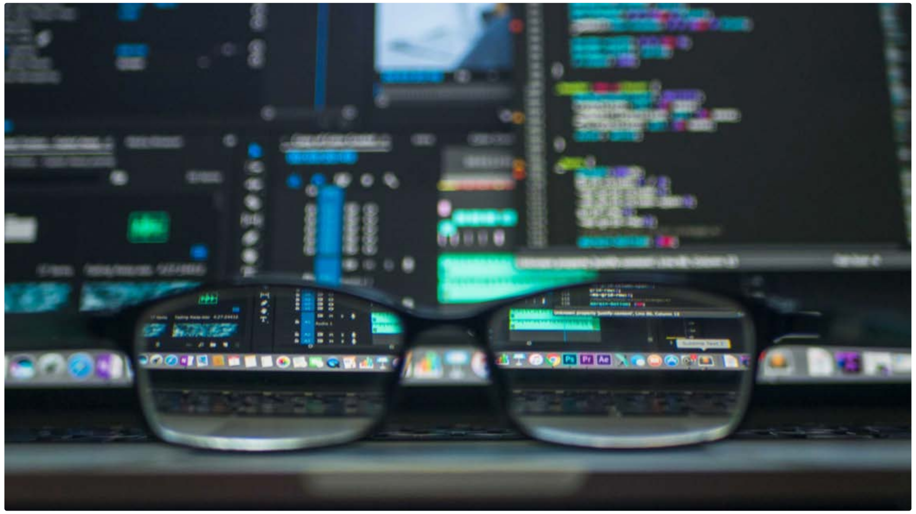
10 | 架构设计流程：识别复杂度

2018-05-19 李运华



10 | 架构设计流程：识别复杂度
李运华
- 00:14 / 10:48

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歧，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！）

XP

2018-05-19

感觉这个专栏定位是扫盲，留言的读者都是老司机

作者回复

2018-05-19

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

pk

2018-05-21

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？

作者回复

2018-05-21

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-Java大后端

2018-05-19

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：

（1）构建复杂度的来源清单——高性能、可用性、扩展性、安全、低成本、规模等。

（2）结合需求、技术、团队、资源等对上述复杂度逐一分析是否需要？是否关键？

“高性能”主要从软件系统未来的TPS、响应时间、服务器资源利用率等客观指标，也可以从用户的主观感受方面去考虑。

“可用性”主要从服务不中断等质量属性，符合行业政策、国家法规等方面去考虑。

“扩展性”则主要从功能需求的未来变更幅度等方面去考虑。

（3）按照上述的分析结论，得到复杂度按照优先级的排序清单，越是排在前面的复杂度，就越关键，就越优先解决。

需要特别注意的是：随着所处的业务阶段不同、外部的技术条件和环境的不同，得到的复杂度问题的优先级排序就会有所不同。一切皆变化。

云辉

2018-05-20

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。

作者回复

2018-05-21

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

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，又对非关键流程做了消息解耦，提高主流程的处理速度。
系统上线后运行稳定，但是到了大促当日发现速度提升很有限，将机器扩展到10台速度只提升了2-3倍，而且数据库和应用压力都很小。
后来分析下来瓶颈竟然是在数据库中间件上。其他系统大量的慢sql导致中间件处排队严重。
现在想起来，其实还是改造前优先级没搞清楚，根据之前的经验就把问题归结到扩展性上，投入了大部分精力在扩展性和可用性上，而没有对原来单机系统的性能做完整的测试和评估。其实当时做一个单机程序的压测大概就可以判断出问题所在(改造过程中也发现老的程序确实有bug)。还有就是高性能的优先级远大于扩展性和可用性，因为这个系统重启的影响非常小。改造的时候过于理想化的，想做一个各个方面均衡完善的架构

作者回复

2018-05-20

所以系统复杂度识别是非常重要的，也没那么容易

三月沙@wecatch

2018-05-29

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

herome

2018-05-19

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， 我们每次要么是if else这种代码走不同的逻辑，要么是 新写个接口，现在我们的代码的维护已经很混乱了，一个简单的逻辑，如查看等级列表，我就要维护好几套接口，好几套逻辑。有想过改成可配置化的，但是没有好的思路。华仔对这方面有好的建议吗？��

作者回复

2018-05-19

试试设计模式的策略模式或者职责链模式

---

joyafa

2018-05-31

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？

作者回复

2018-06-01

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

黑客悟理

2018-05-26

留言全是干货……

Sir Peng

2018-05-24

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

2018-05-24

说服别人重构，选择方案等很多时候要撕逼����

Ryan Feng

2018-05-23

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？

作者回复

2018-05-23

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

Benny

2018-05-22

结合作者的分享，简单分析了一下公司的系统，目前公司的系统主要重心放在了高可用和高扩展，因为业务的连续性和功能多样化是目前比较重视的。当前用户量比较不大，高性能没有重点考虑，但随着业务增长，也慢慢出现问题，消息队列和数据库成为瓶颈。

秋天

2018-05-21

有没有系统提升架构思维的资料推荐，感觉还没有打破那个灵感，请大神指教？

作者回复

2018-05-22

我的专栏就是呢，架构设计目的，架构设计原则，架构设计流程……都是架构设计系统化的技能点

DullBird

2018-05-20

分析现在做的采集系统，主要是控制单系统融合了（前端业务，任务调度,分发，自实现系统内缓存，采集机状态管理，数据入库功能）。采集机主要实现了所有的采集功能，采集机是集群的。
业务性能要求比较低，页面基本没什么人访问，后台业务主要分采集项，频繁的一点就任务A是10分钟1W个任务（不间断），任务B是4小时内完成2W个任务，分析业务不是瓶颈，但是2W个任务由于依赖客户提供的登入方式实际采集耗时在10小时左右（这点问题应该需要协调客户处理或者换方式实现）。
主要复杂度在于系统的耦合性，控制单系统功能太多了，想添加一个任务逻辑的复杂模块，很难加。想拆分成子系统（分为前端子系统，缓存子系统，采集机管理子系统，数据入库子系统，任务调度子系统。前二个优先级比较高，后面的可以等需要的时候拆分），把单系统内的调用替换成rpc框架的系统间调用，但是系统间调用比原来的调用多了一种超时状态。原来的系统状态性比较强，如果远程调用的不确定性，可能需要业务改为无状态的。那么需要改动的量比较大。（rpc调用异常的处理方式了结的比较少，希望老师给点指点，谢谢）。
其次复杂度在于高可用，我们的系统宕机时间没有那么严格，但是需要在一定时间内切换，30分钟即可。采集机集群通过任务轮询，已经实现了高可用。但是控制单系统做高可用有一个问题是单系统实现了内部的缓存，如果不把缓存抽离出来就会有一致性问题。然后实现主备，单节点可用，出了故障另外一台设备升级为主节点，通过协商的方式。（脑裂的情况暂不考虑，客户的内网相对稳定。通过监控人工来恢复。）

我才是二亮

2018-05-19

前段时间在开发一个系统，系统有多个定时任务，其中有一个关联性的任务，当A任务成功后，过5到10分钟执行B任务，但因为系统对高性能要求不高，所以直接采用两个定时任务，一个10分钟跑一次，另外一个15分钟从表中遍历未处理的数据进行处理，了。
如果考虑高性能，可以考虑使用延时队列，当A任务执行成功后，发一条延时消息给消息流系统进行处理。

老师所说的排查法，简单明了，很适合新手对系统架构的分析，感谢分享。

曹铮

2018-05-19

疑问:1 文中描述架构师还是凭借老司机经验发现系统的问题在于耦合度 2 tps qps的表示的意思已经被错用，正确的理解是什么？文中看来t指外部产生的事务？

作者回复

2018-05-19

经验和技巧都不可少，行业背景知识也很重要

张国胜

2018-06-28

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，导致业务很难扩展，基本上就在之前的基础上来回改。

何晓亮

2018-06-26

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？

---

huayu00

2018-06-23

“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？

---

renwotao

2018-06-13

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

2018-06-13

日志压缩，采样，隔离

Ben

2018-06-09

读到了每通知一个新系统就没有了！了！了

吴天

2018-06-05

很有启发，不过对于性能方面有什么测试或者收集工具吗？好确定当前系统的性能情况，以及如何选择架构去满足性能需求？根据一些产品自身有关性能的介绍吗？

作者回复

2018-06-05

性能测试和硬件，场景都相关，测试工具有jmeter，loaderunner

彼得.林

2018-06-01

恳求老师答疑：一直困惑，文中提到的一条微博要发出十几个通知事件，那么是发出一条事件到MQ，然后MQ分别推送到各个子系统进行消费；还是分别发出十几个特定的事件到MQ，然后各个特定事件到指定子系列消费？

作者回复

2018-06-01

发一条，各子系统都来收这一条消息，参考设计模式的订阅模式

重剑

2018-05-29

太棒了，对于我这种菜鸟来说讲的很好！

---

itperson

2018-05-26

以前的系统基本不需要考虑高可用和高性能 但我有个问题在于高性能和高可用具体是否有衡量标准 请在后面的课程详细讲解一下 自己如何证明验证高可用和高性能

作者回复

2018-05-27

从需求出发，如果你们的系统现在只能支持每秒处理1000个请求，那么需求要求每秒2000就是高性能了，高可用也类似，电信行业要求5个9，互联网核心业务要求4个9，企业应用可能3个9也够了

成功

2018-05-24

TPS很多人关注，其Qps更重要

但莫

2018-05-24

请教个问题，多高的tps，qps算高呢，是如何界定的。

作者回复

2018-05-24

看业务复杂度，了解常见开源软件的性能，可以做对比参照

---

sundy

2018-05-24

作者果然是老司机，讲的非常好

作者回复

2018-05-24

老司机开车，又稳又快��

华烬

2018-05-23

两个问题请教一下
1，tps1380对于MySQL库来说的话应该不低了吧
2，扩展性只从功能考虑么，不用考虑容量之类的？

作者回复

2018-05-24

1. mysql的tps和写入的数据有关，我们测试了mysql写入k-v数据，主键存储k，tps上万
2. 容量扩展属于性能部分，叫可伸缩性

---

tick

2018-05-23

qps，tps这些概念还是有些模糊，能说下吗?

作者回复

2018-05-23

请自行搜索

JOEL

2018-05-22

高性能 高可用 可扩展都是质量需求，是质量需求的代表吧。另外还有很多质量需求指标，本系列文章都没提过，但其实个人觉得也需要考虑的。比如公司建立OA系统就需要考虑互操作性，另外还有易用性，安全性，可移植行性等。

作者回复

2018-05-22

是的，我只是列出了常见的，不是只有这几个，例如有的公司项目面临的复杂度是如何以低成本尽量满足客户变态的新技术需求��

On my way

2018-05-22

“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 445，QPS 是 4450，这个量级的数据意味着并不要求高性能。”这里的TPS/QPS是不是算错了？TPS：345；QPS：3450？

作者回复

2018-05-22

嗯，我是拿计算器算的，输入错了，没注意，感谢指正

李志博

2018-05-22

为什么感觉这张的内容前面几张曾经看到过......还有您说的常见系统性能要烂熟于心，请问有具体的博客地址或什么的地方讲都大概什么性能了嘛，我们需要背下来？

作者回复

2018-05-22

前面是理论，现在开始教你怎么落地了。
性能指标需要自己去积累，最好背下来

Mike Wang

2018-05-22

去年参与了个行为系统项目，数据采集到队列，两个消费端，一个到jstorm，一个到es，jstorm存储数据到mongodb和hbase，hive在Hadoop上进行离线计算，最后查询接口通过presto合并mongodb、hive、mysql中查询结果，前期只是考虑了高性能，后面发现处理链路长，容易中断，所以也考虑了高可用，数据采集部分考虑了可扩展，查询程序部分因为业务复杂和工期紧张可扩展未考虑

行下一首歌

2018-05-21

微服务架构，是解决了什么复杂性问题呢

作者回复

2018-05-22

可扩展

YMF_WX1981

2018-05-21

对于一个从没从事过架构工作的前端来说，所谓复杂和识别复杂，我的理解是：当这个顶层设计不能对业务作出良好响应，改动困难，就是复杂。

识别复杂不会哦，按照常规思维，我们总会画这样的框框图，将变化的东西按业务归位到模块，或，有些功能按复用需要抽象成组件，于是有好些系统和子系统，而如何“画”这图？标准数据是？解决什么问题对应技术是？这是我认为改课程的价值。

作者回复

2018-05-22

前端架构和app架构的主张复杂度在于可扩展性

汉斯·冯·拉特

2018-05-21

这些内容我不知道的，由于公司定位、项目定位的原因，没能接触这些，所以还是很有收获的，期待后续的内容！

MichaelJY

2018-05-21

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严重。业务子系统大概可以分为接入层，业务处理层，结算层，现在是一整个系统，经常发现结算层的逻辑变更，会导致接入层和业务层不可用
3.因为历史原因，对于数据校验方面做的比较差，经常出现数据格式问题，当然新增代码已经处理了这些，但是历史债务比较严重，且业务迭代很快
4.整个系统是单点，所有发版升级都会停服

我觉得最急的是4 >2 >1 >3，不知道对不对

作者回复

2018-05-21

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

合民

2018-05-21

这期最大的收获是如何预估系统的高性能要求，找到了方向和实践思路，对我们系统分析后，其实tps和qps并不高，终于不用拍脑门了，哈哈！

作者回复

2018-05-21

脑门拍多了，脑门也疼呀����

Classtag

2018-05-20

我参与设计了一个离线数据生产的系统。主要的功能就是将HDFS上的大文件按天增量刷到线上高可用的存储数据库中比如dynamodb. 慢点有几个：文件都是全量数据,每次写全量不靠谱 太费钱也浪费计算资源；数据必须完全正确地写入，部分数据需要时实行要求. 数据量每天大概5~10T .

作者回复

2018-05-20

你的系统复杂度体现在规模和性能，后面有一章节专门描述

卡莫拉内西

2018-05-20

学习了，要有针对性的分析复杂度，而不是照搬照抄网上的某些架构，要知其所以然，除了，高性能，高可用，可扩展，安全性，成本，五大点之外还有其他需要考虑的点吗

作者回复

2018-05-20

这几个是常见的，不限于这几个，不同业务领域有不同的复杂度

彡工鸟

2018-05-20

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

作者回复

2018-05-20

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

Kyle

2018-05-19

前一份工作由于历史问题，控制层与服务层通过rmi单节点连接。由于服务的增长，rmi调用在横向扩展的时候必须控制层和服务层的服务跟着一起增加，而且单个节点的访问方式无法做到负载均衡。此外由于原始项目没引入maven，我们在包管理上也多了很多重复工作。综合起来，我们项目原始架构的复杂度主要在“可扩展性”上，包括两方面的扩展性问题：代码新功能、新组件引入的复杂性，还有就是系统横向扩展的复杂性。针对这些情况我们将项目重构分三步走：第一步为每个项目引入maven，第二步引入spring boot（当初想法是了解到spring boot能快速引入各种组件，比如mybatis二级缓存，solr搜索引擎，但为了学习怎么去引入这些组件也花了不少功夫），第三步改rmi调用为dubbo + zookeeper方式调用。
今天的课程让我收益最大、最有同感的就是：如华哥所说，重构项目要做的事太多，会让人无从下手，而且我在这个过程中时常焦躁不已，这个时候就要分清轻重缓急，分步执行。

王磊

2018-05-19

我的体会：
1. 高性能，高可用，高扩展即是考量系统整体，也是逐一考量各个模块的，包括自己的模块，也包括第三方中间件用于选型；
2. 这里的复杂度我理解为系统的痛点所在，如果痛点处理不好，系统是失败的；
3. 我理解我们系统的痛点之一是，中间表生成的可扩展，随着业务的发展，中间表需要不断演进，中间表的新增和修改都需要中间表数据重新生成(原始数据还在)，这个过程导致上线耗时;另外也会迫使所有使用此中间表的查询需要修改，需要回归，增加了开发和测试的工作量，也带来额外的风险。在考虑这里该如何改造，还没有好办法。