20 | 从务实的角度，给你架构设计的重点知识和学习路径

2020-04-06 王庆友

架构实战案例解析 进入课程 >

讲述：王庆友
时长 15:01 大小 13.76M

你好，我是王庆友。

到目前为止，我们已经讲完了业务架构和技术架构的相关内容，相信你现在对架构有了更深入的理解。

学习架构呢，要掌握的东西有很多，你是不是开始担心自己一辈子都学不完呢？其实，我们也不需要一下子铺开学习所有的架构技能，重要的是把控好学习的节奏，在适当的时间学习适当的内容，我们可以结合实际工作，一步步地成长。所以今天这一讲，我想给你提供架构学习的重点方向和路径建议。

架构原则汇总

在技术架构篇，我针对系统的高可用、高性能、可伸缩和低成本，给你介绍了很多的架构设计原则，不同的原则对应着不同的目标，这里我把这些架构原则和目标汇总成一个表格，来帮助你更直观地了解它们。

技术架构原则	高可用	可伸缩	低成本
冗余无单点	✓	✓	✓
无状态	✓	✓	✓
可异步处理	✓	✓	✓
可缓存		✓	✓
可水平拆分		✓	✓
计算可并行		✓	✓
短事务/柔性事务	✓	✓	
分级与降级	✓	✓	✓
可熔断/可限流	✓		
可回滚/可禁用	✓		
应用与数据分开	✓	✓	✓
可多机房部署	✓	✓	✓
可复制	✓	✓	
可监控	✓	✓	✓
可虚拟化部署		✓	✓
使用成熟技术	✓		✓
使用同质化硬件	✓	✓	✓

限于篇幅，这里我挑选几个原则来重点说下：

可回滚 / 可禁用

可回滚原则确保了系统可以向后兼容，当系统升级出现问题的时候，我们可以回滚到旧版本，保证系统始终可用。

不过有些时候，系统回滚很困难。举个例子，如果数据库的新旧表结构差异很大，除了回滚代码，我们还要回滚数据库，这样操作起来往往需要很长时间，系统的可回滚性就比较差。所以在设计时，我们要尽量考虑数据库修改和代码的兼容性，并提前做好系统回滚的预案。

可禁用原则要求我们提供功能是否可用的配置，在系统出现故障时，我们能够快速下线相应的功能。比如说，新的商品推荐算法有问题，我们可以通过程序开关禁用这个功能。

使用成熟技术

作为开发人员，我们都很想尝试新技术，但我们知道，新的技术还没有经过充分的验证，它往往会存在很多隐藏的 Bug。

所以，作为架构师，我们在做方案设计的时候，一方面，要从系统的稳定性出发，尽量选择成熟的技术，避免因为新技术的坑而导致系统可用性出现问题；另一方面，选择成熟的技术也意味着选择了团队熟悉的技术，这样学习成本低，落地快。

使用同质化硬件

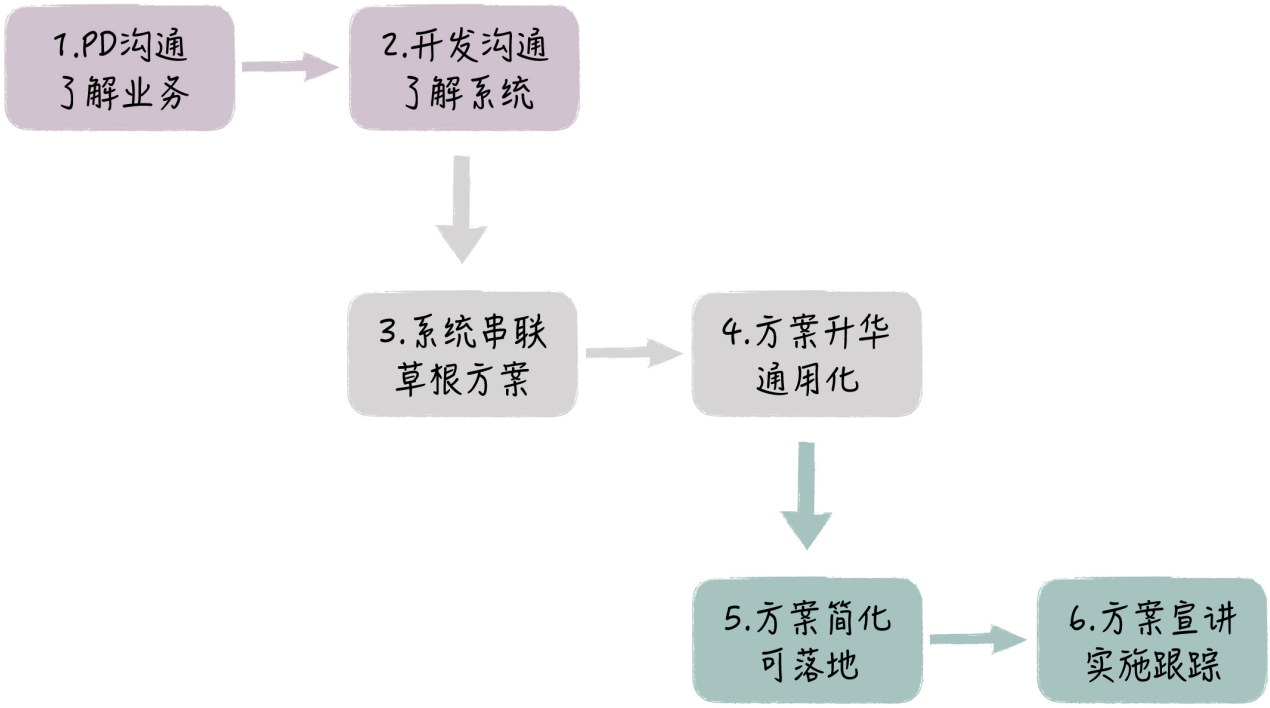
我们在做硬件选型的时候，要尽量选择相同的硬件和相同的配置。

比如说，对于服务器，我们选择同样的 CPU 和内存配置，以及同样的操作系统版本，这样我们更容易通过统一的自动化脚本，对节点进行配置，对系统做水平扩展时也会更加容易。

架构落地过程

这些架构原则都是我们要深入理解，并且在实践中要逐渐运用和掌握的。那么下面，我就带你来了解一下架构的具体落地过程，帮助你更好地理解架构师的职责和技能要求。

简单地说，架构师的职责就是负责设计架构，并跟踪架构的实施过程，解决过程中出现的疑难问题，确保架构顺利落地。在第 1 讲“架构的本质”中，我和你介绍过架构师的能力模型，比如抽象思维、平衡取舍、沟通能力等等。接下来，我就结合架构的落地过程和架构师的能力模型，来具体说下架构师是如何开展工作的。



架构师的工作从接到项目需求，或者从自己主动识别系统当前的问题开始，TA 的工作过程可以分为三个大阶段。

首先，架构师要和产品经理或者业务人员沟通，了解业务；和开发人员沟通，了解系统。

了解完系统和业务后，架构师接下来就要设计具体的方案，方案设计要分三步走：

> 首先，架构师针对业务需求，分解相应功能到现有的各个系统，把系统的各个部分串起来，这个第一版的方案至少要能够在表面上解决当前的问题，这样就形成一个草根的方案。
>
> 然后，架构师要进一步深入思考业务的本质，对现有的草根方案进行升华，比如说，通过抽象，让方案更加通用，可以解决多个类似的或潜在的业务需求，这样，草根的方案就变成了一个高大上的方案，这里很考验架构师的**透过问题看本质**和**抽象总结**的能力，
>
> 接下来，基于现有的各项约束，比如时间、资金和人员技术能力等因素，架构师要对方案进行简化，把高大上的方案变成一个接地气的方案，以最小的代价实现最大的价值，这里很考验架构师的**平衡取舍能力**。

方案设计好之后，最后还要进行**宣讲**，架构师需要说服相关的人员接受方案，并且在后续的方案执行中，负责跟踪架构的落地，如果过程中有疑难问题，架构师还要协助解决。

所以，我们可以看到，架构师在设计方案时，会有一个反复迭代的过程，最终才能得到一个简约而不简单的方案。并且在方案设计的前后，架构师还需要和大量的人员进行沟通，这些都需要架构师具备宽广的知识面和良好的沟通能力。

架构师知识结构

那么，架构师都需要掌握哪些具体的技能呢？这里我给你提供了一个简化的架构师技能图谱，可以帮助你循序渐进地学习这些架构技能。

首先，作为架构师，我们需要了解**计算机硬件和操作系统**的相关知识，它们是负责具体干活的，如果对它们有深入的了解，我们就能知道系统底层是怎么执行的，在做具体设计的时候，我们也就可以做各种优化。比如说，在设计 RPC 通讯框架时，我们可以通过 IO 多路复用和内存零拷贝技术，来提升服务端并发处理请求的能力。

在这之上就是**具体技术**相关的内容，从浅到深可以分为三个部分：

> 第一部分是**开发相关的基本知识**，比如数据结构和算法、具体的开发语言、常用的设计模式以及开发框架等等，这样你就具备了基本的开发能力。

第二部分是**各种中间件知识**，常用的中间件包括数据库、缓存、消息系统、微服务框架等等，对于这些核心中间件，我们不但要了解具体的用法，还要深入理解它们的适用场景。这样你就能写出高效健壮的代码，能够独立承担一个子系统的开发。

继续往下深入，你还要学习**分布式系统相关的知识**，包括底层网络和分布式通信技术，这样你就可以了解系统是怎么连接在一起的。除此之外，你还要了解一些周边的系统，比如大数据平台、运维监控系统、接入系统等等，这样，你就可以了解系统端到端的运行过程，从技术架构上保证系统的稳定可用。

掌握了这些技术能力之后，你就可以逐渐往全面的架构师发展了。比如说，你可以结合业务，来设计应用体系，包括数据模型和服务设计；你可以了解各种应用架构模型，知道它们的优缺点和适用场景，能够定义一个良好的应用依赖关系。

再往上，就是成为业务领域专家。在这个阶段，你已经知道如何通过业务拆分，实现业务之间的解耦；如何通过业务抽象，实现业务的扩展和重用。

到最后，你已经对各种架构设计的目标和架构原则都非常了解了，知道面对一个具体的问题，大致都有哪些解决的手段；然后，经过大量的实践，你能够把技术架构、应用架构、业务架构融会贯通，并针对具体情况，对架构的各个目标做良好的平衡。当然，作为架构师，你还要和一系列的人员打交道，这时候就需要你培养更多的**软技能**，能把复杂的架构问题以简单的方式表达出来。

架构师成长路径

现在，你已经清楚了作为一个架构师，TA 需要具备什么样的知识结构。如果你想成为一名架构师，在不同的成长阶段，你还需要学习不同的内容。这里，我以 Java 为例，进一步给出学习的重点内容，给你提供更具体的参考。

第一个阶段是**初级开发阶段**。

在这个阶段，你需要深入学习数据结构和算法，并且一定要深入掌握单体应用的分层架构，因为这是架构设计的基础。

另外，对 JDK 的一些核心类，你不能仅仅停留在使用层面，而是要深入研读源代码，了解它的内部设计。这样你就知道如何开发一个高效的程序，如何进行各种代码级的调优。

第二个阶段是**高级开发阶段**。

首先，你需要非常了解**设计模式**，每个设计模式都可以看做是一个小型的架构设计，这里面有很好的设计原则和抽象思维，你在做系统设计时可以借鉴它们。

然后，你需要非常了解**核心的中间件**，包括 DB、微服务框架、缓存和消息系统，要清楚地了解它们的适用场景（比如消息系统的削峰、解耦和异步），知道如何对它们进行调优，以及了解它们都有哪些常见的坑等等，核心中间件是我们做技术选型的基础。

同时，你要深入掌握**数据库设计和服务接口设计**，了解它们的最佳设计实践，它们承载了系统核心的业务数据和业务逻辑。

最后，你需要进一步**研读源码**，源码是活的教材，它包含了大量实用的设计原则和技巧。这里我建议你选择一些开源的开发框架和 RPC 通信框架，去深入了解它们内部的实现原理，比如 Spring 和 Netty。

第三个阶段是**架构师阶段**，成为技术专家。

首先，你需要深入了解网络通信，比如说网络分层和 HTTP/TCP 协议，还有各种常见的 RPC 通讯框架，了解它们的特性和适用场景，这样你在设计分布式系统时，就能够进行合理的技术选型。

然后是了解底层系统，包括 JVM、操作系统和硬件原理，再往上延伸到系统的接入部分，了解常见的负载均衡特性和用法，这样你可以对整体的系统有个透彻的了解，把各个环节可以很好地衔接起来。这里，我特别建议你去读下 Java 和 JVM 的规格说明书，了解 Java 的底层设计。

最后，你需要熟练掌握各种设计工具和方法论，比如领域驱动设计和 UML，了解常用的架构设计原则，这样你就能够结合业务，选择合适的应用架构和技术架构并进行落地。在这一阶段，对你总的要求就是能够从**端到端的角度**进行业务分析和系统设计。

第四阶段是**大师阶段**。

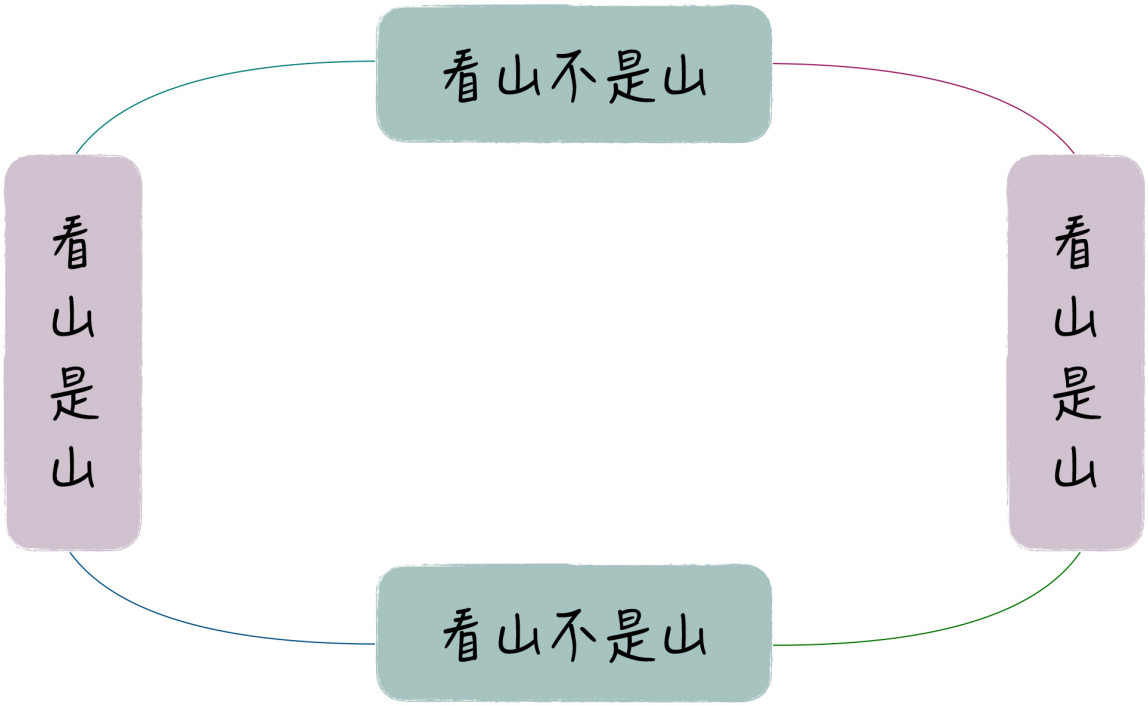
在这个阶段，你需要对架构的各个目标都非常了解，除了业务系统设计，你还要对运维和监控有深入的认知。同时，你需要了解业界的架构实践，跟踪技术的发展趋势，如果出来一项新技术，你可以比较准确地对它进行定位，把它纳入到自己的能力体系当中。

另外，在这个阶段，你也已经通过大量的实践，培养了很好的软技能，比如沟通能力、项目管理能力等等。那么最后，你就能做到技术和业务的融会贯通，可以平衡各种架构目标，设计非常实用和接地气的架构，并保障它的顺利落地。

架构师境界

你可以发现，架构师的能力是一个逐渐提升的过程，如果从架构师的境界来看，由浅到深可以分为四层：第一层看山不是山，第二层看山是山，第三层看山不是山，第四层看山是山。

这是一个螺旋式上升的过程，那么它究竟是什么意思呢？



刚接手项目的时候，你对业务还不太了解，经常会被业务方冒出的术语弄得一愣一愣的，如果把现有问题比作山，那就是横看成岭侧成峰，你根本摸不透，此时**看山不是山**。

经过业务梳理和深入了解系统以后，你能够设计出一个简单的方案，把各个系统串起来，能解决当前的问题，对当前的这个“山”能够看清楚全貌，此时就做到了**看山是山**。但这样的方案往往设计不够，只能解决表面问题，碰到其它类似问题或者问题稍微变形，系统还需要重新开发。

通过进一步抽象，你能够发现问题的本质，明白了原来这个问题是共性的，后续还会有很多类似的问题。然后你就对设计进行总结和升华，得到一个通用的方案，它不光能解决当前的问题，还可以解决潜在的问题。此时，你看到的已经是问题的本质，**看山不是山**。但这样的方案往往会过度设计，太追求通用化，会创造出过多的抽象概念，理解和实现起来都特别困难，过犹不及。

最后回到问题本身，你能够去除过度的抽象，给出的设计简洁明了，增之一分嫌肥，减之一分嫌瘦，既能解决当前的问题，又保留了一定的扩展能力，此时问题还是那个问

题，**山还是那个山**。这样的方案在了解问题本质的基础上，同时考虑到了现状，评估了未来，不多做，不少做。

你可以对照这四个境界，来评估你当前的架构能力，不断地提升对自己的要求。

总结

今天，我汇总了常见的技术架构设计原则，它们都是实践的总结，你在做架构设计时，可以参考这些原则，在项目中采取相应的手段来实现架构目标。值得注意的是，在做具体的架构设计时，你需要对设计进行反复迭代，才能最终得到一个高性价比的方案。

针对架构师的成长，我也给你提供了相应的知识结构和可行的进阶之路，希望你能够一步步成长，最终实现自己的理想。

读万卷书，行万里路。架构师的成长尤其如此，架构没有速成之路，我们先要“读万卷书”，学习各种架构需要的技能，然后“行万里路”，通过大量的实践把架构知识变成架构能力。

最后，给你留一道思考题：一个架构方案从调研到设计再到落地，你觉得最困难的地方是什么？

欢迎在留言区和我互动，我会第一时间给你反馈。如果觉得有收获，也欢迎你把这篇文章分享给你的朋友。感谢阅读，我们下期再见。

上一篇　19 | 综合案例：电商平台技术架构是如何演变的?

下一篇　结束语 | 和你聊聊我的架构心路历程

精选留言 (9)

写留言

AlfredLover
2020-04-07

这课程值得反复阅读

展开

作者回复: 有点干，大家多稀释几遍。

1

mickey
2020-04-08

我觉得最困难的是对业务的现有模式的真正理解和今后的业务的扩展的预见判断。

Geek_kevin
2020-04-07

我就感觉难点应该是沟通问题,如何说服大家认可你的架构方案？如何以低成本的去验证？还有就是确保开发人员真正理解并严格执行架构落地。

展开

每天晒白牙
2020-04-07

架构师之路还有好多路要走

展开

嬰
2020-04-07

最困难的是沟通吧，让别人能够接受这个方案而且还能做的好是挺难的。尤其是涉及到多部门合作，还要协调排期优先级等等。

展开

深山小书童
2020-04-06

读完了整体来说受益匪浅，谢谢老师。总体来说业务架构写的比技术架构好很多，也许是业务架构上的文章在网上比较少见，技术架构的文章太司空见惯。

展开

Jxin
2020-04-06

最难的应该不是这三步。而是落地后持续的演进。毕竟从一个项目长远的角度看，比如说5年10年。持续维护架构健康，并非一人之力可以达成。需要的是团队具备这种持续优化演进架构的共识，并把这个共识一直传承下去。

展开

夜空中最亮的星（华仔...
2020-04-06

学无止境啊

展开

小洛

2020-04-06

从端到端的完整业务去看问题，去做架构设计，但是往往有盲点，考虑不周，在落地的时候有可能发现设计不可行

展开