程序员练级攻略（2018）：分布式架构入门

2018-07-10 陈皓

程序员练级攻略（2018）：分布式架构入门
朗读人：柴巍　13′22″ | 6.12M

学习分布式系统跟学习其它技术非常不一样，分布式系统涵盖的面非常广，具体来说涵盖如下几方面：

- **服务调度**，涉及服务发现、配置管理、弹性伸缩、故障恢复等。
- **资源调度**，涉及对底层资源的调度使用，如计算资源、网络资源和存储资源等。
- **流量调度**，涉及路由、负载均衡、流控、熔断等。
- **数据调度**，涉及数据复本、数据一致性、分布式事务、分库、分表等。
- **容错处理**，涉及隔离、幂等、重试、业务补偿、异步、降级等。
- **自动化运维**，涉及持续集成、持续部署、全栈监控、调用链跟踪等。

所有这些形成了分布式架构的整体复杂度，也造就了分布式系统中的很多很多论文、图书以及很多很多的项目。要学好分布式系统及其架构，我们需要大量的时间和实践才能真正掌握这些技术。

这里有几点需要你注意一下。

- 分布式系统之所以复杂，就是因为其太容易也太经常出错了。这意味着，你要把处理错误的代码当成正常功能的代码来处理。
- 开发一个健壮的分布式系统的成本是单体系统的几百倍甚至几万倍。这意味着，我们要自己开发一个，需要能力很强的开发人员。
- 非常健壮的开源的分布式系统并不多，或者说基本没有。这意味着，如果你要用开源的，那么你需要 hold 得住其源码。
- 管理或是协调多个服务或机器是非常难的。这意味着，我们要去读很多很多的分布式系统的论文。
- 在分布式环境下，出了问题是很难 debug 的。这意味着，我们需要非常好的监控和跟踪系统，还需要经常做演练和测试。
- 在分布式环境下，你需要更科学地分析和统计。这意味着，我们要用 P90 这样的统计指标，而不是平均值，我们还需要做容量计划和评估。
- 在分布式环境下，需要应用服务化。这意味着，我们需要一个服务开发框架，比如 SOA 或微服务。
- 在分布式环境下，故障不可怕，可怕的是影响面过大，时间过长。这意味着，我们需要花时间来开发我们的自动化运维平台。

总之，在分布式环境下，一切都变得非常复杂。要进入这个领域，你需要有足够多的耐性和足够强的心态来接受各式各样的失败。当拥有丰富的实践和经验后，你才会有所建树。这并不是一日之功，你可能要在这个领域花费数年甚至数十年的时间。

分布式架构入门

学习如何设计可扩展的架构将会有助于你成为一个更好的工程师。系统设计是一个很宽泛的话题。在互联网上，关于架构设计原则的资源也是多如牛毛。所以，你需要知道一些基本概念，对此，这里你先读一下下面两篇文章，都非常不错。

- Scalable Web Architecture and Distributed Systems ，这篇文章会给你一个大概的分布式架构是怎么来解决系统扩展性问题的粗略方法。
- Scalability, Availability & Stability Patterns ，这个 PPT 能在扩展性、可用性、稳定性等方面给你一个非常大的架构设计视野和思想，可以让你感受一下大概的全景图。

然后，我更强烈推荐 GitHub 上的一篇文档 - System Design Primer ，这个仓库主要组织收集分布式系统的一些与扩展性相关的资源，它可以帮助你学习如何构建可扩展的架构。

目前这个仓库收集到了好些系统架构和设计的基本方法。其中包括：CAP 理论、一致性模型、可用性模式、DNS、CDN、负载均衡、反向代理、应用层的微服务和服务发现、关系型数据库和 NoSQL、缓存、异步通讯、安全等。

我认为，上面这几篇文章基本足够可以让你入门了，因为其中基本涵盖了所有与系统架构相关的技术。这些技术，足够这世上 90% 以上的公司用了，只有超级巨型的公司才有可能使用更高层次的技术。

分布式理论

下面，我们来学习一下分布式方面的理论知识。

首先，你需要看一下 An introduction to distributed systems。 这只是某个教学课程的提纲，我觉得还是很不错的，几乎涵盖了分布式系统方面的所有知识点，而且辅以简洁并切中要害的说明文字，非常适合初学者提纲挈领地了解知识全貌，快速与现有知识结合，形成知识体系。这也是一个分布式系统的知识图谱，可以让你看到分布式系统的整体全貌。你可以根据这个知识图 Google 下去，然后你会学会所有的东西。

然后，你需要了解一下拜占庭将军问题（Byzantine Generals Problem）。这个问题是莱斯利·兰波特（Leslie Lamport）于 1982 年提出用来解释一致性问题的一个虚构模型（论文地址）。拜占庭是古代东罗马帝国的首都，由于地域宽广，守卫边境的多个将军（系统中的多个节点）需要通过信使来传递消息，达成某些一致的决定。但由于将军中可能存在叛徒（系统中节点出错），这些叛徒将努力向不同的将军发送不同的消息，试图会干扰一致性的达成。拜占庭问题即为在此情况下，如何让忠诚的将军们能达成行动的一致。

对于拜占庭问题来说，假如节点总数为 `N`，叛变将军数为 `F`，则当 `N >= 3F + 1` 时，问题才有解，即拜占庭容错（Byzantine Fault Tolerant，BFT）算法。拜占庭容错算法解决的是，网络通信可靠但节点可能故障情况下一致性该如何达成的问题。

最早由卡斯特罗（Castro）和利斯科夫（Liskov）在 1999 年提出的实用拜占庭容错（Practical Byzantine Fault Tolerant，PBFT）算法，是第一个得到广泛应用的 BFT 算法。只要系统中有 2/3 的节点是正常工作的，则可以保证一致性。PBFT 算法包括三个阶段来达成共识：预准备（Pre-Prepare）、准备（Prepare）和提交（Commit）。

这里有几篇和这个问题相关的文章，推荐阅读。

- Dr.Dobb’ s - The Byzantine Generals Problem
- The Byzantine Generals Problem

- Practicle Byzantine Fault Tolerance

拜占庭容错系统研究中有三个重要理论：CAP、FLP 和 DLS。

- CAP 定理，CAP 理论相信你应该听说过不下 N 次了。CAP 定理是分布式系统设计中最基础也是最为关键的理论。CAP 定理指出，分布式数据存储不可能同时满足以下三个条件：一致性（Consistency）、可用性（Availability）和 分区容忍（Partition tolerance）。 “在网络发生阻断（partition）时，你只能选择数据的一致性（consistency）或可用性（availability），无法两者兼得”。

  论点比较直观：如果网络因阻断而分隔为二，在其中一边我送出一笔交易：“将我的十元给 A”；在另一半我送出另一笔交易：" 将我的十元给 B "。此时系统要不是，a）无可用性，即这两笔交易至少会有一笔交易不会被接受；要不就是，b）无一致性，一半看到的是 A 多了十元而另一半则看到 B 多了十元。要注意的是，CAP 理论和扩展性（scalability）是无关的，在分片（sharded）或非分片的系统皆适用。

- FLP impossibility- 在异步环境中，如果节点间的网络延迟没有上限，只要有一个恶意的节点存在，就没有算法能在有限的时间内达成共识。但值得注意的是， “Las Vegas” algorithms（这个算法又叫撞大运算法，其保证结果正确，只是在运算时所用资源上进行赌博，一个简单的例子是随机快速排序，它的 pivot 是随机选的，但排序结果永远一致）在每一轮皆有一定机率达成共识，随着时间增加，机率会越趋近于 1。而这也是许多成功的共识算法会采用的解决问题的办法。

- 容错的上限 - 由 DLS 论文 ，我们可以得到以下结论。

  - 在部分同步（partially synchronous）的网络环境中（即网络延迟有一定的上限，但我们无法事先知道上限是多少），协议可以容忍最多 1/3 的拜占庭故障（Byzantine fault）。

  - 在异步（asynchronous）的网络环境中，具有确定性质的协议无法容忍任何错误，但这篇论文并没有提及 randomized algorithms，在这种情况下可以容忍最多 1/3 的拜占庭故障。

  - 在同步（synchronous）网络环境中（即网络延迟有上限且上限是已知的），协议可以容忍 100% 的拜占庭故障，但当超过 1/2 的节点为恶意节点时，会有一些限制条件。要注意的是，我们考虑的是 " 具有认证特性的拜占庭模型（authenticated Byzantine）"，而不是 " 一般的拜占庭模型 "；具有认证特性指的是将如今已经过大量研究且成本低廉的公私钥加密机制应用在我们的算法中。

当然，还有一个著名的“8 条荒谬的分布式假设（Fallacies of Distributed Computing）”。

1. 网络是稳定的。
2. 网络传输的延迟是零。
3. 网络的带宽是无穷大。
4. 网络是安全的。
5. 网络的拓扑不会改变。
6. 只有一个系统管理员。
7. 传输数据的成本为零。
8. 整个网络是同构的。

阿尔农·罗特姆 - 盖尔 - 奥兹（Arnon Rotem-Gal-Oz）写了一篇长文 Fallacies of Distributed Computing Explained 来解释为什么这些观点是错误的。另外，加勒思·威尔逊（Gareth Wilson）的文章 则用日常生活中的例子，对这些点做了通俗的解释。为什么我们深刻地认识到这 8 个错误？是因为，这要我们清楚地认识到——在分布式系统中错误是不可能避免的，我们在分布式系统中，能做的不是避免错误，而是要把错误的处理当成功能写在代码中。

下面分享几篇一致性方面的论文。

- 当然，关于经典的 CAP 理论，也存在一些误导的地方，这个问题在 2012 年有一篇论文 CAP Twelve Years Later: How the Rules Have Changed （中译版）中做了一些讨论，主要是说，在 CAP 中最大的问题就是分区，也就是 P，在 P 发生的情况下，非常难以保证 C 和 A。然而，这是强一致性的情况。

  其实，在很多时候，我们并不需要强一致性的系统，所以后来，人们争论关于数据一致性和可用性时，主要是集中在强一致性的 ACID 或最终一致性的 BASE。当时，BASE 还不怎么为世人所接受，主要是大家都觉得 ACID 是最完美的模型，大家很难接受不完美的 BASE。在 CAP 理论中，大家总是觉得需要＂三选二＂，也就是说，P 是必选项，那＂三选二＂的选择题不就变成数据一致性 (consistency)、服务可用性 (availability) 间的＂二选一＂？

  然而，现实却是，P 很少遇到，而 C 和 A 这两个事，工程实践中一致性有不同程度，可用性也有不同等级，在保证分区容错性的前提下，放宽约束后可以兼顾一致性和可用性，两者不是非此即彼。其实，在一个时间可能允许的范围内是可以取舍并交替选择的。

- Harvest, Yield, and Scalable Tolerant Systems ，这篇论文是基于上面那篇 "CAP 12 年后" 的论文写的，它主要提出了 Harvest 和 Yield 概念，并把上面那篇论文中所讨论的东西讲得更为仔细了一些。

- Base: An Acid Alternative （中译版），本文是 eBay 的架构师在 2008 年发表给 ACM 的文章，是一篇解释 BASE 原则，或者说最终一致性的经典文章。文中讨论了 BASE 与 ACID 原则的基本差异, 以及如何设计大型网站以满足不断增长的可伸缩性需求，其中有如何对业务做调整和折中，以及一些具体的折中技术的介绍。一个比较经典的话是——“在对数据库进

行分区后, 为了可用性（Availability）牺牲部分一致性（Consistency）可以显著地提升系统的可伸缩性 (Scalability)”。

- Eventually Consistent，这篇文章是 AWS 的 CTO 维尔纳·沃格尔（Werner Vogel）在 2008 年发布在 ACM Queue 上的一篇数据库方面的重要文章，阐述了 NoSQL 数据库的理论基石——最终一致性，对传统的关系型数据库（ACID，Transaction）做了较好的补充。

小结

好了，总结一下今天分享的内容。文章的开头，我给出了学习分布式架构需要注意的几个关键点，然后列出了入门学习的资源，基本涵盖了所有与系统架构相关的技术。随后讲述了拜占庭容错系统研究中有三个重要理论：CAP、FLP 和 DLS，以及 8 条荒谬的分布式假设，从理论和认知等角度让你更为清楚地理解分布式系统。最后分享了几篇一致性相关的论文，很实用很经典，推荐阅读。

下篇文章中，我将推荐一些分布式架构的经典图书和论文，并给出了导读文字，几乎涵盖了分布式系统架构方面的所有关键的理论知识。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）
- ……

精选留言

ruby 2

干货满满，好好学习～

2018-07-10

无故 1

耗子哥，有大数据相关的吗？

2018-07-10

Ivan Fan 1

希望也能介绍一下c++和java low latency的学习路线

2018-07-10