文章编号:1000-5862(2013)05-0500-05

一种带红黑树的哈希表在物流信息系统中的应用研究

滕少华,林国华
(广东工业大学计算机学院,广东 广州 510006)

摘要:针对在节假期间物流业务量急剧增长及大量用户在线查询订单信息而导致物流信息系统响应速度慢的问题,提出了带红黑树的哈希表,有效地提高了订单查询的速度,并将链表和红黑树进行比较,实验结果表明:带红黑树的哈希表在查找时间上有明显优势.

关键词:物流信息系统;哈希表;红黑树;查询效率
中图分类号:TP 31　　**文献标志码**:A

0 引言

随着经济及信息化技术的迅猛发展,物流快递行业也迅速发展起来. 然而一旦到了春节、五一、十一等节假日期间,快递公司的业务量急剧增长,尤其是近年来,网购业务的急速增长进一步加剧了这种现象[1],据统计,中国快递的日业务量高达 1 000 万件[2],这使得在此期间大量的用户并发性地查询订单信息,导致物流信息系统运行缓慢,响应时间加长. 针对物流信息系统,文献[3]提出使用链表来存储订单信息,链表在查找效率上不高;文献[4]使用哈希表进行快速定位,减少了响应时间. 哈希表在查询方面有独特的优势,然而在碰撞严重的情况下,哈希查找速度会受到影响,特别平时和节假日的业务量差别较大,使得采用哈希表时,存储空间的大小难以取舍. 当存储空间过大,会造成浪费;当存储空间过小,节假日的哈希碰撞会比较严重. 为此本文设计了一个带红黑树的哈希表 RBTHT(Red & Black Tree Hash Table),并应用到物流运输系统中,经仿真实验表明本文的方法能够获得较快的运行效果.

1 物流运输业务流程分析及其数据结构设计

物流运输系统,为人们带来了诸多方便,客户只需要在物流系统中输入快递单号,就可以查询到当前订单的物流状态. 然而节假日期间,由于运输业务量大,查询物流状态的用户也会急剧增长,导致物流运输系统响应速度较慢,严重时甚至出现死机的情况. 为此,本文深入地分析了物流运输的基本业务货物流程,提出了一个带红黑树的哈希表的方案.

1.1 物流运输业务流程

物流运输中订单查询贯穿整个物流过程,客户、物流公司工作人员的订单查询、车辆的调度及运输的跟踪都涉及到订单信息. 具体的物流运输业务流程如图 1 所示.

对于查询过程,主要是对订单的查询,包括订单的物流状态查询、到货签收查询、回单情况查询等[5]. 从图 1 可以看出,随着客户订单量的增大,对订单的查询操作也就越频繁,一方面来自客户的查询,客户为了查询订单的物流状态,频繁地查询订单;另一方面来自物流公司的工作人员,为了及时安排运货及计算订单产生的费用,都必须进行订单查询操作. 在正常情况下,这种查询操作对物流信息系统的影响并不大,然而到了节假日时,物流公司的业务量急剧增长,使得查询订单的操作过于频繁,导致物流信息系统变缓变慢.

哈希表在查询方面有独特的优势,在理想情况下,查询操作的时间复杂度可达到 $O(1)$,可以大大减少查询订单的时间,增大物流信息系统的负载量,为此本文采用哈希表来存储订单的信息.

收稿日期:2013-04-19
基金项目:教育部重点实验室基金(110411),广东省自然科学基金(10451009001004804)和广东省科技计划(2012B091000173)资助项目.
作者简介:滕少华(1962-),男,江西南昌人,教授,博士,主要从事协同计算,数据挖掘和网络安全方面的研究.

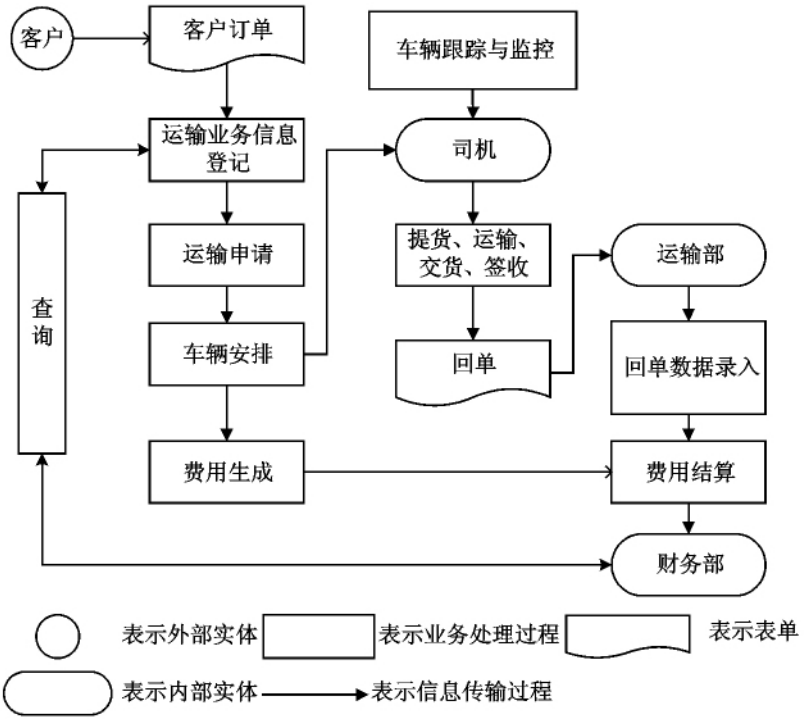

图 1　物流运输业务流程

1.2　数据结构设计

哈希表又称为散列表,其基本思想为:将数据集合中每条记录的关键字 *Key* 作为自变量,然后通过某个映射关系 Hash(*Key*)计算出相应的值,而这个值与一块连续存储空间的地址相对应(常常将这个值映射到数组的下标而不是真正存储地址),最后将这条记录存储到此单元.

用于存储订单集合的数据结构如下:

```
class rb_hashtable
{
private:
    hasher M_hash;              //用于保存哈希函数
    Vector_type M_buckets;      //桶子聚合体
    size_type M_num_elements;   //关键字的个数
};
```

其中哈希桶的聚合体 M_buckets 是一个动态数组,可以动态扩充自身空间.

然而对于大部分情况,哈希碰撞是难以避免的.为处理碰撞,在 JAVA 及 C++标准库[6]中,采用拉链法处理哈希表的冲突问题,该方法把所有碰撞的关键字(称为同义词)存储在一个线性链表中,这个链表称为同义词链表,并将这些链表的表头指针放在数组中.但该实现依然存在问题,如果冲突很大,链表将变得很长,由于对链表进行搜索是一种线性操作,此时消耗时间将线性增长,查找的时间复杂度也将线性增长;在最坏情况下,查找的时间复杂度变为 $O(n)$[7],效率也很慢.但根据文献[8]可知,红黑树为一种基本保持平衡的二叉排序树,在查找、插入、删除方面的最坏时间复杂度都为 $O(\log n)$,查找效率高,可以有效地解决哈希碰撞问题,为此本文将哈希表与红黑树两者相结合,将哈希表的同义词使用红黑树来存储.

对于一棵红黑树,必须满足如下性质[9]:

(i)每个结点或者是红色,或者是黑色;

(ii)根节点只能是黑色;

(iii)任意一个叶子结点都是黑色(叶子结点一般用空指针表示);

(iv)如果某结点为红色,那么相应的儿子结点为黑色;

(v)对于任意一个结点,从这个结点出发,到达任意一个叶子结点所经过的单向路径上的黑色结点数目是相同的.

为此,将订单的真正信息保存在红黑树的每个节点上的,每个订单使用红或黑来标记,由于只有 2 种颜色,所以只需再增加一个二进制位来表示订单的颜色.由于红黑树通过限制路径上颜色的数目,使红黑树基本到达平衡,所以查找时只需要 $O(\log n)$ 的时间复杂度[10],效率较高.

以下为用于保存订单信息的红黑树节点的数据

结构:

```
typedef struct Rb_tree_node
{
    color_type    RBTree_color;
    //节点的颜色,非红即黑
    Base_ptr fatherNode;//父节点
    Base_ptr leftNode;//左节点
    Base_ptr rightNode;//右节点
    key_type key;//关键码,用于保存快递单号
    time_t orderTime;
    // 用于记录提交订单的时间
    time_t signTime;
    //用于记录订单签收的时间
    status_type logistics_status;
    //用于保存订单的物流状态
    bool isSign;//表示订单是否已签收
    value_type    cost;
    //用于记录订单产生的费用
} Rb_tree_node.
```

当存储订单信息的哈希表建立好后,首先根据每天的订单数量设定哈希表的长度,即哈希表的桶数目. 假设每天的平均订单数目为 M,那么将哈希表的长度设置为第一个大于等于 M 值的质数 Z,以尽量减少哈希碰撞[11]. 另外,订单数目的不断增加,势必会影响查询的效率,为此,在收货人签收完订单的 10 d 后(时间可以根据实际情况来设定),应将该订单从哈希表中删除,并将订单信息存回物流信息系统中.

2 物流运输中的基本操作

以某快递公司为例,在物流运输中对订单的主要操作有如下几方面.

2.1 订单生成

当快递公司接受到新的订单,此时快递公司应对包裹的信息进行登记,记录此时的快递单号、提交订单的时间、物流状态及签收情况、订单产生的费用,由于是新的订单,所以物流状态设为空,签收情况为未签收. 以快递单号为关键码,通过哈希函数计算出相应的哈希值,以快递单号为关键码插入到哈希表对应的红黑树中.

以下为订单生成的具体算法:

```
Generate_Orders(RBH,k)
//RBH 为存储订单集合的哈希表,k 为快递单号
i ← hash(k) //计算相应的哈希值
RBT ← buckets [i] of RBH
//找到存储订单信息的红黑树
t ← CreateRBTNode(k)
//以快递单号为关键码,创建一个红黑树节点
p ←root [RBT]
q ← NIL
while p ≠ NIL
//开始查找订单信息应插入的位置
        do  q← p
        if  k < key [p]
    then p← leftNode [p]
else p←rightNode [p]
fatherNode [p] ← q
if  q = NIL
then root [RBT] ← t
else  if  k < key [q]
//根据快递单号将订单信息插入到相应位置
then leftNode [q] ← t
else  rightNode [q] ← t
leftNode [t] ← NIL
rightNode [t] ← NIL
RBTree_color [t] ← COLOR_RED
//将订单的颜色设为红色
Adjust_After_Insert(RBT,t)
//调整红黑树以保持基本平衡.
```

2.2 查询订单

订单生成完之后,便可进行查询订单操作.

在物流信息系统中,输入快递单号,以快递单号为关键码,通过哈希函数计算出相应的哈希值. 通过哈希值找到相应的存储订单信息的红黑树,再以快递单号为关键码,从红黑树中查找出订单信息,最后将信息反馈给用户.

以下为查询订单的具体算法:

```
Search_order(RBH)
//RBH 为用于存储订单集合的哈希表
    k ← input tracking num
    //输入快递单号,记为 k
    i ← hash(k)
    //计算快递单号相应的哈希值
    RBT ← buckets [i] of RBH
    //找到存储订单信息的红黑树
    if  RBT = NIL
//如果存储订单信息的红黑树是空的,则直接返回
```

```
    then return RBT
    p ← RBT
    while(p)          //在相应的红黑树中查找订单
        if  key [p]  = k
            return p
        else if  key [p]  > k
            p ← leftNode [p]
        else p ← rightNode [p]
    return p
    //查询到订单后,将订单信息反馈给用户.
```

2.3　删除订单

对于删除订单操作,并不是指完全从物流信息系统中将该订单信息擦除,而只是从用于保存订单信息的哈希表中删除. 当订单已经签收完之后,可能还需要对订单信息进行查询,所以仍会保留一段时间. 之后才将从哈希表中删除,如果哈希表的订单只增不减,订单数目过大,那么查询的时间又会不断加长. 为此,在订单签收完 10 d 后,同样以快递单号为关键码,计算出相应的哈希值,再从相应的红黑树中将订单信息删除,同时将相应的订单信息存回到物流信息系统当中.

以下为删除订单的具体算法:

```
Delete_order(RBH,k)
// RBH 为存储订单集合的哈希表,k 为快递单号
  q←Search_order(RBH,k)
  //找到相应的订单
  ifisSign(q) //如果订单已经签收
    if  signTime [q]  - Now   > 10 days
    //如果订单签收完后超过 10 d 了
        i← hash(k)
        // 以快递单号为关键码,计算哈希值
        RBT ← buckets [i]  of RBH
        saveToDatabase(q)
        //将订单信息存回物流信息系统数据库
        deleteFromRBT(RBT,k)
        //将订单信息从相应的红黑树中删除.
```

3　性能分析

3.1　时间性能分析

用于存储订单信息的数据结构是影响物流信息系统性能的重要因素. 先假设带红黑树的哈希表的桶数为 b,总节点数为 n.

3.1.1　在查找方面　在理想情况下,所有节点平均分布在每个桶中,那么每个桶的数量为 n/b,此时查找节点的时间复杂度为 $O(\log n/b)$,最坏情况下,所有节点都集中在一个桶中,那么此时带红黑树的哈希表退化为一颗红黑树,此时的时间复杂度为 $O(\log n)$. 这比链表的平均复杂度 $O(n)$ 及红黑树的平均复杂度 $O(\log n)$ 都要更好.

3.1.2　在插入方面　与查找的复杂度一样,带红黑树的哈希表的插入操作最坏时间复杂度为 $O(\log n)$,而平均情况下为 $O(\log n/b)$,整体较传统红黑树的时间复杂度 $O(\log n)$ 要更好;而对于链表,虽然它插入节点的平均复杂度 $O(1)$,但实际上,链表的插入也依赖于查找操作,只有先找到要插入的位置,才能对节点进行删除,所以链表插入操作上的平均时间消耗仍为 $O(n)$.

3.1.3　在删除方面　与插入操作一样,带红黑树的哈希表在最坏情况下退化为一棵红黑树,其最坏的时间复杂度为 $O(\log n)$,在平均情况下为 $O(\log n/b)$,较传统的红黑树 $O(\log n)$ 要好. 而链表同插入操作一样,虽然删除操作的时间复杂度为 $O(1)$,但链表的删除同样依赖于查找操作,只有先查找到要删除的位置才能将其删除,所以链表在平均耗时上同样为 $O(\log n)$.

从以上几方面可以看出带红黑树的哈希表在各方面的耗时都较好. 且当桶数 b 等于节点数 n 时,可以到达 $O(1)$ 的时间复杂度.

3.2　空间性能分析

带红黑树的哈希表的内存开销主要指用于存储桶的空间. 假设带红黑树的哈希表中有 m 个桶,对于 32 位的操作系统来说,指针占用 4 个字节,对于数据信息都存储在红黑树当中,所以这比传统的红黑树只增加 $4 \times m$ 个字节的内存开销. 但这对于内存越来越便宜的今天,用内存换取更快的时间是非常值的.

4　实验结果

实验环境:Ubuntu 10 操作系统,CPU 2.4 GHz,1GB 内存,gcc 4.4.3.

为了加快物流信息系统的查询操作,本文采用了带红黑树的哈希表存储结构. 为检验不同的数据结构对查找性能的影响,本文进行了仿真实验. 以下为链表、红黑树及带红黑树的哈希在不同数目下的测试结果如图 2 所示.

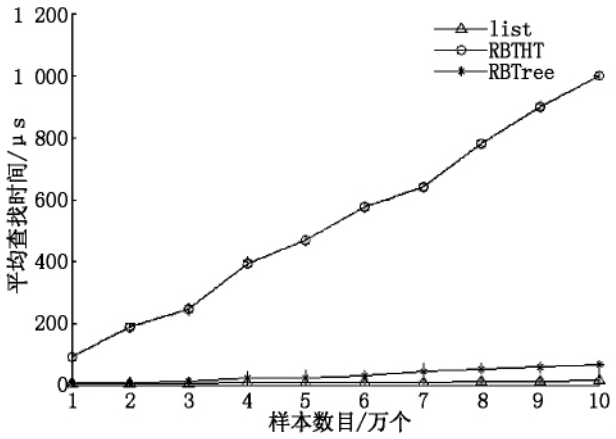

图2 不同数据结构的平均查找效率

从图2可以看出,当使用链表方式时,其查找效率成线性增长的方式,效率较差.红黑树的查找效率在样本数目较少的情况下与带红黑树的哈希表相当,但随着样本数目的增加,带红黑树的哈希表查找效率更优.

5 结论

物流信息系统,给人们带来了很多方便,但节假日期间物流业务量及查询用户的急剧增长给系统带来了沉重压力.为了解决订单查询速度慢的问题,本文提出了带红黑树的哈希表,将哈希表中的同义词用红黑树来保存,避免了由于碰撞而带来查找效率低的问题.实验结果表明,带红黑树哈希表在查找时间上优于链表及红黑树,能有效提高查找效率.

6 参考文献

[1] 康阅春.快递企业“爆仓”问题的解决方案[J].物流技术,2011(16):40-42.
[2] 张英.我国快递行业发展现状及策略探究[J].时代金融,2012(32):201-207.
[3] 王显忠.基于集散分拨的配送管理信息系统研究与实现[D].上海:上海交通大学,2008.
[4] 包辰明.基于RDF的发布/订阅系统的设计与实现[D].广州:中山大学,2011.
[5] 赖玮.通用物流管理系统的研究和实现[D].成都:电子科技大学,2008.
[6] Nicolai M. Iosuttis,C++ standard library:a tutorial and reference[M]. Boston:Addis on Wesley Longman Inc,1999:175-216.
[7] 万成威,邬江兴,李玉峰,等.CAM辅助的哈希表查找性能分析[J].电子与信息学报,2011,33(2):272-277.
[8] Cormen T H.算法导论[M].潘金贵,译.北京:机械工业出版社,2006:163-180.
[9] 唐自立.红黑树的高度[J].苏州大学学报:自然科学版,2006,22(3):33-36.
[10] 陈强璋.一种高效的二叉查找树:红黑树[J].华东师范大学学报:自然科学版,2000(3):39-42.
[11] 侯捷.STL源码剖析[M].武汉:华中科技大学出版社,2002:247-270.
[12] 皮慧娟,魏庆东.物流配送中心选址问题研究[J].江西师范大学学报:自然科学版,2011,35(5):507-511.

The Research for Hash Table with a Red-Black Tree in Logistics Information System

TENG Shao-hua,LIN Guo-hua
(School of Computer Science and Technology,Guangdong University of Technology,Guangzhou Guangdong 510006,China)

Abstract: With the rapid growth of the logistics business and a large number of users using logistics information system to check the order details during the holiday season,the logistics information system's response time becomes very long. For this problem,the red-black tree is used in the hash table to improve the speed of order-search effectively. Compared with link-list and red-black tree,the experimental results show that the hash table with red-black tree has obvious advantages at the aspect of search.

Key words: logistics information system;hash table;red-black tree;search efficiency

(责任编辑:冉小晓)