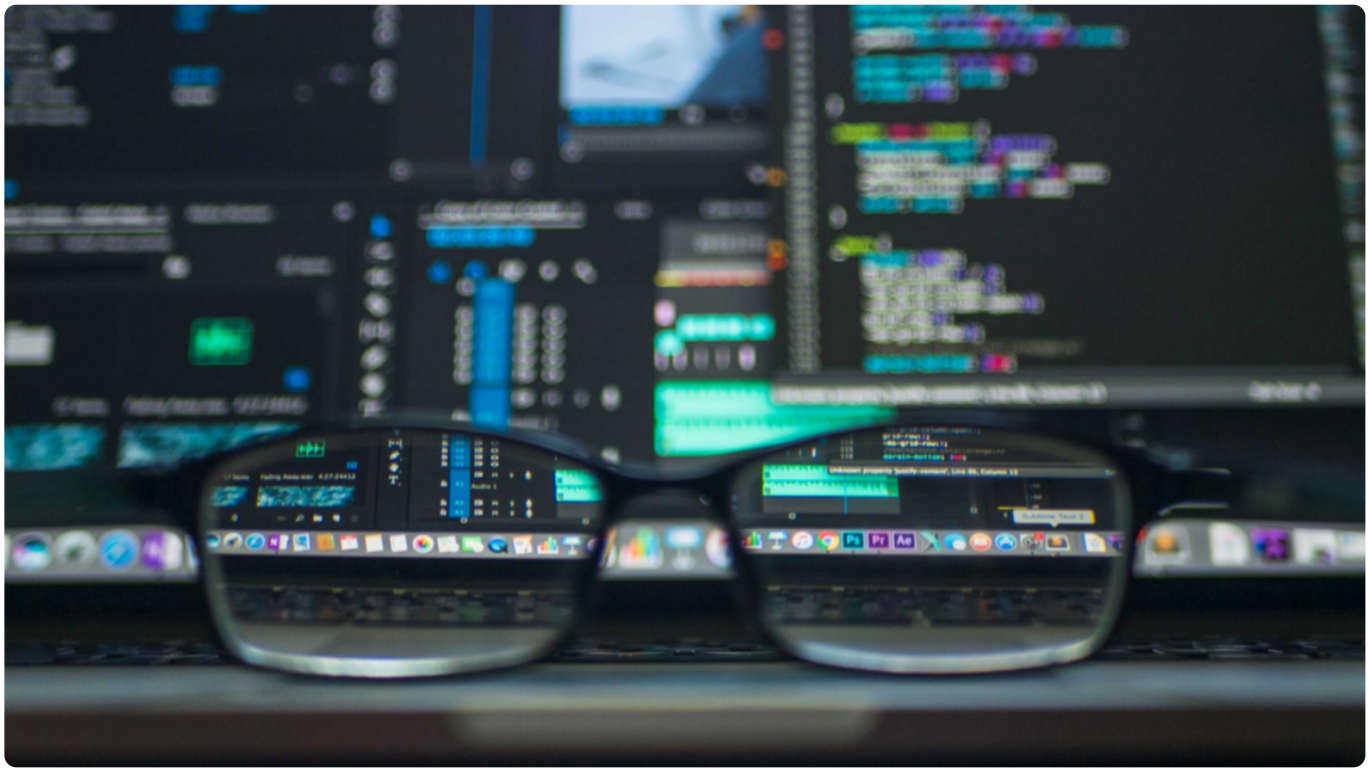
10 | 架构设计流程：识别复杂度

2018-05-19 李运华

从0开始学架构 进入课程 >



讲述：黄洲君
时长 10:48　大小 4.95M

从今天开始，我将分 4 期，结合复杂度来源和架构设计原则，通过一个模拟的设计场景“前浪微博”，和你一起看看在实践中究竟如何进行架构设计。今天先来看架构设计流程第 1 步：识别复杂度。

架构设计第 1 步：识别复杂度

我在前面讲过，架构设计的本质目的是为了解决软件系统的复杂性，所以在我们设计架构时，首先就要分析系统的复杂性。只有正确分析出了系统的复杂性，后续的架构设计方案才不会偏离方向；否则，如果对系统的复杂性判断错误，即使后续的架构设计方案再完美再先进，都是南辕北辙，做的越好，错的越多、越离谱。

例如，如果一个系统的复杂度本来是业务逻辑太复杂，功能耦合严重，架构师却设计了一个 TPS 达到 50000/ 秒的高性能架构，即使这个架构最终的性能再优秀也没有任何意义，因为架构没有解决正确的复杂性问题。

架构的复杂度主要来源于“高性能”“高可用”“可扩展”等几个方面，但架构师在具体判断复杂性的时候，不能生搬硬套，认为任何时候架构都必须同时满足这三方面的要求。实际上大部分场景下，复杂度只是其中的某一个，少数情况下包含其中两个，如果真的出现同时需要解决三个或者三个以上的复杂度，要么说明这个系统之前设计的有问题，要么可能就是架构师的判断出现了失误，即使真的认为要同时满足这三方面的要求，也必须要进行优先级排序。

例如，专栏前面提到过的“亿级用户平台”失败的案例，设计对标腾讯的 QQ，按照腾讯 QQ 的用户量级和功能复杂度进行设计，高性能、高可用、可扩展、安全等技术一应俱全，一开始就设计出了 40 多个子系统，然后投入大量人力开发了将近 1 年时间才跌跌撞撞地正式上线。上线后发现之前的过度设计完全是多此一举，而且带来很多问题：

- 系统复杂无比，运维效率低下，每次业务版本升级都需要十几个子系统同步升级，操作步骤复杂，容易出错，出错后回滚还可能带来二次问题。
- 每次版本开发和升级都需要十几个子系统配合，开发效率低下。
- 子系统数量太多，关系复杂，小问题不断，而且出问题后定位困难。
- 开始设计的号称 TPS 50000/ 秒的系统，实际 TPS 连 500 都不到。

由于业务没有发展，最初的设计人员陆续离开，后来接手的团队，无奈又花了 2 年时间将系统重构，合并很多子系统，将原来 40 多个子系统合并成不到 20 个子系统，整个系统才逐步稳定下来。

如果运气真的不好，接手了一个每个复杂度都存在问题的系统，那应该怎么办呢？答案是一个个来解决问题，不要幻想一次架构重构解决所有问题。例如这个“亿级用户平台”的案例，后来接手的团队其实面临几个主要的问题：系统稳定性不高，经常出各种莫名的小问题；系统子系统数量太多，系统关系复杂，开发效率低；不支持异地多活，机房级别的故障会导致业务整体不可用。如果同时要解决这些问题，就可能会面临这些困境：

- 要做的事情太多，反而感觉无从下手。
- 设计方案本身太复杂，落地时间遥遥无期。

同一个方案要解决不同的复杂性，有的设计点是互相矛盾的。例如，要提升系统可用性，就需要将数据及时存储到硬盘上，而硬盘刷盘反过来又会影响系统性能。

因此，正确的做法是**将主要的复杂度问题列出来，然后根据业务、技术、团队等综合情况进行排序，优先解决当前面临的最主要的复杂度问题。**“亿级用户平台”这个案例，团队就优先选择将子系统的数量降下来，后来发现子系统数量降下来后，不但开发效率提升了，原来经常发生的小问题也基本消失了，于是团队再在这个基础上做了异地多活方案，也取得了非常好的效果。

对于按照复杂度优先级解决的方式，存在一个普遍的担忧：如果按照优先级来解决复杂度，可能会出现解决了优先级排在前面的复杂度后，解决后续复杂度的方案需要将已经落地的方案推倒重来。这个担忧理论上是可能的，但现实中几乎是不可能出现的，原因在于软件系统的可塑性和易变性。对于同一个复杂度问题，软件系统的方案可以有多个，总是可以挑出综合来看性价比最高的方案。

即使架构师决定要推倒重来，这个新的方案也必须能够同时解决已经被解决的复杂度问题，一般来说能够达到这种理想状态的方案基本都是依靠新技术的引入。例如，Hadoop 能够将高可用、高性能、大容量三个大数据处理的复杂度问题同时解决。

识别复杂度对架构师来说是一项挑战，因为原始的需求中并没有哪个地方会明确地说明复杂度在哪里，需要架构师在理解需求的基础上进行分析。有经验的架构师可能一看需求就知道复杂度大概在哪里；如果经验不足，那只能采取“排查法”，从不同的角度逐一进行分析。

识别复杂度实战

我们假想一个创业公司，名称叫作“前浪微博”。前浪微博的业务发展很快，系统也越来越多，系统间协作的效率很低，例如：

用户发一条微博后，微博子系统需要通知审核子系统进行审核，然后通知统计子系统进行统计，再通知广告子系统进行广告预测，接着通知消息子系统进行消息推送……一条微博有十几个通知，目前都是系统间通过接口调用的。每通知一个新系统，微博子系统就要设计接口、进行测试，效率很低，问题定位很麻烦，经常和其他子系统的技术人员产生分岐，微博子系统的开发人员不胜其烦。

用户等级达到 VIP 后，等级子系统要通知福利子系统进行奖品发放，要通知客服子系统安排专属服务人员，要通知商品子系统进行商品打折处理……等级子系统的开发人员也是

不胜其烦。

新来的架构师在梳理这些问题时，结合自己的经验，敏锐地发现了这些问题背后的根源在于架构上各业务子系统强耦合，而消息队列系统正好可以完成子系统的解耦，于是提议要引入消息队列系统。经过一分析二讨论三开会四汇报五审批等一系列操作后，消息队列系统终于立项了。其他背景信息还有：

中间件团队规模不大，大约 6 人左右。

中间件团队熟悉 Java 语言，但有一个新同事 C/C++ 很牛。

开发平台是 Linux，数据库是 MySQL。

目前整个业务系统是单机房部署，没有双机房。

针对前浪微博的消息队列系统，采用“排查法”来分析复杂度，具体分析过程是：

这个消息队列是否需要高性能

我们假设前浪微博系统用户每天发送 1000 万条微博，那么微博子系统一天会产生 1000 万条消息，我们再假设平均一条消息有 10 个子系统读取，那么其他子系统读取的消息大约是 1 亿次。

1000 万和 1 亿看起来很吓人，但对于架构师来说，关注的不是一天的数据，而是 1 秒的数据，即 TPS 和 QPS。我们将数据按照秒来计算，一天内平均每秒写入消息数为 115 条，每秒读取的消息数是 1150 条；再考虑系统的读写并不是完全平均的，设计的目标应该以峰值来计算。峰值一般取平均值的 3 倍，那么消息队列系统的 TPS 是 345，QPS 是 3450，这个量级的数据意味着并不要求高性能。

虽然根据当前业务规模计算的性能要求并不高，但业务会增长，因此系统设计需要考虑一定的性能余量。由于现在的基数较低，为了预留一定的系统容量应对后续业务的发展，我们将设计目标设定为峰值的 4 倍，因此最终的性能要求是：TPS 为 1380，QPS 为 13800。TPS 为 1380 并不高，但 QPS 为 13800 已经比较高了，因此高性能读取是复杂度之一。注意，这里的设计目标设定为峰值的 4 倍是根据业务发展速度来预估的，不是固定为 4 倍，不同的业务可以是 2 倍，也可以是 8 倍，但一般不要设定在 10 倍以上，更不要一上来就按照 100 倍预估。

这个消息队列是否需要高可用性

对于微博子系统来说，如果消息丢了，导致没有审核，然后触犯了国家法律法规，则是非常严重的事情；对于等级子系统来说，如果用户达到相应等级后，系统没有给他奖品和专属服务，则 VIP 用户会很不满意，导致用户流失从而损失收入，虽然也比较关键，但没有审核子系统丢消息那么严重。

综合来看，消息队列需要高可用性，包括消息写入、消息存储、消息读取都需要保证高可用性。

这个消息队列是否需要高可扩展性

消息队列的功能很明确，基本无须扩展，因此可扩展性不是这个消息队列的复杂度关键。

为了方便理解，这里我只排查“高性能”“高可用”“扩展性”这 3 个复杂度，在实际应用中，不同的公司或者团队，可能还有一些其他方面的复杂度分析。例如，金融系统可能需要考虑安全性，有的公司会考虑成本等。

综合分析下来，消息队列的复杂性主要体现在这几个方面：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。

“前浪微博”的消息队列设计才刚完成第 1 步，专栏下一期会根据今天识别的复杂度设计备选方案，前面提到的场景在下一期还会用到哦。

小结

今天我为你讲了架构设计流程的第一个步骤“识别复杂度”，并且通过一个模拟的场景讲述了“排查法”的具体分析方式，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。尝试用排查法分析一下你参与过或者研究过的系统的复杂度，然后与你以前的理解对比一下，看看是否有什么新发现？

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！）

上一篇　架构专栏特别放送 | “华仔，放学别走！”第1期

下一篇　11 | 架构设计流程：设计备选方案

精选留言 (59)

写留言

pk
2018-05-21

67

文中有一句话：TPS 1780 并不高。这个结论怎么来的？作者经验丰富，见过很高的TPS。但作为新手，如何衡量这个性能要求高还是不高呢？

展开

作者回复: 对于架构师来说，常见系统的性能量级需要烂熟于心，例如nginx负载均衡性能是3万左右，mc的读取性能5万左右，kafka号称百万级，zookeeper写入读取2万以上，http请求访问大概在2万左右。
具体的数值和机器配置以及测试案例有关，但大概的量级不会变化很大。

如果是业务系统，由于业务复杂度差异很大，有的每秒500请求可能就是高性能了，因此需要针对业务进行性能测试，确立性能基线，方便后续架构设计做比较。

公号-代码...
2018-05-19

47

识别复杂度心得

架构设计由需求所驱动，本质目的是为了解决软件系统的复杂性；为此，我们在进行架构设计时，需要以理解需求为前提，首要进行系统复杂性的分析。具体做法是：

...

展开

XP
2018-05-19

40

感觉这个专栏定位是扫盲，留言的读者都是老司机

展开

作者回复: 应该是兄台水平高，所以觉得定位是扫盲，实际上很多人不知道这些内容，上网搜索也搜不到的

云辉
2018-05-20

21

复杂度问题，还有是方案简单，但是沟通协调成本高，跨部门了，请问华仔，你们是自己推动，还是技术PMO推动。

展开

作者回复: 架构师推动是主要的，架构师需要五项全能：技术，沟通，推动，管理，撕逼😀😀😀

空档滑行
2018-05-20

16

说下之前改造的一个系统，当时是这个系统从其他系统同步数据，经过一整套流程后将数据拆解到本地库的各个业务表中。
原来的系统是一个单机多线程程序，到了大促的时候延时非常厉害，因为马上又要大促了，首先想到的是扩展性的问题，于是就做了无状态服务拆分，可以横向扩展。在这过程

成因为要控制单个同步实体任务的并发在处理幂等性上也花很大的功夫。做完这个后，...
展开

作者回复: 所以系统复杂度识别是非常重要的，也没那么容易

三月沙@wec...
2018-05-29

9

专栏本身的牛逼之处在于系统化，相似概念其实在日常架构中已经多有涉及，但是专栏用自己独有的方法论浅显易懂的论述了这些概念以及它们的关系，值得在缺乏经验的团队广而告之

黑客悟理
2018-05-26

9

留言全是干货.....
展开

herome
2018-05-19

8

最近在做一个平台化的crm系统，也就是公司各个业务线都可以接入 ，或者外面的业务线也能接入。但是我们在开发过程中，比如查看商家等级列表这个接口，以前所有的接入方，都是查看就是查看，但是有个主要对接方提出了个需求是，查看后如果不存在 就插入一个默认等级。 ，但是其他对接方没有这样的需求，也就说原接口不能变，如果没有其他接入方我直接修改原来的接口即可，根本不考虑兼容。类似的问题数不胜数， 我们每次...
展开

作者回复: 试试设计模式的策略模式或者职责链模式

huayu00
2018-06-23

7

“http请求访问大概在2万左右”
请问这个是指什么，数字怎么得来的？

joyafa
2018-05-31

4

我现在做的是交易转发网关，性能瓶颈很大一部分也受上级券商网关影响，系统无状态，无数据库操作，因为涉及到交易，对可用性要求很高，需要能快速处理客户端的请求，经过多次压测，也没有排查出自身系统问题出在哪里，对系统压测，以及测试得到的数据该怎么分析，怎么找出问题所在？

展开

作者回复: 网关的性能可以参考nginx的性能，如果你们的测试数据和nginx做网关差异很大，那可能是并发模型甚至记日志这些不起眼的操作给拖慢了，如果差不多，那基本就是性能极限了

Sir Peng
2018-05-24

3

华仔，架构师要具备的五项全能中，"撕逼"能在哪种情况中得已体现？

作者回复: 说服别人重构，选择方案等很多时候要撕逼😂😂

彡工鸟
2018-05-20

3

个人看法，在顶层上看一个系统，确实会有高可用，高性能，可扩展等等多个复杂度的要求，而且可能还真的无法分出个优先级。这样就落到都要抓，都要硬，最终无法分出个主次，导致设计四不像的系统出来。针对这种情况，是否就应该动用子系统一说，将系统拆分到一定的粒度，可以聚焦到一个，或者至多两个复杂度上？而子系统自然有子系统的架构设计，技术选型来针对性地解决其复杂度问题

展开

作者回复: 这是一种思路，比较适合业务系统，中间件系统就不太适合这种方式

xiaoya
2018-08-02

2

我是个新手，听一遍就忘了。记住的不多😭不开心。是我接触的太少不太重视这些吧，这些很关键?这么多理论的东西，嗨，我心性太差了。一口老想都吃掉，结果都不好。认识自己驾驭自己，一点点

作者回复: 我也是从新手过来的，坚持就有收获，不要求一次就全部听懂了，

张国胜
2018-06-28

2

高可用和高扩展两块有问题。高可用问题提现在，整个系统很脆弱，容易出现某块的性能故障，如数据库的io暴涨导致所有业务访问出现问题。或者发送验证码的业务出现故障，导致用户无法下单，但没有验证码实际上可以不用与是否下单成功相关联。出现问题后，由于业务复杂，难以排查问题。
高扩展问题提现在，整个系统都是为了完成业务功能在写，基本没有抽象和设计模式，...
展开

何晓亮
2018-06-26

2

留言中提到了常见系统性能量级，问一下windows系统的量是怎么样的？

renwotao
2018-06-13

2

公司架构日志占了百分之二十的cpu，有什么好的解决方案吗

作者回复: 日志压缩，采样，隔离

重剑
2018-05-29

2

太棒了，对于我这种菜鸟来说讲的很好！
展开

Ryan Fen...
2018-05-23

2

tps、qps一般用什么测？我看你的tps是qps十倍，意思就是多少子系统就是多少倍？这个数据不考虑网络通讯和系统复杂度差异吗？

展开

作者回复: 这是假设的，具体业务具体分析，可以用压测工具测试系统的能力，用监控系统监控线上的实际数值

行下一首歌
2018-05-21

2

微服务架构，是解决了什么复杂性问题呢

展开

作者回复: 可扩展

MichaelJY
2018-05-21

2

现在遇到这样的系统:
1.前后端没有分离，因为业务情况需要多窗口显示，然后是通过div分隔的，在多窗口的使用下会出现id冲突或者在a窗口的操作影响到b窗口的数据
2.系统分为三个子系统，业务子系统处理具体业务，公共子系统处理类似登录，权限，审批等，定时任务负责调度，三者之间通过http交互，没有集群，现在业务子系统耦合比较...

展开

作者回复: 优先级需要根据业务和团队综合判断，不是单纯技术角度判断。
例如，如果停服影响不大的话，那2可能优于4；
如果历史债务导致投入很大，那3可能优于2