# 分布式系统原理

任杰

2020 年 5 月 19 日

# 目录

# 序

十多年前我第一次走出校园正式从事软件开发工作。当时的一个任务是写一个批处理程序。这个程序有三个进程，需要在三台不同机器上运行。这些进程之间有先后依赖关系，需要按照特定的顺序启动。当时还没有 zookeeper，也没有 etcd，Raft 算法还没有被发明。当时试了很多方法都没有作用。最后只能靠人来手动维护进程状态。如果发现进程因为启动顺序错误报警的话手动重启正确的进程。

之后我就换了组，慢慢的就把这件事情遗忘了。大概一年多以后我突然想起了这个曾经让我头疼的问题。于是我请教了现在维护这个程序的开发人员。他很开心的给我展示了他的新解决方案。他做了一个很小的改动。之前每个进程如果发现所依赖的进程没有启动的话会出错退出。他把出错逻辑改成了一直重启，这样总有一次会有一个进程能启动正确，多来几次的话三个进程都能启动正确。我想了一想好像是这个道理。当时有一点不甘心。我可以写操作系统和编译器，却为什么连三个进程怎么正确启动都做不到。

多年以后我带着这个疑问进入了互联网的世界。这时候才发现我之前碰到的是分布式系统的一个简单到不能再简单的问题。分布式系统有它特定的领域问题和解决方案。对于做工程的人来说大多数情况下这些问题都有固定的解题套路。但是市面上缺乏能把问题解释的比较清楚的教材。这便是这本书背后的动机。

任杰

上海浦东

2020 年 5 月

# 第一章　基础概念

> A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable
>
> Leslie Lamport
> Email message sent to a DEC SRC bulletin board at 12:23:29 PDT on 28 May 87.

## 1.1　认识分布式系统

一提到分布式系统大家第一个想到的就是互联网。在互联网上大家访问的每个主页背后都有对应的服务器。这些服务器通常都在某个数据中心或者电信机房。所有这些服务器一起提供了互联网服务。大家用到的手机也是电信通讯这个大的分布式系统的一部分。现在大家提到的物联网或者 IOT 也是一个大的分布式系统。这些熟知的分布式系统或多或少都和互联网有关系。那么有没有其它类型的分布式系统呢?

电力系统是一个隐藏很深的分布式系统。和之前提到的互联网、物联网、电信网不同，电力系统几乎是一个纯粹的物理世界的网络。电网有很多发电端，比如火电、风电、水电、太阳能等。这些电能会通过电网输送至千家万户。电网和互联网有很多相似之处。比如都有负载均衡和流量控制。当某些节点出现故障时系统能自行调度。其实电网和互联网的区别只是电子和电磁波的区别，两者甚至连传播的速度都是一样的。

分布式系统有一些共性，可以从分布式系统这个名词的组成看出。分布式系统这个名词有两个组成部分：分布式和系统。分布式意味着存在多个功能单元分散在不同的地方。系统意味着所有这些功能单元能有机地结合在一起完成某个特定的任务。接下来让我们看看什么是系统和为什么要做分布式。

## 1.2　什么是系统

我们把一组单元称为一个系统并不仅仅因为它们仅仅有相似的名字或者颜色。系统和组织一样，都有既定的任务，比如互联网能让用户上网，电网能输电。这些任务通常都有质量要求。比如大家玩网络游戏会关心网络延时，多个人在一幢楼里下载文件时会关心下载的速度快不快。更专业一点的人会有量化的衡量指标，比如延时在百分之 99 的情况下是多少毫秒，即 P99 是多少。因此当我们在设一个分布式系统的时候，首先需要想清楚的是这个系统究竟是用来干什么的，其次是有没有可能用数字来衡量这个系统的质量。

## 1.3 为什么要做分布式

按照存在即合理[1] 的解释，一个系统之所以演变成为了分布式系统，那么它就一定有它背后原因。这个原因如此之强以至于就算分布式系统再难做也得做。一般来讲背后的原因有三个:

1. 物理学的资源稀缺性
2. 经济学的边际效益递减
3. 经济学的性价比权衡

拿一台电脑来举例。抽象来讲一个程序为了运行需要数据和计算资源。数据需要存储在计算机上，首选是存储在内存里。不同容量内存有不同价格。当我们需要单条内存 10 倍内存容量时，这个内存价格通常不是 10 倍而是更高。原因是单条内存容量越高，离物理局限越近，成品率越低，价格也越高，这就是物理资源稀缺性。这意味着额外的一块钱带来的内存容量提升会低于之前一块钱的效果，这就是边际效益递减。因此与其说我把这一块钱投资到已经很大的内存条上，还不如再买一块新的内存。主板在支持内存条数量这个问题上也有边际效应递减的问题，因此当主板支持一定数量的内存条后再增加额外的内存条就不合算了。这时候需要考虑把数据存储在硬盘上，因为硬盘更大更便宜。但如果硬盘的所有属性均优于内存，那么内存应该会慢慢退出消费市场无人购买。那么内存既然依然存在，硬盘必然有一些劣势，那就是硬盘速度大幅低于内存速度，这便是性价比权衡。我们选择了更大的容量，但是牺牲了数据处理速度。当数据量继续提升至硬盘的大小极限时，硬盘需要用磁带机来替换，但是这样数据处理速度会更慢。可选的单机存储方案是有限的，总存在一个数据量极限，超过这个极限之后速度慢的无法接受。这时候必须得考虑用多台机器来存储数据。

计算资源也存在同样的规律。单块 CPU 处理速度因为光速的原因存在极限。这时候需要考虑多块 CPU 来处理数据。但是主板能支持的 CPU 个数通常是有限的。和数据存储碰到的问题一样，当我们对 CPU 处理速度的要求达到一定极限之后必须考虑用多台机器来处理。其它计算资源也面临同样的问题，比如网卡、GPU、路由器。

人们通常会低估单机处理能力，同时高估自己解决分布式系统的能力。因此当我们面临系统设计的问题时，能用单台机器解决就用单台机器，纵向扩容直至财务预算无法满足需求。接下来才能考虑横向扩展，用分布式系统解决方案。

[1] “存在即合理”是对黑格尔的误解。黑格尔的原话是“Was vernünftig ist，das ist wirklich；und was wirklich ist，das ist vernünftig”，即“凡是合乎理性的东西都是现实的；凡是现实的东西都是合乎理性的”。

# 第二章　时间

I'm late, I'm late! For a very important date! No time to say 'hello, goodbye', I'm late, I'm late, I'm late!

The White Rabbit
Alice's Adventures in Wonderland

## 2.1　物理时间

### 2.1.1　时间的定义

时间的定义经过了漫长而且曲折的过程。最开始的时间是按照地球的自转速度来定义的。时间的单位是秒，一秒钟定义为 $\frac{1}{86400}$ 天。后来科学家觉得地球轨道是个椭圆，可能每天的长度不太一致，因此把一秒钟定义为地球按照太阳公转的 $\frac{1}{31556925.9747}$。这两种定义的都是天文秒。随着原子物理的发展，科学家们找到了更为可靠的周期性时间，因此有了原子秒，定义为铯-133 原子在其基态两个超精细能级间跃迁时辐射的 9192631770 个周期所持续的时间。

虽然我们有非常正式的时间定义，但是在实际生活中大家不可能衡量每天的 $\frac{1}{86400}$ 到底有多长，也不会拿本子去数铯-133 原子是不是经过了 9192631770 个周期。因此我们需要其它的方法来提供一个相对准确的时间。电脑的时间是通过由主板 BIOS 的时钟电路来维护的。但是主板提供的这个时间不一定很准，温度的高低都有可能影响准确率。这时候就需要用一台我们认为准确的机器来校对时间不准的机器。这个校对通常是通过一个NTP服务[1]来提供的。

### 2.1.2　时间的作用

时间是个很常见的事物。大家的手机、电脑都会告诉你当前时间是多少。虽然时间很普遍，时间的标准却是国家级战略工程。中国科学院国家授时中心作为我们国家的唯一官方时间提供单位，其成立需要国务院和中央军委批准。原因很简单，火箭、导弹、卫星等都通过自己携带的电脑来做速度的计算。这些武器的运动速度非常快，一秒钟可以移动 7000 米。如果时间仅有 0.001 秒的误差，对应的则是 7 米的误差。

时间除了记录单个物体经历了多久以外，在分布式系统中时间更多是用来衡量不同节点的相对关系。在第2.2节 逻辑时间 中会有更多介绍。

### 2.1.3　时间的性质

在不考虑爱因斯坦的牛顿物理世界里，时间有个非常好的性质是它可以用连续的正实数在表示。因此时间的性质和正实数的性质一致。具体来说时间有如下特性：

1. 两个时间之间可以比大小。时间大表示后发生，时间小表示先发生。

[1] 协议定义在互联网标准 RFC-1305。

2. 两个时间之间的时间间隔有比例关系。比如 3 小时时间间隔是 1 小时时间间隔的 3 倍。

虽然时间间隔之间是存在比例关系，两个时间点之间是不存在比例关系的，比如 2020 年不是 1010 年的两倍。存在这个区别的原因在于时间的值取决于参考点的设置，即 0 点是什么时候。如果是计算机，0 点是 1970 年 1 月 1 日 0 点。如果是公历，0 点是耶稣诞生的时候。

时间的另一个常见误解是人们通常想知道现在是什么时间。在日常生活中如果听见有人问现在几点了，大家会不假思索的看看表，报一个看到的时间。其实严格来讲，当你看到时间的一瞬间这个时间已经过去了，成为了历史时间。这也是人不能两次踏入同一条河流的原因[2]

### 2.1.4 时间的获取

时间是一个正实数，因此时间的精度是无限的[3]。由于计算机的数据表示形式是有限的，这就意味着我们需要对时间精度做一些取舍。一般来说精度越高，获取的难度越大。一般来说有以下几种获取方式：

1. 调用操作系统 API，获取本地机器当时的时间。
2. 通过 NTP 获取网络时间。NTP 是一个层级传播系统。最顶层通过高精度物理设备获取高精度时间。这个时间会逐层传播下去。用户通过 UDP 网络协议获取时间。
3. 通过 PTP[4]获取时间。PTP 精度在毫秒级以下，通常作为金融高频交易系统、电网、电信通讯等系统做时间同步用。PTP 假设时间的生产者通过广播的方式和消费者之间直接通讯，不通过路由器。
4. 通过 GPS 获取时间。GPS 通常提供经纬度坐标和高度这 3 维数据。其实 GPS 还提供第 4 维数据时间。每个 GPS 内都有多个原子钟，这样就算某个原子中出了问题，备份系统也能提供准确时间。

### 2.1.5 时间存在的问题

时间本身没有问题，问题在于我们获取的时间源不准：

1. 主板时间可能因为温度的变化导致时间忽快忽慢。
2. 操作系统可能获取时间后因为内核调度的原因无法及时返回给应用层。
3. 虚拟机作为操作系统之上的应用程序，可能因为中断的原因无法及时返回给虚拟机内的应用。
4. 获取时间的线程可能因为内核调度的原因休眠，无法及时处理获取到的时间。

另外一个极少有人知道的事情是时间并不一定是增加的，时间可能会不变。闰秒是指为保持协调世界时接近于世界时时刻，由国际计量局统一规定在年底或年中（也可能在季末）对协调世界时增加或减少 1 秒的调整。最近一次闰秒在北京时间 2017 年 1 月 1 日 7 时 59 分 59 秒。这时出现了 59 分 60 秒这个奇怪的时间。如果系统使用的是 CLOCK_REALTIME 而不是 CLOCK_MONOTONIC[5]，会发现这一秒时间没有变化。

因此结论是我们很少有正确的物理时间。那怎么办呢？

[2] It is not possible to step twice into the same river according to Heraclitus, or to come into contact twice with a mortal being in the same state。

[3] 准确的说精度是可数的，即 countable。

[4] Precision Time Protocol。

[5] CLOCK_MONOTONIC 记录了系统在启动后经历了多久的时间，因此是一个相对时间，不会因为外界时间的定义发生改变。

## 2.2 逻辑时间

### 2.2.1 逻辑时间的作用

物理时间有非常良好的性质，但是现实世界有种种因素导致我们无法完全利用这些性质。那么让我们看看究竟哪些性质是重要的、必不可缺的，哪些性质是可以放弃的。

时间的一个性质是两个时间点之间可以比大小。这是一个非常有用的信息。一旦我们可以比较两个时间的大小，我们就可以判断其发生的顺序。比如在处理数据库的读写冲突时，我们就可以知道是读操作先发生，还是写操作先发生。这个先后顺序能决定数据库应该用哪种方式来解决冲突。所以时间可以比大小这个性质需要保留。

时间的另一个性质是时间间隔有比例关系。这个貌似是不太常见的性质。比如我们知道一周的时间间隔是一天的 7 倍，而且无论 0 点如何选取，这个比例关系永远是 7 倍。可是这个倍数有什么用呢？所以这个比例关系是一种屠龙之技[6]，比较鸡肋[7]，可以放弃。

在放弃了时间段的比例关系后，我们需要进一步思考还有哪些性质是可以放弃的。在第2.1.4节我们提到过物理时间是无限精度而计算机时间是有限精度。一旦时间的衡量精度下降，如果两个事件发生的足够近，那么我们将无法判断这两个事件的发生先后顺序。既然我们做不到无限精度，有限精度又存在各种问题，还不如寻求另一种时间的表示方式来满足两个时间之间可以比大小这个功能。这便是逻辑时间。

### 2.2.2 逻辑时间的定义

逻辑时间是 Lamport 在 1978 年第一次完整提出的 [Lam78]。在正式定义逻辑时间前我们需要对系统做一些简单的假设。

我们假设在分布式系统中有多个节点。这些节点之间通过点对点发消息的方式来互相沟通。消息的传播需要一定的时间，而且无法预期这个时间需要多久。另外如果一个节点发了多条消息给另一个节点，这些消息可能会乱序到达，而不是保证先发先到。

与消息相关的另一个定义是事件。事件是导致节点状态变化的因素。节点有 3 种不同事件：

1. 发送消息的事件。
2. 接受消息的事件。
3. 内部事件。

对于某个消息来说，其发送和接受都会影响节点的状态。同时节点内部也会自我发生一些状态变化，导致这些内部状态变化的是内部事件。

基本概念到这里已经定义的差不多了，接下来看看事件的先后关系定义：

**Definition 1.** 对于两个事件 $e_i$ 和 $e_j$，它们之间的先后关系 $\rightarrow$ 定义为：

1. 如果 $e_i$ 和 $e_j$ 发生在同一个节点，且 $i < j$，那么 $e_i \rightarrow e_j$。
2. 如果 $e_i$ 是一个消息的发送事件，$e_j$ 是同一个消息的接受事件，那么 $e_i \rightarrow e_j$。
3. 对于第三个事件 $e_k$，如果 $e_i \rightarrow e_j$，而且 $e_j \rightarrow e_k$，那么 $e_i \rightarrow e_k$。

如果 $e_i$ 和 $e_j$ 之间既不满足 $e_i \rightarrow e_j$，也不满足 $e_j \rightarrow e_i$，那么 $e_i$ 和 $e_j$ 之间是并发关系，记为 $e_i \parallel e_j$。

$\rightarrow$ 和 $\parallel$ 的定义需要一些解释。在分布式系统中的各个节点只了解本地信息。所以如果单个节点内发生了两个事件 $e_i$ 和 $e_j$，它是能判断这两个事件发生时打的标记 $i$ 和 $j$ 来判断事件的先后顺序。

但是这个节点对其它节点发生了什么变化并不能准确的了解。唯一了解途径是通过获取其它节点发送的消息。一旦一个节点收到了另一个节点的消息，这个节点可以一定程度猜测另一个节点发生了

[6] 《庄子·列御寇》：“朱泙漫学屠龙于支离益，殚千金之家，三年技成，而无所用其巧。”.
[7] 《后汉书·杨修传》：“夫鸡肋，食之则无所得，弃之则如可惜。”。

什么，从而能确定某些事件的先后顺序。最简单的先后顺序判定是收到消息的时间一定晚于发送消息事件，所以如果 $e_i$ 是一个消息的发送事件，$e_j$ 是同一个消息的接受事件，那么 $e_i$ 一定早于 $e_j$ 发生，因此 $e_i \to e_j$。更抽象一点来讲，每个节点像盲人摸象[8]一样在本地维护了对整个分布式系统的片面认识。一旦节点收到了其它节点的消息，这个节点便能更新它对整个分布式系统的认知，从而更近一步逼近最新的全局状态。虽然更新之后的认知依然是片面的，但比收到消息之前准确了一些。

先后顺序有一个良好的数学性质是它具有可传递性。所以一旦我们发现 $e_i$ 在 $e_j$ 之前发生，而且 $e_j$ 在 $e_k$ 之前发生，根据传递性规则，我们可以肯定 $e_i$ 在 $e_k$ 之前发生。因此如果 $e_i \to e_j$，而且 $e_j \to e_k$，那么 $e_i \to e_k$。

最后剩下一种情况是 $e_i$ 和 $e_j$ 之间没有任何可以推导出来的先后关系。这时候我们“定义”这两个事件是同时发生的。这里的同时指的是在逻辑时间的范畴内，而不是指真实的物理世界内同时发生。

说到这里我们终于可以看一下逻辑时间的定义：

**Definition 2.** 假设 $(S, <)$ 是一个偏序集合，$E : \{e_i\}$ 是所有事件的集合。逻辑时间$C : E \to S$ 是一个函数，满足以下性质：

$$e_i \to e_j \Rightarrow C(e_i) < C(e_j) \tag{2.1}$$

强一致性逻辑时间$C : E \to S$ 需要满足以下性质：

$$e_i \to e_j \Leftrightarrow C(e_i) < C(e_j) \tag{2.2}$$

我们简单解释一下这些深奥的定义。逻辑时间给每个事件赋予了一个数。这些数有一个特殊性质，那就是数之间是可以比大小的。这些数构成了一个集合叫偏序集合。偏序集合的定义是集合成员之间存在大小关系，而且这些大小关系是可以传递的，但是不保证任意两个成员之间都存在大小关系。比如 $\{1, 2, \ldots, 1000, \ldots\}$ 是一个偏序集合，这个偏序集合里任意两个数都可以比大小。所有虚数也是一个偏序集合，只不过只有实数之间可以比大小[9]，虚数之间以及虚数和实数之间不能比大小。

逻辑时间不是给每个事件随意的赋予一个数。这个赋予的过程需要保证大小顺序。如果两个事件之间存在先后顺序，那么它们被赋予的数之间也需要存在等价的大小顺序。强一致性逻辑时间是对这个赋予过程提的更高的要求。逻辑时间只是要求如果事件有先后，那么数值有对应的大小关系。强一致性逻辑时间要求反过来也成立，即如果两个事件被赋予的数值之间有大小关系，那么这两个事件之间存在对应的先后关系。第2.2.3会介绍逻辑时间把事件映射到标量时间，第2.2.4节会介绍强一致性逻辑时间把事件映射到矢量时间。

### 2.2.3 标量时间

问题：相同时间值不表示同时发生时间大不表示发生在后面

### 2.2.4 矢量时间

矢量时间是强一致的，如果不具有可比性表示是同时发生

[8] 《大般涅槃经》三二：“其触牙者即言象形如芦菔根，其触耳者言象如箕，其触头者言象如石，其触鼻者言象如杵，其触脚者言象如木臼，其触脊者言象如床，其触腹者言象如甕，其触尾者言象如绳。”。

[9] 实数是虚数的子集。

# 第三章　一致性

## 3.1　顺序一致性

## 3.2　线性一致性

## 3.3　ACID

## 3.4　最终一致性

# 第四章　共识算法

## 4.1　定义

## 4.2　Raft

## 4.3　Paxos 及其它

## 4.4　复制状态机

# 第五章　数据复制

## 5.1　同步复制

## 5.2　异步复制

## 5.3　Quorun

## 5.4　一致性复制

# 第六章　数据切割

## 6.1　切割原则

## 6.2　一致性哈希

# 第七章　混沌工程

# 第八章　分布式系统案例

## 8.1　TCP

## 8.2　DNS

## 8.3　CDN

## 8.4　Spanner

## 8.5　Kafka

# 参考文献

[Lam78] Leslie Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Commun. ACM*, 21(7):558–565, 1978.

# 索引