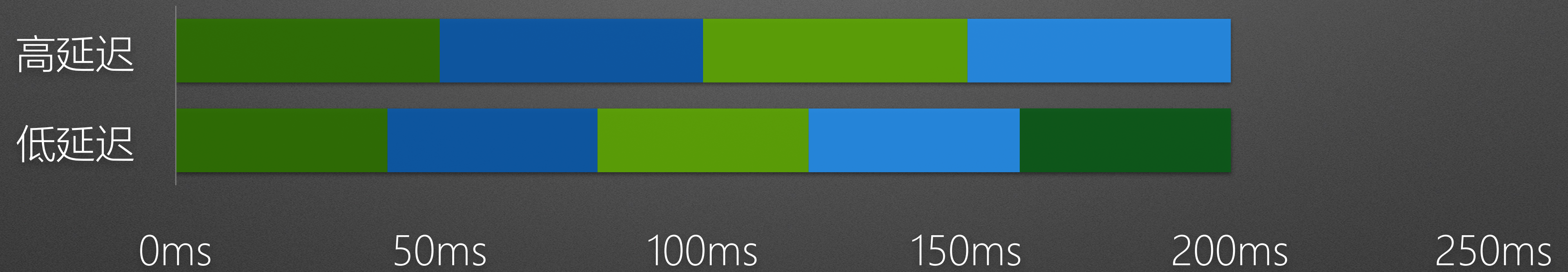


# 低延迟服务开发之路

莫伟强@唯品会



# 服务的延迟



~23 ms

-50 ms

1 s



# 低延迟服务的挑战

**Garbage Collection**





世界停顿了2秒的思考  
——GC并不是一个黑盒子



# Stop the world

Young (ParNew)



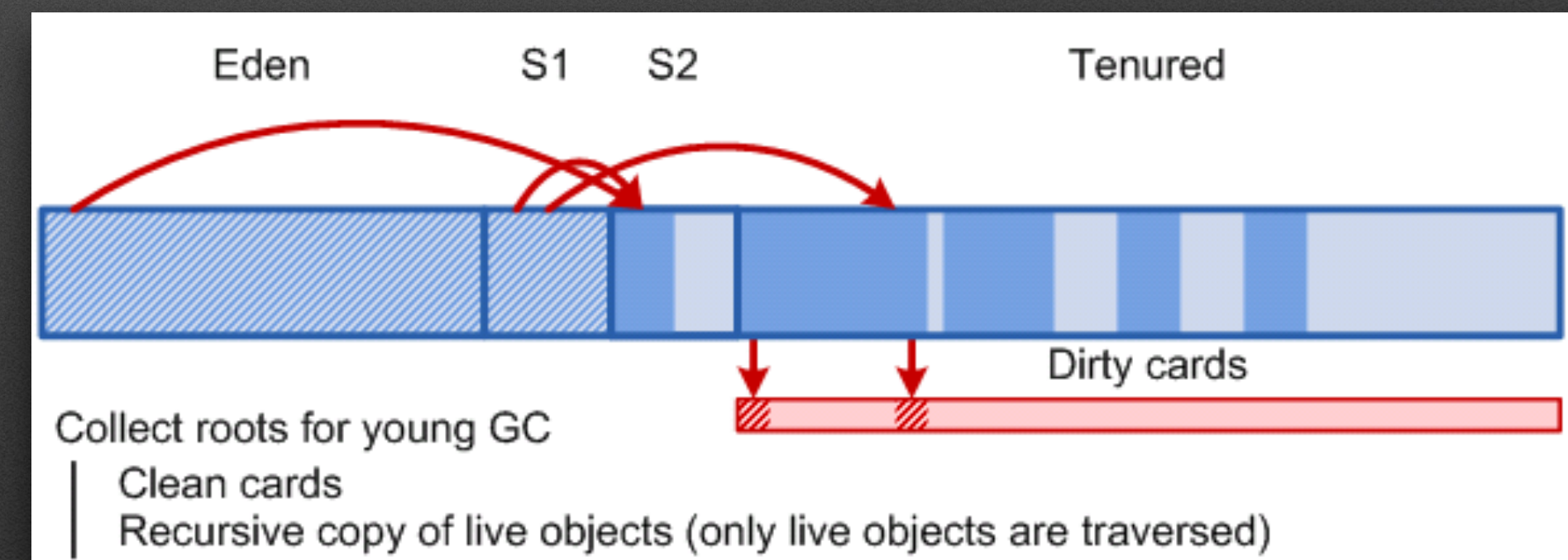
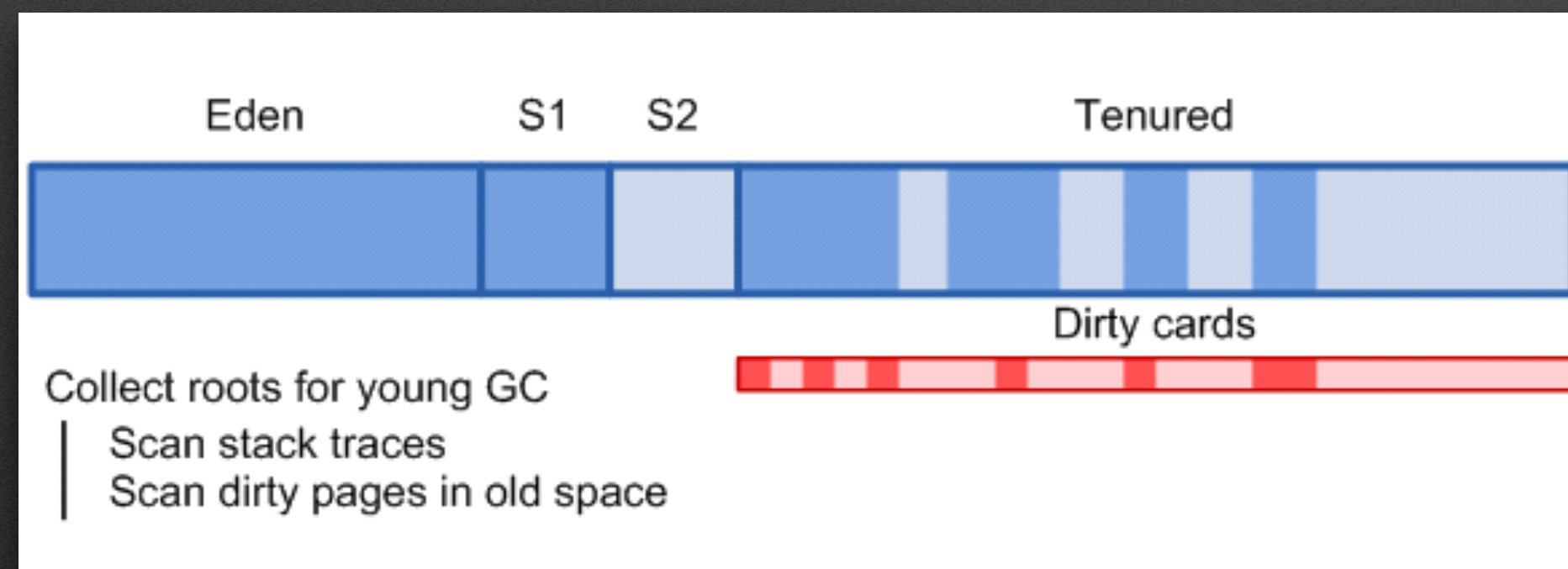
Tenured (CMS)





# ParNew GC

$$T_{\text{young}} = \underbrace{T_{\text{stack\_scan}} + T_{\text{card\_scan}} + T_{\text{old\_scan}}}_{\text{}} + \underbrace{T_{\text{copy}}}_{\text{=====}}$$





# CMS GC

- Stop the world
  - Initial Mark
  - Final Remark
- 秒级的CPU占用
  - Concurrent-mark
  - Concurrent-sweep
- 内存碎片

```
[GC (CMS Initial Mark) [1 CMS-initial-mark 3941842K(6291456K)] 431493
Total time for which application threads were stopped: 0.0395261 secor
[CMS-concurrent-mark-start]
[CMS-concurrent-mark: 0.115/0.115 secs] [Times: user=0.80 sys=0.01, re
[CMS-concurrent-preclean-start]
[CMS-concurrent-preclean: 0.017/0.017 secs] [Times: user=0.05 sys=0.00
[CMS-concurrent-abortable-preclean-start]
preclean due to time 2017-01-25T21:00:53.860+0800: 1236921.690: [CMS-c
[GC CMS Final Remark [YG occupancy: 1046353 K (5505024 K)]2017-01-25
3.951+0800: 1236921.782: [class unloading, 0.0666695 secs]2017-01-25T21:
6291456K)] 4988196K(11796480K), 0.1714246 secs] [Times: user=1.60 sys=0.
Total time for which application threads were stopped: 0.1759287 secor
[CMS-concurrent-sweep-start]
Total time for which application threads were stopped: 0.0044414 secor
Total time for which application threads were stopped: 0.0050080 secor
Total time for which application threads were stopped: 0.0043149 secor
[CMS-concurrent-sweep: 4.746/4.760 secs] [Times: user=6.18 sys=0.16,
[CMS-concurrent-reset-start]
[CMS-concurrent-reset: 0.020/0.020 secs] [Times: user=0.02 sys=0.00, ]
```



# 降低GC的负担

短寿命的小对象

及早离开作用域——设为Null?

减少大对象创建——中间对象

长寿对象的摇篮——缓存数据



# 缓存是一柄双刃剑

- 降低读取延迟

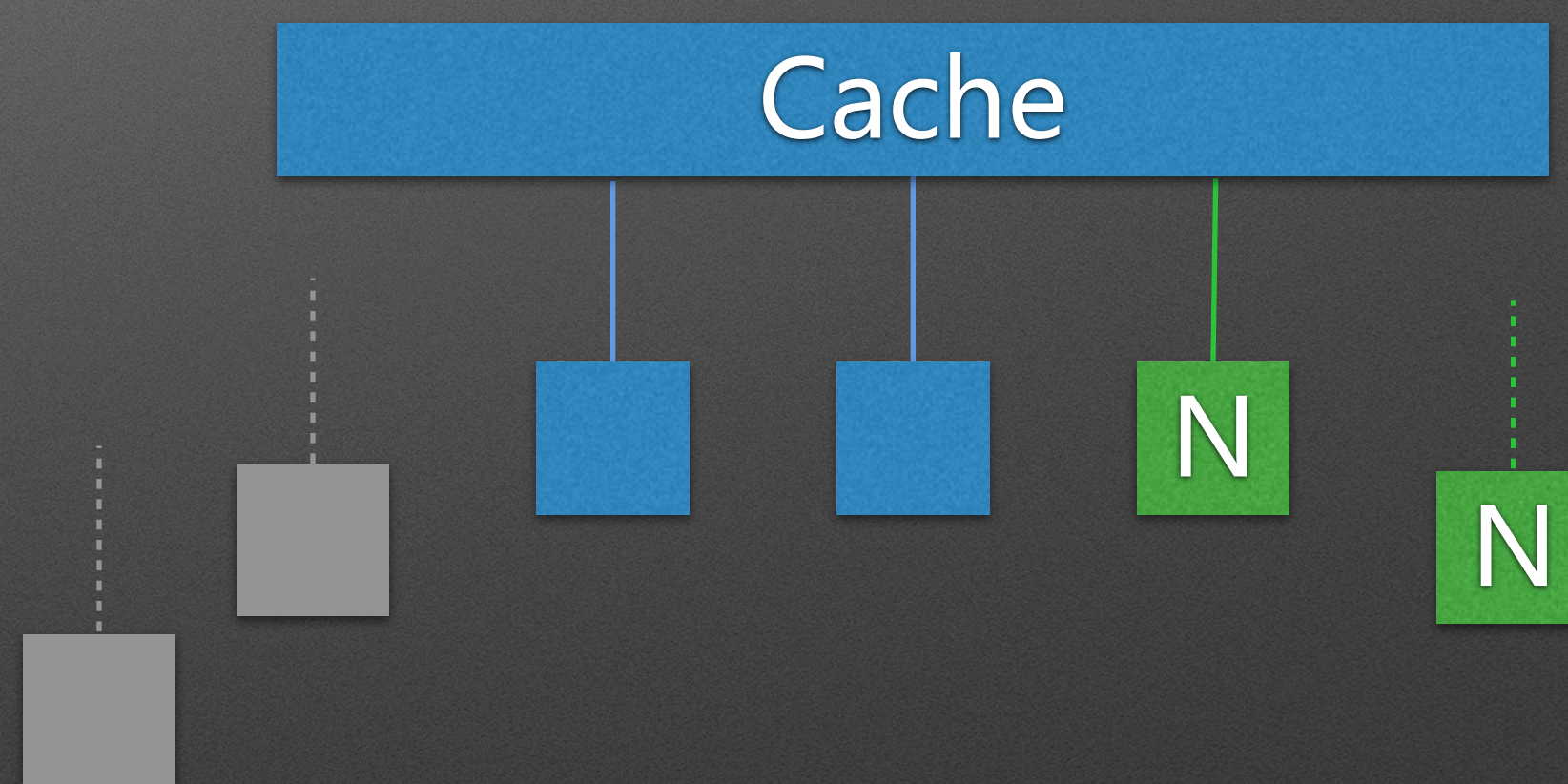


- 提升新生代复制
- 增大老年代空间
- 增加Old GC频率



# 热点数据

- 大缓存量提高命中率
- 高淘汰量提高利用率





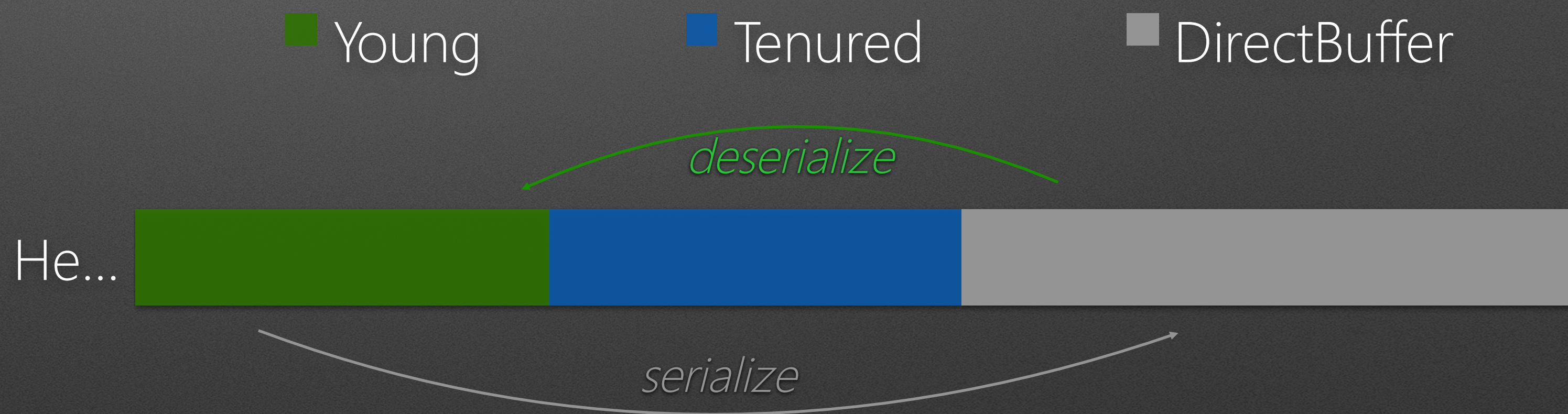
# 堆外内存

初衷

- 不被移动的内存区

副作用

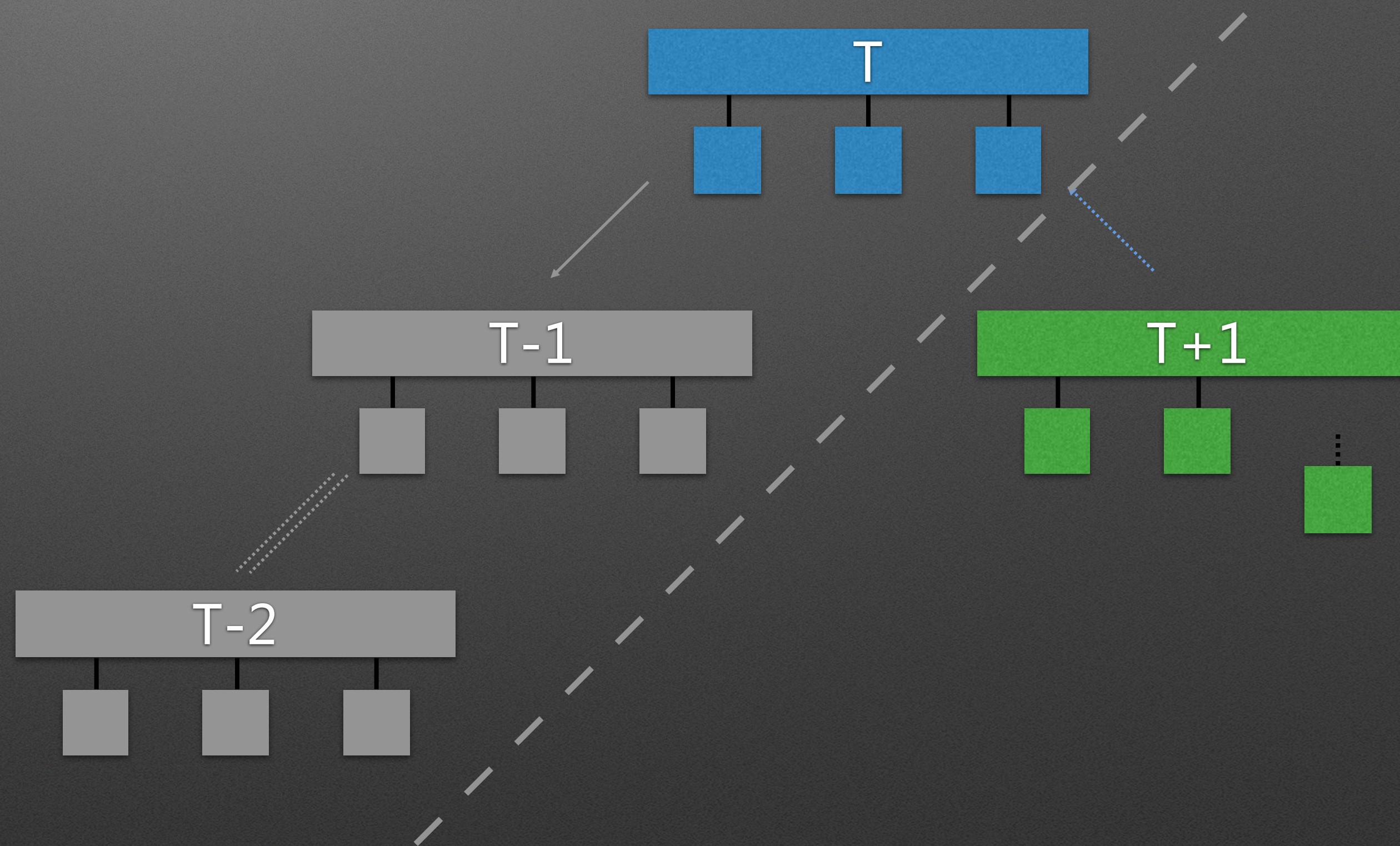
- 额外CPU开销
- 高Eden的填充速度





# 定时置换

- 无锁读取
- 全量置换
- 短时 $T_{copy}$ 高峰
- 伴随Old GC
- 同步延迟

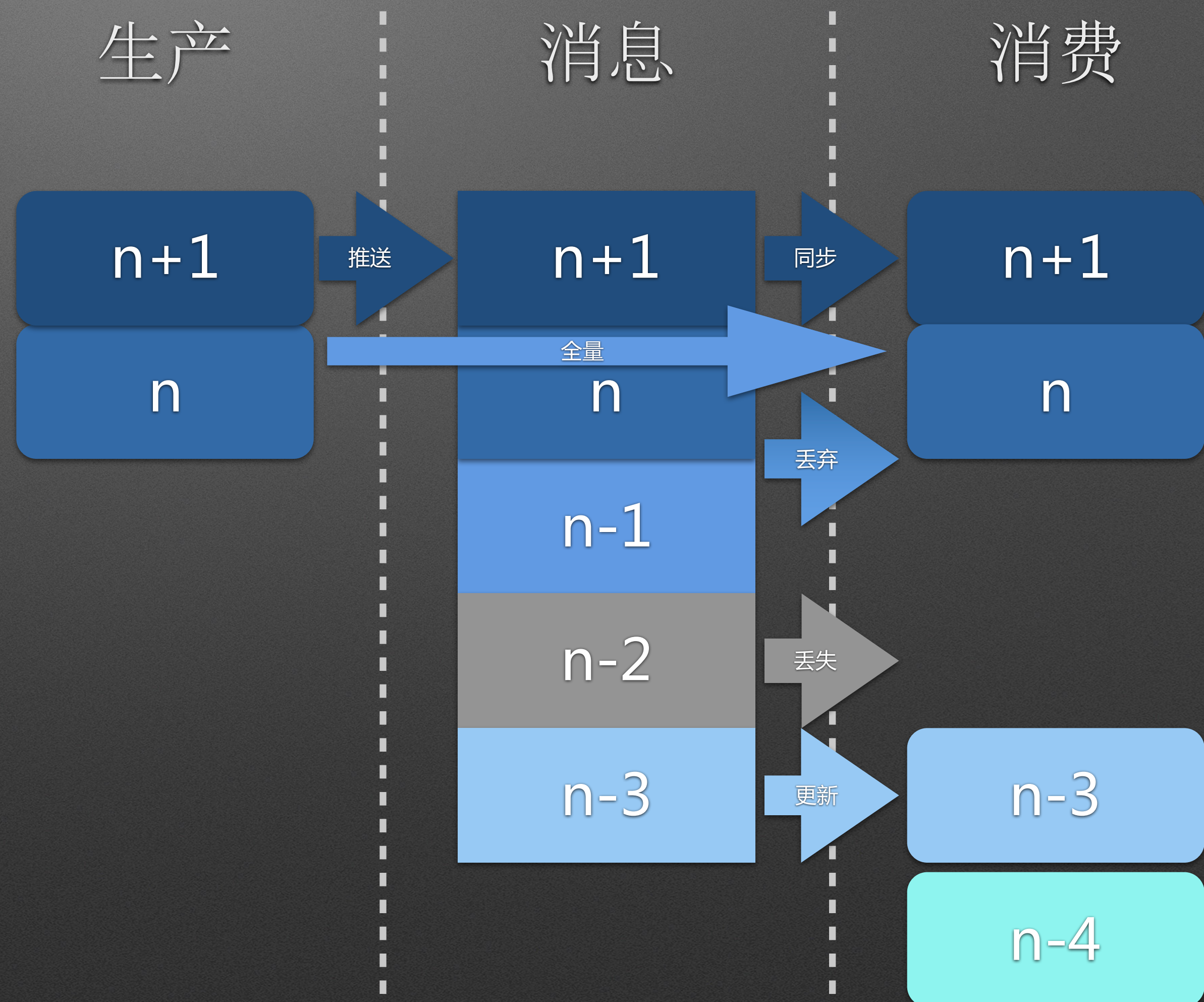




# 增量更新

## 可靠性保证

- 全量快照同步
- 消息版本验证
- 差异对比服务
- 独立检验机制



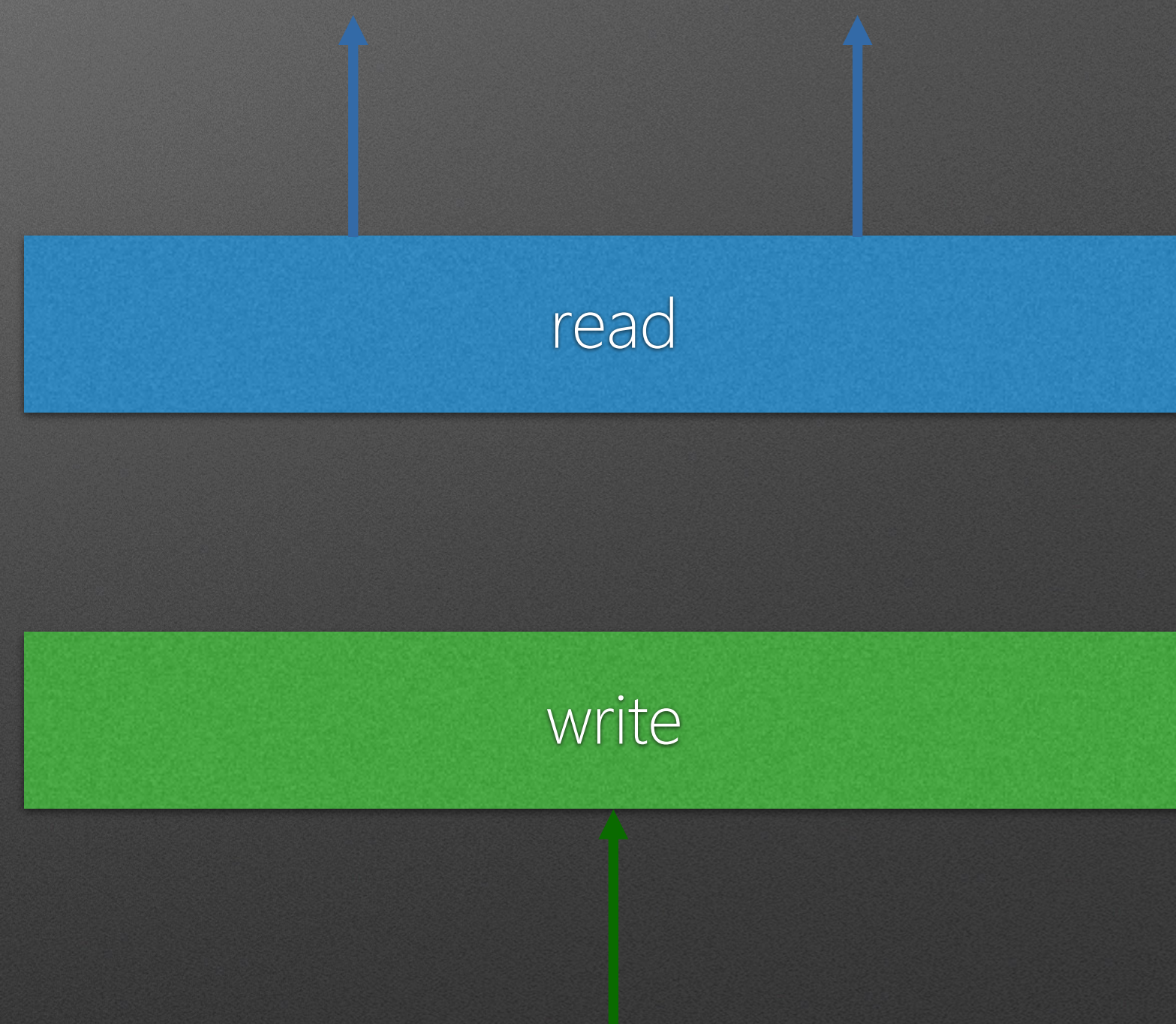


# 对象重用



# 缓存对象重用

## 1. 缓存集合重用

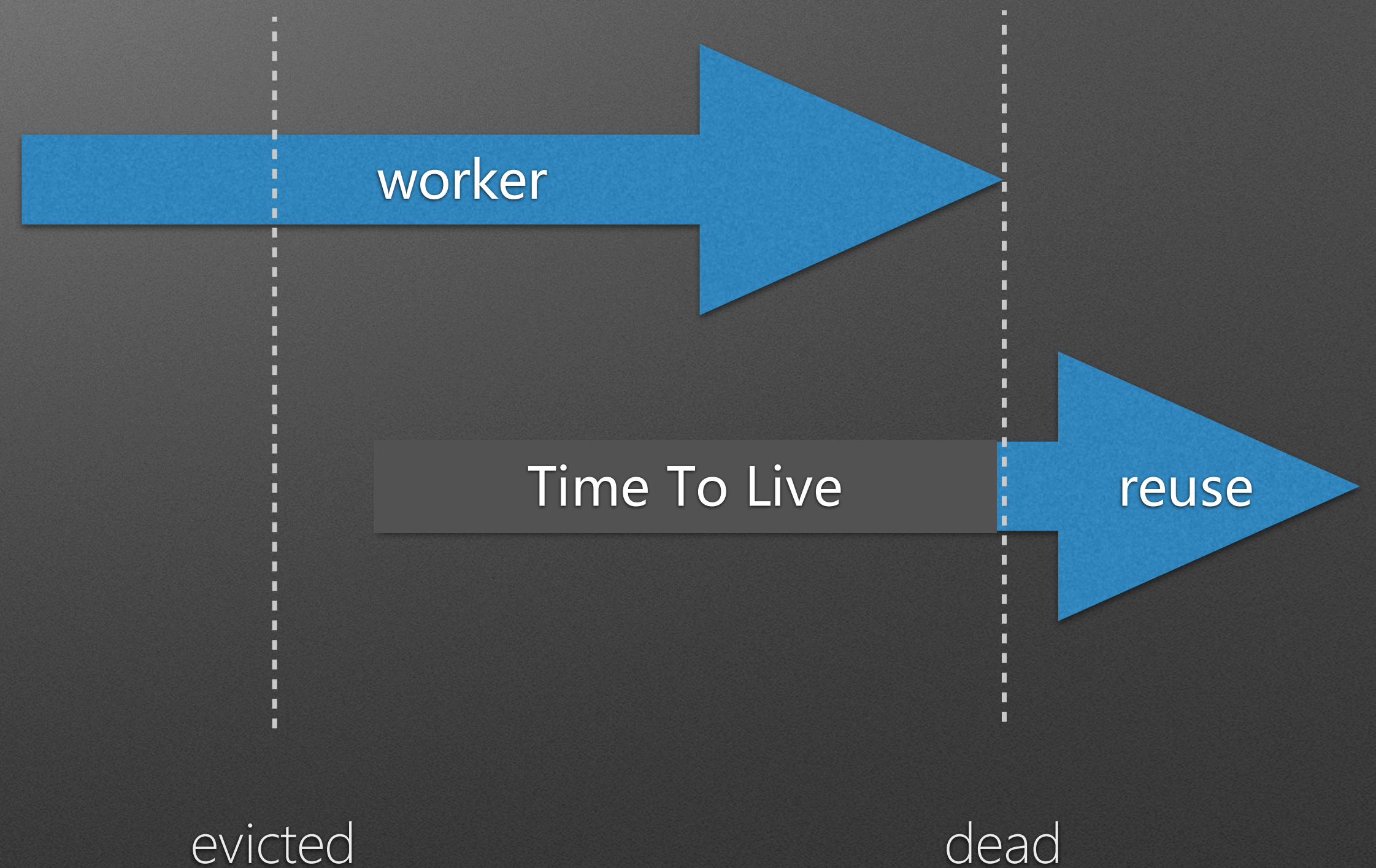




# 缓存对象重用

## 2. 缓存对象重用

- 捕获淘汰对象
- 使用前重置
- 重置前的安全时间

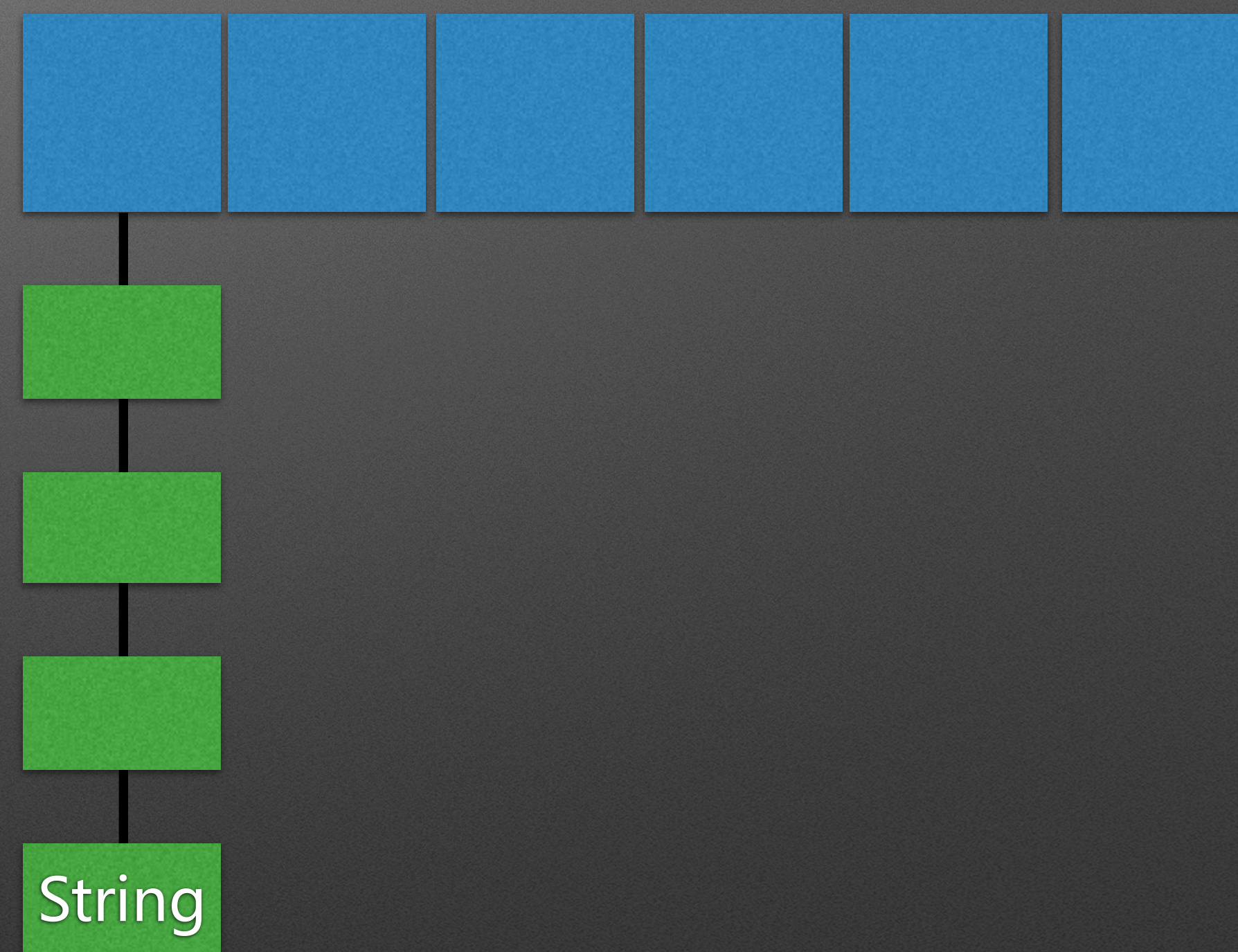




# 缓存对象重用

## 3. 字符串对象重用

× String.intern





# 回顾

- GC会影响低延迟服务的可用性
- 影响GC停顿的一些因素
- 降低缓存更新对GC压力压力策略



Q & A