拓展用户体系

（单点登陆、跨域session共享、第三方登陆、重构全部接口、数据迁移）

原业务线：手机号是作为唯一登录手段

登录/注册相关两张表

tblUserRegister：以手机号为key做分表，存储用户password，session,uid,封禁状态等信息

tblAppUcloud：已uid为key做分表，存储用户头像、昵称、性别、财富值等才用户信息

**Sso实现**

1. 某一域名下登录
2. 用户前端页面输入手机号账号登录
3. Server接收到数据进行基础校验检查手机号是否注册，若未注册返回，若已注册则读取用户的session,password等信息,(session中存储的通行证zybuss)
4. 对用户的密码做校验操作，若用户密码校验失败则在redis中将用户尝试次数+1，若尝试次数超过限制则对用户账号进行短时间封禁操作并清空尝试次数
5. 密码校验成功，生成临时通行证zybuss。然后用以zybuss为key，用户uid,phone,登录时间,登录方式等信息为value存储在redis中。然后将第二步遍历的session,将这些zybuss为key的redis对象的value置为相应的踢出状态，踢出失败则进行异步删除操作，然后将当前的zybuss写入表中的session字段
6. 最后将当前生成的zybuss放入cookie返回给客户端
7. 如果是浏览器登录成功，还会进行跨域共享session，前端会发送请求到后端获得需要同步的cookie的域名。（也可以前端做）
8. 然后以jsonp的格式跨域分别请求上一步获得的域名，服务端收到请求校验后cookie，（根据redis）正确校验后在不同的域名setcookie，并根据请求的callback参数拼接jsonp数据返回给前端

Session存在db中，而zybuss存储在redis中，因为session只有在用户登录、注册、修改密码等操作时需要读取，不需要存在redis中

其他的登录接口还有tokenlogin（验证码登录，逻辑也类似）

注册：1）新增用户信息

2）新增session

问题：步骤1是事物的（生成session逻辑独立，且依赖与uid），但1和2不耦合，但这么处理挺好

事物1：IdAlloc分配一个uid。

2：开启事物

3：register表新增数据，uid与phone映射关系。

4: uloud新增用户信息

作业帮内所有的请求在action层基类会调用Saf\_SmartMain::getUserInfo();读取session中的信息

新增逻辑

oauth2.0：(oauth2.0分为三种角色—客户端、用户和资源服务器，WeChat和qq是不同的授权模式。授权码模式客户端分为客户端的客户端和客户端的server，简化模式则没有)

1. 端吊起第三方sdk
2. 用户授权（qq返回access\_token，ouid,wechat返回code（临时token））
3. 客户端发送上一步返回的数据给服务端
4. 如果是wechat根据code和服务端存储的secret请求微信server端得到token和ouid
5. Server端根据ouid获取用户的注册状态
6. 若已注册生成session返回，若未注册则请求获取用户信息，然后以一个随机生成的key，在redis中存储ouid，然后将这个key和获取的用户信息返回给客户端
7. 客户端进入绑定手机页面，用户可选择绑定新手机号，老手机号和不绑定手机号，用户操作完将信息（手机号）post后端
8. 后端若用户不绑定手机号则执行事物A，绑定手机则确认手机注册状态。若绑定新手机执行事物B，绑定老手机执行事物c
9. 生成session返回

事物A：

Idalloc分配uid,

新增uid和ouid映射关系到oauth表中，

新增ouid与uid反映射关系到map表中

新增 以uid来分表，存储uid与session，forbien信息的新表password中

新增 用户信息到以uid分表的ucloud表中

事物B:

Idalloc分配uid,

新增uid和ouid映射关系到map表中，

新增ouid与uid反映射关系到oauth表中

新增 以uid来分表，存储uid与session，forbien信息的新表password中

新增 用户信息到以uid分表的ucloud表中

新增 phone与uid的映射关系register表中

事物C：

得到用户手机注册的uid，

根据uid从map表中查询用户已绑定的第三方类型，若已绑定展示已经绑定过了。

若未绑定，

新增uid和ouid映射关系到map表中，

新增ouid与uid反映射关系到oauth表中

拆分数据表：

原本的register（phone,uid,session）

拆分为password(uid,session)+register(phone+uid),需要数据迁移

修改业务接口：手机号登录、注册、手机号修改绑定、修改密码、第三方登录注册绑定，各类验证码发送时的关系验证

问题：

1.老用户第三方注册若不绑定手机号，之后会有两个uid，账号无法合并

2.许多业务线强制需要手机号，需跳转入绑定手机号页面（与其他业务线确认）

数据迁移：

条件：一主多重，凌晨，user独立的库，dbProxy

1. 编写数据迁移脚本(提交qa测试)
2. 排查所有接口修改代码（提给qa测试），12点开始db写操作，变成双重操作，直接写原库与生成新表sql发送至消息队列（待定）存储。
3. 线上建好新表
4. 上线双重逻辑的代码，消息队列发送到proxy但不推送到server
5. 12点停掉一个从库
6. 找到与nmq匹配点，记录
7. 用脚本迁移从库中的存量数据到现在还在运行的主库
8. 存量数据处理，Nmq打开发送至任务A执行sql。
9. 待增量sql消耗完
10. 查看消耗脚本日志
11. 日志执行正确，脚本校验数据是否一致
12. 上tips机器自己测试看是否正常，然后上一台机器，看日志是否正常
13. 正常上全量双写读新。
14. 然后回去休息
15. 中午来看有没有人报问题，日志是否正常。然后切换成单读

迁移脚本：

20个分表20个进程。每次20000分片读出，拼接成新sql，执行，sleep(0.1)处理完，释放分片内存。

内网sql执行任务：

一个php脚本，参数是一个token和相应的sql去执行。

问题：

执行出错怎么办？

sql记录出错的sql，再根据uid从原库迁移到现库。

存量数据出错会打warning日志，一直在监控的。

nmq出错怎么办？(线上均限制为4000-5000qps，此时稳定性为99.9999)

队列有稳定性的

nmq挂了怎么办？

改天做这件事

迁移花了多久？

存量两个小时左右，增量和测试一个多小时

为了保证主从延迟做了哪些控制？

作业帮dbuser表在高峰期每秒读写报警数（值不告诉）来控制

怎么控制执行速度的？

增量nmq可以控制速度，并且量不大。存量数据脚本迁移，边度边写，一次读写

Nmq介绍

我的工作：业务重写，迁移脚本，增量处理，错误问题处理。

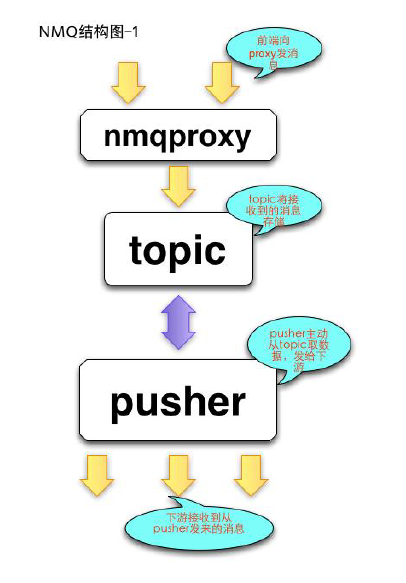
哪些优化：多进程

注释：idalloc(‘name’,’id’),name主键

Alloc实现：select id from idalloc where name=xx for update

Id++

Insert into idalloc values(name,newid);



Proxy无状态，只负责推送到topic

Topic根据业务线划分，有主从**存储消息**

Pusher 推送

提供给第三方的支付与登录：

端内接入掌阅的sdk，掌阅sdk内部是一个书城，提供较为独立的sdk和购买行为。

登录：

1. sdk调起我们作业帮端内的一个授权页面
2. 授权页面校验用户登录状态，若已登录请求authorize接口，未登录走通用登录流程再请求authorize接口
3. Authorize接口根据用户session，和页面传来的平台标识会生成一个唯一的用户标识ouid和token返回客户端，客户端回将这些参数回传给sdk
4. Sdk拿到这些数据后post到他们的服务端，服务端将ouid+token+平台分配的标+需要请求的用户信息传给服务端，服务端进行校验，返回用户信息给第三方服务端。

支付：

1. sdk将商品信息传到端上中转接口
2. 中转接口得到商品id请求后端，后端根据存储的映射关系返回作业帮商品id
3. 中转接口根据作业帮商品id，带着sdk传来的商品信息跳转到支付页面生成订单
4. 通用支付流程
5. 支付往回调业务接口
6. 业务接口将订单id，ouid，支付金额，商品名称，支付方式，第三方平台名称，第三方传的商品信息等，按key的字典序排序依靠公钥进行RSA（RSA\_WITHSHA1）签名然后base64 encode（去除特殊字符）得到签名串。然后将签名的参数+签名+签名方法post掌阅的服务端，如果成功则完成订单，未成功则以每五分一次的方式轮询。

黑板报

介绍：黑板报其实就是一个feed流形式的用户内容阅读模块功能。内容分为自编辑和第三方获取两大模块。第三方主是与由腾讯天天快报和百度秒懂百科合作。依据不同的内容可以进行多个栏目进行展示。日均uv大概是百万左右的水准，不算太高不算太低。

模块的用户行为主要就是列表的文章浏览，和文章详情页的阅读。在列表的首屏会作为热点进行展示。而黑板报文章则是以最新发布最顶的进行缓存。

其余的就是如用户浏览量展示，用户点赞、用户弹幕等相关功能。

浏览量则是以redis 存储，每隔1000 flush to db。弹幕也是如此

文章缓存则是根据文章id读缓存，缓存不存在读db，db里会有文章上线时间，两天内的文章进行缓存。

弹幕：默认关闭请求量不大，所有不缓存。要缓存的的话就以redis 的list进行缓存。

第三方内容腾讯是主动去拉取，

百度百科是主动推送给我们。

都会进入一个审核表，审核完进入黑板报的表。

黑板报：一个自编辑的文章feed流

初版：一个富文本编辑文章列表，前端请求rn，pn，后端select xx from xx order by id limit rn,pn;无缓存

问题：数据量多执行效率低，新增数据会导致 limit rn，pn不一致。

初改版：增加弹幕，点赞，浏览量，修改查询，增加缓存。

点赞浏览量：写：默认写redis，key incr 失败打log直接返回;if value % num == 0 {flush to db};两步操作不可倒置

读：读redis，(not false && not null) return,else 读db,写缓存

弹幕：

量大，每篇大概能有3000弹幕。不可能直接全部缓存用lpush 和rpop缓存50条。

查询修改：首屏select order by id limit pn; 刷新，加载更多都是传文章id

缓存请求->首屏缓存20条

详情页：读缓存，缓存不存在读db，db看上线时间，若上线时间两天内，则缓存。

二改版:

内容分频道，原本就有保留type字段，改版接入腾讯（还尝试了头条），加广告。crontab定时脚本获取新闻做好类型映射插到我们的表中。结果上线有问题，腾讯质量太差，新增审核功能，多了一张审核表，审核之后再插入文章表。

三改版:

样式改版，增加推荐栏目，增加定时上线

四改版：

增加视频，分为秒懂百科和内部视频。进入一个视频详情页会展示10个同类视频。按类别存储视频id，然后shuffle，然后select。

广告审核：

请求广告，广告服务商返回广告相关信息，调用我的服务。

1. 得到请求得到广告的信息。
2. 根据广告服务商以不同的方式匹配跳转链接里的落地页，根据落地页+广告素材做MD5为key，审核状态为value，若已审核将审核状态返回。若未审核，则返回未审核状态。缓存不存在读db，然后写db。若db也不存在，则未进入审核池
3. 然后根据落地页+广告服务商先根据缓存判断广告是否在白名单内，缓存不存在则读db，若是广告主白名单返回通过，若非白名单广告主且广告未在审核池则插入并同时设置缓存广告状态为未审核，若广告主不存在db插入广告主并置缓存（若广告不存也同样插入）。

审核后台：

广告主列表，

广告列表。

问题：不同的广告平台落地页不一样，还有相同广告图片但是url不一样造成审核困难。

主从延迟：

同学圈好友申请。（圈子优化）

三周年

Hive sql跑数据，生成sql灌入。

减负：rpc调用，缓存记录

检索页反馈：

在检索结果的时候要加一个判断，是否出现反馈。

检索->前端请求-> ocr请求->判断qps->nmq/同步检索rpc请求

服务器处理完队列中的检索会存到redis中，若前端判断未返回结果，轮询服务器缓存异步接口，读取缓存后返回。

每日一练

普通增删查改

评论：复用文章的，所以就新增一个假文章，然后用户评论这篇文章。文章和每日一练一一对应。

缓存-当日缓存。

抽取用户bug：没有用户，随机生成用户。随机数取得不对。

微课

主要是一个多对多的索引，

增加时的事务：

1. 添加微课商品，因为有价格策略，所以需要添加多个商品（单独一个事务）
2. 添加任务
3. 添加子任务
4. 添加筛选条件单独表 多对多 gradeID
5. 修改商品ext

问题：商品单独事务无法回滚。

发现两个start transaction会强制提交上一个

修改微课/微任务

微任务与子任务多对多（子任务对应试卷），微课程与多个子课程对应

不能简单的把旧子课程修改或删了，然后新增子课程。不然用户练习进度没了。

所以已经进行的子课程不能修改，前端会比较时间让用户无法编辑，但不能只由前端控制，

所以前端post老子课程有独立id，新子课程没有独立id，读取所有老课程及其时间戳，

逐个前端传过来的子课程比较时间，确认是否可以修改，可以修改的update，不可以修该回滚，新子课程没有id则新增。

定时：功能子课程有推送时间，按时推送，与pm课程有推送时间，设置为整点。整点开始推送。

推送复杂点：部分用户推送失败，重推，如何快速确定需要推送的子课程

原本逻辑，遍历所有课程，查找课程订阅信息表（里面有推送状态），读取需要推的子任务，推送给用户。

漏洞：部分用户推送失败。

直接遍历订阅信息，每个订阅信息里存储一个next push time,每次读取所有next push time <当前 and status not finish的任务，然后反着找到任务子任务信息推送任务，如果全部推完finish，如果没推完，置next\_push\_time为下个推送的任务时间。