39 | 互联网技术演进的模式

2018-07-26 李运华 来自北京

《从0开始学架构》

由于各行业的业务发展轨迹并不完全相同，无法给出一个统一的模板让所有的架构师拿来就套用，因此我以互联网的业务发展为案例，谈谈互联网技术演进的模式，其他行业可以参考分析方法对自己的行业进行分析。

互联网业务千差万别，但由于它们具有“规模决定一切”的相同点，其发展路径也基本上是一致的。互联网业务发展一般分为几个时期：初创期、发展期、竞争期、成熟期。

不同时期的差别主要体现在两个方面：**复杂性、用户规模**。

业务复杂性

互联网业务发展第一个主要方向就是“业务越来越复杂”，我们来看看不同时期业务的复杂性的表现。

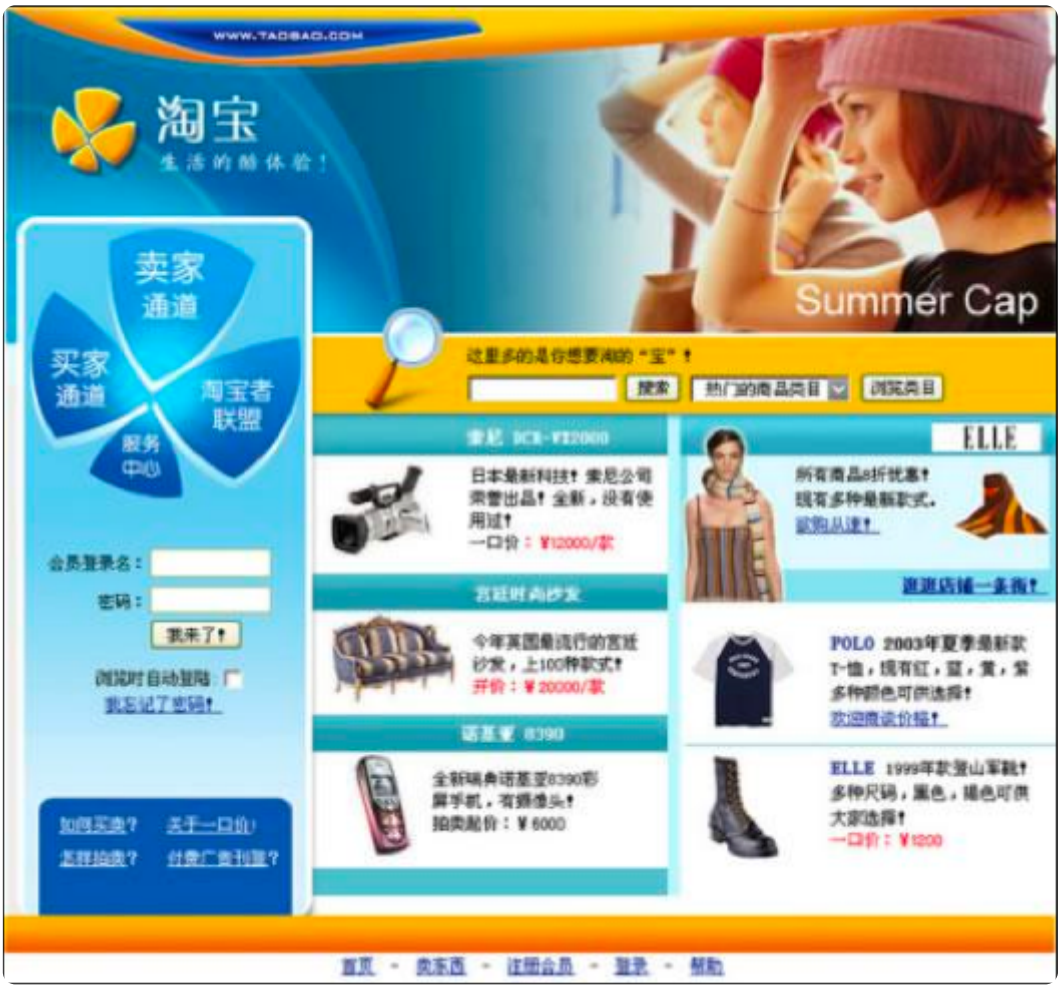
1. 初创期

互联网业务刚开始一般都是一个创新的业务点，这个业务点的重点不在于“完善”，而在于“创新”，只有创新才能吸引用户；而且因为其“新”的特点，其实一开始是不可能很完善的。只有随着越来越多的用户的使用，通过快速迭代试错、用户的反馈等手段，不断地在实践中去完善，才能继续创新。

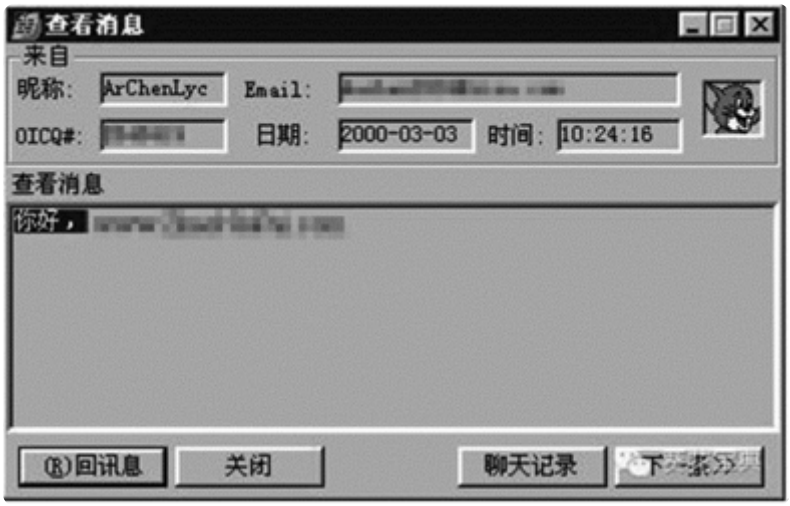
初创期的业务对技术就一个要求：“快”，但这个时候却又是创业团队最弱小的时期，可能就几个技术人员，所以这个时候十八般武艺都需要用上：能买就买，有开源的就用开源的。

我还以淘宝和 QQ 为例。

第一版的淘宝（https://blog.csdn.net/linlin_juejue/article/details/5959171）：



第一版的 QQ（http://www.yixieshi.com/20770.html）：



可以看到最开始的淘宝和 QQ 与现在相比，几乎看不出是同一个业务了。

2. 发展期

当业务推出后经过市场验证如果是可行的，则吸引的用户就会越来越多，此时原来不完善的业务就进入了一个快速发展的时期。业务快速发展时期的主要目的是将原来不完善的业务逐渐完善，因此会有越来越多的新功能不断地加入到系统中。对于绝大部分技术团队来说，这个阶段技术的核心工作是快速地实现各种需求，只有这样才能满足业务发展的需要。

如何做到“快”，一般会经历下面几个阶段。

堆功能期

业务进入快速发展期的初期，此时团队规模也不大，业务需求又很紧，最快实现业务需求的方式是继续在原有的系统里面不断地增加新的功能，重构、优化、架构等方面的工作即使想做，也会受制于人力和业务发展的压力而放在一边。

优化期

“堆功能”的方式在刚开始的时候好用，因为系统还比较简单，但随着功能越来越多，系统开始变得越来越复杂，后面继续堆功能会感到越来越吃力，速度越来越慢。一种典型的场景是做一个需求要改好多地方，一不小心就改出了问题。直到有一天，技术团队或者产品人员再也受不了这种慢速的方式，终于下定决定要解决这个问题了。

如何解决这个问题，一般会分为两派：一派是优化派，一派是架构派。

优化派的核心思想是将现有的系统优化。例如，采用重构、分层、优化某个 MySQL 查询语句，将机械硬盘换成 SSD，将数据库从 MySQL 换成 Oracle，增加 Memcache 缓存等。优化派的优势是对系统改动较小，优化可以比较快速地实施；缺点就是可能过不了多久，系统又撑不住了。

架构派的核心思想是调整系统架构，主要是将原来的大系统拆分为多个互相配合的小系统。例如，将购物系统拆分为登录认证子系统、订单系统、查询系统、分析系统等。架构派的优势是一次调整可以支撑比较长期的业务发展，缺点是动作较大、耗时较长，对业务的发展影响也比较大。

相信在很多公司都遇到这种情况，大部分情况下都是“优化派”会赢，主要的原因还是因为此时“优化”是最快的方式。至于说“优化派”支撑不了多久这个问题，其实也不用考虑太多，因为业务能否发展到那个阶段还是个未知数，保证当下的竞争力是最主要的问题。

架构期

经过优化期后，如果业务能够继续发展，慢慢就会发现优化也顶不住了，毕竟再怎么优化，系统的能力总是有极限的。Oracle 再强大，也不可能一台 Oracle 顶住 1 亿的交易量；小型机再好，也不可能一台机器支持 100 万在线人数。此时已经没有别的选择，只能进行架构调整，在优化期被压制的架构派开始扬眉吐气了，甚至会骄傲地说“看看吧，早就说要进行架构调整，你们偏要优化，现在还是顶不住了吧，哼……”。

架构期可以用的手段很多，但归根结底可以总结为一个字“拆”，什么地方都可以拆。

拆功能：例如，将购物系统拆分为登录认证子系统、订单系统、查询系统、分析系统等。

拆数据库：MySQL 一台变两台，2 台变 4 台，增加 DBProxy、分库分表等。

拆服务器：服务器一台变两台，2 台变 4 台，增加负载均衡的系统，如 Nginx、HAProxy 等。

3. 竞争期

当业务继续发展，已经形成一定规模后，一定会有竞争对手开始加入行业来竞争，毕竟谁都想分一块蛋糕，甚至有可能一不小心还会成为下一个 BAT。当竞争对手加入后，大家互相学习和模仿，业务更加完善，也不断有新的业务创新出来，而且由于竞争的压力，对技术的要求是更上一层楼了。

新业务的创新给技术带来的典型压力就是新的系统会更多，同时，原有的系统也会拆得越来越多。两者合力的一个典型后果就是系统数量在原来的基础上又增加了很多。架构拆分后带来的美好时光又开始慢慢消逝，技术工作又开始进入了“慢”的状态，这又是怎么回事呢？

原来系统数量越来越多，到了一个临界点后就产生了质变，即系统数量的量变带来了技术工作的质变。主要体现在下面几个方面：

重复造轮子

系统越来越多，各系统相似的工作越来越多。例如，每个系统都有存储，都要用缓存，都要用数据库。新建一个系统，这些工作又要都做一遍，即使其他系统已经做过了一遍，这样怎么能快得起来？

系统交互一团乱麻

系统越来越多，各系统的交互关系变成了网状。系统间的交互数量和系统的数量成平方比的关系。例如，4 个系统的交互路径是 6 个，10 个系统的交互路径是 45 个。每实现一个业务需求，都需要几个甚至十几个系统一起改，然后互相调用来调用去，联调成了研发人员的灾难、联测成了测试人员的灾难、部署成了运维的灾难。

针对这个时期业务变化带来的问题，技术工作主要的解决手段有：

平台化

目的在于解决“重复造轮子”的问题。

存储平台化：淘宝的 TFS、京东 JFS。

数据库平台化：百度的 DBProxy、淘宝 TDDL。

缓存平台化：Twitter 的 Twemproxy，豆瓣的 BeansDB、腾讯 TTC。

服务化

目的在于解决“系统交互”的问题，常见的做法是通过消息队列来完成系统间的异步通知，通过服务框架来完成系统间的同步调用。

消息队列：淘宝的 Notify、MetaQ，开源的 Kafka、ActiveMQ 等。

服务框架：Facebook 的 thrift、当当网的 Dubbox、淘宝的 HSF 等。

4. 成熟期

当企业熬过竞争期，成为了行业的领头羊，或者整个行业整体上已经处于比较成熟的阶段，市场地位已经比较牢固后，业务创新的机会已经不大，竞争压力也没有那么激烈，此时求快求新已经没有很大空间，业务上开始转向为“求精”：我们的响应时间是否比竞争对手快？我们的用户体验是否比竞争对手好？我们的成本是否比竞争对手低……

此时技术上其实也基本进入了成熟期，该拆的也拆了，该平台化的也平台化了，技术上能做的大动作其实也不多了，更多的是进行优化。但有时候也会为了满足某个优化，系统做很大的改变。例如，为了将用户响应时间从 200ms 降低到 50ms，可能就需要从很多方面进行优化：CDN、数据库、网络等。这个时候的技术优化没有固定的套路，只能按照竞争的要求，找出自己的弱项，然后逐项优化。在逐项优化时，可以采取之前各个时期采用的手段。

用户规模

互联网业务的发展第二个主要方向就是“用户量越来越大”。互联网业务的发展会经历“初创期、发展期、竞争期、成熟期”几个阶段，不同阶段典型的差别就是用户量的差别，用户量随着业务的发展而越来越大。

用户量增大对技术的影响主要体现在两个方面：性能要求越来越高、可用性要求越来越高。

1. 性能

用户量增大给技术带来的第一个挑战就是性能要求越来越高。以互联网企业最常用的 MySQL 为例，再简单的查询，再高的硬件配置，单台 MySQL 机器支撑的 TPS 和 QPS 最高也就是万级，低的可能是几千，高的也不过几万。当用户量增长后，必然要考虑使用多台 MySQL，从一台 MySQL 到多台 MySQL 不是简单的数量的增加，而是本质上的改变，即原来集中式的存储变为了分布式的存储。

稍微有经验的工程师都会知道，分布式将会带来复杂度的大幅度上升。以 MySQL 为例，分布式 MySQL 要考虑分库分表、读写分离、复制、同步等很多问题。

2. 可用性

用户量增大对技术带来的第二个挑战就是可用性要求越来越高。当你有 1 万个用户的时候，宕机 1 小时可能也没有很大的影响；但当你有了 100 万用户的时候，宕机 10 分钟，投诉电话估计就被打爆了，这些用户再到朋友圈抱怨一下你的系统有多烂，很可能你就不会再有机会发展下一个 100 万用户了。

除了口碑的影响，可用性对收入的影响也会随着用户量增大而增大。1 万用户宕机 1 小时，你可能才损失了几千元；100 万用户宕机 10 分钟，损失可能就是几十万元了。

量变到质变

通过前面的分析，我们可以看到互联网业务驱动技术发展的两大主要因素是复杂性和用户规模，而这两个因素的本质其实都是“量变带来质变”。

究竟用户规模发展到什么阶段才会由量变带来质变，虽然不同的业务有所差别，但基本上可以按照下面这个模型去衡量。

阶段	用户规模	业务阶段	技术影响
婴儿期	0～1万	初创期	用户规模对性能和可用性都没有什么压力，技术人员可以安心睡好觉。
幼儿期	1万～10万	初创期	用户规模对性能和可用性已经有一点压力了，主要体现为单台机器（服务器、数据库）可能已经撑不住了，需要开始考虑拆分机器，但这个时候拆分还比较简单，因为机器数量不会太多。
少年期	10万～100万	发展期	用户规模对性能和可用性已经有较大压力了，除了拆分机器，已经开始需要将原来大一统的业务拆分为更多子业务了。
青年期	100万～1000万	竞争期	用户规模对性能和可用性已经有很大压力了，集群、多机房等手段开始用上了。虽然如此，技术人员还是很高兴的，毕竟到了此时公司已经发展得非常不错了！
壮年期	1000万～1亿	竞争期 & 成熟期	用户规模对性能和可用性已经有非常大的压力了，可能原来的架构和方案已经难以继续扩展下去，需要推倒重来。不过如果你真的身处这样一家公司，虽然可能有点辛苦，但肯定会充满干劲，因为这样的机会非常难得，也非常锻炼人。
巨人期	1亿+	成熟期	和壮年期类似，不过如果你真的身处这样一家公司，虽然可能有点辛苦，但估计做梦都要笑醒了！因为还没有哪个行业能够同时容纳两家1亿+用户的公司。

应对业务质变带来的技术压力，不同时期有不同的处理方式，但不管什么样的方式，其核心目标都是为了满足业务“快”的要求，当发现你的业务快不起来的时候，其实就是技术的水平已经跟不上业务发展的需要了，技术变革和发展的时候就到了。更好的做法是在问题还没有真正暴露出来就能够根据趋势预测下一个转折点，提前做好技术上的准备，这对技术人员的要求是非常高的。

小结

今天我为你讲了互联网技术演进的基本模式，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，参考今天文章的方法，简单分析一下你所在行业，看看是否存在典型的技术演进模式？

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！）

精选留言 (31)

孙振超
2018-09-25

在上一家公司经历了从发展期到竞争期的转变，起初业务增加比较快，各种功能不断向上堆。在后期开始慢慢搭建自己的文件存储系统、数据库中间件、消息中间件，当第一任架构师离职时cto的评价就是各种重构。
现在的公司应该算成熟期，虽然也面对着巨大的竞争压力，内部总结经历过几次大的架构阶段：第一代的大一统架构、第二代烟囱式架构、第三代分布式微服务架构、第四代的多地多中心架构以及现在正在进行的第五代架构升级。参与了多地多中心架构升级，也和腾讯的同学聊过，因而对异地多活有了一些定性认识，对于正在进行的第五代架构，还处于摸索阶段。

作者回复: 你的经历非常有价值，好好总结一下，你也可以开专栏了😄

共 3 条评论 >

46

yoummg
2018-07-29

壮年期的公司，对于一个初级的工程师，应该在这样的公司做好什么事？

作者回复: 在一个领域做精，成为专家

23

大兵
2018-08-06

不知道拼多多的架构演进是怎么样，在短短3年内发展到现在的规模？有在拼多多的同学吗？分享下

作者回复: 应该很快就会在各种技术大会看到他们分享了

共 2 条评论 >

17

hello

2018-07-26

请教李老师，现在微服务架构已经很成熟了，特别是spring cloud 提供了各种基础服务，初创企业一开始就上微服务好像成本也不大，还需要经历从单体架构拆分的过程吗?

作者回复: 可以用spring cloud ，但谨记我在微服务章节提到的三个火枪手原则，不要拆的太细

共 2 条评论 >

12

Bright.亮

2018-07-29

初创型公司，用户还不够一万，已经是分布式了，这样是不是有点儿浪费?

作者回复: 看分布式的规模，如果只是简单拆几个服务是可以的，如果拆分成几十个微服务那就浪费了

10

大光头

2018-07-26

现在公司就处于竞争期，大家重复造轮子以及整合轮子的过程

作者回复: 赶紧成立平台技术部，因为到了这个阶段，说明你们业务已经发展不错了，有资源投入

共 2 条评论 >

9

Kian.Lee

2020-06-29

公司应该属于“孕育期”，去年10月成立，业务类型 SaaS 服务产品，产品已成型，属于激进务实派，全面上云，后端 Spring boot + kotlin 部署 k8s + 公有云普惠型 DevOps，发布流水线，构建、测试、部署自动化，前端 Web VUE、SPA 无服务器部署（OSS），发布流水线，构建、测试、部署自动化，APP Weex+VUE 前端统一技术栈，监控 ARMS、流控 AHAS （代码无侵入），现在最幸福的事情，代码Push ，5分钟就能收到流水线部署成功通知。架构设计应该在“简单”、“适用”的原则上考虑适当的技术前瞻性!

作者回复: WEEX还在维护么? 😂

共 2 条评论 >

7

-

Amos
2018-09-20

看到华仔每条都有回复，真的很用心，虽然专栏已经结束了，依然向你学习。

作者回复: 加油👍

6

小胖狗
2018-07-27

我大概经历了 1万～10万，10万到40万这个阶段。😂😂😂

5

日光倾城
2019-04-28

在一些小的互联网公司跳来跳去，当时都没想过公司处在什么阶段

作者回复: 公司不同阶段对人要求不一样，机会也不一样

3

恒初
2021-03-17

作者能说说项目化运作的产品复杂度如果应对吗

作者回复: 你是说类似2B类的产品么?

如果是外包类的一锤子买卖，基本不需要考虑演进，核心复杂度是如何用尽量少的人力和尽量低的成本来实现客户项目，这样利润才多。

如果是电信设备这类产品，后期的维护合同和升级合同比卖设备的钱还多，不过这种产品演进绝大部分是技术升级，而不是业务量和用户量增加。

3

丹出江湖
2021-10-29

目前业务场景还是单台服务器，领导却想能够嵌入式部署，单机部署，集群部署，功能灵活部

署。团队人员技术和人力资源都有限，真是很头疼。

作者回复: 试试把架构专栏的内容用来说服领导

1

Andy
2021-01-23

互联网系统以“规模决定一切”，如同娱乐圈“流量决定一切”。互联网系统的演化，归纳总结下来，如华哥所言，就这两个方面，业务复杂度和用户规模。架构师的思维方式，就是基于这两个方面，从0到1，再从1到N，量变不断引起质变，如何拥有很好的应对措施，怎么去解决这些问题，并且为下一个阶段做好预防措施和解决方案。那么，架构师就是要每一步都拥有应对方案，这样思考是不是意味着架构思维就开始上道了呢，求问华哥，哈哈

作者回复: 是的，意识到架构和业务的关系，并且能够从业务的角度来设计架构，就已经超越2/3的人了：）

1

flyCoder
2020-10-23

目前公司是做ToB 的，产品还处于摸索阶段，虽然人员比较多，整体上看还是婴儿期，希望公司越做越好，有一个行业标杆的突破性产品，体验一把从婴儿期到壮年期的过程。

作者回复: 有这种机会你就发达了，加油

1

文古
2020-07-24

公司初创期还没有过，就进入 了流产期。公司内部系统多，杂，乱。

作者回复: 这……也可以说在探索

共 3 条评论>

1

谭方敏
2020-03-13

业务复杂性

1初创期，业务重点不在于完善，而在于创新。一开始不可能很完善的，在越来越多用户使用过程中，快速迭代试错，不断地在实践中去完善。

2 发展期，业务推出后经市场验证后如果是可行的，吸引的用户会越来越多，此时不完善的业务就会到了快速发展的时期。
快速发展期一般会有几个阶段：
堆功能期，在原有系统中不断增加新的功能，重构，优化，架构等方面的工作。

优化期，前期堆功能越来越多，系统也会变得越来越复杂，继续堆功能也会越来越吃力，面对这个状况就遇到两派：优化派（优化现有系统），架构派（调整系统架构，拆分）。

架构期，继续拆。

3 竞争期 ，有竞争对手加入，大家相互学习和模仿，原有系统被拆得越来越多，主要体现在
a) 重复造轮子，浪费严重，解决办法是平台化。
b) 系统交互一团麻，系统交互数量跟系统数量正相关，解决方法是服务化，通过消息队列来完成系统间的异步通知，通过服务框架来完成系统间的同步调用。

4成熟期，在业务上开始求精，这段时间技术优化并没有什么套路，根据竞争的需求，找出自己的弱项，然后逐步优化。

用户规模
随着用户规模的增加，带来的两个挑战是性能
和可用性。

物联网行业演进方式跟互联网差不多，大致是按照华仔的这样思路演进的，不过我所在公司还处于幼儿期。

作者回复: 说明发展空间很大😃

1

GeekCoder

2019-07-03

用户规模指的是平台所有用户还是日活?

作者回复: 一般是总用户

1

KingPoker

2018-08-01

这个和开篇的内容类似吧

作者回复: 这篇是详细分析，开篇只是说业务促进技术演进

1

月光宫羽

2022-11-23 来自广东

忽然感觉入职到现在，共经历了3个业务阶段，如下：

1、初创期：单体架构，版本管理是Svn，运维上线效率低下，无流程无规范，CTO还要通宵写代码。一句话总结：混乱和低效。

2、发展期：一个字拆，说是微服务架构，其实是面向服务（SOA）的架构，服务试单纯的模块解耦，数据层面耦合严重；运维上开始引入

git及K8s，效率上有明显提升。测试上一直没有进步，没有自动化测试，简直是没有牙齿的老虎。一句话总结：有点儿模样。

3、竞争期：目前业务规模和交易量已是行业第2，在这个阶段开始建设2地三中心机房，搞机房级别的大入口改造及熔断、演练与复盘变得常态化。也引入Scrum，DevOps理念和方法论，对于需求、项目管理、CICD这些烟囱式的孤岛平台，强行加入CI平台将其从数据层面打通，核心方法使用了

价值流驱动研发流的流模型。开始注重安全，一方面将DevSecOps工具集成到流水线，另一方面使用专业的安全产品定期巡检。一句话总结：虽然没有美丽，但也含苞待放。

未来：set部署，核心链路保护，数据编织，其它质效稳这3方面等等，有很长的路要走，所以急需充电。

作者回复: 很不错的经历，可以自己好好总结提炼一下，毕竟这样的机会不是人人能够遇到


shark

2021-11-17

业务处于壮年期，技术处于青年期，现在就是升级技术，建立壁垒

作者回复: 好好干，大把机会等着你发挥：）