





11 | 架构设计流程：设计备选方案

李运华

00:00 / 11:58

上一期我讲了架构设计流程第1步识别复杂度，确定了系统面临的主要复杂度问题后，方案设计就有了明确的目标，我们就可以开始真正进行架构方案设计了。今天我来讲讲**架构设计流程第2步：设计备选方案**，同样还会结合上期“前浪微博”的场景，谈谈消息队列设计备选方案的实战。

架构设计第2步：设计备选方案

架构师的工作并不神秘，成熟的架构师需要对已经存在的技术非常熟悉，对已经经过验证的架构模式烂熟于心，然后根据自己对业务的理解，挑选合适的架构模式进行组合，再对组合后的方案进行修改和调整。

虽然软件技术经过几十年的发展，新技术层出不穷，但是经过时间考验，已经被各种场景验证过的成熟技术其实更多。例如，高可用的主备方案、集群方案，高性能的负载均衡、多路复用，可扩展的分层、插件化等技术，绝大部分时候我们有了明确的目标后，按图索骥就能够找到可选的解决方案。

只有当这种方式完全无法满足需求的时候，才会考虑进行方案的创新，而事实上方案的创新绝大部分情况下也都是基于已有的成熟技术。

- NoSQL：Key-Value的存储和数据库的索引其实是类似的，Memcache只是把数据库的索引独立出来做成了一个缓存系统。
- Hadoop大文件存储方案，基础其实是集群方案+ 数据复制方案。
- Docker虚拟化，基础是LXC（Linux Containers）。
- LevelDB的文件存储结构是Skip List。

在《技术的本质》一书中，对技术的组合有清晰的阐述：

新技术都是在现有技术的基础上发展起来的，现有技术又来源于先前的技术。将技术进行功能性分组，可以大大简化设计过程，这是技术“模块化”的首要原因。技术的“组合”和“递归”特征，将彻底改变我们对技术本质的认识。

虽说基于已有的技术或者架构模式进行组合，然后调整，大部分情况下就能够得到我们需要的方案，但并不意味着架构设计是一件很简单的事情。因为可选的模式有很多，组合的方案更多，往往一个问题的解决方案有很多个；如果再在组合的方案上进行一些创新，解决方案会更多。因此，如何设计最终的方案，并不是一件容易的事情，这个阶段也是很多架构师容易犯错的地方。

第一种常见的错误：设计最优秀的方案。

很多架构师在设计架构方案时，心里会默认有一种技术情结：我要设计一个优秀的架构，才能体现我的技术能力！例如，高可用的方案中，集群方案明显比主备方案要优秀和强大；高性能的方案中，淘宝的XX方案是业界领先的方案……

根据架构设计原则中“合适原则”和“简单原则”的要求，挑选合适自己业务、团队、技术能力的方案才是好方案；否则要么浪费大量资源开发了无用的系统（例如，之前提过的“亿级用户平台”的案例，设计了TPS 50000q的系统，实际TPS只有500），要么根本无法实现（例如，10个人的团队要开发现在的整个淘宝系统）。

第二种常见的错误：只做一个方案。

很多架构师在做方案设计时，可能心里会简单地对几个方案进行初步的设想，再简单地判断哪个最好，然后就基于这个判断开始进行详细的架构设计了。

这样做有很多弊端：

- 心里评估过于简单，可能没有想得全面，只是因为某一个缺点就把某个方案给否決了，而实际上没有哪个方案是完美的，某个地方有缺点的方案可能是综合来看最好的方案。
- 架构师再怎么牛，经验知识和技能也有局限，有可能某个评估的标准或者经验是不正确的，或者是老的经验不适合新的情况，甚至有的评估标准是架构师自己原来就理解错了。
- 单一方案设计会出现过度辩护的情况，即架构评审时，针对方案存在的问题和疑问，架构师会竭尽全力去为自己的设计进行辩护，经验不足的设计人员可能会随询处理。

因此，架构师需要设计多个备选方案，但方案的数量可以说是无穷无尽的，架构师也不可能穷举所有方案，那合理的做法应该是什么样的呢？

更多一手资源请添加QQ/微信1182316662

- 备选方案的数量以3 ~ 5个为最佳，少于3个方案可能忽略方案的多样性，考虑不周全；多于5个则会造成大量的精力和时间，并且方案之间的差别可能不明显。
- 备选方案的差异要比较明显。例如，主备方案和集群方案差异就很明显，或者同样是主备方案，用ZooKeeper做主备决策和用Keepalived做主备决策的差异也很明显。但是都用ZooKeeper做主备决策，一个检测周期是1分钟，一个检测周期是5分钟，这可不是架构上的差异，而是细节上的差异了，不适合做成两个方案。
- 备选方案的技术不要只局限于已经熟悉的技术。设计架构时，架构师需要将视野放宽，考虑更多可能性。很多架构师或者设计师积累了一些成功的经验，出于快速完成任务和降低风险的目的，可能自觉或者不自觉地倾向于使用自己已经熟悉的技术，对于新的技术有一种不放心的感觉。就像那句俗语说的：“如果你手里有一把锤子，所有的问题在你看来都是钉子”。例如，架构师对MySQL很熟悉，因此不管什么存储都基于MySQL去设计方案，系统性能不够了，首先考虑的就是MySQL分库分表，而事实上也许引入一个Memcache缓存就能够解决问题。

第三种常见的错误：备选方案过于详细。

有的架构师或者设计师在写备选方案时，错误地将备选方案等同于最终的方案，每个备选方案都写得很细。这样做的弊端显而易见：

- 耗费了大量的时间和精力。
- 将注意力集中到细节中，忽略了整体的技术设计，导致备选方案数量不够或者差异不大。
- 评审的时候其他人会被很多细节给绕进去，评审效果很差。例如，评审的时候针对某个定时器应该是1分钟还是30秒，争论得不可开交。

正确的做法是备选阶段关注的是技术选型，而不是技术细节，技术选型的差异要比较明显。例如，采用ZooKeeper和Keepalived两种不同的技术来实现主备，差异就很大；而同样都采用ZooKeeper，一个方案的节点设计是/service/node/master，另一个方案的节点设计是/company/service/master，这两个方案并无明显差异，无须在备选方案设计阶段作为两个不同的备选方案，至于节点路径究竟如何设计，只要在最终的方案中挑选一个进行细化即可。

设计备选方案实战

还是回到“前浪微博”的场景，上期我们通过“排查法”识别了消息队列的复杂性主要体现在：高性能消息读取、高可用消息写入、高可用消息存储、高可用消息读取。接下来进行第2步，设计备选方案。

1. 备选方案1：采用开源的Kafka

Kafka是成熟的开源消息队列方案，功能强大，性能非常高，而且已经比较成熟，很多大公司都在使用。

2. 备选方案2：集群 + MySQL存储

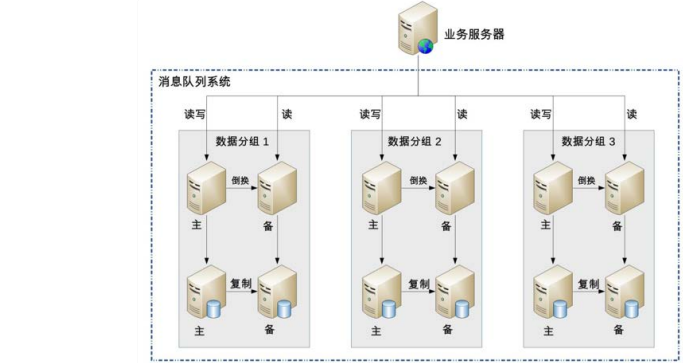
首先考虑单服务器高性能。高性能消息读取属于“计算高可用”的范畴，单服务器高性能备选方案有很多种。考虑到团队的开发语言是Java，虽然有人觉得C/C++语言更加适合与高性能的中间件系统，但架构师综合来看，认为无须为了语言的性能优势而让整个团队切换语言，消息队列系统继续用Java开发。由于Netty是Java领域成熟的高性能网络库，因此架构师选择基于Netty开发消息队列系统。

由于系统设计的QPS是13800，即使单机采用Netty来构建高性能系统，单台服务器支撑这么高的OPS还是有很大风险的，因此架构师选择采取集群方式来满足高性能消息读取，集群的负载均衡算法采用简单的轮询即可。

同理，“高可用写入”和“高性能读取”一样，可以采取集群的方式来满足。因为消息只要写入集群中一台服务器就算成功写入，因此“高可用写入”的集群分配算法和“高性能读取”也一样采用轮询，即正常情况下，客户端将消息依次写入不同的服务器；某台服务器异常的情况下，客户端直接将消息写入下一台正常的服务器即可。

整个系统中最复杂的是“高可用存储”和“高可用读取”，“高可用存储”要求已经写入的消息在单台服务器宕机的情况下不丢失；“高可用读取”要求已经写入的消息在单台服务器宕机的情况下可以继续读取。架构师第一时间想到的就是可以利用MySQL的主备复制功能来达到“高可用存储”的目的，通过服务器的主备方案来达到“高可用读取”的目的。

具体方案：



简单描述一下方案：

- 采用数据分散集群的架构，集群中的服务器进行分组，每个分组存储一部分消息数据。
- 每个分组包含一台主MySQL和一台备MySQL，分组内主备数据复制，分组间数据不同步。
- 正常情况下，分组内的主服务器对外提供消息写入和消息读取服务，备服务器不对外提供服务；主服务器宕机的情况下，备服务器对外提供消息读取的服务。
- 客户端采取轮询的策略写入和读取消息。

3. 备选方案3：集群 + 自研存储方案

在备选方案2的基础上，将MySQL存储替换为自研实现存储方案，因为MySQL的关系型数据库的特点并不是很契合消息队列的数据特点，参考Kafka的做法，可以自己实现一套文件存储和复制方案（此处省略具体的方案描述，实际设计时需要给出方案）。

可以看出，高性能消息读取单机系统设计这部分时并没有多个备选方案可选，备选方案2和备选方案3都采取基于Netty的网络库，用Java语言开发，原因就在于团队的Java背景约束

更多一手资源请添加QQ/微信1182316662

更多一手资源请添加QQ/微信1182316662

了备选的范围。通常情况下，成熟的团队不会轻易改变技术栈，反而是新成立的技术团队更加倾向于采用新技术。上面简单地给出3个备选方案用来示范如何操作，实际中要比上述方案复杂一些，架构师的技术储备越丰富、经验越多，备选方案也会更多，从而才能更好地设计备选方案。例如，开源方案选择可能就包括Kafka、ActiveMQ、RabbitMQ；集群方案的存储既可以考虑用MySQL，也可以考虑用HBase，还可以考虑用Redis与MySQL结合等；自研文件系统也可以有多个，可以参考Kafka，也可以参考LevelDB，还可以参考HBase等。限于篇幅，这里就不一一展开了。

小结
今天我为你讲了架构设计流程的第二个步骤：设计备选方案。基于我们模拟的“前浪微博”消息系统，给出了备选方案的设计样例，希望对你有所帮助。

这就是今天的全部内容，留一道思考题给你吧，除了这三个备选方案，如果让你来设计第四个备选方案，你的方案是什么？

欢迎你把答案写到留言区，和我一起讨论。相信经过深度思考的回答，也会让你对知识的理解更加深刻。（编辑乱入：精彩的留言有机会获得丰厚福利哦！）



公号-Java 大后端	2018-05-23
设计备选方案心得	
经过架构设计流程第 1 步——识别复杂度，确定了系统面临的主要复杂问题，进而明确了设计方案的目标，就可以开展架构设计流程第 2 步——设计备选方案。架构设计备选方案的工作更多的是从需求、团队、技术、资源等综合情况出发，对主流、成熟的架构模式进行选择、组合、调整、创新。	
1. 几种常见的架构设计误区	
(1) 设计最优秀的方案。不要面向“简历”进行架构设计，而是要根据“合适”、“简单”、“演进”的架构设计原则，决策出与需求、团队、技术能力相匹配的合适方案。	
(2) 只做一个方案。一个方案容易陷入思考问题片面、自我坚持的认知陷阱。	
2. 备选方案设计的注意事项	
(1) 备选方案不要过于详细。备选阶段解决的是技术选型问题，而不是技术细节。	
(2) 备选方案的数量以 3~5 个为最佳。	
(3) 备选方案的技术差异要明显。	
(4) 备选方案不要只局限于已经熟悉的技术。	
3. 问题思考	
由于文中已经从开源、自研的角度提出了架构设计方案；为此，从架构设计三原则出发，也可考虑第四个备选方案：上云方案，该方案是直接采用商业解决方案，就好比阿里前期采用IOE类似。	
如果是创业公司的业务早、中期阶段，可直接考虑采用阿里云/腾讯云，性能、HA、伸缩性都有保证。	
此外，在文中备选方案1 - 开源方案中，如果从提供另外一种视角看问题的角度出发，个人会更加倾向于RabbitMQ。首先，RabbitMQ与Kafka都同样具备高可用、稳定性和高性能。但是，通过一些业界测试比较，RabbitMQ比Kafka更成熟、更可靠；其次，在高性能指标方面，Kafka胜出，kafka设计的初衷是处理日志，更适合IO高吞吐的处理。但是，对于“前浪微博”系统的QPS要求，RabbitMQ同样可以驾驭。为此，综合来看，会更加倾向于RabbitMQ。	
最后，通过本文再结合自己做技术最大的感悟是：做事情永远都要有B方案。	
作者回复	2018-05-23
最好是“三方案”，又叫“第三选择”，可以防止思维狭隘，目光短浅，思维盲区等决策陷阱	
陈华英	2018-05-22
文中的方案写得清晰，有理有据。只是在主备切换的时候显得略了一点，上文中提到用zookeeper或者keepalived，可以分为两个备选方案。keepalived比较简单，主要是实现虚IP的漂移，这个对客户端是透明的。如果是zookeeper的话，客户端还需要从zk上读下主节点	
作者回复	2018-05-22
赞，被你发现了，限于篇幅无法细化，主备切换这部分可以有几个备选，zk是一种方式，还有不用zk的也可以。	
星火燎原	2018-05-22
高可用消息存储和读取可以采用mongo和redis 这么高的qps很难保证消息不丢 那么可以采用有消息确认机制和消息回溯的MQ 或者自研rpc的时候考虑消息发送失败的时候重新选择节点然后落盘	
作者回复	2018-05-22
点赞，这是可行方案	
张玮(大圣)	

更多一手资源请添加QQ/微信1182316662	2018-05-22
1.用现成的中间件业务方接入redis能懂吗? 如果用现成的消费端不会用到类似推模式的中间件? 2. 缓存热点问题, 其实在 talr 里做的方案就很棒, 首先本地会维护一个热点区域用于存储集群通知的热点数据	
作者回复	2018-05-23
可以看kafka的设计文档, 有详细描述	
万历十五年	2018-05-22
请教可否列举一下: 1.“已经被各种场景验证过的成熟架构技术”有哪些? 2. 以及类似mysql单表5000万这样的常用极限数据 3.有助于学习和理解架构设计的书籍? 或是最佳实践的推荐?	
作者回复	2018-05-22
后面会讲架构模式	
bluefantasy	2018-05-22
老师好, 个人觉得像新浪微博这种场景使用Kafka,Rabbitmq会比Netty+Mysql自研好很多. 主要原因是这个场景对实时性要求还是比较高的(一般采取消息队列主动推送模式). 开源的消息队列都有对消费者的推送模式. 自研的话, 如果采用消息推送模式, 消息队列服务端需要在服务端记录所有消费者的状态信息, 还要考虑各种异常和消息确认, 实现起来应该是很复杂. 一般公司根本没有这个技术实力. 个人见解, 希望得到老师的回复.	
作者回复	2018-05-22
kafka的文档中有消息队列的详细设计说明, kafka消费时也是采取pull模式而不是push模式	
付之一笑	2018-06-12
为什么不用nginx实现主备	
作者回复	2018-06-13
nginx也可以, 写个lua脚本做逻辑处理, 但这不是常规做法	
孙振超	2018-06-09
单从消息中间件选型上看, 就是开源. 自研以及对开源进行改造三种方案. 后两种的整体成本会高很多, 看明白理解透彻并提出新的解决方案都是很花功夫的事情. 不管是消息中间件还是其他的系统, 本质上都是两个部分: 逻辑运算和数据的增删改查. 为了提升容量和高可用, 逻辑运算部分会采用集群的方式进行部署; 数据部分为保证容量和数据不丢失, 会采用集群和主备的方式进行部署. 数据部分可以选择的方式比较多样: nosql、关系型数据库、文件存储等. 第四种方案可以是对开源的中间件进行改造以更适合自己的场景.	
JOEL	2018-05-22
从需求看消息需要可靠性, 即不能丢消息. 那么kafka没有提供可靠性的保证, 但是rabbitmq有这个保证, 所以选择开源的诶这个场景还是rmq好	
东	2018-05-22
AWS SQS, 如果消息size 大于256k, SQS +S3	
作者回复	2018-05-22
学习了, 不过这个是要上云吧	
Snway	2018-05-22
storm+redis可行? 利用storm流式计算与分布式特性, redis存储特性	
作者回复	2018-05-22
storm用作消息队列, 感觉有点杀鸡用牛刀呢, 虽然功能上是可以, 而且消息读取流程感觉比较难做	
在路上	2018-05-22
选择kafka为啥不选rabbitmq , 他俩主要区别是啥?	
作者回复	2018-05-22
是可以rabbitmq, activemq等方案, 具体的区别等你的答案💎💎	
narry	2018-05-22
个人觉得方案2,3根据架构设计的3个原则已经是一个优秀的架构设计了, 我只是感觉到微博的应用有个比较有特点的特征就是有热点的出现, 如果热点出现, 按方案2,3会出现消息都集中在一个分片中, 从而导致分片所在主机处于饱和的状态, 甚至崩溃掉, 如果能像业务服务器中放一个sidecar(或者内嵌到程序中), 来负责根据规则的消息路由, 并且规则是可以调整的, 这样就可以应对热点问题	
作者回复	2018-05-22
微博有热点, 异步消息没有热点哦, 消息随便发给任何一台服务器都可以, 可以采取轮询或者随机算法	
blackncccc	2018-07-18
方案二消息被分散到各个分组, 在读取消息的时候如何保证整体的顺序呢	
作者回复	2018-07-18
没法保证💎💎	
blackncccc	2018-07-17
不太明白方案二中的数据分散存储, 客户端轮询写入读取. 怎么快速定位数据被写到了哪个分组呢	
作者回复	2018-07-17
不需要定位数据被写入哪个分组了, 客户端只管写入系统就可以了	
孙振超	2018-06-09

本章最大的收获在于三个方案原则，给领导做选择而不是必选题。这句话早有耳闻，但真正运用的不多。三个方案原则最大的好处是让自己更为深入的思考，对问题的了解更为深刻，不会犯太低级的错误。真是知易行难。	
作者回复	2018-06-10
三个选择是决策领域里面防止思维狭隘的一种非常重要和有效的方法	
Geek_59a17f	
例如，高可用的主备方案、集群方案，高性能的负载均衡、多路复用，可扩展的分层、插件化等技术，绝大部分时候我们有了明确的目标后，按图索骥就能够找到可选的解决方案。	
gary	2018-06-07
李老师，我们最近帮银行做系统，发现现在银行业仍然有很多公司使用weblogic / websphere 这样的容器，就目前形势来看weblogic较一些开源方案好像确实更加稳定和安全。目前来看老师现在给的一些技术思路都是偏互联网方向，不知道老师是否有过一些厂商产品方面经验，比如weblogic 的mq性能水平是什么样的，和开源技术比较有哪些优劣势，好像网上这方面资料也不多，所以想请教李老师。另外不知道李老师眼中如何看待这些类似oracle 大厂商产品和未来我们的互联网开源产品在市场上的发展趋势	
作者回复	2018-06-08
我的经验主要在互联网，总体技术趋势是云计算，不管是开源还是闭源。	
Oracle这类企业，要么自己开始提供云计算云存储，要么就是为云厂商生产设备。	
XB.Chan	
备选方案2中，采用集群+MySQL的方式开发，满足QPS13800的需求，但是MySQL按照性能来说800/秒的插入速度算是比较快了，那这样平均算下来13800/800约等于18台服务器？	
作者回复	2018-06-05
MySQL的读写简单表结构（例如k-v）远远超过800的QPS和TPS，我们测试k-v都是上万的	
SHLOMA	
李老师专栏讲解模式很好，菜鸟都能跟着进来，看到此处感觉需要学习好多技术，望老师有机会可以出个架构相关技术学习的专栏	
作者回复	2018-06-01
太多了，还是期望你掌握了整体架构设计理论，然后按照这套理论的指导去自己学习	
arebya	
我们会在设计最终方案的时候穿插考虑其他方案做对比，按照3到5个备选方案的设计，不会长期下来流于形式，而且这对团队来说也是成本。	
作者回复	2018-06-01
流程的作用就是保证换谁来做都不会走偏，要知道能设计出三个备选方案，不是形式上应付了事就能够完成的	
小水	
等华仔出书	
eye	
上esb吧	
天悦	
不要过度设计，好难💎💎	
我的名字叫浩仔、	
如果planA不行我们就用planB	
作者回复	2018-05-25
不是的，每个plan都是可行的，要挑选更合适的plan	
Tom	
请教下，备选方案2里，一个数据分组有两个主？	
作者回复	2018-05-24
一个主服务器，一个主数据库	
稳健的少年	
直接用Kafka或者RocketMQ不好么，为什么需要后面两个那么复杂的解决方案？	
作者回复	2018-05-24
后面的“选择和评估备选方案”有讲	
家瑜	
写文件，异步消费。	
呵呵	
备选方案2里，一个数据分组有两个主，是什么意思？	
作者回复	2018-05-24
一个是主服务器，一个是主数据库	
合民	
说来惭愧，对于这方面的技术，应用非常少，所以学习了文章中的内容，和大牛门的留言。我应用过ELK写入日志和即时检索，很多Kafka+es架构可以成为备选方案，但确实只做到一定程度考虑缓存，因为微博是有热点和实效性的，es内部也有缓存机制。	

更多一手资源请添加QQ/微信1182316662

作者回复	2018-05-25
kafka已经是完备的消息队列了，加上es的目的是什么？	
YMF_WX1981	
要完成的架构是个万能黑洞，做到技术无关性，用总线bus可否实现，进而对外提供满足业务的计算和存储功能。	2018-05-23
王旭东	
最近在学RocketMQ，可以使用RocketMQ（根据请求量决定集群方式，多Master模式还是多Master多Slave模式），配合消息确认和重发机制，保证消息不丢失 高可用消息存储和读取可以用Redis集群+本地数据持久化	2018-05-23
作者回复	2018-05-23
有redis集群就可以不用本地持久化了，不然复杂度比较高	
海洋	
我用Kafka多些，有时候会集群崩溃，那部分消息一般就丢失了	2018-05-23
作者回复	2018-05-23
是的，我们遇到过，如果是日志，丢失问题不大，但业务消息，丢失后修复还是比较蛋疼	
王磊	
原文这句话理解有点困难 高性能消息读取属于“计算高可用”的范畴'	2018-05-23
作者回复	2018-05-23
写错了，属于“计算高性能”范畴	
王磊	
方案二的架构图有些question（不是problem），这个可以理解为分库后的主备方案吗，还是所有数据库是全量，我理解是前者，因为文中说分组间不同步，那么这里缺少一个根据消息路由的模块，对吧	2018-05-23
作者回复	2018-05-23
赞，你可以思考一下怎么做💎💎	
追寻云的痕迹	
实话实说，正在做容器项目的交付和实施，并不想把Docker和虚拟化这两词并列放在一起，因为虚拟化最多是容器技术的一个细节技术点。我们应当看到容器技术引导了应用标准化交付和运维的变革。容器镜像将是运维开发工具链中的标准件。	2018-05-23
刘志刚	
已经学习到这一个章节，我也来留言一下吧，我这边给出的方案是，消息队列这块选型rabbitmq或者rocketmq，存储使用hbase和redis结合，理由如下：消息中间件对于及时性和高性能上这两个还算是比较成熟和稳定的，存储和读取上hbase的单写入性能较高，根据特定key的读取性能也较高，另外一部分，按照用户纬度，或者话题纬度可以缓存到redis	2018-05-23
作者回复	2018-05-23
rocketmq自带存储了	
Classtag	
我觉得前面问的热点问题是说 存储上的热点问题吧。比如数据存储时分片不合理，导致一个分片上有较多的热点微博相关的数据 比如评论 这可就毁了	2018-05-23
作者回复	2018-05-23
消息存储的时候采用随机或者轮询就可以，不要按照业务来分	
eye	
消息高可用持久化传递，作者用了3种方案传递消息第一种是mq类，第二种是netty加数据库，第三种hbase或自研。有同学说了rpc加mq。可以用响应式分布式处理架构akka。	2018-05-22
作者回复	2018-05-23
可以，不过akka的Actor模型比较复杂，适应于异步编程的场景，感觉和消息队列的业务不是很匹配，如果不是很熟练，不建议用	
时间去哪里了？	2018-05-22
也可以考虑用 elasticsearch和spark	
作者回复	2018-05-22
用es怎么做，好像有点困难吧？es是用于全文搜索的	