牛客大学高薪加成系列课

快速列表

牛客大学

- 专业求职辅导 -

1. 快速列表（quicklist）是Redis 3.2新引入的数据结构，该结构是链表和压缩列表的结合；
2. 快速列表是采用双向链表将若干压缩列表连接到一起而组成的数据结构，即它是一个双向链表，链表中的每个节点是一个压缩列表，这种设计能够在时间效率和空间效率之间实现较好的折中。

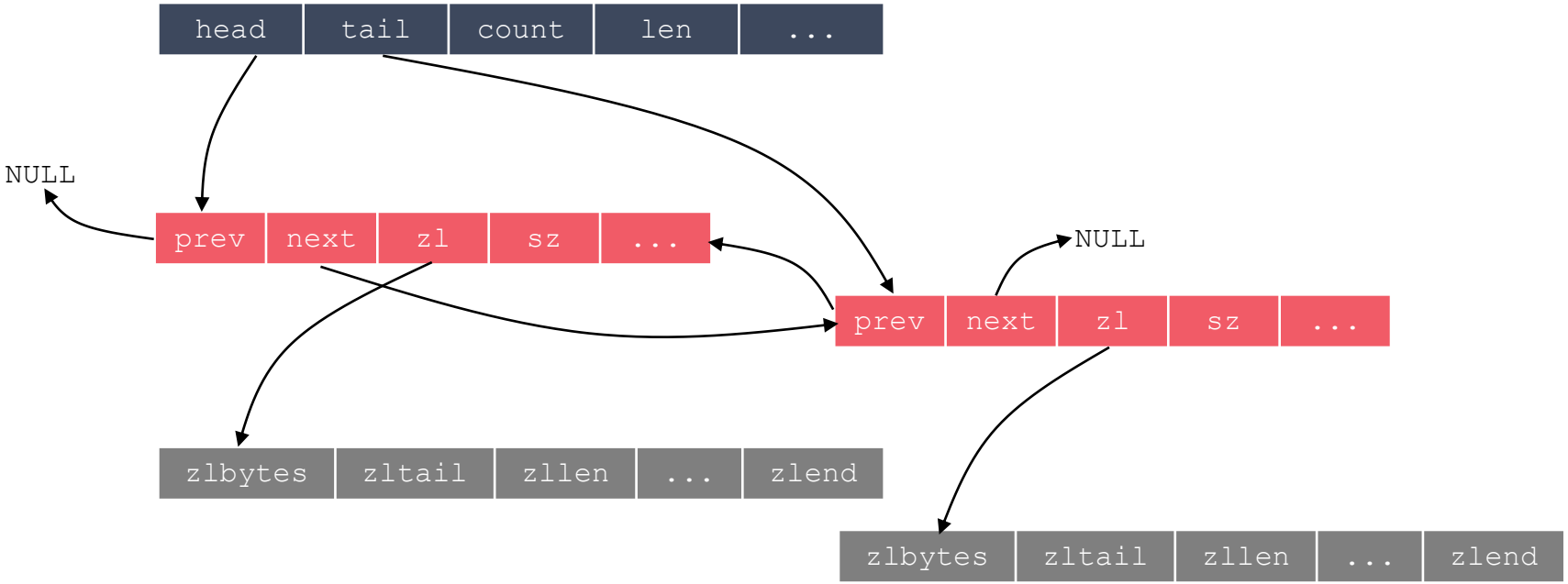
在3.2之前，列表类型是采用压缩列表及双向链表实现的；
从3.2开始，列表类型采用快速列表作为底层的唯一实现。

02 / 快速列表的实现

quicklist.h: quicklist, quicklistNode

```
typedef struct quicklist {
    // 头节点
    quicklistNode *head;
    // 尾节点
    quicklistNode *tail;
    // 压缩列表的元素总数
    unsigned long count;
    // 快速列表的节点个数
    unsigned long len;
    // 压缩列表的最大填充量
    int fill:QL_FILL_BITS;
    // 不参与压缩的节点个数
    unsigned int compress:QL_COMP_BITS;
    // 书签数量
    unsigned int bookmark_count:QL_BM_BITS;
    // 书签数组
    quicklistBookmark bookmarks[];
} quicklist;
```

```
typedef struct quicklistNode {
    // 前一个节点
    struct quicklistNode *prev;
    // 后一个节点
    struct quicklistNode *next;
    // ziplist
    unsigned char *zl;
    // ziplist的字节数量
    unsigned int sz;
    // ziplist的元素个数
    unsigned int count:16;
    // 编码方式(RAW==1 or LZF==2)
    unsigned int encoding:2;
    // 容器类型(NONE==1 or ZIPLIST==2)
    unsigned int container:2;
    // 该节点是否被压缩过
    unsigned int recompress:1;
    // 用于测试期间的验证
    unsigned int attempted_compress : 1;
    // 预留字段
    unsigned int extra : 10;
} quicklistNode;
```


head
tail
count
len
...
NULL
prev
next
zl
sz
...
NULL
prev
next
zl
sz
...
zlbytes
zltail
zllen
...
zlend
zlbytes
zltail
zllen
...
zlend

1. 为了进一步降低ziplist占用的内存空间，Redis允许采用LZF算法，对ziplist进行压缩；
2. LZF算法的基本思想是，数据与前面重复的，记录重复位置以及长度，否则直接记录原始数据；
3. 压缩后的数据分为多个片段，每个片段包括解释字段和数据字段两部分，数据字段可能不存在。

解释字段	数据字段
000LLLLL	1. 长度由解释字段的后5位决定； 2. 直接读取后续的数据字段内容，长度为所有L组成的字面值加1； 3. 例：00000001，代表数据字段长度为2；
LLLooooo	1. 没有数据字段，数据内容与前面内容重复，重复长度小于8； 2. 长度是所有L组成的字面值加2，偏移量是所有o组成的组面值加1； 3. 例：00100000 00000100，代表前面5字节处内容重复，重复3字节；
lllooooo LLLLLLLL oooooooo	1. 没有数据字段，数据内容与前面内容重复； 2. 长度是所有L组成的字面值加9，偏移量是所有o组成的字面值加1； 3. 例：11100000 00000010 00010000，代表与前面17字节处内容重复，重复11字节；

牛客大学
-专业求职辅导-

THANKS

关注【牛客大学】公众号
回复“牛客大学”获取更多求职资料