

自成一派的架构设计方法论
教你体系化的架构设计技能

Broadview[®]
www.broadview.com.cn



从零开始学架构

照着做，你也能成为架构师

李运华 | 著

 中国工信出版集团

 电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

从零开始学架构

照着做，你也能成为架构师

李运华 | 著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

架构设计是技术人员成长和晋升过程中必须掌握的技能，但目前业界缺乏架构师学习和培养方面体系化的知识和实践的指导，本书结合作者多年在架构设计方面的学习、思考、实践，提出了完整的一套架构设计方法论，包括什么是架构、架构设计的目的、架构设计原则、架构设计流程、架构设计模式和技巧、互联网公司技术演进等内容。这套架构设计方法论适合不同行业，比如互联网、企业应用等；也适合不同的技术领域，比如后端架构设计、前端架构设计、客户端架构设计、测试平台架构设计、运维平台架构设计等。

本书由浅入深地阐述了架构设计的相关内容，比较适合以下类型的读者：

- 没有架构设计经验，但对架构设计非常有兴趣，希望学习架构设计技术，提升技术能力，成为“大场面霸”的读者；
- 已经尝试了一些架构设计，但挖了各种“坑”或踩了各种“坑”，希望知道“为什么”的技术人员；
- 具备一定的架构设计经验，想进一步系统化地提升架构设计能力，成为令人羡慕的“高级技术专家”“资深技术专家”的读者。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

从零开始学架构：照着做，你也能成为架构师 / 李运华著. —北京：电子工业出版社，2018.9
ISBN 978-7-121-34791-7

I. ①从… II. ①李… III. ①计算机系统 IV. ①TP30

中国版本图书馆 CIP 数据核字（2018）第 167768 号

责任编辑：陈晓猛

印 刷：三河市良远印务有限公司

装 订：三河市良远印务有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱

邮编：100036

开 本：787×980 1/16 印张：20.25

字数：388.8 千字

版 次：2018 年 9 月第 1 版

印 次：2018 年 11 月第 3 次印刷

定 价：99.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zltts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

序

“业界 A 公司的架构是 X，B 公司的方案是 Y，两个差别比较大，该参考哪一个呢？”

“架构设计要考虑高性能、高可用、高扩展……这么多高 XX，全部设计完成估计要 1 个月，但老大只给了 1 周！怎么办？”

“淘宝的架构是这么做的，我们也要这么做！”

“Docker 和 Kubernetes 现在很流行，我们的架构应该充分应用进来！”

这些对不对？怎样做才是对的？架构师往往挣扎于诸如此类的问题之中，在理性和非理性之间举棋不定、焦虑彷徨。

互联网发展到今天，软件系统早就不是一个万行代码加上一台服务器这样“过家家”的游戏。BAT 的服务器规模已经达到甚至超过百万级。传统企业向互联网的靠拢，势不可挡。

优秀的软件系统架构师就像大海航船舵手，指引着软件前进的方向，让企业在激烈的竞争中拔得头筹（而不是拖后腿）的同时，在企业内部尊享荣光。

只有兼具技术的深度和广度，并能克服人性弱点的资深 IT 从业者，才有机会成为一个优秀的架构师（高薪也是水到渠成的事情）。优秀的架构师是公司的福音，反之就是公司的灾难。如果坏的种子已经埋下，那么“爆雷”是一定的，何时“爆”反而不确定。

架构即人性，不切实际、追求繁杂、好大喜功的非理性的个人诉求，掺杂于甚至暗暗地主宰着整个理性的软件系统架构设计，这将置企业于危险境地！如果成为“人人喊打”的架构师，岂不叹哉！

那么，怎么扎实地成为一名优秀的、人见人爱的架构师？

毫无疑问，本书将给你一个有良心的参考答案——如果你也饱受似懂非懂、雾里看花、管中窥豹及盲人摸象之苦。

本书指出，软件架构是指软件系统的顶层结构，而架构设计的目的是为了解决软件复杂度，

并给出了架构设计的三原则及实现方法，高性能、高可用、可扩展架构的各种模式及实战案例。

有了架构设计的指导思想——“解决软件复杂度”，很多抓心挠肺的问题立刻迎刃而解。

“业界 A 公司的架构是 X，B 公司的方案是 Y，两个差别比较大，该参考哪一个呢？”——理解每个架构方案背后所需要解决的复杂点，然后才能对比自己的业务复杂点，参考复杂点相似的方案。

“架构设计要考虑高性能、高可用、高扩展……这么多高 XX，全部设计完成估计要 1 个月，但老大只给了 1 周时间。”——架构设计并不是要面面俱到，不需要每个架构都具备高性能、高可用、高扩展等特点，而是要识别出复杂点，然后有针对性地解决问题。

“淘宝的架构是这么做的，我们也要这么做！”——淘宝的架构是为了解决淘宝业务的复杂度而设计的，淘宝的业务复杂度并不就是我们的业务复杂度，绝大多数业务的用户量都不可能像淘宝那么大。

“Docker 和 Kubernetes 现在很流行，我们的架构应该充分应用进来！”——Docker 及其编排工具 Kubernetes 不是万能的，只是为了解决资源重用和动态分配而设计的，如果我们的系统复杂度根本不在这方面，引入 Docker 没有什么意义。

本书提出了令人耳目一新的架构设计三原则，即合适原则——合适优于业界领先；简单原则——简单优于复杂；演化原则——演化优于一步到位。这些原则让架构的选型及设计，化繁为简，极具指导性和操作性。

架构从来不是一成不变的，架构也没有一劳永逸。Facebook 每隔 3~5 年就会重新设计一次架构，就像凤凰重生。诚如《三体》所传达的思想，与其维护一个旧世界，不如开创一个新世界。

架构即人性，设计一个符合企业当前情况，又可以演进、不好大喜功的架构，善莫大焉。怎么做？请参阅本书。

我和运华兄从 2016 年认识以来，合作颇多，并且受益良多。特别是 2016 年下半年开始，在中国信息通信研究院的指导下，云计算开源产业联盟、高效运维社区及 DevOps 时代社区联合国内各大 BAT 及传统企业顶级专家，开始编写国内外第一个 DevOps 标准（研发运营一体化能力成熟度模型），并请运华兄作为其中“应用架构及设计”模块的核心编写专家，运华兄的专业和认真，使得 DevOps 标准增色很多。

可喜的是，DevOps 标准于 2018 年 6 月在工信部正式立项成功，并于 2018 年 7 月在联合国 ITU（世界上最早的国际标准化组织）立项成功，这也成为我国在 IT 领域首次对外输出国际标

准（不再是 CMMI、ITIL、IOS 系列这样的标准的被动接受方）。军功章里，也有很多运华兄的功劳。

所以，欣闻运华兄集多年功力而成的大作出版，真是非常开心，相信一定能惠及更多 IT 从业者。

话说回来，不仅研发同学，每个运维同学都应该学一些架构。成长为一名优秀的架构师，或许也是运维人员延续并拓展其职业生涯的一个好机会。毕竟，架构设计的核心是通过高性能、高可用和高扩展，让软件系统更具有可运维性。

运维的架构师之路，可以考虑从本书开始。

萧田国 DevOps 国际标准发起人 高效运维社区发起人

前言

为什么写这本书

每个程序员心中都有一个成为架构师的梦想，梦想是美好的，但道路是曲折的。

我在 2006 年开始参与架构设计，原本以为学习架构设计就像学习一门编程语言一样，先学习基本的语法，再研究细节和原理，然后实践一下就能够快速掌握。但真正实践后才发现，架构设计的难度和复杂度要高很多。从最早开始接触架构设计，到自我感觉初步完整掌握架构设计，至少花费了 6 年时间。等到自我感觉彻底掌握架构设计的精髓，至少花费了 8 年时间（当然，在这个过程中我不是一直在做架构设计）。

我曾经以为是自己天资愚笨才会这样，后来我带了团队，看到几乎每个程序员在尝试架构设计的时候，都面临着我遇到过的各种困惑和瓶颈。特别是我作为职业等级晋升评委的时候，发现很多同学技术能力很强，业务也不错，但却卡在了架构设计这部分。我意识到这应该不是个人天资的问题，而是架构设计本身的一些特性导致的。

我总结了几个架构设计相关的特性。

1. 架构设计的思维和程序设计的思维差异很大。

架构设计的关键思维是**判断和取舍**，程序设计的关键思维是**逻辑和实现**。很多程序员在转变为架构师后，很难一开始就意识到这个差异，还是按照写代码的方式去思考架构，这样会导致很多困惑。

2. 架构设计没有体系化的培训和训练机制。

大学的课程几乎没有架构设计相关的课程，架构设计的书籍更多也只是关注某个架构设计点，没有体系化的架构设计书籍，导致程序员在学习上没有明确的指导，只能自己慢慢摸索，效率低，容易踩坑。

3. 程序员对架构设计的理解存在很多误区。

例如，要成为架构师必须要有很强的技术天分；架构师必须要有很强的创造力；架构设计必须要高大上才能体现架构师能力；架构一定要具备高可用、高性能……这些似是而非的误区让

很多技术人员望而生畏，还没尝试就已经放弃了。

得益于移动互联网技术的快速发展，我有很多机会直接参与架构设计，这些架构背后的业务形形色色，包括社交、电商、游戏、中间件、内部运营系统；用到的技术栈差异也比较大，包括 PHP、Java、C++等。虽然每次架构设计对我来说都是一个新的挑战，但正好也提供了非常好的机会，让我亲身体验不同的架构设计。在这个过程中，我不断学习、思考、实践、总结、改进、交流，逐步形成了自己的一套**架构设计方法论**。

有了这套方法论后，首先，我在做架构设计的时候游刃有余，不管什么样的业务，不管什么样的技术，按照这套方法论都能够设计出优秀的架构，在职业等级面评的时候，就算我之前从来没有接触过对方的业务，也能快速理解对方描述的架构和发现其中做得好或不好的地方；其次，在指导其他同事的时候思路很清晰，容易理解，效果明显。原来对架构设计比较迷茫的同学，通过几次结合案例进行方法论培训，都能够很快地掌握这套方法论并在实践中应用。甚至有很多其他业务线的同学，遇到架构设计的困惑，也来找我交流和指导，按照这套架构设计方法论的指导，能够较快地理清架构设计的思路。

本书的主要出发点就是将这套架构设计方法论分享给更多热爱技术、有架构师梦想的技术人员，降低架构学习的成本，减少架构学习过程中走的弯路，助力大家更快地实现自己的架构师梦想。

本书内容已经在“极客时间”App 上开设了“从 0 开始学架构”的专栏，订阅人数已经超过 25000 人，成为“极客时间”最受欢迎的专栏，能够得到这么多技术朋友的信任，相信书中的内容一定会让你有所收获。

本书的主要内容

本书涵盖了整套架构设计方法论和架构实践，主要包括以下内容。

- **架构基础**：先介绍架构设计的本质、历史背景和目的，然后从复杂度来源，以及架构设计的原则和流程来详细介绍架构基础。
- **高性能架构模式**：从存储高性能、计算高性能方面介绍几种设计方案的典型特征和应用场景。
- **高可用架构模式**：介绍 CAP 原理、FMEA 分析方法，分析常见的高可用存储架构和高可用计算架构，并给出一些设计方法和技巧。
- **可扩展架构模式**：介绍可扩展模式及其基本思想，分析一些常见架构模式。
- **架构实战**：将理论和案例结合，落地前面提到的架构原则、架构流程和架构模式。

本书适合的对象

- 有一定的编程基础的软件开发工程师。

- 对架构设计有兴趣的技术人员。例如，测试、运维等岗位的人员。
- 有初步的架构设计经验，但需要继续提升的技术人员。

勘误与支持

因个人水平有限，且架构设计整体涵盖的技术范围很广，技术深度很深，书中难免有不足之处，还望读者批评指正。如果你对本书有比较好的建议或对书中内容有所疑惑，可与我联系。

Email: yunhua_lee@163.com

致谢

首先感谢王行云、胡晏秋、陈俊良、张怡炘等同事对本书的勘误和审核，帮助完善了本书的很多细节和内容。

其次感谢家人的支持，在写书的过程中父母、妻子承担了家庭的重任，让我能够安心写作。

特别感谢陈晓猛编辑，本书在他不断督促下才最终写完初稿，后期他耐心地指导、审稿、修改，最终才有了本书的诞生。

特别感谢极客时间架构专栏团队郭蕾、何潇、周君凤等负责人，打造了一个非常成功的架构专栏，他们的高要求也让整体内容更加完善、更加优质。

特别感谢高效运维创始人萧田国、特赞科技 CTO 黄勇、腾讯云高级总监熊普江、贝壳金服 2B2C CTO 史海峰、资深技术专家于君泽（右军）、21CTO 社区创始人杜江（洛逸）几位专家对本书的推荐。

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **提交勘误：**您对书中内容的修改意见可在 [提交勘误](#) 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动：**在页面下方 [读者评论](#) 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/34791>



目录

第 1 部分 概念和基础

第 1 章 架构基础	2
1.1 “架构”到底指什么	2
1.1.1 系统与子系统	3
1.1.2 模块与组件	4
1.1.3 框架与架构	5
1.1.4 重新定义架构	7
1.2 架构设计的目的	7
1.2.1 架构设计的误区	7
1.2.2 以史为鉴	9
1.2.3 架构设计的真正目的	13
1.3 复杂度来源	15
1.3.1 高性能	15
1.3.2 高可用	22
1.3.3 可扩展性	28
1.3.4 低成本	31
1.3.5 安全	33
1.3.6 规模	35
1.4 本章小结	36
第 2 章 架构设计原则	38
2.1 合适原则	39

2.2	简单原则	40
2.3	演化原则	44
2.4	本章小结	46
第 3 章	架构设计流程	47
3.1	有的放矢——识别复杂度	47
3.2	按图索骥——设计备选方案	49
3.3	深思熟虑——评估和选择备选方案	51
3.3.1	业务背景	54
3.3.2	备选方案设计	54
3.3.3	备选方案 360 度环评	56
3.4	精雕细琢——详细方案设计	58
3.5	本章小结	59

第 2 部分 高性能架构模式

第 4 章	存储高性能	62
4.1	关系数据库	62
4.1.1	读写分离	62
4.1.2	分库分表	64
4.1.3	实现方法	70
4.2	NoSQL	73
4.2.1	K-V 存储	74
4.2.2	文档数据库	75
4.2.3	列式数据库	79
4.2.4	全文搜索引擎	80
4.3	缓存	84
4.3.1	缓存穿透	85
4.3.2	缓存雪崩	86
4.3.3	缓存热点	87
4.4	本章小结	87

第 5 章 计算高性能	89
5.1 单服务器高性能	89
5.1.1 PPC	90
5.1.2 prefork	91
5.1.3 TPC	92
5.1.4 prethread	93
5.1.5 Reactor	94
5.1.6 Proactor	100
5.2 集群高性能	101
5.2.1 负载均衡分类	101
5.2.2 负载均衡架构	104
5.2.3 负载均衡的算法	105
5.3 本章小结	108

第 3 部分 高可用架构模式

第 6 章 CAP	112
6.1 CAP 理论	113
6.1.1 一致性 (Consistency)	114
6.1.2 可用性	115
6.1.3 分区容忍性 (Partition Tolerance)	116
6.2 CAP 应用	117
6.2.1 CP——Consistency/Partition Tolerance	117
6.2.2 AP——Availability/Partition Tolerance	117
6.3 CAP 细节	118
6.4 ACID、BASE	120
6.4.1 ACID	120
6.4.2 BASE	121
6.5 本章小结	122
第 7 章 FMEA	124
7.1 FMEA 介绍	124

7.2	FMEA 方法	125
7.3	FMEA 实战	129
7.4	本章小结	131
第 8 章	存储高可用	132
8.1	主备复制	132
8.1.1	基本实现	132
8.1.2	优缺点分析	133
8.2	主从复制	134
8.2.1	基本实现	134
8.2.2	优缺点分析	135
8.3	主备倒换与主从倒换	136
8.3.1	设计关键	136
8.3.2	常见架构	137
8.4	主主复制	141
8.5	数据集群	142
8.5.1	数据集中集群	143
8.5.2	数据分散集群	144
8.5.3	分布式事务算法	146
8.5.4	分布式一致性算法	149
8.6	数据分区	152
8.6.1	数据量	152
8.6.2	分区规则	153
8.6.3	复制规则	153
8.7	本章小结	155
第 9 章	计算高可用	156
9.1	主备	157
9.2	主从	158
9.3	对称集群	159
9.4	非对称集群	161
9.5	本章小结	162

第 10 章 业务高可用	163
10.1 异地多活	163
10.1.1 异地多活架构	164
10.1.2 异地多活设计技巧	167
10.1.3 异地多活设计步骤	173
10.2 接口级的故障应对方案	179
10.2.1 降级	180
10.2.2 熔断	181
10.2.3 限流	181
10.2.4 排队	183
10.3 本章小结	184
第 4 部分 可扩展架构模式	
第 11 章 可扩展模式	186
11.1 可扩展概述	186
11.2 可扩展的基本思想	187
11.3 可扩展方式	189
11.4 本章小结	190
第 12 章 分层架构	192
12.1 分层架构类型	192
12.2 分层架构详解	194
12.3 本章小结	198
第 13 章 SOA 架构	199
13.1 SOA 历史	199
13.2 SOA 详解	200
13.3 本章小结	202
第 14 章 微服务	203
14.1 微服务历史	203

14.2	微服务与 SOA 的关系	204
14.3	微服务的陷阱	206
14.4	微服务最佳实践	209
14.4.1	服务粒度	209
14.4.2	拆分方法	210
14.4.3	基础设施	212
14.5	本章小结	221
第 15 章	微内核架构	222
15.1	基本概念	222
15.2	设计关键点	223
15.3	OSGi 架构简析	224
15.4	规则引擎架构简析	226
15.5	本章小结	229

第 5 部分 架构实战

第 16 章	消息队列设计实战	232
16.1	需求	232
16.2	设计流程	233
16.2.1	识别复杂度	233
16.2.2	设计备选方案	234
16.2.3	评估和选择备选方案	236
16.2.4	细化方案	239
16.3	本章小结	240
第 17 章	互联网架构演进	241
17.1	技术演进	241
17.1.1	技术演进的动力	241
17.1.2	淘宝	246
17.1.3	手机 QQ	250
17.1.4	微信	253

17.2	技术演进的模式	255
17.3	互联网业务发展	256
17.3.1	业务复杂性	257
17.3.2	用户规模	261
17.3.3	量变到质变	261
17.4	本章小结	262
第 18 章	互联网架构模板	264
18.1	总体结构	264
18.2	存储层技术	265
18.2.1	SQL	265
18.2.2	NoSQL	266
18.2.3	小文件存储	267
18.2.4	大文件存储	267
18.3	开发层技术	268
18.3.1	开发框架	268
18.3.2	Web 服务器	269
18.3.3	容器	269
18.4	服务层技术	270
18.4.1	配置中心	270
18.4.2	服务中心	271
18.4.3	消息队列	273
18.5	网络层技术	275
18.5.1	负载均衡	275
18.5.2	CDN	277
18.5.3	多机房	278
18.5.4	多中心	279
18.6	用户层技术	279
18.6.1	用户管理	279
18.6.2	消息推送	280
18.6.3	存储云与图片云	281
18.7	业务层技术	282

18.8	平台技术	283
18.8.1	运维平台	283
18.8.2	测试平台	285
18.8.3	数据平台	287
18.8.4	管理平台	288
18.9	本章小结	289
第 19 章	架构重构	290
19.1	有的放矢	291
19.2	合纵连横	295
19.2.1	合纵	295
19.2.2	连横	296
19.3	运筹帷幄	297
19.4	文武双全——项目管理+技术能力	301
19.5	本章小结	302
第 20 章	开源系统	303
20.1	选：如何选择一个开源项目	304
20.1.1	聚焦是否满足业务	304
20.1.2	聚焦是否成熟	304
20.1.3	聚焦运维能力	305
20.2	用：如何使用开源方案	305
20.2.1	深入研究，仔细测试	305
20.2.2	小心应用，灰度发布	306
20.2.3	做好应急，以防万一	306
20.3	改：如何基于开源项目做二次开发	307
20.3.1	保持纯洁，加以包装	307
20.3.2	发明你要的轮子	307
20.4	本章小结	308