83 | 程序员练级攻略：分布式架构工程设计

2018-7-17 陈皓，杨爽

要学好分布式架构，你首先需要学习一些架构指导性的文章和方法论，即分布式架构设计原则。下面是几篇很不错的文章，值得一读。

Designs, Lessons and Advice from Building Large Distributed Systems，Google 杰夫·迪恩（Jeff Dean）2009年一次演讲的PPT。2010年，斯坦福大学请杰夫·迪恩到大学里给他们讲了一节课，你可以在YouTube上看一下，Building Software Systems At Google and Lessons Learned ，其回顾了Google发展的历史。

The Twelve-Factor App ，如今，软件通常会作为一种服务来交付，它们被称为网络应用程序，或软件即服务（SaaS）。12-Factor 为构建SaaS应用提供了方法论，是架构师必读的文章。（中译版）这篇文章在业内的影响力很大，必读！

Notes on Distributed Systems for Young Bloods ，给准备进入分布式系统领域的人的一些忠告。

On Designing and Deploying Internet-Scale Services（中译版），微软Windows Live服务平台的一些经验性的总结文章，很值得一读。

4 Things to Keep in Mind When Building a Platform for the Enterprise ，Box平台VP海蒂·威廉姆斯（Heidi Williams）撰写的一篇文章，阐述了为企业构建平台时需要牢记的四件关于软件设计方面的事：1. Design Broadly, Build Narrowly； 2. Platforms Are Powerful and Flexible. Choose wisely what to expose when!；3. Build Incrementally, Get Feedback, and Iterate；4. Create a Platform-first Mentality。文章中有详细的解读，推荐看看。

Principles of Chaos Engineering ，我们知道，Netflix公司有一个叫Chaos Monkey的东西，这个东西会到分布式系统里“瞎搞”，以此来测试系统的健壮和稳定性。这个视频中，Netflix分享了一些软件架构的经验和原则，值得一看。

Building Fast & Resilient Web Applications ，伊利亚·格里高利克（Ilya Grigorik）在Google I/O 2016上的一次关于如何通过弹力设计来实现快速和可容错的网站架构的演讲，其中有好些经验分享。

Design for Resiliency ，这篇文章带我们全面认识“弹力（Resiliency）”，以及弹力对于系统的重要性，并详细阐述了如何设计和实现系统的弹力。

微软的Azure网站上有一系列的 Design Principle 的文章，你可以看看这几篇： Design for Self-healing 、Design for Scaling Out 和 Design for Evolution 。

Eventually Consistent ，AWS CTO维尔纳·沃格尔斯（Werner Vogels）发布在自己Blog上的一篇关于最终一致性的好文。

Writing Code that Scales ，Rackspace的一篇很不错的博文，告诉我们一些很不错的写出高扩展和高性能代码的工程原则。

Automate and Abstract: Lessons from Facebook on Engineering for Scale ，软件自动化和软件抽象，这是软件工程中最重要的两件事了。通过这篇文章，我们可以看到Facebook的关于这方面的一些经验教训。

设计模式

有了方法论后，你还需要学习一些比较细节的落地的技术。最好的方式就是学习被前人总结出来的设计模式，虽然设计模式也要分场景，但是设计模式可以让你知道一些套路，这些套路对于我们设计的分布式系统有非常大的帮助，不但可以让我们少走一些弯路，而且还能让我们更为系统和健壮地设计我们的架构。

下面是一些分布式架构设计模式的网站。

首先，需要重点推荐的是微软云平台 Azure 上的设计模式。 Cloud Design Patterns ，这个网站上罗列了分布式设计的各种设计模式，可以说是非常全面和完整。对于每一个模式都有详细的说明，并有对其优缺点的讨论，以及适用场景和不适用场景的说明，实在是一个非常不错的学习分布式设计模式的地方。其中有如下分类。

设计模式：可用性；

设计模式：数据管理；

设计模式：设计和实现；

设计模式：消息；

设计模式：管理和监控；

设计模式：性能和扩展；

设计模式：系统弹力；

设计模式：安全。

除此之外，还有其它的一些关于分布式系统设计模式的网站和相关资料。

AWS Cloud Pattern ，这里收集了AWS云平台的一些设计模式。

Design patterns for container-based distributed systems ，这是Google给的一篇论文，其中描述了容器化下的分布式架构的设计模式。

Patterns for distributed systems ，这是一个PPT，其中讲了一些分布式系统的架构模式，你可以顺着到Google里去搜索。

我个人觉得微服务也好，SOA也好，都是分布式系统的一部分，这里有两个网站罗列了各种各样的服务架构模式。

A Pattern Language for Micro-Services ；

SOA Patterns。

当然，还有我在极客时间上写的那些分布式的设计模式的总结。

弹力设计篇，内容包括：认识故障和弹力设计、隔离设计、异步通讯设计、幂等性设计、服务的状态、补偿事务、重试设计、熔断设计、限流设计、降级设计、弹力设计总结。

管理设计篇，内容包括：分布式锁、配置中心、边车模式、服务网格、网关模式、部署升级策略等。

性能设计篇，内容包括：缓存、异步处理、数据库扩展、秒杀、边缘计算等。

设计与工程实践

分布式系统的故障测试

FIT: Failure Injection Testing ，Netflix公司的一篇关于做故障注入测试的文章。

Automated Failure Testing ，同样来自Netflix公司的自动化故障测试的一篇博文。

Automating Failure Testing Research at Internet Scale ，Netflix公司伙同圣克鲁斯加利福尼亚大学和Gremlin游戏公司一同撰写的一篇论文。

弹性伸缩

4 Architecture Issues When Scaling Web Applications: Bottlenecks, Database, CPU, IO ，本文讲解了后端程序的主要性能指标，即响应时间和可伸缩性这两者如何能提高的解

决方案，讨论了包括纵向和横向扩展，可伸缩架构、负载均衡、数据库的伸缩、CPU密集型和I/O密集型程序的考量等。

Scaling Stateful Objects ，这是一本叫《Development&Deployment of Multiplayer Online Games》书中一章内容的节选，讨论了有状态和无状态的节点如何伸缩的问题。虽然还没有写完，但是可以给你一些很不错的基本概念和想法。

Scale Up vs Scale Out: Hidden Costs ，Coding Horror上的一篇有趣的文章，详细分析了可伸缩性架构的不同扩展方案（横向扩展或纵向扩展）所带来的成本差异，帮助你更好地选择合理的扩展方案，可以看看。

Best Practices for Scaling Out ，OpenShift的一篇讨论Scale out最佳实践的文章。

Scalability Worst Practices ，这篇文章讨论了一些最差实践，你需要小心避免。

Reddit: Lessons Learned From Mistakes Made Scaling To 1 Billion Pageviews A Month ，Reddit分享的一些关于系统扩展的经验教训。

下面是几篇关于自动化弹性伸缩的文章。

- Autoscaling Pinterest；
- Square: Autoscaling Based on Request Queuing；
- PayPal: Autoscaling Applications；
- Trivago: Your Definite Guide For Autoscaling Jenkins；
- Scryer: Netflix’ s Predictive Auto Scaling Engine。

一致性哈希

Consistent Hashing ，这是一个一致性哈希的简单教程，其中还有代码示例。

Consistent Hashing: Algorithmic Tradeoffs ，这篇文章讲述了一致性哈希的一些缺陷和坑，以及各种哈希算法的性能比较，最后还给了一组代码仓库，其中有各种哈希算法的实现。

Distributing Content to Open Connect ，Netflix的一个对一致性哈希的实践，提出了Uniform Consistent Hashing，是挺有意思的一篇文章。

Consistent Hashing in Cassandra ，这是Cassandra中使用到的一致性哈希的相关设计。

数据库分布式

Life Beyond Distributed Transactions ，该文是Salesforce的软件架构师帕特·赫兰德（Pat Helland）于2016年12月发表的针对其在2007年CIDR（创新数据库研究会议）上首次发表的同名文章的更新和缩写版本。业界谈到分布式事务通常指两段提交2PC事务（Spring/JEE中JTA等)或者Paxos与Raft，这些事务都有明显缺点和局限性。
而赫兰德在本文讨论的是另外一种基于本地事务情况下的事务机制，它是基于实体和活动（Activity）的概念，其实类似DDD聚合根和领域事件的概念，这种工作流类型事务虽然需要程序员介入，依靠消息系统实现，但可以实现接近无限扩展的大型系统。赫兰德文中提出了重要的观点：“如果你不能使用分布式事务，那么你就只能使用工作流。”

How Sharding Works ，这是一篇很不错的探讨数据Sharding的文章。基本上来说，数据Sharding可能的问题都在这篇文章里谈到了。

Why you don’ t want to shard ，这是Percona的一篇文章，其中表达了，不到万不得已不要做数据库分片。是的，最好还是先按业务来拆分，先把做成微服务的架构，然后把数据集变简单，然后再做Sharding会更好。

How to Scale Big Data Applications ，这也是Percona给出的一篇关于怎样给大数据应用做架构扩展的文章。值得一读。

MySQL Sharding with ProxySQL ，用ProxySQL来支撑MySQL数据分片的一篇实践文章。

缓存

缓存更新的套路，这是我在CoolShell上写的缓存更新的几个设计模式，包括Cache Aside、Read/Write Through、Write Behind Caching。

Design Of A Modern Cache ，设计一个现代化的缓存系统需要注意到的东西。

Netflix: Caching for a Global Netflix ，Netflix公司的全局缓存架构实践。

Facebook: An analysis of Facebook photo caching ，Facebook公司的图片缓存使用分析，这篇文章挺有意思的，用数据来调优不同的缓存大小和算法。

How trivago Reduced Memcached Memory Usage by 50% ，Trivago公司一篇分享自己是如何把Memcached的内存使用率降了一半的实践性文章。很有意思，可以让你学到很多东西。

Caching Internal Service Calls at Yelp ，Yelp公司的缓存系统架构。

消息队列

Understanding When to use RabbitMQ or Apache Kafka ，什么时候使用RabbitMQ，什么时候使用Kafka，通过这篇文章可以让你明白如何做技术决策。

Trello: Why We Chose Kafka For The Trello Socket Architecture ，Trello的Kafka架构分享。

LinkedIn: Running Kafka At Scale ，LinkedIn公司的Kafka架构扩展实践。

Should You Put Several Event Types in the Same Kafka Topic? ，这个问题可能经常困扰你，这篇文章可以为你找到答案。

Billions of Messages a Day - Yelp’ s Real-time Data Pipeline ，Yelp公司每天十亿级实时消息的架构。

Uber: Building Reliable Reprocessing and Dead Letter Queues with Kafka ，Uber公司的Kafka应用。

Uber: Introducing Chaperone: How Uber Engineering Audits Kafka End-to-End ，Uber公司对Kafka消息的端到端审计。

Publishing with Apache Kafka at The New York Times ，纽约时报的Kafka工程实践。

Kafka Streams on Heroku ，Heroku公司的Kafka Streams实践。

Salesforce: How Apache Kafka Inspired Our Platform Events Architecture ，Salesforce的Kafka工程实践。

Exactly-once Semantics are Possible: Here’ s How Kafka Does it ，怎样用Kafka让只发送一次的语义变为可能。这是业界中一个很难的工程问题。

Delivering billions of messages exactly once 同上，这也是一篇挑战消息只发送一次这个技术难题的文章。

Benchmarking Streaming Computation Engines at Yahoo!。Yahoo!的Storm团队在为他们的流式计算做技术选型时，发现市面上缺乏针对不同计算平台的性能基准测试。于是，他们研究并设计了一种方案来做基准测试，测试了Apache Flink、Apache Storm和Apache Spark这三种平台。文中给出了结论和具体的测试方案。（如果原文链接不可用，请尝试搜索引擎对该网页的快照。）

关于日志方面

Using Logs to Build a Solid Data Infrastructure - Martin Kleppmann ，设计基于log结构应用架构的一篇不错的文章。

Building DistributedLog: High-performance replicated log service ，Distributed是Twitter 2016年5月份开源的一个分布式日志系统。在Twitter内部已经使用2年多。其主页在 distributedlog.io。这篇文章讲述了这个高性能日志系统的一些技术细节。另外，其技术负责人是个中国人，其在微信公众号中也分享过这个系统 Twitter高性能分布式日志系统架构解析。

LogDevice: a distributed data store for logs ，Facebook分布式日志系统方面的一些工程分享。

关于性能方面

Understand Latency ，这篇文章收集并整理了一些和系统响应时间相关的文章，可以让你全面了解和Latency有关的系统架构和设计经验方面的知识。

Common Bottlenecks ，文中讲述了20个常见的系统瓶颈。

Performance is a Feature ，Coding Horror上的一篇让你关注性能的文章。

Make Performance Part of Your Workflow ，这篇文章是图书《Designing for Performance》中的节选（国内没有卖的），其中给出来了一些和性能有关的设计上的平衡和美学。

CloudFlare: How we built rate limiting capable of scaling to millions of domains ，讲述了CloudFlare公司是怎样实现他们的限流功能的。从最简单的每客户IP限流开始分析，进一步讲到anycast，在这种情况下PoP的分布式限流是怎样实现的，并详细解释了具体的算法。

关于搜索方面

Instagram: Search Architecture

eBay: The Architecture of eBay Search

eBay: Improving Search Engine Efficiency by over 25%

LinkedIn: Introducing LinkedIn’ s new search architecture

LinkedIn: Search Federation Architecture at LinkedIn

Slack: Search at Slack

DoorDash: Search and Recommendations at DoorDash

Twitter: Search Service at Twitter (2014)

Pinterest: Manas: High Performing Customized Search System

Sherlock: Near Real Time Search Indexing at Flipkart

Airbnb: Nebula: Storage Platform to Build Search Backends

各公司的架构实践

High Scalability ，这个网站会定期分享一些大规模系统架构是怎样构建的，下面是迄今为止各个公司的架构说明。

YouTube Architecture

Scaling Pinterest

Google Architecture

Scaling Twitter

The WhatsApp Architecture

Flickr Architecture

Amazon Architecture

Stack Overflow Architecture

Pinterest Architecture

Tumblr Architecture

Instagram Architecture

TripAdvisor Architecture

Scaling Mailbox

Salesforce Architecture

ESPN Architecture

Uber Architecture

DropBox Design

Splunk Architecture

小结

今天我们分享的内容是高手成长篇分布式架构部分的最后一篇——分布式架构工程设计，讲述了设计原则、设计模式等方面的内容，尤其整理和推荐了国内外知名企业的设计思路和工程实践，十分具有借鉴意义。

下篇文章中，我们将分享微服务架构方面的内容。敬请期待。

下面是《程序员练级攻略》系列文章的目录。

微服务

容器化和自动化运维

机器学习和人工智能

前端基础和底层原理（前端方向）

前端性能优化和框架（前端方向）

UI/UX设计（前端方向）

技术资源集散地

精选留言 25

NonStatic
1531841627

吐槽只放链接的，一般自己是做不出这个链接列表的，更别说读完这些链接了。说句实在话，我花钱就是看耗子哥这个链接列表的！这里的链接列表能让我节省出用来找到和甄别哪片文章该读的学习时间，用来读更多的文章。

文章和书，就要读原始的那个。别人给你讲的都是他/她认为值得讲的，却不一定是你需要的。一个领域，入门的时候可以听听别人讲的，真到用的时候，只有自己读过原始资料才能给你你应该用到的细节。

作者回复 是的，读一手信息是最好的！

二胡1999
1531799669

回答Roger同学的问题：一直都有网页版啊。
所有专栏列表地址：
https://time.geekbang.org/columns

Roger
1531789969

发了这么多链接手机上很难看啊，什么时候有pc或者web端

李春霖
1531876205

应该会有很多人意识不到高价值信息索引意味着什么…… 收录的这些文章估计够我消化几年的了，支持耗子哥坚持下去！

C_love
1531845732

看到很多人吐槽只有链接，其实个人感觉这才是精华。

现在很多知识网上都能找到，但信息过滤所消耗的时间和精力是极大的。有像耗子哥这样的过来人的指引，会极大的减少这样的消耗。

技术学习往往都是自我驱动，由内而外，只看别人的总结或者嚼过的食物一般都会比较片面。从技术的源头去学习才能真正领悟其中三味。很多人追求的总结分析，大多是术，但不是道。

不过每个人需求不同，新人往往更多对术的东西感兴趣，这也很正常。如果基本的coding尚未得心应手就去学习更底层或者更难的知识会感觉眼高手低。这也是耗子哥反复强调要把前面的基础打好，虽然基础部分只有几章内容，但真正融会贯通可是需要几年。只能说专栏目

前的内容并不很适合新人。

所以大家先专注于适合自己的部分去学习就好啦

作者回复 是一个很踏实的人，坚持住□□

雪花飘飘
1533650671

一篇文章就够订阅的钱了，厉害

我是李香兰小朋友
1531787799

耗子哥，可以提个意见吗？作为一个两年的工作经验的人，其实比较希望看到是某种技术的详细介绍或者某个知识点的详细理解，或者是一些工作上的技术选型的一些经验，你最近这几篇文章都是介绍一些书籍资料，感觉对我来说帮助不大，我可以自己去找啊，希望耗子哥理解！

作者回复 最近几篇文章里的那些引用文章，你看了么？都是，技术的深度介绍和深度理解，同样包括技术比较。另外，作为一个两年工作经验的人，最近这几篇文章中罗列的这些技术文章对你来说太深了，你可能看不懂。

另外，在高手篇开篇的时候我说过了——

我假设你在前面已经打下了非常扎实的基础，但是要成为一个高手，基础知识只是一个地基，你还需要很多更为具体的技术。对我来说，就是看各种各样的文章、手册、论文、分享…… 其实，学习到一定程度，就是要从书本中走出去，到社区里和大家一起学习，而且还需要自己找食吃了。所以，对于这里面的文章，有很多都是在罗列各种文章和资源，只是为你梳理信息源，而不是喂你吃饭。

老实说，我已经为你梳理并过滤掉了很多的信息，这里只留下了 30% 我觉得最经济也最有价值的信息。虽然对于不同定位和不同需求的人还可以再对这些信息进行删减，但是觉得我这么一做就会对其它人不公平了。所以，这也是我觉得最小数量集的信息和资源吧。你也可以把我这里的东西当成一个索引来对待。

Milo
1531786385

越来越没意思了，就给个文章列表，花这个钱不值得了！

作者回复 谢谢批评，不过，这可是我精挑细选的文章列表啊，我已过滤了至少7成的信息了..... 这些文章都是含金量非常高的，仔细读一下，你会发现价值所在的，当然，你要打好前面的基础，否则学习这些东西，你的挫败感会很强的.....

三杯酒量
1557496054

为什么全是国外的，国内互联网公司现在也很厉害了，为啥不放点国内的

涛哥迷妹
1531929753

老师请问这些文章您是怎么找到能教教获取这种干货的方法吗？我觉得这个方法更有价值

作者回复 我后面会新开文章说的

Field Li
1531845341

怎么说呢？虽然资料确实都是好的，但是可能很多人跟我一样更想看一些结合作者的理解的文章，或者纯粹作者自身加工的文章吧。而不是纯放些链接上去了，链接的内容很好，但是感觉还是有敷衍的嫌疑

作者回复 我当然可以写我理解后的，但我还是觉得大家应该去读第一手的资料，而不是他人消化过的。这样会更好。

我给的这些文章或者是书籍，讲得都很好了，你先看几篇吧，真的不需要二手的。😁

JasonYang
1531830690

每期都有在看，突然觉得标题过大，程序员练级攻略与进阶 这个话题太大了，看到后面更像是您对自身熟悉的技术领域的老生常谈，链接确实不错，干货满满，但是标题太大了，这么多程序员，好比不同期待的食客，他们期待 你有满汉全席 ，但是你并不完美，你只能做川菜或是粤菜，所以我想由于您的标题过大，总有一部分的人是不满意的，其实酷壳老版的练级攻略之所以所欢迎，一方面是免费指导，另一方面是对未来趋势的预判，第三就是亲切，而且实用，可能是老版太好，难免期待会有落差，其实我一开始挺期待你说 大家都会遇见的瓶

颈或者疑问 如何高效的使用Google 或者其他软技能，因为这些东西具有普遍性，人人皆宜，话说回来，就是由于标题过大，你忙着把主线路说完，却忘了一些重要的东西

> 作者回复 文章开篇说过，这篇文章不是人人皆宜的，而且会是很吓人的，我就是要吓退那些想要速成的人。
>
> 另外，如你所说，我有我的局限，我能保证的不是面面俱到，而是主流技术，不至于让大家学完后没几年就废弃了。

godtrue
1547130253

二话不说，一睹为快。

建斌郭子
1539479034

补充说明，这些链接的价值在于，当你把这些链接里的内容全都看完后，你才会觉得不再需要这篇文章了。陈皓老师(没错，我认为的良师+高师)给了我们一个梯子，对我们来说，最好的做法是先沿着梯子爬过去，看看外面的世界，而不是纠结梯子好不好看

TomnyYear
1534914297

为什么写了那么多篇，没有讨论如何测试分布式的。还有代码级别的测试？

> 作者回复 没有，sorry，以后补上

于曦程
1533173706

耗子哥能单独讲讲搜索引擎吗

小破
1532761374

耗子叔的文章应该能看好些年了。😉

ricktian
1532482997

这些学习资料库的价值远远超过一两篇原创文！大力点赞！

sipom
1532074635

多谢，获益良多！耗子哥，除了资料还能再增加些您多年来工作中的实战经验体会吗？比如分布式架构设计、应用架构设计等方面的。谢谢！

> 作者回复 前面的几个分布式系列的文章都有的

木子李的李
1531932672

支持！！！ 干货最多的专栏，虽然我才一年经验好多看不懂。讲的好多都是通用的方法论，懂得人自然懂

斯盖丸
1531875321

耗子叔，您好。最近在看您推荐的经典书了。有个很古老的问题，这些书里大量的代码要背下来吗，如果不需要，那这些代码需要掌握到什么程度才算过关，谢谢。

> 作者回复 不要背，要了解原理，然后自己按自己的思路实现一下。我后面会有如何学习的文章，敬请期待。

kingeasternsun
1531871224

这才是真正的干货，这才是真正的宝藏，不会挖掘只会张嘴等喂的人永远成长不了

kuzan
1531868453

这些文章链接让读者自己找不知道要花多大力气，支持目前这种方法，希望老师不要改变，师父领进门 修行在个人，成长要靠自己去领悟的

作者回复 谢谢理解。的确花了很大力气并对比了相似文章，还做了好多筛选以让大家关注更重要的信息。

另外，我没有作更多的描述和说明，是希望大家能自己阅读，自己领悟，习惯吃第一手的食物，而不是被别人消化过的。

再次感谢理解！

小鱼儿

1531836973

大部分都是英文版的，有对应中文版就更好了！

作者回复 在《程序修养篇》中说过：“必须指出，再往下走，有一个技能非常重要，那就是英文。如果对这个技能发怵的话，那么你可能无缘成为一个程序员高手了。因为我们所有的计算机技术全部来自于西方国家，所以如果你要想成为一个高手的话，那么必须到信息的源头去。英文的世界真是有价值的信息的集散地”，所以，非常建议你加强自己的英文能力！加油💪

武坤

1531821694

吐槽老师只放链接的，建议一字不差地从开篇读。虽然现在的内容我已经看不懂了，但是我知道这些内容都是精华，在学好专业基础篇后会认真看的。