春节加餐丨深入聊一聊计算机系统的时间

2022-02-02 陈现麟

《深入浅出分布式技术原理》 课程介绍

讲述：张浩

时长 11:42　大小 10.73M

你好，我是陈现麟。

在专栏“概述篇”第二节课“新的挑战”里，我们讨论过分布式系统在时钟上面临的挑战，今天这期春节加餐，我还会和你深入地聊一聊计算机系统的时间。

在计算机系统中，时间是一个非常重要的概念，首先它深刻影响着分布式系统的设计。如果我们想要了解如何简化分布式系统的设计，要先从单机系统的时间问题出发。举个例子来说，在构建分布式系统的时候，如果我们能在每个单机系统中，都获得精确的时间点或时间范围，就能大大简化分布式事务等相关的设计。

其次，在时间方面，分布式系统存在多时钟的问题，理解这个问题之前，也需要先了解单机系统的时间问题。**所以，为了让你深入地了解计算机系统的时间，我们就从单机计算机系统的层面来讨论时间，等你理解以后，再学习分布式系统的时候，就会事半功倍了。**

在计算机系统内部，主要有两种时钟：墙上时钟和单调时钟，它们都可以衡量时间，但却有本质的区别。在这节课中，我将带你了解两种时钟的相关知识，其中的墙上时钟是本节课的重点部分，然后我们再一起探讨如何对两种时钟进行管理。

墙上时钟

学习墙上时钟的相关知识，我们要先从墙上时钟的同步入手，了解时间同步出现误差的原因以及现有的解决方案，之后再分析闰秒出现的原因，以及闰秒的处理方式，最后我们会根据处理方式中的“跳跃式调整”的处理逻辑，来分析 2012 年一个 Linux 服务器宕机的案例。

墙上时钟又叫钟表时间，顾名思义，和我们平时使用的钟表的时间一样，表示形式为日期与时间。在 Linux 系统中，墙上时钟的表示形式为 UTC 时间，记录的是自公元 1970 年 1 月 1 日 0 时 0 分 0 秒以来的秒数和毫秒数（不含闰秒）。

Linux 系统需要处理闰秒的逻辑就是因为 Linux 系统使用 UTC 时间，但是系统中记录的 UTC 时间是不含闰秒的。

墙上时钟的同步

根据墙上时钟的定义，我们可以发现，墙上时钟的标准是在计算机外部定义的，所以确保墙上时钟的准确性就变成了一个问题。计算机内部的计时器为石英钟，但是它不够精确，随着机器的温度波动，会存在过快或者过慢的问题，**所以依靠计算机自身，来维持墙上时钟的准确性是不可能的，这就是计算机系统内的时间需要与外部时间进行同步的原因**。

目前普遍采取的一种方式为：计算机与 NTP 时间服务器定期通过网络同步。很明显，这个方式受限于网络时延的影响，一般来说，至少会有 35 毫秒的偏差，最大的时候可能会超过 1 秒。

在一些对时间精度要求很高的系统中，通过 NTP 进行同步是远远不够的，这时我们可以通过 GPS 接收机，接收标准的墙上时钟，然后在机房内部通过精确时间协议（ PTP ）进行同步。PTP 是一种高精度时间同步协议，可以达到亚微秒级精度，有资料说可达到 30 纳秒左右的偏差精度，但是它需要网络的节点（交换机）支持 PTP 协议，才能实现纳秒量级的同步。

在时间同步这个问题上， Google 的做法更酷，通过 GPS 接收机，接收标准的墙上时钟，然后通过机房内部去部署原子钟，使得它的精度可以达到每 2000 万年才误差 1 秒，用这种方式来防止 GPS 接收机的故障。

接着，再把这些时间协调装置连接到特定数量的主服务器，最后再由主服务器，向整个谷歌网络中运行的其他计算机传输时间读数，即 TrueTime API 。 Google 正是基于上面的时间精度保证，在此基础上实现了第一个可扩展的、全球分布式的数据库 Spanner。

闰秒出现的原因

从上述的讨论中，我们可以知道计算机的墙上时钟通过同步机制，确保时间的误差会保持在一个范围以内。虽然它保证了时间精度，但是因为 Linux 系统中，墙上时钟的表示形式为 UTC 时间，而 UTC 时间是不含闰秒的，所以如何处理闰秒就成为了一个重要的问题，那么我们先来想想闰秒出现的原因。

因为地球自转速率变慢，所以目前的两种时间计量系统：世界时和原子时，它们之间发生了误差，这就是闰秒出现的根本原因，下面我们就从世界时和原子时这两方面，具体来分析一下。

世界时（ UT1 ）以地球自转运动来计量时间，它定义地球自转一周为一天，绕太阳公转一周为一年，这对人们的日常生活非常重要。但是，因为地球自转速率正在变慢，世界时的秒长就会有微小的变化，每天会长千分之几秒，也就是说，后一天的 24 小时会比前一天的 24 小时要长千分之几秒，所以用世界时来度量时间，会出现均匀性非常不好的问题。

原子时取微观世界的铯原子中，两个超精细能级间的跃迁辐射频率来度量时间，精确度非常高，每天快慢不超过千万分之一秒。所以，原子时的均匀性非常好，是度量时间的理想尺度。

可是，原子时与地球空间位置无关，由于地球自转速率正在变慢，如果在某地区使用原子时，从今天开始计时，那么原子时到了明天凌晨 0：00 的时候，地球还需要等千分之几秒才自转完一周。这样一天一天地累积，就会出现原子时到了凌晨 0：00 这个时候，太阳还在地球正上空的情况，这显然是不符合常识的。

所以，为了统一原子时与世界时之间的差距，协调世界时（ UTC ）就产生了。从 1972 年 1 月 1 日 0 时起，协调世界时秒长采用原子时秒长，时刻与世界时的时刻之差保持在正负 0.9 秒之内，必要时用阶跃 1 整秒的方式来调整。

这个 1 整秒的调整，叫做闰秒，如果增加 1 秒就是正闰秒，减少 1 秒就是负闰秒。 UTC 从 1972 年 1 月起正式成为国际标准时间，它是原子时和世界时这两种时间尺度的结合。

闰秒的处理

因为 Linux 系统记录着，自公元 1970 年 1 月 1 日 0 时 0 分 0 秒以来的秒数和毫秒数，但是不含闰秒这种情况，导致了在 Linux 系统中每分钟有 60 秒，每天有 86400 秒是系统定义死的。

所以 Linux 系统需要额外的逻辑来处理闰秒。目前处理闰秒的方式主要有两种，一种是在 Linux 系统上进行跳跃式调整，另一种是在 NTP 服务上进行渐进式调整的 Slew 模型，下面我们具体讲一讲这两种处理逻辑。

跳跃式调整

首先是在 Linux 系统上进行跳跃式调整，当 UTC 时间插入一个正闰秒后，Linux 系统需要跳过 1 秒，即这一秒时间过去后，在 Linux 的时间管理程序中不应该去计时，因为闰秒的这一秒钟在 Linux 系统中不能被表示。

但是，当 UTC 时间插入一个负闰秒后，Linux 系统就需要插入 1 秒，即 Linux 的时间管理程序中要增加 1 秒钟的计时。虽然并没有过去 1 秒钟的时间，但是闰秒的这一秒钟在 Linux 系统中是不存在的。

目前 Linux 系统就是采用这种方式来处理闰秒的，所以在 2012 年 6 月 30 日, UTC 时间插入一个正闰秒的时候，Linux 系统会启动相应的逻辑来处理这个插入的正闰秒，这样就使某些版本的闰秒处理逻辑，触发了一个死锁的 bug，造成了大规模的 Linux 服务器内核死锁而宕机的情况。

Slew 模式

NTP 服务的 Slew 模式并不使用跳跃式修改时间，而是渐进式地调整。比如，当 UTC 时间需要插入一个正闰秒时， NTP 服务就会每秒调整一定的 ms 来缓慢修正时间。这样 Linux 系统从 NTP 服务同步时间的时候，就不会感知闰秒的存在了，内核也就不需要启动闰秒相关的逻辑了。

单调时钟

关于墙上时钟，我们主要讨论的是如何进行时间同步，确保时间精度的问题，而计算机系统中的第二种时钟，**单调时钟是一个相对时钟，不需要与外部的时钟进行同步，较墙上时钟要简单很多**，所以这里我们就简单地分析一下。

单调时钟总是保证时间是向前的，不会出现墙上时钟的回拨问题，所以它非常适合用来测量持续时间段，比如在一个时间点读取单调时钟的值，完成某项工作后再次获得单调时钟的值，时钟值之差就是两次检测之间的时间间隔。

到这里，我们可以看出，墙上时钟是绝对时钟，不同计算机节点上的墙上时间可以进行比较，但是它是有误差的，导致比较的结果不可信；而单调时钟是相对时钟，它的绝对值没有任何意义，有可能是计算机自启动以后经历的纳秒数等，所以，比较不同计算机节点上的单调时钟的值是没有意义的。这两点正是分布式系统面临时钟的问题。

时间的管理

前面我们讨论了墙上时钟和单调时钟，你一定很好奇操作系统内部是如何处理时间的，这里你可以先思考两个问题，我们带着问题再具体讨论。第一个问题是：计算机系统是没有时间概念的机器，那么它怎么来计算与管理时间呢？另一个问题是：计算机系统可以提供微秒甚至纳秒，那么它怎么处理这么高精度的时间呢？

首先，时间的概念对于计算机来说有些模糊，计算机必须在硬件的帮助下才能计算和管理时间。前面说的石英钟就是用来做计算机的系统定时器的，系统定时器以某种固定的频率自行触发时钟中断。由于时钟中断的频率是编程预定的，所以内核知道连续两次时钟中断的间隔时间，这个间隔时间就叫做节拍。**通过时钟中断，内核周期性地更新系统的墙上时钟和单调时钟，从而计算和管理好时间。**

其次，目前系统定时器的中断频率为 1000 HZ，那么计算机能处理的时间精度为 1 ms。然而很多时候需要更加精确的时间，比如 1 微秒，计算机是怎么来解决这个问题的呢？

其实解决的方式非常简单，在每一次计算机启动的时候，计算机都会计算一次 BogoMIPS 的值，这个值的意义是，**处理器在给定的时间内执行指令数，通过 BogoMIPS 值，计算机就可以得到非常小的精度了**。

比如在 1 秒内，计算机执行了 N 条指令，那么计算机的精度就可以达到 N 分之一秒。很明显 N 是一个非常大的数目，因此计算机可以得到非常精确的时间了。

总结

在这节课中，我们从计算机系统内部的两种时钟出发，深入地讨论了时间相关的话题。在墙上时钟这部分，我们讨论了计算机系统时间同步的方式，分析了闰秒产生的原因，以及 Linux 系

统应对闰秒的办法，然后概览性地讲了 Linux 系统是通过时钟中断进行时间的计算与管理的，最后分析了 Linux 系统可以提高时间精度的方法。

思考题

计算机历史上的“千年虫”问题你了解过吗？它和时间有关系吗？

欢迎你在留言区发表你的看法。如果这节课对你有帮助，也推荐你分享给更多的同事、朋友。

上一篇　春节加餐丨系统性思维，高效学习和工作的利器

下一篇　春节加餐丨技术债如房贷，是否借贷怎样取舍？

精选留言 (8)

写留言

Young
2022-02-02

1. clock
- Wall clock; Used to represent date and time; After synchronization, the local clock of our server can jump backward or forward in time
- Monotonic clock; Used to calculate duration; The time ALWAYS moves forward and will not be impacted by variations leading to jumps in time
2. Wall clock
- Linux uses UTC as wall clock
- UTC is not accurate because earth's rotation speed varies
- UTC does not contain Leap Second, so Linux adjust the UTC time to real earth rotation by inserting leap seconds into UTC time scale

3. Wall clock Synchronization
- Normal solution: synchronize with NTP (Network Time Protocol). Because of the network delay, there will be at least a 35 millisecond deviation, and the maximum may exceed 1 second.
- Good solution: receive accurate wall time with GPS, synchronize with other servers via PTP, and can achieve sub-microsecond precision
- Google solution: Google uses a GPS receiver to receive a standard wall clock, and then deploys the atomic clock through the computer room, so that its accuracy can reach an error of only 1 second every 20 million years, in this way to prevent the failure of the GPS receiver. Send accurate time to other servers with TrueTime API
- Google Spanner (distributed database) is based on this.
4. Resolve Leap Second Issues
- Bad Solution: notify the kernel, and allow it to insert the leap second, which cause stepping the system clock backwards
- Good Solution: Slew the leap second (since RHEL 6.8 and RHEL 7.2), allowing this second to be spread out over a period of time instead of an immediate step back in time
5. How computer manage time?
- Clock interrupts is based quartz crystal oscillator. Computer update the wall time during each clock interrupt
- How to deal with 1 microseconds (small precision)?
  BogoMIPS
Ref:
https://access.redhat.com/articles/15145
https://itnext.io/as-a-software-developer-why-should-you-care-about-the-monotonic-clock-7d9c8533595c
https://donggeitnote.com/2021/06/23/clock/

作者回复: 非常棒

7

wd
2022-02-03

这个主题非常棒。如果老师能够在正文中把这些重要概念的英文单词也列出来就更好了，这样的话如果需要学习英文文献的时候可以直接Google加深理解（看到前面网友在评论区已经贴出了一些英文原文，作为课程内容的补充来读，很有帮助）

作者回复: 好的，后面再补上~

1

springXu
2022-02-02

千年虫是上世纪由于历史上对年这个字段使用的是记录当年是用后两位来表示，如1995年是记成95的方式，当到了2000年时变成了00这样了，会产生问题。这个问题就是千年虫问题。
不知道我说的对不对？

作者回复: 对的！

1

静心 魔
2022-03-26

感觉加餐内容很棒呀！

作者回复: 😄

1

Geek_064e72
2022-03-16

原文：”每天会长千分之几秒，也就是说，后一天的 24 小时会比前一天的 24 小时要长千分之几秒“。我的理解：也就是说每三年，我们的一天时间就会快几秒？真有那么快吗？有天体及物理学专业人士吗？不符合常识呀。

Ankhetsin
2022-02-16

chrony怎么样？

作者回复: chrony 挺不错的

peter
2022-02-02

请教老师一个问题：
Q1：是否闰秒，是linux自己判断吗？还是从外部设备获取？

作者回复: 闰秒是墙上时钟才有的概念，是从外部设备获取的，比如 NTP 服务器。

努力努力再努力

2022-02-02

总结：

1. 目标： 分布式系统中 时间尽量准确
2. 时钟的分类：
    2.1 墙上时钟：
    2.2 单调时钟：
3. 墙上时钟：
    3.1 时间同步出现误差的原因：计算机内部用的是石英钟 不够准确，而墙上时钟的标准是计算机外部定义的
    3.2 现有的解决方案：
        3.2.1 计算机与 NTP 时间服务器定期通过网络同步 （受限于网络延时，至少35ms的误差）
        3.2.2 GPS接收机 接收标准的墙上时钟 ，然后在机房内部通过 PTP进行同步（亚微妙级精度， 需要交换机支持PTP协议）
        3.2.3 Google GPS接收机 接收标准的墙上时钟，通过机房内部去部署原子钟 （来防止GPS接收机的故障）
    3.3 闰秒出现的原因
        3.3.1 时间计量系统： 世界时和 原子时，他们之间发生了误差
    3.4 闰秒的处理方式
        3.4.1 linux系统上进行跳跃式调整
        3.4.2 NTP服务上进行渐进式调整的 Slew模型
4. 单调时钟
    4.1 墙上时钟是 绝对时钟
    4.2 墙上时钟的比较是有误差的 不可信。相对时钟是 相对时钟 是 自己机器的，比较没有任何意义
5. 时间的管理

Q1 计算机系统是没有时间概念的，那么他怎么来计算与 管理时间

答： 通过时钟中断 内核周期性的更新系统的墙上时间和单调时钟

Q2 计算机系统可以 提供微妙甚至 纳秒，那么他怎么处理这么高精度的时间呢？

答： 目前系统定时器的中断频率为 1000Hz，在每次计算机启动的时候 会计算一次 BogoMIPS

BogoMIPS； 处理器在给定的时间内执行指令数，通过BogoMIPS值 计算机就可以得到非常小的精度了。

需要实践：

1. 交换机如果不支持PTP 协议 提示什么（不一定能实践，先记一下）
2. 3.2.3 怎么做到防止GPS接收机的故障的

思考题：
出现的原因：系统中年份 用的2位数字表示（计算机内核的问题）
怎么解决： 新主版用4位表示年
想到的 一个类似的事件是 2038年 mysql数据库 timestamp 超出的问题（字段的问题）
解决方法： timestamp 转换到 datetime 百度就有

作者回复: 非常棒！

共 2 条评论 >

1