Research of a Scheduling Optimization algorithm based on Hash Adapter

Wenchuan Yang[1,a], Zhen Fu[2] and Wen Zuo[3]

[1]*Beijing University of Posts and Telecommunications;*
[2]*Beijing University of Posts and Telecommunications;*

Abstract. The performance of hash-based Scheduling optimization algorithm for network flow cannot be bounded under the worst-case because of hash conflicts. To make up for the shortage of the above algorithm, a multiple-hash algorithm based on counting Hash Adapter, which stores the digest of flow needed to be adjusted in the Exact Stream Hash Adapter(ESHA) structure, is designed. Compared with the basic hash table, the integrity of the conversation is maintained and the query performance is improved due to the lower probability of conflict. An example is given as a validation of this method and the result data reflect the practical effect.

1 Introduction

The current network security applications, such as intrusion detection system and the firewall technology needs to flow through each IP packet inspection.

With the rapid development of Internet, network bandwidth growth rate far exceeds the rate of growth of processors. According to the Gilder law, communication bandwidth doubles every 6 months, the growth rate of network bandwidth faster 3 times than the computer's performance. On the other hand, the processor will develop into multiple nuclear age calculation, which brings new opportunities to the IP packet processing.

This paper presents an efficient Do Hash selection algorithm. Scheduling optimization algorithm is mainly divided into the message level scheduling and stream scheduling. Message based Scheduling optimization algorithms, such as Round-Robin load scheduling algorithm with low cost, high efficiency characteristics, is not able to keep the stream (flow) IP packets in the sequence.

Flow based Scheduling optimization algorithm is mainly based on hash algorithm. Jaccob[1] proposed an adaptive scheduling optimization algorithm for session, and the work of this paper is based on 2 points: To redefine the concept of session; and to do a depth research on the performance of Hash and algorithm.

2 Optimization Algorithm

Herein we use some acronyms such as DA SA DP SP, which present the destination address, source address, destination port and source port.

Definition 1 a stream

A stream which has the same <DA, SA, DP, SP> packet is consists of a collection data.

Definition 2 total flow

<DA, SA, DP, SP> identifies semi flow and <SA, DA, SP, DP> identifies another semi flow.

Definition 3 session

<DA, SA, DP, SP> identifies the total flow and the dynamic analysis of the new four tuple <DA ', SA ', DP ', SP ' > identifies the total flow.

Hash based Scheduling optimization algorithm which belong to the same 1.5 stream of data packets are distributed to the same processing nuclear, This is not only to maintain the message sequence but also beneficial to use Cache. But the Hash algorithm can not adapt to the change in the height of network flow.

By researching on statistical stream length, flow characteristics of a heavy tail distribution, the Scheduling optimization algorithm based on the Hash need to adjust to different mapping flow core. In addition, the paper analyzed most of the multimedia protocol signaling and data transmission which used in a different port.

Traditional Scheduling optimization algorithms cannot guarantee the multimedia session to map to a core. Therefore, there is need to adjust the map location according to the flow of information dynamically. This paper achieve it by adjusting the total flow identifier and generating corresponding to the summary information on Digest and then saving it to the exact flow matching Hash Adapter (ESBF). For each incoming IP packet, we extracts <DA, SA, DP, SP> logo and the generation of Digest, and then for querying ESBF structure we can do it based on the <DA, SA, DP, SP> and Digest.

By saving the flow identification to the hash tables, it can dynamically specify a full flow to the corresponding processing nuclear.

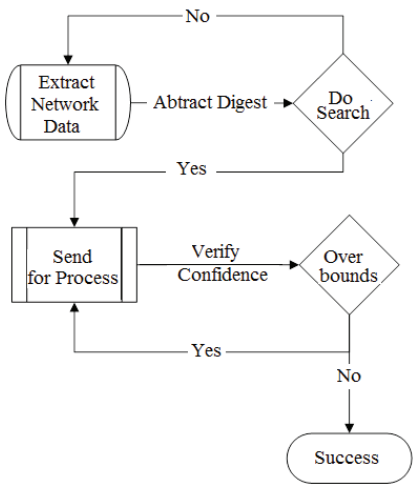

Fig 1. Processing Workflow

Back-end processing send nuclear add rule request (<DA, SA, DP, SP>, ID) according to the processed information, the <DA, SA, DP, SP> identification stream is sent to the designated processing nuclear ID.

3 Full Flow Digest Generation Algorithm

As shown in Figure 1, the majority of Miss rule packets directly mapped the Digest modulus to each core, Digest algorithm randomness and the performance of the Scheduling optimization algorithm has a strong effect.

Wilsley[3] proved that the bit between XOR and displacement operation can improve the hash values of the random character, and it has a very good performance, so this paper uses the XOR and shift to realize the generation of Digest.

Let bdip present DA 16 bit, adip present DP 16 bit, bsip present SA 16 bit, asip present 16 SP 16 bit.

Algorithm use OR (|), XOR (a), right (> >) and left (< <) operation. And hash1, hash2 use 16 bit unsigned integer. N for processing nuclear number.

Digest is a full flow <DA, SA, DP, SP> summary information. Algorithm DIGEST_HASH as following:

```
DIGEST_HASH(<DA , SA , DP , SP>){
    hash1 =asip<<3|asip>>(16-3);
    hash2 =bsip<<3|bsip>>(16-3);
    hash =hash1∧hash2∧dport;
    hash1 =adip<<3|adip>>(16-3);
    hash2 =bdip<<3|bdip>>(16-3);
    digest=hash1∧hash2∧sport;
    return digest;
}
```

Through the DIGEST_HASH algorithm, which belongs to the same total flow will generate the same Digest and map to the same core, thus it preserve the integrity of the conversation.

Table 1 to 4 based on Hash function operands are compared

Table 1. Hash function operands and its evaluative info

Network	Evaluative info	Hash optimzer factor
low	worst	(0,0,0,0.25)
low	worse	(0.2,0.3,0.3,0.4)
low	bad	(0.25,0.35,0.45,0.6)
medium	general	(0.3,0.5,0.5,0.7)
high	good	(0.5,0.65,0.75,0.8)
high	better	(0.7,0.8,0.8,1)
high	best	(0.7,1,1,1)

4 Exact Match of the Hash Adapter Algorithm

Adjusting mapping to different core needs to realize the precise flow matching. It can use basic Hash table to realize the precise flow matching, but due to a conflict exists, it cannot guarantee worst case performance.

Next serveral section are designed to support the Hash Adapter algorithm ESBF of accurate flow matching, it can satisfy the high-speed link to high speed flow classification requirements, and ensure the low collision probability.

5 Hash Adapter algorithm

First we introduced the Hash Adapter[4] (Hash Adapter, BF), which give a global U, BF a n bit vector B[i] (1<i<n) that set S, |S|=m, initialization time for all i, *B[i]=0(1<i<n)*.

The filter also needs to introduce k independent hash functions $h_1,h_2,\ldots\ h_k$,each function has the range for {1,2, ... , n}. hash function elements use the U as input, the elements in the U uniformly mapped to the range of function space.

For each element $h_x \in S$, set the $B[h_i(x)]=1$ (the same location may be repeatedly placed 1); thus, Hash Adapter can indicate the collection S.

Query for the elements of Y, if for any I (1<i<k), $h_i(y)=1$, $y \in S$, otherwise $y \notin S$.

From the above description it can be seen that for an element of Y, if $y \in S$, then return true conclusion.

But if $y \notin S$, it may also return a true conclusion, namely the false positive cases.

In the M members are Hash to the Hash Adapter, the one is for 0's probability.

$$p = (1-\frac{1}{n})^{km} \approx e^{-km/n} \tag{1}$$

False positive probability for

$$f = (1-(1-\frac{1}{n})^{km})^k \approx (1-e^{-km/n})^k = (1-p)^k \tag{2}$$

When m and n are a constant, we get the derivative for (2) as

$$f_{\min} = \left(\frac{1}{2}\right)^k \quad k = \lfloor \ln 2(n/m) \rfloor \tag{3}$$

Hash Adapter will get the lowest false positive rate at this time.

6 ESBF structure

Hash Adapter based on bit vector does not support elements removed, so the Increment Hash Adapter (Increment Hash Adapter, CBF) appeared later.

CBF standard BF digit group expansion for a small counter (Counter) each, if it inserts an element to the corresponding K (k is the Hash function a number) ,counter value plus 1, Counter value minus 1 if it deletes elements to the corresponding K.

CBF gives BF a delete operation through multiple occupancy of several times the storage cost. In this section the design of the ESBF algorithm is based on CBF, it use CBF multiple Hash functions to ensure the algorithm has lower conflict probability[5], its structure is shown in figure 2.

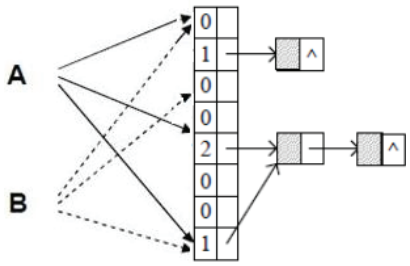

Fig 2. ESBF data structure

$h_i(x)$ is for k independent Hash function, the data structure in the CBF change into a chain list header node, the node is defined as follows:

```
Class Linknode{ //Linked list node
unsigned short digest; //Full flow Abstract
int id; // Specified number of processing cores
Linknode next;};
Class Linkhead { //Chain Header node
unsigned int counter; //counter
Linknode next;};
```

we take CBF as the vector length of 65536, and the CBF of each dimension is LinkHead structure. Input algorithm x=<DA,SA,DP,SP>.

The insertion algorithm code is as follows

Procedure Insertion Algorithm Insertelem(x, id)

{ Digest ← DIGEST_HASH(x)
Create Linked list node and assign digest & id
for i=L_{m-1} downto 1 do{
if(LHEAD(L_1,…,L_m).sup[i]<=sup(LHEAD(L_1,…,L_m))
and LHEAD(L_1,…,L_m).conf[i]>= min_α{
COUNTER[++length]=item(i);
output COUNTER and LHEAD(L_1,…,L_m).count[i]/n;
build LHEAD(L_1,…,L_m,i) on LHEAD(L_1,…,L_m);
If(there is an linked headin LHEAD(L_1,…,L_m,i))
add(LHEAD(L_1,…,L_m,i)); }
length--;}

Digest which was generated by Insertion algorithm were inserted into the K list. In order to save space, if the nodes are added to a linked list tail, K list can share a single linked list node.

7 ESBF Time Complexity Analysis

Because the algorithm memory access time is longer, so here it used the memory access times as its main operation to estimate the time complexity of algorithm.

The first analysis algorithm of query time complexity, for a total of MK into ESBF, average each counter value is mk/n, the query is always traverse the K (hash function number) counter in the smallest list corresponding to the mk/n item.

Therefore, the average time complexity is mk/n, and for optimization of the Hash Adapter $k = \lfloor \ln 2(n/m) \rfloor$, so the time complexity is O(k). Similarly, add and delete elements operate time complexity is O(k).

From the theoretical analysis of ESBF and basic hash table the average chain length is shown as follows in the same probability of cases.

For the basic hash table, length L is i probability when hit after needs to query the list

$$P(L=i \mid L>0)=\frac{P(L=0,L>0)}{P(L>0)}=\frac{\binom{m}{i}\left(\frac{1}{n}\right)^{i}\left(1-\frac{1}{n}\right)^{m-i}}{1-\left(1-\frac{1}{n}\right)^{m}} \quad (4)$$

For ESBF, with K Hash function, by the above analysis, the length L for i probability when hit after needs to query the is equivalent to finding the minimum value of K counters for the I probability $P(L_{smallest} = i)$.

$P(L= i)$ is a counter in a value for the i probability, $P(L>i)$ is K counter value which is greater than the probability of i. It can be:

$$P(L=i)=\binom{mk-k}{i-1}\left(\frac{1}{n}\right)^{i-1}\left(1-\frac{1}{n}\right)^{[(mk-k)-(i-1)]} \quad (5)$$

$$P(L>i)=1-\sum_{h=1}^{i}P(L=h) \quad (6)$$

$$P(L_{smallest}=i)=\sum_{j=1}^{k}\left[\binom{k}{j}\left(P(L=i)^{j}\times P(L>i)^{(k-j)}\right)\right] \quad (7)$$

Data in Table 2 and 3 show the result with or without Optimzer. In table 2 the n=65 536, m=5 000, k=5 calculated the 2 list length probability distribution.From table 3 we can find, ESBF chain lengths of 2 and 3 probability is much smaller than the underlying hash table hash table.

The basic chain length of 2 probability is 3.67%, ESBF is 0.32%, the latter as 1/11 less than; List length was 3 and the probability of the basic hash table is 0.09%, and ESBF 0.Thus, ESBF conflict probability is much lower than the underlying hash table, which ensure that the ESBF query performance.

Table 2. Probability distribution of chain length for Normal Hash

Length for	Probability				
	0	0	0	0	1
2	0	1	0	0	0
3	1	0	0	0	0
4	0	0	0	0	1

Table 3. Probability distribution of chain length for Hash with Optimizer

Length for	Probability				
	0.005	0.01	0.02	0.03	0.0
2	1	0	0	0	0
3	0	0	1	0	0

8 Experiments

Packet classification performance testing tools ClassBench simulation meet the heavy tail distribution flow accord with the actual traffic characteristics. Flow in the flow is assumed according to the back-end processing need or according to the systems of various nuclear load conditions need to be forwarded to the designated processing nucleus of the flow.

The flow of Digest and specify the processing of nuclear ID hash table and inserted into the basic structure of ESBF. on the basic of hash table and ESBF structure list length statistics, Table 4 in m=65 536, k=3 conditions,We get comparison table based on basic hash table and ESBF list length statistical. Among them, n is to simulate the resulting stream number, each of which is divided into 2 rows. The length of statistical data is shown as follows which was based on the behavior of basic hash table list and behavior under ESBF list length statistics

Table 4. Statistc for Chain length for various data

Chain	Number for Data						
	1000	2000	3000	4000	5000	6000	7000
1	966	1920	2855	3702	4621	5531	6134
2	15	32	55	121	231	311	427
3	0	0	3	5	7	18	24
4	0	0	0	0	3	0	3

The evaluation for complaint variables must be compatible, and it should fit for both quantitative and qualitative information. So the quantitative information must be properly transformed into a dimensionless value before expressed as the fuzzy numbers.

Because of experimental data is used to meet the heavy tail distribution of flow, which exist in the same Baotou, so the table in the same row list length statistics may be smaller than the test data set n. From the watch 2 in can see, 1 for n 000~4 000, ESBF no length is 3, and the ratio of length 2 is far less than the underlying hash table. When n is 5000, ESBF is 3 of the length ratio of only basic hash table 1/2. In the case of small n, ESBF list length was significantly shortened basic hash, and worst case time complexity is smaller than the underlying hash table.

As n is greater than 4000, although the two worst case time complexity equivalent, but ESBF algorithm meet the worst case that probabilities is below the basic hash table. So ESBF algorithm performance is better than the basic query hash table.

But it should be noted that, because ESBF uses multiple Hash functions, it will be the same as a key value is mapped to the K position, which makes the ESBF the load factor become the basic Hash table K times, on the other hand it led to conflict probability.

This section can be obtained by experiments, the load factor is lower, ESBF effectively specify the full stream which sent to the corresponding processing nuclear according to the system processing nuclear load condition and requirement of application. It meets the growing IP packet processing applications.

Compared to the underlying hash table, the algorithm for the conflict has better probability constraints, the Scheduling optimization algorithm has higher performance. Because of its simple and efficient, how to reduce the conflict has yet to be further studied and solved.

9 Conclusions

According to the characteristics of network streaming media application protocol and network security application needs, this paper put forward load balance based on CBF algorithm, this algorithm is mainly composed of a full flow abstract algorithm and ESBF algorithm. It Focuses on the research of Scheduling optimization algorithm based on Hash performance problems, and committe to reducing what would function conflict brought negative effect for the entire algorithm performance, and from the theoretical and experimental results it verify the algorithm's advantages.

References

1. G. Jaccob, A. C. Kay, S. J. Spencer, Loving those who justify inequality: The effects of system threat on attraction to women who embody benevolent sexist ideals. Psychological Science, 19, pp.**20-21**, (2008).
2. S. Levin, Intergroup biases as a function of social dominance orientation, ingroup status, and ingroup identification. Unpublished master's thesis, University of California, Los Angeles,pp.**121-127**,(1992).
3. S. Wilsley, C. M. Federico, J. Sidanius,J. L. Rabinowitz, Social dominance orientation and intergroup bias: The legitimation of favoritism for high-status groups. Personality and Social Psychology bulletin, 28,pp.**144-157**, (2002).
4. C. Oishi, F. Yoshida, Black sheep effect and ingroup favoritism in social identity perspective.Shinrigaku kenkyu, 73, pp.**405-411**, (2002).
5. J. L. Rabinowitz, Go with the flow or fight the power The interactive effects of social dominance orientation and perceived injustice on support for the status quo. Political Psychology, 20, pp.**1-24**,(1999).
6. K. J. Reynolds, J. C. Turner, S. A. Haslam, M. K. Ryan, B. Bizumic, E. Subasic, Does personality explain in-group identification and discrimination Evidence from the minimal group paradigm. British Journal of Social Psychology, 46, pp.**517-539**, (2007).