**为啥 redis 使用跳表(skiplist)而不是使用 red-black？**

当数据元素个数少于某值，redis会使用l跳表来存储元素，而不使用hash。  
如果把跳表换成红黑树，redis的性能会有多大的变化？

[Hak Ego](https://www.zhihu.com/people/hak-ego)

题主的问题没有说清楚，有些概念混淆。  
1. “当数据元素个数少于某值，redis会使用l跳表来存储元素，而不使用hash。” 这里应该是ziplist而不是跳表。ziplist的使用属于redis具体实现上，针对较少元素时用时间换空间的一种优化方法。  
2. redis使用跳表的地方是在ZSET(sorted list)里，zset是redis支持的一类数据结构，属于redis对外接口的一部分。请查阅Z\*系列的命令。  
如果题主是想问跳表和红黑树的话，就不要提节省内存的ziplist。这两个是不同level上的概念。

[**于康**](https://www.zhihu.com/people/cyukang)，买飞机找我~~

[amazingjxq](https://www.zhihu.com/people/amazingjxq)、[十三](https://www.zhihu.com/people/shi-san-97" \o "十三)、[陈乐群](https://www.zhihu.com/people/abcdabcd987) [等人赞同](javascript:;)

1 skiplist的复杂度和红黑树一样，而且实现起来更简单。  
2 在并发环境下skiplist有另外一个优势，红黑树在插入和删除的时候可能需要做一些rebalance的操作，这样的操作可能会涉及到整个树的其他部分，而skiplist的操作显然更加局部性一些，锁需要盯住的节点更少，因此在这样的情况下性能好一些。  
  
具体可以参考Herb Sutter写的[Choose Concurrency-Friendly Data Structures](https://link.zhihu.com/?target=http%3A//www.drdobbs.com/parallel/choose-concurrency-friendly-data-structu/208801371" \t "_blank).  
  
另外这篇论文里有更详细的说明和对比，page50~53：  
[http://www.cl.cam.ac.uk/research/srg/netos/papers/2007-cpwl.pdf](https://link.zhihu.com/?target=http%3A//www.cl.cam.ac.uk/research/srg/netos/papers/2007-cpwl.pdf)  
  
囧 原来和一楼答案差不多。

[**dccmx**](https://www.zhihu.com/people/dccmx)，搞技术的

[空成](https://www.zhihu.com/people/kong-cheng-09-03)、[freeNik](https://www.zhihu.com/people/freenik)、[Witkey Fan](https://www.zhihu.com/people/witkey.fan) [等人赞同](javascript:;)

redis使用跳表(ziplist)?  
首先，跳表是skiplist？不是ziplist。ziplist在redis中是一个非常省内存的**链表**（代价是性能略低），所以在hash元素的个数很少（比如只有几十个），那么用这个结构来存储则可以在性能损失很小的情况下节约很多内存（redis是内存数据库啊，能省还是要省的）。好这个问题清楚了。  
  
至于这个问题：**请问，有人用C实现过红黑树么？**实在看不懂。  
  
========无敌的分隔线=======  
  
不过提问者所引出的另一个问题倒是值得讨论一下，就是在server端，对并发和性能有要求的情况下，如何选择合适的数据结构（这里是跳跃表和红黑树）。  
如果单纯比较性能，跳跃表和红黑树可以说相差不大，但是加上并发的环境就不一样了，如果要更新数据，跳跃表需要更新的部分就比较少，锁的东西也就比较少，所以不同线程争锁的代价就相对少了，而红黑树有个平衡的过程，牵涉到大量的节点，争锁的代价也就相对较高了。性能也就不如前者了。  
  
不过这些对redis这个单进程单线程server来说都是浮云。

之前所有的答案都不太靠谱（完全扯淡）  
  
请看开发者说的，他为什么选用skiplist [The Skip list](//link.zhihu.com/?target=https%3A//news.ycombinator.com/item%3Fid%3D1171423)

There are a few reasons:

1) They are not very memory intensive. It's up to you basically. Changing parameters about the probability of a node to have a given number of levels will make then *less* memory intensive than btrees.

2) A sorted set is often target of many ZRANGE or ZREVRANGE operations, that is, traversing the skip list as a linked list. With this operation the cache locality of skip lists is at least as good as with other kind of balanced trees.

3) They are simpler to implement, debug, and so forth. For instance thanks to the skip list simplicity I received a patch (already in Redis master) with augmented skip lists implementing ZRANK in O(log(N)). It required little changes to the code.

About the Append Only durability & speed, I don't think it is a good idea to optimize Redis at cost of more code and more complexity for a use case that IMHO should be rare for the Redis target (fsync() at every command). Almost no one is using this feature even with ACID SQL databases, as the performance hint is big anyway.

About threads: our experience shows that Redis is mostly I/O bound. I'm using threads to serve things from Virtual Memory. The long term solution to exploit all the cores, assuming your link is so fast that you can saturate a single core, is running multiple instances of Redis (**no locks, almost fully scalable linearly with number of cores)**, and using the "Redis Cluster" solution that I plan to develop in the future.

红黑树在并发情况下的平衡需要锁第一个是比较复杂，调整涉及到多个节点的并发修改  
第二个是对于越接近根节点的地方越容易产生竞争，即使是不同叶子节点的操作由于平衡操作也可能逐级向上涉及到接近根的节点  
而对于skiplist类的链表可以用cas来并发操作节点，比较容易实现，且更加局部化。

全世界都知道  
[FAQ – Redis](https://link.zhihu.com/?target=http%3A//www.redis.io/topics/faq)  
Redis is single threaded, how can I exploit multiple CPU / cores?

It's very unlikely that CPU becomes your bottleneck with Redis, as usually Redis is either memory or network bound. For instance using pipelining Redis running on an average Linux system can deliver even 500k requests per second, so if your application mainly uses O(N) orO(log(N)) commands it is hardly going to use too much CPU.

However to maximize CPU usage you can start multiple instances of Redis in the same box and treat them as different servers. At some point a single box may not be enough anyway, so if you want to use multiple CPUs you can start thinking at some way to shard earlier.

You can find more information about using multiple Redis instances in the [Partitioning page](https://link.zhihu.com/?target=http%3A//www.redis.io/topics/partitioning)

faq写了不共享数据么

using multiple Redis instances的话，每个redis instance是单独的进程，怎么会共享数据呢，如果有数据共享，请说明理由哈

两个都保证数据按照一定顺序存放了，跳表的优势就是能快速拿到子序列（两次定位2\*logN), 而红黑树达不到这种性能(其实是不擅长)

[**为什么工程中都用红黑树，而不是其他平衡二叉树？**](http://www.zhihu.com/question/27542473)

[**我的上铺叫路遥**](http://www.zhihu.com/people/begeekmyfriend)**，Be geek my friend!**

《算法导论》作者之一Thomas Cormen在Quora上征询第四版内容意见（当然目前仅仅是征询意见），RB-tree赫然在Remove清单上：[http://www.quora.com/As-we-start-planning-the-next-edition-of-Introduction-to-Algorithms-CLRS-what-should-we-add-and-what-should-we-remove](http://link.zhihu.com/?target=http%3A//www.quora.com/As-we-start-planning-the-next-edition-of-Introduction-to-Algorithms-CLRS-what-should-we-add-and-what-should-we-remove)  
  
取而代之的是谁呢？**skiplist当仁不让。**  
  
首先Mark Allan Weiss的《数据结构与算法分析C++描述》（我个人非常推荐的算法书）“确定性跳跃表”\*(determistic skiplist)一节说“性能似乎比RB-tree要强”，还有省内存，具体见书。  
  
其次**skiplist结构在有两个RB-tree无法实现的杀手锏：**其一是可以像数组水平顺序遍历key，这也是B+tree对B-tree的优点；其二对key排名(ranking)，见redis实现。  
  
还有代码，我个人在github上的参考redis的实现，我代码简洁，但多用了一个指针（双链表），写了以后简直不想写RB-tree了，尤其是删除（你看过STL源码吧）：[leo-ma/skiplist · GitHub](http://link.zhihu.com/?target=https%3A//github.com/leo-ma/skiplist)  
  
同样作为内存字典结构，skiplist有什么应用限制吗？微博上曾有人指出“**不是任何场景都能接受 skiplist 的随机性**”。