Meaning of max size of Nginx "large_client_header_buffers" directive

Asked 6 years, 1 month ago Modified 6 months ago Viewed 39k times

32

Nginx docs.

```
Syntax: large_client_header_buffers number size;
Default: large_client_header_buffers 4 8k;
Context: http, server

Sets the maximum number and size of buffers used for reading large client request
header.
```

I understand what the buffers size is but I do not understand what the buffers number is.

How does the processing change depending on the number of buffers?

nginx nginx-config

Share Improve this question Follow

asked Mar 1, 2019 at 2:42

t_uma66

321 ● 1 ● 3 ● 4

2 Answers

Sorted by: Highest score (default)

60

So I have been fighting with some HTTP header length most of the night and have had to figure this out.

The TL;DR buffer size is how big your buffer is, the buffer number is how many you have. So your total capacity is num_buffs*buff_size + 1kb regular header buffer = Total Capacity, with the caveat that a header will only go into a buffer if there is enough space in the buffer, or in other words a header will not get split amongst buffers.

For source I have been working on figuring out how buffering works for the past several hours by making numerous curl requests with headers of varying sizes.

The detailed explanation. Within Nginx there is the default header buffer that is configured with the `client_header_buffer_size` directive. When a request comes in the headers are first read into this buffer, the `large_client_header_buffers` do not get engaged as long as the sum total

size of the request headers does not exceed the value configured for `client_header_buffer_size`, by default 1kb.

However once we breach that limit then things get interesting. As Nginx reads the headers into the buffer it will continue to read them into the `client_header_buffer` until a header arrives that is larger than the space left in the buffer, at which point the `large_client_header_buffers` become active and the whole header will then be read into the first `large_client_buffer`. Nginx will then continue reading headers into the `client_header_buffer` until it hits another header that cannot fit in the space remaining in the `client_header_buffer` at which point it will then check if it can put the request header in the first `large_client_buffer`. If it can't it will then check if it can place the header in the second `large_client_buffer` this process will occur on each buffer until one of two conditions are met:

1. All headers are successfully processed and read into buffers

or

2. There is not enough space left in any of the buffers to read in the remaining header(s), either because there are no more buffers that have enough empty space or because the request header size exceeds the size configured for the buffers.

When condition number 2 occurs Nginx will then respond with an error indicating the request is too large.

Let's go through some examples to make this concrete.

For our examples we will assume we have configured the `client_header_buffer`, referred to as CHB, to a size of 10kb and we have configured two `large_client_header_buffers` that have a size of 20kb each, referred to as LCHB1 and LCHB2 respectively.

Scenario 1 Vanilla:

curl https://example.com -H 'h1: 3kb-long' -H 'h2: 2kb-long'

h2 | |

h1 | |

CHB | LCHB1 | LCHB2

In this situation our headers only total 5kb together and so easily fit into the primary buffer, and we could support multiple more headers in the primary buffer so long as none of them exceeded 5kb in size, either individually or collectively.

Scenario 2 A header larger than the CHB buffer :

curl https://example.com -H 'h1: 14kb-long'

Empty | h1 |

CHB | LCHB1 | LCHB2

In this instance the header gets read straight into the large buffer because there is no room for it in the primary buffer due to the single header exceeding the size configured for the primary buffer.

Scenario 3 All the buffers used:

curl https://example.com -H 'h1: 19kb', -H 'h2: 19kb' -H 'h3: 9kb'

h3 | h1 | h2

CHB| LCHB1 | LCHB2

In this case we receive a header that cannot go into the primary buffer but just barely fits into one of the large buffers and so the first header goes there. Then the next header comes in and can also not go in the primary buffer, but there is a slot for it in the second large buffer so it goes there. Then the final header can fit within the confines of the primary buffer

Scenario 3 Too Many Headers:

curl https://example.com -H 'h1: 19kb', -H 'h2: 19kb' -H 'h3: 9kb' -H 'h4: 2kb'

h4 | h3 | h1 | h2

Error | CHB | LCHB1 | LCHB2

In this case the scenario starts to play out similar to Scenario 3; however we hit an issue when we introduce an additional 2kb header. So with 19kb of 20kb used in each of the large buffers and 1kb remaining in the primary buffer we have 3kb of buffer space left and so we should be able to handle the final 2kb header right? Wrong, my friend. The problem is that when the 2kb header arrives Nginx looks in the primary buffer and sees that there is only 1kb of space left there so the header can't go there, it then checks the first large buffer, but still only 1kb of space so it can't go there, finally it checks the final large buffer only to find it still only has 1kb of space. At this point Nginx returns an error indicating that it received a bad request as it has no where to read the header into.

So in summary a buffer size is how large of a buffer you have but the number of buffers is a multiplier of that number, that is how many different buffers you have to hold the request headers.

Share Improve this answer Follow

answered Oct 30, 2019 at 12:01

hpoe

852 ● 6 ● 11

1 Thank you for the research and detail!!!!!!!! After experimentation I was beginning to expect a behavior such as this but the nginx doc didn't seem to tell the full story. – J. Campbell Mar 17, 2022 at 21:45

@hpoe Any idea of the performance implications when headers don't fit into the smaller buffers? What guides one on the tradeoffs of making the primary buffers larger or smaller? If most requests don't fit in the primary buffer what is the consequence? – Joshua Enfield May 17, 2022 at 20:51

8

@hpoe's answer is great, but wanted to add some discrepancies with how I see this works now. Apparently, when nginx sees that a header does not fit into the current buffer, it allocates a large buffer for it, and then this buffer is made the current buffer.

See here and there in nginx code - when large buffer is allocated it's assigned to `r->header_in` which is then used to read in the next header.

This means that success of request parsing depends on the order in which the headers arrive. For example, if you have `large_client_header_buffers 4 4k` in the configuration, if you get:

```
<2.5k header>
<2.5k header>
<2.5k header>
<4k header>
<100b header>
```

nginx will allocate large buffers for the first 4 headers (because they would not fit into the primary 1k buffer, and also can't combine together to fit into a large 4k buffer) and then try to fit the last header into the last large buffer (which is full already), and fail. But if the `100b header` came first then there would be no problem - it would have been read into the primary (1k) buffer.

Share Improve this answer Follow

answered Nov 30, 2022 at 7:26

astgtciv

770 ● 6 ● 15

Start asking to get answers

Find the answer to your question by asking.

Ask question

Explore related questions

nginx nginx-config

See similar questions with these tags.