**支持keep alive长连接**

当使用nginx作为反向代理时，为了支持长连接，需要做到两点：

1. 从client到nginx的连接是长连接
2. 从nginx到server的连接是长连接

从HTTP协议的角度看，nginx在这个过程中，对于客户端它扮演着HTTP服务器端的角色。而对于真正的服务器端（在nginx的术语中称为upstream）nginx又扮演着HTTP客户端的角色。

**保持和client的长连接**

为了在client和nginx之间保持上连接，有两个要求：

1. client发送的HTTP请求要求keep alive
2. nginx设置上支持keep alive

* [HTTP配置](https://skyao.gitbooks.io/leaning-nginx/content/documentation/keep_alive.html#http配置)
  + [keepalive\_timeout指令](https://skyao.gitbooks.io/leaning-nginx/content/documentation/keep_alive.html#keepalivetimeout指令)
  + [keepalive\_requests指令](https://skyao.gitbooks.io/leaning-nginx/content/documentation/keep_alive.html#keepaliverequests指令)
  + [保持和server的长连接](https://skyao.gitbooks.io/leaning-nginx/content/documentation/keep_alive.html#保持和server的长连接)
* [upstream设置](https://skyao.gitbooks.io/leaning-nginx/content/documentation/keep_alive.html#upstream设置)
  + [keepalive参数的理解](https://skyao.gitbooks.io/leaning-nginx/content/documentation/keep_alive.html#keepalive参数的理解)
  + [总结](https://skyao.gitbooks.io/leaning-nginx/content/documentation/keep_alive.html#总结)
* [location设置](https://skyao.gitbooks.io/leaning-nginx/content/documentation/keep_alive.html#location设置)

**HTTP配置**

默认情况下，nginx已经自动开启了对client连接的keep alive支持。一般场景可以直接使用，但是对于一些比较特殊的场景，还是有必要调整个别参数。

需要修改nginx的配置文件(在nginx安装目录下的conf/nginx.conf):

http {

keepalive\_timeout 120s 120s;

keepalive\_requests 10000;

}

**keepalive\_timeout指令**

keepalive\_timeout指令的语法：

Syntax: keepalive\_timeout timeout [header\_timeout];

Default: keepalive\_timeout 75s;

Context: http, server, location

第一个参数设置keep-alive客户端连接在服务器端保持开启的超时值。值为0会禁用keep-alive客户端连接。可选的第二个参数在响应的header域中设置一个值“Keep-Alive: timeout=time”。这两个参数可以不一样。

注：默认75s一般情况下也够用，对于一些请求比较大的内部服务器通讯的场景，适当加大为120s或者300s。第二个参数通常可以不用设置。

**keepalive\_requests指令**

keepalive\_requests指令用于设置一个keep-alive连接上可以服务的请求的最大数量。当最大请求数量达到时，连接被关闭。默认是100。

这个参数的真实含义，是指一个keep alive建立之后，nginx就会为这个连接设置一个计数器，记录这个keep alive的长连接上已经接收并处理的客户端请求的数量。如果达到这个参数设置的最大值时，则nginx会强行关闭这个长连接，逼迫客户端不得不重新建立新的长连接。

这个参数往往被大多数人忽略，因为大多数情况下当QPS(每秒请求数)不是很高时，默认值100凑合够用。但是，对于一些QPS比较高（比如超过10000QPS，甚至达到30000,50000甚至更高) 的场景，默认的100就显得太低。

简单计算一下，QPS=10000时，客户端每秒发送10000个请求(通常建立有多个长连接)，每个连接只能最多跑100次请求，意味着平均每秒钟就会有100个长连接因此被nginx关闭。同样意味着为了保持QPS，客户端不得不每秒中重新新建100个连接。因此，如果用netstat命令看客户端机器，就会发现有大量的TIME\_WAIT的socket连接(即使此时keep alive已经在client和nginx之间生效)。

因此对于QPS较高的场景，非常有必要加大这个参数，以避免出现大量连接被生成再抛弃的情况，减少TIME\_WAIT。

**保持和server的长连接**

为了让nginx和server（nginx称为upstream）之间保持长连接，典型设置如下：

http {

upstream BACKEND {

server 192.168.0.1：8080 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.0.2：8080 weight=1 max\_fails=2 fail\_timeout=30s;

keepalive 300; // 这个很重要！

}

server {

listen 8080 default\_server;

server\_name "";

location / {

proxy\_pass http://BACKEND;

proxy\_set\_header Host $Host;

proxy\_set\_header x-forwarded-for $remote\_addr;

proxy\_set\_header X-Real-IP $remote\_addr;

add\_header Cache-Control no-store;

add\_header Pragma no-cache;

proxy\_http\_version 1.1; // 这两个最好也设置

proxy\_set\_header Connection "";

client\_max\_body\_size 3072k;

client\_body\_buffer\_size 128k;

}

}

}

**upstream设置**

upstream设置中，有个参数要特别的小心，就是这个keepalive。

大多数未仔细研读过nginx的同学通常都会**误解**这个参数，有些人理解为这里的keepalive是设置是否打开长连接，以为应该设置为on/off。有些人会被前面的keepalive\_timeout误导，以为这里也是设置keepalive的timeout。

但是实际上这个keepalive参数的含义非常的奇特，请小心看[nginx文档](http://nginx.org/en/docs/http/ngx_http_upstream_module.html#keepalive)中的说明:

Syntax: keepalive connections;

Default: —

Context: upstream

Activates the cache for connections to upstream servers. 激活到upstream服务器的连接缓存。

The connections parameter sets the maximum number of idle keepalive connections to upstream servers that are preserved in the cache of each worker process. When this number is exceeded, the least recently used connections are closed. connections参数设置每个worker进程在缓冲中保持的到upstream服务器的空闲keepalive连接的最大数量.当这个数量被突破时，最近使用最少的连接将被关闭。

It should be particularly noted that the keepalive directive does not limit the total number of connections to upstream servers that an nginx worker process can open. The connections parameter should be set to a number small enough to let upstream servers process new incoming connections as well. 特别提醒：keepalive指令不会限制一个nginx worker进程到upstream服务器连接的总数量。connections参数应该设置为一个足够小的数字来让upstream服务器来处理新进来的连接。

在这里可以看到，前面的几种猜测可以确认是错误的了：

1. keepalive不是on/off之类的开关
2. keepalive不是timeout，不是用来设置超时值

很多人读到这里的文档之后，会产生另外一个误解：认为这个参数是设置到upstream服务器的长连接的数量，分歧在于是最大连接数还是最小连接数，不得不说这也是一个挺逗的分歧......

回到nginx的文档，请特别注意这句话，至关重要：

The connections parameter sets the maximum number of idle keepalive connections to upstream servers connections参数设置到upstream服务器的空闲keepalive连接的最大数量

请仔细体会这个"idle"的概念，何为idle。大多数人之所以误解为是到upstream服务器的最大长连接数，一般都是因为看到了文档中的这句话，而漏看了这个"idle"一词。

然后继续看文档后面另外一句话：

When this number is exceeded, the least recently used connections are closed. 当这个数量被突破时，最近使用最少的连接将被关闭。

这句话更是大大强化了前面关于keepalive设置的是最大长连接数的误解：如果连接数超过keepalive的限制，就关闭连接。这不是赤裸裸的最大连接数么？

但是nginx的文档立马给出了指示，否定了最大连接数的可能：

It should be particularly noted that the keepalive directive does not limit the total number of connections to upstream servers that an nginx worker process can open. 特别提醒：keepalive指令不会限制一个nginx worker进程到upstream服务器连接的总数量。

**keepalive参数的理解**

要真正理解keepalive参数的含义，请回到文档中的这句：

The connections parameter sets the maximum number of idle keepalive connections to upstream servers connections参数设置到upstream服务器的空闲keepalive连接的最大数量

请注意**空闲**keepalive连接的最大数量中空闲这个关键字。

为了能让大家理解这个概念，我们先假设一个场景： 有一个HTTP服务，作为upstream服务器接收请求，响应时间为100毫秒。如果要达到10000 QPS的性能，就需要在nginx和upstream服务器之间建立大约1000条HTTP连接。nginx为此建立连接池，然后请求过来时为每个请求分配一个连接，请求结束时回收连接放入连接池中，连接的状态也就更改为idle。

我们再假设这个upstream服务器的keepalive参数设置比较小，比如常见的10.

假设请求和响应是均匀而平稳的，那么这1000条连接应该都是一放回连接池就立即被后续请求申请使用，线程池中的idle线程会非常的少，趋进于零。我们以10毫秒为一个单位，来看连接的情况(注意场景是1000个线程+100毫秒响应时间，每秒有10000个请求完成)：

* 每10毫秒有100个新请求，需要100个连接
* 每10毫秒有100个请求结束，可以释放100个连接
* 如果请求和应答都均匀，则10毫秒内释放的连接刚好够用，不需要新建连接，连接池空闲连接为零

然后再回到现实世界，请求通常不是足够的均匀和平稳，为了简化问题，我们假设应答始终都是平稳的，只是请求不平稳，第一个10毫秒只有50,第二个10毫秒有150：

1. 下一个10毫秒，有100个连接结束请求回收连接到连接池，但是假设此时请求不均匀10毫秒内没有预计的100个请求进来，而是只有50个请求。注意此时连接池回收了100个连接又分配出去50个连接，因此连接池内有50个空闲连接。
2. 然后注意看keepalive=10的设置，这意味着连接池中最多容许保留有10个空闲连接。因此nginx不得不将这50个空闲连接中的40个关闭，只留下10个。
3. 再下一个10个毫秒，有150个请求进来，有100个请求结束任务释放连接。150 - 100 = 50,空缺了50个连接，减掉前面连接池保留的10个空闲连接，nginx不得不新建40个新连接来满足要求。

我们可以看到，在短短的20毫秒内，仅仅因为请求不够均匀，就导致nginx在前10毫秒判断空闲连接过多关闭了40个连接，而后10毫秒又不得不新建40个连接来弥补连接的不足。

再来一次类似的场景，假设请求是均匀的，而应答不再均匀，前10毫秒只有50个请求结束，后10毫秒有150个：

1. 前10毫秒，进来100个请求，结束50个请求，导致连接不够用，nginx为此新建50个连接
2. 后10毫秒，进来100个请求，结束150个请求，导致空闲连接过多，ngixn为此关闭了150-100-10=40个空闲连接

特别提醒的时，第二个应答不均匀的场景实际上是对应第一个请求不均匀的场景：正是因为请求不均匀，所以导致100毫秒之后这些请求的应答必然不均匀。

现实世界中的请求往往和理想状态有巨大差异，请求不均匀，服务器处理请求的时间也不平稳，这理论上的大概1000个连接在反复的回收和再分配的过程中，必然出现两种非常矛盾场景在短时间内反复： 1. 连接不够用，造成新建连接 2. 连接空闲，造成关闭连接。从而使得总连接数出现反复震荡，不断的创建新连接和关闭连接，使得长连接的效果被大大削弱。

造成连接数量反复震荡的一个推手，就是这个keepalive 这个最大空闲连接数。毕竟连接池中的1000个连接在频繁利用时，出现短时间内多余10个空闲连接的概率实在太高。因此为了避免出现上面的连接震荡，必须考虑加大这个参数，比如上面的场景如果将keepalive设置为100或者200,就可以非常有效的缓冲请求和应答不均匀。

**总结**

keepalive 这个参数一定要小心设置，尤其对于QPS比较高的场景，推荐先做一下估算，根据QPS和平均响应时间大体能计算出需要的长连接的数量。比如前面10000 QPS和100毫秒响应时间就可以推算出需要的长连接数量大概是1000. 然后将keepalive设置为这个长连接数量的10%到30%。

比较懒的同学，可以直接设置为keepalive=1000之类的，一般都OK的了。

**location设置**

location中有两个参数需要设置：

http {

server {

location / {

proxy\_http\_version 1.1; // 这两个最好也设置

proxy\_set\_header Connection "";

}

}

}

HTTP协议中对长连接的支持是从1.1版本之后才有的，因此最好通过proxy\_http\_version指令设置为"1.1"，而"Connection" header应该被清理。清理的意思，我的理解，是清理从client过来的http header，因为即使是client和nginx之间是短连接，nginx和upstream之间也是可以开启长连接的。这种情况下必须清理来自client请求中的"Connection" header。