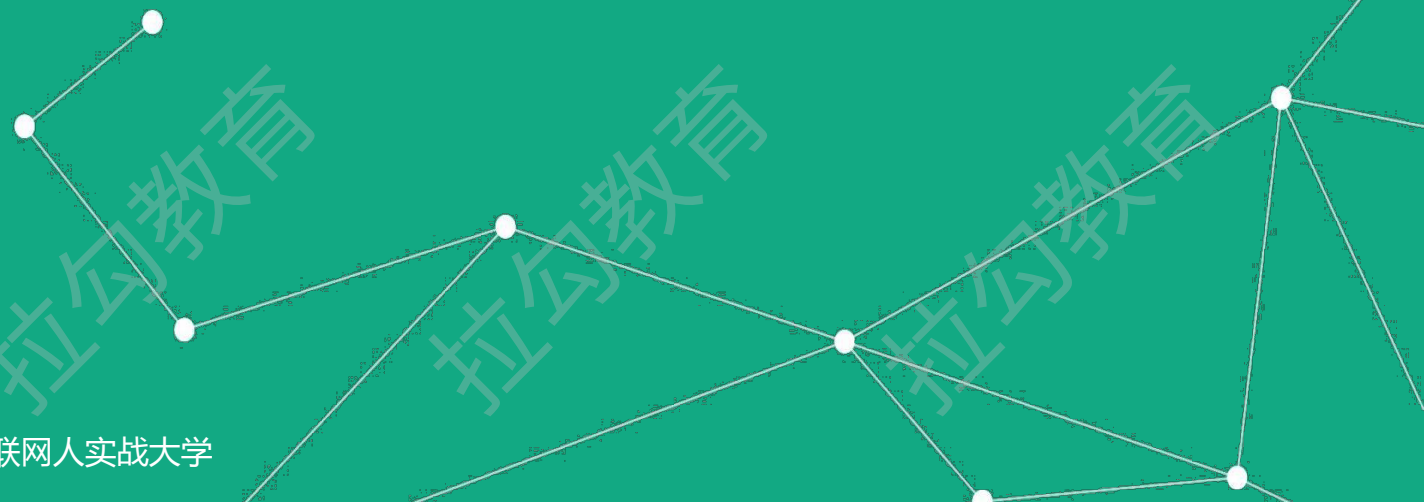
课时 04

链表在 Apache Kafka 中的应用

1. 如何重新设计定时器算法
2. 维护无序定时器列表
3. 维护有序定时器列表
4. 维护定时器“时间轮”
5. “时间轮”变种算法
6. Apache Kafka 的 Purgatory 组件



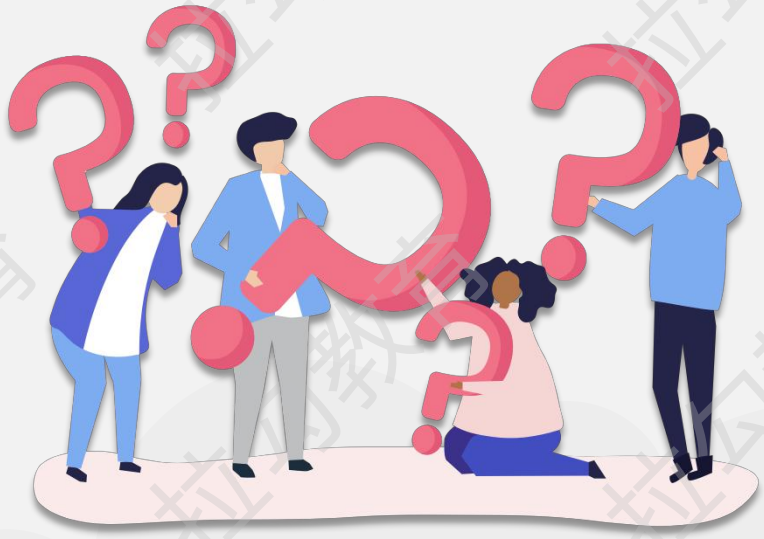
如何重新设计定时器算法

定时器（Timer）

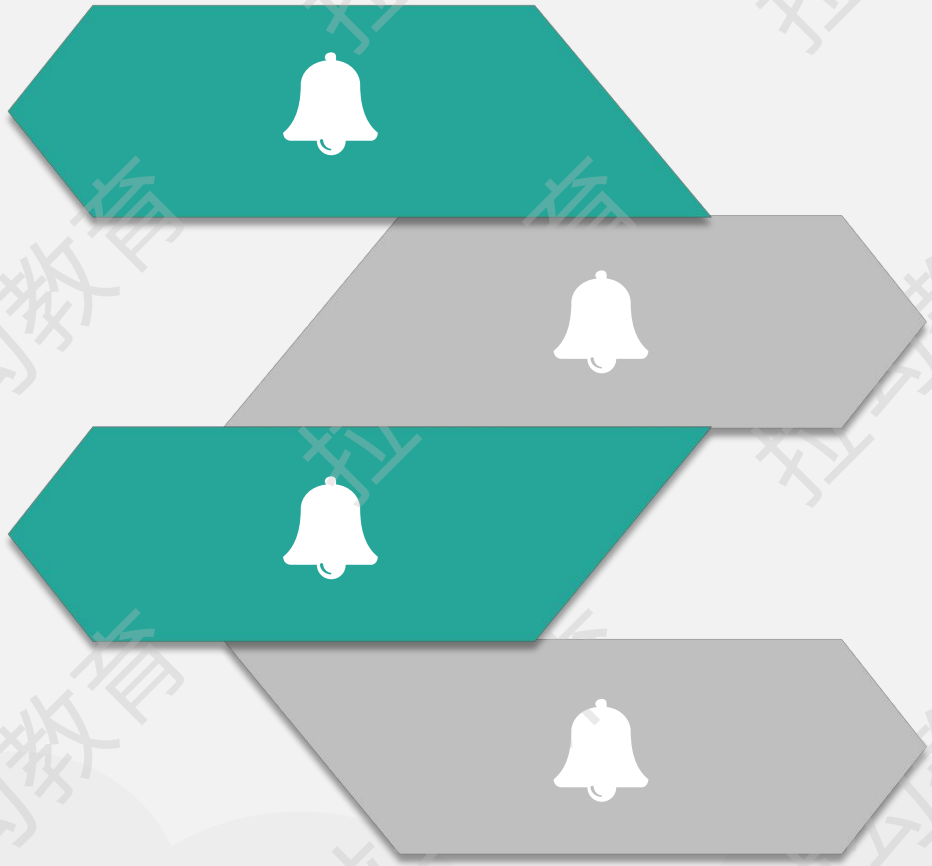
像在写程序时使用到的 Java Timer 类，或者是在 Linux 中制定定时任务时所使用的 cron 命令

亦或是在 BSD TCP 网络协议中检测网络数据包是否需要重新发送的算法里，其实都使用了定时器这个概念

如果重新设计定时器算法，该如何设计呢???



如何重新设计定时器算法



初始化定时器

删除定时器

定时器超时进程

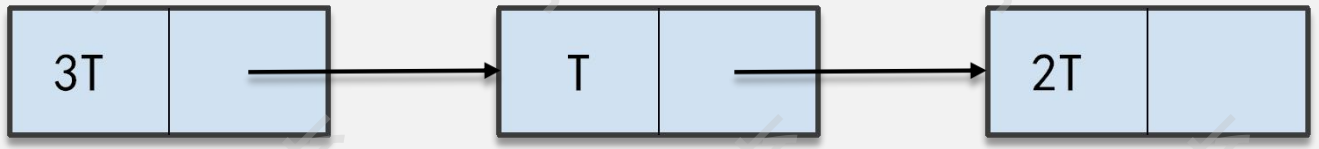
定时器检测进程

维护无序定时器列表

在数组中插入一个新的元素所需要的时间复杂度是 O(N)

而在链表的结尾插入一个新的节点所需要的时间复杂度是 O(1)

所以在这里可以选择用链表来维护定时器列表



维护无序定时器列表

在数组中插入一个新的元素所需要的时间复杂度是 O(N)

而在链表的结尾插入一个新的节点所需要的时间复杂度是 O(1)

所以在这里可以选择用链表来维护定时器列表

又插入了一个新定时器，将会在 T 时间后超时，会将新的定时器数据结构插入到链表结尾

维护有序定时器列表

一

每次插入一个新的定时器

并不是将它插入到链表的结尾，

而是从头遍历一遍链表将定时器

的超时时间按从小到大的顺序插

入到定时器列表中

不同点

二

每次插入新定时器时

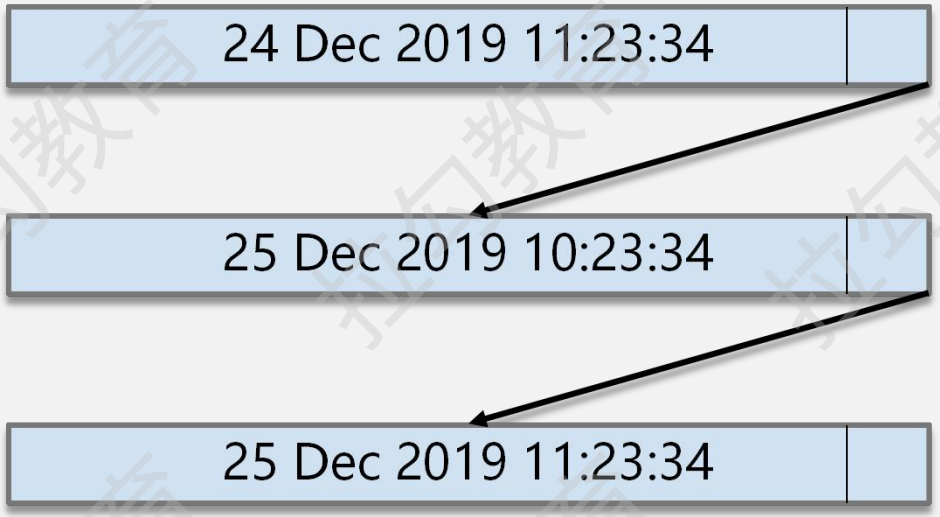
并不是保存超时时间，而是根据

当前系统时间和超时时间算出一

个绝对时间出来

维护有序定时器列表

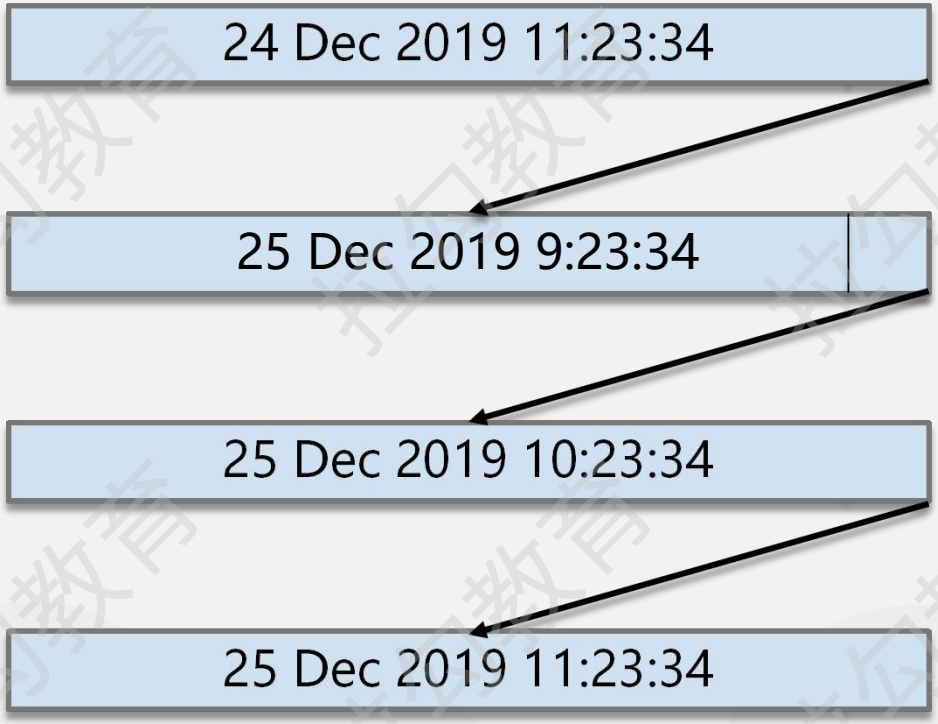
假设原来的有序定时器列表



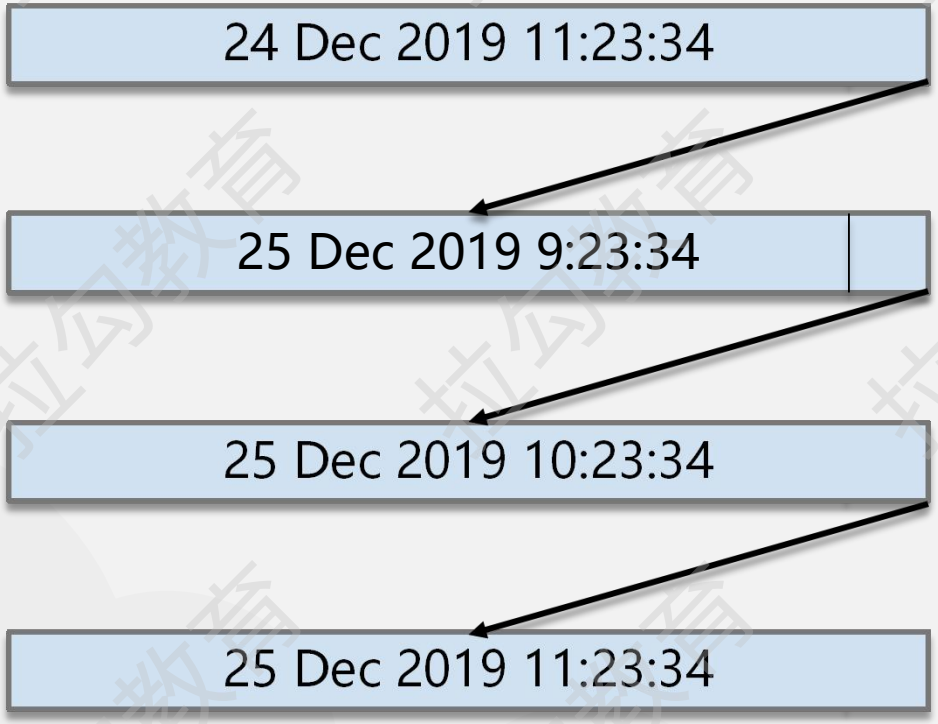
维护有序定时器列表

插入一个新的定时器

超时的绝对时间算出为 25 Dec 2019 9:23:34



维护有序定时器列表



好处

执行定时器检测进程和删除定时器的时间复杂度为 O(1)，但因为要按照时间从小到大排列定时器，每次插入的时候都需要遍历一次定时器列表，所以插入定时器的时间复杂度为 O(N)

维护定时器“时间轮”

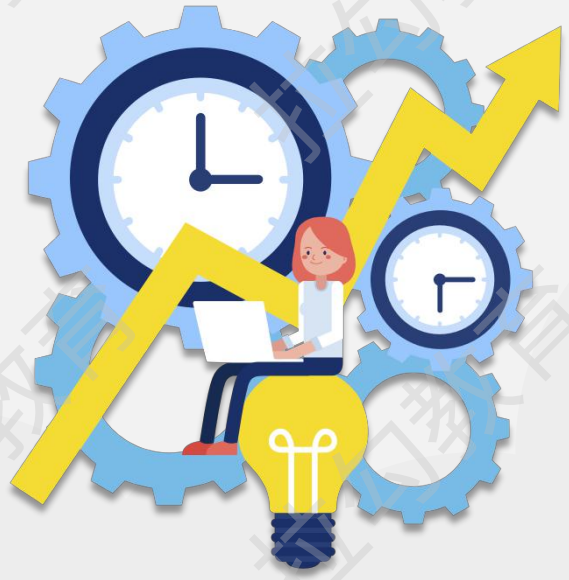
“时间轮”（Timing-wheel ）

在概念上是一个用数组并且数组元素为链表的数据结构来维护的定时器列表

常常伴随着溢出列表（Overflow List）来维护那些无法在数组范围内表达的定时器

基本的“时间轮”会将定时器的超时时间划分到不同的周期（Cycle）中去

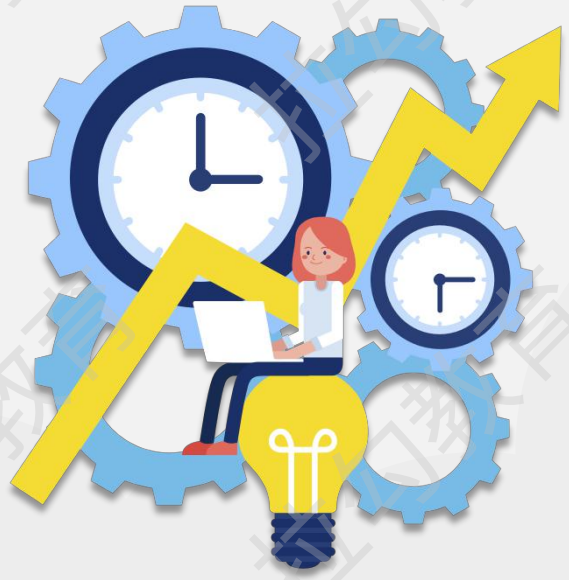
数组的大小决定了一个周期的大小



维护定时器“时间轮”

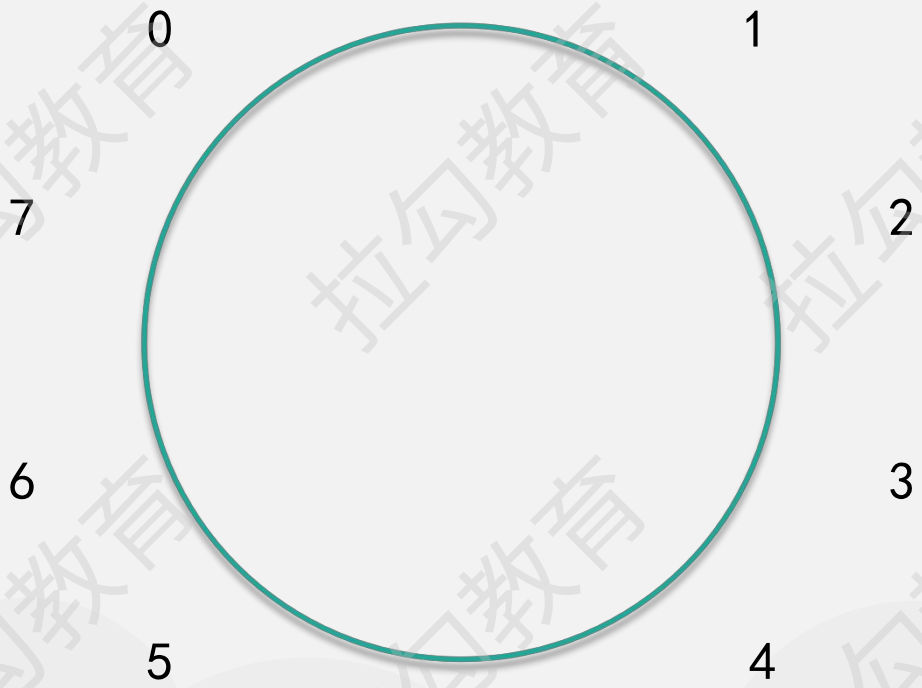
维护一个最基本的“时间轮”还需要维护以下几个变量：

- “时间轮”的周期数，用 S 来表示
- “时间轮”的周期大小，用 N 来表示
- “时间轮”数组现在所指向的索引，用 i 来表示



维护定时器“时间轮”

将数组想象成一个环



维护定时器“时间轮”

假设现在的时间是 S×N + 2，表示这个“时间轮”的当前周期为 S，数组索引为 2

同时假设这个“时间轮”已经维护了一部分定时器链表

维护定时器“时间轮”

如果我们想新插入一个超时时间为 T 的新定时器进这个时间轮，因为 T 小于这个“时间轮”周期的大小 8T，所以表示这个定时器可以被插入到当前的“时间轮”中，插入的位置为当前索引为 1 + 2 % 8 = 3 ，插入新定时器后的“时间轮”如下

维护定时器“时间轮”

还需等待的周期：9T / 8T = 1

新定时器插入时的索引位置：(9T + 2T) % 8T = 3

“时间轮”变种算法

基本的“时间轮”插入操作因为维护了一个溢出列表导致定时器的插入操作无法做到 O(1) 的时间复杂度

所以为了 O(1) 时间复杂度的插入操作，一种变种的“时间轮”算法就被提出了

在这个变种的“时间轮”算法里，我们加了一个 MaxInterval 的限制

这个 MaxInterval 其实也就是我们定义出的“时间轮”数组的大小

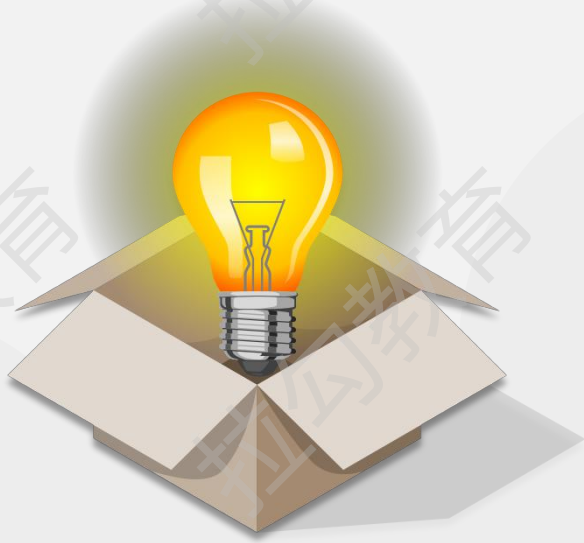
Apache Kafka 的 Purgatory 组件

Apache Kafka 是一个开源的消息系统项目，主要用于提供一个实时处理消息事件的服务

与计算机网络里面的 TCP 协议需要用到大量定时器来判断是否需要重新发送丢失的网络包一样

在 Kafka 里面，因为它所提供的服务需要判断所发送出去的消息事件是否没有被订阅消息的用户接受到

Kafka 也需要用到大量的定时器来判断发出的消息是否超时然后重发消息



Apache Kafka 的 Purgatory 组件

这个任务就落在了 Purgatory 组件上

在旧版本里，维护定时器的任务采用的是 Java 的 DelayQueue 类来实现的

因为 Kafka 中所有的最大消息超时时间都已经被写在了配置文件里

即新版本的 Purgatory 组件则采用的了上面所提到的变种“时间轮”算法，将插入定时器的操作性能大大提升

根据 Kafka 所提供的检测结果

采用 DelayQueue 时所能处理的最大吞吐率为 25000 RPS

采用了变种“时间轮”算法之后，最大吞吐率则达到了 105000 RPS