Nginx Hash Tables

Table of contents :

1. Introduction to Hash Tables in nginx
2. Hash Table structure
3. Hash Functions
4. Hash Lookup

Introduction :

The concept of Hash Table is implemented in nginx using these two files :

* ngx\_hash.c -> Present in src/core/ directory
* ngx\_hash.h -> Present in src/core/ directory

A hashtable is a data structure used to implement an associative array, a structure that can map keys to values. A hash table uses a hash function to compute an index into an array of buckets or slots, from which the correct value can be found.

The hash table consists of key value pairs <key,value>.There exists a hash function which takes the key as input and produces a hash value which is then mapped or indexed to one of the bucket.

A block in a hash table is called a bucket.

Hash Table Structure :

The Hash Table structure is defined in the code as :

typedef struct {

ngx\_hash\_elt\_t \*\* buckets;

ngx\_uint\_t size;

} ngx\_hash\_t;

Here

buckets : Each block of a hash table is called a bucket.The buckets is a pointer to first bucket or block of the hash table.

size : Specifies the number of slots or buckets in the hash table.

The hash table element structure is defined as :

typedef struct {

void \* value;

u\_short len ;

the the u\_char name [1];

} ngx\_hash\_elt\_t;

value : the value is a void pointer and points to the data to be used/hashed

len : length represents the length of the data

name[1] : the address of the key element to be used for hashing

In addition to this the hash table in nginx also supports wildcard searching. It can search for strings like abc\* or \*abc or abc.

The hash table initialization structure is defined as :

typedef struct {

ngx\_hash\_t \*hash;

ngx\_hash\_key\_pt key;

ngx\_uint\_t max\_size;

ngx\_uint\_t bucket\_size;

char \*name;

ngx\_pool\_t \*pool;

ngx\_pool\_t \*temp\_pool;

} ngx\_hash\_init\_t;

Here ,

hash : pointer to the hash table defined by structure ngx\_hash\_t

key : The key is the element for which the hash value is calculated by using hash function

bucket\_size : The maximum size of the bucket .

max\_size : The maximum no of buckets to be used. .the larger the number of the buckets the lesser will be the collisions.

name : A name is given to the hash structure.This name is only used in error logs.

pool : the pointer to the pool which is to used for memory allocation.

temp\_pool : The pointer to the temporary data space if needed

Here we see that a key is taken as input by the hash function and the corresponding hash value is generated.

This is represented as <key,value> pair.

The structure of the key is defined as :

typedef struct {

ngx\_str\_t key;

ngx\_uint\_t key\_hash;

void \*value;

} ngx\_hash\_key\_t;

Here ,

key : A key is nothing but a typical nginx string of type ngx\_str\_t.This is the input to the hash function.

key\_hash : A string which is the hashed value of the input key.A hash function is responsible for converting key to

hash.

value : After calculating the hash of the key it is put into a particular index in the hash table .This index is called

the value.

Hash Functions :

All the hash function which are used to calculate the hash value for the given key call the ngx\_hash macro. This macro returns a long integer.

# define ngx\_hash (key, c) ((ngx\_uint\_t) key \* 31 + c) / / hash macro

This macro is simple to understand.It just returns a calculated long integer.

This various hash functions are :

1. ngx\_uint\_t ngx\_hash\_key (u\_char \* data, size\_t len);

2. ngx\_uint\_t ngx\_hash\_key\_lc (u\_char \* data, size\_t len);

3. ngx\_uint\_t ngx\_hash\_strlow (u\_char \* dst, u\_char \* src, size\_t n);

Definition of ngx\_hash\_key(u\_char \* data, size\_t len);

This function takes the data and its length as parameter and calculates its hash value.

ngx\_uint\_t

ngx\_hash\_key(u\_char \*data, size\_t len)

{

ngx\_uint\_t i, key;

key = 0;

for (i = 0; i < len; i++) {

key = ngx\_hash(key, data[i]);

}

return key;

}

The function calculation ngx\_hash\_key can be expressed as the following equation.

1. Key [0] = data [0]
2. Key [1] = data [0] \* 31 + data [1]
3. Key [2] = (data [0] \* 31 + data [1]) \* 31 + data [2]
4. ...
5. Key [len-1] = ((((data [0] \* 31 + data [1]) \* 31 + data [2]) \* 31) ... data [len-2]) \* 31 + data [len -1]

key [len-1] is the the incoming parameter data corresponding hash value.

Hash Lookup :

One of the main use of the hash table is efficient searching.The hash lookup operation is implemented by the ngx\_hash\_find () function.

1. / / By the key name, len information in the hash table hash point for the key corresponding to the value
2. **void** \*
3. ngx\_hash\_find (ngx\_hash\_t \* hash, ngx\_uint\_t key, u\_char \* name, **size\_t** len)
4. {
5. ngx\_uint\_t i;
6. ngx\_hash\_elt\_t \* ELT;
8. elt = hash-> buckets [key% hash-> size]; / / find where key stored in the bucket (the bucket elts address)
10. **if** (elt == NULL) {
11. **return** NULL;
12. }
14. **while** (elt-> value) {
15. **if** (len = **(size\_t)** the elt-len) {/ / first determine the length
16. **goto** next;
17. }
19. **for** (i = 0; i <len; i + +) {
20. **if** (name [i]! = elt-> name [i]) {/ / Next name (here the characters match)
21. **goto** next;
22. }
23. }
25. **return** elt-> value; / / match is successful, the direct to return the ngx\_hash\_elt\_t structure value field
27. next:
28. / / Note from elt-> name [0] address at the back offset, the offset just add the elt len can, and then 4-byte aligned
29. elt = (ngx\_hash\_elt\_t \*) ngx\_align\_ptr (& elt-> name [0] + elt-> len,
30. **sizeof (void** \*));
31. **continue;**
32. }
34. **return** NULL;
35. }

Lookup operation is quite simple, calculated directly by the key element value.The bucket contains the starting address of the of the ngx\_hash\_elt\_t data area. The string to be found is judged according to the length and name content, if the match is successful, then the value field corresponding to it is returned.