户就算完成了注册。而其中最复杂的就是选用一个合适的数据方案去进行存储，作业帮的用户表等相关数据经过数年的沉淀结构已经稳定，而在新增功能并要保持最优的数据结构时则必须进行表

结构的重构，而表的结构必然涉及到数量巨大的数据迁移问题，作为一个公司最核心的数据表之一用户表，做它的迁移可以说是非常困难而又有巨大的风险。

3.4 本章小结

黑板报的功能初期过于单一，用户只是单纯地去浏览文章，参与感过低，粘性不高需求目的，为了增加黑板报功能多样性及提高用户参与感和互动性，对黑板报进行数次系统的优化和功能的迭代。本章主要从黑板报的列表功能、增加第三方内容功能、短视频功能、优惠券功能等数个方面进行了工作上的需求分析。同时对于新增的需求在项目前期还涉及可行性分析与成本分析，在确定开发周期后则对相应的功能中开发的难点进行相应的排查与分析。至此，项目前期的需求与难点分析工作就算告一段落。

第四章 系统方案设计与实现

4.1 黑板报初版功能设计

黑板报最初设计为一个简单的内容阅读功能，一个内容平台最主要的就是数据，所以其数据结构的设计较为重要。分析其前后台的功能需求，前台为普通的列表，后台为普通的对表数据进行增删查改的编辑平台，所以数据结构的设计也较为简单，黑板报表结构如下：

代码 4-1 黑板报表 sql

```
CREATE TABLE `tblBlackboardArticle` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `type` tinyint(1) NOT NULL DEFAULT '0' COMMENT '黑板报文章的分类',
  `title` varchar(100) NOT NULL DEFAULT '' COMMENT '文章的标题',
  `author` varchar(50) NOT NULL DEFAULT '' COMMENT '文章作者名字',
  `avatar` varchar(255) NOT NULL DEFAULT '' COMMENT '作者图像',
  `content_type` tinyint(1) NOT NULL DEFAULT '1' COMMENT '内容实体的类型，
fe 展示的时候后边可能会用，1=>div 片段内容',
  `content` mediumblob COMMENT '内容实体',
  `start_time` int(11) NOT NULL DEFAULT '0' COMMENT '上线时间',
  `end_time` int(11) NOT NULL DEFAULT '0' COMMENT '下线时间',
  `img` varchar(255) NOT NULL DEFAULT '' COMMENT '文章对应的顶部大图',
  `thumbnail` varchar(1000) NOT NULL DEFAULT '' COMMENT '文章对应的缩略图',
  `page_view` int(11) NOT NULL DEFAULT '0' COMMENT '访问量',
  `is_delete` tinyint(1) NOT NULL DEFAULT '0' COMMENT '是否删除，1 删除，0
使用',
  `created_at` int(11) NOT NULL DEFAULT '0' COMMENT '创建时间',
  `updated_at` int(11) NOT NULL DEFAULT '0' COMMENT '更新时间',
  `related_id` int(10) NOT NULL DEFAULT '0' COMMENT '关联 id',
  `recommend` tinyint(1) NOT NULL DEFAULT '0' COMMENT '是否推荐 0 普通，1 推
荐',
  `channel` varchar(100) NOT NULL DEFAULT '' COMMENT '文章的栏目',
  `data_type` tinyint(1) NOT NULL DEFAULT '1' COMMENT '文章数据类型 1 原创 2
腾讯 3 百度 4 短视频',
  PRIMARY KEY (`id`),
  KEY `start_time` (`start_time`),
  KEY `type_data` (`type`,`data_type`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='黑板报文章表';
```

表的主要字段为 type、title、channel、content 等存储黑板报内容的字段，表的主键为普通的自增 id。表拥有两个二级索引——start_time 和 type_data。Start_time 主要用于前台列表展示时按时间展示，type_data 则是前台文章列表进行筛选时优化查询的字段字段。

文章的前台功能为简单的列表形式的展示，初次进入页面时执行 query：

代码 4-2 首屏查询 sql

```
select `黑板报列表所需字段` from tblBlackboardArticle where is_delete=0 and start_time<`当前时间戳` order by start_time limit `首屏条数 n;
```

然后将查询的结果进行整理打包成 json 格式返回给前端进行渲染展示。

文章列表下拉时，前端会传第一条文章的时间戳 b 给后端，后端则是执行 query：

代码 4-3 文章刷新 sql

```
select `黑板报列表所需字段` from tblBlackboardArticle where start_time > b limit `最多刷新条数 n` and start_time<`当前时间戳` and is_delete=0  order by start_time desc;
```

去读取数据库中存储的更旧的文章。

文章列表上拉时，前端则是传最后一条文章的时间戳 d 给后端，后端执行 query：

代码 4-4 加载更多 sql

```
select `黑板报列表所需字段` from tblBlackboardArticle where start_time > d limit `加载条数 n` and start_time<`当前时间戳` and is_delete=0  order by start_time desc;
```

去读取数据库中存储的更新的文章。

后台功能都是基于一个黑板报内容表的简单 update、insert、select 操作，在文章更新时则 update 相应字段，增加时 insert 一条记录，删除时为了保持前后台数据一致，则不采取硬删除而是将黑板报的 is_delete 字段置为 1 进行软删。

为了实现黑板报文章更新的小红点功能，黑板报将最新的文章更新时间写在了用户打开 app 时必须请求的一个后端通用配置接口——appconfigdelay 中。客户端每次读取该接口返回的黑板报更新时间与端上记录的黑板报入口上次点击时间进行比较，若更新时间戳大于上次点击时间戳则在黑板报入口位置展示一个小红点提示用户有内容更新从而吸引用户点击。

4.2 黑板报改版相关问题

4.2.1 黑板报相关缓存设计

在黑板报请求的流量提升之后，为了降低 db 压力，并降低用户的请求时长提升用户体验，决定在列表页和热点的文章的详情页增加缓存。同时也注意到 appconfigdelay 接口每天有数亿次的请求，若每次都要读 db 请求最新文章时间，毫无疑问是一个巨大的资源浪费，会拖慢接口请求时长，故也要将其优化，将其从对 db 的读成对缓存读。

其中文章详情页的缓存较为简单，缓存系统可直接选择 memcached，将文章 id 作为 key，文章详情作为 value 进行缓存。但同时涉及的问题是，黑板报进行增删改操作不仅要对 db 进行操作还需对 cache 进行相应的操作。为简化对缓存的操作以保证数据的一致性，我将缓存的内容设为 3 分钟，3 分钟后缓存失效，用户请求发现没有缓存时则读取 DB，读取返回数据后在将内容写入缓存。这样就避免了数据源变更时去处理缓存，数据修改时在下一次读取时 db 就会被自动刷新。

较详情页缓存较为复杂的是列表缓存，作为用户阅读黑板报必须进入的列表页面，它采用 feed 流样式。简单来说就是所有的数据请求都是异步，请求类型分为三种，首次进入的首屏数据请求，上拉更新操作与下拉加载操作。最初由数据库实现的时候特别简单——下拉时根据上线时间排序查找表中上线时间小于请求刷新时前端传来的时间戳，下拉时根据上线时间排序查找表中上线时间大于请求刷新时前端传来的时间戳。但增加缓存时需要考虑的则居多，首先缓存的目标对象应该是热点对象，所以我将缓存对象设为前十屏的内容。缓存将文章列表 id 数组作为单独对象处理，再将前十屏列表每一个文章对象单独缓存。在缓存有效期内请求缓存的 id 列表，读到应该加载的文章的 id，再根据 id 得到缓存的文章对象拼凑成 json 对象返回给前端进行展示。

至于其中请求量最大的后端通用配置接口 appconfigdelay 中对黑板报最新时间戳则也从简单的对数据库的读转到了以缓存为主数据库为备份的设计。接口的请求逻辑转成了第一步对缓存进行读取，若缓存为空用旧逻辑去 db 中取最新的文章时间戳然后将其写入缓存中。在这个设计中，最大的问题就是对缓存的更新问题，在新增文章时需要将最新的文章时间戳更新到缓存中，在删除最新文章时也要将删除之后列表最新的文章的时间戳更新到缓存中。故在对最新文章的数

据记录进行操作后会掉一个通用的函数去读 db 重置缓存从而保证缓存数据的一致性。

同时，使用缓存时应注意避免单个节点压力过大以致瘫痪。故应注意平衡缓存集群之间的压力。因为列表页为最热数据，且只有单 key。则必然存储该 key 的节点会承受更高的压力。所以将相同 id 列表做多个数据在多个节点存储，每次请求随机取其中一个 key，让请求分散到不同节点，均衡各个节点之间的压力。

在黑板报上会存在一个缓存同时失效的问题，引起这个问题的主要原因还是在黑板报高并发的时候。黑板报最初设计的所有的 key 的缓存时间为 5 分钟，并且大多是同时刷新，在并发很高时可能会出在某一个时间同时生成了很多的缓存，并且过期时间都一样，这个时候就可能引发一当过期时间到后，这些缓存同时失效，请求全部转发到 DB，DB 可能会压力过重。发现这个问题的我选择了一个简单方案就时讲缓存失效时间分散开，就是在原有的失效时间基础上增加一个随机值，缓存时间为 rand(60,60*5)秒，这样每一个缓存的过期时间的重复率就会降低，就很难引发集体失效的事件。

另外在黑板报的上线过程中出现过几次 db 负载过大现象，最终发现都是对于不存在的文章 id 的请求。因为黑板报文章缓存的 key 是根据文章 id 生成的，所以当请求了不存在的文章发现缓存不存在就会去请求 db。为了解决这个问题采用了类似布隆过滤器的方法，在处理所有对文章的请求时先将其请求的 id 进行判断是否可能存在，若不存在直接返回空，从而避免了对底层存储系统的查询压力。

另外，由于黑板报中，用户相关的数据量非常大，这里将部分信息如文章列表展示的图片等数据缓存在客户端 localStroage 中，用来减轻服务器的压力。

4.2.2 兼容天天快报数据源问题

由于腾讯天天快报相关文章数据更新较快，为了保持对新文章尽早获得，在服务器使用 crontab 开启了一个定时任务，每五分钟去执行一次，使最新的文章能尽快进到我们的数据表中。

定时任务配置如下：`*/5 * * * * cd /home/homework/app/napi/script/ && /home/homework/php/bin/php getTencentNews.php > /home/homework/a.log 2>&1`

脚本执行的主要任务是通过 curl 去请求腾讯方提供的接口，将数据格式整

理存入数据表 A 中。由于内网的集群大多没有外网权限，因此在请求腾讯方接口时，在 curl 请求上配置了 proxy，通过代理机器转发请求才能正确访问腾讯提供的接口，获得其返回的数据。以下为请求腾讯新闻数据接口代码：

代码 4-5 请求腾讯新闻

```
int function requestNews($channel_code, $start,$size,$tmp=null){
       $ret = empty($tmp)?array():$tmp;
       $url = self::REQUEST_URL. "?refer=" .self::REFER . "&app_key=" .
self::APP_KEY . "&channel_code=" . $channel_code . "&start=" . $start .
"&size=" . $size."&h=1";
       $curl = curl_init();
       curl_setopt($curl, CURLOPT_URL, $url);
       curl_setopt ($curl, CURLOPT_PROXY, $this->proxy);
       curl_setopt($curl, CURLOPT_HTTPGET, 1);
       curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
       curl_setopt($curl, CURLOPT_HEADER, 0);
       curl_setopt($curl,CURLOPT_TIMEOUT,5);
       $res  = curl_exec($curl);
       curl_close($curl);
       $result     =   json_decode($result,true);
       if($res  ===  false){
           Bd_log::warning('curl exec error,url:'.$url.'
error:'.curl_error($curl));
           return false;
       }
       if ($result['ret'] === 0) {
           $list = $result['data']['list'];
           if(empty($list)){
               Bd_Log::warning("get tencent $url news empty:");
               $list   =   [];
           }
           //整理数据格式
           foreach ($list as $item) {
               $news    =   array(
                   'url'=>$item['content']['url'],
                   'articletype'=>$item['content']['articletype'],
                   'id'=>$item['content']['id'],
                   'title'=>$item['content']['title'],
                   'src'=>$item['content']['src'],
                   'abstract'=>$item['content']['abstract'],
                   'thumbnails_pic'=>$item['content']['thumbnails_pic'],
```

```
                'timestamp'=>$item['content']['timestamp'],
                'image33_l'=>$item['content']['image33_l'],
                'channel_code'=>$channel_code,
            );
            $ret[]      =   $news;
        }
    }
    else{
        Bd_Log::warning("get tencent $url news error");
        return false;
    }
    return $ret;
}}
```

另外由于从第三方直接获取的内容未经审核就直接投放在端上展示风险较高，可能会有大量低质文章进入。所以在 curl 返回第三方数据后的第一步，我们会将腾讯接口返回的数据打包成通用的数据格式，通过 rpc 调用作业帮的黄反作弊接口如果返回 FALSE 则跳过不处理该数据，如果接口返回 TRUE 则将腾讯接口的数据整理成相应的字段插入腾讯新闻审核表中进入后台审核环节。审核过程中可在后台（审核界面见图 4-1）对腾讯天天快报的文章点击上线，则会将文章的需要展示的关键字段从审核表取出根据配置的映射关系插入表 B 中（存储原创黑板报文章的表），点击下线则会根据主键 id 将数据从表 b 中的下线字段置为 1。这样分为两表存储数据是由于腾讯提供的内容大部分为不可用数据，这么做可以保证文章的内容和减少无效的数据进入核心表 B，减少表的无效数据，使得列表读取和缓存速度更快。同时使两边数据存储最终在同一个表中也能更好的实现腾讯天天快报文章和作业帮原创文章按照时间按时间来进行数据的混排。

图 4-1 审核后台页面

4.2.3 短视频增加时的数据改造

在最初两次改版中，黑板报列表的数据都是用一个表 A 存储数据，这次拆成了四个表，原有的表 A 存储了几个来源的黑板报文章的所有数据。这次的拆分成四个表分别为表 A 存储黑板报的文章的 id，筛选字段和排序字段，并依靠这些字段建立索引。表 B 则是存储原创黑板报的文章内容、封面、作者、标题、栏目等核心内容相关的字段和与表 A 关联的 id，与之类似，表 C 则存储腾讯天天快报的文章封面、类型、文章连接这些字段内容和表 A 关联的 id，表 D 则是的短视频视频链接、视频封面、视频时长这些数据和表 A 关联的 id。

在请求列表的时候，最初请求表 a，得到文章列表的 id，再去根据数据类型分别去 join 表 B、C、D 然后等到列表数据。由于表 A 建了列表所需的相关索引，所以查询速度非常快,并且与之关联的 B、C、D 表也以表 A 的主键 id 建立了唯一索引，所以这个关联查询在查询性能上也不会成为瓶颈。

在实现短视频同类列表功能时，对于服务端来说，得到文章的类别，和该类别下的文章都很简单。问题查找随机数据时涉及 mysql 的一个实现机制问题，mysql 的 sql 语句“SELECT * FROM table_name ORDER BY rand () LIMIT n”会执行一个全表扫描操作，性能相当的慢，如果使用原生的这个 sql 进行查询，无疑是会拖垮服务器的慢查询，故绝对不可使用。如果采用自己设的随机数进行查询，不知道表的最大自增 id 也是无法做的。故采用了缓存，将视频的 id 列表进行了一层缓存，相当于依靠缓存加了一层现有 map，每次在整个集合中随机取出 10 个 id，然后查询这 10 个 id 的数据就能生成逐个列表了。

代码 4-6 获取推荐视频代码

```
private function getRecommendVideo($id,$type) {
 //得到黑板报某类型视频 id 列表
   $list = $this->objArticle->getVideoIDListByType($type);
   if(empty($list)) {
       return [];
   }
   $lenth= count($list);
   $tmp = [];
   $mark= 10;
   if($lenth > 11) {
      //打乱视频排序
       shuffle($list);
       for ($i=0;$i<$mark;$i++) {
```

```
                if($list[$i] != $id){
                    $tmp[] = $list[$i];
                }
                else{
                    $mark++;
                };
            }
        }
        else{
            foreach ($list as $item) {
                if($item != $id) {
                    $tmp[] = $item;
                }
            }
        }
        $list = $tmp;
        if(!empty($list)) {
            $conds = [
                "id in (".implode(',',$list).')'
            ];
        }
        else{
            return [];
        }
       //查询推荐视频
        $data =
$this->objArticle->getListByConds($conds,['id','img','content','thumbnail
','title']);
        if($data){
            return $data;
        }
        else{
            return [];
        }
     }
```

4.3 黑板报优惠券场景的缓存并发问题

黑板报优惠券相关功能是在单独的一个优惠券中心，整个优惠券中心分为前端和后端，我所负责的是后端 RPC 接口的开发。接口中包含“查券”和“领券”两个方法。项目中的优惠券列表在 Redis 中以 List 的形式存储，查询时的逻辑很简单

(1) 查询缓存，如果缓存存在，返回结果；

(2) 缓存不存在，查询数据库

(3) 把查询数据库的结果循环放入缓存

然而按照这样的流程去实现后却在上线后出现了列表中每个优惠券都展现了两次。经过排查确认了是以下原因，在某个时间点缓存不存在，请求量又很大的时候，会出现缓存并发的问题。也就是多个进程会重复去查询 DB，又重复去更新缓存。这其中重复查询 DB 是次要问题，而重复更新缓存则是主要问题。假如有两个线程同时进入上述的第三个阶段，各自进行 rpush 操作，那么最终会在优惠券列表的缓存中插入两组同样的数据。怎么解决呢？用、锁机制？显然不行，因为线上环境通常都是多个服务器组成的集群。于是想到了利用分布式锁。所谓分布式锁有很多种，可以利用 ZooKeeper、MemCache、Redis 来实现。其中 Redis 的方式比较简单，无非是利用一个服务器之间共享的 Key，以及 Setnx 指令。当第一个线程执行 Setnx，会存储对应的键值，相当于成功获得锁。当后续再有线程对同于的 Key 执行 Setnx 指令，则会返回空，相当于抢锁失败。同时，为了防止一个线程因意外情况而长久把持着锁，程序对 Key 设置了 1 秒的过期时间。

归纳一下修改后的逻辑：

(4) 查询缓存，如果缓存存在，返回结果；

(5) 缓存不存在，查询数据库

(6) 争夺分布式锁

(7) 成功获得锁，把查询数据库的结果循环放入缓存

但是再次上线却发现这个问题依然存在，所有的优惠券都展示了两次。诡异的 bug 又重现了，发现了上次的改动仍然存在一个致命的漏洞。在这里我们假定缓存不存在，刚好有两个线程 A 和 B 一后一先进入到代码块。第一阶段，线程 A 刚开始查询优惠券缓存，线程 B 正尝试获取分布式锁。第二阶段，由于缓存不存在，线程 A 开始查询数据库，线程 B 成功获得锁，开始更新缓存：第三阶段，线程 A 尝试获得分布式锁，而线程 B 已经释放分布式锁：第四阶段，线程 A 获得了锁，又一次更新缓存，而线程 B 已经成功返回：就这样，缓存被重复更新了两次，所以再次出现数据重复的 bug。

这种局面如何破解呢？其实不难，经过查找相关资料，了解到了双重检测机制。只需在线程成功得到锁以后，再次判断优惠券缓存的存在即可。得出了最终的设计方案：

(1) 查询缓存，如果缓存存在，返回结果；

(2) .缓存不存在，查询数据库

(3) 争夺分布式锁

(4) 成功获得锁，再次判断缓存的存在

(5) 如果缓存仍旧不存在，把查询数据库的结果循环放入缓存

(6) 释放分布式锁。

具体代码的实现逻辑如下：

代码 4-7 更新优惠券列表缓存

```
 //如果优惠券列表存在，从缓存中查询
if($redis.exist("优惠券列表 key")){
  return redis.lrange("优惠券列表 key",0,-1)
}
//缓存不存在 从 db 查询优惠券
$couponList = $this->getListFromDb();
//争夺锁
if($redis.setnx("分布式锁","ok",2) != null) {
//获得分布式锁，再次判断优惠券缓存的存在
  if($redis.exist("优惠券列表 key")){
      return $couponList;
  }
  //把 db 查询结果放入缓存
  foreach($couponList as $coupon){
      $redis.rpush("优惠券列表 key",coupon)
  }
  //释放分布式锁
  $redis.del("分布式锁");
}
return $couponList;
```

如图 3-3 所示为查询优惠券列表的业务流程图。

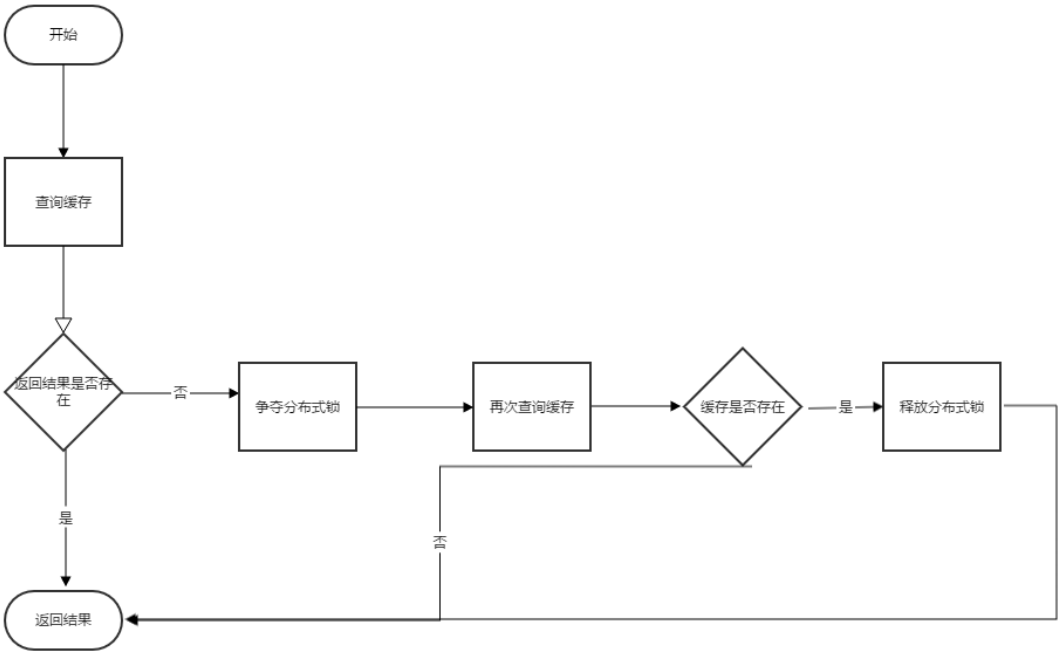

图 4-3 查询优惠券列表的业务流程图

经过上述的改版数次改版开发，在优惠券列表的接口保证了数据的一致性，不会产生因高并发产生的 redis 存储的数据错误问题。

4.4 增加第三方登录时的数据迁移问题

在介绍第三方注册前首先要介绍一下通用的 oauth 注册协议流程。市场主流的第三方账号登录系统都是采用 oauth2.0 协议，所以在核心模块的调用流程都相似，因此在编写基于多个第三方平台登陆功能时，使用的底层代码都相同，数据结构也都相似。具体注册流程如下:

（A）用户访问客户端，后者将前者导向认证服务器。

（B）用户选择是否给予客户端授权。

（C）假设用户给予授权，认证服务器将用户导向客户端事先指定的”重定向 URI”（redirection URI），同时附上一个授权码。

（D）客户端收到授权码，附上早先的”重定向 URI”，向认证服务器申请令牌。这一步是在客户端的后台的服务器上完成的，对用户不可见。

（E）认证服务器核对了授权码和重定向 URI，确认无误后，向客户端发送访问令牌（access token）和更新令牌（refresh token）

（F）服务端存储用户的 openid、头像等用户信息

（G）生成 session 存储用户登陆信息并将 session 的 key 返回客户端进行存储。

Oauth2.0 授权逻辑如图 4-2 所示：

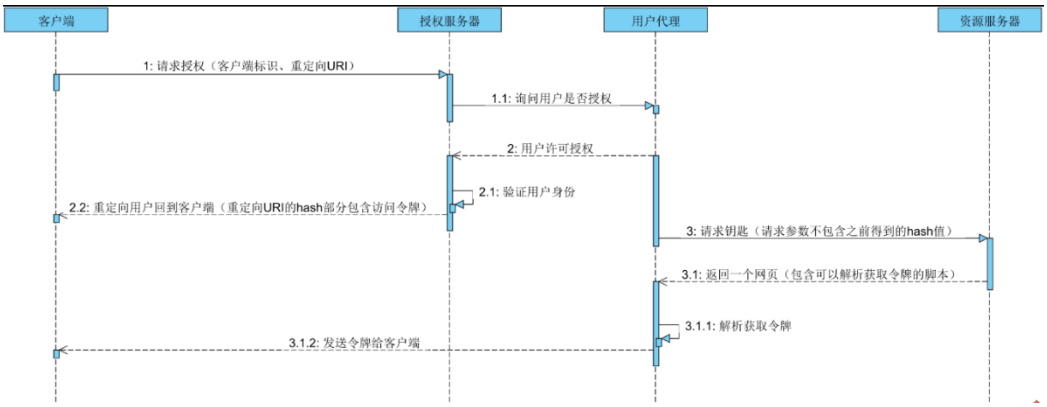

图 4-2 oauth 授权流程示意图

以下为获取腾讯 qq 提供的用户信息的相关代码：

代码 4-8 获取腾讯 qq 用户信息

```
/**
    * 获取腾讯 QQ 用户信息。<br>
    * @see Session_Oauth_Clients_Base::getUserInfo()
    */
   public function getUserInfo($ouid, $accessToken) {
       $query    = array(
           "openid"            => $ouid,
           "format"            => "json",
           "access_token"       => $accessToken,
           "oauth_consumer_key" => $this->appId,
       );
       $url      = sprintf("%s/get_simple_userinfo", $this->oauthApi);
       $response = $this->oauthRequest($url, $query);
       if (false === $response) {
           return $response;
       }
       $response = json_decode($response, true);
       $user     = array();
       if (0 === $response["ret"]) {
           $user["uname"]  = $response["nickname"];
```

```
            $user["sex"]    = ($response["gender"] === "女") ? 2 :
(($response["gender"] === "男") ? 1 : 0);
            $user["avatar"] = $response['figureurl_qq_1'];
        } else {
            $arg  = array(
                "reqeustUrl" => $url,
                "errmsg"     => $response["msg"],
            );
            Bd_Log::warning("get qq userinfo failed");
            return false;
        }
        return $user;
    }
```

在实现第三方登陆时面临的最主要的问题是数据的不兼容，之前作业帮所有的用户都是以手机来注册，用户注册登录相关的主要包含两个表，表 A 是以手机号作为分表的用户注册表，存储手机号及其对应的 uid 和用户注册状态 session 和 password，表 B 是以 uid 做水平分表的存储用户资料信息的表。原有的登录流程是通过手机号查询表 A 的记录，并在表 A 中写入记录用户登录状态的 session。因此再次之上新增了与其平级的表 C 依靠 openid 来分表的用户表以支撑第三方登陆功能，也需要 session、password 等字段，则这些字段会表 A 与表 C 中冗余在登录注册时造成许多问题。因此需要将 session、password 等用户登入注册状态字段从旧有的依靠手机号进行分表的用户表里迁移到一个新增表 D 中。把登入流程修改为依靠表 A 或表 C 查找用户的注册状态，若用户已注册则生成 session 写入表 D 中，并通过表 B 获取用户资料相关字段。如此一来就涉及了数据迁移，将线上的表 A 中的字段拆出来单建一张表，由于业务线不能因为数据迁移停止，因此工作非常的困难。

在作业帮这种大规模生产环境中，这可能意味着要迁移上一亿数据，其中包含数千万的条活跃的数据，以及重构数以千行计的代码。用户表有上亿规模的用户数据。对于我们的生产数据库来说，做一次与所有这些数据都相关的大型迁移就意味着非常非常多的工作。想象一下，假如以一种顺序的方式，每迁移一条订阅数据要一秒钟，那要完成上亿条数据的迁移就要耗时超过三年。作业帮上的业务一直在运行。我们升级所有东西都是不停机操作的，而不是可以在某个计划好的维护窗口内更新。因为在迁移过程中我们不能简单地中止订阅服务，所以我们必须 100%地在所有服务都在线的情况下完成迁移操作。

把亿级的数据从一张数据库表迁移到另一张中，这很困难，但对于许多公司来说这又是不得不做的事。在做类似的大型迁移时，有种大家非常容易接受的四步双写模式。这里是具体的步骤：

(1) 向旧表和新表双重写入，以保持它们之间数据的同步；
(2) 把代码库中所有读数据的操作都指向新表；
(3) 把依赖旧数据模型的旧数据删掉；
(4) 把依赖旧数据模型的旧数据删掉；

因为我们 zh 次数据迁移中涉及数亿存量，使用我们这次的操作较普通做法更为复杂。作为作业帮的重要的数据表之一，用户表有着一主多重和一备。而我们这次的数据迁移主要是依靠着备库来实现的，具体流程如下：

(1) 在时间点 A 前上线新逻辑的代码，维持原有逻辑的同时，将所有的时间点 A 后的写操作生成对新表的写操作的 sql 按时间发送到消息队列中进行暂存；
(2) 到达时间点 A，我们将备库与主库切断数据同步，保留时间点 A 的数据快照；
(3) 依靠从库的数据生成新的用户表的数据；
(4) 存量数据迁移完后消化队列中的 sql；
(5) 当数据库中的在凌晨延迟降低到 0.1s 下时，一台机器上线新的代码逻辑（将对旧表的增删查改替换成对新表的增删查改）。然后确认那条新逻辑的机器是否有数据不一致、错误日志的问题；
(6) 完成第五步后，将该备库的数据同步到主库和从库中；
(7) 数据稳定性监控。在确定系统稳定运行后，这次数据迁移就告一段落了；

4.5 功能上线后的数据回归

在相应功能上线之后，需要提供各类的可视化报表提供给产品方进行数据回归，数据回归的实现主要由前端、后端和数据方。当前端监控到用户进行相应操作时（如点击、浏览、刷新等）则会发送打点请求到后端，后端接收到打点请求则会进行打点操作，在日志文件新增一条 log。日志文件会被数据方定时处理生成相应的 hive 表的数据。

在生成 hive 数据表及相关数据后可在作业帮的 uda 数据任务平台（如图 4-3）编写 hive sql、python、shell 等脚本去定时运行跑出产品方要求的初始化数

据，并存在指定机器的固定磁盘位置上。

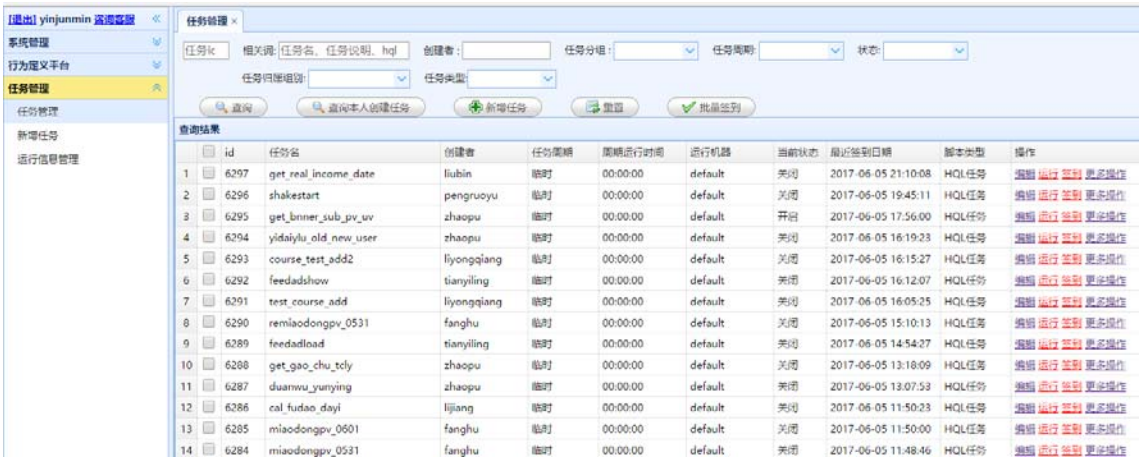

图 4-3 uda 数据平台

以下为 uda 平台编写的黑板报日留存相关数据脚本：

代码 4-9 黑板报日留存 shell 脚本

```
   cd /home/iknow/dataspace/UDA
  for ((i = {@date_timestamp-7}; i <= {@date_timestamp}; i += 86400)); do
      ymd_i=$(date "+%Y%m%d" -d "1970-01-01 UTC $i seconds")
      cur=$ymd_i
      if [[ ! -f zy_blackboard_cuid/$ymd_i/zy_blackboard_cuid ]]; then
        continue
    fi
      t=$(wc -l zy_blackboard_cuid/$ymd_i/zy_blackboard_cuid | awk '{print
$1}')
      cur=$cur$(echo -e "\t")$t
      for ((j = i+86400; j<= {@date_timestamp}; j += 86400)); do
          ymd_j=$(date "+%Y%m%d" -d "1970-01-01 UTC $j seconds")
        if [[ ! -f zy_blackboard_cuid/$ymd_j/zy_blackboard_cuid ]]; then
            continue
        fi
          t=$({@PYTHON} single_stay.py
zy_blackboard_cuid/$ymd_i/zy_blackboard_cuid
zy_blackboard_cuid/$ymd_j/zy_blackboard_cuid   | awk '{print $3}' || echo
-e "0")
          cur=$cur$(echo -e "\t")$t
      done
      echo "$cur"
  donereturn $coup
```

在 uda 平台运行生成初始化的数据之后就要去生成可视化的数据报表，可视

化的报表建立在百度知道的 crm 平台。在 crm 平台新建相关报表任务后则回去自动加载上一步 uda 运行出的数据。根据上一步的数据与选中的模型绘制相关的图表（如图 4-4 为黑板报数据报表）。

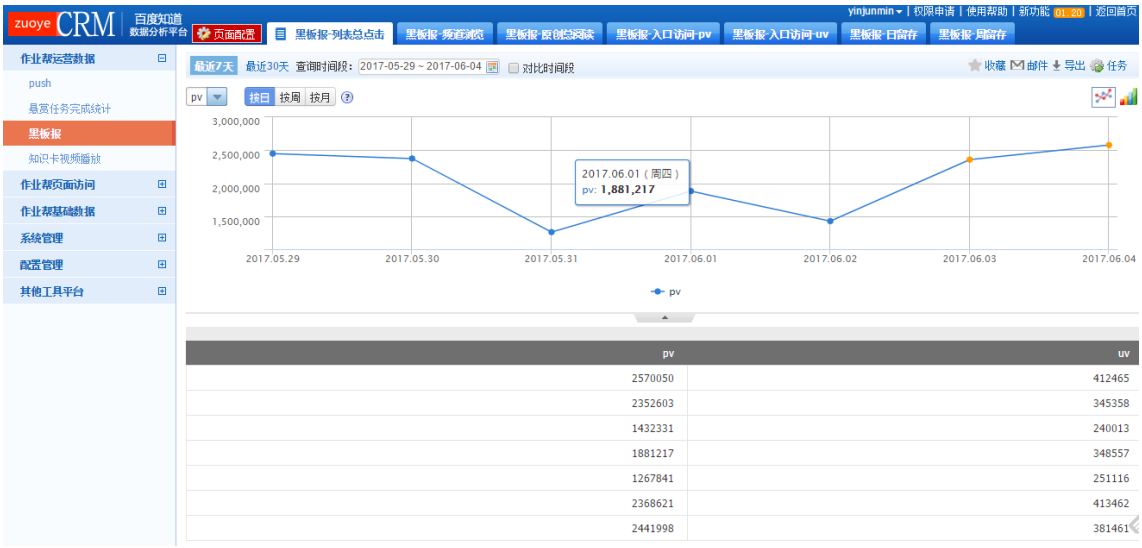

图 4-4 黑板报数据报表

4.6 本章小结

本章主要对在作业帮平台开发的黑板报列表功能、增加第三方数据源功能、增加短视频功能、优惠券功能等的系统方案的设计与实现采用了相关的图文进行了相关的介绍。对于项目开发中的难点如缓存设计、数据迁移等工作进行了详尽的介绍与分析，并对开发工作中的核心代码进行了分析与展示。在相关项目上线后，依然也相应的工作需要进行，数据回归系统的开发就是其中之一。为了确定项目的效果在项目后期需要编写相关的脚本并利用相关的工具绘制出可视化的图表提供给产品方进行回归。

第五章 工程计划管控与执行情况

5.1 工程计划

5.1.1 业务进程管控

工作中为了保证团队对的成员对业务的理解与确认项目不同时期的情况，在不同时期需要有不同的会议。团队的会议分为计划会议、每日站会、评审会议、和回顾会议。

评审会议是在做好阶段性规划之后，合理的和大家协调各自的时间，并预留开发周期后，安排项目相关的产品经理、测试人员、开发负责人等评审相关需求。确认项目的合理性、寻找项目的问题点，确认项目中的难点。需求评审前 PM 会将 MRD 邮件给项目组成员，所以最好在需求评审前根据 MRD 大致了解一下需求内容，对于有疑问或者觉得不合理的地方在需求评审中及早提出。项目启动之前的计划是非常重要的，通常有一下一些内容：a.项目的风险和预期（通常是风险越大的越先做,根据具体的项目情况）b.做好阶段性计划，预留好开发周期 c.设置项目的优先级顺序，从高到低。

每日站会的目标是让所有成员都能掌握团队的进度，能及时反馈. 敏捷开发依赖于开发团队的自组织特性，而自组织团队依赖于对团队承诺的遵守和维护。心理学上说，公开的承诺，我们会更加努力的去遵守和维护。所以每日站会是一个用来加强团队承诺的重要事情。敏捷开发强调语言沟通，每日站会是一个信息同步的重要渠道，也是同步团队成员间进度依赖的重要方式。每个成员都需要关心迭代的进度，轮流组织很容易，提高团队成员在项目中的参与度，每个人都很喜欢，也很乐意，也无形中增加了一次锻炼的机会。但不是每个人都可以组织的很好，或者从迭代整体上起思考组织晨会，很有可能只是一次循规蹈矩的一次主持，昨天做了什么，遇到什么问题，今天做了什么。这也没问题，至少参与了，SM 的职责就是发现这些问题，并把它以更好的方式惠及成员，无形中减轻了自己的压力，也提高了团队的建设。

计划会议是团队最重要的会议之一，会确认团队在未来一段时间来的开发走向、需求方向。计划会议也确定了团队迭代的速率，根据以往团队表现及数据以

及当前客观环境对团队的影响来对团队的迭代速率做出基本的判断，用于计算团队在项目时间内所能完成的故事规模点数，如果和项目范围差距偏大，需要做出及时的调整，增加人员或者缩小项目范围。

在每个版本迭代结束后，团队会聚在一起开回顾会议，目的是回顾一下团队在流程、人际关系以及工具方面做得如何。团队识别出哪些做得好，哪些做得不好，并找出潜在的改进事项，为将来的改进制定计划。所有的会议都是限团队总是在 Scrum 的框架内，改进他们自己的流程。

针对复杂工程性实施问题的计划:1.进行需求分析，经过一段必要的需求沉淀、确定项目时间；2.按照预定时间完成样品开发；3.针对开发过程中出现的 bug 和业务逻辑问题进行修复；4.与产品方进一步交流，对样品进行需求确认；5.完善产品，实时跟进项目，根据用户反馈调整。执行情况：按照计划已经跟进了项目进度，解决了产品经理提出了关于网页的 bug 问题，对各个维护中的网站进行了一系列简单的安全测试，修复使用中问题。由于项目是模块化完成，并且是多个项目同时进行，所以有的问题还没能得到妥善解决，后续将继续跟进项目进程。对于复杂工程问题，除了实施已经提到的执行方案外，还要通过查阅相关电子资料和书籍文献解决。

5.2 开发计划

为了保证项目团队按时保质地完成项目目标，便于项目团队成员更好地了解项目情况，使项目工作开展的各个过程合理有序，有必要以文件化的形式，把对于在项目生命周期内的工作任务范围、各项工作的任务分解、项目团队组织结构、各团队成员的工作责任、团队内外沟通协作方式、开发进度、经费预算、项目内外环境条件、风险对策等内容以书面的方式描述出来，作为项目团队成员以及项目干系人之间的共识与约定，项目生命周期内的所有项目活动的行动基础，项目团队开展和检查项目工作的依据。

针对复杂工程性实施问题，通常在项目流程中有以下步骤 1.进行需求分析，经过一段必要的需求沉淀、确定项目时间；2.按照预定时间完成样品开发；3.针对开发过程中出现的 bug 和业务逻辑问题进行修复；4.与产品方进一步交流，对样品进行需求确认；5.完善产品，实时跟进项目，根据用户反馈调整。

在测试环节中为做好集成测试和验收测试，需为如何组织测试制订实施计划。计划应包括测试的内容、进度、条件、人员、测试用例的选取原则、测试结

果允许的偏差范围等。编写测试 case 主要由 QA 进行。编写测试 case 根据项目需求、开发设计来进行编写。测试 case 包括但不仅限于目录、操作步骤、预期结果、执行结果、优先级等内容。对较复杂的需求、设计编写测试 case 时，适当的分类、分级可以是 case 更清晰明了。测试的过程包括编写测试 case、测试 Case 评审、准入 case，功能测试、性能测试、兼容性测试、安全测试、稳定性测试等环节。在测试工作完成以后，应提交测试计划执行情况的说明， 对测试结果加以分析，并提出测试的结论意见。在 server 端接口测试完成之后，RD 对 server 代码进行上线，上线完成之后 QA 进行线上回归，验证上线代码、上线步骤是否正确。 上线回归主要需要关注以下几点：（1）上线之后，上线代码相关功能是否正常、对其他功能是否有影响；（2）对于老接口的修改，需要回归老版本中使用此接口的功能是否正常；（3）对于多模块上线，每上一个模块，都要及时验证是否对线上版本造成影响。

在工作中还需定时向自己所在的小组负责人提交的项目进展情况 报告，报告应包括进度计划与实际执行情况的比较、阶段成果、遇到的问题和解决的办法以及下个月的打算等。在软件项目开发完成以后，应与项目实施计划对照， 总结实际执行的情况，如进度、成果、资源利用、成本和投入的人力，此外，还需对开发工作做出评价，总结出经验和教训。

5.3 执行情况

黑板报的多次改版、几个重大的项目上线与活动项目开发都按照计划完成开发并通过测试成功上线。线上的回归过程中页都快速的解决了测试提出的一些细节漏洞问题。在开发过程中对各个维护中的模块站进行了一系列简单的安全测试，修复使用中问题。由于项目是模块化完成，并且是多个项目同时进行，所以有的问题还没能得到妥善解决，后续将继续跟进项目进程。对于复杂工程问题，除了实施已经提到的执行方案外，还要通过查阅相关电子资料和书籍文献解决。

在现有的项目优化中，使用技术方法通过重构改进了产品的运行效率。从构建模式、实现方法、代码风格上进行了多方面的知识整理、分析和优化。并以此为契机，强化了效率优化的意识，学习了效率优化的方法，同时，增强了研发中兼顾效率的意识。

第六章 职业素养的学习

6.1 程序员的职业道德

6.1.1 为人正直，忠于职守

人们常说：先做人再做事，它强调的是人的品质问题，一个人做事再优秀，但为人方面却有不良问题，相信他也得不到众人的敬佩与赞赏；企业招聘人才，通常很难找到技术能力及各方面素质都完全符合要求的人选，因此很多企业会将很多的考核点放在人员的道德品质及忠诚度上，一个人的技术水平暂时落后并不可怕，只要他具有良好的品质、并且积极进取忠心耿耿，那么他也是一个不可多得的人才，他会通过自己的努力在较短的时间内达到企业的要求。

在辛苦的开发过程中，一点特别铭记在心：尊重他人的智慧。其他如：树立正确的技能观。决不能利用自己的技能去从事危害公众利益的活动，包括构造虚假信息和不良内容、制造电脑病毒、参与盗版活动、黑客活动等。具有良好的工作责任性，不能以追求个人利益为目的，不随意向他人泄露工作和客户机密。做到这些之后，想必内心将是有一番更为深刻的体验，找准自己的定位，然后付诸行动，将知识化为实践。

我所认识的程序员中大多数人都为人正直，忠厚老实，而且通常羞涩内向，无论是做事还是说话，我们都喜欢直来直去，没有城府，不懂拐弯抹角，也不会油腔滑调，可以说我们是一群可爱的好人，我并不是在自吹自擂，很多时候我都庆幸自己在多年之后还能拥有那份纯朴与善良。总而言之,既然选择了这一行，就好好去爱，忠于软件事业，无论是做一名普通的程序员还是 xx 经理，都要始终坚持严格要求自己，对同事以诚相待,对工作忠于职守,在一家公司做多久就要认真负责多久。

6.1.2 严守商业秘密

目前在国内的 IT 行业，我认为最可怕的并不是盗版问题，而是 IT 人才流失时技术（比如源码、文档等等）也跟着一起流失，这是防不胜防的，有些大公司比如一些外企，做得相对较好，计算机上所有外部设备的接口都被封掉，不得上 INETERNET，不得往外发 EMAIL 或被监控，但不用多说，这种管理方式有利有弊，通常国内的中小 IT 企业很难做到这一点，因此基本上靠员工的自觉与人品，这就正如心理医生必须为病人保守秘密一样，作为一个程序员或 IT 人才，当你

从事某个产品的研发或接手一个项目时，你得为与此产品或项目有关的东西比如源码、文档等资料保密，这些东西无疑是公司的商业秘密。

6.1.3 尊重别人的劳动

这一点不仅仅是我们这一行的，它应是各行各业一个最基本的职业道德，我们从小就被灌输“五讲四美三热爱”的思想，尽管如此，你在大街小巷还是经常看到乱吐口水乱丢垃圾乱撞红灯的现象，这不得不让我们怀疑中国的教育模式与质量，回到 IT 行业，泛滥成灾就是盗版问题，没有使用过盗版软件的 IT 人员真的是极少极少，可能没有，呵呵，在一定的程度上这应不是道德问题，我是一个好人，但有时我不得不用盗版，因为我穷啊，一个软件就花我两个月的薪水，我还用生活吗？有关软件的盗版问题，太多的原因，太多的问题，并非三言两语可以说清道明，我能说的就是：劳动是伟大的，是光荣的，劳动是辛苦的，劳动是要付出心血和汗水的，因此任何人的劳动成果都应受到称赞，任何人的劳动都应受到尊重。

6.2 重视能力的提升

如何做到简单可依赖，我认为是工程师最重要的一环。

web 开发技术一更新命的特别快，新技术、新技巧在不断地被发明出来。我们讲作为 web 工程师的成长，我认为，主要在两个方面，一部分是“能力”，一部分是“知识”。 我个人的观点，能力占百分之八十，知识占百分之二十。

我认为占重点是能力，是非常稳定的，我认为能力是三大块：编程能力、架构能力、工程能力、编程能力，就是用代码解决问题的能力，你编程能力越强，就能解决越复杂的问题，细分又有调试、OS 原理、算法、数据结构、这些的支撑，你才能解决各种麻烦的问题、架构能力，则是解决代码规模的问题，当一个系统足够复杂，你会写每一块，能解决每一个问题，不、接口隔离，也包含认识业务建立抽象模型，也有一些常见的模式，比如经典的 MVC，还有设计层面，面向对象、设计模式、于你能搞定整个系统，这就需要架构能力，架构能力包含了一些意识，比如解耦。最后工程能力，则是解决协作的问题，当系统规模更大，光靠一个人，是没办法完成的，如何保证几个高手互相能够配合好？如何保证项目里面水平最差的人不拖后腿？这个工程化建设，往往会跨越多个业务，以汇报关系上的团队为单位来做。 包括前后端解耦，模块化，质量保证，代码风格。

其实不难看出来，这三项，其实是有顺序的，低、级、小团队，编程能力一项就能应付，越资深的前端，越大的公司和团队，越是需要后面的技能，但是这

里其实资深前端，大团队，对能力的需求，是非常看重的，不是说资深的前端，编程能力就可以变差。社区总会拥有一些声音，对工程能力，对架构能力持有一种抵触的态度，觉得比较虚，觉得不需要。实际上以某些人所在的岗位来说，也没错，毕竟公司、团队的状态确实可能用不到，但是以个人成长的角度来看，就是大错特错。

6.3 建立自己的知识体系

作为一个程序员、一名工程师，“建立自己的知识体系”是非常重要的。

第一步，寻找线索。你要了解一个知识，可以先找一本书，看看别人都写了什么，了解这个知识学习路径、拓展性。找一些比较准确的，你可以确定它真的足够全面的资料当作线索。如用 wsc 的标准、作为线索，我就很有信心同样可能比较适合做的资料，还有一些标准文档的附录，和源代码里的结构定义。

第二步，是建立联系。操作同一组数据，正是面向对象的核心概念，建立对应关系，依据高效原则来面对新知识

第三步，是分类。建立联系以后，我们依据知识之间的联系，进行分类，就可以得到一张图谱，在这个图里面，你就可以非常清楚地知道，哪些知识，是非常重要的，哪些，其实是可以互相替代的，而一旦有你之前没见过的东西，你又能通过把它放到图谱里，来快速理解它，或者找出一些很好的替代方案。

第四步，是追本溯源。当我对一个知识体系的全貌有了概念以后，占了全面两个字，接下来需要确认它的准确性。 很多知识，在社区，会拥有很多的争议，该相信谁呢，这是个问题。 而我的答案，就是追本溯源，去找它最初的讨论和定义。

第七章 对软件工程的认识

7.1 实践中的软件工程

7.1.1 软件过程中的需求

所谓"需求分析"，是指对要解决的问题进行详细的分析，弄清楚问题的要求，包括需要输入什么数据，要得到什么结果，最后应输出什么。可以说，在软件工程当中的“需求分析”就是确定要计算机“做什么”，要达到什么样的效果。可以说需求分析是做系统之前必做的。

而在实际的工作中与学习中最大的差别之一就是对需求的分析理解与确认。在学校内的课程中，往往是急于开发，在需求沉淀和分析中较少花功夫。那么该如何分析需求？当你接受一个项目的设计任务时，可能会遇到比较尴尬的情况，那就是需求有问题！具体的情况可能有：a）需求很简单，几句话或者是一页纸的需求。b）需求很详细，可能有几十页甚至上百页，这些需求是招标书中提供的，或者是客户直接提供的。不要以为有这么多需求是好事，这些需求通常是前后有点矛盾、逻辑有点混乱，甚至是不知所云滴！c）你们有专门的部门或者专职完成了需求工作，提供了一份需求文档。你也不要以为这是好事，因为很可能会出现这样的情况：需求文档的提供者认为自己写的需求文档很牛逼，水平很高；但负责实现的开发部门认为该文档质量太差，比方说：不考虑实现的可能性和难度，没有考虑到客户的真正需求等等。有时候甚至出现开发部门忽略掉需求部门，直接找客户重新调研，重新编写需求文档的情况。

在需求分析过程中有两个必须思考的问题，一是还有哪些不确定或不具体的需求点？二是哪些需求对技术提出了怎样的要求？在需求分析阶段就应当有大致的技术选型，在细节处也不能大意，在细节处一些不重要的需求点可能造成技术的巨大难点，确认这些地方并与需求方进行沟通协调是非常重要的。

7.1.2 软件工程中的分析与设计

在需求评审结束后，在进入模块的设计阶段，通常需要思考以下问题：a）什么角色会用本系统？b）这些角色通过本系统分别能干什么事情？c）角色之间

有可能会有继承关系，请留意父类能干的事情，子类也能干。

UML 是需求分析的有力工具，我的工作中很喜欢用 UML。但你的工作中、你的需求文档中可能会没有 UML 图，不管你们是否用 UML 图，分析需求时都需要从业务的角度思考这些问题。a）什么人（角色）会用这个系统？b）这些人（角色）分别需要用系统的什么功能？

需求分析其实是一个复杂的高难度的话题，回答上述两个问题仅仅是需求分析的第一步而已，我们还需要深入分析业务，包括业务流程、业务数据结构等等。如果之前的需求工作不到位，就需要我们立马开展软件设计工作，其实将会埋下很多地雷，后患无穷。

分析设计功能时，要更加深入思考需求，考虑到各种不同的业务场景和特殊情况。了解业务的热点重点。设计好优秀的数据结构，确定哪些点会有怎样的业务压力，有多少并发量，以此确认缓存方案。同时需要思考需求提出的细节问题,需要思考各种技术方案，找到最平衡、高效、开发成本低的方案。

7.1.3 架构设计

开发的最关键而又最基础的就是架构，架构直接决定了整个项目开发的成败，人人都对架构有所了解，而熟悉者却少之又少。

软件架构是一系列相关的抽象模式，用于指导大型软件系统各个方面的设计。 软件架构是一个系统的草图。软件架构描述的对象是直接构成系统的抽象组件。各个组件之间的连接则明确和相对细致地描述组件之间的通讯。在实现阶段，这些抽象组件被细化为实际的组件，比如具体某个类或者对象。在面向对象领域中，组件之间的连接通常用接口(计算机科学)来实现。 软件体系结构是构建计算机软件实践的基础。与建筑师设定建筑项目的设计原则和目标，作为绘图员画图的基础一样，一个软件架构师或者系统架构师陈述软件构架以作为满足不同客户需求的实际系统设计方案的基础。

上述的都说软件架构的概念，是一个容易理解的概念，多数工程师会从直觉上来认识它，但要给出精确的定义很困难。特别是，很难明确地区分设计和构架：构架属于设计的一方面，它集中于某些具体的特征。在实际开发中架构设计是难度很高的事情，需要你全面考虑需求，考虑工期限制预算限制，考虑项目组人员的水平，而做出的一种平衡。需求并不是由一个个孤立的事件组成的，业务概念、业务流程其实是贯穿多个用户故事的，软件设计应该多从业务概念、业务

流程的角度来思考；表面上看上去一个事件对应一组界面、一组上下文。其实界面之间是很可能有重用和共享部分的；业务逻辑层中的一些类很难将其分拆开来与事件、界面组一一对应，存在交叉、共享和重用的可能。在数据操作层的代码，有可能是用工具自动生成的、通用的；数据层中的某张表，通常会支撑多个事件而不是一个事件。

我们的系统大部分会涉及到多个客户端及服务器，我们需要思考我们的系统需要怎样的客户端及服务器。我们需要思考这些客户端及服务器之间的物理联系方式，例如：是局域网方式、互联网，还是两者都支持？是 HTTP 或是 HTTPS？等等。拆解时可参考分层架构，但需要拆分得更加具体。除了规划好内部各部分的关系，还需要规划内部的各部分与外部之间的关系。在拆解的过程中，问题会越来越多，也会越来越细。拆解过程中也可能会发现之前初步架构设计中不合理或遗漏的地方。。

7.2 软件工程的发展及其影响

软件是由计算机程序和程序设计的概念发展演化而来的，是在程序和程序设计发展到一定规模并且逐步商品化的过程中形成的。软件开发经历了程序设计阶段、软件设计阶段和软件工程阶段的演变过程。

软件工程的发展对人们的生活的改变是方方面面的。在人类文明发展的历史中，人们经历了口口相传，传递信息与故事，经历了文字发明时期的记载与传承，随后人们经历了技术革新的时代，从造纸，到电视、收音机、电脑，手机等多屏、多展示形式的时期后，如今，人们已经进入到第四个时代，毫无疑问互联网。

一种技术从工具属性、从应用层面到社会生活，往往需要经历很长的过程。珍妮纺纱机从一项新技术到改变纺织行业，再到后来被定义为工业革命的肇始，影响东、西方经济格局，其跨度至少需要几十年。互联网也同样。但因为这种影响是滞后的，所以，我们就难免会处于身份的尴尬之中：旧制度和新时代在我们身上会形成观念的错位。越是以前成功的企业，转型越是艰难，这就是“创新者的窘境”——一个技术领先的企业在面临突破性技术时，会因为对原有生态系统的过度适应而面临失败。

我实习所在的作业帮属于互联网教育行业。移动互联网，将会成为教育领域最大的一场革命，它既可以传输文字，也可以传输音频与视频，同时，它还能实

现人们之间的互动与分享，科技+教育是当前教育产业的关键词。

互联网信息近乎零成本扩散和传递的属性，使得互联网教育难以像传统培训机构那样依靠教室的物理实体来限制知识“外泄”，从而实现重复授课并获利。关于互联网教育的问题，我们可以换一种方式，用互联网思维来解决。

优质教育资源的稀缺必须通过互联网科技让它传播得更加广泛，让优质的内容到达山区和需要的人跟前。通过数据和算法的进步，人类就是因为有了梦想，就是因为在克服很多问题的过程中有了设想的能力、空想的能力，在人类遇到每一次灾难和危机的时候才能跨越过去，并且成为更强大的人类。一切互联网和科技带来的优势最终都是效率优势。对于大型机构来说，企业运营效率尤其重要；对于学生和家长来说，学生的学习效率很重要；对于互联网创业公司来说，是找到新的模式帮助效率的提升，只有效率的提升互联网教育的价值才有可能会被无限的放大。

未来的在线教育场景，除了传统的 PC、手机、平板电脑以外，随着智能电视的发展，还会加入家庭客厅的场景，这给婴童和老年教育提供了很好的平台，也是切入家庭在线教育的可能。技术创新是在线教育的基础，工具类产品积累的庞大用户量为有道在线教育的发展奠定了良好的基础。以技术创新为驱动力，坚持深度垂直发展方向，打造精品课程和产品，种种举措让我们看到了在线教育发展的潜力。

通过互联网发展，让过去很难接受教育的各方面人群享受教育。只有教育强，国家才能强。

7.3 软件发展过程中的问题

如今软件行业与人们生活最息息相关的一方面就是互联网，作为人们生活和软件接触的接口，互联网行业的发展依旧存在许多问题。

软件、互联网在给人们带来诸多便捷和机遇的同时，也带来了许多的问题和挑战。首先，互联网技术与传统产业的融合对信息安全的要求极高。如果出现安全问题，不仅会带来巨大的经济损失，还有可能危及国家利益和人民生命安全。从这个角度来说，互联网技术是把双刃剑。在优化传统产业、为生活带来便利的同时，一旦在技术漏洞、人员管理、数据备份等某个环节稍有闪失，就会出现满盘皆输的局面。

其次，“互联网+”模式所需要的数据开放程度不够。一方面国家、行业、企

业以邻为壑，各自为战，通过独立标准和封闭技术系统阻挡网络联通和数据流动；另一方面，公共数据分散在政府各个部门中不成体系，而且部门壁垒森严，存在难打通、数据开放不足等问题。数据的开放不足会阻碍创新，而数据的过分开放则又会带来安全隐患。所以政府部门需要在保证信息安全的前提下，更多地开放数据，找到其中的平衡点。

最后，信息行业迅速发展也暴露并放大了一些传统行业存在的风险。由于在互联网与传统行业向融合产生的新运营模式发展迅猛，而相关的法律法规以及其他制度都相对来说不够完善，导致一些企业铤而走险，对风险未加以足够的控制。以互联网金融行业为例，互联网金融是个崭新的行业，没有充足的经验可以借鉴，目前的准入门槛低、发展良莠不齐。信用风险、流动性风险、操控风险相比于传统金融行业更大、并且没有标准化的风控体系，出现了一些经营不善、跑路等现象。在互联网金融飞速发展的同时，如何完善中国互联网金融监管、做好互联网金融征信系统、建立严格的风控体系，这些问题都急需解决。

现阶段，持续优化基础设施，传统行业与互联网行业融合转型非常重要。互联网网络架构和布局全面优化提升将进一步增强宽带网络基础设施的支撑能力，改善互联网用户的上网体验，并将优化互联网布局，提高全网的通信质量和安全性能。同时，还将带动移动互联网、物联网、云计算、大数据等新兴产业的规模集聚和创新突破，为促进当地和周边地区经济社会转型升级和区域协调发展发挥积极作用。同时促进大数据产业及信息化发展、建议数据开放并制定相应法律和政策。随着云计算模式的兴起以及社交网络、移动计算和传感器等新渠道和技术不断涌现，数字信息的类型和来源变得愈加复杂，规模急剧扩大，大数据时代已经到来。我国信息化法制建设中，应紧紧抓住该历史机遇，制定大数据开放相关立法及政策，积极开展大数据技术应用，充分发挥示范效应，促进大数据产业发展及我国信息化建设。

第八章 结束语

8.1 本文内容

在本文中，精简的描述了我实习单位、与实习岗位的概括。针对我所在公司的岗位的环境、发展状况、现阶段发展的阻力与前景等问题都做了简要的概述。对于实习过程中遇到的复杂问题，如黑板报多次改版中做得数据结构变更、缓存优化等问题都进行了归纳，并对实施方案做了详尽的描述。在项目的实施过程中，针对了这些复杂工程的问题进行推理分析，做出的设计满足需求的总体设计和详细设计。在实现第三方登陆的过程中，对流程设计方案进行优选，设计较为合理，相关编码经过测试虽有漏洞最终也都经过部分修改都符合标准进行了上线。在项目结束后对项目结果进行分析和解释，并进行数据分析、功能的稳定性判断、用户行为分析。通过信息综合得到合理有效的结论，与企业导师讨论当前版本问题和优化点，并以当前版本的功能反馈来帮助确定下一版本能有更有价值、展性的开发方向。并在项目完结后回顾自己的技术不足点，以此进行更一步的拓展学习，帮助自己在职业道路上更好的发展。

8.2 顶岗实习项目课题有待进一步解决的问题及方向

在项目的开发过程中，更加了解了 db，缓存等技术。更加了解了公司各方向业务线下的底层架构。但是在更深层的技术上，很多技术细节上，考虑的都非常不到位。例如在使用缓存的初期，没有好好的考虑好缓存穿透、缓存并发和缓存失效等问题。

对于维护性的考虑不足,项目是做出来了,但是由于文档不全,代码格式混乱,代码的复用性不高，导致 2 次开发非常困难。数据结构的重复使用与拓展性没有考虑好，应该在项目初期思考自己的项目的走向，确定项目拓展后现有的数据结构能否随之很好的改变。初期开发的模块耦合度过高,一个一个模块应该相对独立,尽量一个模块只完成一个功能,避免相互牵扯。

在作为一个工程师的心态上页存在着不足，开发完一个产品提交测试的时候,总是希望没有错误,但是大家都知道是不可能的,当测试员提出问题的时候,刚刚开始总是感觉很难受,特别是一些小问题也写入 bug 文档,感觉就是针对自己。这个就是心态问题,其实做测试的更难,只要你知道这是工作,对事不对人,

有问题就改那么就没什么事情了.

8.3 对软件工程实践以及软件工程领域发展的认识

软件工程实践是对软件技术的实际运用，是非常重要的一环，光了解书本上的理论知识不是目的，最重要的还是对所学知识的实际运用。软件工程实践就是检验所学知识水平的过程，也是我们所提倡的“理论源于实践，实践创造价值”的具体实施。

《计算机科学技术百科全书》中的定义：软件工程是应用计算机科学、数学及管理科学等原理，开发软件的工程。软件工程借鉴传统工程的原则、方法，以提高质量、降低成本。其中，计算机科学、数学用于构建模型与算法，工程科学用于制定规范、设计范型、评估成本及确定权衡，管理科学用于计划、资源、质量、成本等管理。目前相比起来较普及的一种定义为：软件工程是研究和应用如何以系统性的、规范化的、可定量的过程化方法去开发和维护软件，以及如何把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来。

软件工程发展至今，已较成熟。原理，规范，流程，技术都有相应的详尽规定。有理由相信，软件工程会在当今以及未来人类发展进程中依然会发挥不可替代的作用。作为一名从事软件行业的学生，我会在今后的学习生涯中努力汲取知识养分，为成为一名优秀的软件工程师打下坚实基础。

8.4 本人毕业设计（顶岗实习）收获及体会

又是经历了数月的实习，不算长也不算短。经过这半年多的工作、学习愈发的理解了何为团队、何为工程。在技术上对于 web 开发的整个技术体系也有了更完善的理解与应用。在工作流程中对于需求分析、功能设计、代码实现、测试等整个流程也进一步熟悉。

实习是把我们在学校所学的理论知识，运用到客观实际中去，是自己所学到的理论知识有用武之地，只学不实践，那么所学的就等于零。理论应该与时间相结合。另一方面，实践可以为以后找工作打基础。通过这段时间的实习，学到一些在学校里学不到的东西。因为环境不同，接触的人与事不同，从中学到的东西自然就不一样。要学会从实践中学习，从学习中时间。快速发展的社会导致了对于人才的要求就会越来越高，我们不只要学好学校所学到的知识，还要不断充生活中，实践中学其他知识，不断从各方面武装自己，才能在竞争中突出自己，表现自己。

在短暂的实习过程中，面对又一次的实习与工作场景的变化，我再一次理解了自己所学的知识的肤浅和在实践运用中知识的匮乏，刚开始的一段时间里，对一些工作无从下手，茫然不知所措，这让我感到非常的难过。在学校总以为自己学的不错，一旦接触到时间，才发现自己知道所掌握的远远不够。而同时这一段经历我想一定会成为将来走向正式岗位的坚实的基石。

参考文献

[1] 侯建彬.创业有所为[R]杭州:浙江大学，2016-08-21

[2] 尹博学. 百度数据库中间层详解[R]北京：北京国际会议中心，2012-04-13

[3] 施瓦茨,扎伊采夫.高性能 MySQL（第 3 版）[M]. 北京：电子工业出版社 2013-05-01.

[4] 姜承尧. MySQL 技术内幕:InnoDB 存储引擎(第 2 版）[M]. 北京：机械工业出版社 2013-06-05

[5] Baron Schwartz, Vadim Tkachenko, Peter Zaitsev.High performance MySQL [M] Cambridge O'Reilly Media 2012-04-02

致谢

本论文的工作是在我的企业指导导师陆海霞老师与院内代管教师刘玓老师的悉心指导下完成的。在我报告撰写中，刘玓老师为我提供了无私帮助与指导评价，将使我一直受益。在这数月的实习中，我的企业导师陆海霞她一直积极的支持着我的工作，为我答疑解惑、指点迷津，告诉我一个合格工程师应该具备着什么、应该如何去更好的学习。其次要感谢工作中其他的同事、领导们，在与他们共事的日子里，时不时接受他们的帮助指导，每一天都有着点点滴滴的收获。此外还要感谢祝愿他们在工作和学习上取得更大的成绩！

外文资料原文

High performance MySQL

II. Optimization Overview

Any database application eventually hits hardware limits as the database becomes more and more busy. A DBA must evaluate whether it is possible to tune the application or reconfigure the server to avoid these bottlenecks, or whether more hardware resources are required. System bottlenecks typically arise from these sources: Disk seeks,CPU cycles, [4].

外文资料译文

高性能 MySQL

二．优化摘要

随着数据库请求越来越繁忙，任何数据库应用程序最终都会在硬件层面上遇到瓶颈限制。 DBA 必须评估是否可以调整应用程序或重新配置服务器以避免这些瓶颈，或者是否需要更多的硬件资源。 系统瓶颈通常源于这些来源磁盘寻道、CPU 处理和主存限制。