讲师介绍

Hash　　QQ：805921455

从事Java软件研发十年。

前新浪支付核心成员、

咪咕视讯(中国移动)项目经理、

对分布式架构、高性能编程有深入的研究。

明天，你一定会感谢今天奋力拼搏的你

核心原理Memcached内存管理

分布式高并发—缓存技术

目录

课程安排

01

Memcached内存分配机制

Slab内存分配

02

memcached内存设计及管理

03

Memcached缓存策略 - LRU

分段LRU、LRU Crawler

04

总结

干货太多，不总结，我怕丢

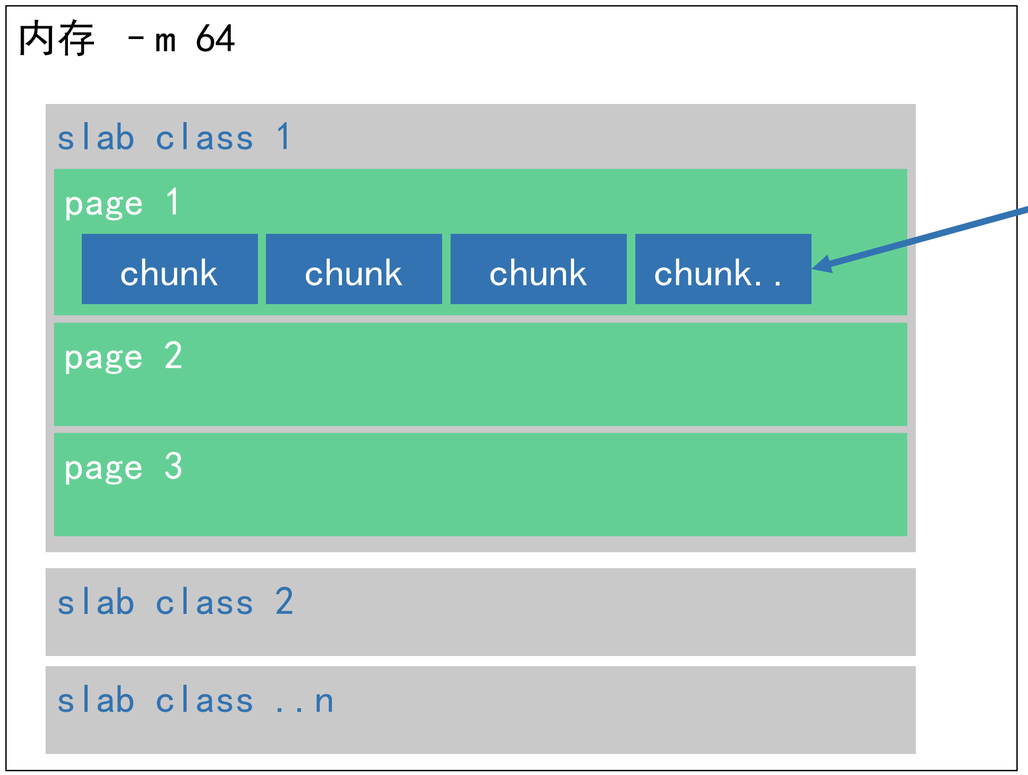
目录

Memcached内存分配机制

Memcached内存分配

启动 Memcached 时，-m指定内存大小，将信息保存到缓存中后才开始分配和保留物理内存。

通过 Slab allocation 机制对内存进行管理。（此处结合画图学习）



最大内存默认64，通过-m调整。

内存空间由slab classes构成，内存以slab page为单位去申请，分配到对应的slab class。

slab page ：最大1兆，由1个或多个chunk组成

chunk：实际存储数据的单元

目录

memcached内存设计及管理

内存设计及管理

Item

为键值数据的实际储存结构。item主要由公共属性、数据部分两个部分组成。

Chunk

由申请的连续内存块平均切分而成，用来存放Item数据，根据Item大小找到近似的Chunk。

Slab

管理特定大小的 chunk 的集合。Memcached每次默认分配的一个连续内存块为1M大小，它们被切分为不同大小的chunk。

Hash table

Memcached的哈希表采用链接法实现。hashtable被分成多个桶bucket，哈希冲突，通过h_next指针形成bucket下链接的单向链表。

LUR

Memcached中每个slab中都维护了一个LRU链表，来组织该slab中已经被分配的item块，用于记录“最近最少使用”的item信息。

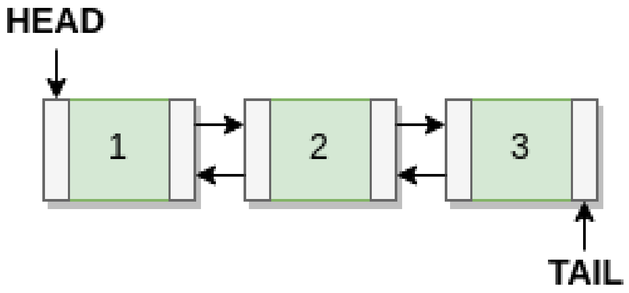
目录

Memcached缓存策略 - LRU

Memcached缓存策略 - LRU

在1.4.x及更早版本中，memcached中的LRU是标准的双向链表：有头部和尾部。将新物品插入头部，从尾部弹出驱逐物。

如果访问某个项目，则将其从其位置取消链接，然后重新链接到头部（此处称为“碰撞”），返回到LRU的顶部。

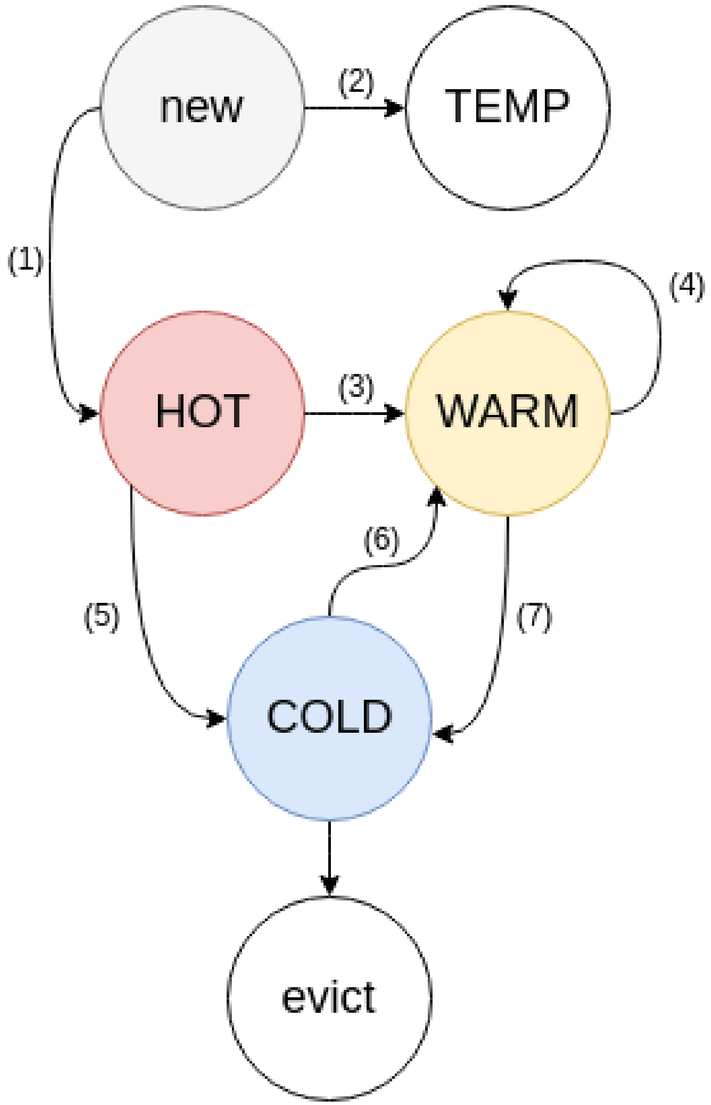


下面这些情况，带有超时时间的记录会被删除

1. 被人为删除
2. 被set覆盖
3. 被动删除：过期后，被get、add等命令访问
4. 主动清除：LRU机制

问题点：碰撞几率太高，对同一个链表的修改导致大量的互斥锁争抢，导致CPU使用率高或者响应变慢。

Memcached缓存策略 - 分段LRU



每个Slab-class安排一个LRU，每个LRU拆分为四个子LRU类型。

每个存储的数据都有两个标志位：FETCHED、ACTIVE

- FETCHED：该数据曾经被请求过
- ACTIVE： 该数据有两次或以上被请求，当数据被移动时移除。

TEMP： 该队列中的 item TTL 通常只有几秒，不会被挪动。

具体时间可配置 stats settings temporary_ttl选项

HOT：试用队列，数据不会长久存在该链表，一旦数据到达队列的尾部，则开始移动。

如果物品处于活动状态，它将被移动到WARM，非活动状态，它将被移动到COLD。

WARM： 访问量不大的数据

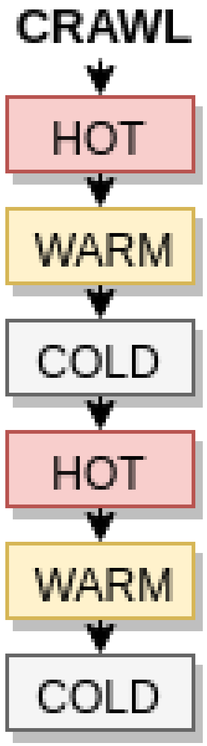
如果物品处于活动状态，它将被移动到warm头部，非活动状态，它将被移动到COLD。

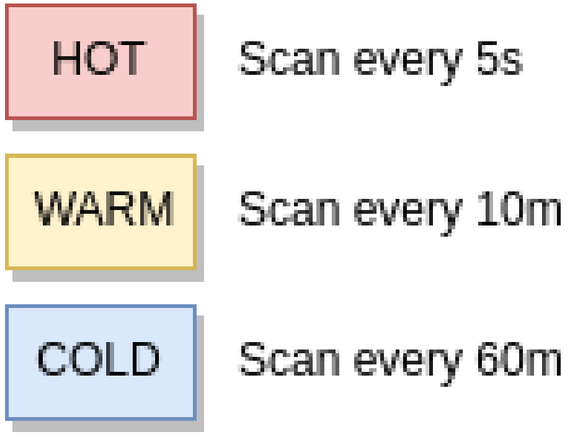
COLD：最不活跃的数据

回收时如果处于active状态，则移动到warm，否则删除。

总结：碰撞率变小了，提高了性能。

Memcached缓存策略 - LRU Crawler





LRU爬虫是一个单独的后台线程

专门用来处理失效的数据

检查每个slab class中每个子LRU链表

目录

总结

总结

Slab allocation内存分配机制，避免内存碎片，但会有内存浪费

Item、Chunk、Slab、Hashtable、LRU对应的结构

LRU碰撞

LRU分段

LRU爬虫

谢谢观看