

高效 IP 路由查找实验报告

张磊 2017K8009922027

一、实验题目

高效 IP 路由查找实验

二、实验内容

1. 实现最基本的前缀查找；
2. 调研并实现某种 IP 前缀查找方案；
3. 基于 forwarding-table.txt 数据集(Network, Prefix Length, Port):
本实验仅考虑静态数据集，不考虑表的添加和更新；
以前缀查找结果为基准，检查所实现的 IP 前缀查找是否正确；
对比基本前缀树和所实现的 IP 前缀查找的性能；

三、实验流程

1. 本次实验使用 C 语言实现；
2. 编写从 forwarding-table.txt 文件中提取数据的函数；
3. 实现最基本的单比特前缀树，包括构造和查询；

```
typedef struct Node {  
    u32 ip;  
    u32 netowrk;  
    u16 prefix;           // 从0开始计算, 第prefix位不相同  
    u16 mask;  
    u16 port;  
    struct Node *left;  
    struct Node *right;  
} Node_t;
```

单比特前缀树节点

4. 验证单比特前缀树查找结果的检查；

5. 在单比特前缀树的基础上实现双比特前缀树，包括构造和查询；

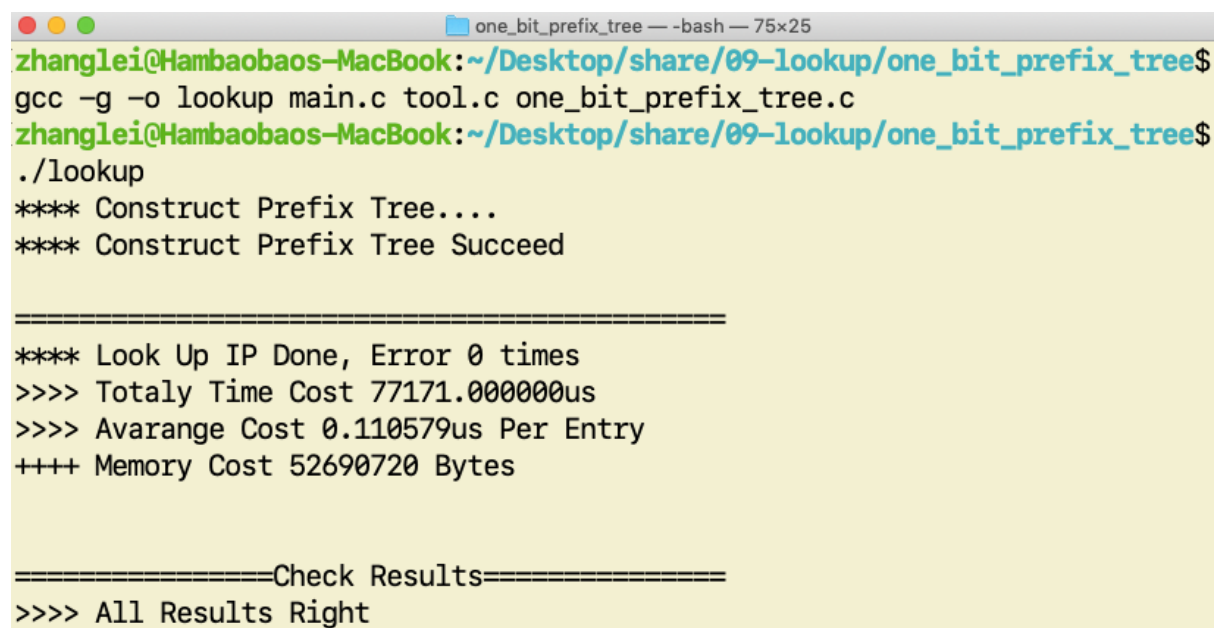
```
typedef struct Node {
    u32 ip;
    u16 prefix;           // 从0开始计算，第prefix位不相同
    u16 odd_port;
    u16 even_port;
    struct Node *child00;
    struct Node *child01;
    struct Node *child10;
    struct Node *child11;
} Node_t;
```

双比特前缀树节点

6. 验证双比特前缀树的查找结果；

四、实验结果

1. 单比特前缀查找：



```
one_bit_prefix_tree -- -bash -- 75x25
zhanglei@Hambaobaos-MacBook:~/Desktop/share/09-lookup/one_bit_prefix_tree$
gcc -g -o lookup main.c tool.c one_bit_prefix_tree.c
zhanglei@Hambaobaos-MacBook:~/Desktop/share/09-lookup/one_bit_prefix_tree$
./lookup
**** Construct Prefix Tree....
**** Construct Prefix Tree Succeed

=====
**** Look Up IP Done, Error 0 times
>>>> Totaly Time Cost 77171.000000us
>>>> Avarange Cost 0.110579us Per Entry
++++ Memory Cost 52690720 Bytes

=====Check Results=====
>>>> All Results Right
```

单比特前缀树查找结果

2. 双比特前缀查找:

```
zhanglei@Hambaobaos-MacBook:~/Desktop/share/09-lookup/two_bits_prefix_tree$  
gcc -g -o lookup main.c tool.c two_bits_prefix_tree.c  
zhanglei@Hambaobaos-MacBook:~/Desktop/share/09-lookup/two_bits_prefix_tree$  
./lookup  
**** Construct Prefix Tree....  
**** Construct Prefix Tree Succeed  
  
=====  
>>>> Look Up IP Done, Error 0 times  
>>>> Totaly Time Cost 53801.000000us  
>>>> Avarange Cost 0.077092us Per Entry  
++++ Memory Cost 51514944 Bytes  
  
=====  
====Check Results=====
```

双比特前缀树查找结果

五、 实验分析

1. 对比单比特和双比特前缀树的结果，发现虽然双比特前缀树消耗的内存只比单比特前缀树少了一点，几乎可以忽略不计，但是双比特前缀树的查询速度相比单比特前缀树却提升了约 30%;
2. 我尝试了删除双比特前缀树节点结构体中的一些不需要的条目，但是消耗的内存大小却没有变小，原因应该是结构体才用了 64 字节对齐，但是我定义的结构体的大小小于 64 字节，所以没有变化;
3. 如果想要进一步减小内存消耗，只能通过压缩前缀树的方法，但是由于时间关系，我实现的双比特压缩前缀树还有 bug，所以没法提交;
4. 为了不影响查询的速度，对查询结果的检查放到了程序的最后统一检查，本次实验中对内存消耗的统计只计算了 malloc 函数分配的前缀树的节点消耗的内存，本次实验使用的系统是 macOS 10.14.6 没有在此前的 ubuntu 系统下进行，实验结果在不同环境下可能会有微小差别;

六、 反思总结

1. 本次实验理论上很简单，思路也很清楚，但是实现起来的时候却遇到了很多 bug，有些是一开始考虑问题的时候没有考虑全面，等到程序写完了，运行的时候出了问题，才发现还有问题；
2. 自从去年数据结构课以后就没怎么写过树结构的程序，所以这次开始写的时候由于指针的使用出了许多问题，还好最后都解决了，看来是时候好好复习一下数据结构了；
3. 在双比特前缀树的实现中，由于 prefix_len 为奇数的条目可能和 prefix_len 为偶数的条目产生重复，如果依然采用单比特前缀树的结构会出现后添加的条目覆盖之前添加的条目的情况，所以在双比特的前缀树中，我把 port 氛围了 odd_port 和 even_port 两个，分别表示当 prefix_len 为奇数和偶数时的转出端口；

七、 参考文献

i

ii

ⁱ 中国科学院大学 2020 春计算机网络研讨课 09-高效 IP 路由查找实验课件

ⁱⁱ 中国科学院大学 2020 春计算机网络研讨课 09-高效 IP 路由查找实验附件