45 | 架构重构内功心法第一式：有的放矢

2018-08-09 李运华

45 | 架构重构内功心法第一式：有的放矢

李运华

00:00 / 10:22

在专栏第8期"架构设计三原则"中的演化原则部分，我提到了系统的架构是不断演化的，少部分架构演化可能需要推倒重来进行重写，但绝大部分的架构演化都是通过架构重构来实现的。相比全新的架构设计来说，架构重构对架构师的要求更高，主要体现在：

- 业务已经上线，不能停下来

架构重构时，业务已经上线运行了，重构既需要尽量保证业务继续往前发展，又要完成架构调整，这就好比"给飞行中的波音747换引擎"；而如果是新设计架构，业务还没有上线，则即使做砸了对业务也不会有太大影响。

- 关联方众多，牵一发动全身

架构重构涉及的业务关联方很多，不同关联方的资源投入程度、业务发展速度、对架构痛点的敏感度等有很大差异，如何尽量减少对关联方的影响，或者协调关联方统一行动，是一项很大的挑战；而如果是新设计架构，则在新架构上线前，对关联方没有影响。

- 旧架构的约束

架构重构需要在旧的架构基础上进行，这是一个很强的约束，会限制架构师的技术选择范围；而如果是新设计架构，则架构师的技术选择余地大得多。

即使是我们决定推倒到重来，完全抛弃旧的架构而去设计新的架构，新架构也会受到旧架构的约束和影响，因为业务在旧架构上产生的数据是不能推倒重来的，新架构必须考虑如何将旧架构产生的数据转换过来。

因此，架构重构对架构师的综合能力要求非常高，业务上要求架构师能够说服产品经理暂缓甚至暂停业务来进行架构重构；团队上需要架构师能够与其他团队达成一致的架构重构计划和步骤；技术上需要架构师给出让技术团队认可的架构重构方案。

总之，架构重构需要架构师既要说得动老板，也要镇得住同事；既要技术攻关，又要协调资源；既要保证业务正常发展，又要在指定时间内完成目标……总之就是十八般武艺要样样精通。

说了那么多架构重构的难度，千万不要被困难所吓倒，架构师正是需要在原来一团乱麻中找到线索，然后重新穿针引线，帮助业务进一步腾飞发展。接下来我将分3期传授我的架构重构内功心法，今天先来看第一式：有的放矢。

通常情况下，当系统架构不满足业务的发展时，其表现形式是系统不断出现各种问题，轻微一点的如系统响应慢、数据错误、某些用户访问失败等，严重的可能是宕机、数据库瘫痪、数据丢失等，或者系统的开发效率很低。开始的时候，技术团队可能只针对具体的问题去解决，解决一个算一个，但如果持续时间较长，例如持续了半年甚至一年情况都不见好转，此时可能有人想到了系统的架构是否存在问题，讨论是否是因为架构原因导致了各种问题。一旦确定需要进行架构重构，就会由架构师牵头来进行架构重构的分析。

当架构师真正开始进行架构重构分析时，就会发现自己好像进了一个迷雾森林，到处都是问题，每个问题都需要解决，不知道出路在哪里，感觉如果要解决所有这些问题，架构重构其实也无能为力。有的架构师一上来搜集了系统当前存在的问题，然后汇总成一个100行的Excel表格，看到这样一个表格就懵了：这么多问题，要到猴年马月才能全部解决完啊？

期望通过架构重构来解决所有问题当然是不现实的，所以架构师的首要任务是从一大堆纷繁复杂的问题中识别出真正要通过架构重构来解决的问题，集中力量快速解决，而不是想着通过架构重构来解决所有的问题。否则就会陷入人少事多头绪乱的处境，团队累死累活弄个大半年，最后发现好像什么都做了，但每个问题都依然存在。尤其是对于刚接手一个新系统的架构师或者技术主管来说，一定要控制住"新官上任三把火"的冲动，避免搞大饼式或者运动式的重构和优化。

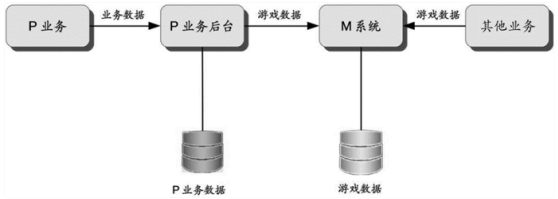
我们来看几个具体的重构案例。

1.后台系统重构：解决不合理的耦合

M系统是一个后台管理系统，负责管理所有游戏相关的数据，重构的主要原因是因为系统耦合了P业务独有的数据和所有业务公用的数据，导致可扩展性比较差。其大概架构如下图所示。

举一个简单的例子：数据库中的某张表，一部分字段是所有业务公用的“游戏数据”，一部分字段是P业务系统“独有的数据”，开发时如果要改这张表，代码和逻辑都很复杂，改起来效率很低。

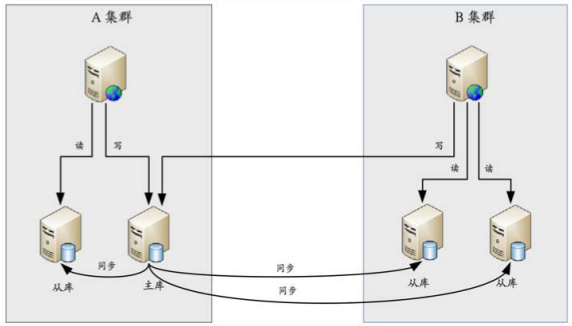
针对M系统存在的问题，重构目标就是将游戏数据和业务数据拆分，解开两者的耦合，使得两个系统都能够独立快速发展。重构的方案如下图所示。



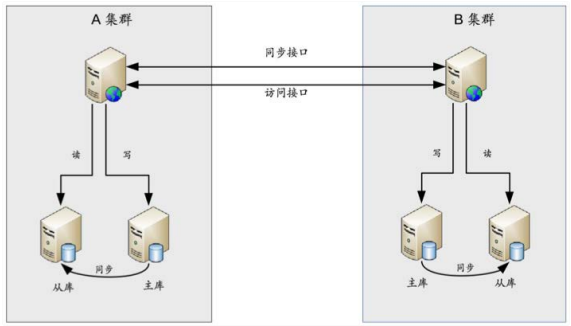
重构后的效果非常明显，重构后的M系统和P业务后台系统每月上线版本数是重构前的4倍！

2.游戏接入系统重构：解决全局单点的可用性问题

S系统是游戏接入的核心系统，一旦S系统故障，大量游戏玩家就不能登录游戏。而S系统并不具备多中心的能力，一旦主机房宕机，整个S系统业务就不可用了。其大概架构如下图所示，可以看出数据库主库是全局单点，一旦数据库主库不可用，两个集群的写业务都不可用了。



针对S系统存在的问题，重构目标就是实现双中心，使得任意一个机房都能够提供完整的服务，在某个机房故障时，另外一个机房能够全部接管所有业务。重构方案如下图所示。



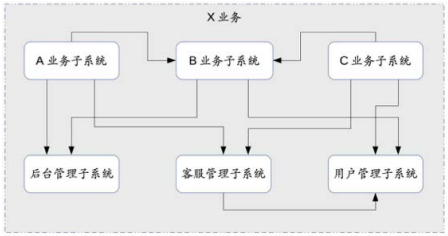
重构后系统的可用性从3个9提升到4个9，重构前最夸张的一个月有4次较大的线上故障，重构后虽然也经历了机房交换机宕机、运营商线路故障、机柜断电等问题，但对业务都没有什么大的影响。

3.X系统：解决大系统带来的开发效率问题

X系统是创新业务的主系统，之前在业务快速尝试和快速发展期间，怎么方便怎么操作，怎么快速怎么做，系统设计并未投入太多精力和时间，很多东西都“塞”到同一个系统中，导致到了现在已经改不动了。做一个新功能或者新业务，需要花费大量的时间来讨论和梳理各种业务逻辑，一不小心就踩个大坑。X系统的架构如下图所示。

X系统的问题看起来和M系统比较类似，都是可扩展性存在问题，但其实根本原因不一样：M系统是因为耦合了不同业务的数据导致系统可扩展性不足，而X系统是因为将业务相关的所有功能都放在同一个系统中，导致系统可扩展性不足；同时，所有功能都在一个系统中，也可能导致一个功能出问题，整站不可用。比如说某个功能把数据库拖慢了，整站所有业务跟着都慢了。

针对X系统存在的问题，重构目标是将各个功能拆分到不同的子系统中，降低单个系统的复杂度。重构后的架构如下图所示（仅仅是示例，实际架构远比下图复杂）。



重构后各个系统之间通过接口交互，虽然看似增加了接口的工作量，但整体来说，各系统的发展和开发速度比原来快了很多，系统也相对更加简单，也不会出现某个子系统有问题，所有业务都有问题。

这三个系统重构的方案，现在回过头来看，感觉是理所当然的，但实际上当时做分析和决策时，远远没有这么简单。以M系统为例，当时我们接手后遇到的问题有很多，例如：

- 数据经常出错。
- M系统是单机，单机宕机后所有后台操作就不能进行了。
- 性能比较差，有的操作耗时好久。
- 界面比较丑，操作不人性化。
- 历史上经过几手转接，代码比较混乱。
- 业务数据和游戏数据耦合，开发效率很低。

从这么多问题中识别出重构的目标，并不是一目了然的；而如果想一下全部解决所有这些问题，人力和时间又不够！所以架构师需要透过问题表象看到问题本质，找出真正需要通过架构重构解决的核心问题，从而做到有的放矢，既不会耗费大量的人力和时间投入，又能够解决核心问题。这对架构师的分析和判断能力要求非常高，既不能看到问题就想到要架构重构，也不能只是针对问题进行系统优化，判断到底是采取架构重构还是采取系统优化，可能不同的架构师和团队都有不同的看法。这里分享一个简单的做法：假设我们现在需要从0开始设计当前系统，新架构和老架构是否类似？如果差异不大，说明采取系统优化即可；如果差异很大，那可能就要进行系统重构了。

那原来发现的那些非架构重构问题怎么办呢？当然不能放任不管。以M系统为例，我们在重构完成后，又启动了多个优化的项目去优化这些问题，但此时的优化主要由团队内部完成即可，和其他团队没有太多关联，优化的速度是很快的。如果没有重构就进行优化，则每次优化都要拉一大堆关联业务的团队来讨论方案，效率非常低下！

小结

今天我为你讲了架构重构的时候需要做到有的放矢，避免像通过架构重构来解决所有问题，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，分析一下你目前开发的系统，你觉得需要架构重构吗？原因和理由是什么？

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！）

彼得.林

2018-08-09

总之，架构重构需要架构师既要说得动老板，也要镇得住同事；既要技术攻关，又要协调资源；既要保证业务正常发展，又要在指定时间内完成目标……总之就是十八般武艺要样样精通。

老师，这些都是架构师的软技能吧？市场上的课都偏技术，可以讲讲这些吗？谢谢

作者回复

2018-08-09

这个够开三个专栏了��并且我也不太擅长这些软技能培训

吴科��

2018-08-09

系统优化，我们公司就是一个字，拆，拆数据库，拆服务，没有依赖就没有伤害

作者回复

2018-08-09

经典，没有依赖就没有伤害，但实际上我们依赖还是少不了，我们尽量降低依赖程度，接口依赖是比较可控的

A:春哥大魔王

2018-08-09

重构系统首先是建立在熟悉系统业务上，有了全局的业务视角在架构选型和构件上才会有的放矢。

华哥可以开个架构师软技能专栏，继续订阅。

作者回复

2018-08-10

没这个能力呀，写几篇文章可以，这个专栏有点难��

张玮(大圣)

2018-08-09

确定是不是要重构挺难的，那就等到问题频出，改到想吐的时候做重构吧，

大公司在创新，小公司也在创新，一般2.3年的系统设计不会太差吧，除非影响业务了，影响老板心情了，

归纳下：能优化的别修补，能修补的别重构，保证核心业务OK，更好的支撑创新业务线发展吧 哈哈

如果要重构，拉都拉不住，那就做好预备方案，保证新旧系统并行，时间可控，做好切换时风险控制，要不然，萌萌哒，��

作者回复

2018-08-10

是的，架构重构是大动作，不要轻易采用，大招不要乱放��

恋着歌

2018-08-13

当初的一个 demo 到现在已经做了3年了，迟迟没有重构。作为一个前端我感觉现在急需数据库层的重构。现在一个业务涉及多个 site，但是一个 site 一个库，最近上线通过程序 addsite 功能，导致多种问题。1，要实现动态数据源，2，要引入 mq 通知另一台 tomcat 更新数据源。历时一个半月才上线了一个粗糙版本。我给的方案是同一个业务就同一个数据库，上面的两问题都不存在了。现在重构的最大阻力是旧数据迁移，请问老师在这放面有比较好的建议吗？

作者回复

2018-08-13

重构通常都会涉及数据，为了避免风险，我们一般都是分步骤实施，从易到难，先将关联度不大的，比较独立的数据重构，再来动复杂的。

你的这个案例业务上的场景我没看的太懂，如果你说的site是我常规理解的站点，那一个site一个库反而是合理的，至于mq通知tomcat更新数据源，没理解这样。

建议和后端同学一起分析讨论重构方案。

Geek_6dbf4e

2018-08-12

老师，我们现在公司遇到的问题和你说的X系统很相似，现在也在逐步将那些热门业务做成微服务，拆分出去，但是涉及到的数据表，拆到其它RDS上去，涉及到的跨库查询很麻烦。现在只能做成接口了

作者回复

2018-08-13

这个没办法避免，短期内大家会不习惯，原来一条SQL语句搞定的事情，现在要调用接口还要改逻辑，但长远来看是好的，SQL的问题在于后续的系统越来越复杂后，很容易出现将整个系统拖垮的慢查询，业务扩展也很麻烦，因为逻辑都在SQL中了

one day

2018-08-13

老师好，接上老师的回复。原来单一的数据库，业务模块分开，但数据层面查询关联太多，不痛不知，至于为什么是禁止关联查询，理想状态，关联拆开，数据库没也拆开？？多台数据库的话？

合能力便可用。至于拆不开的，通常在管理端的业务，可建数据仓库，c端用户的情况，关联拆在数据库层面，服务之间调用聚合数据。慢操作分两种情况，慢查询，利用异步加同步的方法，聚合结果，慢修改，利用mq的能力不断优化，上自治重构机制去应对最终有效。当然是那个，会考虑很多问题，还有身在何处，不准怎么成长自己。重构也许最大的收获就是体验。

作者回复

2018-08-10

那就可以尝试，毕竟我对你们的业务不熟悉，意见仅供参考

宋岩

2018-08-10

老师您好，我们现在面临一个运行18年的.net业务系统，业务一刻不能停止，大约2000张oracle表，大量的存储过程函数等实现的业务逻辑，现在要用java来重构，面临的问题就是好多大表都在Oracle中，想去o就涉及到很多表关联，做不到垂直拆分，业务耦合太紧，您有什么好的建议或者我们该从哪方面切入？

作者回复

2018-08-10

1. 数据先不动，先拆分代码到多个子系统，然后再拆分数据
2. 重写一套新的，然后做数据割接

老板决心大的话，方案2实施更快，但风险高；老板决心不大的话，那就方案1，实施周期长，风险会小一些

黑小子在路上

2018-08-10

目前系统采用类似微服务模式。如有1后台管理，2商家pc后台，3商家app接口服务，4server，都是独立部署。前面的1 2 3都依赖4。目前的问题是1也会访问数据库，server也访问数据库，有一部分重复。2 和3 也有重复逻辑。目前一个小需求，都可能要改多个端，由多个人介入，开发成本比较高。近期准备重构，使用多模块形式来聚合工程，提取公用服务，但会保留现有独立部署模式，以此来达到提效的目的:-O

作者回复

2018-08-10

多模块聚合不是好的重构手段，版本开发和测试部署一样比较麻烦，我认为你们应该将公共功能独立为服务

one day

2018-08-09

我公司正巧在重构系统。定期3个月。电商系统。原来是dubbo 服务间调用 ，redis 缓存，mysql，模块化开发，云平台,svn。问题，关联查询太多，dubbo部署的相关模块太多，业务复杂度太高，业务调用同步慢，一搞活动，数据库就爆，业务逻辑懂的人换了一茬又一茬。目标spring cloud,docker,k8s,git,禁止关联查询即单表，解藕慢操作即异步，接口细粒度，层次调用分明，服务隔离能缓存就缓存，业务大拆解。最终会是什么样，静待。学了老师这么多，手术台上实习了，哈哈. 缓存，异步，事务消息最终一致，强制单表查询，业务边界，规范。牵一发动全身，业务等待，加班加班.....

作者回复

2018-08-10

感觉这样还是没有有的放矢呢，重构不是什么问题都解决。
例如关联查询，电商业务是不可避免的，禁止关联查询，业务改动非常大，如果是关联查询导致性能问题，首先应该分析一下慢查询，80%的慢查询集中在20%的语句上。
例如慢操作改为异步，实现起来没那么容易，有的就是要同步调用，而有的可以用消息队列异步通知异步处理。

三木子

2018-08-09

可惜，现在做的体统很小，不给机会重构啊。

作者回复

2018-08-10

那就先做大��

空档滑行

2018-08-09

当前的系统刚刚做过一次重构，当时选择重构的原因有这么几个，原系统有一个大的面向客户的web系统和多个单独的进程组成，前后端在一个war包，后端服务单机远行，两者通过db交换数据。问题有这么几个，web系统累积了5年的代码，改个bug已经随时搞挂整个web端。后端服务单点问题。数据库压力很大，一个功能的sql拖垮整个库。
重构的思路是，先将单体后端进程服务化，服务化过程中非核心数据从主表分离出来。web系统结构不动，上线一个服务就从原来的查db改成调用服务。服务梳理差不多了将web端重新实现前后端分离。做完这些后将数据库结构重新设计。总的来说跟文中的第三个例子有点像。

作者回复

2018-08-09

思路很清晰����

JustDoIt

2018-08-09

刚刚经历过"狗血"的重构，也是新官上任，把我们持久层有jpa换成mybatis,老板不懂技术，同意重构，新来的人问我的意见，我反对只是换持久层，感觉二者各有优劣，只是换持久层劳民伤财，新官听不进，后面搞了一个月多月，草草收场，上线了发现问题比之前更多，请问，真有必要把jpa换mybatis吗？

作者回复

2018-08-09

这类协议的选择，一般建议哪个熟悉用哪个，不熟悉的方案，别人用的再好，自己用也是一堆问题��

吴科��

2018-08-09

系统优化，我们公司就是一个字，拆，拆数据库，拆服务，没有依赖就没有伤害