06.29. 悲观锁和乐观锁

乐观锁和悲观锁，是为了解决并发场景下的数据竞争问题。

乐观锁：认为在操作数据时，不会由别的线程操作数据。不会上锁。在更新的时候判断在此期间是否被修改。如果没被修改，则执行操作。否则不执行（会自旋，不停重试）

悲观锁：认为会由别的线程操作数据。直接上锁。

实现：乐观锁：① CAS ② 版本号　悲观锁：加锁

① CAS (compare and swap)

3个操作数：读写的内存位置（V），进行比较的预期值（A）和写入的新值（B）

如果V中的值等于A，那么写入B。否则不停重试。

CAS是靠CPU支持的原子操作，硬件层面保证

② 版本号机制。数据中增加一个字段 version. 每修改一次，版本号 +1. 访问时同时读取版本号。更新数据时，如果版

若没变则更新。

CAS只能保证单变量操作原子性

并发少，读多：乐观锁

并发多，写多：悲观锁

CAS：ABA问题 A->B->A

高并发下不停重试开销大

功能受限

06.28 go中HashMap的实现

go中声明并初始化HashMap：

map := make(map[string]string)

key类型 value类型

key的种类可以是：布尔、数字、string，指针、chan、interface、struct，只包含上述类型的数组。

不能是：slice，map，function.

只要能支持==和!=操作 就可以做为key.

HashMap基本原理：①hash函数 ②如何解决冲突.

go的实现：① 在runtime/alg.go中定义3各种基础类的hash func equal func. 得到hash值后用与运算得到桶的编号

② 一般有开放地址法和拉链法。go中用的是拉链法。

桶的结构 bmap 每个桶可以存8个 k-v对.

h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	tophash, 存hash的前八位
K_1	K_2	K_3	K_4					
K_5	K_6	K_7	K_8					
V_1	V_2	V_3	V_4					
V_5	V_6	V_7	V_8					

overflow *bmap 溢出桶 (8个装不下可以再链一个桶)

常规桶的个数为 2^B, 一定是2的幂次。hash值与 (2^B-1) 与运算后,定位桶编号。与运算与取模效果一致,但更快,需要与2的幂次才可以。

如果B>4, 溢出桶的个数为 2^{B-4}, 否则无溢出桶

预分配

hmap{

count int

flags uint8

B uint8

noverflow uint8

hash0 uint32

buckets unsafe.Pointer

oldbuckets unsafe.Pointer

nevacuate uintptr (指下未迁移的桶)

*mapextra

extra

}

*mapextra {
overflow *[]*bmap
oldoverflow *[]*bmap
nextOverflow *bmap

扩容规则

$\frac{count}{(2^B)} > 6.5$ → 考虑扩容

Load factor

Load factor ≤ 6.5 → 等量扩容

noverflow 较多

$B \leq 15$, noverflow $\geq 2^B$

$B > 15$, noverflow $\geq 2^{15}$

扩容不是一步到位，是一个扩容，迁移分摊到多次的等表操作中：渐进式扩容，可以避免一次性扩容带来的性能抖动。

06.27 红黑树，B树和B+树.

红黑树是二叉树，大规模数据存储时红黑树很深，所需IO很多，效率低下，对存储设备寿命的影响也很大。因此，我们需要使用多叉的搜索树，来降低树的高度，提高效率。

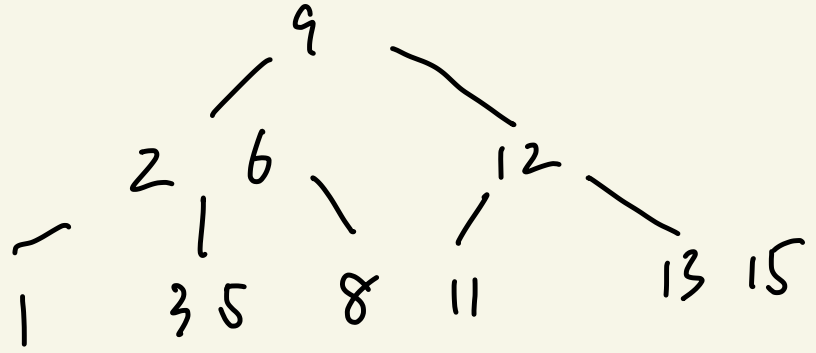
B树：m叉的搜索树，m称为B树的阶，m取决于磁盘页的大小。$m>2$.

~~根结点的孩子树：$[2,m]$~~

~~非根非叶结点的孩子数：$[m/2,m]$~~

非叶结点的关键字个数(k-v对)：孩子数-1.

k个关键字个数把节点拆成k+1段，指向k+1个孩子，



所有叶子节点位于同一层。

与B+树不同的点：关键字分布在整棵树

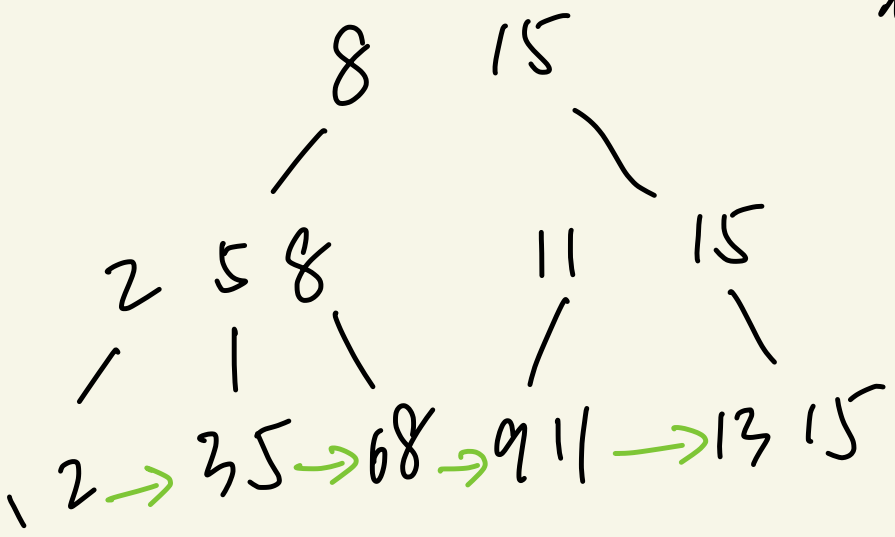
任何关键字只出现一次

搜索可能停止在非叶节点

B+树：一个节点K个关键字，拆成K段，指向K个孩子

（B树会差1）。 根结点至少

关键字是B树的最大元素，一些关键字是出现多次的。



非叶节点不存储V，只作索引

叶节点才存K-V

所有叶节点用链表相连，range-query方便

子树个数，关键字个数：

B：每个节点最多有m个子树；根节点最少2个子树，其他非叶节点最少ceil(m/2)个子树。非叶节点：关键字个数=子树个数-1.

B+：子树个数的情况和B树一样，关键字数=子树个数

B+树优点：非叶节点不存储V的信息，进一步压缩空间，m可以进一步变大。range-query方便（B树找数也中序便利）

B树：可能可以不到叶节点就找到，更快。（B+树：只能到叶节点找到V，但是IO性能更稳定）

06.26 Python内存管理特点

1. 引用计数

Python内部记录3对象有多少个引用。对象创建时会创建一个引用计数，当引用计数为0，会被GC.

① 分配新名称/放入容器，引用计数增加

② del/引用超出作用域/重新赋值 引用计数减少

③ sys.getrefcount() 获取引用计数

2. GC：检查引用计数为0的对象，清除内存空间

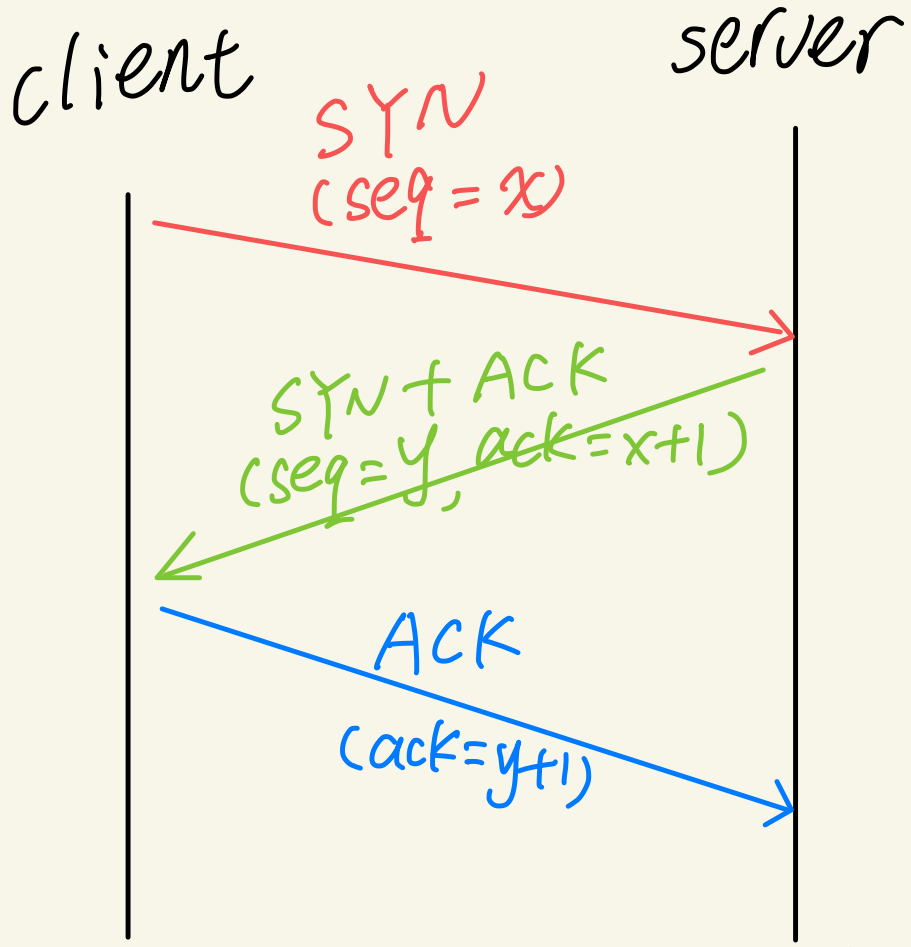
3. 内存池：很多小内存申请释放频繁，会影响效率。引入内存池机制，用于管理小块内存。能够减少内存碎片。

① GC将内存返还给内存池而非操作系统

② 小于256B的对象使用Pymalloc分配器，大的使用系统malloc

③ Int, Float, List等之间不共享内存池

06.23 TCP 3-way handshake process



TCP usea the 3-way handshake process to establish a secure and reliable communication between the client and server.

① client sends the SYN to the server (synchronization) with SYN Flag as 1, means client wants to establish a secure connection with the server. seq = x, which is a random number, assigned to the first bit of the data.

② Server receives the SYN from client, sends SYN + Ack to the client, with SYN and Ack Flag as 1. ack number is set as x+1, tell client that the server has successfully receives the previous SYN from the client. Anothe seq is set as y, which is a random number, assigned to the first bit of the data.

③ Client receives the SYN+ACK from the server, send an ACK to the server, with ack number set as y+1.

After the 3-way handshake, the TCP connection is established.

06.22

oop features,

06.21 Describe what happens when you click on a URL in a browser

① You type the URL, for example, www.leetcode.com into the browser.

② The browser will contact the DNS (domain name system), to convert the human-readable URL into the numerical IP address.

③ With the IP address, the browser can send an HTTP request to the server.

④ If the server receives the HTTP request from the browser, it will prase

the request, and send an HTTP response to the browser.

⑤ After receiving the response, the browser will render the HTML encoded in the response. If there are additional rescourses embedden in the HTML, steps 3~5 will be repeated.

4 layer cache:

browser, os, router, ISP

recursively request the IP address

Root DNS

Top-level .com

Second-level microsoft.com

Third-level download.microsoft.com

TCP connection, 3-way handshake