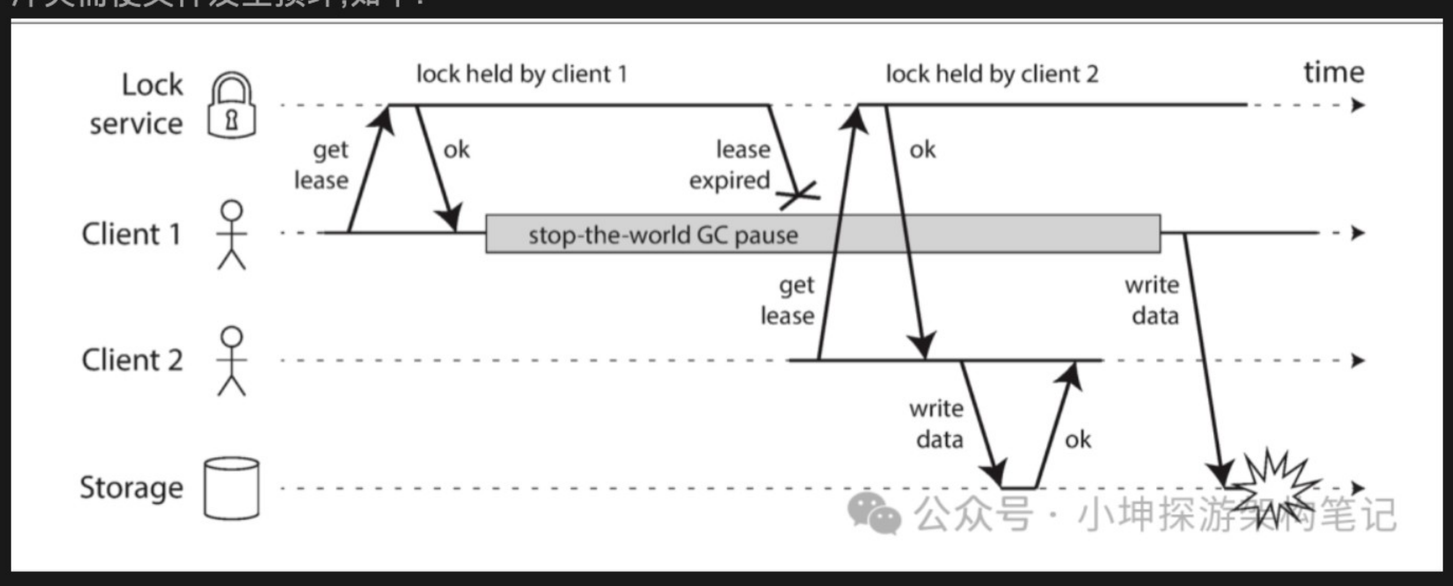
如何构建故障容忍的分布式系统

原创 疾风先生 小坤探游架构笔记 2025年05月11日 20:00 广东

在前面我们讲述基于资源利用率以及成本的权衡,大部分情况下将会采用廉价且不可靠的组件来构建我们的分布式系统,比如异步网络模型的无界延迟、时钟漂移以及进程暂停是现实中存在且不可避免的问题,那么在实践中是如何去构建一个故障容忍的分布式系统呢? 今天聊聊我的思考.

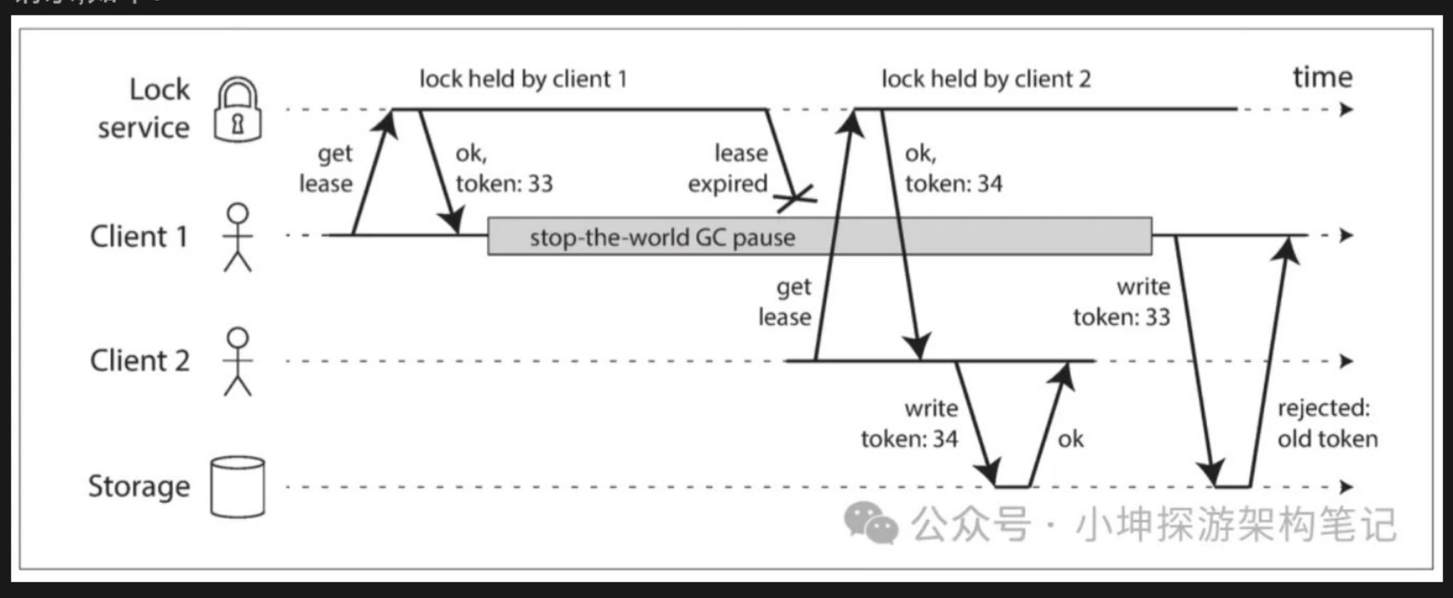
Fencing Tokens案例

这里我举一个在上一篇中我们谈到获取租约ID的例子, 其中由于发生STW导致写入同一份数据存在冲突而使文件发生损坏,如下:



那么这里我们仔细思考下, 导致我们文件发生故障的根本原因是什么呢? 在上图我们可以知道是两个客户端都持有对相同文件数据的写入锁并能够正常写入数据, 但是在Storage层并没有一个机制让它知道对于操作相同文件数据到底该以什么机制为准.

那么该怎么去解决这个问题,在这里我们会采用Fencing Tokens机制, 即在获取锁的同时引入一个Token, 这个Token是唯一且具备单调递增的机制, 实现Token的方式我们可以采用google的TrueTime的时间戳或者是逻辑时钟实现.这样当客户端往Storage写入数据的时候会携带对应的Token给到Storage并告诉他以最新的Token为准,携带的Token小于已持久化的Token直接拒绝写入请求,如下:



聊到这里我们回到现实,仔细思考下上面的故障我们业务层面能够容忍吗? 好, 我们再换个问题,上述的Lock Service如果在某个时间段T不可用, 然后又恢复可用再次提供正常获取锁的服务响应, 那么这个时候我们再思考下, 这个时间段T由于锁服务发生宕机导致故Client端一直无响应, 那么对于这个故障我们业务层面能够容忍吗?

Safety & Liveness

通过上述的Fecning Tokens例子, 我们可以看到Lock Service为锁机制提供一个生成Tokens的算法属性:

- 唯一性: 任意两次请求Lock Service获取锁返回的Tokens是不一样的.
- 单调有序性: 如果请求X获取锁返回的Tokens为Tx, 请求y返回的锁Token为Ty, 且请求X是先于请求Y, 那么一定满足Tx < Ty.
- 可用性: 若一个节点请求Fecning Tokens且未发生崩溃, 则该节点最终将收到响应.

在上述的算法属性中,唯一性以及单调有序性是分布式系统的Safety属性, 而可用性是指分布式系统的Liveness属性.安全性通常被定义为“不会发生糟糕的事情”，而活性则被定义为“最终会发生好的事情”。然而，最好不要对这些不正式的定义过度解读，因为“好”与“坏”的含义是主观的。安全性和活性的实际定义是精确且基于数学的:

> 安全性: 如果违背了系统安全性, 那么对于系统的损坏是无法撤销的, 因为损坏已经造成, 比如上述例子中没有引入Token导致文件损坏, 这种损坏是无法撤销且需要我们工程师介入修复处理.

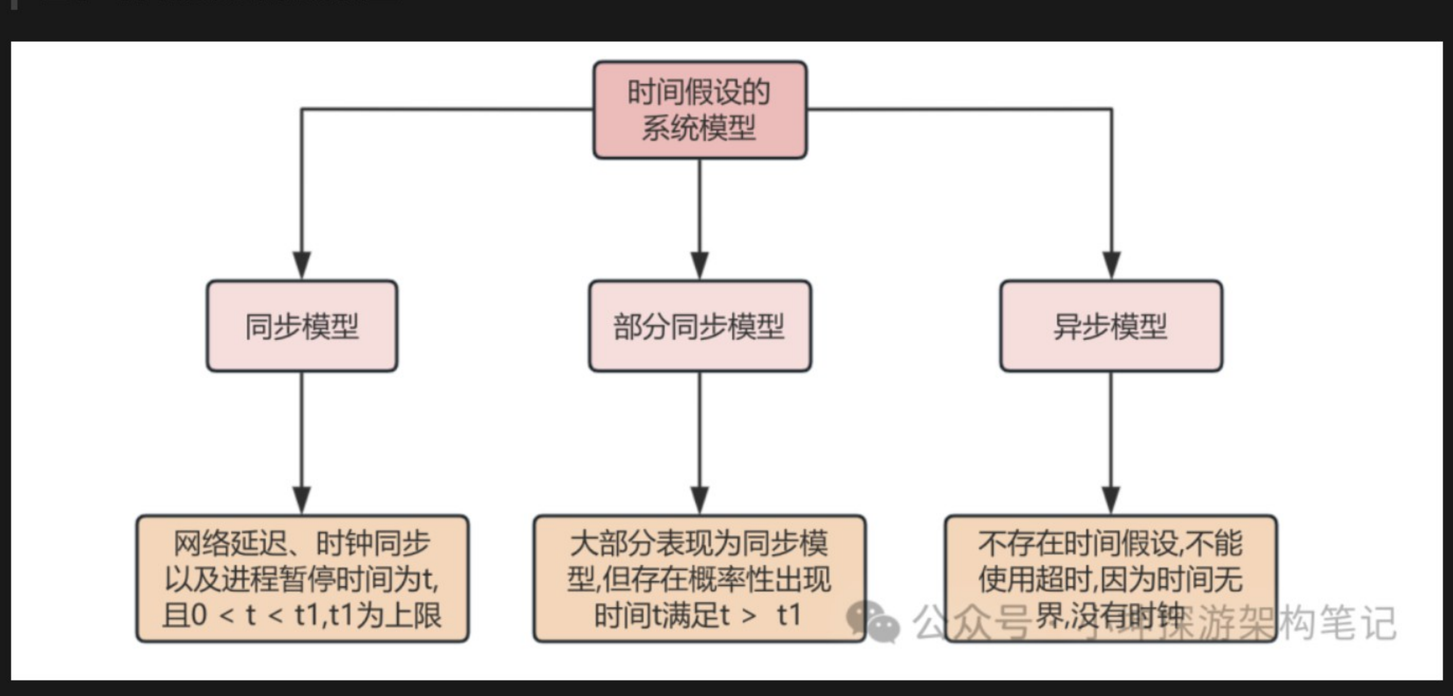
> 活性: 系统在某一时间段内无法满足,但是经过有限时间之后它将会恢复重新提供服务. 比如上述例子节点发生crash然后重启, 在此期间有5s左右的时间将无法提供服务.

这个时候我们再回到上述的例子,我们要设计一个Lock Service的分布式系统,我们要优先保证safety属性,即在设计上要保证Tokens的生成具备唯一性以及单调有序性的情况下,再考虑我们的可用性,因为对外不可用并不影响数据文件被损坏的情况,因此这里我们构建一个具备故障容忍的Lock Service服务, 这里的故障容忍是指能够容忍服务的不可用,但在实际运行中,我们的服务大部分时候都是可用的,因为故障总是能够在有限时间T恢复回去的.

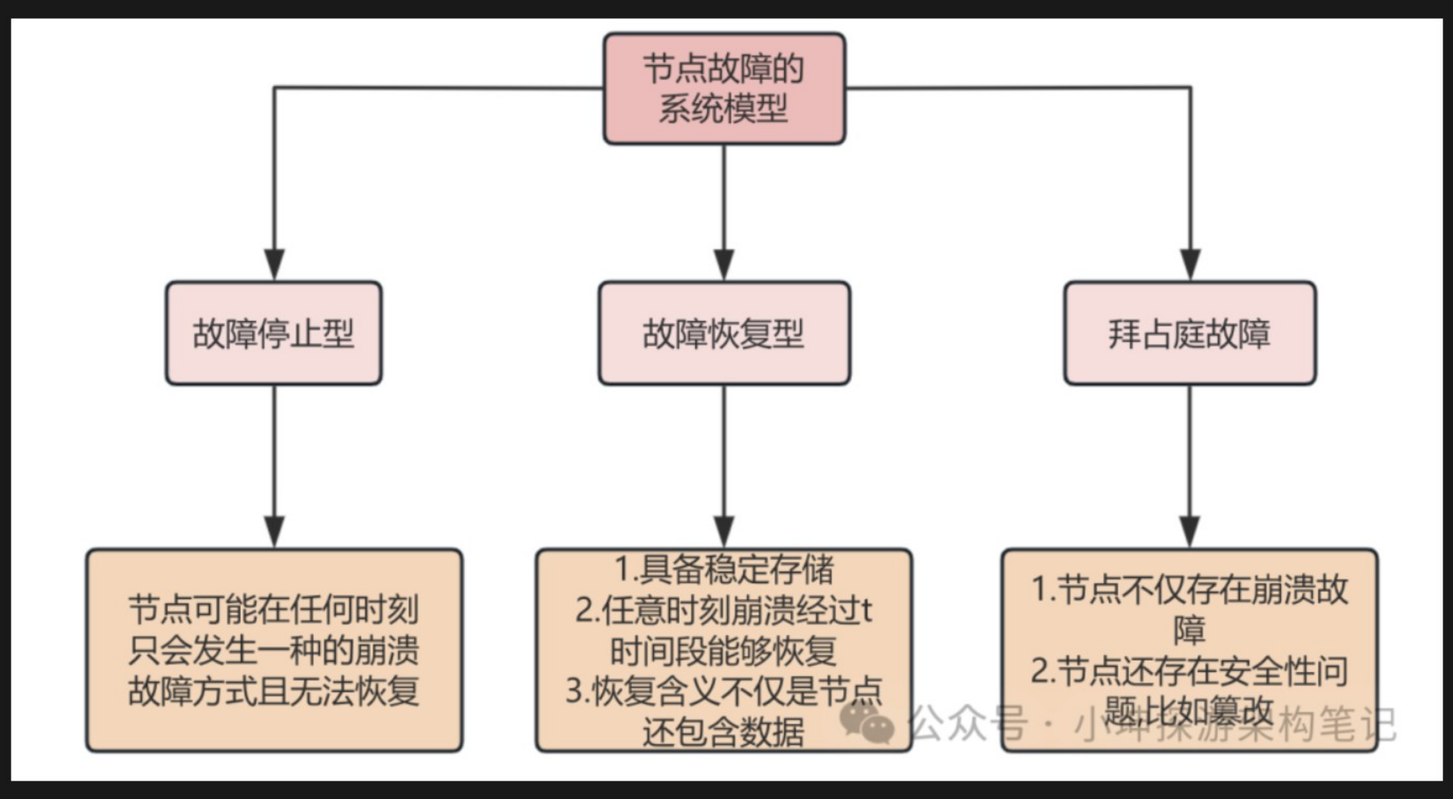
分布式抽象系统模型

通过上述例子,我们的分布式系统问题可以通过制定对应的算法策略来解决.比如我们要构建一个具备故障容忍的分布式系统, 那么我们设计的算法就需要容忍对应的故障,算法并不依赖于软硬件的配置细节,而是通过将分布式系统抽象为一个系统模型来将各类故障进行形式化,并基于系统模型的假设前提下**考量算法设计的正确性, 即满足安全性以及活性属性**. 一般地, 我们会将分布式系统抽象为以下两大类系统模型:

基于时间假设的系统模型



基于节点故障的系统模型



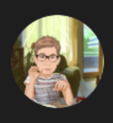
之前在架构建模曾讲过, 模型只是我们对现实世界的抽象, 是一项用于辅助我们在设计系统过程中识别问题复杂度的工具.比如上述故障恢复模型的算法, 如果我们将数据存在在内存中, 那么当其中的节点服务发生故障时候,我们的数据就消失了;

如果我们将数据存储磁盘中，那么当磁盘也发生故障的时候怎么办呢？可能我们会采用RAID阵列磁盘来或者使用SAN、NAS等来保证.但是即便我们给出策略保证数据可靠性,我们也会面临着数据的一致性问题,而数据一致性问题又可以拆分强一致性以及弱一致性(或者称最终一致性),在这里我把强一致性归为属于安全性属性,而弱一致性归为活性属性.

总结

因此，区分安全性以及活性属性有助于我们应对复杂的分布式系统模型,因为我们在构建一个具备故障容忍的分布式系统过程中必须要识别到哪些故障是我们业务层面是无法接受容忍的,哪些故障是允许我们接受容忍的,这就要求我们在进行架构设计过程能够识别到系统的安全性以及活性属性，并在两者之间进行权衡取舍然后做出对应的架构决策.

你好,我是疾风先生，主要从事互联网搜广推行业，技术栈为java/go/python，记录并分享个人对技术的理解与思考，欢迎关注我的公众号，致力于做一个有深度,有广度,有故事的工程师,欢迎成长的路上有你陪伴,关注后回复greek可添加私人微信,欢迎技术互动和交流,谢谢!



小坤探游架构笔记

10年后端技术架构设计 | AI工程化基础建设 & 存储架构设计 & 性能优化 | 前网易、斗鱼、i...

52篇原创内容

公众号

分布式架构 · 目录

< 上一篇
分布式系统不可靠时钟问题

下一篇 >
共识建立的艺术：揭秘如何巧妙避开FLP定理

个人观点，仅供参考