清华大学

位图索引编码的研究

-在大数据中的应用

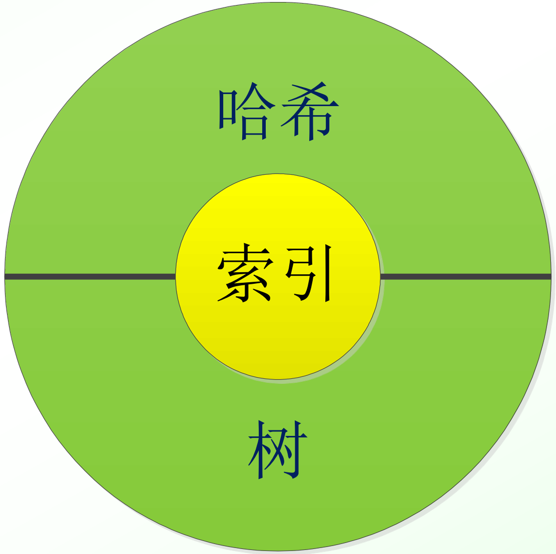
Saturn

www.nslab-Saturn.net

2014年05月09日

索引

- 人类进入大数据时代，数据查找离不开索引
- 传统索引使用哈希和树这两类最基本的数据结构

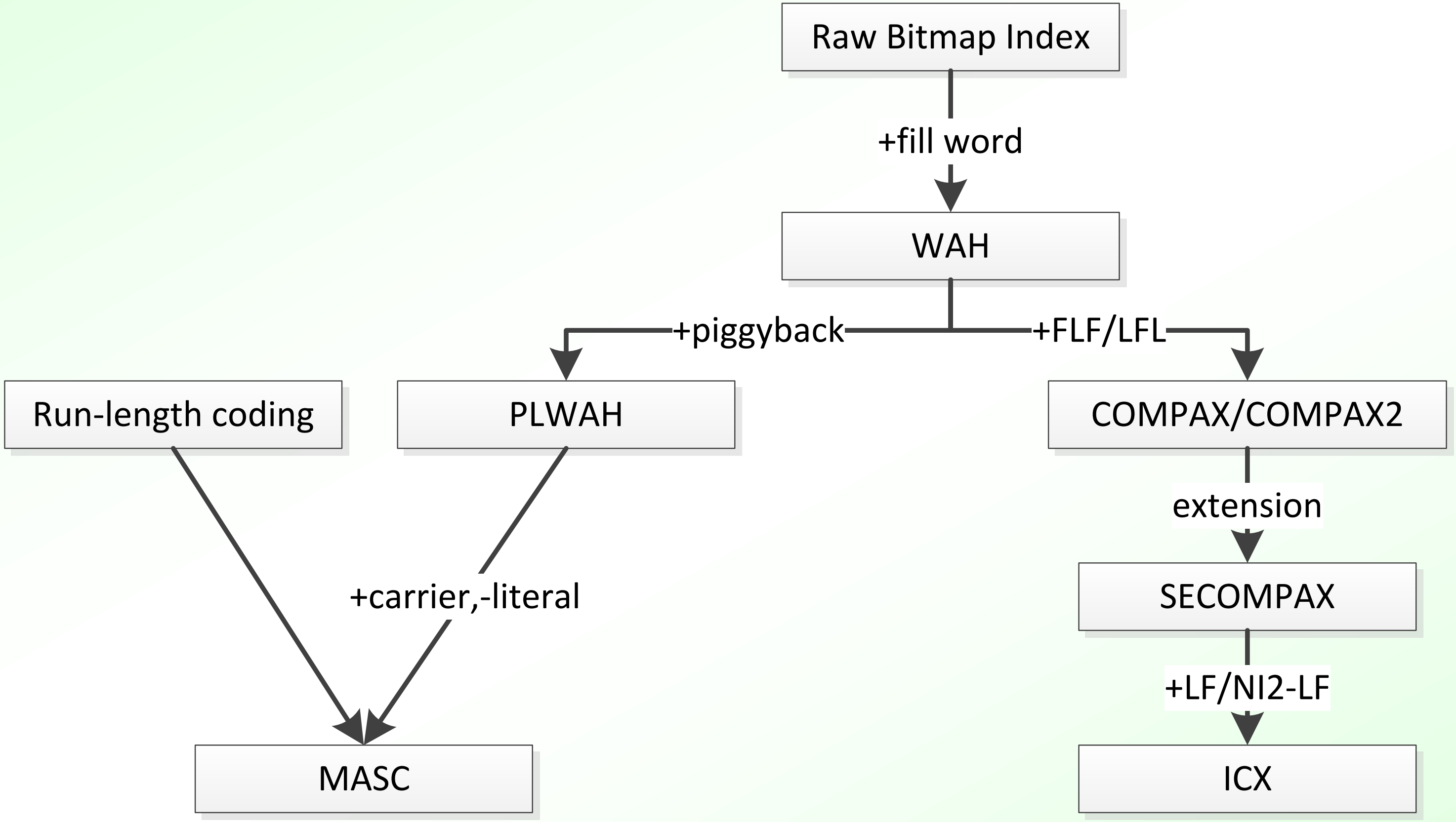


位图索引 Bitmap Index

- 位图索引特点
  - 数据查找操作灵活
  - 数据查找速度快
  - 空间消耗大
- 位图索引编码
  - 压缩索引大小
  - 加速索引查找
- 位图索引应用
  - 广泛应用于数据管理，如数据仓库， NoSQL等

row ID	X	bitmap index b_0 =0	b_1 =1	b_2 =2	b_3 =3
1	0	1	0	0	0
2	1	0	1	0	0
3	3	0	0	0	1
4	2	0	0	1	0
5	3	0	0	0	1
6	3	0	0	0	1
7	1	0	1	0	0
8	3	0	0	0	1

提出的位图索引编码算法之间关联



算法全称

- WAH: Word-Aligned Hybrid
- PLWAH: Position List Word-Aligned Hybrid
- COMPAX: COMPressed Adaptive indeX format
- SECOMPAX: Scope Extended COMPressed Adaptive indeX
- ICX: Improved CompaX
- MASC: MAximized Stride with Carrier

常用术语

- Chunk: 31比特为单位的块
- Word: 32比特字
- Fill： 一个word/chunk全部为0或1
- Literal: 一个word/chunk不全为0或1
  – Literal顾名思义”字面意思”，表示该信息未做处理，原封不动保留

0-BBC (Byte-aligned Bitmap Code)

- Oracle数据库的位图索引编码
- "The bitmap bytes are classified as *gaps* containing only zeros or only ones and *maps* containing a mixture of both. Continuous gaps are encoded by their byte length and a fill bit differentiating between zero and one gaps. Stretches of map bytes encode themselves and are accompanied with the stretch length in bytes. A pair *(gap, map)* is encoded into a single *atom* composed of a *control byte* followed by optional gap length and map. Single *map byte* having only one bit different than the others is encoded directly into a control byte using a "different bit" position within this map byte."

1-WAH(Word-Aligned Hybrid)

- 每31bits分块，基于块单位进行压缩，全0为0-fill，全1为1-fill，否则为literal（直接保留）.
- 相邻的一串连续0-fill压成一个0-fill word，相邻的一串连续1-fill压成1个1-fill word.
- WAH核心是两种方法：（1）Run Length Encoding(RLE); （2）不能RLE部分直接保留

128 bits	1*1,20*0,3*1,79*0,25*1			
31-bit groups	1,20*0,3*1,7*0	62*0	10*0,21*1	4*1
literal (hex)	40000380	00000000 00000000	001FFFFF	0000000F
WAH (hex)	40000380	80000002	001FFFFF	0000000F

	uncompressed (in 31-bit groups)				
A	40000380	00000000	00000000	001FFFFF	0000000F
B	7FFFFFFF	7FFFFFFF	7C0001E0	3FE00000	00000003
C	40000380	00000000	00000000	00000000	00000003
	compressed				
A	40000380	80000002		001FFFFF	0000000F
B	C0000002		7C0001E0	3FE00000	00000003
C	40000380	80000003			00000003

经典算法1-WAH

- **优点：**
  - 简单有效，速度快
- **不足：**
  - bit序列分段存储→ WAH Fill’s counter表示空间使用不够→ 有空间富余 → 可以用于piggyback 合适的“literal”
  - bit序列连续0或1的数目不够大→ WAH Fill’s counter表示空间使用不够→ 有空间富余→ 可以用于piggyback 合适的“literal”
- **进一步改进：PLWAH/COMPAX**
  - F和L组合的二次合并压缩，为优化指明方向
  - 出现各种新的算法，SECOMPAX/ICX/MASC

2-PLWAH(Position List Word-Aligned Hybrid)

- **在WAH的基础上**
- **1. 引入NI概念**
  - Nearly Identical(几乎等同): Literal word几乎等同于Fill word，可合并压缩
  - 细分Literal word 为nearly identical 0-fill word 和普通literal word
  - 本质上是”dirty bit” 的思想
- **2.二次压缩**
  - 用piggyback将NI的Literal合并入Fill word
  - “piggyback”:如果0-fill序列下一个literal中只有一个非0比特，那么将这个“1” piggyback到前一个fill word中.
  - 如果有多个“1”，可以把位置列表(Position List) piggyback到前一个fill word中

经典算法2-PLWAH

Uncompressed bitmap organized in groups of 31 bits:

```
0000000000 0000000000 0000000000 0
0000000000 0000000001 0000000000 0
0000000000 0000000000 0000000000 0
0000000000 0000000000 0000000000 0
0000000100 0000000000 0000000000 0
0000000000 0000000100 0000000000 0
```

31 bits　11 bits

Merging consecutive homogenous groups:

```
0000000000 0000000000 0000000000 0
0000000000 0000000001 0000000000 0
0000000000 0000000000 0000000000 0
0000000000 0000000000 0000000000 0
0000000100 0000000000 0000000000 0
0000000000 0000000100 0000000000 0
```

2 groups merged

Encoding 32 bits fill words:

1 0 0000000000 0000000000 0000000001	0 Fill word, counter = 1
0 0000000000 0000000001 0000000000 0	Literal word
1 0 0000000000 0000000000 0000000010	0 Fill word, counter = 2
0 0000000100 0000000000 0000000000 0	Literal word
0 0000000000 0000000100 0000000000 0	Literal word

Encoding sparse 32 bits literal words:

1 0 1010000000 0000000000 0000000001	0 Fill word, cnt = 1, pos = 20
1 0 0100000000 0000000000 0000000010	0 Fill word, cnt = 2, pos = 8
0 0000000000 0000000100 0000000000 0	Literal word

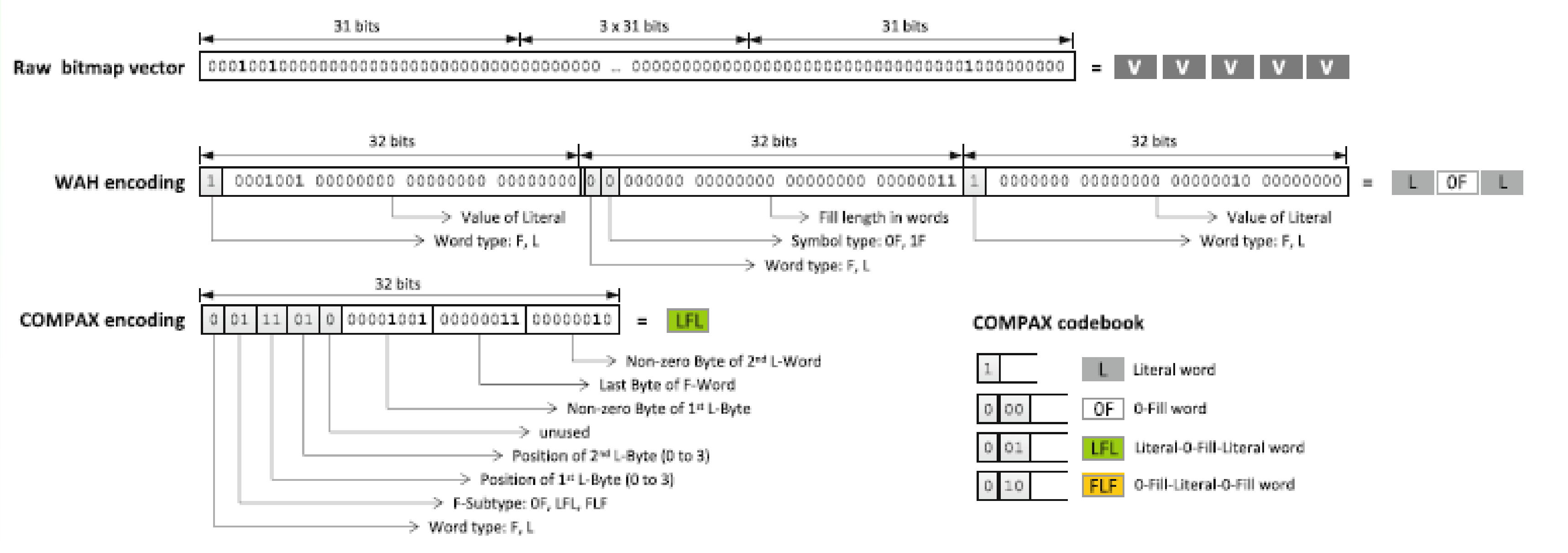
经典算法2-PLWAH

- **优点：**
  - 减少WAH中literal word出现的数量，节省空间
- **缺点：**
  - piggyback了position的信息，position的值为(0-31)，编码需5bit，开销较大
  - 挤压了counter空间，需设计Adaptive Counter,在piggyback较多position时尤为明显
  - 1.piggyback最多可以带5个(position)
  - 2.Adaptive Counter(相当于将2个或更多fill word的counter合并
- **进一步改进：COMPAX**

3-COMPAX(COMPressed Adaptive indeX)

- **在PLWAH基础上进行改进**
- **1. 引入"dirty byte" 概念**
  – dirty byte概念:相对全0 chunk,所有的非零比特在同一个byte内
  – 沿用nearly identical literal word概念，细分Literal word类型
  – "dirty byte"本质是更小的literal，是PLWAH中 "dirty bit"的推广
- **2.增添了码本，二次合并压缩**
  – 加入了LFL(literal-fill-literal)以及FLF(fill-literal-fill)
  – LFL：两个L都只有一个dirty byte（但F限定为0-fill）
  – FLF：L只有一个dirty byte

经典算法3-COMPAX



经典算法3-COMPAX改进

- 优点：
  - 增加编码类型，提高压缩率
  - 增强版本-COMPAX2
    - 在0-fill基础上，增加1-fill
    - 增加LFL中literal-1 fill-literal类型
- 进一步改进：SECOMPAX

4-SECOMPAX(Scope Extended COMPAX)

- **以COMPAX2为基础**
- **1. 扩充”dirty byte”定义**
  - 扩充”dirty byte”定义，新定义带dirty byte的nearly identical literal (NI-L)
  - 除原来是和全零chunk比所有非零位均在一个byte中，即0-NI-L
  - 增添和全一chunk比所有非一位均在一个byte中的dirty byte类型，即1-NI-L
- **2. 扩展LFL和FLF组合类型**
  - LFL：除了0-NI-L + F + 0-NI-L类型外，补充三种类型，即0-NI-L + F + 1-NI-L， 1-NI-L + F + 0-NI-L， 1-NI-L + F + 1-NI-L；
  - FLF：除了0F + 0-NI-L + 0F与1F+0-NI-L+1F 两种外，补充六种类型，即0F + 1-NI-L + 0F，1F + 1-NI-L + 1F； 1F + 1-NI-L + 0F， 0F + 1-NI-L + 1F； 1F + 0-NI-L + 0F， 0F + 0-NI-L + 1F；

新算法-SECOMPAX

Origin sequence	0000000 00000000 00111010 00000000 00...0(3*31) 1011010 00000000 00000000 00000000					
WAH encoding	1	0000000 00000000 00111010 00000000	0	0000000 00000000 00000000 00000011	1	1011010 00000000 00000000 00000000
COMPAX encoding	001	01110	00111010 00000011 01011010			
SECOMPAX encoding	001	00111	00111010 00000011 01011010			

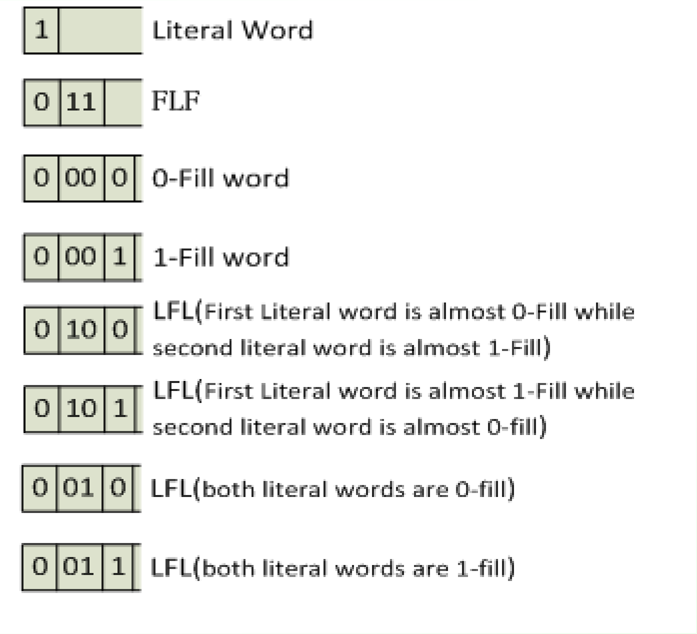
Origin sequence	00...0(7*31) 1111111 11001001 11111111 11111111 11...1(3*31)					
WAH encoding	0	0000000 00000000 00000000 00000111	1	1111111 11001001 11111111 11111111	0	0000000 00000000 00000000 00000011
COMPAX encoding	000	00000 00000000 00000000 00000111	1	1111111 11001001 11111111 11111111	011	00000 00000000 00000000 00000011
SECOMPAX encoding	011	01110	00000111 11001001 00000011			

Origin sequence	1111111 11111111 11000111 11111111 00…0(3*31 bits) 0011111 11111111 11111111 11111111					
WAH encoding	1	1111111 11111111 11000111 11111111	0	0000000 00000000 00000000 00000011	1	0011111 11111111 11111111 11111111
COMPAX encoding	1	1111111 11111111 11000111 11111111	000	00000 00000000 00000000 00000011	1	0011111 11111111 11111111 11111111
SECOMPAX encoding	001	10111	11000111 00000011 10011111			

- 在位图中“1”出现比例占多，出现NI近F的情况下，SECOMPAX算法是COMPAX算法的***3***倍

新算法-SECOMPAX

- 核心思想：编码0-NI-L与1-NI-L



新算法-SECOMPAX

- **优点：**
  - 将COMPAX提出的LFL与FLF概念进一步推广，相对COMPAX有更出色的压缩率
  - 在0和1局部数量相近时效果更加明显
- **进一步改进：ICX**

新算法-ICX(Improved COMPAX)

- **在SECOMPAX基础上进一步改进**
- **进一步细分literal word**
  - ”dirty bytes” 数目扩充为2
  - ”dirty bytes”的位置组合共6种，即4中取2的组合
- **补充两个新组合类型(LF与NI2-LF)**
  - LF：一个nearly identical的literal + F，相当于对LFL情况补充
  - NI2-LF：带2个dirty byte的literal + F
- **设计新码本**

新算法-ICX(Improved COMPAX)

LF:

Origin sequence	1111111 11111111 11000111 11111111 00...0(3*31 bits) 0011111 11111111 00011111 11111111					
WAH encoding	1	1111111 11111111 11000111 11111111	0	0000000 00000000 00000000 00000011	1	0011111 11111111 00011111 11111111
COMPAX encoding	1	1111111 11111111 11000111 11111111	000	00000 00000000 00000000 00000011	1	0011111 11111111 00011111 11111111
ICX encoding	00001	101	11000111 00000000 00000011	1	0011111 11111111 00011111 11111111	

NI2-LF：

Origin sequence	1111111 11001111 11000111 11111111 11...1(11*31 bits)			
WAH encoding	1	1111111 11001111 11000111 11111111	0	1000000 00000000 00000000 00001011
COMPAX encoding	1	1111111 11001111 11000111 11111111	011	00000 00000000 00000000 00001011
ICX encoding	0001	1011	11000111 11001111 10001011	

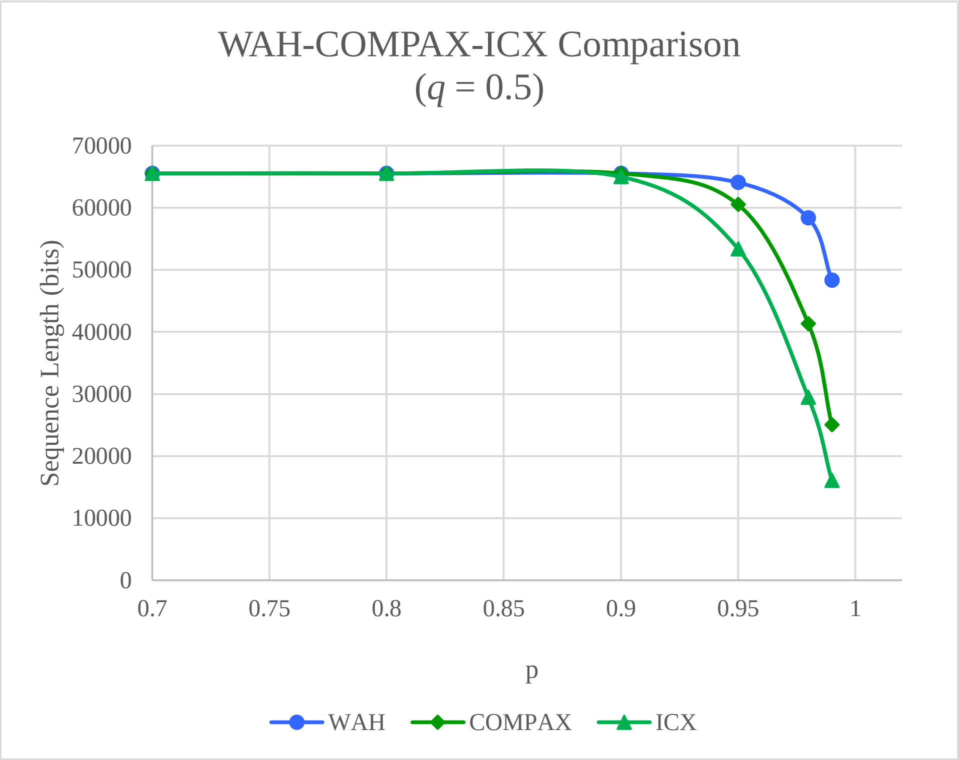
新算法-ICX (Improved COMPAX)

- 核心思想：编码2个dirty Bytes
- 码本codebook：

1	Literal Word
0 11	FLF
0 10 0	LFL(First Literal word is almost 0-Fill while second literal word is almost 1-Fill)
0 10 1	LFL(First Literal word is almost 1-Fill while second literal word is almost 0-fill)
0 01 0	LFL(both literal words are almost 0-fill)
0 01 1	LFL(both literal words are almost 1-fill)
0 00 1	NI2-LF(Literal word has two dirty bytes)
0 00 0 1	LF (Literal word is almost 0-fill/1-fill)
0 00 0 0	Fill Word

新算法-ICX(Improved COMPAX)

- 数据集:
- 概率生成01比特串
- p:
- q:



新算法-ICX(Improved COMPAX)

- **优点：**
  - 在SECOMPAX基础上，增加了可编码的类型
  - 在0/1局部数量相近（在同一数量级），且分布不完全规律时效果明显
- **缺点：**
  - 增加的判断开销
  - 增加的复杂度

编码思路转变

- WAH 编码时，以31bit分块chunk，使得游程编码RLE(Run Length Encoding)也以31bit为单位，引入了一个先验缺陷
- 为什么不可以先按bit为单位进行RLE编码先？
- 受PLWAH以及排序后位图实际排布启发，引入优化的第二条思路-MASC

新算法-MASC(MAximized Stride with Carrier)

- 改变编码方式，不再以chunk为单位，转而寻求最大编码长度，注重连续的0/1比特（与游程编码类似）
- 保留0-fill和1-fill概念，但是counter进行变动，能将非整数chunk的连续0/1也编码进来
- 对0-fill 增加carrier，最多可携带连续30个1.

新算法-MASC-原理介绍

0 0 000000 00000000 00000000 001 01110
1 0 000000 00000000 00000000 001 00101
0 1 00100 0 00000000 00000000 010 11001
0 0 000000 00000000 00000000 001 01110
0000000000 0000000000 0000000000 0
0000000000 0001111000 0000000000 0

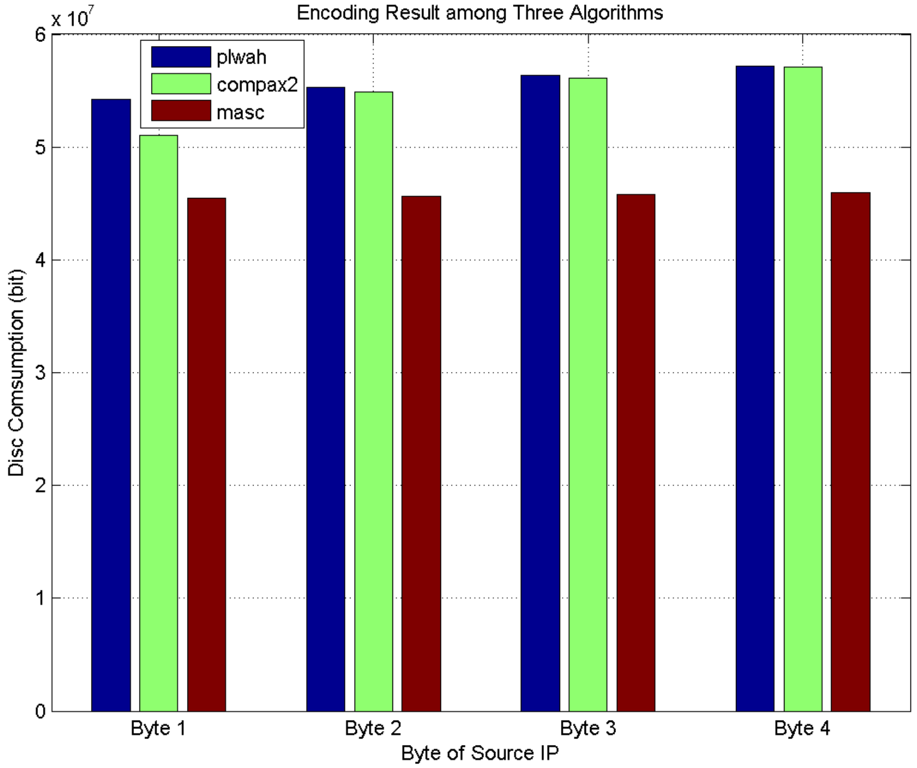
1 0 00000 0 00000000 00000000 00000001
0 0000000 00000011 11111111 11111111
0 1111111 11111111 11110000 00000000
1 0 00000 0 00000000 00000000 00000010
0 0000000 00000011 11000000 00000000
1 0 00000 0 00000000 00000000 00000001

e = 0, Chunk = 1, Additional = 0
pe = 1, Chunk = 0, Additional = 18
= 0, Chunk = 1, Additional = 13
pe = 0, Chunk = 0, Additional = 12
= 1, Chunk = 0, Additional = 37
pe = 0, Chunk = 0, Additional = 14
= 1, Chunk = 0, Additional = 4
Type = 0, Chunk = 1, Additional = 14
Type = 0, Chunk = 2, Additional = 12
Type = 0, Chunk = 2, Additional = 25

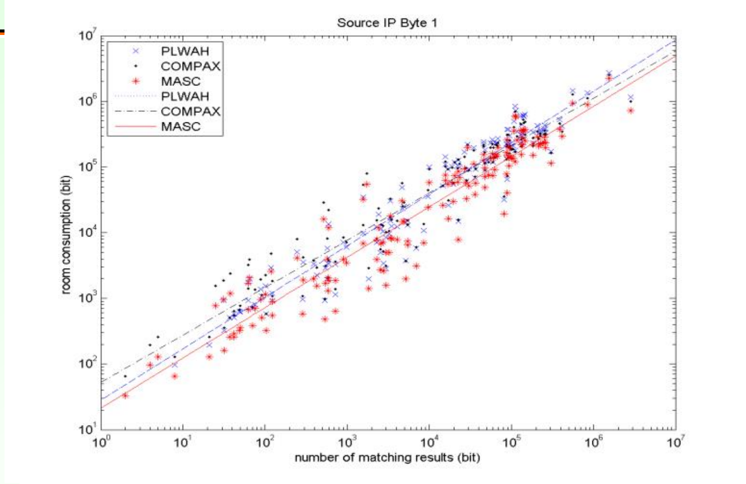
Type = 1, Chunk = 1, Additional = 6

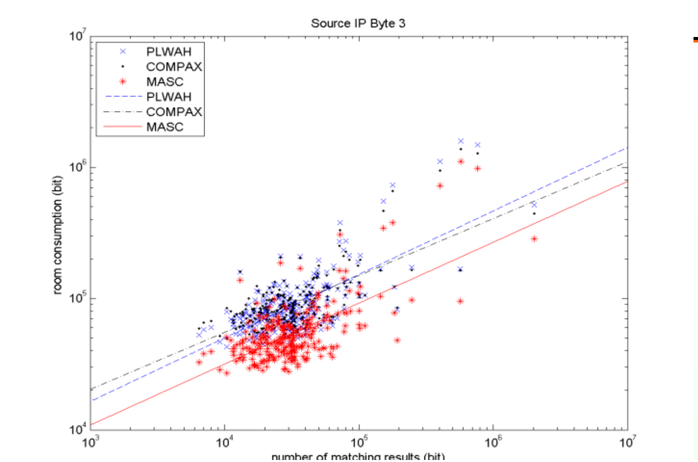
新算法-MASC-实验评估

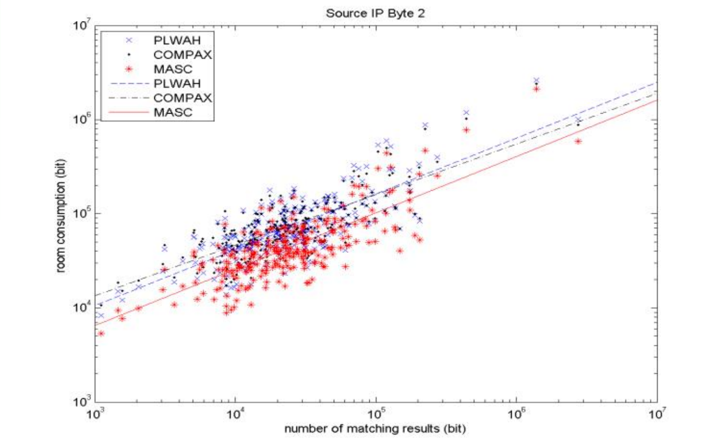
- 性能比较：
- 18.07%优于PLWAH
- 16.59%优于COMPAX2
- 数据集：CAIDA-2013

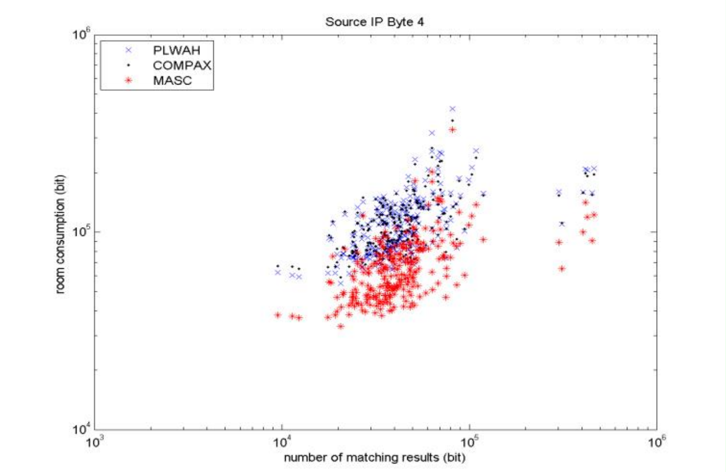


Source IP 4字节图

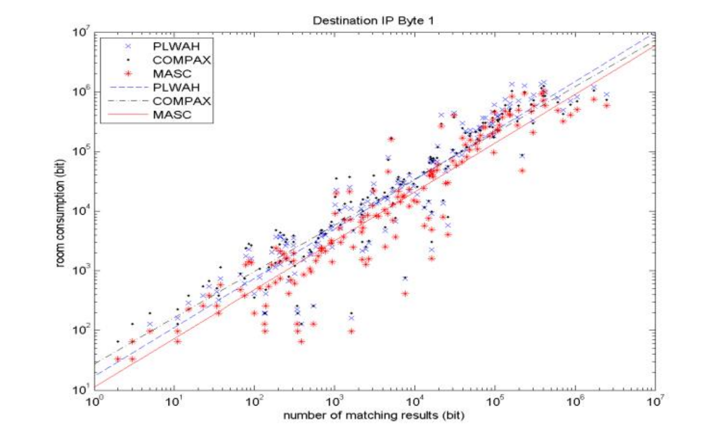


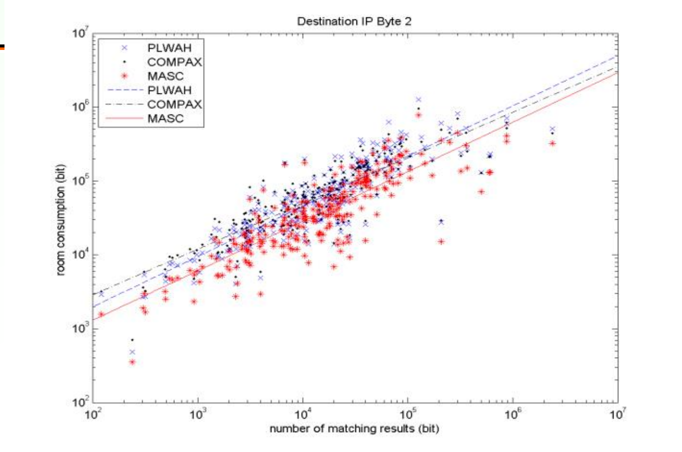


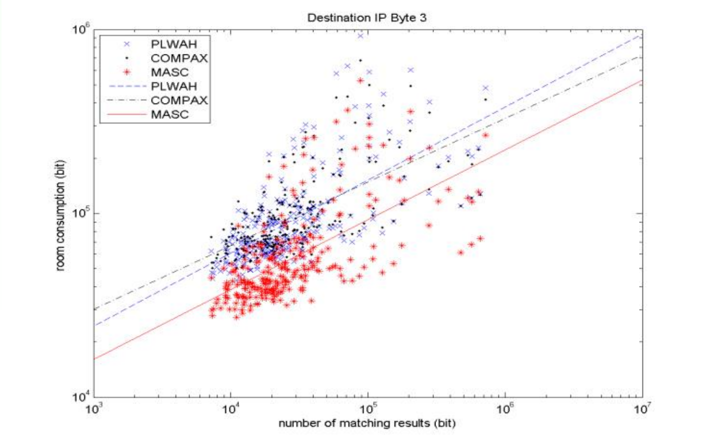


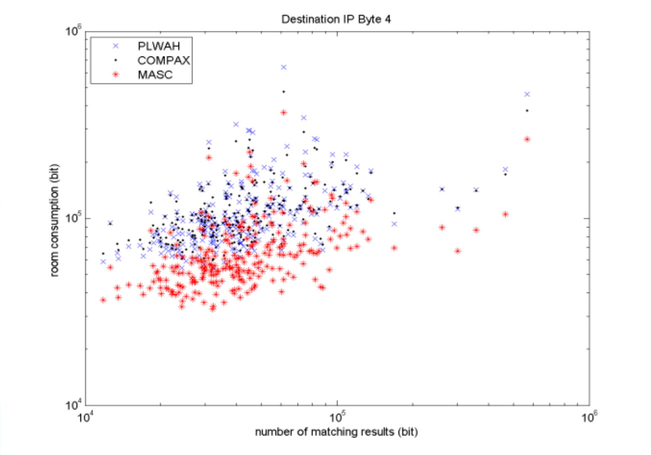


DstIP 4字节图

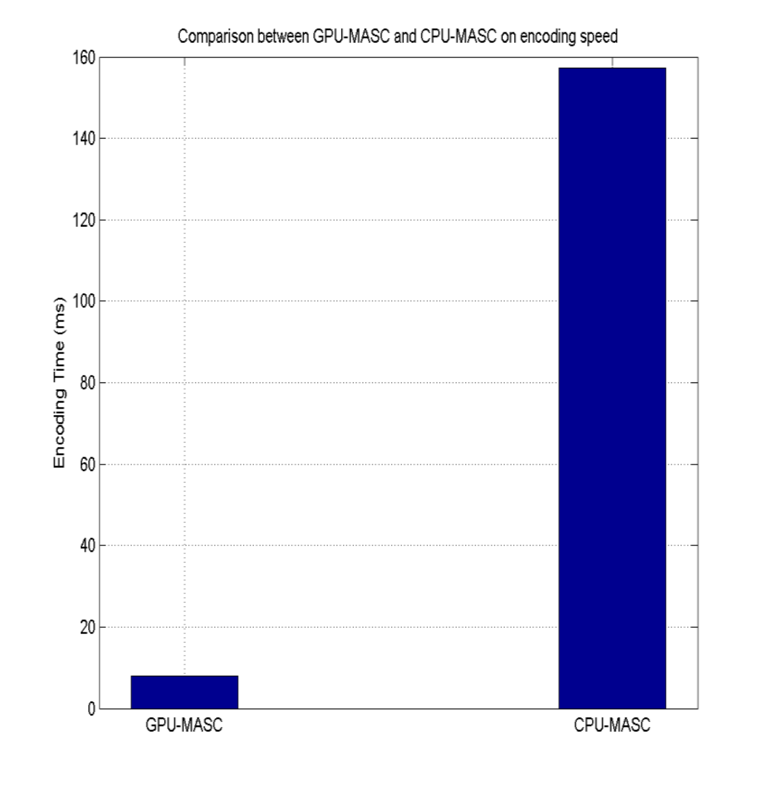

Destination IP Byte 1
PLWAH
COMPAX
MASC
PLWAH
COMPAX
MASC
room consumption (bit)
number of matching results (bit)

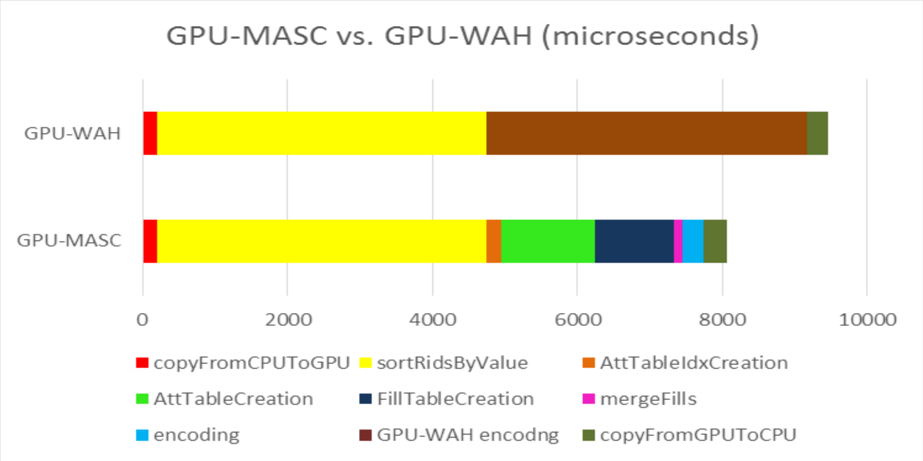

Destination IP Byte 2
PLWAH
COMPAX
MASC
PLWAH
COMPAX
MASC
room consumption (bit)
number of matching results (bit)


Destination IP Byte 3
PLWAH
COMPAX
MASC
PLWAH
COMPAX
MASC
room consumption (bit)
number of matching results (bit)


Destination IP Byte 4
PLWAH
COMPAX
MASC
room consumption (bit)
number of matching results (bit)

GPU-MASC 加速效果



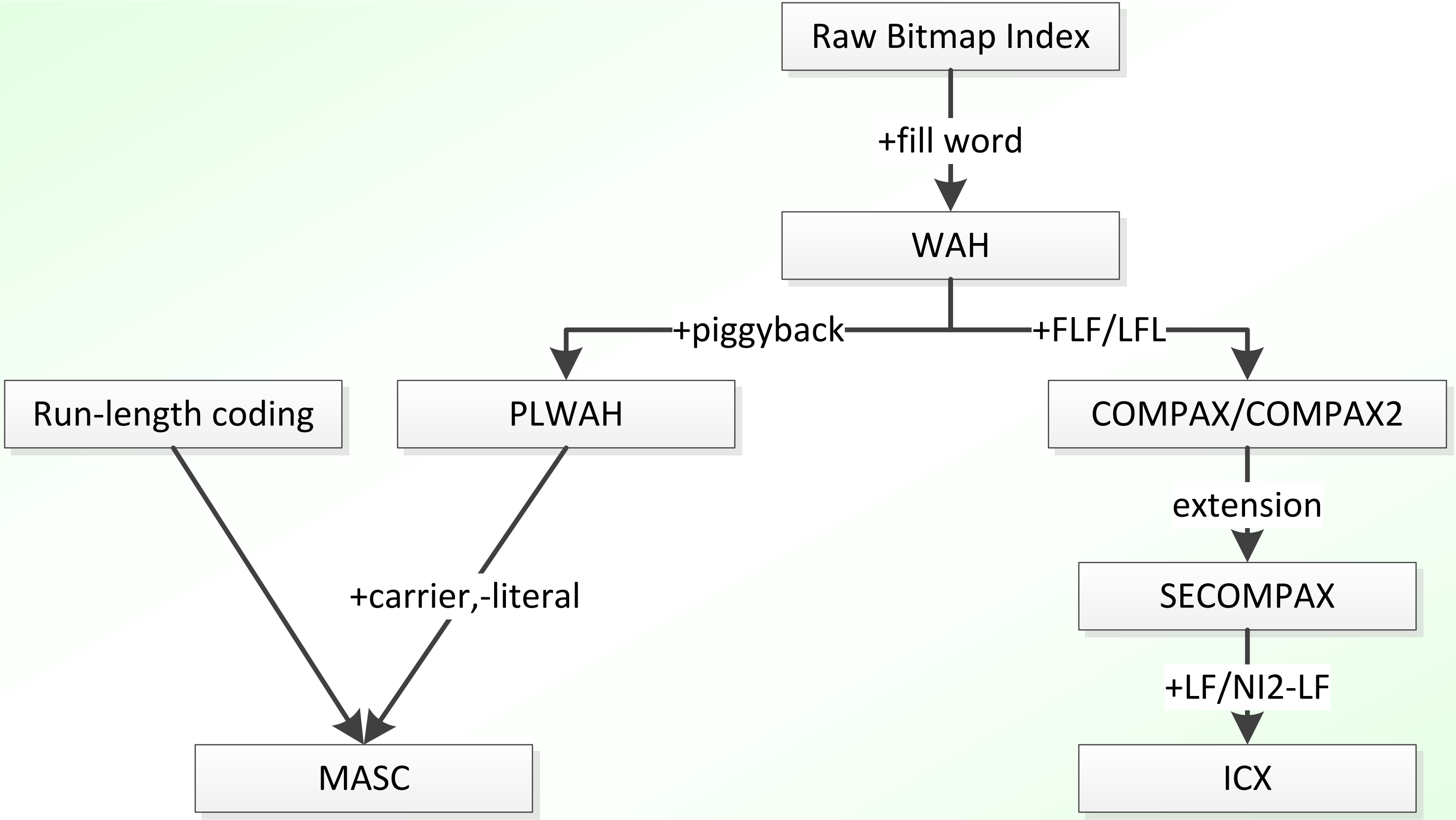


Input	**13,581,810 records**
GPU	GeForce GTX 760 (1152 cores)
CPU	Intel(R) Pentium(R) Dual CPU E2160 @ 1.80GHz
OS	Ubuntu 13.04 64 bit

新算法-MASC-创新点

- 优点：
  - 专注于对于连续0、1比特的压缩，优化并最终去掉了literal的概念，码本简洁且压缩效果提升明显
- 不足：
  - Counter大小是WAH的counter的31倍
  - 需要重排序reorder，预处理后产生bit clusters效应
- 改进：
  - 增加查询表以弥补查询速度可能的缺陷

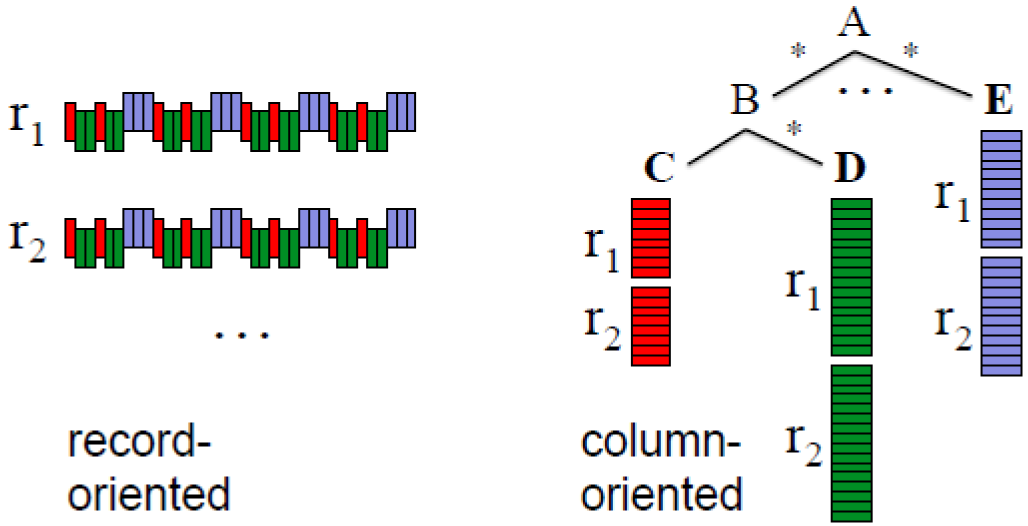
算法之间关联-roadmap



位图索引编码应用

- 生物信息序列比对
- 图可达性查询
- 网流检索取证

列存储数据-Columnar Storage

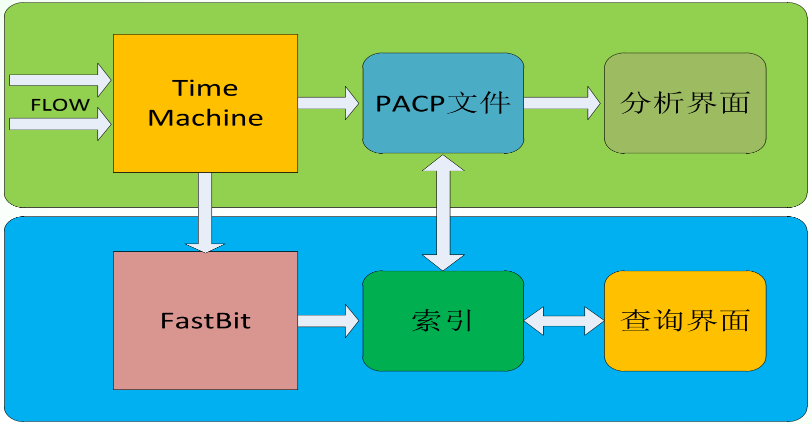


TIFAflow

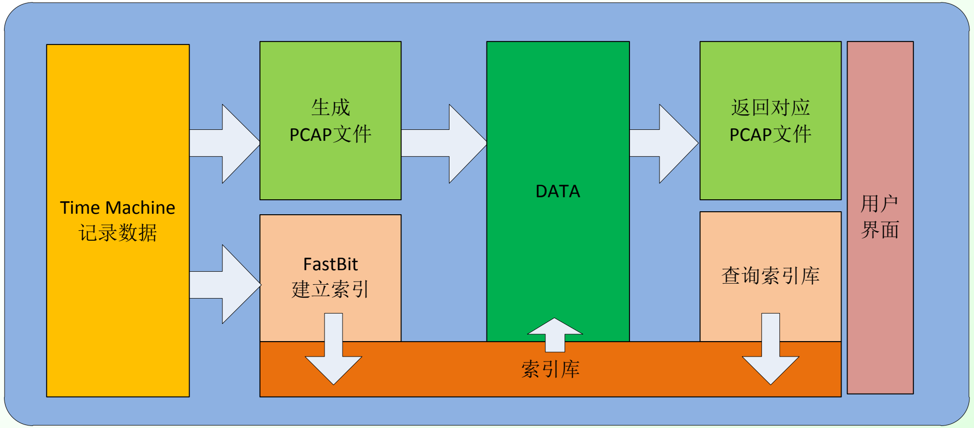
- 研究问题：网络安全事件难以追溯定位，如斯诺登披露的网络攻击事件
- 研究挑战：
  - 1)骨干链路速率高，流量大，存储速度慢
  - 2)索引空间消耗大，查询速度慢

J. Li et al., TIFA: Enabling Real-Time Querying and Storage of Massive Stream Data. Proc. of International Conference on Networking and Distributed Computing (ICNDC), 2011.

- 研究创新：1)基于流粒度的存储与查询；2)位图索引编码算法

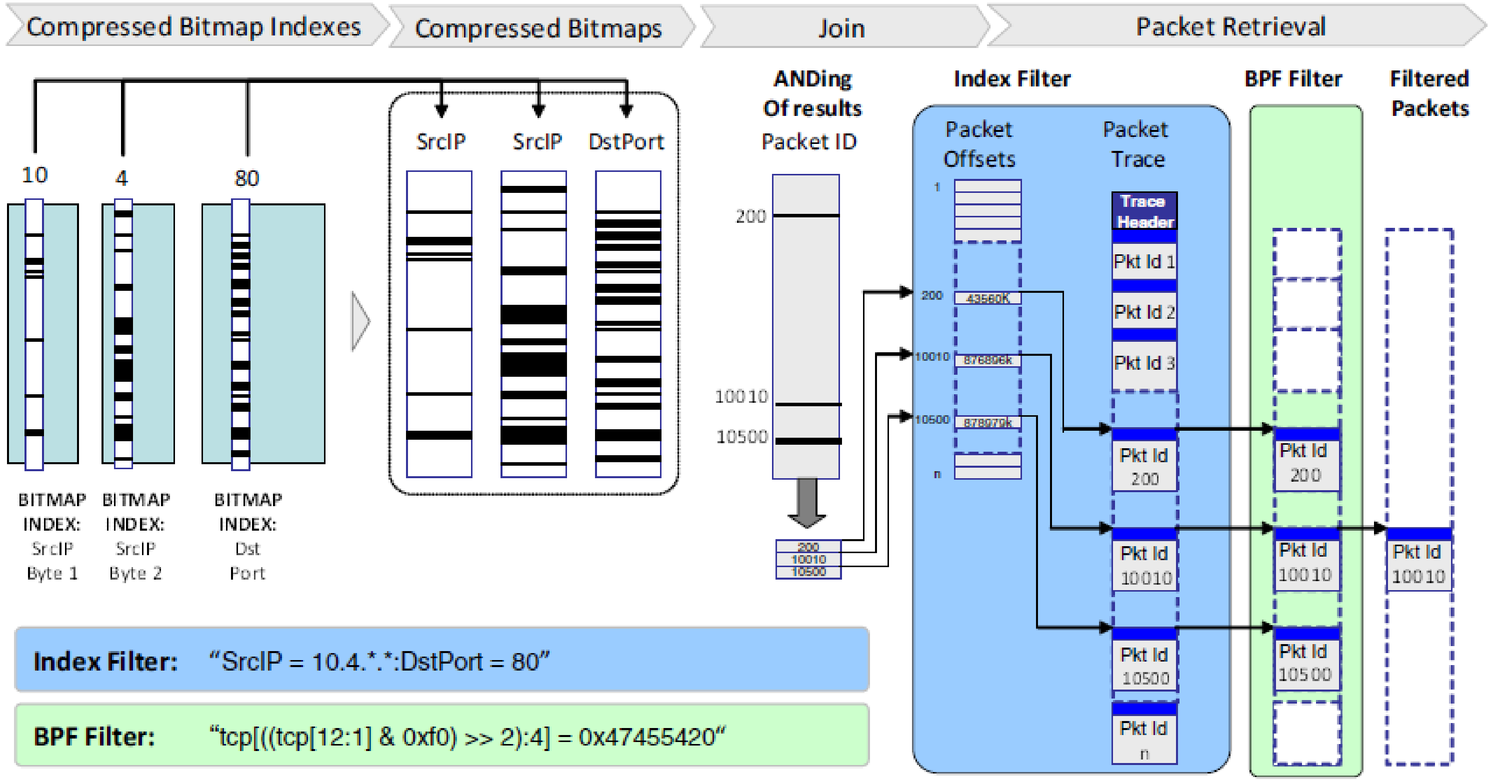


TIFA 系统结构



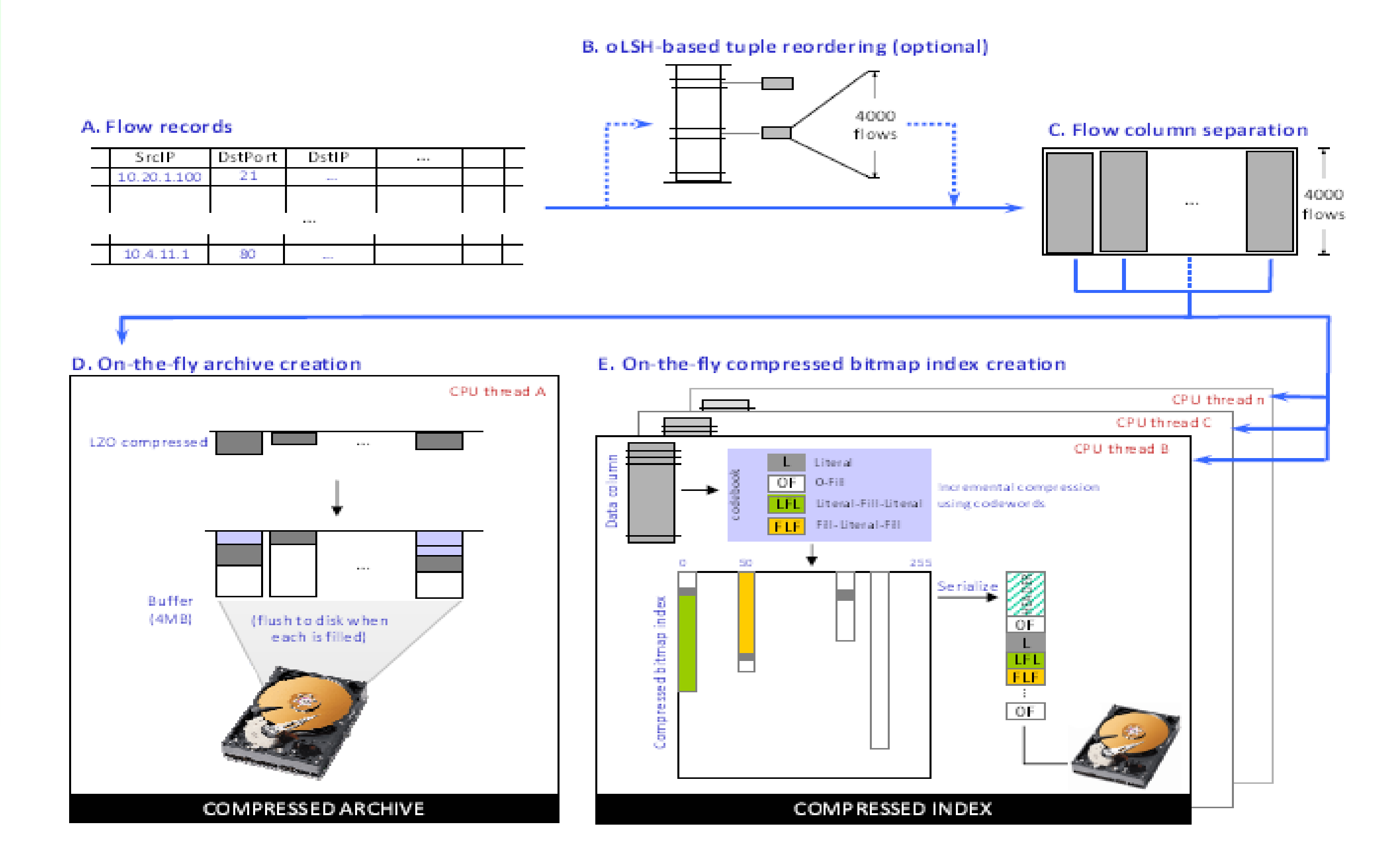
工作流程

PcapIndex



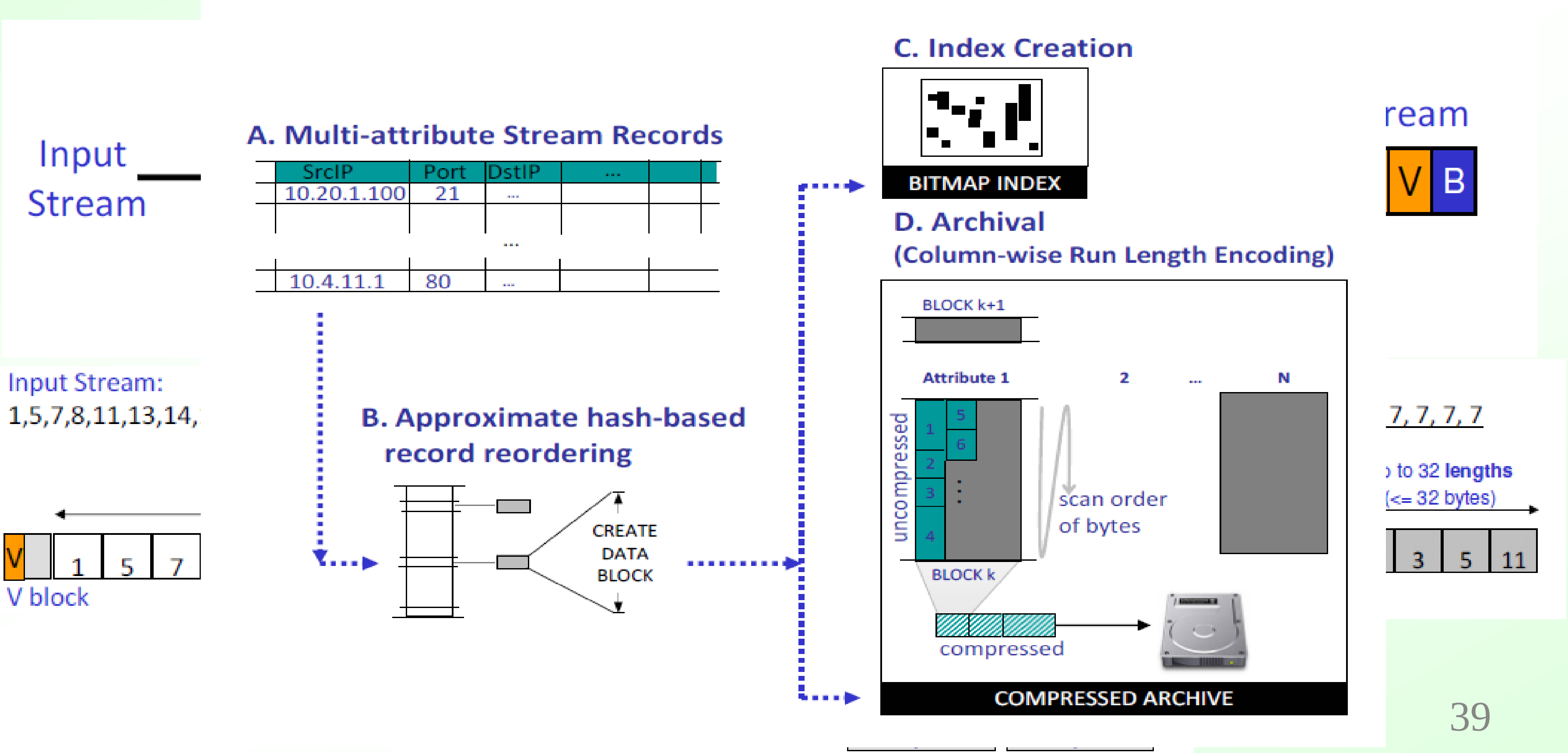
Compressed Bitmap Indexes
Compressed Bitmaps
Join
Packet Retrieval
10
4
80
BITMAP INDEX: SrcIP Byte 1
BITMAP INDEX: SrcIP Byte 2
BITMAP INDEX: Dst Port
SrcIP
SrcIP
DstPort
ANDing Of results
Packet ID
200
10010
10500
Index Filter
Packet Offsets
Packet Trace
Trace Header
Pkt Id 1
Pkt Id 2
Pkt Id 3
Pkt Id 200
Pkt Id 10010
Pkt Id 10500
Pkt Id n
BPF Filter
Filtered Packets
Index Filter: "SrcIP = 10.4.*.*:DstPort = 80"
BPF Filter: "tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x47455420"

NET-Fli



B. oLSH-based tuple reordering (optional)
4000 flows
A. Flow records
SrcIP
DstPort
DstIP
...
10.20.1.100
21
10.4.11.1
80
C. Flow column separation
4000 flows
D. On-the-fly archive creation
CPU thread A
LZO compressed
Buffer (4MB)
(flush to disk when each is filled)
COMPRESSED ARCHIVE
E. On-the-fly compressed bitmap index creation
CPU thread n
CPU thread C
CPU thread B
Data column
codebook
L Literal
OF 0-Fill
LFL Literal-Fill-Literal
FLF Fill-Literal-Fill
Incremental compression using codewords
0
50
255
Compressed bitmap index
Serialize
OF
L
LFL
FLF
OF
COMPRESSED INDEX

RasterZip



Input
Stream
A. Multi-attribute Stream Records
SrcIP
Port
DstIP
...
10.20.1.100
21
...
10.4.11.1
80
C. Index Creation
BITMAP INDEX
D. Archival
(Column-wise Run Length Encoding)
ream
V
B
Input Stream:
1,5,7,8,11,13,14,
V
1
5
7
V block
B. Approximate hash-based
record reordering
CREATE
DATA
BLOCK
BLOCK k+1
Attribute 1
2
...
N
uncompressed
1
5
6
2
3
4
scan order
of bytes
BLOCK k
compressed
COMPRESSED ARCHIVE
7, 7, 7, 7
to 32 lengths
(<= 32 bytes)
3
5
11

参考文献

1. [1] Wu, Ming-Chuan, Alejandro P. Buchmann, and P. Larson. Encoded Bitmap Indexes and Their Use for Data Warehouse Optimization. Shaker, 2001.
2. [2] Wu, Kesheng, Ekow J. Otoo, and Arie Shoshani. "Optimizing bitmap indices with efficient compression." ACM Transactions on Database Systems (TODS) 31, no. 1 (2006): 1-38.
3. [3] F. Deli`ege and T. B. Pedersen. Position list word aligned hybrid: optimizing space and performance for compressed bitmaps. Proc. of the 13th Int. Conf. on Extending Database Technology, EDBT ‘10, 2010.
4. [4] Fusco, Francesco, Michail Vlachos, and Marc Ph Stoecklin. "Real-time creation of bitmap indexes on streaming network data." The VLDB Journal-The International Journal on Very Large Data Bases 21, no. 3 (2012): 287-307.
5. [5] Fusco, Francesco, Michail Vlachos, and Xenofontas Dimitropoulos. "RasterZip: compressing network monitoring data with support for partial decompression." Proceedings of the 12th ACM SIGCOMM Conference on Internet measurement, IMC’12, 2012.
6. [6] Fusco, F., Dimitropoulos, X., Vlachos, M., & Deri, L. pcapIndex: an index for network packet traces with legacy compatibility. ACM SIGCOMM Computer Communication Review, 42(1), 47-53, 2012.
7. [7] Papadogiannakis, Antonis, Michalis Polychronakis, and Evangelos P. Markatos. "Scap: stream-oriented network traffic capture and analysis for high-speed networks." In Proceedings of the 2013 conference on Internet measurement conference, pp. 441-454. ACM, 2013.

Patents

- [1] BBC, Byte aligned data compression, DEC/Oracle, www.google.com/patents/US5363098.
- [2] WAH, Word aligned bitmap compression method, data structure, and apparatus, UC Berkeley, www.google.com/patents/US6831575.
- [3] COMPAX, Network analysis, IBM, www.google.com/patents/US20120054160.

研究小结

- 论文投稿
  - SECOMPAX: a bitmap index compression algorithm
  - ICX: a new bitmap index compression scheme
  - MASC: a bitmap index encoding algorithm for fast data retrieval
- 专利申请
  - 一种位图索引编码方法
  - 一种新的位图索引编码方法
  - 最大步进携带的位图索引编码方法

谢谢！