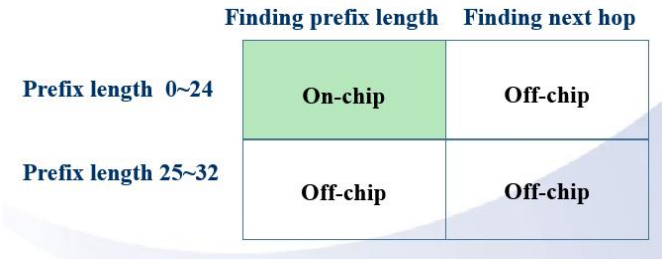
基于 SAIL 算法的 FIB 查表详细说明

一、 FIB 查表概述

简介

骨干路由器的转发信息库（FIB）规模迅速扩大。理想的 IP 查找算法应该能够实现恒定而小的 IP 查找时间和片上存储器使用。但是，以前的 IP 查找算法不能同时满足这两个要求。作者提出的 SAIL，一是种 IP 查找的分裂方法，分两个维度对查找过程拆分。一个维度沿着即查找前缀长度和查找下跳拆分，另一个拆分沿着前缀长度拆分。对于存储容量受限的片上存储，只在片上存储前缀长度小于等于 24 的 IP。作者在 SAIL 的基础上又提出了面向查找优化、面向更新优化以及面向多 FIB 查表优化的算法。经过在 CPU，FPGA，GPU 和多核平台上的实验，结果表明，SAIL 算法比众所周知的 IP 查找算法快几倍甚至两个数量级。



图一 SAIL 算法基本思想

功能描述

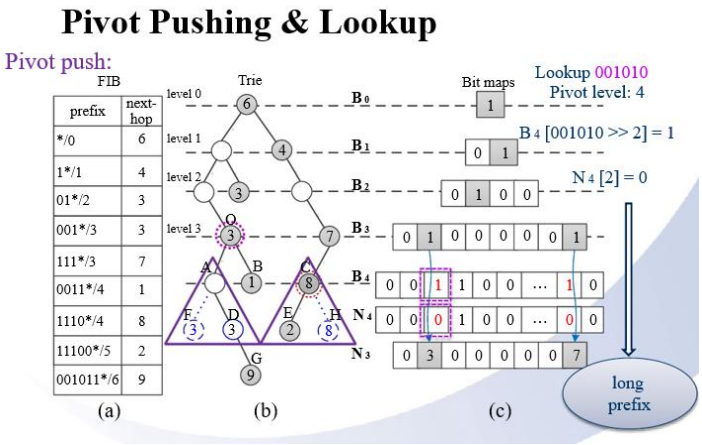
SAIL 是一种 IP 查找的分裂方法。它将 IP 查找问题沿着两个维度分解，如图一所示。首先，沿着查找过程的维度将 IP 查找问题分解为两个子问题：1.查

找前缀（即查找长度与给定 IP 地址相匹配的最长前缀）；2.找到下一跳（即查找最长匹配前缀的下一跳）。

其次，将 IP 查找问题分成前缀长度维度的两个子问题：1.长度≤24；2.长度≥25。这种划分是基于骨干路由器上 几乎所有的流量的最长的匹配的前缀长度≤24。这种假设前提在骨干路由器上通常是成立的，通过这种分裂使 SAIL 算法能够专注于优化最长匹配前缀长度≤24 的流量的 IP 查找,提　　的查表转发性能。

系统模型

给定一个 IP 地址，因为它可能匹配一个长前缀，SAIL 提出了一种称为 pivot pushing 的技术来解决这个问题。基本策略是对于给定的 IP 地址，首先测试其最长匹配前缀长度在[0,24]还是[25,32]之内;在这次测试之后，就可以只在短前缀或长前缀中搜索，但不是两者都搜索。级别 24 称为（pivot level）枢轴级别，图二是假定枢轴级为 4　SAIL　。



图二 SAIL 模型

为了进一步提高查找速度，作者又提出了面向查找的优化。通过前缀扩展，首先将所有在 0～15 级的实体节点（在路由表中有匹配前缀的节点）推到 16 级;其次，将 16 级的所有内部节点和 17～23 级的所有实体节点推到 24 级;第三，把 24 级的所有内部节点和 25 级到 31 级的所有实体节点都推到了 32 级。称之为 3 级推送。对于级别 16，数据结构有三个数组：位图数组 B16 [0..2^{16} -1]，下一跳数组 N16 [0..2^{16} -1]和块 ID 数组 C16 [0..2^{16}-1] ，其中块 ID 从 1 开始。对于级别 24，数据结构具有三个数组：位图数组 B24，下一跳数组 N24 和块 ID 数组 C24，其中每个数组的大小是 16 上的内部节点的数量乘 2^8.对于 32 级，数据结构有一个数组：下一跳数组 N32，其大小是级别 24 内部节点的数量乘 2^8。

二、 SAIL 软件实现架构

为了获得真实的 FIB，作者使用服务器与中国的一级路由器建立对等关系，以便服务器可以从一级路由器接收 FIB 更新，但不会向一级路由器通告新的前缀;因此，服务器逐渐从第一层路由器获得整个 FIB。在服务器上，作者使用开源的 Quagga 来每小时转储 FIB。 2013 年 10 月 22 日上午 8 点至 10 月 23 日下午 21 点之间以每小时捕捉 10 分钟流量的间隔捕获一个一级路由器接口中的实际流量。 另外，作者从 www.ripe.net 下载了 18 个真正的 FIB。其中 6 个是从 2008 年到 2013 年每年 1 月 1 日的上午 8 点时的 FIB。另外 12 个分别于 2013 年 8 月 8 日上午 8:00 从 12 台路由器下载。作者还生成了 37 个综合流量 IP。前 25 个包含随机选择目的地的数据包。其他 12 个是在 2013 年 8 月 8 日 08:00 AM 从 12 台路由器下载的 12 个 FIB 中为每个前缀均匀生成流量而获得的;因此，可保证每个前缀都有相同的机会被合成流量击中。

作者在四个指标上评估了 SAIL 算法：以 pps（每秒数据包数量）为单位的查找速度，以 MB 为单位的片上存储器大小，以微秒为单位的查找延迟以及以每次更新的访存总数为单位的更新速度。

SAIL 在论文中包括 CPU、GPU、FPGA 以及多核的实现。对于片上存储器大小，只能评估 SAIL 算法的 FPGA 实现。对于算法性能，作者使用 C ++实现了 SAIL 算法和 PBF、LC-trie，Tree Bitmap、Lulea 的比较。并通过穷举搜索验证了所有算法的正确性：首先构造一个详尽的 2^{32} = 4G 查找表，词表只存储下一跳地址，0~2^{32} 的任意 IP 地址下一跳对应该表中　　条目;其次，对于每个 IP 地址，将查找结果与此穷举表查找的结果进行比较，实验都通过了此验证。

三、 SAIL 接口

输入文件及其格式

1.根据路由表构建比特图

IP 地址	前缀长度	下一跳端口
1.0.224.0	/19	48
1.5.0.0	/16	136
1.8.1.0	/24	96

一

查表输出格式

比特图构建好后，输入任意 IP 地址，给出下一跳端口号：

任意 IP 地址表示为 32 位无符号整型数

Unsigned int	IP 地址(二进制)	IP 地址
17327872	0001.00001000.01100111.00000000	1.8.103.0
1190860032	01000110.11111011.00010101.00000000	70.251.21.0

表二

构建比特图接口：

输入 fib_file 由图一中的 50 万条信息构建

存储流量信息：

输入表二中的 IP 地址，存为数组，供进一步处理。

IP 查表接口：

给定一条流量包的 IP，返回 fib 表中匹配的最长前缀对应的端口：

```
LPMPort=tFib.sailLookup(traffic[i]);
```

更新接口：

根据更新文件（如表三）更新比特图：

```
int updateEntryCount = BFLevelPushingTrieUpdate(updateFile, &tFib);
```

年月日	时分秒	是否 A/W	IP 地址	前缀长	下一跳
20120101	80010	A	203.192.217.0	/24	71
20120101	80010	W	203.192.248.0	/23	
20120101	80010	A	188.65.208.0	/21	90
20120101	80012	W	143.73.207.0	/24	

表三

四、SAIL 软件系统数据类型及结构体

CFib 类

```
class CFib
{
public:
    FibTrie* m_pTrie;                   //Fib 数的根
    int allNodeCount;                   //记录 Fib 树的所有节点，包括根节点
    int solidNodeCount;                     //记录 Fib 树的所有实节点
    int LevelPort[40][200];

    unsigned int prefix32_num;
    unsigned int lenofchain;

    /*********************************Trie Table*************************************/
    unsigned int currentChunkNum24;//#   级别  24 的  chunk 个数
    unsigned int currentChunkNum32;//# 级别  32 的  chunk 个数

    struct sailTable_16  *levelTable16;//table in level 16
    struct sailTable_24  *levelTable24;//table in level 24
    struct sailTable_32  *levelTable32;//table in level 32

    struct segInfo_16       segmentUpdate16;  //the updating segment in level 16
    struct segInfo_24       segmentUpdate24;  //the updating segment in level 24
    struct segInfo_32       segmentUpdate32;  //the updating segment in level 32

    unsigned int                chunkUpdateNum24;         //# of the updating chunks in level 24
    unsigned int                chunkUpdateNum32;         //# of the updating chunks in level 32

    struct chunkInfo            chunkUpdate24[CHUNK_MAX_24];   //the updating chunks in level 24
    struct chunkInfo            chunkUpdate32[CHUNK_MAX_32];   //the updating chunks in level 32
    /*********************************End*************************************/


    /*********************************CDF stat*************************************/
    //unsigned int CBFInsertNum;
    //unsigned int CBFDelNum;
    unsigned int trueUpdateNum;                         // true total update item
    unsigned int invalid;
    unsigned int invalid0;
    unsigned int invalid1;
```

```
    unsigned int invalid2;

    unsigned int deepest;

    uint64_t memory_access;

    //unsigned int CBFInsertArray[25000];
    //unsigned int CBFDeleteArray[25000];
    /*********************************End**********************************/


    CFib(void);
    ~CFib(void);

    // creat a new FIBTRIE ndoe
    void CreateNewNode(FibTrie* &pTrie);
    //get the total number of nodes in RibTrie
    void ytGetNodeCounts();
    void Pretraversal(FibTrie* pTrie);
    //output the result
    void OutputTrie(FibTrie* pTrie,string sFileName,string oldPortfile);
    void OutputTrie_32(FibTrie* pTrie);

    bool IsLeaf(FibTrie * pNode);
private:
    //get and output all the nexthop in Trie
    void CFib::GetTrieHops(FibTrie* pTrie,unsigned int iVal,int iBitLen,ofstream* fout,bool ifnewPort);
    void CFib::GetTrieHops_32(FibTrie* pTrie,unsigned int iVal,int iBitLen,ofstream* fout,bool ifnewPort);

public:
    unsigned int BuildFibFromFile(string sFileName);
    //void Update(int insertport, char *insert_C, int operation_type);
    void Update(int insertport, char *insert_C, int operation_type, char *sprefix);    // used for test. By Qiaobin
Fu

    int num_level[50];

    void CFib::LevelStatistic(FibTrie* pTrie, unsigned int level);

    unsigned int CFib::btod(char *bstr);

    /*********************************Trie Table**********************************/
    void sailTableInit();
    void updateInfoInit();
```

```
    void subTrieChunkUpdate(FibTrie* root, unsigned int prefix, struct subTrieUpdateArg *arg);
    unsigned int sailLookup(unsigned int ip);
    unsigned int * TrafficRead(char *traffic_file);
    //unsigned int sailPerformanceTest(char *traffi_file, char* fib_file);
    unsigned int getBitsValue(unsigned int prefix, unsigned char prefixLen, unsigned char start, unsigned char len);

    void checkTable(FibTrie *root, unsigned int prefix);
    void trieDestroy(FibTrie *root);
    /************************************End*****************************************/


    /*****************************BFstat*****************************************/
    unsigned int GetAncestorHop(FibTrie* pTrie);
    //void subTrieUpdate(FibTrie* root, unsigned int default_port, int operation);
    unsigned int subTrieUpdate(FibTrie* root, unsigned int default_port, unsigned int prefix, bool ifUpdate, struct subTrieUpdateArg *arg);    //
    void subTrieLevelPushing(FibTrie* root, unsigned int default_port, unsigned int prefix, struct subLevelPushArg* arg);
    bool isTheRange(unsigned int prefixLen, unsigned int visitedNodeNum);  // 2014-05-07, Qiaobin Fu
    void ytTriePortTest(FibTrie* pTrie);
    bool isCorrectTrie(FibTrie* pTrie);
    /*******************************End*************************************************/
};
```

分割及枢轴推送相关结构体：

```
struct sailTable_16 {      //struct in level 16
    short element[LEVEL16_ELE_NUM]; // >0 --- offset; <0 --- nexthop;
};

struct sailTable_24 {      //struct in level 24
    unsigned char nexthop[LEVEL24_CHUNK_NUM * CHUNK];  // flag+next-hop array
    unsigned short offset[LEVEL24_CHUNK_NUM * CHUNK];    // offset array
};

struct sailTable_32 {      //struct in level 32
    unsigned char nexthop[LEVEL32_CHUNK_NUM * CHUNK];  // next-hop array in level 32
};

struct segInfo_16 {
    int chunk_id;
    short nexthop[CHUNK];
};
```

```
struct segInfo_24 {
	int chunk_id;
	unsigned char nexthop[CHUNK];  // the updating chunk id and the value of segment to update    in level 24
};

struct segInfo_32 {
	int chunk_id;
	unsigned char nexthop[CHUNK];  // the updating chunk id and the value of segment to update    in level 32
};

struct chunkInfo {
	int chunk_id;
	unsigned char nexthop[CHUNK];  // the updating chunk id and the value to update    in level 24 or 32
};

struct subLevelPushArg {
	int bornStatus;
	int chunkID_24;
	int chunkID_32;
};

struct subTrieUpdateArg {
	unsigned char segLevel; //16¡¢24¡¢32
	unsigned char length;
};
```

五、SAIL 软件系统功能

CFib 方法

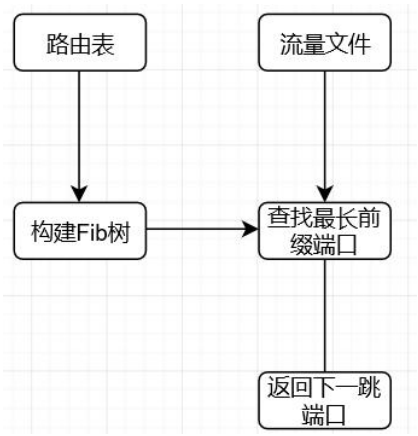
unsigned int CFib::BuildFibFromFile(string sFileName)
void CFib::checkTable(FibTrie *root, unsigned int prefix)
void CFib::LevelStatistic(FibTrie* pTrie, unsigned int level)
unsigned int CFib::btod(char *bstr)
bool CFib::IsLeaf(FibTrie * pNode)
void CFib::Pretraversal(FibTrie* pTrie)

void CFib::ytGetNodeCounts()
void CFib::OutputTrie(FibTrie* pTrie,string sFileName,string oldPortfile)
void CFib::OutputTrie_32(FibTrie* pTrie)
void CFib::GetTrieHops_32(FibTrie* pTrie,unsigned int iVal,int iBitLen,ofstream* fout,bool ifnewPort)
void CFib::GetTrieHops(FibTrie* pTrie,unsigned int iVal,int iBitLen,ofstream* fout,bool ifnewPort)
unsigned int CFib::BuildFibFromFile(string sFileName)

void CFib::subTrieChunkUpdate(FibTrie* root, unsigned int prefix, struct subTrieUpdateArg *arg)
unsigned int CFib::subTrieUpdate(FibTrie* root, unsigned int default_port, unsigned int prefix, bool ifUpdate, struct subTrieUpdateArg *arg) {
void CFib::subTrieLevelPushing(FibTrie* pTrie, unsigned int default_port, unsigned int prefix, struct subLevelPushArg* arg) {
bool CFib::isTheRange(unsigned int prefixLen, unsigned int visitedNodeNum)
void CFib::Update(int insertport, char *insert_C, int operation_type, char* sprefix)
unsigned int CFib::sailLookup(unsigned int ip)
unsigned int * CFib::TrafficRead(char *traffic_file)
void CFib::ytTriePortTest(FibTrie* pTrie)
bool CFib::isCorrectTrie(FibTrie *pTrie)

六、SAIL 软件系统处理流程

查表流程图：



总体运行情况：

```
sail algorithm starts...

    trace read complete...
        frequency=3897229
        LMPport=201
        Lookup time=10248371
        Throughput is:   97.576 Mpps

    Parsing updates1.txt
        update performance: readline=677923     time=672037us
        speed=1.0087584 Mups
        Destroying the trie......
        The trie has been destroyed!

Mission complete, Press any key to exit...              Stage One: The Initial Trie Constructi
```

更新情况：

```
Mission complete, Press any key to exit...              Stage One: The Initial Trie Construction

The total number of Trie node is :      131071.
The total number of solid Trie node is :        65536.

**********************Trie Correct Test**********************
The trie structure is correct!
*****************************End*****************************


                Stage Two: The First Round Update
Open file rib.txt error!

The total number of Trie node is :      131071.
The total number of solid Trie node is :        65536.
The total number of routing items in FRib file is :     0.

**********************Trie Correct Test**********************
The trie structure is correct!
*****************************End*****************************


                Stage Three: The Second Round Update

    Parsing updates1.txt
        update performance: readline=677923     time=60899954us
        speed=0.0111317 Mups

The total memory access is :    21269317.
The total number of Trie node is :      2489821.
The total number of solid Trie node is :        1244911.
The total number of updated routing items is :  677923.

**********************Trie Correct Test**********************
The trie structure is correct!
*****************************End*****************************


                Update Statistics

The total number of true update items is :      113192.
The total number of invalid update items is :   564731.
The detailed invalid items:
        invalid0 = 7239 invalid1 = 541791       invalid2 = 15701
```

遍历 Fib 树验证：

```
************************sail Lookup Correct Test************************
                total   4294967296
                length  cycles          percent nexthop
                32      429490000023%   100.00% 177
                Total number of garbage roaming route: 0
                Equal!!!!
*********************************End*********************************

Mission Complete, Press any key to continue...
请按任意键继续. . .
```

七、软件代码文件列表

测试文件结构

Fibtire 树结构和 CFib 类头文件

枢轴级 sailTable_16/24/32 结构定义头文件

CFib 类子函数文件

主测试程序