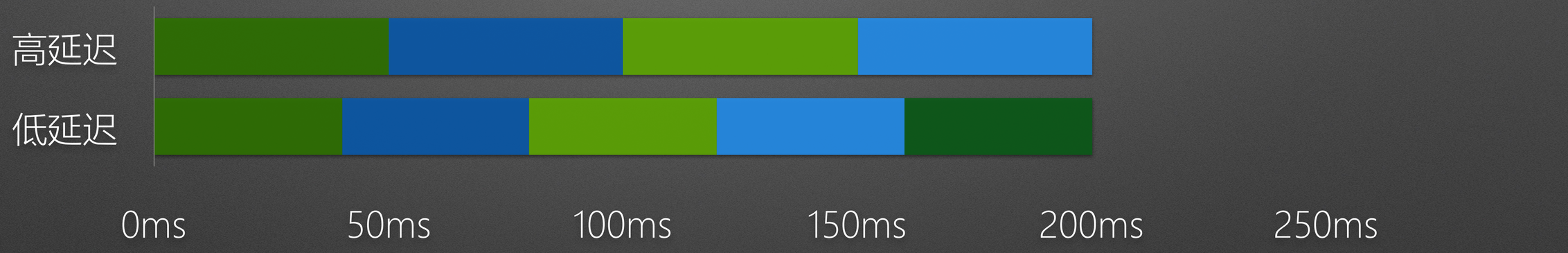
低延迟服务开发之路

莫伟强@唯品会

服务的延迟



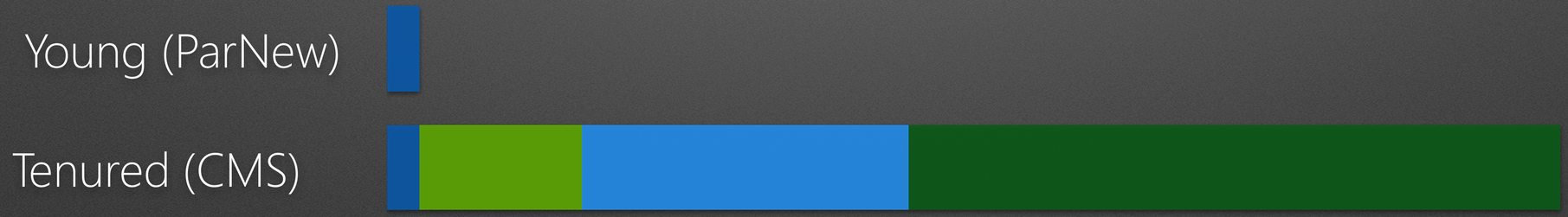
~23 ms
-50 ms
1 s

低延迟服务的挑战

Garbage Collection

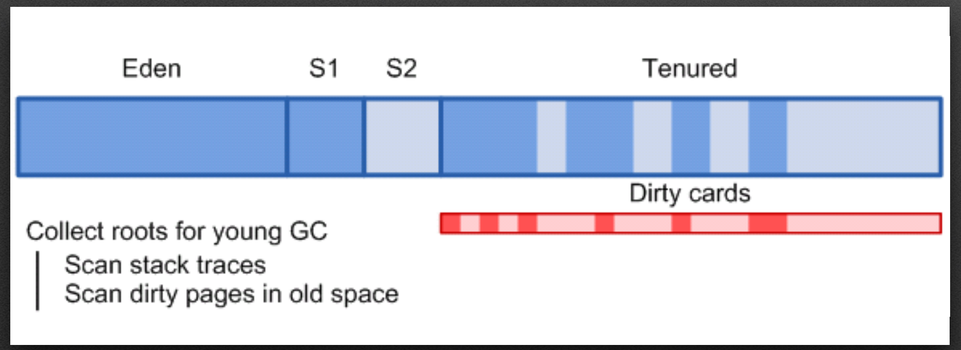

世界停顿了2秒的思考
——GC并不是一个黑盒子

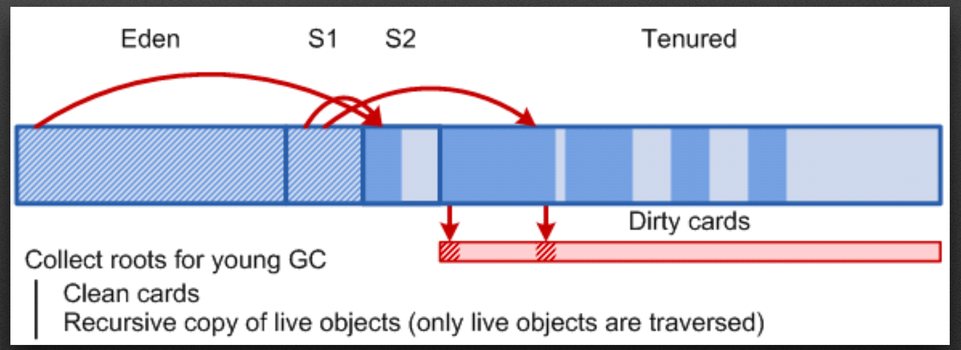
Stop the world



ParNew GC

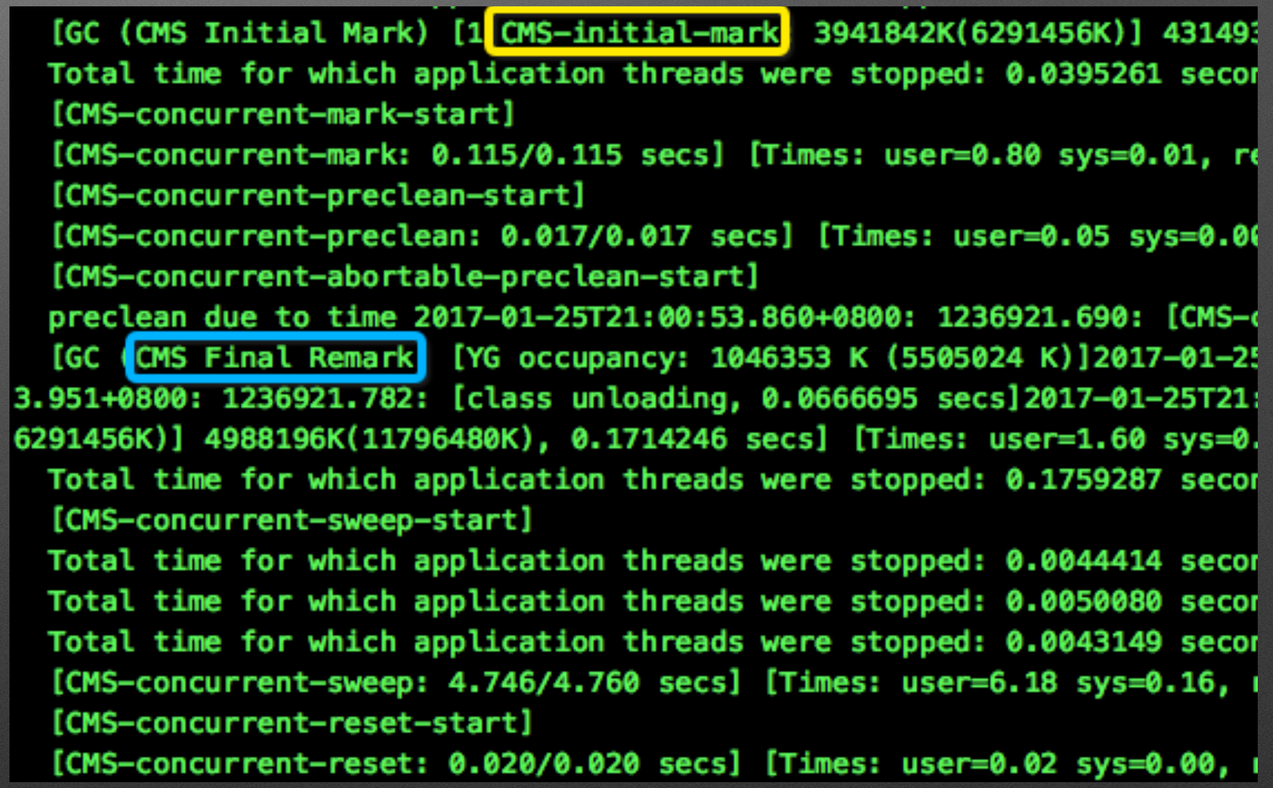
$$T_{young} = T_{stack\_scan} + T_{card\_scan} + T_{old\_scan} + T_{copy}$$





CMS GC

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



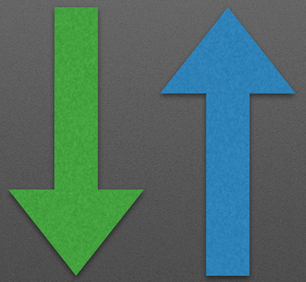
降低GC的负担

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

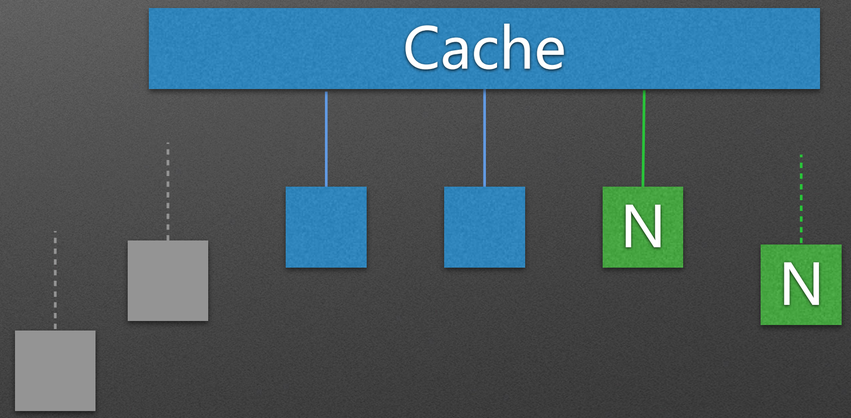
- 降低读取延迟



- 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



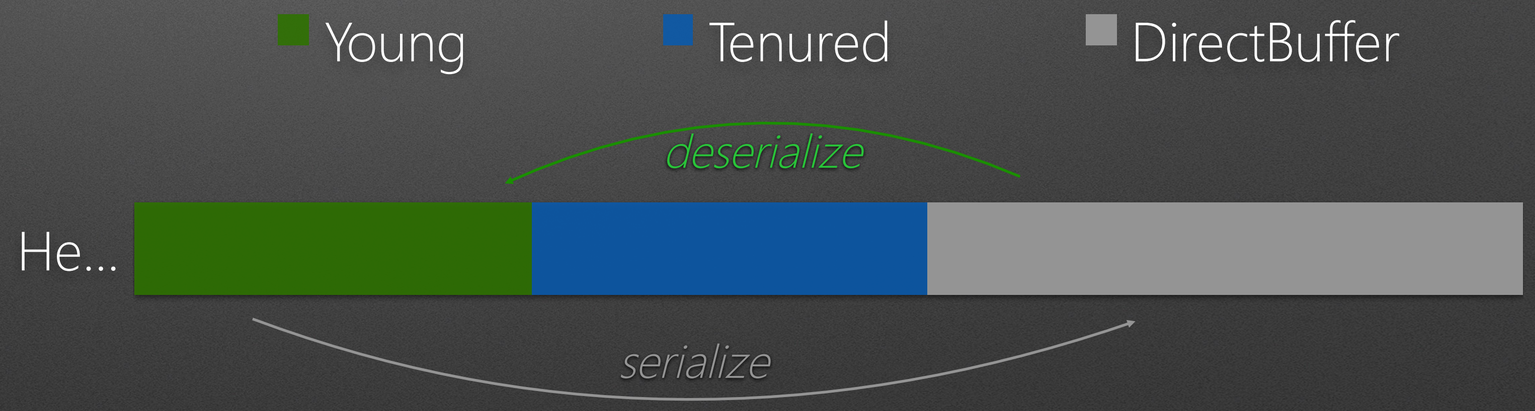
堆外内存

初衷

- 不被移动的内存区

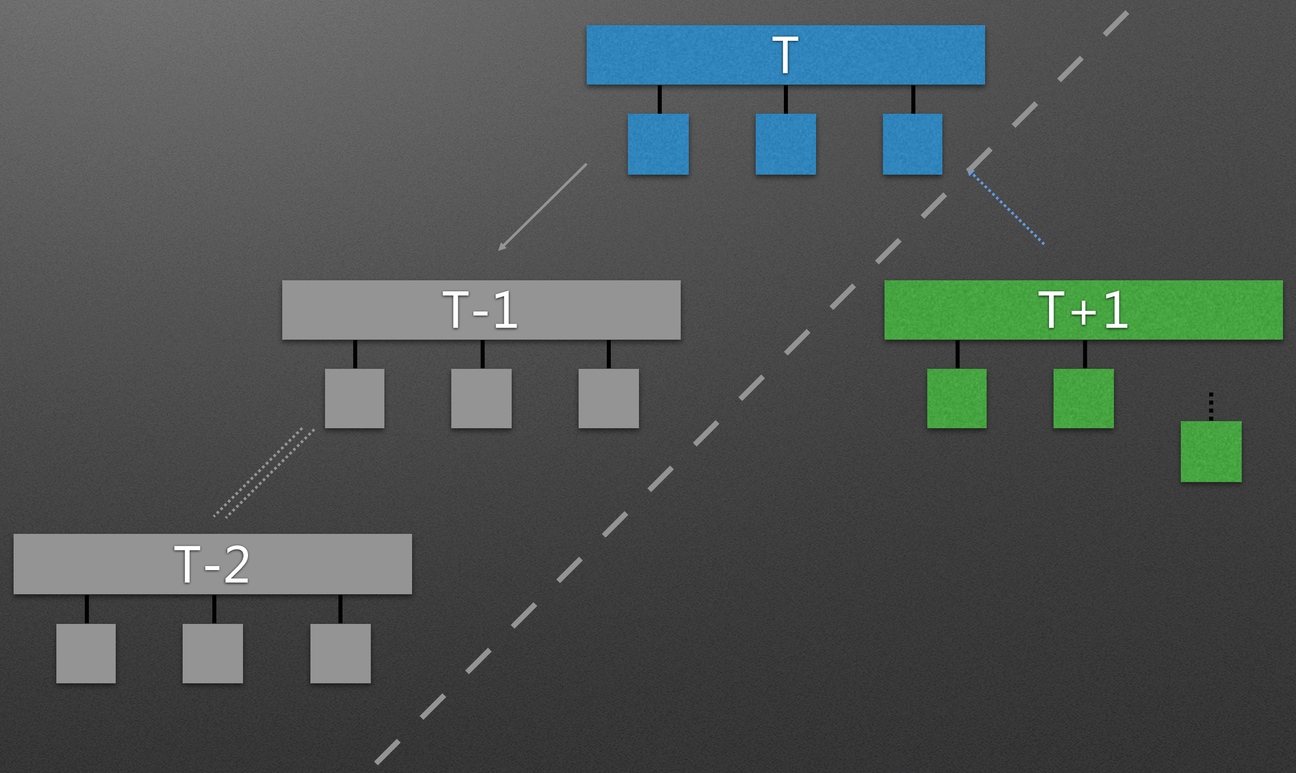
副作用

- 额外CPU开销
- 高Eden的填充速度



定时置换

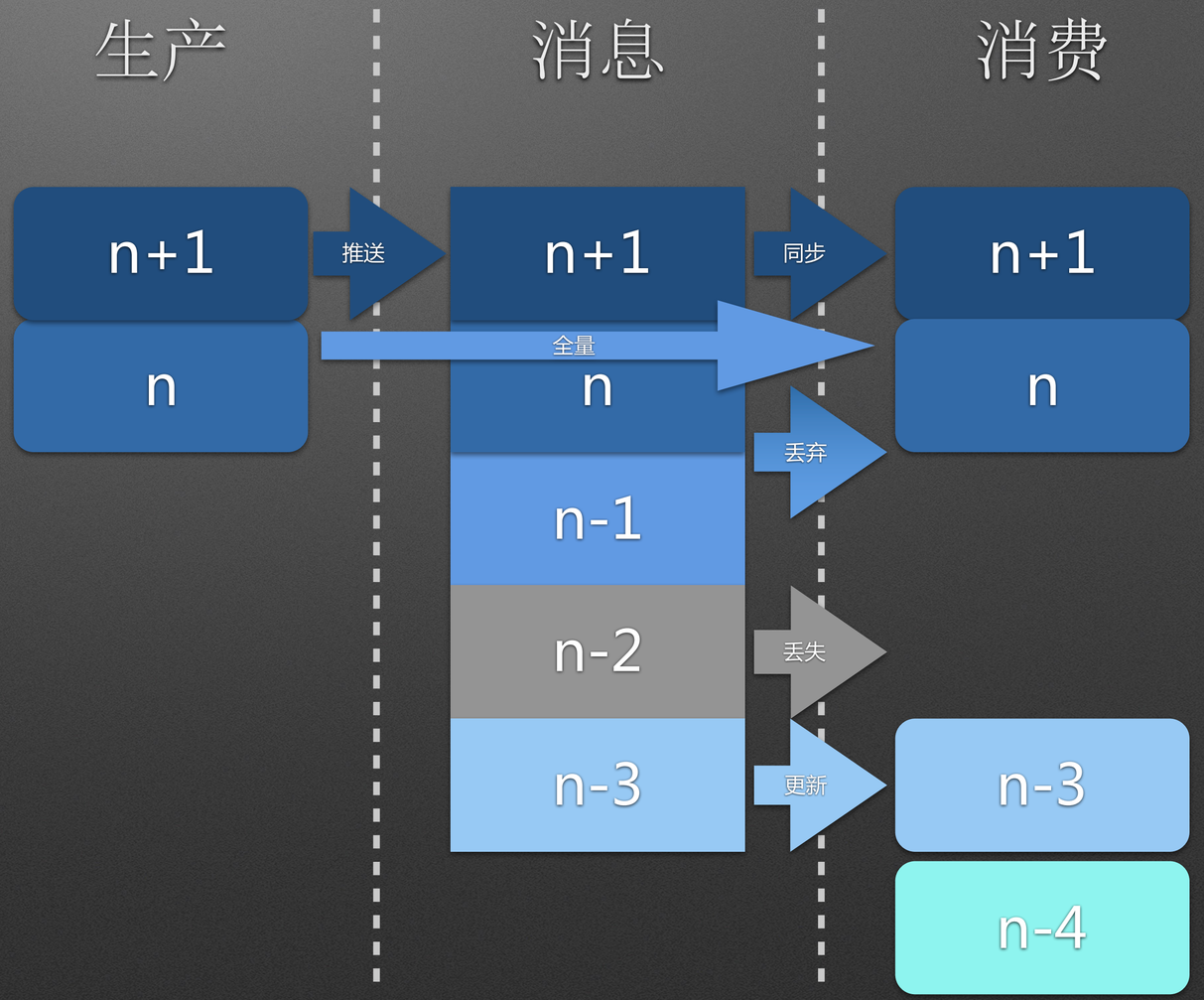
- 无锁读取
- 全量置换
  - 短时T_{copy}高峰
  - 伴随Old GC
- 同步延迟



增量更新

可靠性保证

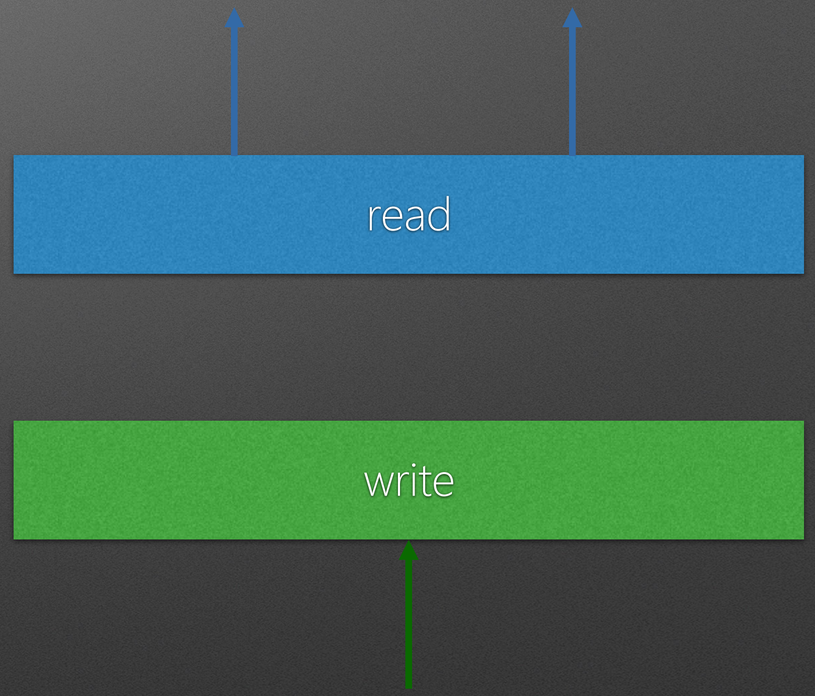
- 全量快照同步
- 消息版本验证
- 差异对比服务
- 独立检验机制



对象重用

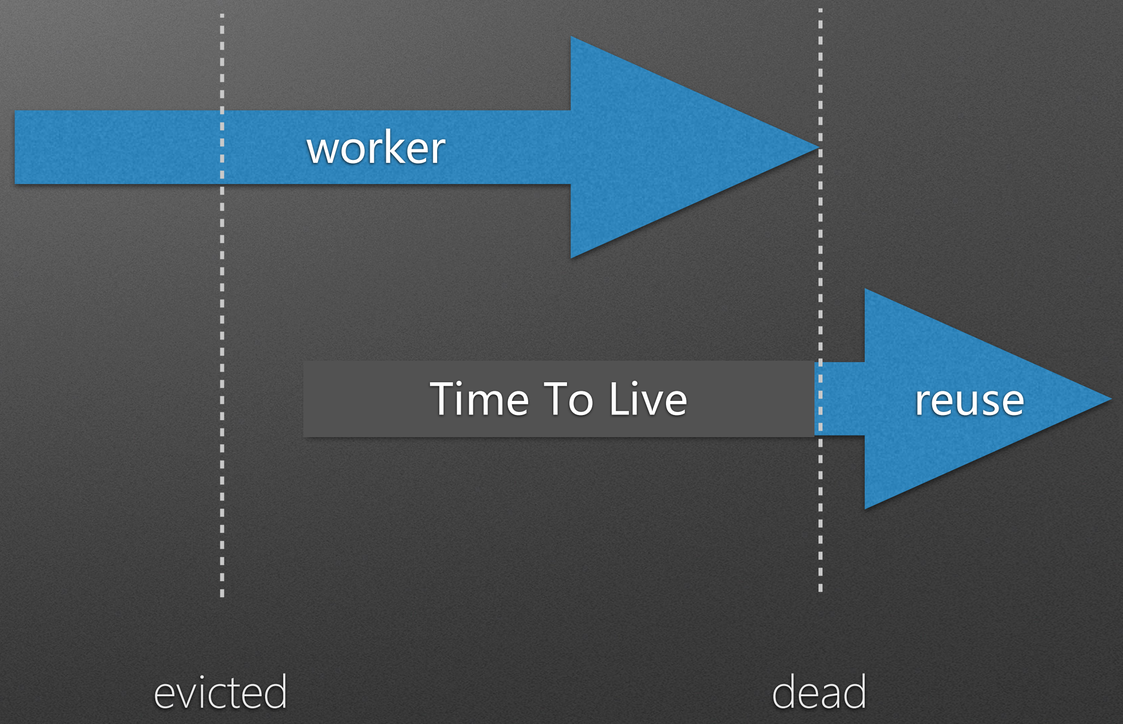
缓存对象重用

1. 缓存集合重用



缓存对象重用

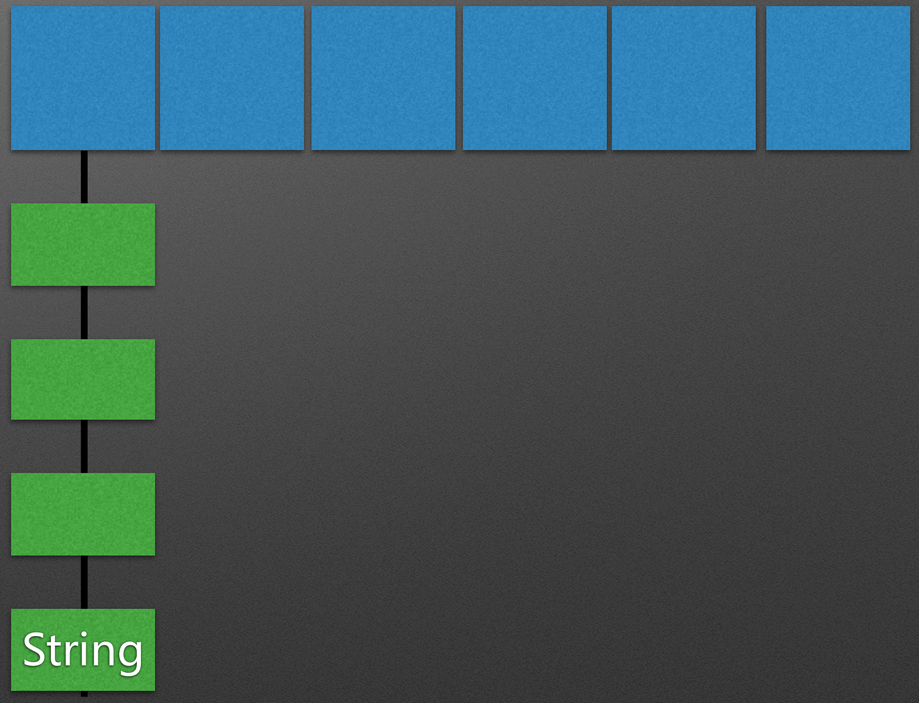
2. 缓存对象重用
   - 捕获淘汰对象
   - 使用前重置
   - 重置前的安全时间



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