STUDY & RESEARCH

# [翻译]面向工程师的分布式系统理论

2014/08/12 | MIKESPOOK | LEAVE A COMMENT

如果一个工程师说起话来像 PhD. 走起路来像 PhD. 干起活來像 PhD. 那他是什么？
他是一个懂理论的工程师……

原文在此。
————翻译分隔线————

## 面向工程师的分布式系统理论

Gwen Shapira，系统工程师明星，现在是 Cloudera 的全职工程师，在 Twitter 上问了一个问题，引发了我的思考。

我以前可能这样回答“好吧，这里有 FLP 的论文，而这是 Paxor的论文，还有这是拜占庭将军的论文……”，并且我会列出一个长长的原材料的清单，如果你够快的话大概可以在六个月的时间里看一遍。但是，现在我觉得提供一大堆关于理论的论文对于学习分布式系统理论往往是错误的道路（除非你在读 PhD）。论文通常都很深奥，也很复杂，需要认真的学习，且便随着巨大成就的是丰富的经验，和相关的业务上下文。有什么是需要这样高等级的工程师专家的呢？

不幸的是，与此同时，缺少一个好的能够通向这些汇总材料的“桥梁”，提炼并融汇分布式系统理论中重要的结论和思路；而一些进行这些总结的材料的难度又过大。在思考的时候，我又想到另外一个问题：

分布式系统工程师应当学习什么样的分布式系统理论？

就这个例子来说，应当是没这么恐怖的、简单一些的理论。因此，我试着整理了一份我作为分布式系统工程师，每天的工作会应用到的一些基本概念的清单。我觉得这个清单应该足够支持分布式系统工程师设计新的系统。如果我漏掉了什么，务必告诉我！

## 第一站

这四篇阅读材料很好的展示了建设分布式系统的挑战是什么。它们都在描绘分布式系统工程师需要克服的，经过抽象的技术难题的轮廓，并设定了后续更进一步的研究的策略。

有趣并有益的分布式系统 是一本简短的书，它试图涵盖关于分布式系统的一些基本问题，包括时序的角色和不同的复制策略。

年轻人的分布式系统注解——不是理论，不过作为你其他阅读材料的基础是很好的弥补。

分布式系统注解——经典论文，解释了为什么不能像对待本地对象那样进行远程交互。

分布式计算的错误——对于系统设计者来说，在设定分布式计算的策略时容易犯的 8 个错误。

## 失败与时序

分布式系统工程师面对的各种困难可以被归结为下面的两个基本原因：

- 进程可能失败
- 没有好的方式通知，事务已经完成

这两者之间有者很深的内在联系，如果如果进程共享它们的时序，那么失败就可以被检测到，那么算法和原理就可以被正确的实现。大多数轻卡下，我们假设两个节点之间绝对无法共享当前时序，或者共享时序消耗的速度。

你应当知道：

（局部）层级失败模式：崩溃即停 -> 忽略 -> 拜占庭。你应当了解在上层发生的事情，也可能发生在下层，在下层不可能发生的事情，也不可能发生在上层。

在没有同步时钟的情况下，如何判断一个事件是否发生在另一事件之前。这意味着 Lamport 时钟以及更普适的 Vector 时钟，不过也要看看 Dynamo 论文。

哪怕只有一个错误，但是它对我们分布式系统的正确性到底会产生多大的影响（参考我在下面关于 FLP 结论的注解）。

不同的时序模型：同步、部分同步和异步（当我找到更好的引用时会加上链接）。

## 容错的基本悖论

一个系统在不降级的情况下容忍一些错误的发生，就必须运行起来像这些错误从未发生过一样。也就是说，通常系统的某些部分必须做一些冗余的工作，但是超过必要的冗余工作通常会导致性能的损失和资源的消耗。这就是为系统添加容错的悖论。

你应当知道：

Quorum 技术保证了单一副本的可串行性。参阅 Skeen 的原始论文，不过 Wikipedia 的条目会更好。

关于 2 段提交、3 段提交和 Paxos，以及为什么它们具有不同的容错特性。

关于最终一致性以及其他技术是如何寻找在更少的开销下为系统的行为提供保障，来避开这一悖论的。Dynamo 论文是一个良好的开端，Pat Helland 经典的超越事务一定得读。

## 基本构成

在分布式系统中有一些公认的基本构建模块，不过还有其他一些也开始浮现。你应当了解下面这些问题是什么，以及去哪寻找它们的答案：

领袖选举（例如 Bully 算法）

一致快照（例如这些来自 Chandy 和 Lamport 的经典论文）

一致性（参阅 2PC 的博文和上面的 Paxos）

分布式状态机复制（Wikipedia 就可以了，Lampson 的论文更加规范不过很枯燥）。

## 基本结论

一些事实只要主观判断。自然，还有一些需要做得更多，不过这也就是乐趣所在：

当在进程之间可能丢失消息的时候，你无法在实现一致性存储的同时响应所有的请求。这是 CAP 原则。

一致性不可能同时满足以下条件 a) 总是正确 b) 在异步系统中只要有一个机器发生故障，系统总是能终止运行——停止失败（FLP 结论）。在给出证明之前，首先是一个我在洛杉矶演讲的幻灯片：我们喜爱的论文，我希望它能合理的解释这个结论。
建议：确实没有必要理解其证明。

通常一致性无法在少于两轮的通讯中解决

## 实际系统

最重要的经验是不断的阅读新的、实际的系统的说明，并且判断它们的设计思路。一遍一遍的重复。一些建议：

Google：
GFS、Spanner、F1、Chubby、BigTable、MillWheel、Omega、Dapper。Paxos Made Live，The Tail At Scale。

非 Google：
Dryad、Cassandra、Ceph、RAMCloud、HyperDex、PNUTS

## 附言

如果你驯服了这个列表中的所有的理论和技术，我很愿意与你讨论一下在 Cloudera 的圈养分布式系统的工程师职位。

分布式系统 理论 翻译