Cuckoo 性能测试

简介

哈希表是一种可以快速寻找目标键值对的方式，但是在多个键对应的哈希值相同时就会发生“冲突”。为了应对这种冲突，不同的哈希表会有不同的冲突应对策略——线性哈希在对应哈希值的基础上继续向下查找空位，并将其放在第一个空位上；链式哈希表则会将对应哈希值的所有键值对串成链表，查找时遍历链表。

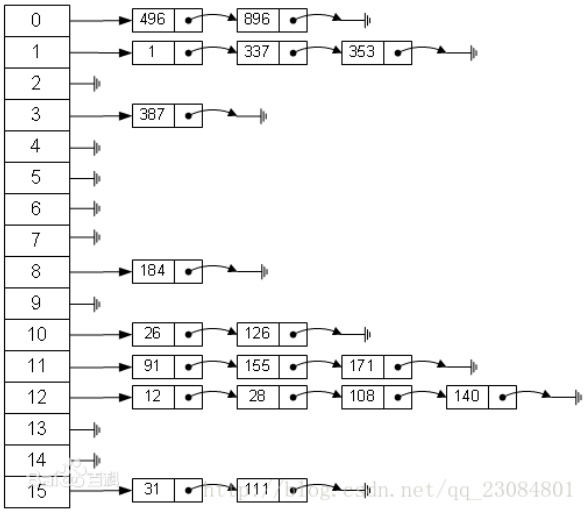

图 1 链式哈希表

Cuckoo 采用了不同的设计来减少冲突情况下查找的代价。即每个键有且只有两个不同的哈希值，如果两个哈希值对应的位置存在空位就存放在其中。如果两个哈希值对应的位置不存在空位，那就需要转移对应位置中已有的键值对，直到所有键值对都落在它们两个哈希值之一对应的指定位置处。

当然这种方法在键值对转移过程中可能会出现死循环，解决方法是路径检测加上重新哈希。检测循环路径的方法也比较简单，可以预先设定一个阈值，当循环次数或者递归调用次数超过设定阈值时，就可以认为产生了循环路径。

具体过程见课件。

实验内容

本实验需要测试 Cuckoo 在不同负载因子（哈希表的负载因子是指的键的总数与哈希表容量的比值）下查找的性能。测试的键值对可以都使用键来模拟，即键与值都是相同的数字。

你需要做的：

1. 实现 Cuckoo 哈希算法，并且通过设定较大阈值的方式避免死循环。
2. 以哈希表容量为 100000 为例，在负载因子为 0.1, 0.5, 0.75, 1.0 的情况下，插入数据

并测试 Cuckoo 哈希的查找时间（测试查找成功的情况即可）。

3. 可选的，可以尝试测试串行 put 多 get 的优化。在购物网站的场景下 put 操作较少而 get 操作较多，这样使用串行 put 操作可以避免并行 put 操作带来的同步开销。该部分需要设计一套 benchmark，记录一系列的 put 和 get 操作及顺序（put 要远少于 get 操作，至少数量级的差距）。然后分别在并行 get 下测试并行 put 和串行 put 的效率。