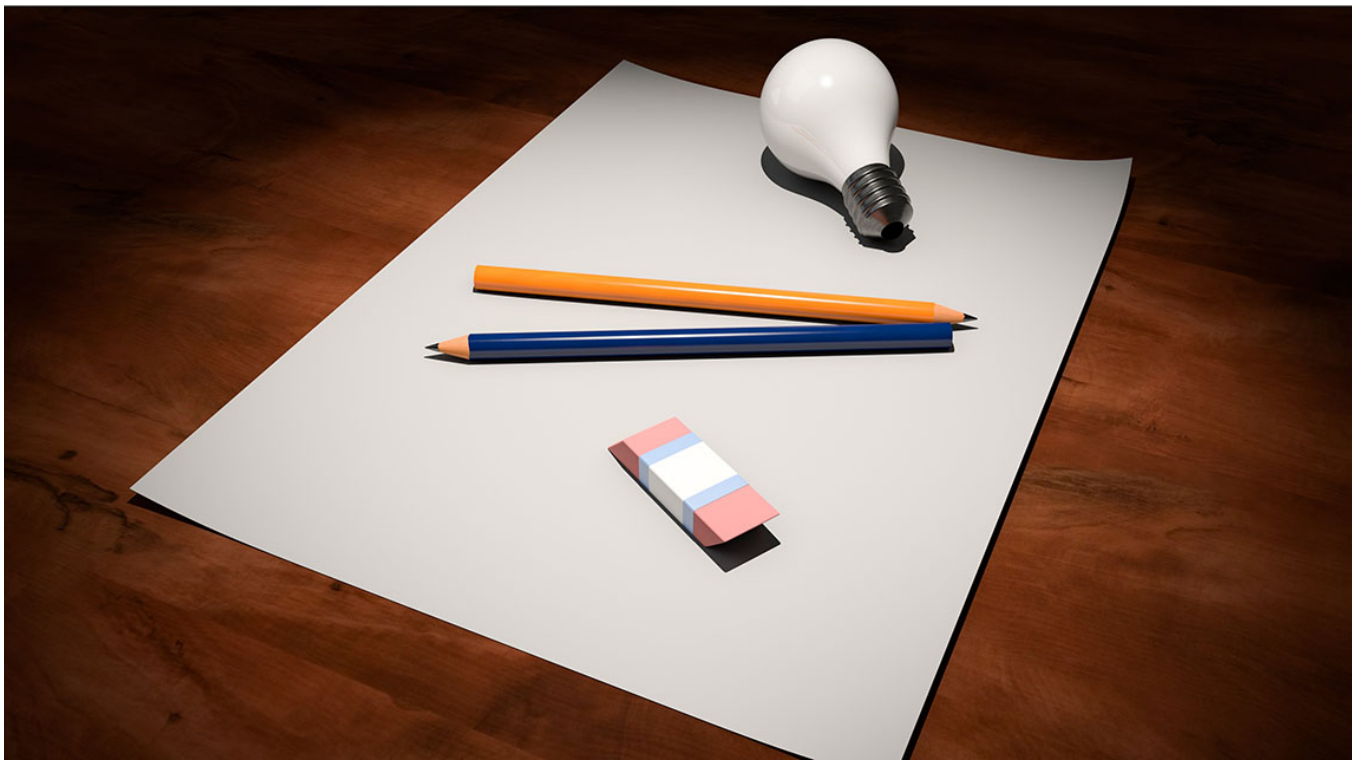
程序员练级攻略（2018）：分布式架构工程设计

2018-07-17 陈皓，杨爽



程序员练级攻略（2018）：分布式架构工程设计
朗读人：柴巍　17′13″ | 7.89M

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（2018）》系列文章的目录（持续更新中）。

- 开篇词

- 入门篇
  - 零基础启蒙
  - 正式入门
- 修养篇
  - 程序员修养
- 专业基础篇
  - 编程语言
  - 理论学科
  - 系统知识
- 软件设计篇
  - 软件设计
- 高手成长篇
  - Linux 系统、内存和网络（系统底层知识）
  - 异步 I/O 模型和 Lock-Free 编程（系统底层知识）
  - Java 底层知识
  - 数据库
  - 分布式架构入门（分布式架构）
  - 分布式架构经典图书和论文（分布式架构）
  - 分布式架构工程设计 (分布式架构)
  - 微服务
  - ……

精选留言

二胡1999 4

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

2018-07-17

Roger 4

发了这么多链接手机上很难看啊，什么时候有pc或者web端

2018-07-17

我是李香兰小朋友 2

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！

2018-07-17

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。

2018-07-17

Milo 2

越来越没意思了，就给个文章列表，花这个钱不值得了！

2018-07-17

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.....

2018-07-17

武坤 0

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。

2018-07-17