图文 18 基于冷热数据分离方案优化后的LRU链表，是如何解决之前的问题的?

详情 评论

基于冷热数据分离方案优化后的LRU链表，是如何解决之前的问题的?

如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑
如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

具体加群方式，请参见目录菜单下的文档：**《MySQL专栏付费用户如何加群》**（购买后可见）

1、对于预读以及全表扫描加载进来的一大堆缓存页

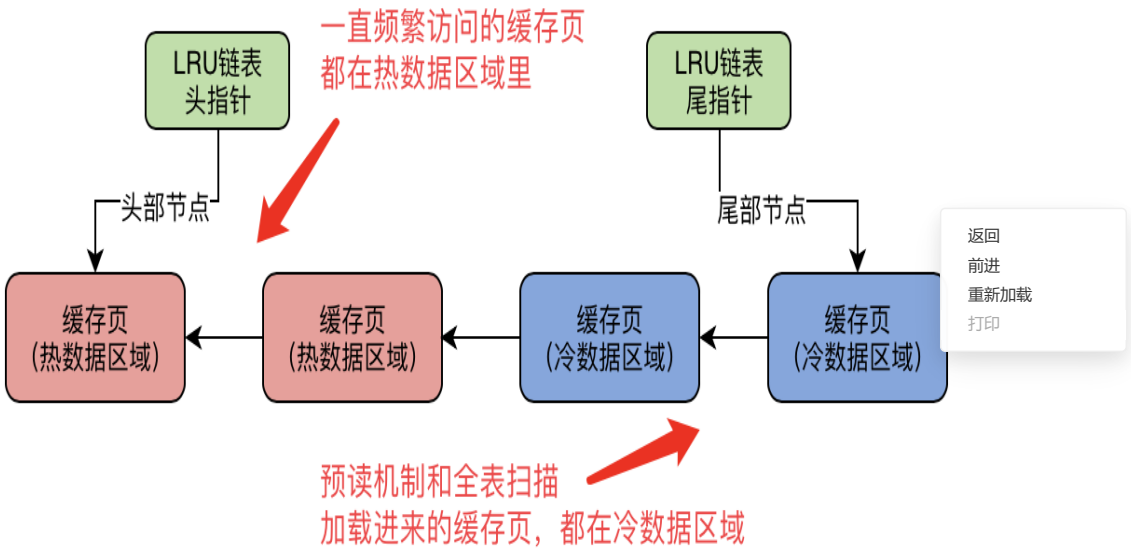
现在我们已经看完了LRU链表的冷热数据分离的方案，那么我们接着看这个冷热数据分离之后的LRU链表，他是如何解决之前遇到的一大堆问题的?

首先我们思考一下，在这样的一个LRU链表方案下，预读机制以及全表扫描加载进来的一大堆缓存页，他们会放在哪里?

明显是放在LRU链表的冷数据区域的前面啊!

假设这个时候热数据区域已经有很多被频繁访问的缓存页了，你会发现热数据区域还是存放被频繁访问的缓存页的，只要热数据区域有缓存页被访问，他还是会被移动到热数据区域的链表头部去。

所以此时你看下图，你会发现，预读机制和全表扫描加载进来的一大堆缓存页，此时都在冷数据区域里，跟热数据区域里的频繁访问的缓存页，是没关系的!



2、预读机制和全表扫描加载进来的缓存页，能进热数据区域吗?

接着我们看第二个问题，预读机制和全表扫描机制加载进来的缓存页，什么时候能进热数据区域呢?

如果你仅仅是一个全表扫描的查询，此时你肯定是在1s内就把一大堆缓存页加载进来，然后就访问了这些缓存页一下，通常这些操作1s内就结束了。

所以基于目前的一个机制，可以确定的是，这种情况下，那些缓存页是不会从冷数据区域转移到热数据区域的!

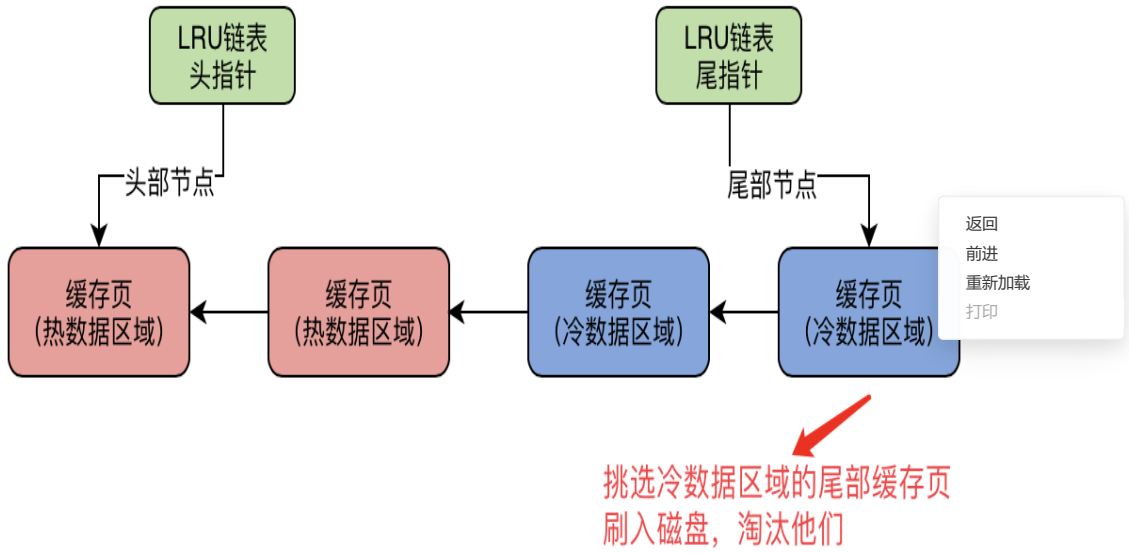
除非你在冷数据区域里的缓存页，在1s之后还被人访问了，那么此时他们就会判定为未来可能会被频繁访问的缓存页，然后移动到热数据区域的链表头部去!

3、如果此时缓存页不够了，需要淘汰一些缓存，会怎么样?

接着我们看，假设此时缓存页不够了，需要淘汰一些缓存页，此时会怎么做?

那就很简单了，直接就是可以找到LRU链表中的冷数据区域的尾部的缓存页，他们肯定是之前被加载进来的，而且加载进来1s过后都没人访问过，说明这个缓存页压根儿就没人愿意去访问他!他就是冷数据!

所以此时就直接淘汰冷数据区域的尾部的缓存页，刷入磁盘，就可以了，我们看下图。



4、之前的一大堆问题解决了吗?

在这样的一套缓存页分冷热数据的加载方案，以及冷数据转化为热数据的时间限制方案，还有就是淘汰缓存页的时候优先淘汰冷数据区域的方案，基于这套方案，大家会发现，之前发现的问题，完美的被解决了。

因为那种预读机制以及全表扫描机制加载进来的数据页，大部分都会在1s之内访问一下，之后可能就再也不访问了，所以这种缓存页基本上都会留在冷数据区域里。然后频繁访问的缓存页还是会留在热数据区域里。

当你要淘汰缓存的时候，优先就是会选择冷数据区域的尾部的缓存页，这就是非常合理的了！他不会让刚加载进来的缓存页占据LRU链表的头部，频繁访问的缓存页在LRU链表的尾部，淘汰的时候淘汰尾部的频繁访问的缓存页了！

问题完美的被解决了。

这就是LRU链表冷热数据分离的一套机制。

5、总结

通过这几篇文章的学习，我们已经彻底搞定了LRU链表的设计机制，刚加载数据的缓存页都是放冷数据区域的头部的，1s过后被访问了才会放热数据区域的头部，热数据区域的缓存页被访问了，就会自动放到头部去。

这样的话，实际上冷数据区域放的都是加载进来的缓存页，最多在1s内被访问过，之后就再也没访问过的冷数据缓存页！